



Politecnico di Torino

IV Facoltà di Ingegneria

Corso di Laurea in Ing. Logistica e della Produzione

Corso di Laurea in Ing. dell'Organizzazione d'Impresa

06AZNEG/06AZNDI - Fondamenti di Informatica

```
#include <stdlib.h>
#include <string.h>
#include <ctype.h>

#define MAXPAROLA 30
#define MAXRIGA 80

int main(int argc, char *argv[])
{
    int freq[MAXPAROLA]; /* vettore di contatori
                           delle frequenze delle lunghezze delle parole */
    char riga[MAXRIGA];
    int i, inizio, lunghezza;
    FILE *f;

    for(i=0; i<MAXPAROLA; i++)
        freq[i]=0;

    if(argc != 2)
    {
        fprintf(stderr, "ERRORE, serve un parametro con il nome del file\n");
        exit(1);
    }
    f = fopen(argv[1], "r");
    if(f==NULL)
    {
        fprintf(stderr, "ERRORE, impossibile aprire il file %s\n", argv[1]);
        exit(1);
    }

    while( fgets(riga, MAXRIGA, f) != NULL )
```



Programmazione in C

Raccolta di lucidi

Anno accademico 2009/2010

Fulvio Corno



Questo materiale è soggetto alla licenza d'uso
Creative Commons Attribuzione-Non commerciale-Non opere derivate 2.5
(vedi <http://creativecommons.org/licenses/by-nc-nd/2.5/it/>)

```
#include <stdio.h>
#include <ctype.h>
#define MAXPARI 30
#define MAXIMA 60

int main(int argc, char *argv[])
{
    int freq[MAXPARI]; /* vettore di conteggio delle frequenze delle parole */
    int i, max, longhezza;
    FILE *fp;

    if(argc != 2)
    {
        fprintf(stderr, "ERRORE: non è possibile aprire il file %s", argv[1]);
        exit(1);
    }
    fp = fopen(argv[1], "r");
    if(fp == NULL)
    {
        fprintf(stderr, "ERRORE: impossibile aprire il file %s", argv[1]);
        exit(1);
    }

    for(i=0; i<MAXPARI; i++)
        freq[i] = 0;
    longhezza = 0;
    max = 0;
```

Programmazione in C

Unità Primo programma in C

Primo programma in C

- ▶ Introduzione al linguaggio C
- ▶ Struttura minima di un file C
- ▶ Sottoinsieme minimale di istruzioni
- ▶ Compilare il primo programma
- ▶ Esercizi proposti
- ▶ Sommario

2

Riferimenti al materiale

» Testi

- Kernighan & Ritchie: capitolo 1
- Cabodi, Quer, Sonza Reorda: capitoli 1, 3
- Dietel & Dietel: capitolo 1

» Dispense

- Scheda: "Primo programma in C"

3

```
#include <sys/types.h>
#include <sys/conf.h>
#define MAXPDATA 30
#define MAXREGS 80

int main(int argc, char *argv[])
{
    int freq[MAXPDATA]; /* vettore di contenuto delle frequenze delle lunghezze delle parole */
    int i, max, length; /* i = indice, length = lunghezza */

    if(argc > 1)
        if((length = open(argv[1], O_RDONLY)) < 0)
            sysfatal("ERRORE, non posso aprire il file %s", argv[1]);
    if(read(length, freq, MAXPDATA) != MAXPDATA)
        sysfatal("ERRORE, impossibile leggere il file %s", argv[1]);
    close(length);
}
```

Primo programma in C

Introduzione al linguaggio C

(Merge 14.3)

Genesi del linguaggio C



Brian Kernighan Dennis Ritchie

- Sviluppato tra il 1969 ed il 1973 presso gli AT&T Bell Laboratories
 - B. Kernighan e D. Ritchie
 - Per uso interno
 - Legato allo sviluppo del sistema operativo Unix
- Nel 1978 viene pubblicato "The Programming Language", prima specifica ufficiale del linguaggio
 - Detto "K&R"



5

Obiettivi del linguaggio

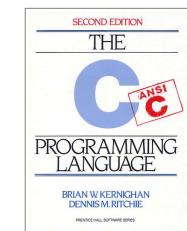
(Merge 14.3)

- Insieme minimale di costrutti base
 - Semplicità del compilatore
- Sintassi estremamente sintetica
 - Talvolta criptica
- Pensato da programmatore per programmatore
 - Elevata efficienza
 - Per nulla user friendly
- Portabile
 - Indipendente dalla macchina
- Disponibilità di una libreria standard di funzioni

6

Evoluzione del linguaggio (1/2)

- 1978, K&R C
- 1989, ANSI C (o C89)
 - Frutto del lavoro di standardizzazione del comitato X3J11 dell'American National Standards Institute
 - Standard X3.159-1989 "Programming Language C"
 - Seconda edizione del K&R



7

Evoluzione del linguaggio (2/2)

(Merge 14.3)

- 1990, ISO C (o C90)
 - Ratifica da parte della International Organization for Standardization dello standard ANSI C
 - ISO/IEC 9899:1990
- 1999, ISO C99
 - Revisione compiuta negli anni '90
 - INCITS-ANSI/ISO/IEC 9899-1999
 - 550 pagine
 - <http://www.open-std.org/jtc1/sc22/wg14/>
 - Supportata da molti (non tutti) i compilatori

8

Diffusione attuale

- I linguaggi attualmente più diffusi al mondo sono:
 - C
 - C++, un'evoluzione del C
 - Java, la cui sintassi è tratta da C++
 - C#, estremamente simile a Java e C++
- Il linguaggio C è uno dei linguaggi più diffusi
- La sintassi del linguaggio C è ripresa da tutti gli altri linguaggi principali

9

Principali vantaggi del C

- Basato su relativamente pochi costrutti da apprendere
- Enorme disponibilità di documentazione ed esempi
- Buona disponibilità di ambienti di sviluppo gratuiti
- Disponibile su qualsiasi configurazione hardware
- Elevata efficienza di elaborazione
- Adatto a vari tipi di applicazioni
 - Programmi di sistema
 - Elaborazione numerica
 - Programmi interattivi

10

Principali svantaggi del C

- Scarsa leggibilità di alcuni costrutti
- Facilità nel commettere errori di programmazione
 - Molti costrutti "pericolosi" sono permessi dal linguaggio e quindi non vengono segnalati dal compilatore
 - Alcuni errori di digitazione possono causare comportamenti errati
- Difficoltà nella realizzazione di interfacce grafiche
- Complessità nell'elaborazione dei testi

11

Un esempio

```
#include <stdio.h>
int main(void)
{
    printf("hello, world\n");
    return 0;
}
```

12

```

#include <stdio.h>
#include <string.h>
#include <ctype.h>

#define MAXPAROLA 30
#define MAXSIGA 80

int main(int argc, char *argv[])
{
    int lungMAXPAROLA; /* valore di costante
    const int lungMAXSIGA = 80; /* valore di costante */
    char digit[MAXSIGA];
    int i, indice, lungMAXSIGA;

    for(i=0; i<MAXSIGA; i++)
        digit[i] = '\0';

    if(argc > 1)
    {
        strcpy(digit, argv[1]);
        lungMAXSIGA = strlen(argv[1]);
    }
    else
    {
        printf("ERRORE, imposta il nome del file\n");
        exit(1);
    }

    while(1)
    {
        if(indice > lungMAXSIGA - 1)
            indice = 0;
        if(digit[indice] == '0')
            break;
        indice++;
    }

    printf("digit[%d]=%c\n", lungMAXSIGA, digit[lungMAXSIGA - 1]);
}

```

Primo programma in C

Struttura minima di un file C

- Applicazioni C in modo “console”
- Struttura del programma
- Commenti
- Direttive #include
- Definizione di variabili
- Corpo del main

2

Struttura minima di un file C

```

#include <stdio.h>
#include <string.h>
#include <ctype.h>

#define MAXPAROLA 30
#define MAXSIGA 80

int main(int argc, char *argv[])
{
    int lungMAXPAROLA; /* valore di costante
    const int lungMAXSIGA = 80; /* valore di costante */
    char digit[MAXSIGA];
    int i, indice, lungMAXSIGA;

    for(i=0; i<MAXSIGA; i++)
        digit[i] = '\0';

    if(argc > 1)
    {
        strcpy(digit, argv[1]);
        lungMAXSIGA = strlen(argv[1]);
    }
    else
    {
        printf("ERRORE, imposta il nome del file\n");
        exit(1);
    }

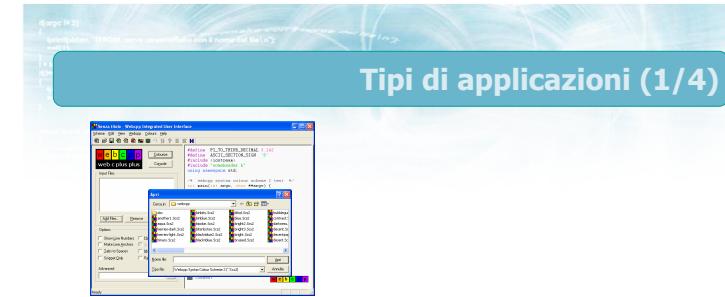
    while(1)
    {
        if(indice > lungMAXSIGA - 1)
            indice = 0;
        if(digit[indice] == '0')
            break;
        indice++;
    }

    printf("digit[%d]=%c\n", lungMAXSIGA, digit[lungMAXSIGA - 1]);
}

```

Struttura minima di un file C

Applicazioni C in modo “console”

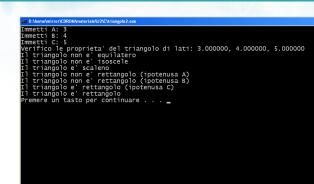
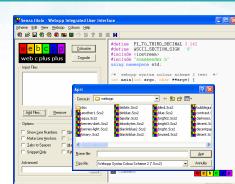


Tipi di applicazioni (1/4)

- Applicazioni grafiche
 - Interazione mediante mouse e finestre
 - Visualizzazione di testi e grafica
 - Elaborazione concorrente

4

Tipi di applicazioni (2/4)



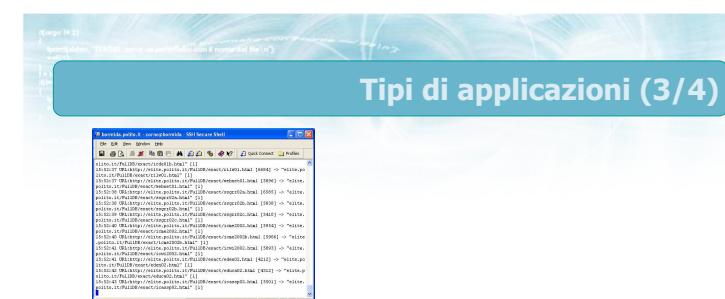
Applicazioni grafiche

- Interazione mediante mouse e finestre
- Visualizzazione di testi e grafica
- Elaborazione concorrente

Applicazioni “console”

- Interazione mediante tastiera
- Visualizzazione di soli caratteri
- Elaborazione sequenziale

5

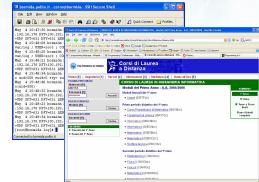
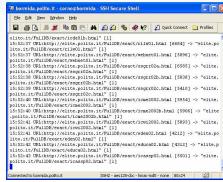


Tipi di applicazioni (3/4)

- Applicazioni batch
 - Nessuna interazione utente
 - Compiti lunghi e ripetitivi
 - Elaborazione numerica, trasferimenti in rete

6

Tipi di applicazioni (4/4)



Applicazioni batch

- Nessuna interazione utente
- Compiti lunghi e ripetitivi
- Elaborazione numerica, trasferimenti in rete

Applicazioni server

- Nessuna interazione utente
- Realizzano funzioni di sistema
- Server locali o server Internet

7

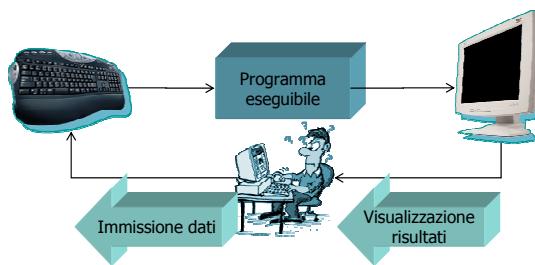
Applicazioni "console"

Interazione utente limitata a due casi

- Stampa di messaggi, informazioni e dati a video
 - Immissione di un dato via tastiera
- ▶ L'insieme tastiera+video viene detto **terminale**
- ▶ Nessuna caratteristica grafica
- ▶ Elaborazione
- Sequenziale
 - Interattiva
 - Mono-utente

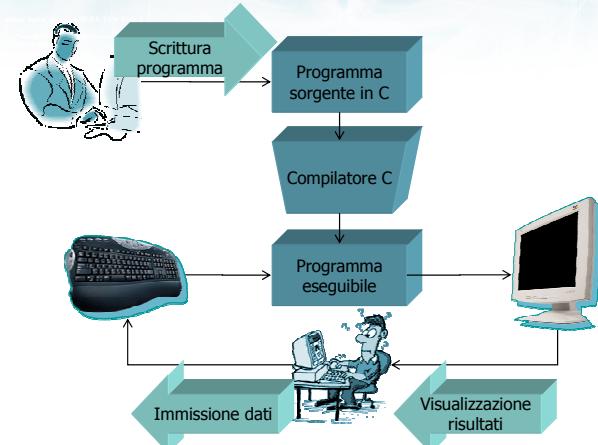
8

Modello di applicazioni "console"



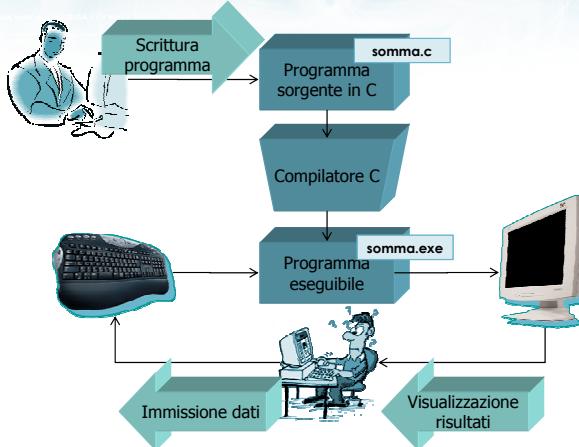
9

Modello di applicazioni "console"



10

Modello di applicazioni "console"



11

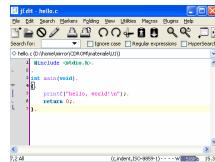
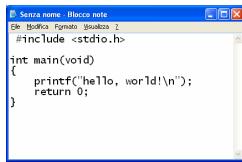
Compilatore C

- ▶ Traduce i programmi **sorgenti** scritti in linguaggio C in programmi **eseguibili**
- ▶ È a sua volta un programma eseguibile, a disposizione del programmatore
- ▶ Controlla l'assenza di **errori di sintassi** del linguaggio
- ▶ Non serve all'utente finale del programma
- ▶ Ne esistono diversi, sia gratuiti che commerciali

12

Scrittura del programma

- Un sorgente C è un normale file di testo
- Si utilizza un editor di testi
 - Blocco Note
 - Editor specializzati per programmatore



13

Editor per programmatore

- Colorazione ed evidenziazione della sintassi
- Indentazione automatica
- Attivazione automatica della compilazione
- Identificazione delle parentesi corrispondenti
- Molti disponibili, sia gratuiti che commerciali

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

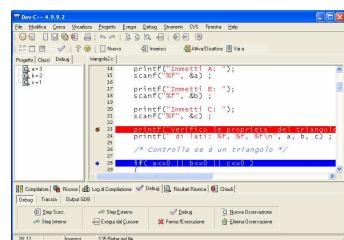
int main()
{
    char MAXPAROLA = 30;
    char MAXIGRA = 80;
    int lungMAXPAROLA;
    int lungMAXIGRA;
    char riga[MAXIGRA];
    int i, j, k, lungMAXIGRA;
    i = 0;
    k = 0;
    while(1)
    {
        if(lung(riga, MAXIGRA) != NULL)
        {
            if(strstr(riga, "STOP") != NULL)
            {
                exit(0);
            }
            else
            {
                if(strstr(riga, "MAXPAROLA") != NULL)
                {
                    lungMAXPAROLA = lung(riga, MAXIGRA);
                    i = 0;
                    j = 0;
                    k = 0;
                    for(j=0; j<lungMAXPAROLA; j++)
                    {
                        if(isalpha(riga[i]))
                        {
                            if(riga[i] == 'A' || riga[i] == 'E' || riga[i] == 'I' || riga[i] == 'O' || riga[i] == 'U')
                            {
                                k++;
                            }
                        }
                    }
                    if(k == 5)
                    {
                        printf("Parola accettata\n");
                    }
                    else
                    {
                        printf("Parola non accettata\n");
                    }
                }
                else
                {
                    if(strstr(riga, "MAXIGRA") != NULL)
                    {
                        lungMAXIGRA = lung(riga, MAXIGRA);
                        i = 0;
                        j = 0;
                        k = 0;
                        for(j=0; j<lungMAXIGRA; j++)
                        {
                            if(isalpha(riga[i]))
                            {
                                k++;
                            }
                        }
                        if(k == lungMAXIGRA)
                        {
                            printf("Stringa accettata\n");
                        }
                        else
                        {
                            printf("Stringa non accettata\n");
                        }
                    }
                }
            }
        }
    }
}
```

Struttura minima di un file C

Struttura del programma

Ambienti integrati

- Applicazioni software integrate che contengono al loro interno
 - Un editor di testi per programmatore
 - Un compilatore C
 - Un ambiente di verifica dei programmi (debugger)
- IDE: Integrated Development Environment



15

Struttura di un sorgente in C

```
#include <stdio.h>

int main(void)
{
    int a ;
    a = 3 ;
    printf("hello, world\n");
    printf("the magic number is %d\n", a) ;
    return 0;
}
```

17

Struttura di un sorgente in C

```
#include <stdio.h>
```

```
int main(void)
{
    int a ;
    a = 3 ;
    printf("hello, world\n");
    printf("the magic number is %d\n", a) ;
    return 0;
}
```

Programma principale
(funzione main)

18

Struttura di un sorgente in C

```
#include <stdio.h>

int main(void)
{
    int a ;
    a = 3 ;
    printf("hello, world\n");
    printf("the magic number is %d\n", a) ;
}

return 0;
}
```

Parentesi graffe che delimitano il main

19

```
#include <stdio.h>
```

```
int main(void)
```

```
{
```

```
    int a ;
```

```
    a = 3 ;
```

```
    printf("hello, world\n");
```

```
    printf("the magic number is %d\n", a) ;
```

```
    return 0;
```

Variabili utilizzate dal programma

20

Struttura di un sorgente in C

```
#include <stdio.h>

int main(void)
{
    int a ;
    Istruzioni eseguibili
    a = 3 ;
    printf("hello, world\n");
    printf("the magic number is %d\n", a) ;
    return 0;
}
```

21

```
#include <stdio.h>
```

```
int main(void)
```

```
{
```

```
    int a ;
```

```
    a = 3 ;
```

```
    printf("hello, world\n");
```

```
    printf("the magic number is %d\n", a) ;
```

```
    return 0;
}
```

Richiamo delle librerie utilizzate

22

In generale

```
#include delle librerie
int main(void)
{
    definizione variabili
    istruzioni eseguibili
}
```

23

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

#define MAXPAROLA 30
#define MAXIGRA 80

int main(int argc, char *argv[])
{
    int largMAXPAROLA; /* valore di costante
    deve frequentemente raggiungere delle grosse */
    char c, AXIGRA;
    int i, indice, lunghezza;
    FILE *f;

    for(i=0; i<MAXPAROLA; i++)
        largMAXPAROLA[i] = '\0';

    if(argc<2)
    {
        printf("Usage: %s <file>\n", argv[0]);
        exit(1);
    }
    else
    {
        f = fopen(argv[1], "r");
        if(f==NULL)
        {
            printf("Error opening file %s\n", argv[1]);
            exit(1);
        }
    }
    while(fgets(indice, MAXIGRA, f)!=NULL)
    {
        //qui va la logica
    }
}
```

Struttura minima di un file C

Commenti

Commenti

- Il testo presente in un sorgente C deve essere analizzato dal compilatore C, quindi deve sottostare a tutte le regole sintattiche del linguaggio
- Per aggiungere annotazioni, commenti, spiegazioni, note, ... si può usare un **commento** all'interno del sorgente

```
/* io sono un commento */
```

25

Sintassi

- Un commento è una qualsiasi sequenza di caratteri (anche su più righe) che:
 - Inizia con la coppia di caratteri `/*`
 - Termina con la coppia di caratteri `*/`
- Non è permesso annidare commenti
 - All'interno di un commento non devono comparire i caratteri `/*`
- Tutto ciò che è compreso tra `/*` e `*/` viene ignorato dal compilatore C

26

Esempio

```
/* programma: hello.c
   autore: fulvio corno
   */

/* accedi alla libreria standard */
#include <stdio.h>

int main(void)
{
    int a ; /* numero magico */
    a = 3 ; /* assegno un valore */

    /* salutiamo l'utente */
    printf("hello, world\n") ;
    printf("the magic number is %d\n", a) ;

    return 0;
}
```

28

Spazi bianchi

- Oltre ai commenti, il compilatore ignora tutti gli spazi bianchi
 - Spazi tra un'istruzione e la successiva
 - Spazi ad inizio linea
 - Spazi intorno alla punteggiatura
 - Righe vuote
- La spaziatura viene utilizzata per rendere il sorgente C più ordinato e più facilmente comprensibile

Esempio

```
/* programma: hello.c autore: fulvio corno */
/* accedi alla libreria standard */
#include <stdio.h>
int main(void)
{ int a ; /* numero magico */ a = 3 ;
  /* assegno un valore */
  /* salutiamo l'utente */ printf("hello, world\n") ;
  printf("the magic number is %d\n", a) ; return 0;
}
```

29

Esempio

```
/* programma: hello.c autore: fulvio corno */
/* accedi alla libreria standard */
#include <stdio.h>
int main(void)
{ int a ; /* numero magico */ a = 3 ;
  /* assegno un valore */
  /* salutiamo l'utente */ printf("hello, world\n") ;
  printf("the magic number is %d\n", a) ; return 0;
}
```

```
#include <stdio.h>
int main(void)
{ int a ; a = 3 ; printf("hello, world\n") ;
  printf("the magic number is %d\n", a) ; return 0; }
```

30

- Ogni compilatore C dispone di diverse librerie di funzioni già pronte per l'uso
- Il programmatore può utilizzare le funzioni di libreria
- È necessario dichiarare a quali librerie si vuole avere accesso
 - Direttive #include ad inizio programma
 - Aggiunge al programma le dichiarazioni di tutte le funzioni di tale libreria, permettendo al programmatore di richiamare tali funzioni

32

Struttura minima di un file C

Direttive #include

```
#include <NomeLibreria.h>
```

➤ Librerie principali:

- **#include <stdio.h>**
 - Funzioni di lettura/scrittura su terminale e su file
- **#include <stdlib.h>**
 - Funzioni base per interazione con sistema operativo
- **#include <math.h>**
 - Funzioni matematiche
- **#include <string.h>**
 - Elaborazione di testi

33

34

Avvertenze

- A differenza della regola generale, nelle direttive **#include** la spaziatura è importante
 - Il carattere **#** deve essere il primo della riga
 - Può esserci una sola **#include** per riga
 - La direttiva **#include** non va terminata con **;**
- Dimenticare una **#include** potrà portare ad errori nel corpo del **main**, quando si chiameranno le funzioni relative

Suggerimenti

➤ Iniziare sempre il sorgente C con le seguenti linee:

```
/* programma: NomeFile.c
autore: NomeAutoreDelProgramma
BreveDescrizioneDelProgramma
*/
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

int main(void)
{
    ...
}
```

35

Definizione di variabili



```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

#define MAXPAROLA 30
#define MAXIGRA 80

int main(int argc, char *argv[])
{
    int freqMAXPAROLA; /* variabile di controllo
    delle frequenze delle lunghezze delle parole */
    char cAXIGRA;
    int i, indice, lunghezza;
    FILE *f;

    f=fopen(argv[1], "r");
    if(f==NULL)
    {
        fprintf(stderr, "ERRORE: impossibile aprire il file %s", argv[1]);
        exit(1);
    }

    for(i=0; i<MAXIGRA; i++)
        freqMAXPAROLA[i]=0;

    while(fscanf(f, "%c", &cAXIGRA)!=EOF)
    {
        if(isalpha(cAXIGRA))
        {
            indice=cAXIGRA-'A';
            lunghezza=indice+1;
            freqMAXPAROLA[lunghezza]++;
        }
    }

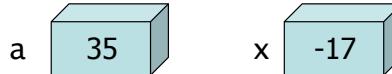
    printf("I file ha %d parole\n", freqMAXPAROLA[0]);
    for(i=1; i<MAXIGRA; i++)
        printf("%d\t%d\n", i, freqMAXPAROLA[i]);
}

exit(0);
```

Struttura minima di un file C

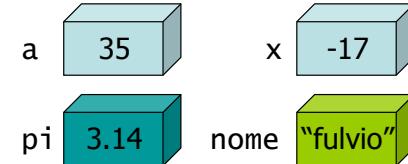
Variabili

- Il programma memorizza le informazioni sulle quali lavora all'interno di **variabili**
- Ogni variabile è caratterizzata da:
 - **Tipo di dato**
 - **Nome**
 - **Valore corrente**



37

- Il programma memorizza le informazioni sulle quali lavora all'interno di **variabili**
- Ogni variabile è caratterizzata da:
 - **Tipo di dato**
 - **Nome**
 - **Valore corrente**



38

Tipo di dato

- Definisce l'insieme dei **valori ammissibili** per la variabile

	Numeri interi, positivi o negativi
	Numeri reali
	Stringhe di testo
	Singoli caratteri di testo

39

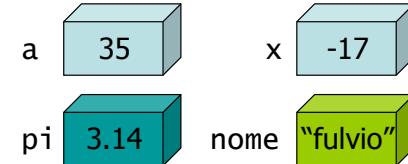
Nome

- Il programmatore assegna un nome a ciascuna variabile
- Dovrebbe rappresentare lo scopo dei valori contenuti nella variabile
- Sintetico, rappresentativo, mnemonico, facile da scrivere

41

Variabili

- Il programma memorizza le informazioni sulle quali lavora all'interno di **variabili**
- Ogni variabile è caratterizzata da:
 - **Tipo di dato**
 - **Nome**
 - **Valore corrente**



38

Tipo di dato

- Definisce l'insieme dei **valori ammissibili** per la variabile

<code>int</code>		Numeri interi, positivi o negativi
<code>float</code>		Numeri reali
		Stringhe di testo
<code>char</code>		Singoli caratteri di testo

40

Nomi ammissibili

- Il **primo** carattere deve essere una **lettera**
- I successivi possono essere **lettere o numeri**
- Lettere maiuscole e minuscole sono **diverse**
- Il simbolo `_` viene considerato come una lettera
- Non devono essere nomi **riservati** dal linguaggio

42

Esempi di nomi

a b a1 a2

a b a1 a2

num n N somma max

43

44

Esempi di nomi

a b a1 a2
num n N somma max
area perimetro perim

a b a1 a2
num n N somma max
area perimetro perim
n_elementi Nelementi risultato

45

46

Esempi di nomi

a b a1 a2
num n N somma max
area perimetro perim
n_elementi Nelementi risultato
trovato nome risposta

47

Definizione di variabili

- » Ogni variabile deve essere **definita prima** di poterla utilizzare
- » Definizioni all'inizio della funzione **main**
- » Sintassi della definizione
 - ***TipoVariabile NomeVariabile ;***

```
int main(void)
{
    int a ;
    int b ;
    float x ;
    . . .
}
```

48

Definizione di variabili

- Ogni variabile deve essere **definita prima** di poterla utilizzare
- Definizioni all'inizio della funzione main
- Sintassi della definizione

- TipoVariabile NomeVariabile ;**
- TipoVariabile NomeVariabile, NomeVariabile ;**

```
int main(void)
{
    int a ;
    int b ;
    float x ;
    . . .
}
```

```
int main(void)
{
    int a, b ;
    float x ;
    . . .
}
```

49

Valore contenuto

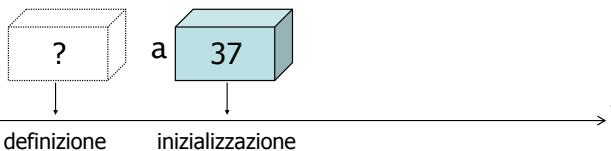
- Ogni variabile, in ogni istante di tempo, possiede un certo valore
- Le variabili appena definite hanno valore ignoto
- Variabili non inizializzate
- In momenti diversi il valore può cambiare



50

Valore contenuto

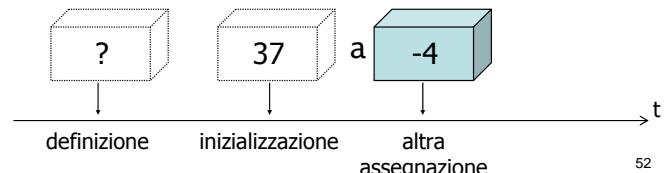
- Ogni variabile, in ogni istante di tempo, possiede un certo valore
- Le variabili appena definite hanno valore ignoto
- Variabili non inizializzate
- In momenti diversi il valore può cambiare



51

Valore contenuto

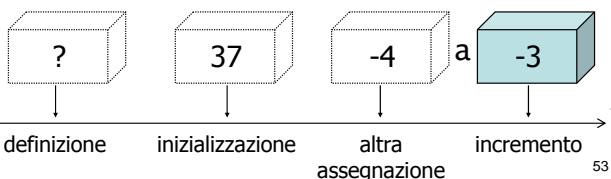
- Ogni variabile, in ogni istante di tempo, possiede un certo valore
- Le variabili appena definite hanno valore ignoto
- Variabili non inizializzate
- In momenti diversi il valore può cambiare



52

Valore contenuto

- Ogni variabile, in ogni istante di tempo, possiede un certo valore
- Le variabili appena definite hanno valore ignoto
- Variabili non inizializzate
- In momenti diversi il valore può cambiare



53

Struttura minima di un file C

Corpo del main

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

#define MAXPAROLA 30
#define MAXIGRA 80

int main(int argc, char *argv[])
{
    int freqMAXPAROLA; /* variabile di controllo delle frequenze delle parole */
    char c; /* carattere AXIGRA */
    int i, indice, lunghezza; /* altre variabili */

    freqMAXPAROLA = 0;

    for(i=0; i<MAXIGRA; i++)
        freqMAXPAROLA++;

    if(argc > 1)
        strcpy(argv[1], "TESTORE");
    else
        strcpy(argv[1], "TESTORE");

    if(freqMAXPAROLA == 0)
        printf("ERRORE: impossibile aprire il file %s", argv[1]);
    else
        printf("Parola %s\n", argv[1]);
}

main() {
    puts("TESTORE");
}
```

Istruzioni eseguibili

- » La funzione `main`, dopo le definizioni di variabili, contiene le vere e proprie **istruzioni eseguibili**
- » Ciascuna istruzione è terminata da `;`
- » Tutte le istruzioni sono comprese nelle `{ ... }`
- » Le istruzioni vengono eseguite in **ordine**
- » Dopo aver eseguito l'ultima istruzione, il programma **termina**

55

Esempio

```
/* programma: hello.c
   autore: fulvio corno
*/
/* accedi alla libreria standard */
#include <stdio.h>

int main(void)
{
    int a ; /* numero magico */
    a = 3 ; /* assegno un valore */

    /* salutiamo l'utente */
    printf("hello, world\n") ;
    printf("the magic number is %d\n", a) ;

    return 0;
}
```

Tipologie di istruzioni

Istruzioni operative

- **Lettura dati**
 - `scanf("%d", &a) ;`
- **Stampa risultati**
 - `printf("%d", a) ;`
- **Elaborazione numerica**
 - `a = b + c ;`
 - `b = b + 1 ;`
 - `c = 42 ;`
 - `c = sqrt(a) ;`

57

Tipologie di istruzioni

Istruzioni operative

- **Lettura dati**
 - `scanf("%d", &a) ;`
- **Stampa risultati**
 - `printf("%d", a) ;`
- **Elaborazione numerica**
 - `a = b + c ;`
 - `b = b + 1 ;`
 - `c = 42 ;`
 - `c = sqrt(a) ;`

Istruzioni di controllo

- **Modificano il controllo di flusso**
 - Scelte
 - Iterazioni
 - Chiamate a funzioni
 - Interruzioni e salti
- **Predefinite dal linguaggio C**
 - `if else while`
 - `for return`
 - `switch case`
 - `break continue`
 - `goto`

58

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

#define MAXPAROLA 30
#define MAXSIGA 80

int main(int argc, char *argv[])
{
    int freqMAXPAROLA; /* valore di controllo delle frequenze delle lunghezze delle parole */
    char freqMAXSIGA; /* valore di controllo delle frequenze delle lunghezze delle parole */
    int i, indice, lunghezza;
    FILE *fp;
    char c;
    int nlinee;
    int nparole;
    int nSIGA;
    float totale;
    float percentuale;
    int maxParola;
    int maxSIGA;
    int maxNlinee;
    int maxNparole;
    int maxNSIGA;
    int maxLunghezza;
    int maxIndice;
    int maxNlineeSIGA;
    int maxNparoleSIGA;
    int maxNSIGA(SIGA);

    if(argc != 2)
    {
        fprintf(stderr, "ERRORE: non è possibile aprire il file %s", argv[1]);
        exit(1);
    }

    fp = fopen(argv[1], "r");
    if(fp == NULL)
    {
        fprintf(stderr, "ERRORE: impossibile aprire il file %s", argv[1]);
        exit(1);
    }

    while(fgets(diga, MAXSIGA, fp) != NULL)
    {
        nlinee++;
        nSIGA++;
        if(strlen(diga) > maxLunghezza)
            maxLunghezza = strlen(diga);
        if(strlen(diga) > maxParola)
            maxParola = strlen(diga);
        if(strlen(diga) > maxSIGA)
            maxSIGA = strlen(diga);
        if(strlen(diga) > maxNlinee)
            maxNlinee = strlen(diga);
        if(strlen(diga) > maxNparole)
            maxNparole = strlen(diga);
        if(strlen(diga) > maxNSIGA(SIGA))
            maxNSIGA(SIGA) = strlen(diga);
        if(strlen(diga) > maxIndice)
            maxIndice = strlen(diga);
    }
}
```

Primo programma in C

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

#define MAXPAROLA 30
#define MAXSIGA 80

int main(int argc, char *argv[])
{
    int freqMAXPAROLA; /* valore di controllo delle frequenze delle lunghezze delle parole */
    char freqMAXSIGA; /* valore di controllo delle frequenze delle lunghezze delle parole */
    int i, indice, lunghezza;
    FILE *fp;
    char c;
    int nlinee;
    int nparole;
    int nSIGA;
    float totale;
    float percentuale;
    int maxParola;
    int maxSIGA;
    int maxNlinee;
    int maxNparole;
    int maxNSIGA;
    int maxLunghezza;
    int maxIndice;
    int maxNlineeSIGA;
    int maxNparoleSIGA;
    int maxNSIGA(SIGA);

    if(argc != 2)
    {
        fprintf(stderr, "ERRORE: non è possibile aprire il file %s", argv[1]);
        exit(1);
    }

    fp = fopen(argv[1], "r");
    if(fp == NULL)
    {
        fprintf(stderr, "ERRORE: impossibile aprire il file %s", argv[1]);
        exit(1);
    }

    while(fgets(diga, MAXSIGA, fp) != NULL)
    {
        nlinee++;
        nSIGA++;
        if(strlen(diga) > maxLunghezza)
            maxLunghezza = strlen(diga);
        if(strlen(diga) > maxParola)
            maxParola = strlen(diga);
        if(strlen(diga) > maxSIGA)
            maxSIGA = strlen(diga);
        if(strlen(diga) > maxNlinee)
            maxNlinee = strlen(diga);
        if(strlen(diga) > maxNparole)
            maxNparole = strlen(diga);
        if(strlen(diga) > maxNSIGA(SIGA))
            maxNSIGA(SIGA) = strlen(diga);
        if(strlen(diga) > maxIndice)
            maxIndice = strlen(diga);
    }
}
```

Sottoinsieme minimale di istruzioni

I tipi int e float

Sottoinsieme minimale di istruzioni

- ▶ I tipi **int** e **float**
- ▶ Istruzione **printf** – semplificata
- ▶ Istruzione **scanf** – semplificata
- ▶ Istruzione di assegnazione
- ▶ Semplici espressioni aritmetiche

2

Tipi di dato

- ▶ Ogni costante, ogni variabile, ogni espressione appartiene ad un determinato **tipo**
- ▶ Il tipo determina
 - L'insieme dei valori che la costante, variabile o espressione può assumere
 - L'insieme delle operazioni lecite su tali valori
- ▶ I tipi possono essere
 - Semplici (o scalari): singoli valori
 - Strutturati: insiemi di più valori semplici

4

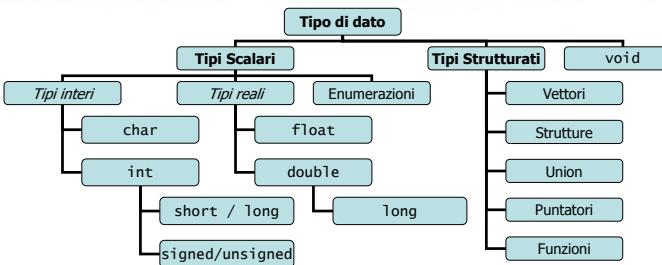
Caratteristiche generali

- ▶ I valori ammessi per ciascun tipo non sono fissati dallo standard
- ▶ Dipendono dal compilatore e dal sistema operativo
 - **Ampiezza dei tipi di dato "naturale" per ogni calcolatore**
- ▶ Maggior portabilità
- ▶ Maggior efficienza
- ▶ Nessuna garanzia di uniformità tra piattaforme diverse

5

6

Il sistema dei tipi C



Il tipo int

- » Il tipo più importante del linguaggio C
- » Valori interi, positivi o negativi
- » Codificato in complemento a due
- » Espresso solitamente su 16 bit oppure 32 bit
 - 16 bit: da -32 768 a +32 767
 - 32 bit: da -2 147 483 648 a +2 147 483 647
 - In generale: da INT_MIN a INT_MAX
- #include <limits.h>

35
int

7

Esempi

```
int i, j ;  
int N ;  
int x ;  
  
i = 0 ;  
j = 2 ;  
N = 100 ;  
x = -3124 ;
```

i 0
j 2
N 100
x -3124

8

Il tipo float

- » Valori reali
 - Frazionari
 - Esterini all'intervallo permesso per i numeri interi
- » Codificato in virgola mobile, singola precisione
- » Espresso solitamente su 32 bit
 - da $\pm 1.17 \times 10^{-38}$ a $\pm 3.40 \times 10^{+38}$
 - circa 6 cifre di precisione
 - In generale: da FLT_MIN a FLT_MAX
- #include <float.h>

3.14
float

9

Esempi

```
float a, b ;  
float pigr ;  
float Nav, Qe ;  
  
a = 3.1 ;  
b = 2.0 ;  
pigr = 3.1415926 ;  
Nav = 6.022e23 ;  
Qe = 1.6e-19 ;
```

a 3.1
b 2.0
pigr 3.1415
Nav 6.02×10^{23}
Qe 1.6×10^{-19}

10



Sottoinsieme minimale di istruzioni

Istruzione printf – semplificata

Istruzioni di stampa

- » Stampa di messaggi informativi
- » Stampa di comando "a capo"
- » Stampa di valori di variabili
- » Stampa di valori di espressioni calcolate
- » Stampa di messaggi contenenti valori



12

Stampa di messaggi

```
printf("Benvenuto\n") ;  
printf("Immetti un numero: ") ;  
printf("\n");
```

Benvenuto

Immetti un numero: _

Stampa di espressioni

```
printf("%d\n", i-j) ;  
printf("%d\n", N*2) ;  
printf("%f\n", Nav * Qe) ;
```

-2

200

96352.000000

Stampa di variabili

```
printf("%d ", j) ;  
printf("%d\n", N) ;  
printf("%f\n", pigr) ;  
printf("%f\n", Nav) ;
```

2 100

3.141593

602200013124147500000000.000000

Sintassi istruzione printf

- » `#include <stdio.h>`
- » `printf("formato ", valore/i) ;`
- » **Formato:**
 - Testo libero (compresi spazi) → viene stampato letteralmente
 - Simboli `\n` → va a capo
 - Simboli `%d` → stampa un `int`
 - Simboli `%f` → stampa un `float`
- » **Valore/i:**
 - Variabile o espressione
 - Di tipo `int` o `float`, corrispondente al simbolo `%`

17

Casi particolari (1/2)

- » Per stampare il simbolo `%` occorre ripeterlo due volte
 - `printf("Sondaggio: %f%%\n", psi) ;`
 - `%f` → stampa `psi`
 - `%%` → stampa un simbolo `%`
 - `\n` → va a capo
- `Sondaggio: 43.12%`

18

Casi particolari (2/2)

- È possibile stampare più di un valore nella stessa istruzione
 - `printf("voti: %d su %d\n", voti, tot) ;`
 - primo %d → stampa voti
 - secondo %d → stampa tot
 - **voti: 18 su 45**

```
#include <stdio.h>
#include <limits.h>
#include <ctype.h>

#define MAXPAROLA 30
#define MSG 60

int main(int argc, char *argv[])
{
    int lunghezzaMAXPAROLA; /* *valore di controllo della frequenza delle lunghezze delle parole */
    int i, lunghezza; /* i = indice, lunghezza = parola */
    char linea[MAXPAROLA]; /* linea da leggere */
    int nlinee; /* numero di righe lette */

    if (argc != 2)
    {
        fprintf(stderr, "ERRORE: non è possibile indicare il nome del file\n");
        exit(1);
    }
    if ((nlinee = fopen(argv[1], "r")) == NULL)
    {
        fprintf(stderr, "ERRORE: impossibile aprire il file %s", argv[1]);
        exit(1);
    }

    /* legge la prima linea */
    if (fscanf(nlinee, "%s", linea) == 1) /* se nulla... */
    {
        /* ... */
    }
}
```

Sottoinsieme minimale di istruzioni

Istruzione scanf – semplificata

19

Istruzioni di lettura

- Lettura di un valore intero
- Lettura di un valore reale



21

Lettura di un intero

`scanf("%d", &N) ;`

213

Lettura di un reale

`scanf("%F", &a) ;`

12.5

Sintassi istruzione scanf

- `#include <stdio.h>`
- `scanf("formato ", &variabile) ;`
- Formato:
 - Simboli %d → legge un int
 - Simboli %f → legge un float
- Variabile:
 - Di tipo int o float, corrispondente al simbolo %
 - Sempre preceduta dal simbolo &

24



Suggerimento

- Combinare le istruzioni `printf` e `scanf` per guidare l'utente nell'immissione dei dati
 - Ogni `scanf` deve essere preceduta da una `printf` che indica quale dato il programma si aspetta

```
printf("Immetti il numero: ");
scanf("%d", &N);
printf("Numero immesso: %d\n", N);
```

25

Errore frequente

- Dimenticare il simbolo & nelle istruzioni `scanf`

```
printf("Immetti il numero: ");
scanf("%d", N);
```

forma corretta

```
printf("Immetti il numero: ");
scanf("%d", &N);
```

26



Errore frequente

- Dimenticare le variabili da stampare nelle istruzioni `printf`

```
printf("Numero immesso: %d\n");
```

forma corretta

```
printf("Numero immesso: %d\n", N);
```

27

#include <stdio.h>
#include <string.h>
#include <ctype.h>

#define MAXPAROLA 30
#define MAXIGRA 80

```
int main(int argc, char *argv[])
{
    int lungMAXPAROLA; // variezza di parola
    char maxParola[MAXPAROLA]; // stringa che contiene la parola più lunga
    char lungMAXIGRA; // variezza di riga
    char riga[MAXIGRA];
    int i, j, k, nriga;
    FILE *f;

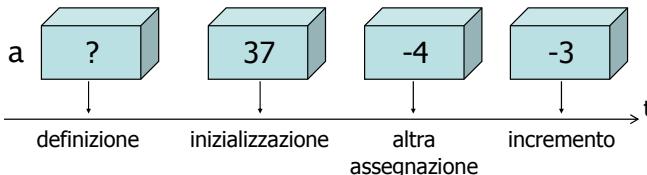
    f=fopen(argv[1], "r");
    if(f==NULL)
        printf("ERRORE, non è possibile aprire il file %s", argv[1]);
    else
        fgets(maxParola, MAXPAROLA, f);
        nriga=1;
        while(fgets(riga, MAXIGRA, f)) {
            if(strcmp(riga, maxParola) > 0) {
                maxParola[0]=riga[0];
                lungMAXIGRA=lungMAXIGRA+1;
            }
            else
                nriga++;
        }
    printf("lungMAXIGRA=%d\n", lungMAXIGRA);
    printf("nriga=%d\n", nriga);
}
```

Sottoinsieme minimaile di istruzioni

Istruzione di assegnazione

Assegnazione delle variabili

- Il valore di una variabile
 - Deve essere inizializzato, la prima volta
 - Può essere aggiornato, quante volte si vuole
- Per assegnare un nuovo valore ad una variabile si usa l'operatore =



29

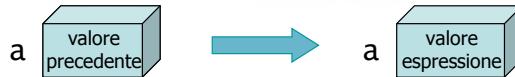
Assegnazione di variabili

- Assegnazione del valore di una costante
 - `i = 0 ;`
 - `a = 3.0 ;`
- Assegnazione del valore di un'altra variabile
 - `i = N ;`
 - `b = a ;`
- Assegnazione del valore di un'espressione
 - `j = N - i ;`
 - `b = a * 2 - 1 ;`

30

Sintassi dell'assegnazione

» $\text{variabile} = \text{espressione};$



» **Passo 1:** si valuta il valore corrente dell'espressione

- Per tutte le variabili che compaiono nell'espressione, si usa il valore corrente
- Può comparire anche la stessa variabile oggetto dell'assegnazione

» **Passo 2:** si memorizza tale valore nella variabile, cancellando il valore precedente

31

Esempi

» $N = 3;$

32

Esempi

» $N = 3;$

» $a = b;$

- Non confondere con $b = a;$

33

34

Esempi

» $N = 3;$

» $a = b;$

- Non confondere con $b = a;$

» $a = a + 1;$

- Incrementa a di un'unità

» $a + 1 = a;$

- Errore

35

36

Quesito

» Che operazione svolge il seguente frammento di programma?

$a = b;$
 $b = a;$

Risposta

- Che operazione svolge il seguente frammento di programma?

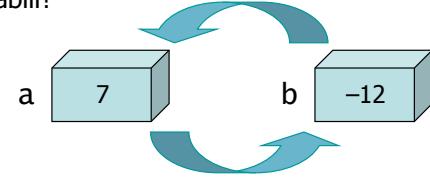
```
a = b ;
b = a ;
```

- Il valore corrente di b viene copiato in a
 ● Il valore vecchio di a viene perso
 ➤ Il (nuovo) valore corrente di a (uguale a b) viene ricopiato in b (operazione inutile)

37

Quesito

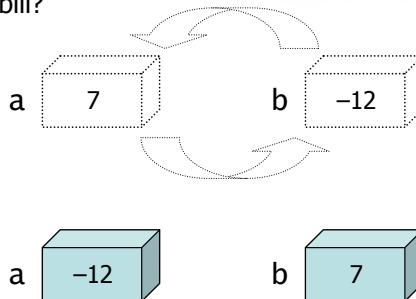
- Come fare a scambiare tra di loro i valori di due variabili?



38

Quesito

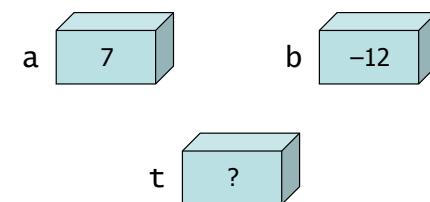
- Come fare a scambiare tra di loro i valori di due variabili?



39

Risposta

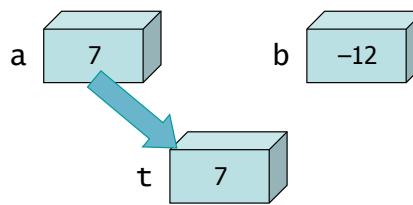
```
t = a ;
a = b ;
b = t ;
```



40

Risposta

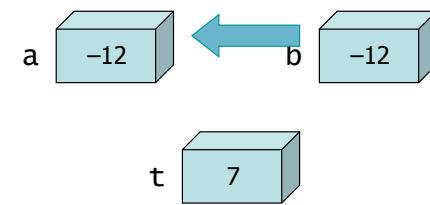
```
t = a ;
a = b ;
b = t ;
```



41

Risposta

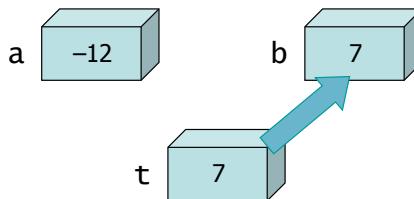
```
t = a ;
a = b ;
b = t ;
```



42

Risposta

```
t = a ;
a = b ;
b = t ;
```



43

Sottoinsieme minimale di istruzioni

Semplici espressioni aritmetiche

Espressioni aritmetiche

- » Ovunque sia richiesto il valore di una variabile, è possibile usare un'espressione aritmetica
 - Nei valori da stampare con la funzione `printf`
 - Nei valori da assegnare ad una variabile
- » Le espressioni si possono costruire ricorrendo a:
 - Operatori: + - * /
 - Parentesi: (...)
 - Funzioni di libreria: `sqrt()`, `sin()`, `cos()`, ...

45

Operatori principali

- » Somma: `a+b`
- » Sottrazione: `a-b`
- » Moltiplicazione: `a*b`
- » Divisione: `a/b`
 - Divisione intera (risultato troncato) se entrambi gli operandi sono `int`
- » Resto della divisione: `a%b`
 - Solo tra operandi `int`
- » Cambio di segno: `-a`

46

Alcuni operatori avanzati

- » Incremento: `i++`
- » Decremento: `N--`
- » Conversione ad intero: `(int)a`
- » Conversione a reale: `(float)N`

47

Funzioni di libreria

- » `#include <math.h>`
- » Funzioni algebriche
 - `fabs, sqrt, cbrt, pow, hypot, ceil, floor, round, trunc, fmod`
- » Funzioni esponenziali e logaritmiche
 - `exp, exp2, log, log10, log2`
- » Funzioni trigonometriche e iperboliche
 - `cos, sin, tan, cosh, sinh, tanh`
- » Funzioni trigonometriche e iperboliche inverse
 - `acos, asin, atan, atan2, acosh, asinh, atanh`

48

- » Si possono costruire espressioni complicate a piacere utilizzando le parentesi
- » Per maggiore leggibilità, abbondare con le parentesi ed usare la spaziatura e l'incollonamento in modo ordinato
- » Si utilizzano sempre le parentesi tonde

```
x1 = ( -b + sqrt( b*b - 4*a*c ) ) /  
      ( 2*a ) ;
```

```
A = sqrt( p * (p-a) * (p-b) * (p-c)) ;
```

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

#define MAXPAROLA 30
#define MAXSIGA 80

int main(int argc, char *argv[])
{
    int lung(MAXPAROLA); /* valore di controllo delle frequenze delle lunghezze delle parole */
    int i, indice, lunghezza;
    FILE *fptr;
    char digit[MAXSIGA];
    char doppMAXSIGA;
    int nlinee;
    int c;

    if(argc != 2)
    {
        fprintf(stderr, "ERRORE: non è possibile aprire il file %s", argv[1]);
        exit(1);
    }
    else
    {
        fptr = fopen(argv[1], "r");
        if(fptr == NULL)
        {
            fprintf(stderr, "ERRORE: impossibile aprire il file %s", argv[1]);
            exit(1);
        }
    }

    while(fgets(digit, MAXSIGA, fptr) != NULL)
    {
        lunghezza = strlen(digit);
        if(lung < lunghezza)
            lung = lunghezza;
        for(i=0; i<lung; i++)
            digit[i] = toupper(digit[i]);
        if(digit[lung-1] == '\n')
            digit[lung-1] = '\0';
        else
            digit[lung] = '\0';

        if(isalpha(digit[0]))
        {
            indice = digit[0] - 'A';
            if(indice > 25)
                indice = 0;
            if(indice >= 0 & indice < lung)
                nlinee[indice]++;
        }
        else
        {
            if(indice >= 0 & indice < lung)
                nlinee[26]++;
        }
    }

    if(fptr != NULL)
        fclose(fptr);
}
```

Primo programma in C

Compilare il primo programma

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

#define MAXPAROLA 30
#define MAXSIGA 80

int main(int argc, char *argv[])
{
    int lung(MAXPAROLA); /* valore di controllo delle frequenze delle lunghezze delle parole */
    int i, indice, lunghezza;
    FILE *fptr;
    char digit[MAXSIGA];
    char doppMAXSIGA;
    int nlinee;
    int c;

    if(argc != 2)
    {
        fprintf(stderr, "ERRORE: non è possibile aprire il file %s", argv[1]);
        exit(1);
    }
    else
    {
        fptr = fopen(argv[1], "r");
        if(fptr == NULL)
        {
            fprintf(stderr, "ERRORE: impossibile aprire il file %s", argv[1]);
            exit(1);
        }
    }

    while(fgets(digit, MAXSIGA, fptr) != NULL)
    {
        lunghezza = strlen(digit);
        if(lung < lunghezza)
            lung = lunghezza;
        for(i=0; i<lung; i++)
            digit[i] = toupper(digit[i]);
        if(digit[lung-1] == '\n')
            digit[lung-1] = '\0';
        else
            digit[lung] = '\0';

        if(isalpha(digit[0]))
        {
            indice = digit[0] - 'A';
            if(indice > 25)
                indice = 0;
            if(indice >= 0 & indice < lung)
                nlinee[indice]++;
        }
        else
        {
            if(indice >= 0 & indice < lung)
                nlinee[26]++;
        }
    }

    if(fptr != NULL)
        fclose(fptr);
}
```

Compilare il primo programma

Un semplice programma

(Argomenti)

Scrivere un programma che somma due numeri.

Compilare il primo programma

- ▶ Un semplice programma
- ▶ L'ambiente di sviluppo Dev-C++
- ▶ Codifica del programma
- ▶ Compilazione e correzione errori
- ▶ Esecuzione e verifica

2

```
(Argomenti)
```

Somma due numeri

Immetti il primo numero: _

Prompt dei comandi

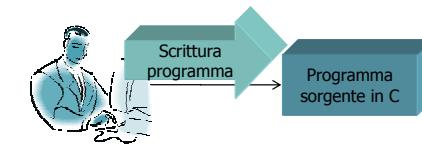
5

(Argomenti)

Scrivere un programma che somma due numeri.

Esercizio “Somma due numeri”

- ▶ Si realizzi un programma in linguaggio C che acquisisca da tastiera due numeri interi (detti A e B) e che stampi a video il valore della somma di tali numeri



4

```
(Argomenti)
```

Somma due numeri

Immetti il primo numero: **18**

Immetti il secondo numero: _

Prompt dei comandi

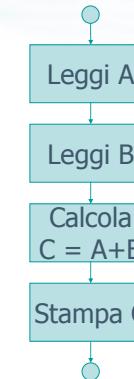
6

Analisi

```
ex: Prompt dei comandi
Somma due numeri
Immetti il primo numero: 18
Immetti il secondo numero: 3
La somma di 18 + 3 vale: 21
```

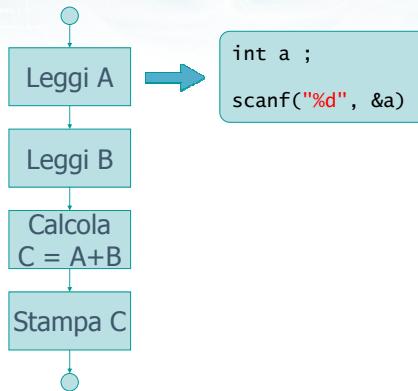
7

Diagramma di flusso



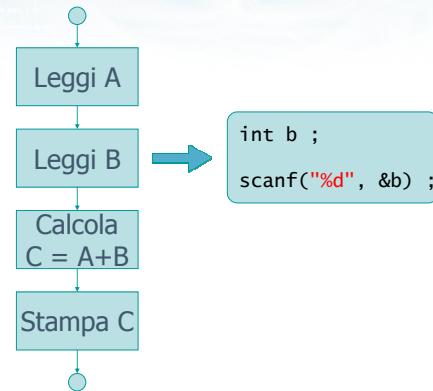
8

Traduzione in C (1/4)



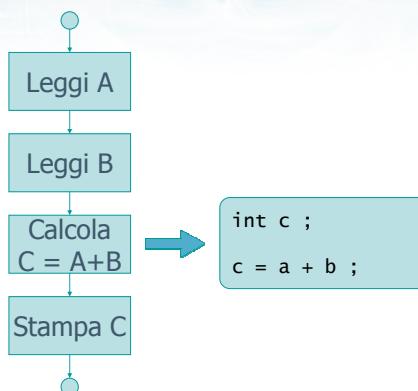
9

Traduzione in C (2/4)



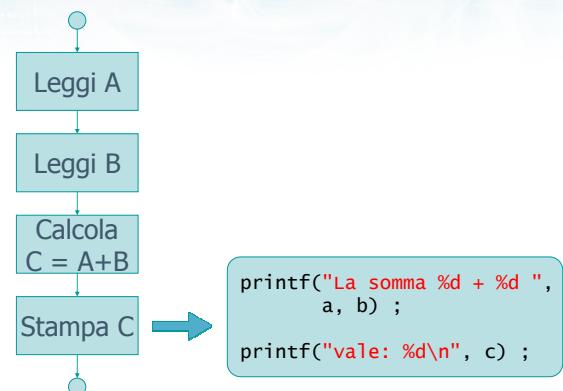
10

Traduzione in C (3/4)



11

Traduzione in C (4/4)



12

```
#include <stdio.h>
#include <ctype.h>

#define MAXFAROLA 30
#define MAXCIA 80

int main(int argc, char *argv[])
{
    int freqMAXFAROLA; /* valori di confronto delle frequenze delle parole */
    int i, indice, lunghezza;
    int lettera;
    FILE *fp;
    fp=fopen(argv[1], "r");
    if (fp==NULL)
        exit(1);
    fscanf(fp, "%d", &freqMAXFAROLA);
    for (i=0; i<freqMAXFAROLA; i++)
        fscanf(fp, "%s", argv[i+1]);
    fclose(fp);
}

int main()
{
    int a, b, c;
    int a1, b1, c1;
    int somma;
    printf("Inserisci il primo numero: ");
    scanf("%d", &a);
    printf("Inserisci il secondo numero: ");
    scanf("%d", &b);
    c = a + b;
    printf("La somma di %d + %d vale: %d\n", a, b, c);
    system("pause"); // tieni aperta la finestra
}
```

Compilare il primo programma

L'ambiente di sviluppo Dev-C++

14

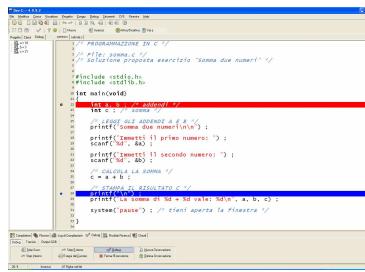
- Occorre identificare ed installare
 - Un editor (possibilmente per programmatore)
 - Un compilatore
 - Un debugger
- Oppure trovare un Integrated Development Environment che integri tutte le funzionalità precedenti
- Esistono molte soluzioni gratuite



IDE per C in ambiente Windows

➤ Dev-C++

- <http://www.bloodshed.net>



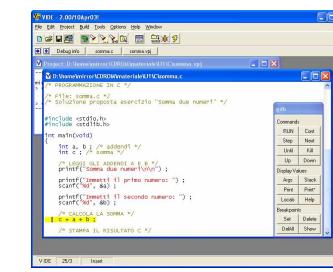
15



IDE per C in ambiente Windows

➤ V IDE

- <http://www.objectcentral.org>



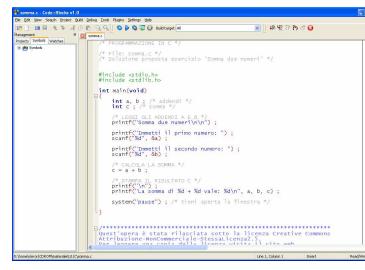
16



IDE per C in ambiente Windows

➤ Code::Blocks

- <http://www.codeblocks.org>



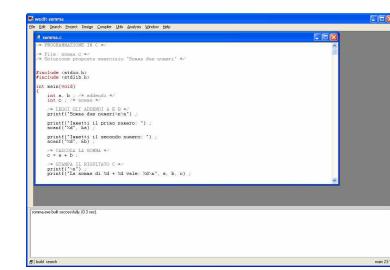
17



IDE per C in ambiente Windows

➤ lcc-win32

- <http://www.cs.virginia.edu/~lcc-win32/>



18

Interfaccia di Dev-C++

```


1 // PROGRAMMAZIONE IN C //
2 // File: somma.c //
3 // soluzione proposta esercizio "Somma due numeri" //
4
5 #include <stdio.h>
6 #include <stdlib.h>
7
8 int main(void)
9 {
10     int a, b; /* addendi */
11     int c; /* somma */
12
13     /* LEGGI GLI ADDENDI A E B */
14     printf("Somma due numeri!\n");
15
16     printf("Inserisci il primo numero: ");
17     scanf("%d", &a);
18
19     printf("Inserisci il secondo numero: ");
20     scanf("%d", &b);
21
22     /* CALCOLA LA SOMMA */
23     c = a + b;
24
25     /* STAMPA IL RISULTATO C */
26     printf("La somma di %d + %d vale: %d\n", a, b, c);
27
28     system("pause"); /* tieni aperta la finestra */
29 }


```

Interfaccia di Dev-C++

Menù e toolbar

Editor programma sorgente

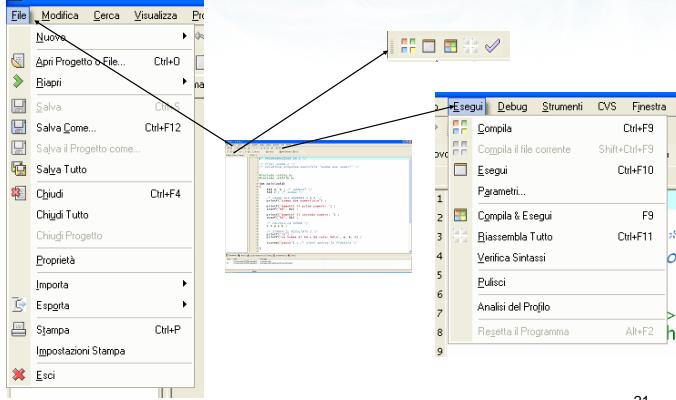
Messaggi errore

```


1 // PROGRAMMAZIONE IN C //
2 // File: somma.c //
3 // soluzione proposta esercizio "Somma due numeri" //
4
5 #include <stdio.h>
6 #include <stdlib.h>
7
8 int main(void)
9 {
10     int a, b; /* addendi */
11     int c; /* somma */
12
13     /* LEGGI GLI ADDENDI A E B */
14     printf("Somma due numeri!\n");
15
16     printf("Inserisci il primo numero: ");
17     scanf("%d", &a);
18
19     printf("Inserisci il secondo numero: ");
20     scanf("%d", &b);
21
22     /* CALCOLA LA SOMMA */
23     c = a + b;
24
25     /* STAMPA IL RISULTATO C */
26     printf("La somma di %d + %d vale: %d\n", a, b, c);
27
28     system("pause"); /* tieni aperta la finestra */
29 }


```

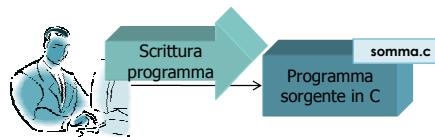
Menu principali



21

Codifica del programma

- A partire dal diagramma di flusso
- Utilizziamo un editor per immettere le istruzioni C
- Creiamo un file sorgente somma.c



23

```


#include <stdio.h>
#include <math.h>
#include <ctype.h>

#define MAXFACOLIA 80
#define MAXIGRA 80

int main(int argc, char *argv[])
{
    int lungMAXFACOLIA; /* variezza massima della stringa */
    int lungMAXIGRA; /* variezza massima della stringa */
    int i, j, k, lungMAXIGRA;
    char riga[MAXIGRA];
    FILE *f;

    if(argc != 2)
    {
        printf("Usage: %s <file>\n", argv[0]);
        exit(1);
    }

    f = fopen(argv[1], "r");
    if(f == NULL)
    {
        perror("Error, impossibile aprire il file %s", argv[1]);
        exit(1);
    }

    while(fgets(riga, MAXIGRA, f) != NULL)
    {
        for(j=0; riga[j] != '\0'; j++)
        {
            if(isdigit(riga[j]))
            {
                for(k=j; riga[k] != '\0'; k++)
                {
                    if(isdigit(riga[k]))
                    {
                        lungMAXFACOLIA++;
                    }
                    else
                    {
                        lungMAXIGRA++;
                    }
                }
            }
        }
    }

    printf("Lunghezza massima della stringa %s: %d\n", argv[1], lungMAXFACOLIA);
    printf("Lunghezza massima della stringa %s: %d\n", argv[1], lungMAXIGRA);
}


```

Compilare il primo programma

Codifica del programma

```


1 // PROGRAMMAZIONE IN C //
2 // File: somma.c //
3 // soluzione proposta esercizio "Somma due numeri" //
4
5 #include <stdio.h>
6 #include <stdlib.h>
7
8 int main(void)
9 {
10     int a, b; /* addendi */
11     int c; /* somma */
12
13     /* LEGGI GLI ADDENDI A E B */
14     printf("Somma due numeri!\n");
15
16     printf("Inserisci il primo numero: ");
17     scanf("%d", &a);
18
19     printf("Inserisci il secondo numero: ");
20     scanf("%d", &b);
21
22     /* CALCOLA LA SOMMA */
23     c = a + b;
24
25     /* STAMPA IL RISULTATO C */
26     printf("La somma di %d + %d vale: %d\n", a, b, c);
27
28     system("pause"); /* tieni aperta la finestra */
29 }


```

- Codifichiamo il programma in Dev-C++

somma.c

```


1 // PROGRAMMAZIONE IN C //
2 // File: somma.c //
3 // soluzione proposta esercizio "Somma due numeri" //
4
5 #include <stdio.h>
6 #include <stdlib.h>
7
8 int main(void)
9 {
10     int a, b; /* addendi */
11     int c; /* somma */
12
13     /* LEGGI GLI ADDENDI A E B */
14     printf("Somma due numeri!\n");
15
16     printf("Inserisci il primo numero: ");
17     scanf("%d", &a);
18
19     printf("Inserisci il secondo numero: ");
20     scanf("%d", &b);
21
22     /* CALCOLA LA SOMMA */
23     c = a + b;
24
25     /* STAMPA IL RISULTATO C */
26     printf("La somma di %d + %d vale: %d\n", a, b, c);
27
28     system("pause"); /* tieni aperta la finestra */
29 }


```

24

Soluzione proposta (1/2)

```
#include <stdio.h>
#include <stdlib.h>

int main(void)
{
    int a, b ; /* addendi */
    int c ; /* somma */

    /* LEGGI GLI ADDENDI A E B */
    printf("Somma due numeri\n\n") ;

    printf("Immetti il primo numero: ") ;
    scanf("%d", &a) ;
    printf("Immetti il secondo numero: ") ;
    scanf("%d", &b) ;
}
```

25

Soluzione proposta (2/2)

```
/* CALCOLA LA SOMMA */
c = a + b ;

/* STAMPA IL RISULTATO C */
printf("La somma di %d + %d vale: %d\n",
a, b, c) ;
}
```

26

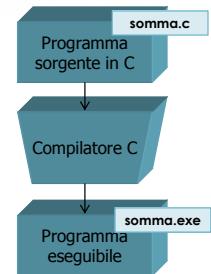


Compilare il primo programma

Compilazione e correzione errori

Compilazione del programma

- Attivare il compilatore sul programma sorgente `somma.c`
- Il compilatore verifica che non ci siano **errori di sintassi**
- In assenza di errori, viene generato il programma eseguibile `somma.exe`



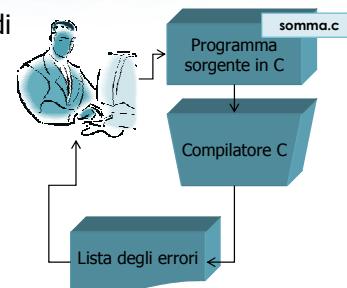
28

Correzione errori di sintassi

Compilazione "Somma due numeri"

- Compiliamo il programma

- Il compilatore genera una lista di messaggi di errore
- Capire il messaggio
- Identificare il punto errato nel programma
- Trovare la soluzione
- Correggere il programma
- Generare una nuova versione del file sorgente



29

```
Compilazione Somma due numeri
$ gcc somma.c
$ ./somma
Somma due numeri
Immetti il primo numero: 5
Immetti il secondo numero: 10
La somma di 5 + 10 vale: 15
$
```

30

```

#include <stdio.h>
#include <ctype.h>
#define MAXPARI 30
#define MAXIMA 80

int main(int argc, char *argv[])
{
    int freqMAXPARI[] /* vettore di contenuto delle frequenze delle lunghezze delle parole */;
    int i, min, lunghezza;

    if(argc != 2)
    {
        fprintf(stderr, "ERRORE: non è possibile aprire il file %s", argv[1]);
        exit(1);
    }

    if(freqMAXPARI == NULL)
    {
        fprintf(stderr, "ERRORE: impossibile aprire il file %s", argv[1]);
        exit(1);
    }

    min = freqMAXPARI[0];
    lunghezza = freqMAXPARI[0];
    for(i=1; freqMAXPARI[i] != 0; i++)
    {
        if(freqMAXPARI[i] < min)
            min = freqMAXPARI[i];
        if(freqMAXPARI[i] > lunghezza)
            lunghezza = freqMAXPARI[i];
    }

    printf("Parola più lunga: %s\n", freqMAXPARI[min]);
    return 0;
}

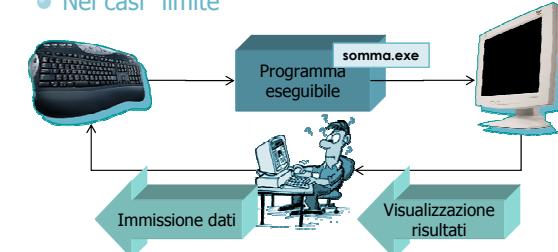
```

Compilare il primo programma

Esecuzione e verifica

Verifica del programma

- Ci mettiamo nei panni dell'utente finale
- Eseguiamo il programma
- Verifichiamo che funzioni correttamente
 - Nei casi "normali"
 - Nei casi "limite"



32

Errori in esecuzione

Tipologie di errori possibili:

- Crash del programma
 - Blocco imposto dal sistema operativo
- Blocco del programma
 - Ciclo "infinito"
- Risultati errati
 - (Quasi) sempre
 - Solo in alcuni casi (con alcuni dati ma non con altri)

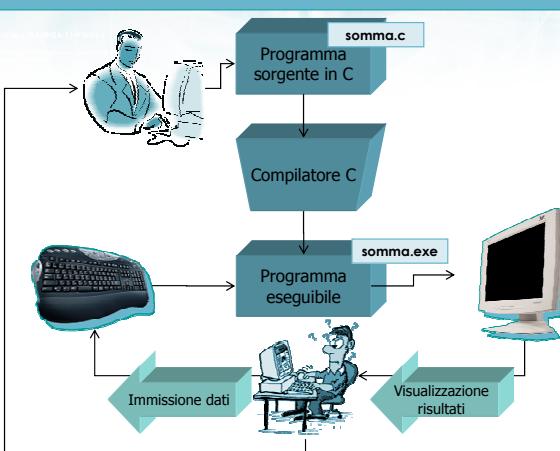
33

Correzione errori di esecuzione

- Lavoro da "detective"
- Risalire dai sintomi alle cause del malfunzionamento
- Formulare delle ipotesi sulla causa dell'errore e verificarle
- Una volta trovato l'errore, cercare una soluzione
- A seconda della gravità, occorrerà modificare
 - Il sorgente C
 - L'algoritmo risolutivo
 - L'approccio generale

34

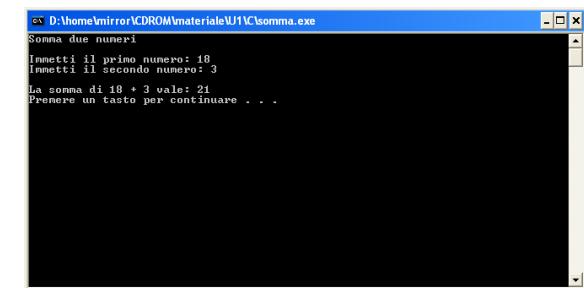
Correzione errori di esecuzione



35

Verifica "Somma due numeri"

- Eseguiamo il programma con alcuni dati di prova, verificandone il comportamento corretto



36

- ▶ Esercizio "Equazione di primo grado"
- ▶ Esercizio "Calcolo di aree"
- ▶ Esercizio "Somma minuti"

Primo programma in C

Esercizi proposti

2

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>
```

```
#define MAXPAROLA 30
#define MAXSIGA 80
```

```
int main(int argc, char *argv[])
{
    int lungMAXPAROLA; /* valore di controllo delle frequenze delle lunghezze delle parole */
    char lungMAXSIGA; /* valore di controllo delle lunghezze delle sigle */
    int i;
}
```

```
for(i=0; i<MAXPAROLA; i++)
    lungi[i] = 0;
```

```
{ for(i=0; i<MAXSIGA; i++)
    lungi[i] = 0;
    i=MAXSIGA;
    lungi[i] = 1;
}
lungi[0] = 1;
lungi[1] = 1;
lungi[2] = 1;
lungi[3] = 1;
lungi[4] = 1;
lungi[5] = 1;
lungi[6] = 1;
lungi[7] = 1;
lungi[8] = 1;
lungi[9] = 1;
lungi[10] = 1;
lungi[11] = 1;
lungi[12] = 1;
lungi[13] = 1;
lungi[14] = 1;
lungi[15] = 1;
lungi[16] = 1;
lungi[17] = 1;
lungi[18] = 1;
lungi[19] = 1;
lungi[20] = 1;
lungi[21] = 1;
lungi[22] = 1;
lungi[23] = 1;
lungi[24] = 1;
lungi[25] = 1;
lungi[26] = 1;
lungi[27] = 1;
lungi[28] = 1;
lungi[29] = 1;
}
```

Esercizi proposti

Esercizio "Equazione di primo grado"

4

Esercizio "Equazione di primo grado"

- ▶ Data l'equazione

$$\bullet \quad ax + b = 0$$

con a e b inseriti da tastiera, determinare il valore di x che risolve l'equazione

```
00 Prompt dei comandi
```

EQUAZIONE DI PRIMO GRADO
 $a x + b = 0$

Inserisci il valore di a: 2.5
 Inserisci il valore di b: 3.2

La soluzione dell'equazione e':
 $x = -1.280000$

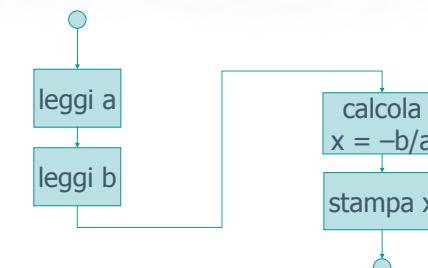
Analisi

5

Soluzione



[primogradoc](#)



6

```

#include <stdio.h>
#include <ctype.h>

#define MAXPARIOLA 30
#define MAXRGA 60

int main(int argc, char *argv[])
{
    int freqMAXPARIOLA; /* valori di confronto delle frequenze delle parola */
    int freqMAXRGA; /* valori di confronto delle frequenze delle parole */

    float lato; /* lato del quadrato */
    float diametro; /* diametro del cerchio */
    float latoEquilatero; /* lato del triangolo equilatero */

    if(argc != 2)
    {
        fprintf(stderr, "Usage: %s <file>\n", argv[0]);
        exit(1);
    }

    lato = freqMAXPARIOLA;
    diametro = freqMAXRGA;
    latoEquilatero = 0.0;

    if(argv[1])
    {
        FILE *fp;
        fp = fopen(argv[1], "r");
        if(fp == NULL)
        {
            perror("Error opening file");
            exit(1);
        }
        fscanf(fp, "%f", &lato);
        fscanf(fp, "%f", &diametro);
        fscanf(fp, "%f", &latoEquilatero);
        fclose(fp);
    }

    printf("Quadrato di lato %f\n", lato);
    printf("Cerchio di diametro %f\n", diametro);
    printf("Triangolo eq. di lato %f\n", latoEquilatero);

    return 0;
}

```

Esercizi proposti

Esercizio “Calcolo di aree”

- Si scriva un programma in linguaggio C che, dato un numero reale immesso da tastiera, detto D, calcoli e stampi:

- L’area del quadrato di lato D
- L’area del cerchio di diametro D
- L’area del triangolo equilatero di lato D

Esercizio “Calcolo di aree”

8

Analisi

Prompt dei comandi

```

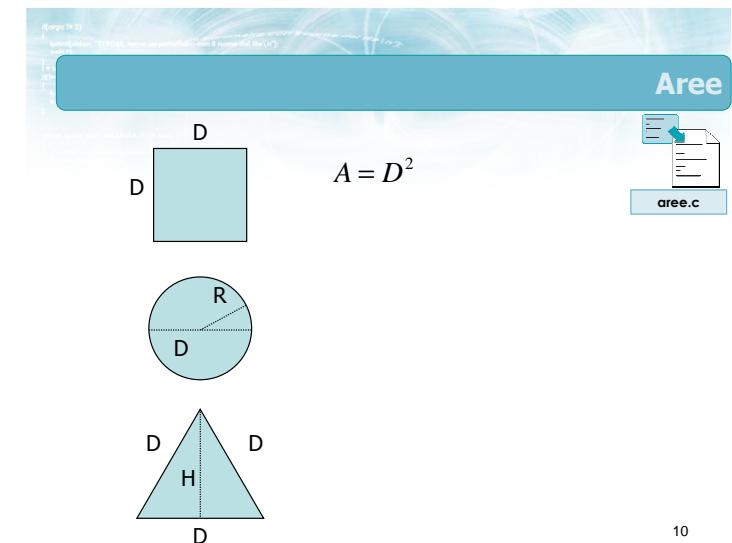
CALCOLO DI AREE

Immetti il valore di D: 2

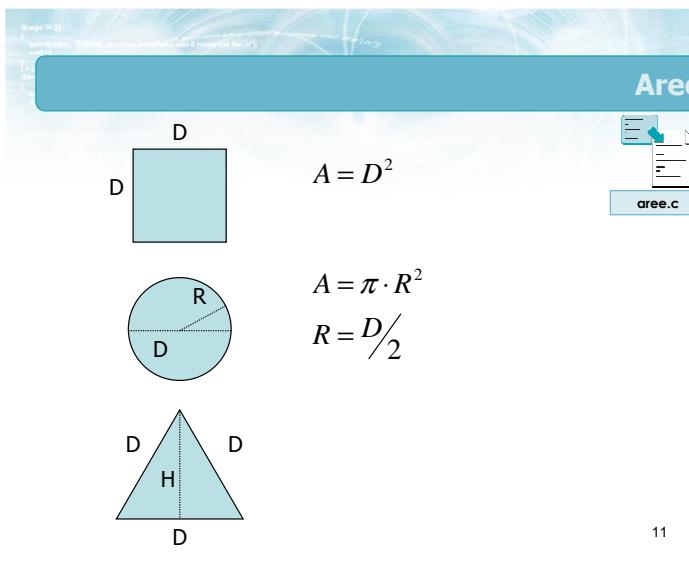
Le aree calcolate sono:
Quadrato di lato 2.000000 = 4.000000
Cerchio di diametro 2.000000 = 3.140000
Triangolo eq. di lato 2.000000 = 1.732051

```

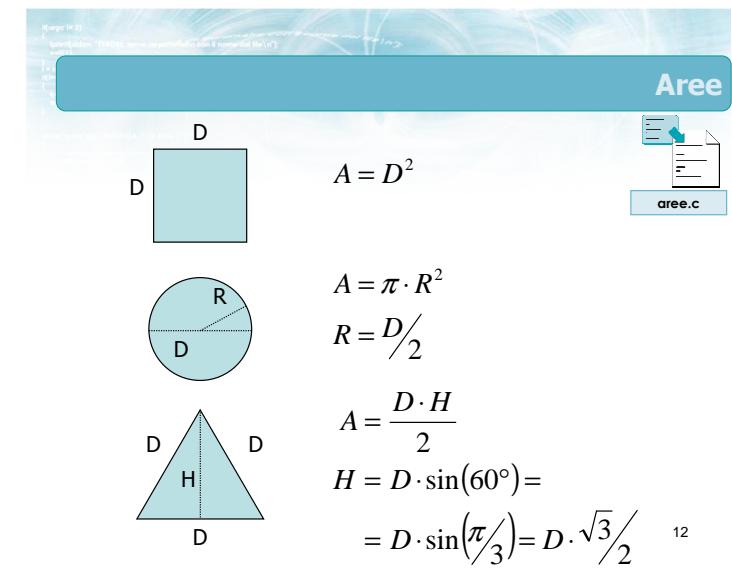
9



10



11



12

Avvertenze

- Per le funzioni matematiche (`sin`, `sqrt`, ...) occorre includere `math.h`
- Gli argomenti delle funzioni trigonometriche (`sin`, `cos`, ...) devono essere espressi in radianti
- Il calcolo del quadrato si ottiene moltiplicando la variabile per se stessa: $D^2 = D \times D$
- Il valore di π deve essere definito dal programmatore in un'apposita variabile
 - La costante `M_PI`, definita in `math.h`, non è più supportata dallo standard ANSI C

13

Esercizi proposti

Esercizio "Somma minuti"

Esercizio "Somma minuti" (1/2)

- Un consulente deve calcolare il numero di ore e minuti per cui ha lavorato per un cliente
- Il consulente ha lavorato in due distinte sessioni di lavoro, per ciascuna delle quali ha annotato il numero di ore e il numero di minuti impiegati



15

Esercizio "Somma minuti" (2/2)

- Si scriva un programma in C che, a partire dalle ore e minuti della prima sessione di lavoro e dalle ore e minuti della seconda sessione di lavoro, calcoli il numero di ore e minuti complessivi

16

Analisi

```
Prompt dei comandi
SOMMA MINUTI

Sessione di lavoro 1:
Numero di ore: 2
Numero di minuti: 45

Sessione di lavoro 2:
Numero di ore: 1
Numero di minuti: 30

Tempo totale: 4 ore e 15 minuti
```

17

Aritmetica dell'orologio

- Diciamo:
 - `ore1, min1` le ore/minuti della prima sessione
 - `ore2, min2` le ore/minuti della seconda sessione
 - `oretot, mintot` le ore/minuti totali
- Non è possibile semplicemente sommare ore e minuti separatamente, in quanto $min1+min2$ potrebbe essere maggiore di 59
- Bisogna tener conto del "riporto" nella somma dei minuti

18

Soluzione

- » $\text{mintot} = (\text{min1} + \text{min2}) \text{ modulo } 60$
- » $\text{oretot} = \text{ore1} + \text{ore2} + \text{riporto}$
 - $\text{riporto} = \text{parte intera di } (\text{min1} + \text{min2}) / 60$



minuti.c

Soluzione

- » $\text{mintot} = (\text{min1} + \text{min2}) \text{ modulo } 60$
- » $\text{oretot} = \text{ore1} + \text{ore2} + \text{riporto}$
 - $\text{riporto} = \text{parte intera di } (\text{min1} + \text{min2}) / 60$



minuti.c

```
int ore1, ore2, oretot ;  
int min1, min2, mintot, riporto ;  
  
...  
  
mintot = (min1 + min2) % 60 ;  
  
riporto = (min1 + min2) / 60 ;  
  
oretot = ore1 + ore2 + riporto ;
```

- ▶ Presentazione del linguaggio C
- ▶ Struttura base di un file sorgente in C
- ▶ Istruzioni minime per iniziare a programmare
 - Tipi fondamentali `int` e `float`
 - Istruzioni fondamentali di input/output
 - Istruzione di assegnazione
- ▶ Operazioni necessarie per compilare ed eseguire il programma

Primo programma in C

Sommario



Suggerimenti

- ▶ Analizzare sempre il comportamento previsto del programma **prima** di iniziare a scrivere il sorgente
 - Interazione con l'utente
 - Risoluzione manuale con carta e penna
- ▶ Abbondare con i **commenti**
- ▶ Leggere con attenzione tutti i messaggi di **errore** e di **warning** del compilatore, e correggerli
- ▶ Verificare il programma con diversi **dati di prova**

Materiale aggiuntivo

- ▶ Sul CD-ROM
 - Testi e soluzioni degli esercizi trattati nei lucidi
 - Scheda sintetica
 - Esercizi risolti
 - Esercizi proposti
- ▶ Esercizi proposti da altri libri di testo

```
#include <stdio.h>
#include <ctype.h>
#define MAXPARI 30
#define MAXIMPARI 80

int main(int argc, char *argv[])
{
    int freqMAXPAROLA; /* valore di confronto delle frequenze delle parole */
    int i, indice, lunghezza;
    FILE *f;

    if(argc != 2)
    {
        fprintf(stderr, "Errore: non sono stati inseriti i parametri richiesti.\n");
        exit(1);
    }

    f = fopen(argv[1], "r");
    if(f == NULL)
    {
        fprintf(stderr, "Errore: impossibile aprire il file %s", argv[1]);
        exit(1);
    }

    /* legge la parola più frequente */
    fscanf(f, "%s", MAXPAROLA);
    freqMAXPAROLA = lunghezza(MAXPAROLA);

    /* legge le altre parole */
    while(fscanf(f, "%s", MAXIMPARI) != EOF)
    {
        lunghezza(MAXIMPARI);
        if(lunghezza(MAXIMPARI) > freqMAXPAROLA)
            freqMAXPAROLA = lunghezza(MAXIMPARI);
    }

    /* stampa la parola più frequente */
    printf("La parola più frequente è %s\n", MAXPAROLA);
    exit(0);
}
```

Programmazione in C

Unità Scelte ed alternative

Scelte ed alternative

- » Il controllo di flusso
- » Istruzione if-else
- » Condizioni complesse
- » Istruzioni if-else annidate
- » Istruzione switch
- » Esercizi proposti
- » Sommario

2

Riferimenti al materiale

» Testi

- Kernighan & Ritchie: capitolo 3
- Cabodi, Quer, Sonza Reorda: capitoli 2, 4
- Dietel & Dietel: capitoli 2, 3

» Dispense

- Scheda: "Istruzioni di scelta in C"

3

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

#define MAXPAROLA 30
#define MAXIGRA 80

int main(int argc, char *argv[])
{
    int lung(MAXPAROLA); /* valore di condizione
    dove lunghezza della parola */
    char lungMAXIGRA;
    int i, indice, lunghezza;
    i=0;

    if(argc != 2)
    {
        printf("Errore: %s non ha un parametro da cui il numero del file (%s)\n", argv[0], argv[1]);
        exit(1);
    }

    if(isupper(argv[1][0]))
    {
        lung = MAXIGRA;
        lungMAXIGRA = 'A';
    }
    else
    {
        lung = MAXPAROLA;
        lungMAXIGRA = 'a';
    }

    for(i=0; i<lung; i++)
    {
        if(argv[1][i] == 'z')
        {
            printf("Errore: impossibile aprire il file %s", argv[1]);
            exit(1);
        }
    }

    if((fopen(argv[1], "r")) == NULL)
    {
        printf("Errore: non posso aprire il file %s", argv[1]);
        exit(1);
    }
}
```

Scelte ed alternative

Il controllo di flusso

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

#define MAXPAROLA 30
#define MAXIGRA 80

int main(int argc, char *argv[])
{
    int lung(MAXPAROLA); /* valore di condizione
    dove lunghezza della parola */
    char lungMAXIGRA;
    int i, indice, lunghezza;
    i=0;

    for(i=0; i<lung; i++)
    {
        if(argv[1][i] == 'z')
        {
            printf("Errore: impossibile aprire il file %s", argv[1]);
            exit(1);
        }
    }

    if((fopen(argv[1], "r")) == NULL)
    {
        printf("Errore: non posso aprire il file %s", argv[1]);
        exit(1);
    }
}
```

Il controllo di flusso

Concetto di flusso e di scelta



Il controllo di flusso

- ▶ Concetto di flusso e di scelta
- ▶ Rappresentazione grafica
- ▶ Condizioni booleane semplici
- ▶ Esempio

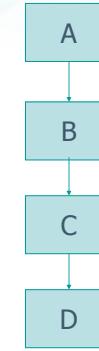
5



Flusso di esecuzione lineare

- ▶ Il programma C viene eseguito, di norma dalla prima istruzione all'ultima
- ▶ Il **flusso di esecuzione** è quindi normalmente di tipo **lineare**

```
istruzione_A ;
istruzione_B ;
istruzione_C ;
istruzione_D ;
```



7

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

int main()
{
    printf("Voto: %d\n", 18);
    printf("Voto: %d\n", 28);
}
```

Eccezioni



- ▶ In taluni casi il flusso di esecuzione deve variare:

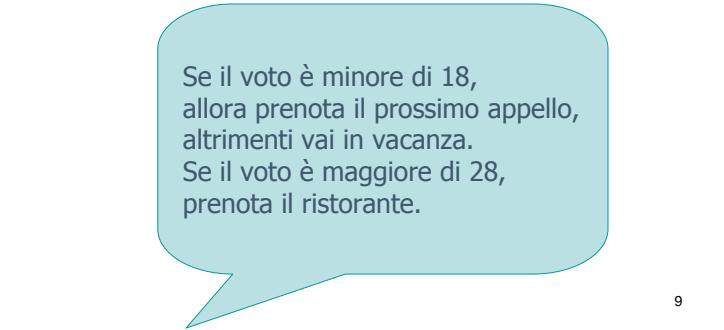
- in funzione del valore di qualche variabile di ingresso, occorre fare operazioni diverse
- una certa operazione deve essere ripetuta più volte
- alcune parti del programma vengono attivate solo se l'utente lo richiede
- ...
- ▶ In tal caso, il flusso di esecuzione non segue più esattamente l'ordine di scrittura delle istruzioni nel codice sorgente

8

Se il voto è minore di 18,
allora prenota il prossimo appello,
altrimenti vai in vacanza.
Se il voto è maggiore di 28,
prenota il ristorante.

Scelte

- ▶ Il tipo più semplice di flusso non lineare è costituito dalle scelte (o alternative)



9

Anatomia di una scelta

- Una condizione di scelta è caratterizzata da tre informazioni:
 - la "condizione" che determina la scelta (il voto è minore di 18)
 - le "operazioni" da svolgere qualora la condizione sia "vera" (prenota il prossimo appello)
 - le "operazioni" da svolgere qualora la condizione sia "falsa" (vai in vacanza)
- Le "operazioni" potrebbero anche essere assenti

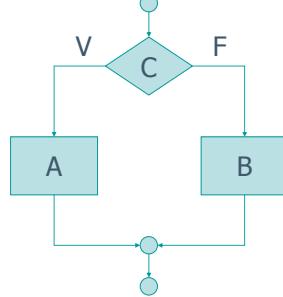


Il controllo di flusso

Rappresentazione grafica

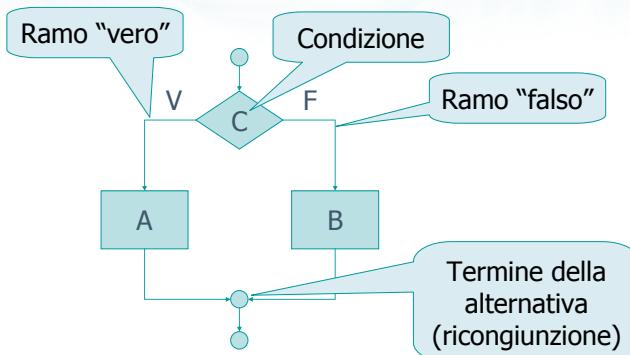
10

Notazione grafica



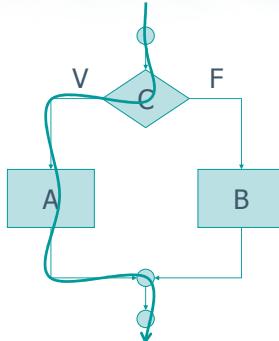
12

Notazione grafica



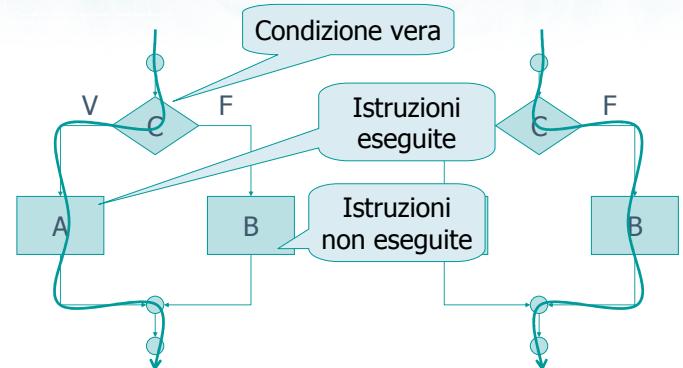
13

Flussi di esecuzione



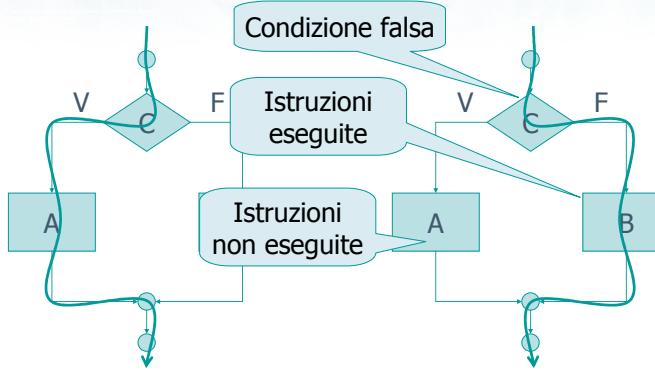
14

Flussi di esecuzione



15

Flussi di esecuzione



16

Il controllo di flusso

Condizioni booleane semplici

Il concetto di condizione

- » Che tipo di espressioni si utilizzano per esprimere la condizione **C** ?
 - devono dare una risposta univoca
 - Vero
 - Falso
 - sono basate sui valori delle variabili del programma
- » Le espressioni di questo tipo sono dette **booleane** in quanto generano dei valori che rispettano le leggi della logica di Boole (Vero o Falso)

18

Condizioni di confronto semplici

- » Spesso le condizioni sono espresse sotto forma di **confronti**
- » Confronto di uguaglianza
 - Uguale
 - Diverso
- » Confronto di ordine
 - Maggiore
 - Minore
 - Maggiore o uguale
 - Minore o uguale

19



Il controllo di flusso

Esempio

Equazioni di primo grado

- » Analizziamo il flusso di esecuzione di un programma in grado di risolvere le **equazioni di primo grado**:
- » Data l'equazione
 - $a x + b = 0$
- con **a** e **b** inseriti da tastiera, determinare il valore di **x** che risolve l'equazione

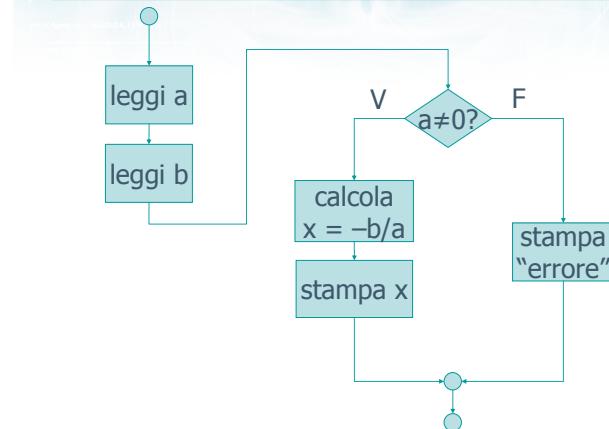
21

Equazione risolutiva

- » Dai corsi di matematica sappiamo che la soluzione dell'equazione:
 - $a x + b = 0$si può esprimere mediante la formula:
 - $x = -b / a$
- » Tale formula è valida solo se $a \neq 0$
- » Il programma dovrà comportarsi in modo diverso a seconda che a sia nullo o no

22

Soluzione



23

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

#define MAXPAROLA 30
#define MAXIGA 80

int main(int argc, char *argv[])
{
    int lung(MAXPAROLA); /* valore di confronto delle frequenze delle parole */
    char rig(MAXIGA); /* riga inserita dall'utente */
    int i, indice, lungparola;
    FILE *fptr;

    if(argc != 2)
    {
        printf("Errore: l'unico argomento deve essere il nome del file (*.c)\n");
        exit(1);
    }

    if((fptr = fopen(argv[1], "r")) == NULL)
    {
        printf("Errore: impossibile aprire il file %s", argv[1]);
        exit(1);
    }

    fgets(riga, MAXIGA, fptr) == NULL;
}
```

Scelte ed alternative

Istruzione if-else

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

#define MAXPAROLA 30
#define MAXIGA 80

int main(int argc, char *argv[])
{
    int lung(MAXPAROLA); /* valore di confronto delle frequenze delle parole */
    char rig(MAXIGA); /* riga inserita dall'utente */
    int i, indice, lungparola;
    FILE *fptr;

    for(i=0; MAXIGA(i); i++)
        riga[i] = '\0';

    if(argc != 2)
    {
        printf("Errore: l'unico argomento deve essere il nome del file (*.c)\n");
        exit(1);
    }

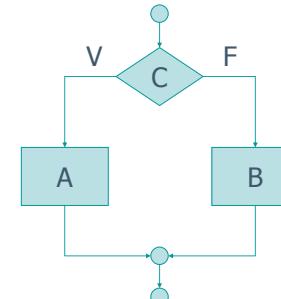
    if((fptr = fopen(argv[1], "r")) == NULL)
    {
        printf("Errore: impossibile aprire il file %s", argv[1]);
        exit(1);
    }

    fgets(riga, MAXIGA, fptr) == NULL;
}
```

Istruzione if-else

Sintassi dell'istruzione

```
if ( C )
{
    A ;
}
else
{
    B ;
}
```



5

```
(Argomento: Istruzioni di scelta in C)
Scrivere un programma che legge un file contenente una lista di parole e calcola la parola più frequente.
```

Istruzione if-else

- ▶ Sintassi dell'istruzione
- ▶ Operatori di confronto
- ▶ Esercizio proposto di esempio
- ▶ Risoluzione esercizio (parte I)
- ▶ Esecuzione del programma
- ▶ Completamento esercizio
- ▶ Risoluzione esercizio (parte II)

2

```
(Argomento: Istruzioni di scelta in C)
Scrivere un programma che legge un file contenente una lista di parole e calcola la parola più frequente.
```

Istruzioni di scelta in C

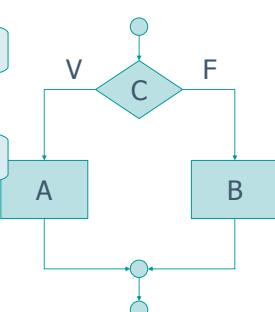
- ▶ L'operazione di scelta avviene mediante l'istruzione **if-else**
- ▶ Le condizioni di scelta possono essere semplici od elaborate
- ▶ Le scelte possono essere liberamente combinate tra loro, annidate
- ▶ In alcuni casi si può usare l'istruzione **switch** (trattata nella lezione "Istruzione switch")

4

```
(Argomento: Istruzioni di scelta in C)
Scrivere un programma che legge un file contenente una lista di parole e calcola la parola più frequente.
```

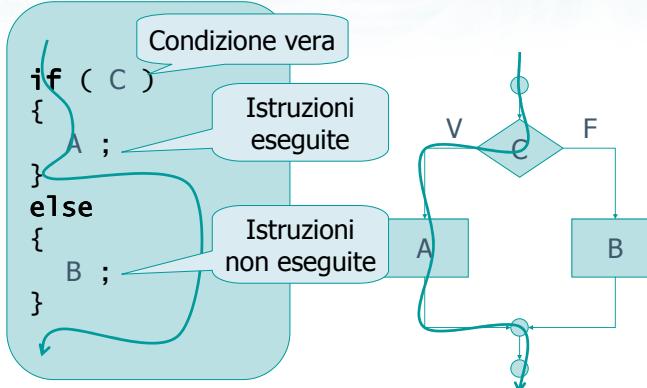
Istruzione if-else

Condizione
if (C)
{ A ; }
else
{ B ; }



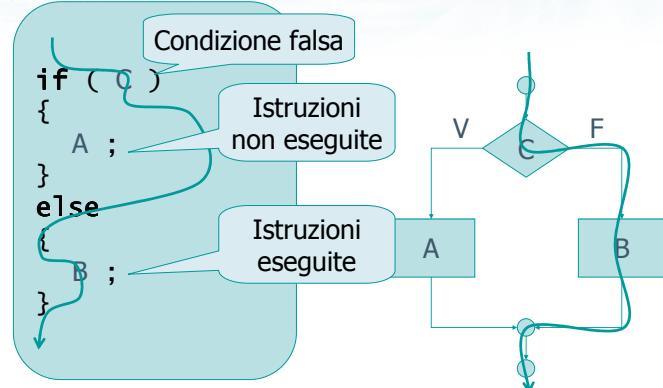
6

Condizione vera



7

Condizione falsa



8

Esempio

```

int a, b ;

printf("Immetti un numero: ");
scanf("%d", &a) ;
if ( a > 0 )
{
    printf("positivo\n") ;
    b = a ;
}
else
{
    printf("negativo\n") ;
    b = -a ;
}
printf("Il valore assoluto di %d e' %d", a, b) ;
  
```



9

Suggerimento

- Ad ogni nuova parentesi graffa aperta `{`, inserire degli **spazi aggiuntivi** ad inizio riga
- Tale tecnica, detta **indentazione**, permette una migliore leggibilità del codice
- È visivamente immediato riconoscere l'inizio e la fine dei blocchi di istruzioni
- Molti ambienti di sviluppo hanno funzioni di indentazione **automatica** o semi-automatica

10

Note

```

if ( C )
{
    A ;
}
else
{
    B ;
}
  
```

- La condizione `C` può essere semplice o complessa
- Il blocco `A` può essere composto da una sola istruzione, o da più istruzioni
- Il blocco `B` può essere composto da una sola istruzione, o da più istruzioni

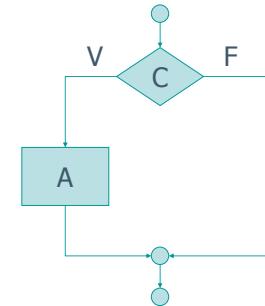
11

Caso particolare: istruzione `if`

```

if ( C )
{
    A ;
}
  
```

Manca la condizione else



12

Esempio

```
int a ;

printf("Immetti un numero: ");
scanf("%d", &a) ;
if ( a < 0 )
{
    /* è negativo, gli cambio segno */
    a = -a ;
}
printf("Il valore assoluto e' %d", a) ;
```



13

Caso particolare: parentesi graffe

```
if ( C )
A ;
else
{
    B ;
}
```

```
if ( C )
{
    A ;
}
else
    B ;
```

- Se il blocco A è composto da una sola istruzione, allora le parentesi graffe relative si possono omettere.
- Lo stesso vale per il blocco B.

```
if ( C )
A ;
else
B ;
```

14

Esempio

```
int a ;

printf("Immetti un numero: ");
scanf("%d", &a) ;

if ( a < 0 ) /* è negativo, gli cambio segno */
    a = -a ;

printf("Il valore assoluto e' %d", a) ;
```



15

Errore frequente

- È errato mettere il simbolo di punto-e-virgola ; dopo l'istruzione if

```
if ( a > 0 ) ;
a = -a ;
```

viene interpretato come

```
if ( a > 0 )
/*nulla*;
a = -a ;
```

forma corretta

```
if ( a > 0 )
a = -a ;
```

16



Errore frequente

- È errato fidarsi della sola indentazione

```
if ( a > 0 )
printf("neg");
a = -a ;
```

viene interpretato come

```
if ( a > 0 )
printf("neg");
a = -a ;
```

forma corretta

```
if ( a > 0 )
{
    printf("neg");
    a = -a ;
}
```

17



Suggerimento

- Anche se vi è una sola istruzione nel blocco "vero" o nel blocco "falso", **utilizzare sempre le parentesi graffe**
- In tal modo il programma sarà più leggibile, e sarà più facile aggiungere eventuali nuove istruzioni senza incappare in errori

18

```
#include <stdio.h>
#include <ctype.h>

#define MAXPARILO 30
#define MAXPARA 80

int main(int argc, char *argv[])
{
    int freqMAXPARILO; /* valori di confronto delle frequenze delle parole */
    int freqMAXPARA; /* valori di confronto delle lunghezze delle parole */

    if(argc != 2)
    {
        fprintf(stderr, "Usage: %s file\n", argv[0]);
        exit(1);
    }
    freqMAXPARILO = atoi(argv[1]);
    freqMAXPARA = 0;
    if(freqMAXPARILO >= 0 & freqMAXPARILO <= 30)
    {
        freqMAXPARA = freqMAXPARILO;
    }
    else
    {
        fprintf(stderr, "Error: impossibile aprire il file %s", argv[1]);
        exit(1);
    }
    FILE *fp = fopen(argv[1], "r");
    if(fp == NULL)
    {
        fprintf(stderr, "Error: non è possibile aprire il file %s", argv[1]);
        exit(1);
    }
    /* legge il file */
    /* stampa le parole con frequenza maggiore o uguale a freqMAXPARILO */
    /* stampa le parole con lunghezza maggiore o uguale a freqMAXPARA */
    /* stampa le parole con frequenza minore di freqMAXPARILO */
    /* stampa le parole con lunghezza minore di freqMAXPARA */
    /* stampa le parole con frequenza compresa tra freqMAXPARILO e freqMAXPARA */
}
```

Istruzione if-else

Operatori di confronto

(Argomenti) Introduzione - Cenni di storia - Linguaggio C - Struttura del linguaggio C - Operatori - Istruzioni - Funzioni - Esempi - Problemi - Risposte - Consigli - Bibliografia - Forum - Contatti - Sito dell'autore - Home

Le condizioni

- La condizione C è solitamente basata su un'operazione di confronto

- determinare se una variabile è uguale o meno a zero, o a un altro valore costante
- determinare se una variabile è uguale ad un'altra variabile, o è diversa, o è maggiore, o minore, ...
- determinare se una espressione, calcolata a partire da una o più variabili, è uguale, diversa, maggiore, minore, ... di una costante, o una variabile, o un'altra espressione

20

Operatori di confronto in C

Uguaglianza

- Uguale:** $a == b$
- Diverso:** $a != b$

Ordine

- Maggiore:** $a > b$
- Minore:** $a < b$
- Maggiore o uguale:** $a >= b$
- Minore o uguale:** $a <= b$

21

22

Esempi

- if(a == 0) ...**
- if(a == b) ...**
- if(a < 0) ...**
- if(a+b > 3) ...**
- if(x*y != y*x) ...**
- if(a/2 == (a+1)/2) ...**

Esempi

- if(a == 0) ...**
- if(a == b) ...**
- if(a < 0) ...**
- if(a+b > 3) ...**
- if(x*y != y*x) ...**
- if(a/2 == (a+1)/2) ...**

23

24

Esempi

- » if(a == 0) ...
- » if(a == b) ...
- » if(a < 0) ...
- » if(a+b > 3) ...
- » if(x*y != y*x) ...
- » if(a/2 == (a+1)/2) ...

25

Esempi

- » if(a == 0) ...
- » if(a == b) ...
- » if(a < 0) ...
- » if(a+b > 3) ...
- » if(x*y != y*x) ...
- » if(a/2 == (a+1)/2) ...

26

Esempi

- » if(a == 0) ...
- » if(a == b) ...
- » if(a < 0) ...
- » if(a+b > 3) ...
- » if(x*y != y*x) ...
- » if(a/2 == (a+1)/2) ...

27

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

#define MAXPAROLA 30
#define MAXIGRA 80

int main(int argc, char *argv[])
{
    int lungMAXPAROLA; /* vettore di confronto
    con le massime lunghezze delle parole */
    char c[MAXIGRA];
    int i, indice, lungParola;
    FILE *f;

    f=fopen(argv[1], "r");
    if(f==NULL)
        printf("ERRORE: impossibile aprire il file %s", argv[1]);
    else
    {
        fgets(c, MAXIGRA, f);
        if(c[0]=='.')
            printf("File vuoto\n");
        else
        {
            lungMAXPAROLA=0;
            for(i=0; c[i] != '\0'; i++)
                if(isalpha(c[i]))
                    lungMAXPAROLA++;
            if(lungMAXPAROLA > MAXPAROLA)
                printf("Parola troppo lunga\n");
            else
            {
                lungParola=0;
                for(i=0; c[i] != '\0'; i++)
                    if(isalpha(c[i]))
                        lungParola++;
                if(lungParola > MAXPAROLA)
                    printf("Parola troppo lunga\n");
                else
                {
                    for(i=0; c[i] != '\0'; i++)
                        if(isalpha(c[i]))
                            c[i]=tolower(c[i]);
                    if(strcmp(c, "c") == 0)
                        indice=1;
                    else
                        indice=0;
                    printf("%d\n", indice);
                }
            }
        }
    }
    fclose(f);
}
```

Istruzione if-else

Esercizio proposto di esempio



Errore frequente

- » Confondere l'operatore di **assegnazione** = con l'operatore di **confronto** ==
- » Regola pratica: le parentesi tonde richiedono ==
- » Regola pratica: il punto-e-virgola richiede =

```
/* assegnazione */
a = b + c ;
/* confronto */
if ( a == b+c )...
```

```
/* assegnazione */
a == b + c ;
/* confronto */
if ( a = b+c )...
```

28

- » Si scriva un programma in linguaggio C che legga due numeri da tastiera, detti A e B, e determini le seguenti informazioni, stampandole a video:
 - determini se B è un numero positivo o negativo
 - determini se A è un numero pari o dispari
 - calcoli il valore di A+B
 - determini quale scelta dei segni nell'espressione $(\pm A) + (\pm B)$ porta al risultato massimo, e quale è questo valore massimo.

30

Struttura generale

- » Leggi A e B
- » Controlla il segno di B
 - Stampa il messaggio opportuno
- » Controlla la parità di A
 - Stampa il messaggio opportuno
- » Calcola A+B
 - Stampa il risultato
- » ...l'ultimo punto è più difficile
 - ...ci pensiamo dopo!

31

Lettura dei dati

- » Leggi A e B

```
int a, b ;  
  
printf("Immetti A");  
scanf("%d", &a) ;  
  
printf("Immetti B");  
scanf("%d", &b) ;
```

32

Controllo del segno

- » Controlla il segno di B
 - Stampa il messaggio opportuno

```
if( b > 0 )  
{  
    printf("B e' positivo\n");  
}  
else  
{  
    printf("B e' negativo o nullo\n");  
}
```

33

Controllo della parità

- » Controlla la parità di A
 - Stampa il messaggio opportuno

```
if( "a è pari" )  
{  
    printf("A e' pari\n");  
}  
else  
{  
    printf("A e' dispari\n");  
}
```

34

Numeri pari

- » Come determinare se un numero è pari?
- » Calcoliamo la divisione per 2, e controlliamo il resto
 - Se il resto della divisione per 2 vale 0, allora il numero è pari
 - Se il resto della divisione per 2 vale 1, allora il numero è dispari
- » Il calcolo del resto si ottiene con l'operatore %

`if("a è pari")`

`if((a % 2) == 0)`

35

```
#include <iomanip.h>  
#include <string.h>  
#include <ctype.h>  
  
#define MAXPAROLA 30  
#define MAXIGRA 80  
  
int main(argc, char *argv[]){  
    int freqMAXPAROLA; /* variabile di controllo  
    delle frequenze delle lunghezze delle parole */  
    char c, AXIGRA[ ];  
    int i, indice, lunghezza ;  
    FILE *f ;  
  
    f=fopen(MAXIGRA, "r") ;  
    if(f==NULL){  
        cout<<"ERRORE: impossibile aprire il file \"IGRA\"."<<endl ;  
        exit(1) ;  
    }  
  
    while(fgets(AXIGRA, MAXIGRA, f)!=NULL){  
        /*  
        *  
        */  
    }  
  
    fclose(f) ;  
}
```

Istruzione if-else

Risoluzione esercizio (parte I)

Risoluzione esercizio "Controlla A e B"

- Risolviamo interattivamente l'esercizio



controlla-ab-v1.c

```
#include <stdio.h>
#include <limits.h>
#include <ctype.h>

#define MAXPAROLA 30
#define MAXSIGA 80

int main(argc)
{
    int a, b;
    if(argc != 2) {
        printf("Errore, inserire 2 argomenti\n");
        exit(1);
    }
    a = atoi(argv[1]);
    b = atoi(argv[2]);
    if(a < 0 || b < 0) {
        printf("I segni di A e B non sono corretti\n");
        exit(1);
    }
    if(a > b) {
        printf("A > B\n");
    } else if(b > a) {
        printf("B > A\n");
    } else {
        printf("A = B\n");
    }
}
```

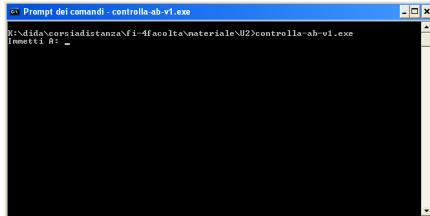
37

Istruzione if-else

Esecuzione del programma

Verifica esercizio "Controlla A e B"

- Compiliamo il programma
- Eseguiamolo con alcuni dati di prova, verificandone il comportamento corretto



39

```
#include <stdio.h>
#include <limits.h>
#include <ctype.h>

#define MAXPAROLA 30
#define MAXSIGA 80

int main(argc)
{
    int a, b;
    if(argc != 2) {
        printf("Errore, inserire 2 argomenti\n");
        exit(1);
    }
    a = atoi(argv[1]);
    b = atoi(argv[2]);
    if(a < 0 || b < 0) {
        printf("I segni di A e B non sono corretti\n");
        exit(1);
    }
    if(a > b) {
        printf("A > B\n");
    } else if(b > a) {
        printf("B > A\n");
    } else {
        printf("A = B\n");
    }
}
```

Istruzione if-else

Completamento esercizio

Completamento esercizio "Controlla A e B"

- Abbiamo verificato il corretto funzionamento
- Rimane da implementare il punto:
 - determini quale scelta dei segni nell'espressione $(\pm A) + (\pm B)$ porta al risultato massimo, e quale è questo valore massimo.
- Appaiono possibili diverse strategie:
 - Calcolare le 4 combinazioni e scegliere il massimo
 - Riscrivere algebricamente l'espressione

41

- La prima strategia prevede di calcolare:

- $R1 = (+A) + (+B)$
- $R2 = (+A) + (-B)$
- $R3 = (-A) + (+B)$
- $R4 = (-A) + (-B)$

- Dopo avere calcolato queste 4 variabili, occorre confrontarle per determinare quale è maggiore di tutte le altre.

- In questo caso è inutilmente macchinoso.

Strategia 1

Strategia 2

- Ragionando algebricamente, la massima somma che si può ottenere dall'espressione $(\pm A) + (\pm B)$ sarà quando
 - $\pm A$ è positivo
 - $\pm B$ è positivo
- In altre parole, è sufficiente calcolare la somma dei valori assoluti
 - $|A| + |B|$

43

Valore assoluto

- Il valore assoluto di una variabile è pari a
 - il valore dell'opposto della variabile
 - se la variabile è negativa
 - il valore della variabile stessa
 - se la variabile è positiva

44

Valore assoluto

- Il valore assoluto di una variabile è pari a
 - il valore dell'opposto della variabile
 - se la variabile è negativa
 - il valore della variabile stessa
 - se la variabile è positiva

```
if( a<0 )
{
    va = -a ;
}
else
{
    va = a ;
}
```

45

Valore assoluto

- Il valore assoluto di una variabile è pari a
 - il valore dell'opposto della variabile
 - se la variabile è negativa
 - il valore della variabile stessa
 - se la variabile è positiva

```
if( a<0 )
{
    va = -a ;
}
else
{
    va = a ;
}
```

```
if( a<0 )
{
    a = -a ;
}
```

46

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

#define MAXPAROLA 30
#define MAXIGRA 80

int main()
{
    char str[MAXIGRA]; /* vettore di caratteri
    dove trascrivere l'ingresso della parola */
    char c;
    int i, indice, lunghezza;
    i = 0;

    if((c = getchar()) == '#') /* fine */
        exit(0);

    while(c != '#')
    {
        if(c == ' ') /* spazio */
            continue;
        else
        {
            str[i] = c;
            i++;
        }
        c = getchar();
    }

    str[i] = '\0';
    lunghezza = i;
}

int main()
{
    int a, b;
    printf("Inserisci due numeri interi: ");
    scanf("%d %d", &a, &b);
    if(a > b)
        swap(&a, &b);
    else
        swap(&b, &a);
    printf("I numeri sono: %d e %d", a, b);
}
```

Istruzione if-else

Risoluzione esercizio (parte II)

```
#include <stdio.h>
#include <math.h>
#include <string.h>
#include <ctype.h>
#include <conio.h>

char str[100];
int a, b;
char c;
int main()
{
    printf("Inserisci due numeri interi: ");
    scanf("%d %d", &a, &b);
    if(a > b)
        swap(&a, &b);
    else
        swap(&b, &a);
    printf("I numeri sono: %d e %d", a, b);
}

void swap(int *a, int *b)
{
    int temp;
    temp = *a;
    *a = *b;
    *b = temp;
}
```

- Completiamo l'esercizio, codificandolo e verificandolo



48

- » Operatori booleani
- » Operatori booleani in C
- » Esercizio proposto
- » Verifica della soluzione

Scelte ed alternative

Condizioni complesse

2

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>
```

```
#define MAXPAROLA 30
#define MAXIGRA 80
```

```
int main(int argc, char *argv[])
{
    int lungMAXPAROLA; /* valore di confronto
    cioè lunghezza massima della parola */
    char rigaMAXIGRA; /* lunghezza massima
    delle righe */
    int i;
}
```

```
for(i=0; i<MAXIGRA; i++)
    lungi[i]=0;

if(argc > 1)
{
    if(strcmp(argv[1], "TESTC") == 0)
        lungMAXPAROLA = 5;
    else
        lungMAXPAROLA = 10;
}

for(i=0; i<MAXIGRA; i++)
{
    if(lungi[i] > lungMAXPAROLA)
        break;
    if(isspace(argv[i][0]))
        continue;
    if(isupper(argv[i][0]))
        argv[i][0] = argv[i][0] - 'A' + 'a';
    else
        argv[i][0] = argv[i][0] - 'a' + 'A';

    if(argv[i][0] >='A' & argv[i][0] <='Z')
        lungi[i]++;
}
```

Condizioni complesse

Operatori booleani

4

```
{if(argc > 1)
    if(strcmp(argv[1], "TESTB") == 0)
        lungMAXPAROLA = 5;
    else
        lungMAXPAROLA = 10;
}

for(i=0; i<MAXIGRA; i++)
{
    if(lungi[i] > lungMAXPAROLA)
        break;
    if(isspace(argv[i][0]))
        continue;
    if(isupper(argv[i][0]))
        argv[i][0] = argv[i][0] - 'A' + 'a';
    else
        argv[i][0] = argv[i][0] - 'a' + 'A';

    if(argv[i][0] >='A' & argv[i][0] <='Z')
        lungi[i]++;
}
```

Operatori booleani

- » Date due qualsiasi condizioni booleane X ed Y (condizioni semplici, o a loro volta complesse):
- » **X AND Y**
 - è vero se sia X che Y sono veri
- » **X OR Y**
 - è vero se è vero X (indipendentemente da Y) oppure se è vero Y (indipendentemente da X) o se sono veri entrambi
- » **NOT X**
 - è vero se X è falso, è falso se X è vero

5

```
{if(argc > 1)
    if(strcmp(argv[1], "TESTB") == 0)
        lungMAXPAROLA = 5;
    else
        lungMAXPAROLA = 10;
}

for(i=0; i<MAXIGRA; i++)
{
    if(lungi[i] > lungMAXPAROLA)
        break;
    if(isspace(argv[i][0]))
        continue;
    if(isupper(argv[i][0]))
        argv[i][0] = argv[i][0] - 'A' + 'a';
    else
        argv[i][0] = argv[i][0] - 'a' + 'A';

    if(argv[i][0] >='A' & argv[i][0] <='Z')
        lungi[i]++;
}
```

Logica Booleana

- » Le condizioni “semplici” (es. confronti) forniscono un valore booleano (vero/falso)
- » Spesso occorre prendere delle scelte in funzione del valore di più condizioni semplici
 - Es: x è compreso tra a e b?
 - $x \geq a$, e contemporaneamente $x \leq b$
 - Es: ci sono promossi?
 - $voto1 \geq 18$, oppure $voto2 \geq 18$
- » A questo scopo si possono usare gli operatori booleani

4

```
{if(argc > 1)
    if(strcmp(argv[1], "TESTB") == 0)
        lungMAXPAROLA = 5;
    else
        lungMAXPAROLA = 10;
}

for(i=0; i<MAXIGRA; i++)
{
    if(lungi[i] > lungMAXPAROLA)
        break;
    if(isspace(argv[i][0]))
        continue;
    if(isupper(argv[i][0]))
        argv[i][0] = argv[i][0] - 'A' + 'a';
    else
        argv[i][0] = argv[i][0] - 'a' + 'A';

    if(argv[i][0] >='A' & argv[i][0] <='Z')
        lungi[i]++;
}
```

Esempi

- » x è compreso tra a e b?
 - $(x \geq a) \text{ AND } (x \leq b)$
 - se so già che $b \geq a$
 - $((b \geq a) \text{ AND } (x \geq a)) \text{ AND } ((x \leq b) \text{ OR } (b < a) \text{ AND } (x \leq a)) \text{ AND } (x \geq b)$
 - nel caso generale
- » ci sono promossi?
 - $(voto1 \geq 18) \text{ OR } (voto2 \geq 18)$

6

```
#include <stdio.h>
#include <ctype.h>

#define MAXPARIOLA 30
#define MAXRGA 80

int main(int argc, char *argv[])
{
    int freqMAXPARIOLA; /* valori di confronto delle frequenze delle parole */
    int freqMAXRGA; /* valori di confronto delle lunghezze delle parole */

    if(argc != 2)
    {
        fprintf(stderr, "Usage: %s file\n", argv[0]);
        exit(1);
    }

    freqMAXPARIOLA = atoi(argv[1]);
    freqMAXRGA = atoi(argv[2]);

    if(freqMAXPARIOLA > MAXPARIOLA || freqMAXRGA > MAXRGA)
    {
        fprintf(stderr, "MAXPARIOLA, imposta valore minore o uguale a %d", MAXPARIOLA);
        exit(1);
    }
}
```

Condizioni complesse

Operatori booleani in C

Operatori booleani in C

Operatore booleano	Sintassi in C	Esempio
AND	&&	(x>=a)&&(x<=b)
OR		(v1>=18) (v2>=18)
NOT	!	!(a>b)

8

Contesto di utilizzo

- » Solitamente gli operatori booleani **&&** **||** **!** si utilizzano all'interno della condizione dell'istruzione **if**, per costruire condizioni complesse
 - Più avanti vedremo come si usano anche nella condizione del costrutto **while**
- » Tali operatori lavorano su operandi che solitamente sono:
 - Condizioni semplici (es. **(a>b) || (a!=1)**)
 - Risultati di altri operatori booleani (es. **((a>b)&&(b>c)) || (c==0)**)

9

Precedenza degli operatori

- » Quando più operatori booleani e di confronto sono presenti nella stessa espressione, vengono valutati come segue:
 - Prima gli operatori di confronto
 - == != > < >= <=
 - Poi la negazione NOT
 - !
 - In seguito la congiunzione AND
 - &&
 - Infine la disgiunzione OR
 - ||

10

Esempi

```
if ( a>0 && b>0 )
```

Esempi

```
if ( a>0 && b>0 )
```

```
if ( a<=0 || b<=0 )
```

11

12

Esempi

```
if ( a>0 && b>0 )
```

```
if ( a<=0 || b<=0 )
```

```
if ( !(a>0 && b>0) )
```

Esempi

```
if ( a>0 && b>0 )
```

```
if ( a<=0 || b<=0 )
```

```
if ( a==0 || (a!=0 && b==0) )
```

13

14

Esempi

```
if ( a>0 && b>0 )
```

```
if ( a<=0 || b<=0 )
```

```
if ( a==0 || (a!=0 && b==0) )
```

```
if ( b>=a && x>=a && x<=b ||  
b<a && x<=a && x>=b )
```

Esempi

```
if ( a>0 && b>0 )
```

```
if ( a<=0 || b<=0 )
```

```
if ( a==0 || (a!=0 && b==0) )
```

```
if ( b>=a && x>=a && x<=b ||  
b<a && x<=a && x>=b )
```

15

16

Esempi

```
if ( a>0 && b>0 )
```

```
if ( a<=0 || b<=0 )
```

```
if ( a==0 || (a!=0 && b==0) )
```

```
if ( b>=a && x>=a && x<=b ||  
b<a && x<=a && x>=b )
```

Esempi

```
if ( a>0 && b>0 )
```

```
if ( a<=0 || b<=0 )
```

```
if ( a==0 || (a!=0 && b==0) )
```

```
if ( b>=a && x>=a && x<=b ||  
b<a && x<=a && x>=b )
```

17

18

Esempi

```
if ( a>0 && b>0 )
```

```
if ( a<=0 || b<=0 )
```

```
if ( a==0 || (a!=0 && b==0) )
```

```
if ( b>=a && x>=a && x<=b ||  
b< a && x<=a && x>=b )
```

```
if ( ((b>=a) && (x>=a) && (x<=b)) ||  
((b<a) && (x<=a) && (x>=b))  
)
```

19

Suggerimento

- In presenza di espressioni complesse, è sempre conveniente abbondare con le parentesi

- leggibilità

- indipendenza dalle precedenze degli operatori

~~if (b>=a && x>=a && x<=b ||
b< a && x<=a && x>=b)~~~~if (((b>=a) && (x>=a) && (x<=b)) ||
((b<a) && (x<=a) && (x>=b))
)~~

20



Errore frequente

- È errato usare in successione più operatori di confronto senza collegarli mediante operatori booleani

```
if ( a > b > 0 )
```

forma corretta

```
if ( (a > b) &&  
     (b > 0)  
)
```

```
if ( a==b==c )
```

forma corretta

```
if ( (a == b) &&  
     (b == c)  
)
```

22



Errore frequente

- È errato "sottintendere" parte di un confronto
- Esempio: "se a o b sono diversi da uno"

```
if ( a || b != 1 )
```

forma corretta

```
if ( (a!=1) ||  
     (b!=1) )
```

```
if ( (a||b) != 1 )
```

forma corretta

```
#include <stdio.h>  
#include <string.h>  
#include <ctype.h>  
  
#define MAXPAROLA 30  
#define MAXIGRA 80  
  
int main(argc, argv){  
    char *arg[argc];  
    int lenMAXPAROLA; /* vettore di confronto  
    delle lunghezze delle parole */  
    char *MAXIGRA; /*  
    int i, indice, lunghezza ;  
    FILE *f ;  
  
    for(i=0; i<argc; i++)  
        arg[i] = argv[i];  
  
    if(arg[0] == NULL) /* se non viene specificato il nome del file */  
        lenMAXPAROLA = 10 ;  
    else  
        lenMAXPAROLA = strlen(argv[0]);  
  
    if(arg[1] == NULL) /* se non viene specificato il nome del file */  
        MAXIGRA = "TESTIGRA";  
    else  
        MAXIGRA = arg[1];  
  
    f = fopen(MAXIGRA, "r");  
    if(f == NULL) {  
        printf("ERRORE: impossibile aprire il file %s", MAXIGRA);  
        exit(1);  
    }  
  
    printf("Lung. %d\n", lenMAXPAROLA);  
    printf("Lung. %d\n", strlen(MAXIGRA));  
}
```

Condizioni complesse

Esercizio proposto

Esercizio "Classificazione triangolo 1"

- Si scriva un programma in linguaggio C che legga da tastiera i valori delle lunghezze dei tre lati di un triangolo (detti A, B e C), e determini:

- se il triangolo è equilatero
- se il triangolo è isoscele
- se il triangolo è scaleno
- se il triangolo è rettangolo

- Nota: si assuma, per il momento, che i valori A, B, C descrivano correttamente un triangolo

24

Analisi del problema

- Ricordiamo le condizioni matematiche relative alla classificazione dei triangoli:
- Equilatero: le lunghezze dei tre lati A, B, C sono uguali tra loro
 - Isoscele: le lunghezze di [almeno] due dei tre lati A, B, C sono uguali tra loro
 - ogni triangolo equilatero è anche isoscele
 - Scaleno: le lunghezze dei tre lati A, B, C sono tutte diverse tra loro
 - Rettangolo: possiede un angolo retto
 - vale il teorema di Pitagora

25

Espressioni matematiche (I)

- Equilatero
- $A = B = C$
 - $(A = B) \text{ AND } (B = C) \text{ AND } (A = C)$
 - $(A = B) \text{ AND } (B = C)$
- Isoscele
- $(A = B) \text{ OR } (B = C) \text{ OR } (A = C)$
- Scaleno
- $(A \neq B) \text{ AND } (A \neq C) \text{ AND } (B \neq C)$

26

Espressioni matematiche (II)

Rettangolo

- Teorema di Pitagora
 - Ipotenusa² = Cateto² + Cateto²
- L'ipotenusa può essere uno qualunque dei lati A, B oppure C
- $(A^2 = B^2+C^2) \text{ OR } (B^2 = A^2+C^2) \text{ OR } (C^2 = A^2+B^2)$

27

Condizioni in C

- Equilatero
- $a==b \text{ && } b==c$
- Isoscele
- $a==b \text{ || } b==c \text{ || } a==c$
- Scaleno
- $a!=b \text{ && } b!=c \text{ && } a!=c$
- Rettangolo
- $(a*a == b*b + c*c) \text{ || }$
 - $(b*b == a*a + c*c) \text{ || }$
 - $(c*c == a*a + b*b)$

28

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

#define MAXPAROLA 30
#define MAXIGRA 80

int main(argc, argv, char *argv[])
{
    int lungMAXPAROLA; /* variabile di controllo
    per la lunghezza delle parole */
    char parolaAXINGAL;
    int i, indice, lungparola;
    FILE *f;

    for(i=0; i<MAXPAROLA; i++)
        lungMAXPAROLA++;

    f = fopen(argv[1], "r");
    if(f == NULL)
    {
        printf("ERRORE: impossibile aprire il file %s", argv[1]);
        exit(1);
    }

    if(fscanf(f, "%c", &parolaAXINGAL) != 1)
    {
        printf("ERRORE: impossibile leggere il carattere %c", parolaAXINGAL);
        exit(1);
    }

    if(parolaAXINGAL != '#')
    {
        printf("ERRORE: non si è inserito il simbolo di fine file (%c)", parolaAXINGAL);
        exit(1);
    }

    if(fscanf(f, "%s", argv[2]) != 1)
    {
        printf("ERRORE: impossibile leggere la parola inserita (%s)", argv[2]);
        exit(1);
    }

    if(strlen(argv[2]) > MAXIGRA)
    {
        printf("ERRORE: la parola inserita (%s) supera il limite massimo (%d)", argv[2], MAXIGRA);
        exit(1);
    }

    fclose(f);
}
```

Condizioni complesse

Verifica della soluzione

```
Verifica "Classificazione triangolo 1"

➤ Analizziamo il codice dell'esercizio e
verifichiamone il corretto funzionamento
```

```
File C:\...\Saluzione\proposta.esercizio.2.2.A.c
1 //include <stdio.h>
2 //include <csttio.h>
3 #define MAXLATO 10
4
5 void main(void)
6 {
7     int a, b, c;
8     /* Inizializzazione */
9     a = 0; b = 0; c = 0;
10    printf("Inserire i lati del triangolo: ");
11    scanf("%d", &a);
12    scanf("%d", &b);
13    scanf("%d", &c);
14
15    /* Controllo se i lati sono diversi */
16    if(a != b && b != c && a != c)
17    {
18        /* Scaleno */
19        printf("Scaleno\n");
20    }
21    else if(a == b && b == c && a == c)
22    {
23        /* Equilatero */
24        printf("Equilatero\n");
25    }
26    else
27    {
28        /* Isoscele */
29        printf("Isoscele\n");
30    }
31 }
```

30

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

#define MAXPAROLA 30
#define MAXIGRA 80
```

```
int main(int argc, char *argv[])
{
    int lungMAXPAROLA; /* valore di costante
    dove lungMAXPAROLA è la lunghezza della parola */
    int i, indice, lungparola;
    char digi[MAXIGRA];
    FILE *fptr;

    if(argc != 2)
    {
        fprintf(stderr, "ERRORE: non sono presenti dati il numero del file\n");
        exit(1);
    }

    if(indice = fopen(argv[1], "r"))
    {
        fscanf(stdin, "%d", &lungMAXPAROLA);
        fscanf(stdin, "%s", digi);
        lungparola = strlen(digi);
        if(lungparola > MAXIGRA)
        {
            fprintf(stderr, "ERRORE: impossibile aprire il file %s", argv[1]);
            exit(1);
        }
    }
    else
    {
        fprintf(stderr, "ERRORE: non è possibile aprire il file %s", argv[1]);
        exit(1);
    }

    while(fgets(digi, MAXIGRA, fptr) != NULL)
```

Scelte ed alternative

Istruzioni if-else annidate

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

#define MAXPAROLA 30
#define MAXIGRA 80

int main(int argc, char *argv[])
{
    int lungMAXPAROLA; /* valore di costante
    dove lungMAXPAROLA è la lunghezza della parola */
    int i, indice, lungparola;
    char digi[MAXIGRA];
    FILE *fptr;

    if(argc != 2)
    {
        fprintf(stderr, "ERRORE: non sono presenti dati il numero del file\n");
        exit(1);
    }

    if(indice = fopen(argv[1], "r"))
    {
        fscanf(stdin, "%d", &lungMAXPAROLA);
        fscanf(stdin, "%s", digi);
        lungparola = strlen(digi);
        if(lungparola > MAXIGRA)
        {
            fprintf(stderr, "ERRORE: impossibile aprire il file %s", argv[1]);
            exit(1);
        }
    }
    else
    {
        fprintf(stderr, "ERRORE: non è possibile aprire il file %s", argv[1]);
        exit(1);
    }

    while(fgets(digi, MAXIGRA, fptr) != NULL)
```

Istruzioni if-else annidate

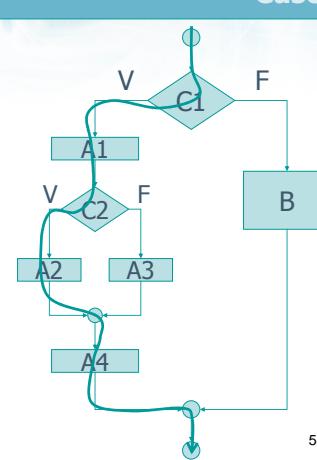
Annidamento di istruzioni if-else

```
(Merge de 3)
Spostare linea "FILE*fptr; non è possibile aprire il numero del file\n";
```

Istruzioni if-else annidate

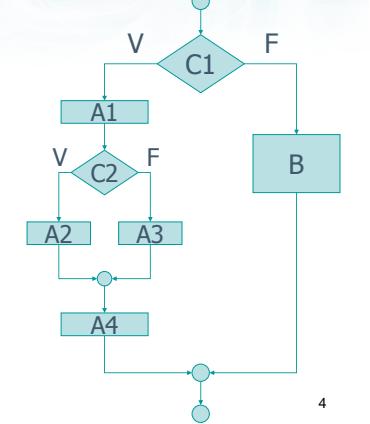
- ▶ Annidamento di istruzioni if-else
- ▶ Opzionalità del ramo else
- ▶ Catene if-else if-...-else
- ▶ Esercizio proposto
- ▶ Verifica della soluzione

2



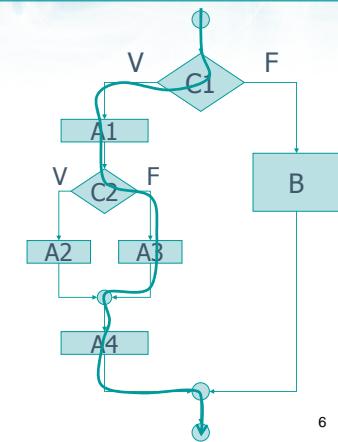
5

- ▶ C1 vero, C2 vero
 - Istruzioni eseguite: A1, A2, A4



4

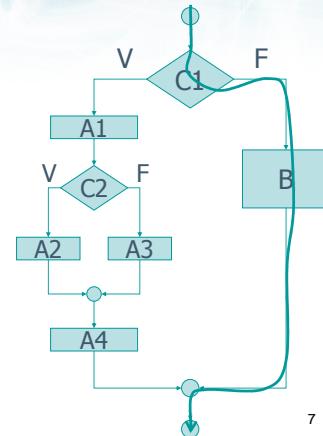
- ▶ C1 vero, C2 vero
 - Istruzioni eseguite: A1, A2, A4
- ▶ C1 vero, C2 falso
 - Istruzioni eseguite: A1, A3, A4



6

Caso 3

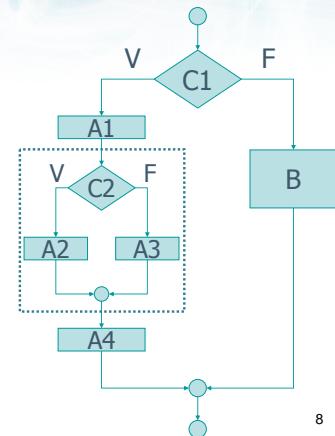
- C1 vero, C2 vero
 - Istruzioni eseguite: A1, A2, A4
- C1 vero, C2 falso
 - Istruzioni eseguite: A1, A3, A4
- C1 falso, C2 indifferente
 - Istruzioni eseguite: B



7

Corretto annidamento

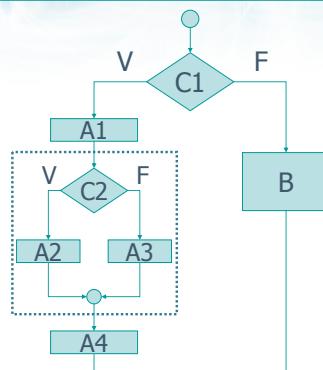
- L'intero blocco di scelta più interno (dalla condizione fino al ricongiungimento) deve essere completamente contenuto all'interno di uno dei rami del blocco più esterno



8

Sintassi C

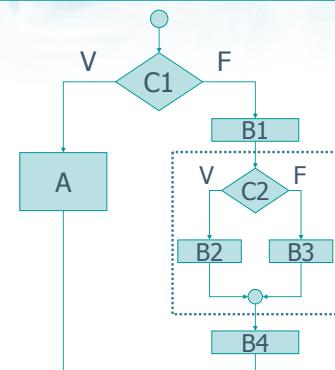
```
if ( C1 )
{
  A1 ;
  if ( C2 )
  {
    A2 ;
  }
  else
  {
    A3 ;
  }
  A4 ;
}
else
{
  B ;
}
```



9

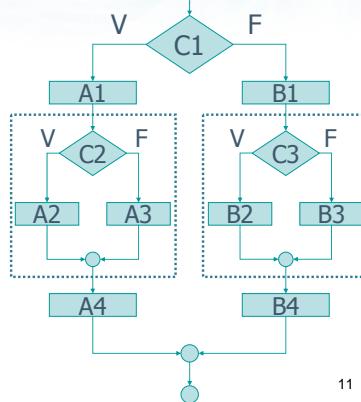
Sintassi C

```
if ( C1 )
{
  A ;
}
else
{
  B1 ;
  if ( C2 )
  {
    B2 ;
  }
  else
  {
    B3 ;
  }
  B4 ;
}
```



10

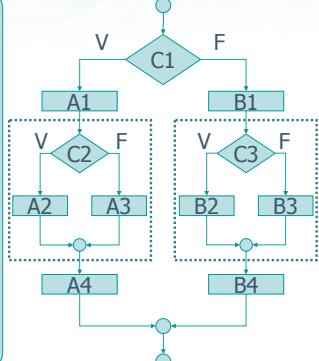
Caso generale



11

Sintassi C

```
if ( C1 )
{
  A1 ;
  if ( C2 )
  {
    A2 ;
  }
  else
  {
    A3 ;
  }
  A4 ;
}
else
{
  B1 ;
  if ( C3 )
  {
    B2 ;
  }
  else
  {
    B3 ;
  }
  B4 ;
}
```



12

Sintassi C

- » Un'istruzione **if** può comparire ovunque
 - anche all'interno del blocco "vero" o "falso" di un'altra istruzione **if**
- » Occorre garantire il corretto annidamento delle istruzioni
 - le istruzioni annidate vanno completamente contenute tra le parentesi graffe **{...}**

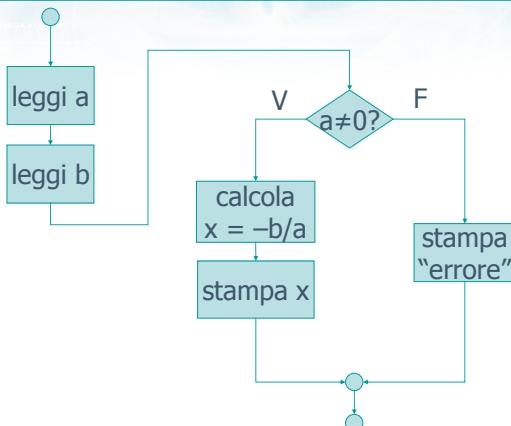
13

Esempio

- » Ricordiamo l'esercizio sull'algoritmo risolutivo delle equazioni di primo grado
 - $a x + b = 0$
- » La soluzione è:
 - $x = -b / a$
 - solo se $a \neq 0$

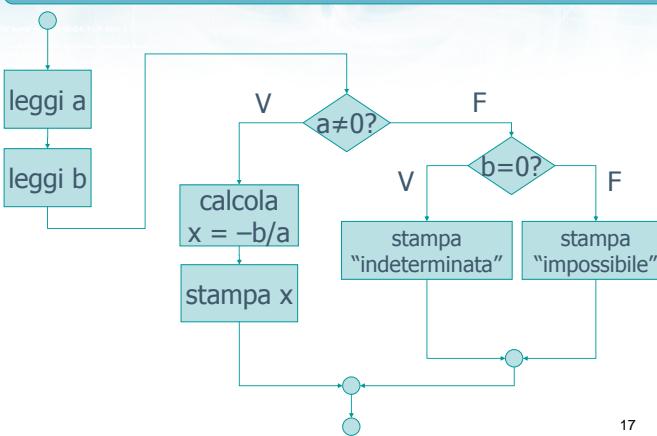
14

Soluzione (parziale)



15

Soluzione (completa)



17

Esempio

- » Ricordiamo l'esercizio sull'algoritmo risolutivo delle equazioni di primo grado
 - $a x + b = 0$
 - solo se $a \neq 0$
- » La soluzione è:
 - $x = -b / a$
 - solo se $a \neq 0$
 - $x = \text{indeterminato} (\text{infinite soluzioni})$
 - se $a=0$ e $b=0$
 - $x = \text{impossibile} (\text{nessuna soluzione})$
 - se $a=0$ e $b \neq 0$

16

Soluzione in C (1/2)

```

#include <stdio.h>
#include <stdlib.h>

int main(void)
{
    float a, b ;
    float x ;

    printf("Risoluzione eq. di primo grado\n");
    printf("Equazione: a x + b = 0\n") ;

    /* Leggi A e B */
    printf("Immetti coefficiente a: ");
    scanf("%f", &a) ;

    printf("Immetti coefficiente b: ");
    scanf("%f", &b) ;
  
```



Soluzione in C (2/2)

```

if( a != 0 )
{
    x = - b / a ;
    printf("La soluzione e' x = %f\n", x) ;
}
else
{
    if( b==0 )
    {
        printf("Equazione indeterminata\n");
    }
    else
    {
        printf("Equazione impossibile\n");
    }
}

```

Istruzioni if-else annidate

Opzionalità del ramo else

Forme abbreviate

Ricordiamo che:

- Se il ramo "vero" oppure il ramo "falso" è composto da una sola istruzione, allora le parentesi graffe {} sono opzionali
- Se il ramo "falso" non contiene istruzioni, allora la clausola else si può omettere
- Nel contesto di if annidati, queste regole possono creare una potenziale ambiguità

21

Esempio

```

if( a>0 )
{
    c = a ;
}
else
{
    if( b>0 )
    {
        c = b ;
    }
    else
    {
        c = 0 ;
    }
}

```

```

if( a>0 )
    c = a ;
else
    if( b>0 )
        c = b ;
    else
        c = 0 ;

```



22

Esempio problematico

```

if( a>0 )
if( b>0 )
c = a + b ;
else
c = 0 ;

```

```

if( a>0 )
if( b>0 )
    c = a + b ;
else
    c = 0 ;

```

?

```

if( a>0 )
if( b>0 )
    c = a + b ;
else
    c = 0 ;

```

23

- Ogni clausola else, in assenza di parentesi graffe che ne esplicitino l'attribuzione, è da intendersi riferita all'istruzione if più vicina (per la quale non sia ancora stata attribuita una clausola else)

```

if( a>0 )
if( b>0 ) ←
    c = a + b ;
else →
    c = 0 ;

```

```

if( a>0 )
{
    if( b>0 )
        c = a + b ;
    else
        c = 0 ;
}

```

24



Suggerimento

- In presenza di **if** annidate, è conveniente abbondare con le parentesi graffe

```
if( a>0 )
{
    if( b>0 )
        c = a + b ;
    else
        c = 0 ;
}
```

```
if( a>0 )
if( b>0 )
c = a + b ;
else
c = 0 ;
```

```
if( a>0 )
{
    if( b>0 )
        c = a + b ;
    else
        c = 0 ;
}
```

```
#include <stdio.h>
#include <conio.h>
#include <ctype.h>

#define MAXPAROLA 30
#define MAXSIGLA 60

int main(argc, char *argv[])
{
    int freq[MAXPAROLA]; /* *vettore di contatori delle frequenze delle lunghezze delle parole */
    int maxLMAXSIGLA; /* l'indice massimo */
    int i, maxL, maxL2;
    FILE *fp;

    fp=fopen(argv[1], "r");
    if(fp==NULL)
    {
        fprintf(stderr, "ERRORE: non sono partito da un file\n");
        exit(1);
    }

    for(i=0;i<MAXSIGLA;i++)
        freq[i]=0;

    maxLMAXSIGLA=0;
    maxL=0;
    maxL2=0;

    while(fgets(argv[1], MAXPAROLA, fp))
    {
        i=0;
        maxL=0;
        maxL2=0;
        for(i=0; argv[1][i] != '\0'; i++)
        {
            if(isupper(argv[1][i])) /* se è maiuscola */
                freq[i]++;
            if(freq[i]>freq[maxL])
                maxL=i;
            if(freq[i]>freq[maxL2])
                maxL2=i;
        }
    }

    printf("Parola con la maggiore lunghezza (%d): %s\n", maxL, argv[1]);
    printf("Parola con la seconda maggiore lunghezza (%d): %s\n", maxL2, argv[1]);
}
```

Istruzioni if-else annidate

Catene if-else if-...-else

Catene di istruzioni condizionali

- Talvolta occorre verificare, in sequenza, una serie di condizioni particolari, trovando la prima condizione vera tra quelle possibili
- Esempio:

- Dato un numero intero tra 1 e 12, che rappresenta il mese corrente, stampare il nome del mese per esteso ("Gennaio" ... "Dicembre")

27

```
if( mese == 1 )
    printf("Gennaio\n") ;
else
{
    if( mese == 2 )
        printf("Febbraio\n") ;
    else
    {
        if( mese == 3 )
            printf("Marzo\n") ;
        else
        {
            if( mese == 4 )
                printf("Aprile\n") ;
            else
            {
                .... /* continua fino a 12 */
            }
        }
    }
}
```

Soluzione

```
if( mese == 1 )
    printf("Gennaio\n") ;
else if( mese == 2 )
    printf("Febbraio\n") ;
else if( mese == 3 )
    printf("Marzo\n") ;
else if( mese == 4 )
    printf("Aprile\n") ;
else if( mese == 5 )
    printf("Maggio\n") ;
else if( mese == 6 )
    printf("Giugno\n") ;
else if( mese == 7 )
    printf("Luglio\n") ;
else if( mese == 8 )
    printf("Agosto\n") ;
else if( mese == 9 )
    printf("Settembre\n") ;
else if( mese == 10 )
    printf("Ottobre\n") ;
else if( mese == 11 )
    printf("Novembre\n") ;
else if( mese == 12 )
    printf("Dicembre\n") ;
else
    printf("MESE ERRATO!\n") ;
```

Analisi della soluzione

- Annidamento eccessivo
- Scarsa leggibilità
- Difficile identificazione delle {...} corrispondenti
- Esistono formattazioni migliori?

29

```
if( mese == 1 )
    printf("Gennaio\n") ;
else if( mese == 2 )
    printf("Febbraio\n") ;
else if( mese == 3 )
    printf("Marzo\n") ;
else if( mese == 4 )
    printf("Aprile\n") ;
else if( mese == 5 )
    printf("Maggio\n") ;
.....
else if( mese == 9 )
    printf("Settembre\n") ;
else if( mese == 10 )
    printf("Ottobre\n") ;
else if( mese == 11 )
    printf("Novembre\n") ;
else if( mese == 12 )
    printf("Dicembre\n") ;
else
    printf("MESE ERRATO!\n") ;
```

Soluzione più leggibile

In generale...

- ▶ In ogni ramo "falso" c'è una sola istruzione: la **if-else annidata**
 - anche se a sua volta contiene altre istruzioni, è comunque considerata una sola istruzione
- ▶ È possibile omettere le parentesi nel ramo **else**
- ▶ È conveniente rimuovere l'indentazione
- ▶ Ricordarsi dell'**else finale**

```
if ( c1 )
{
    A1 ;
}
else if ( c2 )
{
    A2 ;
}
else if ( c3 )
{
    A3 ;
}.....
else
{
    An ;
}
```

31

Istruzioni if-else annidate

Esercizio proposto

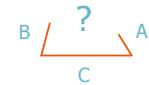
Esercizio "Classificazione triangolo 2"

- ▶ Si scriva un programma in linguaggio C che legga da tastiera i valori delle lunghezze dei tre lati di un triangolo (detti A, B e C), e determini:
 - se il triangolo è equilatero
 - se il triangolo è isoscele
 - se il triangolo è scaleno
 - se il triangolo è rettangolo
- ▶ Il programma, prima di classificare il triangolo, controlli se i numeri A, B, C rappresentano correttamente un triangolo

33

- ▶ Data una terna di numeri A, B, C, non è detto che si possa costruire un triangolo con tali lunghezze dei lati

- La lunghezza di ciascun lato deve essere un numero positivo
- Ogni lato deve essere minore della somma degli altri due
- Ogni lato deve essere maggiore della differenza degli altri due



34

Struttura proposta

```
if ( i lati non sono positivi)
{
    printf("errore") ;
}
else if (ogni lato non è minore della somma degli altri)
{
    printf("errore") ;
}
else if (ogni lato non è maggiore della differenza degli altri)
{
    printf("errore") ;
}
else
{
    /* caso normale */
    ...vedi triangolo.c
}
```

Condizioni booleane

- ▶ "i lati non sono positivi"
 - $a \leq 0 \ ||\ b \leq 0 \ ||\ c \leq 0$
- ▶ "ogni lato non è minore della somma degli altri"
 - $a >= b+c \ ||\ b >= a+c \ ||\ c >= a+b$
- ▶ "ogni lato non è maggiore della differenza degli altri"
 - attenzione: la differenza va presa in valore assoluto!
 - prima di calcolare $A-B$, occorre verificare che $A>B$, altrimenti bisogna calcolare $B-A$

36

Condizioni booleane (2)

» "ogni lato non è maggiore della differenza degli altri"

- $(b > c) \&& a \leq b - c$ ||
- $(b \leq c) \&& a \leq c - b$ ||
- $(a > c) \&& b \leq a - c$ ||
- $(a \leq c) \&& b \leq c - a$ ||
- $(a > b) \&& c \leq b - a$ ||
- $(a \leq b) \&& c \leq a - b$ ||

```
#include <stdio.h>
#include <limits.h>
#include <ctype.h>

#define MAXPAROLA 30
#define MAXSIGA 60

int main(int argc, char *argv[])
{
    int lungMAXPAROLA; /* valore di controllo
                        delle frequenze delle lunghezze delle parole */
    int i, infMax, lungMax;
    int f1, f2, lungF1, lungF2;

    if(argc != 2)
    {
        fprintf(stderr, "ERRORE: non sono stati inseriti i parametri del file\n");
        exit(1);
    }
    if((f1 = fopen(argv[1], "r")) == NULL)
    {
        fprintf(stderr, "ERRORE: impossibile aprire il file %s", argv[1]);
        exit(1);
    }

    lungMAXPAROLA = 10;
    lungMax = 0;
}
```

Istruzioni if-else annidate

Verifica della soluzione

37

Verifica "Classificazione triangolo 2"

» Analizziamo il codice dell'esercizio e verifichiamone il corretto funzionamento



```
Dev-C++ 4.9.9.2
File Edit Options Debug Project Options OS Fresh Info
m2241
1 // Soluzione proposta esercizio 2.2.A.
2
3 #include <cs51.h>
4
5 void main(void)
6 {
7     int a, b, c;
8     int ret;
9
10    /* Let's ask for three integers. */
11    printf("Please enter three integers: ");
12    scanf("%d %d %d", &a, &b, &c);
13
14    /* Compute the sum of the first two numbers. */
15    ret = a + b;
16
17    /* Check if the third number is greater than or equal to the sum of the first two. */
18    if(c >= ret)
19    {
20        /* If so, print the message. */
21        printf("The third number is greater than or equal to the sum of the first two.\n");
22    }
23 }
```

39

```
#include <stdio.h>
#include <ctype.h>
#include <string.h>

#define MAXPAROLA 30
#define MAXSIGA 80
```

Scelte ed alternative

Istruzione switch

```
#include <stdio.h>
#include <ctype.h>
#include <string.h>

#define MAXPAROLA 30
#define MAXSIGA 80

int main(int argc, char *argv[])
{
    int lungMAXPAROLA; /* valore di costante
    const char *lungMAXSIGA; /* valore di costante */
    char sigMAXSIGA;
    int i, indice, lungparola;
    FILE *fptr;

    fptr=fopen(argv[1], "r");
    if(fptr==NULL)
    {
        printf("Errore: non è possibile aprire il file %s", argv[1]);
        exit(1);
    }

    lungMAXSIGA=MAXSIGA;
    lungparola=MAXPAROLA;
    indice=0;
```

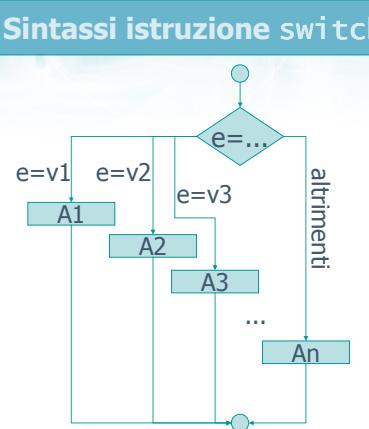
Istruzione switch

Sintassi dell'istruzione

```
switch ( e )
{
    case v1:
        A1 ;
    break ;

    case v2:
        A2 ;
    break ;

    case v3:
        A3 ;
    break ;
    .....
    default:
        An ;
}
```



5

[Merge 1c 3] Spiegazione: l'ELENCO serve un particolare caso il numero del file (n)

Istruzione switch

- ▶ Sintassi dell'istruzione
- ▶ Particolarità dell'istruzione
- ▶ Esercizio proposto
- ▶ Verifica della soluzione

2

Scelte multiple

- ▶ Quando occorre compiere una sequenza di scelte, in funzione del valore di una variabile, occorre una catena di if-else
- ▶ Lo stesso risultato si può ottenere in forma più compatta mediante l'istruzione switch

```
if( mese == 1 )
    printf("Gennaio\n") ;
else if( mese == 2 )
    printf("Febbraio\n") ;
else if( mese == 3 )
    printf("Marzo\n") ;
else if( mese == 4 )
    printf("Aprile\n") ;
else if( mese == 5 )
    printf("Maggio\n") ;
.....
else if( mese == 9 )
    printf("Settembre\n") ;
else if( mese == 10 )
    printf("Ottobre\n") ;
else if( mese == 11 )
    printf("Novembre\n") ;
else if( mese == 12 )
    printf("Dicembre\n") ;
else
    printf("MESE ERRATO!\n") ;
```

[Merge 1c 3] Spiegazione: l'ELENCO serve un particolare caso il numero del file (n)

Precisazioni (1/2)

- ▶ L'espressione e può essere una variabile oppure un'espressione aritmetica
 - Il tipo di dato deve essere int, char o enum
- ▶ Ciascun caso è identificato da una costante
 - L'espressione e viene confrontata con il valore delle costanti v1...vn
 - Il tipo di dato deve essere compatibile
- ▶ Ciascun caso è delimitato da case...break
 - Non vi sono parentesi graffe {...}

6

Precisazioni (2/2)

- I casi possono apparire in qualsiasi ordine
 - Devono essere tutti diversi
- Verrà selezionato al più un caso
- Il caso `default` viene valutato se e solo se nessuno degli altri casi è stato considerato
 - Opzionale, ma sempre consigliato

7

L'istruzione break

- Il significato di `break` è di portare l'esecuzione del programma fino al termine del costrutto `switch`
 - "Salta alla chiusa graffa": `}`
- In assenza di `break`, l'esecuzione proseguirebbe attraverso il caso successivo
 - Né il prossimo `case`, né eventuali parentesi graffe, possono fermare l'esecuzione lineare

8

Casi multipli

- Potrebbe essere necessario eseguire lo stesso codice in corrispondenza di diversi valori dell'espressione
- È possibile accomunare più casi, indicandoli consecutivamente

```
switch( ora )
{
    case 12:
        pranzo = 1 ;
        break;
    case 13:
        pranzo = 1 ;
        break;
}
```

switch(ora)
{
 case 12:
 case 13:
 pranzo = 1 ;
 break;
}

9



Istruzione switch

Particolarità dell'istruzione



Esempio

```
switch( mese )
{
    case 1:
        printf("Gennaio\n") ;
        break ;
    case 2:
        printf("Febbraio\n") ;
        break ;
    case 3:
        printf("Marzo\n") ;
        break ;
    case 4:
        printf("Aprile\n") ;
        break ;
    .....
    case 12:
        printf("Dicembre\n") ;
        break ;
    default:
        printf("MESE ERRATO!\n") ;
}
```

- L'istruzione `switch` è anomala sotto diversi punti di vista:
 - Non utilizza le parentesi graffe per l'annidamento delle istruzioni interne
 - Prevede solamente il controllo di uguaglianza `e==v`
 - Richiede che i valori da confrontare siano costanti
 - `break` e `default` sono opzionali
- Occorre una forte disciplina nell'utilizzarla correttamente!

12



Errore frequente

- È errato dimenticare l'istruzione break al termine di ogni caso

viene interpretato come

```
switch( ora )
{
    case 12:
        pranzo = 1 ;
        cena = 1 ;
        break;
    case 20:
        cena = 1 ;
        break;
}
```

```
switch( ora )
{
    case 12:
        pranzo = 1 ;
    case 20:
        cena = 1 ;
        break;
}
```

forma corretta

```
switch( ora )
{
    case 12:
        pranzo = 1 ;
        break;
    case 20:
        cena = 1 ;
        break;
}
```



Errore frequente

- È errato utilizzare variabili come valori dei singoli case
- Se non è possibile usare costanti, allora utilizzare delle catene di if-else

forma corretta

```
switch( ora )
{
    case orapranzo:
        pranzo = 1 ;
        break;
    case oracena:
        cena = 1 ;
        break;
}
```

```
if( ora==orapranzo )
{
    pranzo = 1 ;
}
else if( ora==oracena )
{
    cena = 1 ;
}
```

Errore frequente

- È errato pensare di poter fare confronti di ordine, o confronti multipli
- Utilizzare sequenze di if-else
- Non usare else if se i casi non sono mutuamente esclusivi

forma corretta

```
switch( ora )
{
    case <12:
        mattino = 1 ;
        break;
    case 12 || 20:
        pasti = 1 ;
        break;
}
```

```
if( ora<12 )
{
    mattino = 1 ;
}
if( ora==12 || 20 )
{
    pasti = 1 ;
}
```

Suggerimento

- Utilizzare sempre l'istruzione default
- Anche se non vi è ragione apparente, è opportuno che il programma intercetti valori errati della variabile

```
switch( ora )
{
    .....
    default:
        printf("Errore: valore inatteso
                %d per la variabile ora\n", ora) ;
}
```

16

Suggerimento

- Posizionare l'istruzione default come ultimo caso dello switch
 - Potrebbe stare ovunque
 - Maggiore leggibilità
- L'istruzione break finale si può omettere

```
#include <iostream.h>
#include <string.h>
#include <ctype.h>

#define MAXPAROLA 30
#define MAXIGRA 80

int main(int argc, char *argv[])
{
    int freqMAXPAROLA; /* vettore di costanti
                        delle frequenze complessive delle parole */
    char c, AXIGRA;
    int i, indice, lunghezza;
    FILE *f;

    freqMAXPAROLA = freqMAXIGRA = 0;

    f = fopen(argv[1], "r");
    if (f == NULL)
    {
        cout << "ERRORE: impossibile aprire il file " << argv[1] << endl;
        exit(1);
    }

    while ((c = fgetc(f)) != EOF)
    {
        if (isalpha(c))
        {
            if (c >= 'A' && c <= 'Z')
                c = c - 'A' + 'a';
            else if (c >= 'a' && c <= 'z')
                c = c + 'A' - 'a';

            if (c == ' ')
            {
                if (freqMAXIGRA > freqMAXPAROLA)
                    freqMAXIGRA++;
                else if (freqMAXIGRA == freqMAXPAROLA)
                    freqMAXIGRA++;
            }
            else
            {
                if (freqMAXPAROLA > freqMAXIGRA)
                    freqMAXPAROLA++;
                else if (freqMAXPAROLA == freqMAXIGRA)
                    freqMAXPAROLA++;
            }
        }
    }

    cout << freqMAXIGRA << endl;
    cout << freqMAXPAROLA << endl;
}
```

Istruzione switch

Esercizio proposto

Esercizio "Semplice calcolatrice"

- Si scriva un programma in linguaggio C che implementi una semplice calcolatrice in grado di compiere le 4 operazioni (+ - × ÷) tra numeri interi
- Il programma presenterà un semplice menù da cui l'utente indicherà (con un numero tra 1 e 4) l'operazione da svolgere
- In seguito il programma acquisirà da tastiera i due operandi e stamperà il risultato dell'operazione

19

Soluzione (1/4)

```
#include <stdio.h>
#include <stdlib.h>

int main(void)
{
    int op ;
    int a, b, c ;
    int err ;

    printf("Semplice calcolatrice\n\n") ;
    printf("Inserisci 1 per la somma\n");
    printf("Inserisci 2 per la sottrazione\n");
    printf("Inserisci 3 per la moltiplicazione\n");
    printf("Inserisci 4 per la divisione\n");

    printf("La tua scelta:") ;
    scanf("%d", &op) ;
```



Soluzione (2/4)

```
printf("Inserisci il primo operando: ") ;
scanf("%d", &a) ;

printf("Inserisci il secondo operando: ") ;
scanf("%d", &b) ;
```

21

Soluzione (3/4)

```
err = 0 ;
switch( op )
{
    case 1:
        c = a + b ;
        break ;
    case 2:
        c = a - b ;
        break ;
    case 3:
        c = a * b ;
        break ;
    case 4:
        c = a / b ;
        break ;
    default:
        printf("Operazione errata\n") ;
        err = 1 ;
}
```

Soluzione (3/4)

```
err = 0 ;
switch( op )
{
    case 4:
        if( b == 0 )
        {
            printf("Divisione per zero!\n");
            err = 1 ;
        }
    case 2:
        c =
    case 3:
        c =
    case 4:
        c = a / b ;
        break ;
    default:
        printf("Operazione errata\n") ;
        err = 1 ;
}
```

Soluzione (4/4)

```
if( err == 0 )
{
    printf("Il risultato vale: %d\n", c) ;
}
```

24

```
#include <stdio.h>
#include <ctype.h>

#define MAXPVAROLA 30
#define MAXRG 80

int main(int argc, char *argv[])
{
    int freqMAXPVAROLA; /* valore di confronto delle frequenze delle parole */
    int i, min, lunghezza;
    FILE *f;
    int c;
    int flag=0;
    char s[100];
    if(argc > 1) {
        freqMAXPVAROLA = atoi(argv[1]);
        if(freqMAXPVAROLA < 0) {
            fprintf(stderr, "ERRORE: impossibile aprire il file %s", argv[1]);
            exit(1);
        }
    } else {
        freqMAXPVAROLA = 10;
    }
    if((f=fopen(argv[0], "r")) == NULL) {
        fprintf(stderr, "ERRORE: non è possibile aprire il file %s", argv[0]);
        exit(1);
    }
    /* legge le parole dalla riga */
    while(fgets(s, 100, f) != NULL) {
        /* rimuove gli spazi iniziali della riga */
        while((c=fgetc(f)) == ' ' || c == '\t') {
            fseek(f, -1, SEEK_CUR);
        }
        /* legge la parola */
        min = 0;
        lunghezza = 0;
        for(i=0; s[i] != ' ' && s[i] != '\n'; i++) {
            lunghezza++;
        }
        /* se la parola è più corta della parola di confronto */
        if(lunghezza < freqMAXPVAROLA) {
            /* si legge la parola */
            if(flag) {
                printf(" %s", s);
            } else {
                printf("%s", s);
            }
            /* si controlla se la parola è già stata stampata */
            if(flag) {
                /* si legge la parola */
                if(strcmp(s, s1) == 0) {
                    /* se la parola è già stata stampata */
                    /* si controlla se la parola è più corta della parola di confronto */
                    if(lunghezza < freqMAXPVAROLA) {
                        /* si legge la parola */
                        if(flag) {
                            printf(" %s", s);
                        } else {
                            printf("%s", s);
                        }
                    }
                }
            }
            /* si salva la parola per confrontarla con le successive */
            s1 = s;
            flag = 1;
        }
    }
    /* si chiude il file */
    fclose(f);
}
```

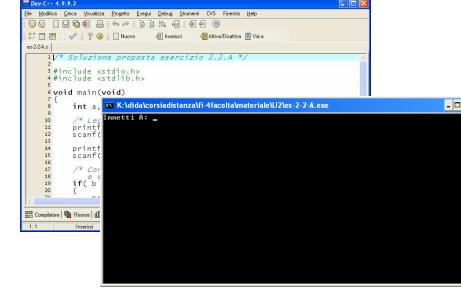
Istruzione switch

Verifica della soluzione

Verifica "Semplice calcolatrice"

Analizziamo il codice dell'esercizio e verifichiamone il corretto funzionamento


calcola.c



- ▶ Esercizi sul calcolo del massimo
- ▶ Esercizio "Equazione di secondo grado"
- ▶ Esercizio "Re e Regina"

Scelte ed alternative

Esercizi proposti

2

```
#include <stdio.h>
#include <math.h>
#include <ctype.h>
```

```
#define MAXPAROLA 30
#define MAXIGRA 80
```

```
int main(int argc, char *argv[])
{
    int max(MAXPAROLA); /* valore di controllo delle frequenze delle lunghezze delle parole */
    char maxIGRA; /* carattere che rappresenta il numero del max (A o B) */
    int i;
}
```

```
for(i=0; i<MAXPAROLA; i++)
    max=0;
```

```
if(argc > 1)
    for(i=0; i<argc-1; i++)
        if(isalpha(argv[i])) /* se è un carattere alfanumerico */
            max=1;
    else if(argv[i] == 'A') /* se è una parola A */
        max=1;
    else if(argv[i] == 'B') /* se è una parola B */
        max=0;
    else
        fprintf(stderr, "ERRORE, imponendo opere il file %s", argv[1]);
}
```

```
while(fgets(diga, MAXIGRA, f) != NULL)
```

Esercizi proposti

Esercizi sul calcolo del massimo

4

- ▶ Si scriva un programma in linguaggio C che acquisisca due numeri interi da tastiera e:
 - determini, stampando un messaggio opportuno quale dei due numeri (il primo o il secondo) sia maggiore
 - stampi il valore di tale numero
- ▶ Si trascuri il caso in cui i due numeri siano uguali

Esercizio "Calcolo del massimo"

- ▶ Chiamiamo A e B i due numeri introdotti dall'utente
- ▶ Chiamiamo MAX il valore del maggiore tra i due
- ▶ Occorre verificare la condizione A>B
 - Se A>B, allora MAX sarà pari ad A
 - Altrimenti, MAX sarà pari a B

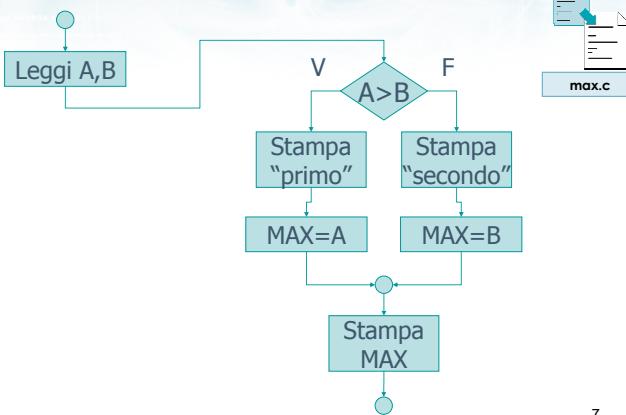
Analisi

5

6

```
PS: Prompt dei comandi
CALCOLO DEL MASSIMO TRA DUE NUMERI
Inserisci il primo numero: 10
Inserisci il secondo numero: -3
Il maggiore tra i numeri inseriti e' il primo
Il valore del numero maggiore e' 10
```

Diagramma di flusso



7

Esercizio "Calcolo del massimo a 3"

- Si scriva un programma in linguaggio C che acquisisca **tre** numeri interi da tastiera e:
 - determini, stampando un messaggio opportuno quale dei **tre** numeri (il primo, il secondo o il terzo) sia maggiore
 - stampi il valore di tale numero
- Si trascuri il caso in cui i numeri siano uguali

8

Esempio

```

$ Prompt dei comandi: max3.c
CALCOLO DEL MASSIMO TRA TRE NUMERI
Inserisci il primo numero: 10
Inserisci il secondo numero: -3
Inserisci il terzo numero: 15

Il maggiore tra i numeri inseriti e' il terzo
Il valore del numero maggiore e' 15

```

9

Analisi

- Chiamiamo A, B, C i tre numeri inseriti
- Si può procedere in due modi
 - Usando istruzioni **if annidate**

• se $A > B$, allora controlla se $A > C \dots$



10

Analisi

- Chiamiamo A, B, C i tre numeri inseriti
- Si può procedere in due modi

- Usando istruzioni **if annidate**
 - se $A > B$, allora controlla se $A > C \dots$



- Usando espressioni condizionali complesse
 - se $A > B$ e $A > C \dots$



11

```

#include <stdio.h>
#include <string.h>
#include <ctype.h>

#define MAXPAROLA 30
#define MAXIGRA 80

int main(int argc, char *argv[])
{
    int freqMAXPAROLA; /* variabile di controllo
                        delle frequenze delle parole */
    char *parolaAXIGRA;
    int i, indice, lunghezza;
    FILE *f;

    f=fopen(argv[1], "r");
    if(f==NULL)
        exit(1);
    freqMAXPAROLA=0;
    parolaAXIGRA=(char *)malloc(sizeof(char)*MAXIGRA);
    if(parolaAXIGRA==NULL)
        exit(2);
    strcpy(parolaAXIGRA, " ");
    lunghezza=0;
    for(i=0; i<MAXIGRA; i++)
        parolaAXIGRA[i]=0;
    while(fgets(parolaAXIGRA, MAXIGRA, f))
    {
        lunghezza=strlen(parolaAXIGRA);
        if(freqMAXPAROLA<lunghezza)
            freqMAXPAROLA=lunghezza;
        else if(freqMAXPAROLA==lunghezza)
            freqMAXIGRA++;
    }
    printf("Parola con la maggiore lunghezza: %s\n", parolaAXIGRA);
    free(parolaAXIGRA);
    fclose(f);
}

```

Esercizi proposti

Esercizio "Equazione di secondo grado"

Esercizio "Equazione di secondo grado"

- » Data l'equazione

- $a x^2 + b x + c = 0$

con a , b e c inseriti da tastiera, determinare il valore (o i valori) di x che risolvono l'equazione

13

Analisi

- » Ricordiamo la **formula risolutiva** per le equazioni di secondo grado

$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

- » La quantità sotto radice è il **discriminante** Δ

- » Vi sono vari casi possibili

- se $a \neq 0$ o $a=0$
- se $\Delta > 0$, $\Delta=0$ o $\Delta < 0$

14

Casi possibili

Caso	Situazione	Soluzione/i
$a = 0$	Equazione di primo grado	$x = -c/b$ Impossibile se $b=0, c \neq 0$ Indeterminata se $b=0, c=0$
$a \neq 0$ $\Delta > 0$	Due soluzioni reali distinte	$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$
$a \neq 0$ $\Delta = 0$	Due soluzioni reali coincidenti	$x_1 = x_2 = \frac{-b}{2a}$
$a \neq 0$ $\Delta < 0$	Due soluzioni complesse coniugate	$x_{1,2} = R \pm iC$ $R = -b/2a, C = \sqrt{ b^2 - 4ac }/2a$

15

Soluzione



secondogradoc.c

- » Acquisire a , b , c
- » Se $a=0$, risolvere l'equazione di primo grado
 - Ri-usiare il codice già scritto, facendo attenzione al nome delle variabili
- » Calcolare il discriminante Δ
 - Il nome della variabile sarà delta
- » In funzione del segno di delta, usare la formula opportuna

16

```
#include <iostream.h>
#include <string.h>
#include <ctype.h>

#define MAXPAROLA 30
#define MAXIGRA 80

int main(int argc, char *argv[])
{
    int largMAXPAROLA; /* vertice di confronto
    per trovare la lunghezza delle parole */
    char parolaAXIGRA[ ];
    int i, indice, lunghezza;
    FILE *f;

    if(argc<2)
        cout << "Inserire un file da leggere" << endl;
    else
    {
        f=fopen(argv[1], "r");
        if(f==NULL)
            cout << "File non trovato" << endl;
        else
        {
            cout << "File aperto" << endl;
            cout << "Ricerca parola" << endl;
            cout << "Inserire la parola da cercare dal file" << endl;
            cin >> parolaAXIGRA;
            cout << "Parola cercata: " << parolaAXIGRA << endl;
            cout << "Ricerca in corso..." << endl;
            cout << "Ricerca terminata" << endl;
            cout << "La parola cercata è stata trovata" << endl;
            cout << "Ricerca terminata" << endl;
        }
    }
    cout << "File chiuso" << endl;
    return 0;
}
```

Esercizi proposti

Esercizio "Re e Regina"

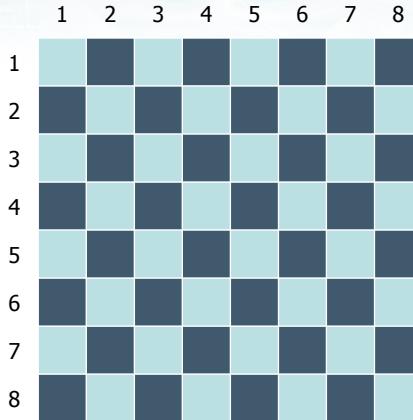
Il programma deve leggere da un file di testo la posizione del Re e della Regina su una scacchiera 8x8. Il file contiene due righe, la prima con la posizione del Re (ad esempio "B7") e la seconda con la posizione della Regina (ad esempio "D5"). Il programma deve determinare se la Regina è in posizione tale da poter mangiare il Re.

Esercizio "Re e Regina"

- » Su una scacchiera 8x8 sono posizionati due pezzi: il Re bianco e la Regina nera
- » Si scriva un programma in linguaggio C che, acquisite le posizioni del Re e della Regina, determini se la Regina è in posizione tale da poter mangiare il Re
 - Le posizioni dei due pezzi sono identificate mediante la riga e la colonna su cui si trovano, espresse come numeri interi tra 1 e 8

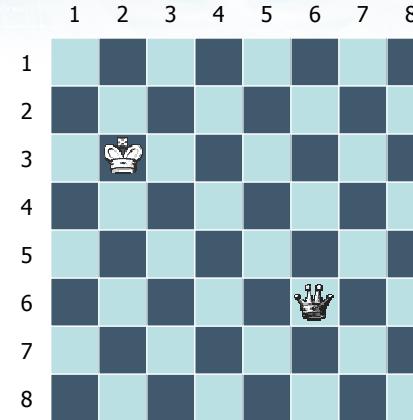
18

Analisi



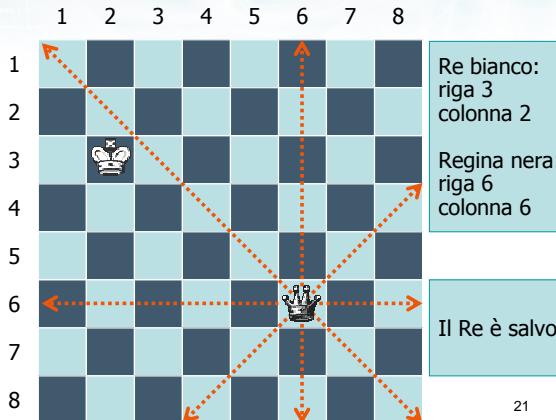
19

Analisi



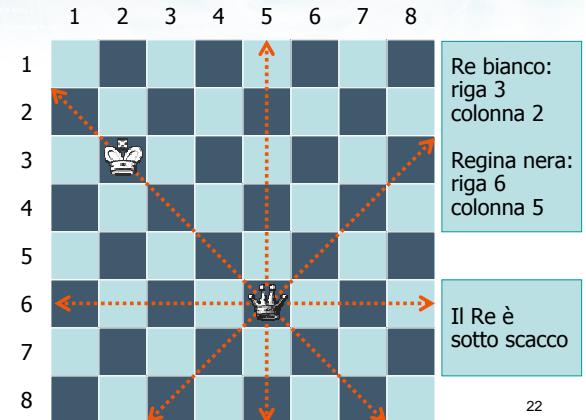
20

Analisi



21

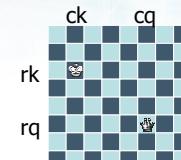
Analisi



22

Soluzione (1/2)

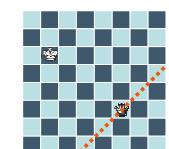
- » Acquisire le coordinate dei pezzi
 - Re: rk, ck
 - Regina: rq, cq
- » Controllare se la Regina
 - è sulla stessa riga del Re
 - $rq == rk$
 - è sulla stessa colonna del Re
 - $cq == ck$
 - è sulla stessa diagonale discendente del Re
 - è sulla stessa diagonale ascendente del Re



23

Soluzione (1/2) - Diagonali

- » Diagonale discendente
 - $r-c$ costante
 - $rq-cq == rk-ck$
- » Diagonale ascendente
 - $r+c$ costante
 - $rq+cq == rk+cq$



24

Soluzione (2/2)

- » Conviene utilizzare una variabile logica scacco
 - **inizializzare** scacco=0
 - fare i vari tipi di controlli
 - se si verifica una condizione di scacco, porre scacco=1
- » Al termine dei controlli, in funzione del valore di scacco, stampare il messaggio opportuno
 - se scacco==0, il Re è salvo
 - se scacco==1, il Re è sotto scacco



scacco.c

- Ramificazione del flusso di esecuzione
- Istruzione if-else
- Condizioni Booleane semplici e complesse
- Annidamento di istruzioni if-else
- Istruzione switch

Scelte ed alternative

Sommario

2

Tecniche di programmazione

- Catene di istruzioni if-else if-...-else
- Annidamento delle istruzioni o condizioni Booleane complesse
- Uso di variabili logiche per tenere traccia delle condizioni incontrate
- Istruzione switch per sostituire alcuni tipi di catene if-else if
- Uso di else e default per catturare condizioni anomale

3

Suggerimenti

- Analizzare sempre tutti i casi possibili prima di iniziare a scrivere il programma
- Abbondare con le parentesi graffe
- Curare l'indentazione
- Aggiungere commenti in corrispondenza della clausola else e della graffa di chiusura

4

Materiale aggiuntivo

- Sul CD-ROM
 - Testi e soluzioni degli esercizi trattati nei lucidi
 - Scheda sintetica
 - Esercizi risolti
 - Esercizi proposti
- Esercizi proposti da altri libri di testo

5

```
#include <stdio.h>
#include <ctype.h>
#define MAXPOLA 30
#define MAXSIGA 60

int main(int argc, char *argv[])
{
    int freqMAXPOLA; /* valore di confronto delle frequenze delle piazze */
    int freqSIGA; /* valore di confronto delle frequenze delle sigle */
    int i, inicio, longitud;
    FILE *f;

    inicio = longitud = 0;
    freqSIGA = MAXSIGA;
    freqMAXPOLA = MAXPOLA;
    if (argc > 1)
        freqSIGA = atoi(argv[1]);
    if (argc > 2)
        freqMAXPOLA = atoi(argv[2]);
    f = fopen("input.txt", "r");
    if (f == NULL)
    {
        fprintf(stderr, "ERRORE: impossibile aprire il file %s\n", argv[1]);
        exit(1);
    }
    while (fscanf(f, "%d", &inicio) != EOF)
    {
        fscanf(f, "%d", &longitud);
        if (freqSIGA <= inicio)
            longitud += freqSIGA;
        else
            longitud -= freqSIGA;
        if (freqMAXPOLA <= longitud)
            longitud += freqMAXPOLA;
        else
            longitud -= freqMAXPOLA;
        printf("%d\n", longitud);
    }
    fclose(f);
}
```

Programmazione in C

Unità Cicli ed iterazioni

2

- » La ripetizione
- » Istruzione while
- » Schemi ricorrenti nei cicli
- » Istruzione for
- » Approfondimenti
- » Esercizi proposti
- » Sommario

Riferimenti al materiale

» Testi

- Kernighan & Ritchie: capitolo 3
- Cabodi, Quer, Sonza Reorda: capitolo 4
- Dietel & Dietel: capitolo 4

» Dispense

- Scheda: "Cicli ed iterazioni in C"

3

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

#define MAXPAROLA 30
#define MAXSIGA 80

int main(int argc, char *argv[])
{
    int lung(MAXPAROLA); /* valore di controllo delle frequenze delle parole */
    char c[MAXSIGA]; /* riga che contiene una parola */
    char sig(MAXSIGA);
    int i, indice, lunghezza;
    FILE *fp;

    fp=fopen(argv[1], "r");
    if(fp==NULL)
    {
        printf("ERRORE, non è possibile aprire il file %s", argv[1]);
        exit(1);
    }

    fgets(c, MAXSIGA, fp);
    i=0;
    lunghezza=0;
    while(fgets(c, MAXSIGA, fp)!=NULL)
    {
        if(c[lunghezza]=='.')
            break;
        else
        {
            if(isalpha(c[lunghezza]))
                c[lunghezza]=toupper(c[lunghezza]);
            else
                c[lunghezza]=space(c[lunghezza]);
        }
        lunghezza++;
    }
    c[lunghezza]=0;
    fclose(fp);
}
```

Cicli ed iterazioni

La ripetizione

5

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

#define MAXPAROLA 30
#define MAXSIGA 80

int main(int argc, char *argv[])
{
    int lung(MAXPAROLA); /* valore di controllo delle frequenze delle parole */
    char c[MAXSIGA];
    int i, indice, lunghezza;
    FILE *fp;

    fp=fopen(argv[1], "r");
    if(fp==NULL)
    {
        printf("ERRORE, non è possibile aprire il file %s", argv[1]);
        exit(1);
    }

    fgets(c, MAXSIGA, fp);
    i=0;
    lunghezza=0;
    while(fgets(c, MAXSIGA, fp)!=NULL)
    {
        if(c[lunghezza]=='.')
            break;
        else
        {
            if(isalpha(c[lunghezza]))
                c[lunghezza]=toupper(c[lunghezza]);
            else
                c[lunghezza]=space(c[lunghezza]);
        }
        lunghezza++;
    }
    c[lunghezza]=0;
    fclose(fp);
}
```

La ripetizione

Concetto di ciclo

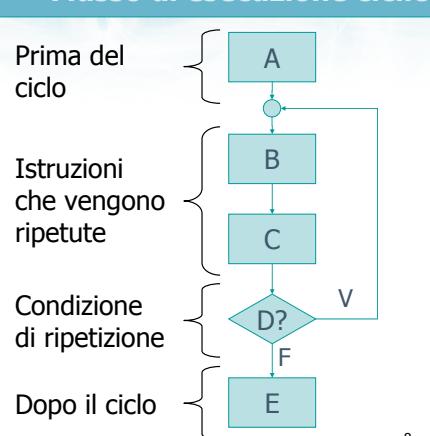
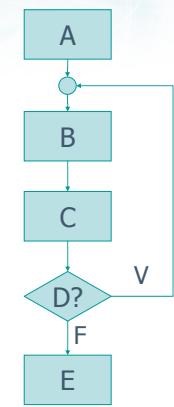
7



- ▶ Concetto di ciclo
- ▶ Struttura di un ciclo
- ▶ Numero di iterazioni note
- ▶ Numero di iterazioni ignote



- ▶ È spesso utile poter **ripetere** alcune parti del programma più volte
- ▶ Nel diagramma di flusso, corrisponde a "tornare indietro" ad un blocco precedente
- ▶ Solitamente la ripetizione è controllata da una condizione booleana



8



- ▶ Ogni ciclo porta in sé il rischio di un grave errore di programmazione: il fatto che il ciclo venga ripetuto indefinitamente, senza mai uscire
- ▶ Il programmatore deve garantire che ogni ciclo, dopo un certo numero di iterazioni, venga terminato
 - La condizione booleana di controllo dell'iterazione **deve** divenire falsa



Errore frequente

9

Istruzioni eseguibili ed eseguite

Istruzioni eseguibili

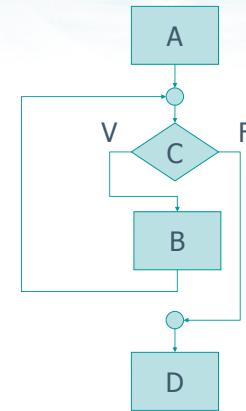
- Le istruzioni che fanno parte del programma
- Corrispondono alle istruzioni del sorgente C

Istruzioni eseguite

- Le istruzioni effettivamente eseguite durante una specifica esecuzione del programma
- Dipendono dai dati inseriti
- Nel caso di scelte, alcune istruzioni eseguibili non verranno eseguite
- Nel caso di cicli, alcune istruzioni eseguibili verranno eseguite varie volte

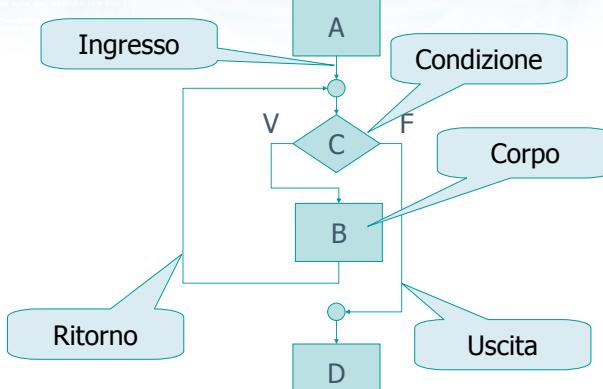
10

Notazione grafica (while)



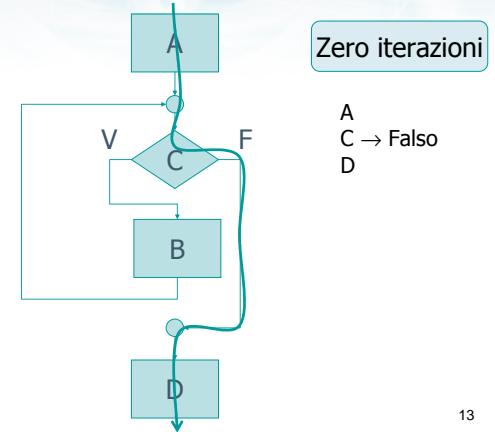
11

Notazione grafica (while)



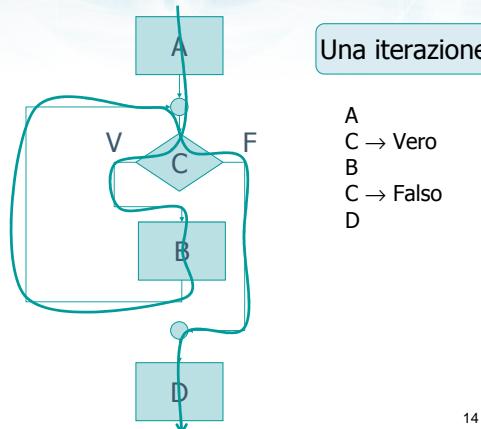
12

Flussi di esecuzione



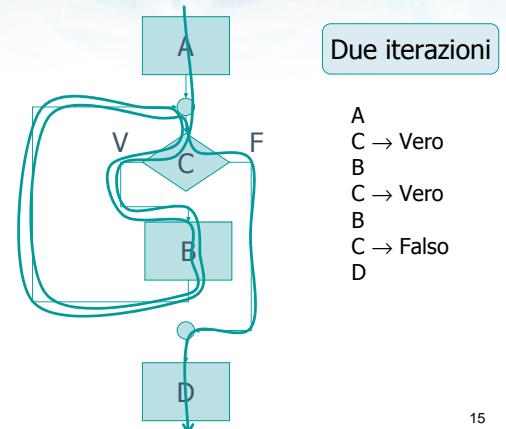
13

Flussi di esecuzione



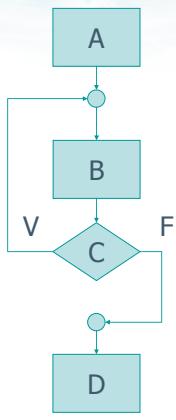
14

Flussi di esecuzione



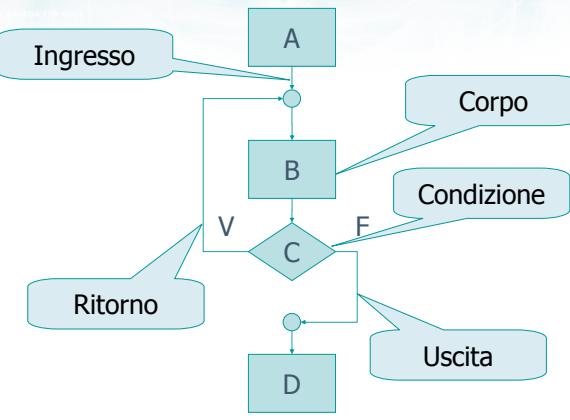
15

Notazione grafica (do-while)



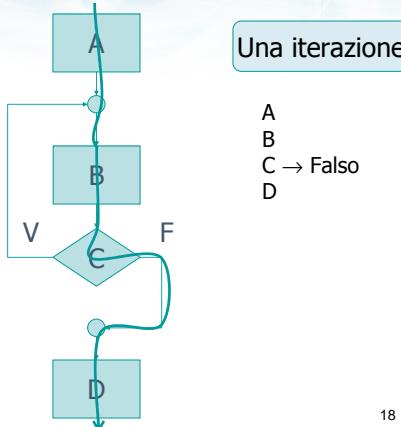
16

Notazione grafica (do-while)



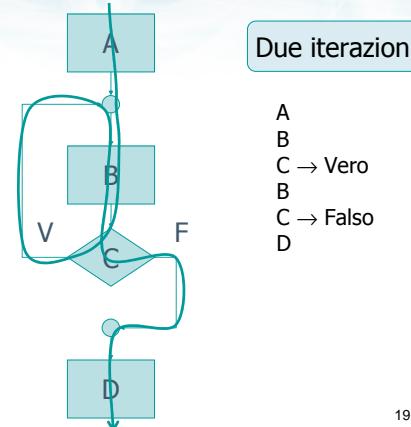
17

Flussi di esecuzione



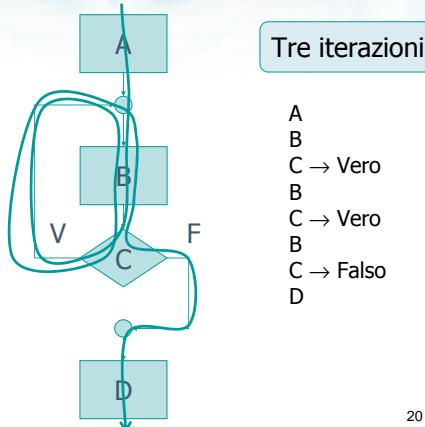
18

Flussi di esecuzione



19

Flussi di esecuzione



20

```

#include <iostream.h>
#include <string.h>
#include <ctype.h>

#define MAXPAROLA 30
#define MAXIGRA 80

int maxParolaAng, char *ang[40];
int freqMAXPAROLA ; /* numero di caratteri della frequenza delle parole */
char *angMAXIGRA ;
int i, indice, lunghezza ;
FILE *f ;

for(i=0; i<MAXPAROLA; i++)
    maxParolaAng[i]=0;

// Funzione che legge un file con le parole e le inserisce nel file ang
void leggiAng()
{
    FILE *f;
    f=fopen("ang.txt", "r");
    if(f==NULL)
        cout<<"ERRORE: impossibile aprire il file ang.txt";
    else
        while(fgets(ang[i], MAXIGRA, f)!=NULL)
            i++;
    fclose(f);
}

void leggiAng()
{
    FILE *f;
    f=fopen("ang.txt", "r");
    if(f==NULL)
        cout<<"ERRORE: impossibile aprire il file ang.txt";
    else
        while(fgets(ang[i], MAXIGRA, f)!=NULL)
            i++;
    fclose(f);
}
  
```

La ripetizione

Struttura di un ciclo

Problemi

- Nello strutturare un ciclo occorre garantire:
 - Che il ciclo possa terminare
 - Che il numero di iterazioni sia quello desiderato
- Il corpo centrale del ciclo può venire eseguito più volte:
 - La prima volta lavorerà con variabili che sono state inizializzate al di fuori del ciclo
 - Le volte successive lavorerà con variabili che possono essere state modificare nell'iterazione precedente
 - Garantire la correttezza sia della prima, che delle altre iterazioni

22

Anatomia di un ciclo (1/5)

- Conviene concepire il ciclo come 4 fasi
 - Inizializzazione
 - Condizione di ripetizione
 - Corpo
 - Aggiornamento

23

Anatomia di un ciclo (2/5)

- Conviene concepire il ciclo come 4 fasi
 - Inizializzazione
 - Assegnazione del valore iniziale a tutte le variabili che vengono lette durante il ciclo (nel corpo o nella condizione)
 - Condizione di ripetizione
 - Corpo
 - Aggiornamento

24

Anatomia di un ciclo (3/5)

- Conviene concepire il ciclo come 4 fasi
 - Inizializzazione
 - Condizione di ripetizione
 - Condizione, di solito inizialmente vera, che al termine del ciclo diventerà falsa
 - Deve dipendere da variabili che saranno modificate all'interno del ciclo (nel corpo o nell'aggiornamento)
 - Corpo
 - Aggiornamento

25

Anatomia di un ciclo (4/5)

- Conviene concepire il ciclo come 4 fasi
 - Inizializzazione
 - Condizione di ripetizione
 - Corpo
 - Le istruzioni che effettivamente occorre ripetere
 - Sono lo scopo per cui il ciclo viene realizzato
 - Posso usare le variabili inizializzate
 - Posso modificare le variabili
 - Aggiornamento

26

Anatomia di un ciclo (5/5)

- Conviene concepire il ciclo come 4 fasi
 - Inizializzazione
 - Condizione di ripetizione
 - Corpo
 - Aggiornamento
 - Modifica di una o più variabili in grado di aggiornare il valore della condizione di ripetizione
 - Tengono "traccia" del progresso dell'iterazione

27

```
#include <stdio.h>
#include <math.h>
#include <string.h>

#define MAXPARIOLA 30
#define MAXRGA 80

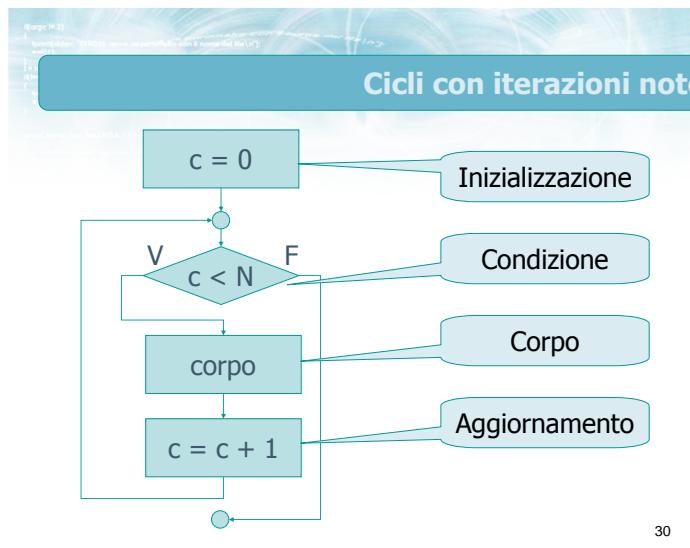
int main(int argc, char *argv[])
{
    int freqMAXPARIOLA; /* valori di confronto delle frequenze delle parola */
    int freqMAXRGA; /* valori di confronto delle lunghezze delle parole */
    int i, indice, lunghezza;
    FILE *fp;
    fp=fopen(argv[1], "r");
    if(fp==NULL)
    {
        fprintf(stderr, "ERRORE: non è possibile aprire il file %s", argv[1]);
        exit(1);
    }
    fscanf(fp, "%d", &freqMAXPARIOLA);
    fscanf(fp, "%d", &freqMAXRGA);
    fscanf(fp, "%d", &l lunghezza);
    fscanf(fp, "%s", &indice);
    if(lunghezza<1)
    {
        fprintf(stderr, "ERRORE: imprecisione operativa %s", argv[1]);
        exit(1);
    }
    if(freqMAXPARIOLA>MAXPARIOLA || freqMAXRGA>MAXRGA || l>lunghezza)
    {
        fprintf(stderr, "ERRORE: valori di confronto o lunghezza superano i limiti del file %s", argv[1]);
        exit(1);
    }
    if(freqMAXPARIOLA==0 || freqMAXRGA==0)
    {
        fprintf(stderr, "ERRORE: valori di confronto sono nulli nel file %s", argv[1]);
        exit(1);
    }
}
```

La ripetizione

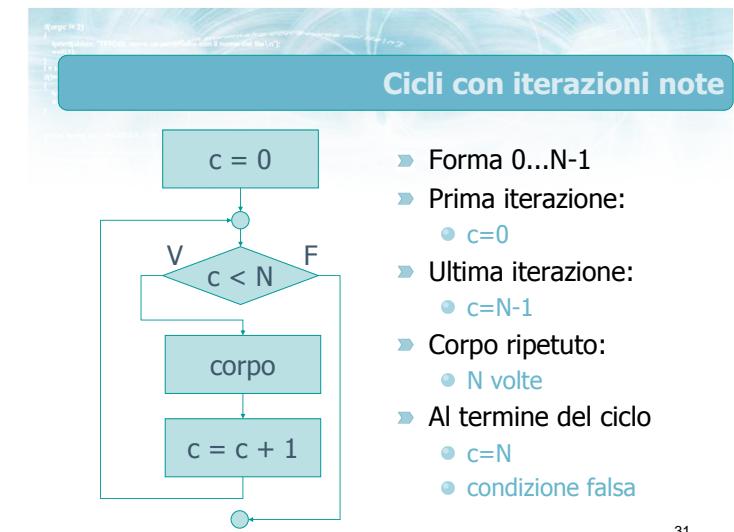
Numero di iterazioni note

- Cicli in cui il numero di iterazioni sia noto a priori, ossia prima di entrare nel ciclo stesso
 - Solitamente si usa una variabile "contatore"
 - L'aggiornamento consiste in un incremento o decremento della variabile
- Cicli in cui il numero di iterazioni non sia noto a priori, ma dipenda dai dati elaborati nel ciclo
 - Solitamente si usa una condizione dipendente da una variabile letta da tastiera oppure calcolata nel corpo del ciclo
 - Difficile distinguere il corpo dall'aggiornamento
 - Problema di inizializzazione

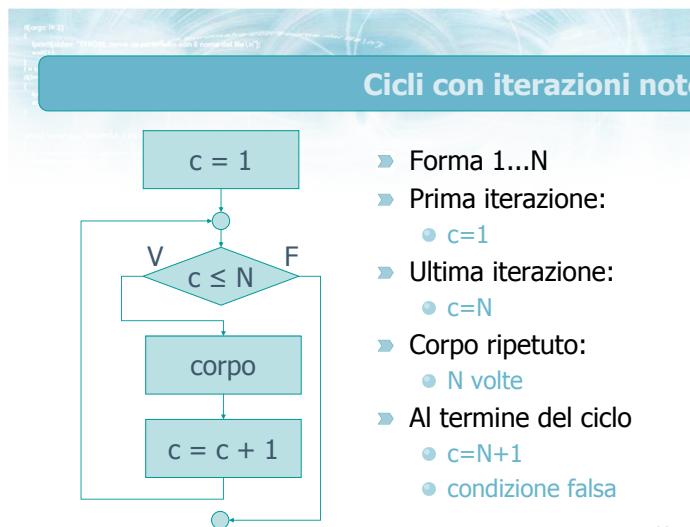
29



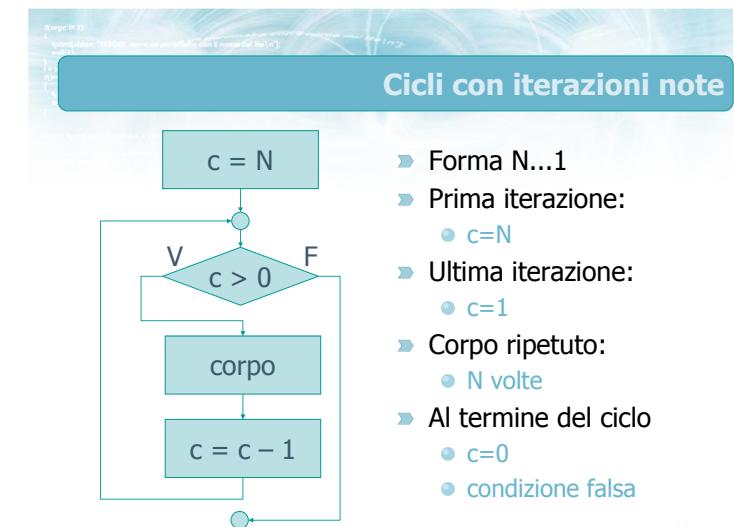
30



31

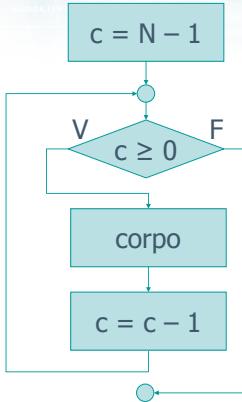


32



33

Cicli con iterazioni note



- » Forma N-1...0
- » Prima iterazione:
 - $c=N-1$
- » Ultima iterazione:
 - $c=0$
- » Corpo ripetuto:
 - N volte
- » Al termine del ciclo:
 - $c=-1$
 - condizione falsa

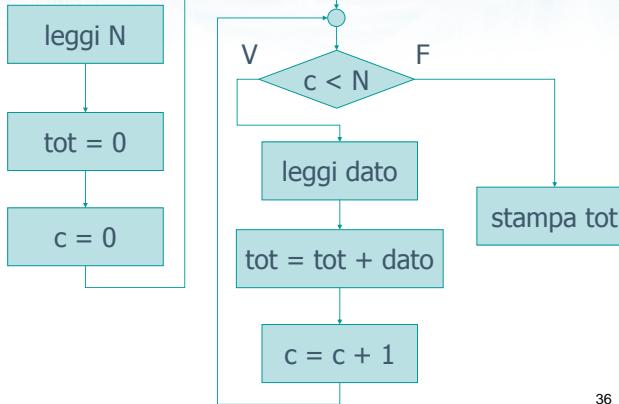
34

Esempio

- » Acquisire da tastiera una sequenza di numeri interi e stamparne la somma.
- » Il programma
 - inizialmente chiede all'utente quanti numeri intende inserire
 - in seguito richiede uno ad uno i dati
 - infine stampa la somma

35

Soluzione



36

#include <iostream.h>

#include <string.h>

#include <ctype.h>

#define MAXPAROLA 30

#define MAXIGRA 80

int main(int argc, char *argv[])
{
 int lungMAXPAROLA; /* /> valore di costante
 const int lungMAXIGRA; /* /> valore di costante
 char riga[MAXIGRA];
 int riga, lungMAX;
 riga[0] = '\0';

 for(0; riga[MAXIGRA-1] != '\0';)
 {
 if((riga[lungMAX] = getchar()) == '\n')
 {
 if(riga[lungMAX] == '\r')
 riga[lungMAX] = '\0';
 else
 riga[lungMAX] = '\n';
 }
 else
 riga[lungMAX] = riga[lungMAX-1];
 lungMAX++;
 }
 if(argc > 1)
 {
 if(strcmp(argv[1], "TOTALE") == 0)
 {
 if(lungMAX == 0)
 cout << "Nessun carattere inserito" << endl;
 else
 cout << "La somma è " << tot << endl;
 }
 else
 cout << "Argomento scorretto" << endl;
 }
 while((riga[lungMAX] != '\0') && (lungMAX < MAXIGRA));
}

La ripetizione

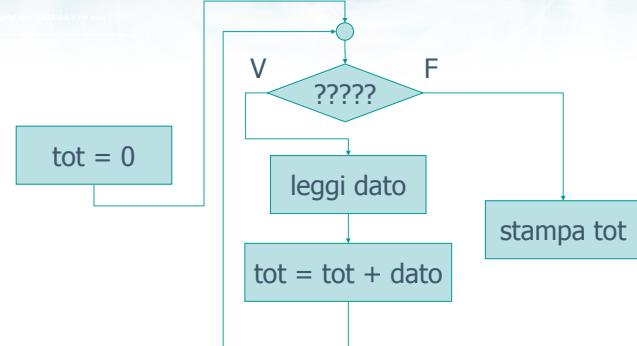
Numero di iterazioni ignote

Cicli con iterazioni ignote

- » Non esiste uno schema generale
- » Esempio:
 - Acquisire da tastiera una sequenza di numeri interi e stamparne la somma.
 - Il termine della sequenza viene indicato inserendo un dato pari a zero.

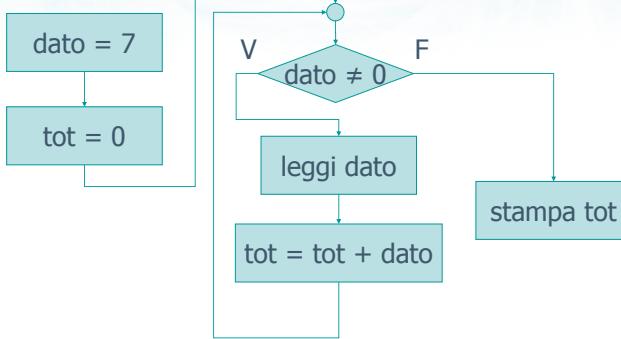
38

Soluzione parziale



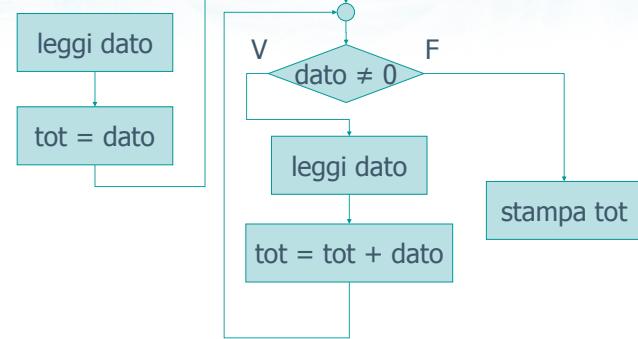
39

Soluzione 1



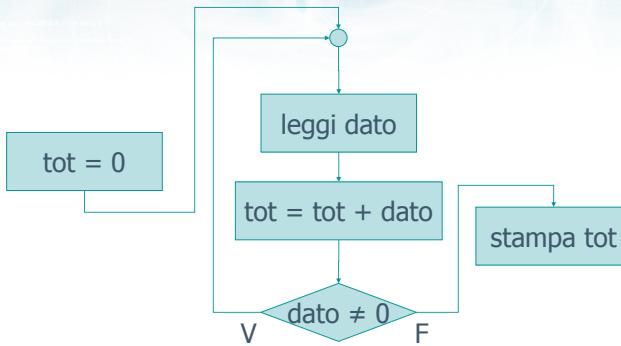
40

Soluzione 2



41

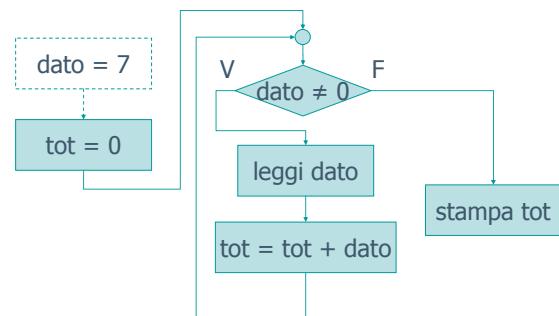
Soluzione 3



42

Errore frequente

- Dimenticare l'inizializzazione di una variabile utilizzata all'interno del ciclo

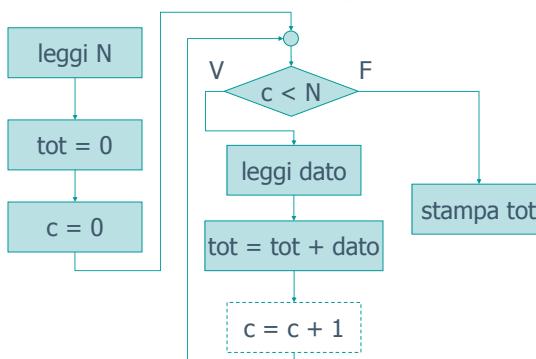


43



Errore frequente

- Dimenticare l'incremento della variabile contatore

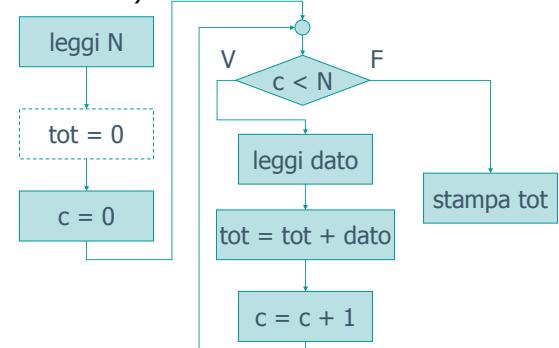


44



Errore frequente

- Dimenticare di inizializzare le altre variabili (oltre al contatore)



45

```
#include <stdio.h>
#include <ctype.h>
#include <limits.h>
#include <math.h>

#define MAXPAROLA 30
#define MAXSIG 80
```

int main(int argc, char *argv[])
{
 int freqMAXPAROLA; /* valore di controllo delle frequenze delle parole */
 char freq[MAXPAROLA]; /* frequenze delle parole */
 char freqMAXSIG; /* valore di controllo delle sigle */
 char freqSIG[MAXSIG]; /* frequenze delle sigle */
 int i, indice, lunghezza;

for(i=0; i<MAXPAROLA; i++)
 freq[i]=0;

for(i=0; i<MAXSIG; i++)
 freqSIG[i]=0;

if(argc > 1)
{
 if(freqSIG[0] != '\0')
 fprintf(stderr, "ERRORE: impossibile aprire il file SIGA\n");
 else
 {
 FILE *fpSIGA;
 fpSIGA=fopen(argv[1], "r");
 if(fpSIGA == NULL)
 fprintf(stderr, "ERRORE: impossibile aprire il file SIGA\n");
 else
 {
 fscanf(fpSIGA, "%c", freqSIG);
 while(fgets(freqSIG, MAXSIG, fpSIGA) != NULL)
 fscanf(fpSIGA, "%c", freqSIG);
 fclose(fpSIGA);
 }
 }
}

if(argc > 2)
{
 if(freqMAXSIG[0] != '\0')
 fprintf(stderr, "ERRORE: impossibile aprire il file SIGA\n");
 else
 {
 FILE *fpSIGA;
 fpSIGA=fopen(argv[2], "r");
 if(fpSIGA == NULL)
 fprintf(stderr, "ERRORE: impossibile aprire il file SIGA\n");
 else
 {
 fscanf(fpSIGA, "%c", freqMAXSIG);
 while(fgets(freqMAXSIG, MAXSIG, fpSIGA) != NULL)
 fscanf(fpSIGA, "%c", freqMAXSIG);
 fclose(fpSIGA);
 }
 }
}

if(argc > 3)
{
 if(freqMAXPAROLA[0] != '\0')
 fprintf(stderr, "ERRORE: impossibile aprire il file SIGA\n");
 else
 {
 FILE *fpSIGA;
 fpSIGA=fopen(argv[3], "r");
 if(fpSIGA == NULL)
 fprintf(stderr, "ERRORE: impossibile aprire il file SIGA\n");
 else
 {
 fscanf(fpSIGA, "%c", freqMAXPAROLA);
 while(fgets(freqMAXPAROLA, MAXPAROLA, fpSIGA) != NULL)
 fscanf(fpSIGA, "%c", freqMAXPAROLA);
 fclose(fpSIGA);
 }
 }
}

Cicli ed iterazioni

Istruzione while

```
#include <stdio.h>
#include <ctype.h>
#include <limits.h>
#include <math.h>

#define MAXPAROLA 30
#define MAXSIG 80
```

int main(int argc, char *argv[])
{
 int freqMAXPAROLA; /* valore di controllo delle frequenze delle parole */
 char freq[MAXPAROLA]; /* frequenze delle parole */
 char freqMAXSIG; /* valore di controllo delle sigle */
 char freqSIG[MAXSIG]; /* frequenze delle sigle */
 int i, indice, lunghezza;

for(i=0; i<MAXPAROLA; i++)
 freq[i]=0;

for(i=0; i<MAXSIG; i++)
 freqSIG[i]=0;

if(argc > 1)
{
 if(freqSIG[0] != '\0')
 fprintf(stderr, "ERRORE: impossibile aprire il file SIGA\n");
 else
 {
 FILE *fpSIGA;
 fpSIGA=fopen(argv[1], "r");
 if(fpSIGA == NULL)
 fprintf(stderr, "ERRORE: impossibile aprire il file SIGA\n");
 else
 {
 fscanf(fpSIGA, "%c", freqSIG);
 while(fgets(freqSIG, MAXSIG, fpSIGA) != NULL)
 fscanf(fpSIGA, "%c", freqSIG);
 fclose(fpSIGA);
 }
 }
}

if(argc > 2)
{
 if(freqMAXSIG[0] != '\0')
 fprintf(stderr, "ERRORE: impossibile aprire il file SIGA\n");
 else
 {
 FILE *fpSIGA;
 fpSIGA=fopen(argv[2], "r");
 if(fpSIGA == NULL)
 fprintf(stderr, "ERRORE: impossibile aprire il file SIGA\n");
 else
 {
 fscanf(fpSIGA, "%c", freqMAXSIG);
 while(fgets(freqMAXSIG, MAXSIG, fpSIGA) != NULL)
 fscanf(fpSIGA, "%c", freqMAXSIG);
 fclose(fpSIGA);
 }
 }
}

if(argc > 3)
{
 if(freqMAXPAROLA[0] != '\0')
 fprintf(stderr, "ERRORE: impossibile aprire il file SIGA\n");
 else
 {
 FILE *fpSIGA;
 fpSIGA=fopen(argv[3], "r");
 if(fpSIGA == NULL)
 fprintf(stderr, "ERRORE: impossibile aprire il file SIGA\n");
 else
 {
 fscanf(fpSIGA, "%c", freqMAXPAROLA);
 while(fgets(freqMAXPAROLA, MAXPAROLA, fpSIGA) != NULL)
 fscanf(fpSIGA, "%c", freqMAXPAROLA);
 fclose(fpSIGA);
 }
 }
}

Istruzione while

Sintassi dell'istruzione

(Argomento: Istruzioni di ripetizione in C)

Istruzione while

- ▶ Sintassi dell'istruzione
- ▶ Esercizio "Media aritmetica"
- ▶ Esecuzione del programma
- ▶ Cicli while annidati
- ▶ Esercizio "Quadrato"

2

(Argomento: Istruzioni di ripetizione in C)

Istruzioni di ripetizione in C

- ▶ Nel linguaggio C esistono tre distinte istruzioni di iterazione
 - **while**
 - **do-while**
 - **for**
- ▶ La forma più generale è l'istruzione di tipo **while**
- ▶ L'istruzione **do-while** si usa in taluni contesti (es. controllo errori di input)
- ▶ L'istruzione **for** è usatissima, soprattutto per numero di iterazioni noto

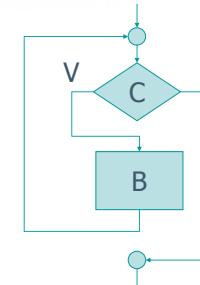
4

(Argomento: Comportamento del while)

Comportamento del while

1. Valuta la condizione C
2. Se C è falsa, salta completamente l'iterazione e vai all'istruzione che segue la }
3. Se C è vera, esegui una volta il blocco di istruzioni B
4. Al termine del blocco B, ritorna al punto 1. per rivalutare la condizione C

```
while ( C )
{
    B ;
}
```



5

6

Numero di iterazioni note

```
int i, N ;  
  
i = 0 ;  
while ( i < N )  
{  
    /* Corpo dell'iterazione */  
    ...  
  
    i = i + 1 ;  
}
```

7

Esempio

```
int i ;  
  
i = 1 ;  
while ( i <= 10 )  
{  
    printf("Numero = %d\n", i) ;  
    i = i + 1 ;  
}
```

8

Esempio

```
int i, n ;  
float f ;  
.... /* leggi n */ ....  
i = 2 ;  
f = 1.0 ;  
while ( i <= n )  
{  
    f = f * i ;  
    i = i + 1 ;  
}  
printf("Fattoriale di %d = %f\n",  
n, f);
```

9

Particolarità

- Nel caso in cui il corpo del `while` sia composto di una sola istruzione, si possono omettere le parentesi graffe
 - Non succede quasi mai

```
while ( C )  
{  
    B ;  
}
```

```
while ( C )  
    B ;
```

```
#include <stdio.h>  
#include <string.h>  
#include <ctype.h>  
  
#define MAXPAROLA 30  
#define MAXIGRA 80  
  
int main(int argc, char *argv[]){  
    int lungMAXPAROLA; /* vettore di condizioni  
    per la lettura della lunghezza delle parole */  
    char codaMAXIGRA[ ];  
    int i, indice, lungparola ;  
    FILE *f;  
  
    f=fopen(argv[1], "r");  
    if(f==NULL){  
        printf("ERRORE: non è possibile aprire il file %s", argv[1]);  
        exit(1);  
    }  
  
    /* legge la parola da inserire nel vettore */  
    for(i=0; i<lungMAXPAROLA; i++)  
        lungparola[i]=fgetc(f);  
  
    /* legge la parola da inserire nel vettore */  
    for(i=0; i<lungMAXIGRA; i++)  
        codaMAXIGRA[i]=fgetc(f);  
  
    /* chiude il file */  
    fclose(f);  
    if(f!=NULL)  
        printf("ERRORE: impossibile chiudere il file %s", argv[1]);  
    else  
        printf("file %s letto con successo", argv[1]);  
  
    printf("\n");  
    printf("Inserisci una parola da inserire nel vettore: ");  
    gets(codaMAXIGRA);  
    if(strlen(codaMAXIGRA)>lungMAXIGRA){  
        printf("ERRORE: la parola inserita è troppo lunga (%d>%d)",  
        strlen(codaMAXIGRA), lungMAXIGRA);  
        exit(1);  
    }  
  
    /* scrive la parola nel vettore */  
    for(i=0; i<lungMAXIGRA; i++)  
        codaMAXIGRA[i]=fputc(codaMAXIGRA[i], f);  
  
    /* chiude il file */  
    fclose(f);  
    if(f!=NULL)  
        printf("ERRORE: impossibile chiudere il file %s", argv[1]);  
    else  
        printf("file %s scritto con successo", argv[1]);  
  
    printf("\n");  
    return 0;  
}
```

Istruzione while

Esercizio “Media aritmetica”

- Si realizzi un programma C in grado di
 - Leggere un numero naturale n
 - Leggere n numeri reali
 - Calcolare e visualizzare la media aritmetica di tali numeri
- Osservazione
 - Attenzione al caso in cui $n \leq 0$

Analisi

```
Prompt dei comandi
MEDIA ARITMETICA
Introduci n: 3
Ora introduci 3 valori
Valore 1: 6.5
Valore 2: 2.5
Valore 3: 3.0
Risultato: 4.000000
```

13

Algoritmo

- Acquisisci n
- Inizializza totale = 0
- Ripeti n volte
 - Acquisisci un dato
 - Somma il dato al totale dei dati acquisiti
- Calcola e stampa la media = totale / n

14

Algoritmo

- Acquisisci n
- Se $n > 0$
 - Inizializza totale = 0
 - Ripeti n volte
 - Acquisisci un dato
 - Somma il dato al totale dei dati acquisiti
 - Calcola e stampa la media = totale / n
- Altrimenti stampa messaggio di errore

15

Traduzione in C (1/3)

```
#include <stdio.h>
#include <stdlib.h>

int main(void)
{
    int i, n ;
    float dato ;
    float somma ;

    printf("MEDIA ARITMETICA\n");

    /* Leggi n */
    printf("Introduci n: ");
    scanf("%d", &n) ;
```

16

Traduzione in C (2/3)

```
/* Controlla la correttezza del valore n */
if( n>0 )
{
    /* n corretto... procedi! */
    ...vedi lucido seguente...
}
else
{
    /* n errato in quanto e' n <= 0 */
    printf("Non ci sono dati da inserire\n");
    printf("Impossibile calcolare la media\n");
}

/* main */
```

17

Traduzione in C (3/3)

```
/* Leggi i valori e calcola la media */
printf("Ora immetti %d valori\n", n) ;
somma = 0.0 ;
i = 0 ;
while( i < n )
{
    printf("Valore %d: ", i+1) ;
    scanf("%f", &dato) ;

    somma = somma + dato ;

    i = i + 1 ;
}

printf("Risultato: %f\n", somma/n) ;
```

18

```

#include <stdio.h>
#include <ctype.h>
#include <string.h>

#define MAXPAROLA 30
#define MAXDIGA 80

int main(int argc, char *argv[])
{
    int lung(MAXPAROLA); /* valore di controllo delle frequenze delle lunghezze delle parole */
    int i, indice, lunghezza;
    FILE *fptr;
    char digi[MAXDIGA];
    int n=0;
    float somma=0.0;
    float media=0.0;
    int maxlung=0;
    int minlung=MAXDIGA, f=0;
    int flagdig=0;
    if(argc!=2)
    {
        printf("ERRORE: non sono stati inseriti i parametri richiesti\n");
        exit(1);
    }
    else
    {
        fptr=fopen(argv[1], "r");
        if(fptr==NULL)
        {
            printf("ERRORE: impossibile aprire il file %s", argv[1]);
            exit(1);
        }
        while(fgets(digi, MAXDIGA, fptr)!=NULL)
        {
            lung=digi[0];
            if(lung>maxlung)
                maxlung=lung;
            if(lung<minlung)
                minlung=lung;
            lunghezza=digi[lung];
            if(lunghezza=='\0')
                break;
            if(flagdig==0)
                indice=0;
            else
                indice=lung-1;
            if(isdigit(digi[indice]))
                somma+=lung*digi[indice];
            else
                f=1;
            if(f==1)
                break;
            if(indice==0)
                n++;
            flagdig=1;
        }
        if(f==1)
            printf("ERRORE: impossibile aprire il file %s", argv[1]);
        else
        {
            media=somma/n;
            printf("Media aritmetica: %.2f\n", media);
        }
        fclose(fptr);
    }
}

```

Istruzione while

Esecuzione del programma

```

#include <stdio.h>
#include <ctype.h>
#include <string.h>

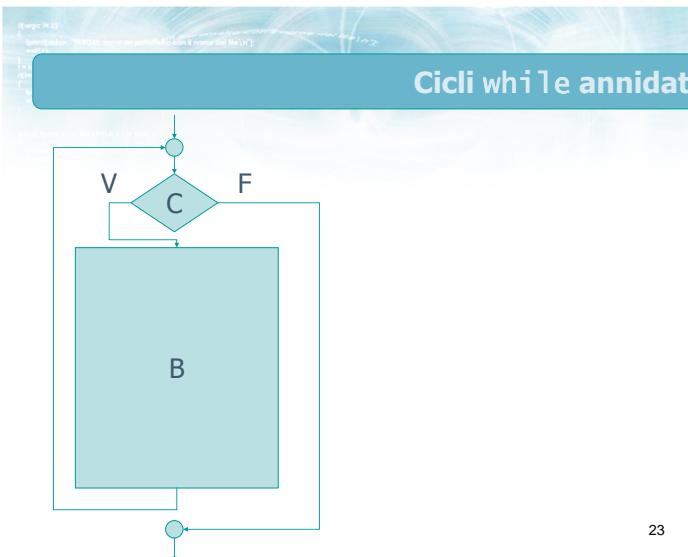
#define MAXPAROLA 30
#define MAXDIGA 80

int main(int argc, char *argv[])
{
    int lung(MAXPAROLA); /* valore di controllo delle frequenze delle lunghezze delle parole */
    int i, indice, lunghezza;
    FILE *fptr;
    char digi[MAXDIGA];
    int n=0;
    float somma=0.0;
    float media=0.0;
    int maxlung=0;
    int minlung=MAXDIGA, f=0;
    int flagdig=0;
    if(argc!=2)
    {
        printf("ERRORE: non sono stati inseriti i parametri richiesti\n");
        exit(1);
    }
    else
    {
        fptr=fopen(argv[1], "r");
        if(fptr==NULL)
        {
            printf("ERRORE: impossibile aprire il file %s", argv[1]);
            exit(1);
        }
        while(fgets(digi, MAXDIGA, fptr)!=NULL)
        {
            lung=digi[0];
            if(lung>maxlung)
                maxlung=lung;
            if(lung<minlung)
                minlung=lung;
            lunghezza=digi[lung];
            if(lunghezza=='\0')
                break;
            if(flagdig==0)
                indice=0;
            else
                indice=lung-1;
            if(isdigit(digi[indice]))
                somma+=lung*digi[indice];
            else
                f=1;
            if(f==1)
                break;
            if(indice==0)
                n++;
            flagdig=1;
        }
        if(f==1)
            printf("ERRORE: impossibile aprire il file %s", argv[1]);
        else
        {
            media=somma/n;
            printf("Media aritmetica: %.2f\n", media);
        }
        fclose(fptr);
    }
}

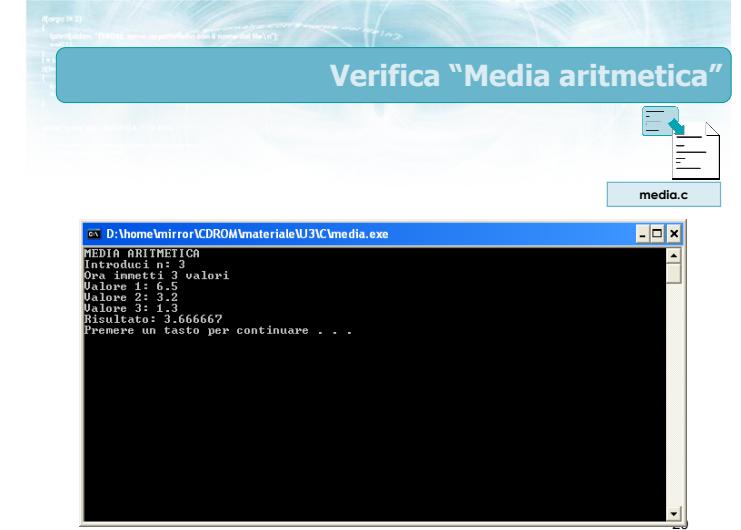
```

Istruzione while

Cicli while annidati

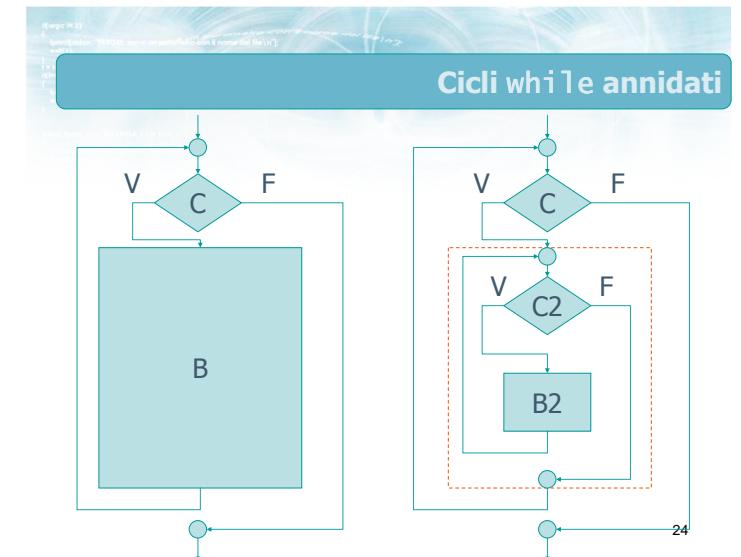


23



- All'interno del corpo del ciclo while è possibile racchiudere qualsiasi altra istruzione C
- In particolare, è possibile racchiudere un'istruzione while all'interno di un'altra istruzione while
- In tal caso, per ogni singola iterazione del ciclo while più esterno, vi saranno tutte le iterazioni previste per il ciclo più interno

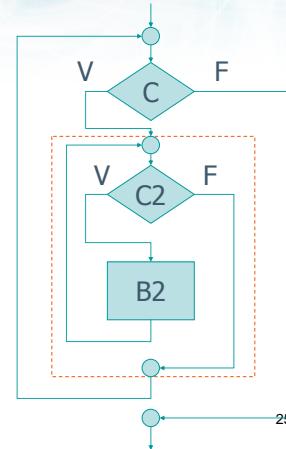
22



24

Cicli while annidati

```
while( c )
{
    while( c2 )
    {
        B2 ;
    }
}
```



Esempio

```
i = 0 ;
while( i<N )
{
    j = 0 ;
    while( j<N )
    {
        printf("i=%d - j=%d\n", i, j);
        j = j + 1 ;
    }
    i = i + 1 ;
}
```

26

Esempio

```
i = 0 ;
while( i<N )
{
    j = 0 ;
    while( j<N )
    {
        printf("i=%d - j=%d\n", i, j);
        j = j + 1 ;
    }
    i = i + 1 ;
}
```

```
i=0 - j=0
i=0 - j=1
i=0 - j=2
i=1 - j=0
i=1 - j=1
i=1 - j=2
i=2 - j=0
i=2 - j=1
i=2 - j=2
```

27



```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

#define MAXPAROLA 30
#define MAXIGA 80

int main(int argc, char *argv[])
{
    int lungMAXPAROLA; /* variezza di parola */
    char *lungMAXIGA; /* variezza di riga */
    char riga[MAXIGA];
    int i, lungMAXI;
    i=0;
}

for(;; i<MAXPAROLA; i++)
{
    if( (i==0) || (i==MAXIGA) )
    {
        printf("Inserire una parola: ");
        fgets(riga, MAXIGA, stdin);
        if( (i==0) || (i==MAXIGA) )
        {
            printf("Inserire un numero: ");
            scanf("%d", &lungMAXPAROLA);
        }
    }
    else
    {
        if( (i==0) || (i==MAXIGA) )
        {
            printf("Inserire un numero: ");
            scanf("%d", &lungMAXI);
        }
    }
}

while( (riga[i]>='0') && (riga[i]<='9') )
{
    if( (i==0) || (i==MAXIGA) )
    {
        printf("Inserire una parola: ");
        fgets(riga, MAXIGA, stdin);
        if( (i==0) || (i==MAXIGA) )
        {
            printf("Inserire un numero: ");
            scanf("%d", &lungMAXPAROLA);
        }
    }
    else
    {
        if( (i==0) || (i==MAXIGA) )
        {
            printf("Inserire un numero: ");
            scanf("%d", &lungMAXI);
        }
    }
}
```

Istruzione while

Esercizio "Quadrato"

Esercizio "Quadrato"

- Si realizzi un programma C in grado di
 - Leggere un numero naturale n
 - Visualizzare un quadrato di lato n costituito da asterischi

29

Analisi

```
PS C:\Users\ASUS\OneDrive - Università "Federico II" di Napoli\Scrivania\Scrivania> Prompt dei comandi
QUADRATO
Introduci n: 5
*****
*****
*****
*****
*****
```

30

Algoritmo

- » Acquisisci n
- » Ripeti n volte
 - Stampa una riga di n asterischi

31

Algoritmo

- » Acquisisci n
- » Ripeti n volte
 - Stampa una riga di n asterischi
- Ripeti n volte
 - Stampa un singolo asterisco
 - Vai a capo

32

Traduzione in C



```
i = 0 ;
while( i<n )
{
    j = 0 ;
    while( j<n )
    {
        printf("*") ;
        j = j + 1 ;
    }

    printf("\n") ;

    i = i + 1 ;
}
```

33

Traduzione in C



```
i = 0 ;
while( i<n )
{
    j = 0 ;
    while( j<n )
    {
        printf("*") ;
        j = j + 1 ;
    }

    printf("\n") ;

    i = i + 1 ;
}
```

Stampa n asterischi

Vai a capo

35

Traduzione in C



```
i = 0 ;
while( i<n )
{
    j = 0 ;
    while( j<n )
    {
        printf("*") ;
        j = j + 1 ;
    }

    printf("\n") ;

    i = i + 1 ;
}
```

Ripeti n volte

Stampa una riga di n asterischi

34

Traduzione in C



```
i = 0 ;
while( i<n )
{
    j = 0 ;
    while( j<n )
    {
        printf("*") ;
        j = j + 1 ;
    }

    printf("\n") ;

    i = i + 1 ;
}
```

Ripeti n volte

Stampa un asterisco

36

Schemi ricorrenti nei cicli

- Contatori
- Accumulatori
- Flag
- Esistenza e universalità

Cicli ed iterazioni

Schemi ricorrenti nei cicli

2

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>
```

```
#define MAXPAROLA 30
#define MAXSIGA 80
```

```
int main(int argc, char *argv[])
{
    int lungMAXPAROLA; /* valore di controllo delle frequenze delle lunghezze delle parole */
    char lungMAXSIGA; /* valore di controllo delle frequenze delle lunghezze delle sigle */
    int i, indice, lungparola;
```

```
for(i=0; i<MAXPAROLA; i++)
    lungparola = 0;

for(i=0; i<MAXSIGA; i++)
    lungMAXSIGA = 0;

if(argc > 1)
{
    strcpy(lungMAXSIGA, "TUTTI");
    strcpy(lungMAXPAROLA, "TUTTI");
    lungparola = 1;
    lungMAXSIGA = 1;
}

if(argc > 2)
{
    if(strcmp(argv[1], "TUTTI") == 0)
        lungparola = 1;
    else
        lungparola = atoi(argv[1]);
}

if(argc > 3)
{
    if(strcmp(argv[2], "TUTTI") == 0)
        lungMAXSIGA = 1;
    else
        lungMAXSIGA = atoi(argv[2]);
}
```

Schemi ricorrenti nei cicli

Contatori

4

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>
```

```
#define MAXPAROLA 30
#define MAXSIGA 80
```

```
int main(int argc, char *argv[])
{
    int lungMAXPAROLA; /* valore di controllo delle frequenze delle lunghezze delle parole */
    char lungMAXSIGA; /* valore di controllo delle frequenze delle lunghezze delle sigle */
    int i, indice, lungparola;
```

```
for(i=0; i<MAXPAROLA; i++)
    lungparola = 0;

for(i=0; i<MAXSIGA; i++)
    lungMAXSIGA = 0;

if(argc > 1)
{
    strcpy(lungMAXSIGA, "TUTTI");
    strcpy(lungMAXPAROLA, "TUTTI");
    lungparola = 1;
    lungMAXSIGA = 1;
}

if(argc > 2)
{
    if(strcmp(argv[1], "TUTTI") == 0)
        lungparola = 1;
    else
        lungparola = atoi(argv[1]);
}

if(argc > 3)
{
    if(strcmp(argv[2], "TUTTI") == 0)
        lungMAXSIGA = 1;
    else
        lungMAXSIGA = atoi(argv[2]);
}
```

Contatori

- Spesso in un ciclo è utile sapere
 - Quante iterazioni sono state fatte
 - Quante iterazioni rimangono da fare
 - Quale numero di iterazione sia quella corrente
- Per questi scopi si usano delle "normali" variabili intere, dette **contatori**
 - Iniziate prima del ciclo
 - Incrementate/decrementate ad ogni iterazione
 - Oppure incrementate/decrementate ogni volta che si riscontra una certa condizione

```
int i, N ;
...
i = 0 ;
while ( i < N )
{
    printf("Iterazione %d\n", i+1) ;
    i = i + 1 ;
}
```

Contatori

5

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>
```

```
#define MAXPAROLA 30
#define MAXSIGA 80
```

```
int main(int argc, char *argv[])
{
    int lungMAXPAROLA; /* valore di controllo delle frequenze delle lunghezze delle parole */
    char lungMAXSIGA; /* valore di controllo delle frequenze delle lunghezze delle sigle */
    int i, indice, lungparola;
```

```
for(i=0; i<MAXPAROLA; i++)
    lungparola = 0;

for(i=0; i<MAXSIGA; i++)
    lungMAXSIGA = 0;

if(argc > 1)
{
    strcpy(lungMAXSIGA, "TUTTI");
    strcpy(lungMAXPAROLA, "TUTTI");
    lungparola = 1;
    lungMAXSIGA = 1;
}

if(argc > 2)
{
    if(strcmp(argv[1], "TUTTI") == 0)
        lungparola = 1;
    else
        lungparola = atoi(argv[1]);
}

if(argc > 3)
{
    if(strcmp(argv[2], "TUTTI") == 0)
        lungMAXSIGA = 1;
    else
        lungMAXSIGA = atoi(argv[2]);
}
```

Esempio

- Scrivere un programma in C che
 - legga dall'utente 10 numeri interi
 - al termine dell'inserimento, stampi
 - quanti tra i numeri inseriti sono positivi
 - quanti tra i numeri inseriti sono negativi
 - quanti tra i numeri inseriti sono nulli

6

Soluzione (1/3)

```
int npos, nneg, nzero ;  
....  
npos = 0 ;  
nneg = 0 ;  
nzero = 0 ;
```

7

Soluzione (2/3)

```
i = 0 ;  
while( i<n )  
{  
    printf("Inserisci dato %d: ", i+1);  
    scanf("%d", &dato);  
  
    if( dato>0 )  
        npos = npos + 1 ;  
    else if( dato<0 )  
        nneg = nneg + 1 ;  
    else  
        nzero = nzero + 1 ;  
  
    i = i + 1 ;  
}
```

8

Soluzione (3/3)

```
printf("Numeri positivi: %d\n", npos);  
printf("Numeri negativi: %d\n", nneg);  
printf("Numeri nulli: %d\n", nzero);
```

9

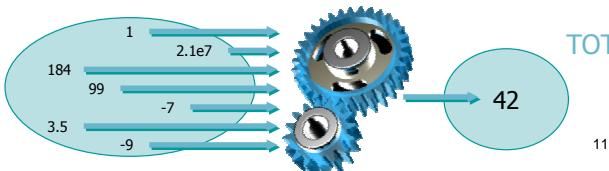
```
#include <stdio.h>  
#include <string.h>  
#include <ctype.h>  
  
#define MAXPAROLA 30  
#define MAXIGA 80  
  
int main(int argc, char *argv[]){  
    int lungMAXPAROLA; /* variezza di parola */  
    char *parola; /* parola inserita dalla tastiera */  
    char doppMAXIGA[ ];  
    int i, j, lungMAXI;  
    i=0; j=0;  
  
    for(i=0; i<MAXPAROLA; i++)  
        doppMAXIGA[i] = '\0';  
  
    if(argc > 1){  
        /* spazio per la parola inserita dalla tastiera */  
        if(strlen(argv[1]) > MAXPAROLA){  
            printf("ERRORE, inserisci una parola di al massimo %d caratteri\n", lungMAXPAROLA);  
            exit(1);  
        }  
        /* legge la parola inserita dalla tastiera */  
        if(argv[1][0] == '\0')  
            printf("ERRORE, inserisci una parola non vuota\n");  
        else  
            i = 0;  
        while( (igual(argv[1], MAXIGA, i)) != NULL )  
            i++;  
    }
```

Schemi ricorrenti nei cicli

Accumulatori

Accumulatori (1/2)

- Spesso in un ciclo occorre calcolare un valore TOT che dipende dall'insieme dei valori analizzati nelle singole iterazioni
- Esempi:
 - TOT = sommatoria dei dati analizzati
 - TOT = produttoria dei dati analizzati
 - TOT = massimo, minimo dei dati analizzati



11

Accumulatori (2/2)

- In questo caso si usano delle variabili (interi o reali) dette **accumulatori**
 - Inizializzare TOT al valore che dovrebbe avere in assenza di dati (come se fosse n=0)
 - Ad ogni iterazione, aggiornare TOT tenendo conto del dato appena analizzato
 - Al termine del ciclo, TOT avrà il valore desiderato

12

Esempio: somma primi 10 interi

- » Si scriva un programma in C che stampi il valore della somma dei primi 10 numeri interi

13

Soluzione: somma primi 10 interi

```
int tot ;  
  
i = 1 ;  
tot = 0 ;  
while( i<=10 )  
{  
    tot = tot + i ;  
  
    i = i + 1 ;  
}  
  
printf("La somma dei numeri da 1 a 10 ");  
printf("vale %d\n", tot) ;
```

15

14

Inizializzazione di TOT

- Qual è la somma dei primi 0 numeri interi?
 - TOT = 0

Aggiornamento di TOT

- Sapendo che TOT è la somma dei primi ($i-1$) numeri interi, e sapendo che il prossimo numero intero da sommare vale i , quanto dovrà valere TOT?

$$\bullet \text{TOT} = \text{TOT} + i$$

Analisi

Inizializzazione di TOT

- Qual è il valore del fattoriale per $K=0$?
 - TOT = 1.0

Aggiornamento di TOT

- Sapendo che TOT è pari al fattoriale di $i-1$, e sapendo che il prossimo numero da considerare è i , quanto dovrà valere TOT?
 - $\text{TOT} = \text{TOT} * i$

17

18

Esempio: fattoriale di K

- » Si scriva un programma in C che, dato un numero intero K , calcoli e stampi il fattoriale di K
- » $\text{TOT} = K!$

$$TOT = K! = \prod_{i=1}^{i=K} i$$

16

Soluzione: fattoriale di K

```
float tot ;
```

```
i = 1 ;  
tot = 1.0 ;  
while( i<=K )  
{  
    tot = tot * i ;  
  
    i = i + 1 ;  
}
```

```
printf("Il fattoriale di %d ", K);  
printf("vale %f\n", tot) ;
```

Esempio: massimo

» Si scriva un programma in C che

- acquisisca da tastiera N numeri reali
- stampi il valore massimo tra i numeri acquisiti

19

Esempio: massimo

```
int max ;  
  
i = 0 ;  
max = INT_MIN ;  
while( i<N )  
{  
    scanf("%d", &dato) ;  
    if(dato>max)  
        max = dato ;  
  
    i = i + 1 ;  
}  
  
printf("Massimo = %d\n", max);
```

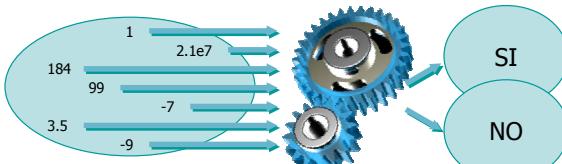
21

Flag, indicatori, variabili logiche

» Spesso occorre analizzare una serie di dati per determinare se si verifica una certa condizione

» Esempi:

- Tra i dati inseriti esiste il numero 100?
- Esistono due numeri consecutivi uguali?



23

Analisi

» Inizializzazione di TOT

- Qual è il valore del massimo in un insieme di 0 numeri?

- Non esiste, non è definito!
- TOT = numero molto piccolo, che non possa certamente essere scambiato con il massimo

» Aggiornamento di TOT

- Sapendo che TOT è pari al massimo dei primi i-1 dati, e sapendo che il prossimo dato da considerare è d, quanto dovrà valere TOT?

 - Se d<=TOT, allora TOT rimane il massimo
 - Se d>TOT, allora il nuovo massimo sarà TOT=d

20

```
#include <stdio.h>  
#include <string.h>  
#include <ctype.h>  
  
#define MAXPAROLA 30  
#define MAXIGRA 80  
  
int main(int argc, char *argv[]){  
    int lungMAXPAROLA; /* dimensione massima della parola */  
    int lungMAXIGRA; /* dimensione massima della riga */  
    char riga[MAXIGRA];  
    int i, j, lungparola;  
    FILE *f;  
  
    for(i=0; i<MAXPAROLA; i++)  
        lungparola++;  
  
    f=fopen(argv[1], "r");  
    if(f==NULL){  
        printf("ERRORE, impossibile aprire il file %s", argv[1]);  
        exit(1);  
    }  
  
    while(fgets(riga, MAXIGRA, f)!=NULL){  
        ...  
    }  
}
```

Schemi ricorrenti nei cicli

Flag

» Nel momento in cui si "scopre" il fatto, non si può interrompere l'elaborazione ma occorre comunque terminare il ciclo

» Al termine del ciclo, come fare a "ricordarsi" se si era "scoperto" il fatto o no?

Problemi

24

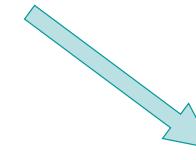
Una possibile soluzione

- Per sapere
 - Se una certa condizione si verifica è possibile contare
 - Quante volte quella condizione si verifica ed in seguito verificare
 - Verificare se il conteggio è diverso da zero
- Ci riconduciamo ad un problema risolubile per mezzo di una variabile contatore

25

Esempio 1

Tra i dati inseriti esiste il numero 100?



Conta quante volte tra i dati inseriti compare il numero 100

Il conteggio è > 0 ?

26

Soluzione (1/3)

```
int i, n ;  
int dato ;  
int conta ;  
/* conta il numero di "100" letti */  
  
printf("TROVA 100\n");  
  
n = 10 ;  
  
printf("Inserisci %d numeri\n", n);
```

trova100v1.c

27

Soluzione (2/3)

```
conta = 0 ;  
  
i = 0 ;  
while( i<n )  
{  
    printf("Inserisci dato %d: ", i+1);  
    scanf("%d", &dato);  
  
    if( dato == 100 )  
        conta = conta + 1 ;  
  
    i = i + 1 ;  
}
```

trova100v1.c

28

Soluzione (3/3)

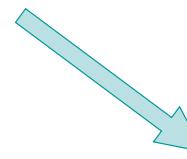
```
if( conta != 0 )  
    printf("Ho trovato il 100\n");  
else  
    printf("NON ho trovato il 100\n");
```

trova100v1.c

29

Esempio 2

Esistono due numeri consecutivi uguali?



Conta quante volte due numeri consecutivi sono uguali

Il conteggio è > 0 ?

30

Soluzione (1/3)

```
int i, n ;
int dato ;
int precedente ;
int conta ;
/* conta il numero di "doppioni" trovati */

printf("TROVA UGUALI\n");
n = 10 ;

printf("Inserisci %d interi\n", n);
```

31

ugualiv1.c

Soluzione (2/3)

```
if( conta != 0 )
    printf("Ho trovato dei numeri
           consecutivi uguali\n");
else
    printf("NON ho trovato dei numeri
           consecutivi uguali\n");
```

33

32

ugualiv1.c

Svantaggi

- Il contatore determina quante volte si verifica la condizione ricercata
- In realtà non mi serve sapere quante volte, ma solo se si è verificata almeno una volta
- Usiamo un contatore “degenero”, che una volta arrivato ad 1 non si incrementa più

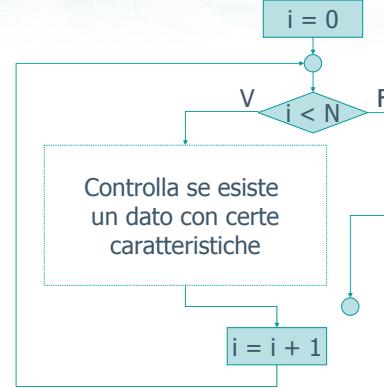
Variabili “flag”

- 
- Variabili intere che possono assumere solo due valori
 - Variabile = 0 ⇒ la condizione non si è verificata
 - Variabile = 1 ⇒ la condizione si è verificata
 - Viene inizializzata a 0 prima del ciclo
 - Se la condizione si verifica all'interno del ciclo, viene posta a 1
 - Al termine del ciclo si verifica il valore
 - Sinonimi: Flag, Variabile logica, Variabile booleana, Indicatore

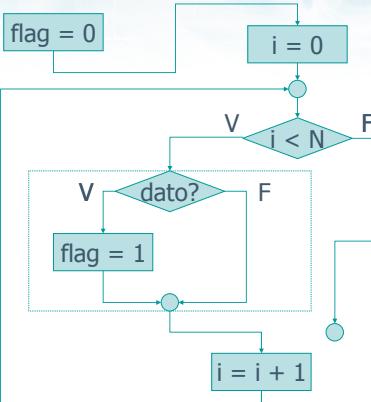
35

36

Analisi



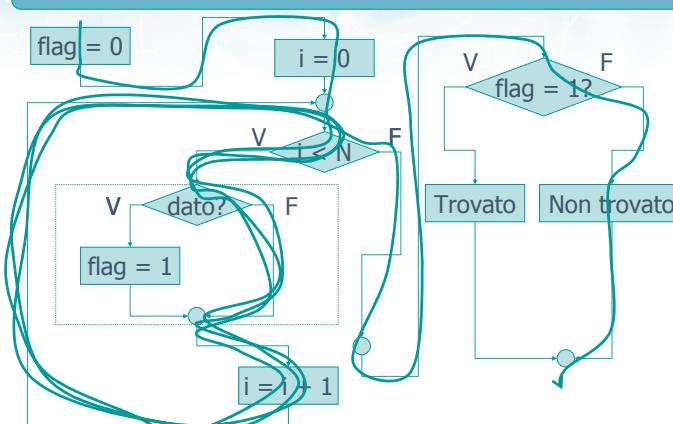
Analisi



37

38

Analisi



39

40

Soluzione con flag – esempio 1

```

int trovato ; /* ho visto il numero "100"? */
...
trovato = 0 ;

i = 0 ;
while( i < n )
{
    scanf("%d", &dato);

    if( dato == 100 )
        trovato = 1 ;

    i = i + 1 ;
}

if( trovato != 0 )
    printf("Trovato il numero 100\n");
else
    printf("NON trovato il numero 100\n");
  
```

trova100v2.c

Soluzione con flag – esempio 2

```

int doppi ; /* trovati "duplicati" ? */
...
doppi = 0 ;

precedente = INT_MAX ;
i = 0 ;
while( i < n )
{
    scanf("%d", &dato);

    if( dato == precedente )
        doppi = 1 ;

    precedente = dato ;
    i = i + 1 ;
}

if( doppi != 0 )
    printf("Trovati consecutivi uguali\n");
  
```

ugualiV2.c

```
#include <stdio.h>
#include <ctype.h>

#define MAXPARIOLA 30
#define MAXBAGNO 80

int main(int argc, char *argv[])
{
    int freqMAXPARIOLA; /* frequenza delle lunghezze delle parole */
    int freqMAXBAGNO; /* lunghezza delle parole */

    if(argc != 2)
    {
        fprintf(stderr, "ERRORE: non sono presenti dati di input nel file %s\n", argv[0]);
        exit(1);
    }

    freqMAXPARIOLA = atoi(argv[1]);
    freqMAXBAGNO = freqMAXPARIOLA / 2;
    if(freqMAXBAGNO == 0)
    {
        fprintf(stderr, "ERRORE: impossibile aprire il file %s", argv[1]);
        exit(1);
    }

    printf("Aperto file %s, MAXBAGNO = %d\n", argv[1], freqMAXBAGNO);
}
```

Schemi ricorrenti nei cicli

Esistenza e universalità

44

Ricerca di esistenza o universalità

- L'utilizzo dei flag è più utile quando si desiderano verificare delle proprietà su un certo insieme di dati
 - È vero che tutti i dati verificano la proprietà?
 - È vero che almeno un dato verifica la proprietà?
 - È vero che nessun dato verifica la proprietà?
 - È vero che almeno un dato non verifica la proprietà?

Esempi

- Verificare che tutti i dati inseriti dall'utente siano positivi
- Determinare se una sequenza di dati inseriti dall'utente è crescente
- Due numeri non sono primi tra loro se hanno almeno un divisore comune
 - esiste almeno un numero che sia divisore dei due numeri dati
- Un poligono regolare ha tutti i lati di lunghezza uguale
 - ogni coppia di lati consecutivi ha uguale lunghezza

45

Formalizzazione

- È vero che tutti i dati verificano la proprietà?
 - $\forall x : P(x)$
- È vero che almeno un dato verifica la proprietà?
 - $\exists x : P(x)$
- È vero che nessun dato verifica la proprietà?
 - $\forall x : \text{not } P(x)$
- È vero che almeno un dato non verifica la proprietà?
 - $\exists x : \text{not } P(x)$

46

Realizzazione (1/2)

- Esistenza: $\exists x : P(x)$
 - Inizializzo flag $F = 0$
 - Ciclo su tutte le x
 - Se $P(x)$ è vera
 - Pongo $F = 1$
 - Se $F = 1$, l'esistenza è dimostrata
 - Se $F = 0$, l'esistenza è negata

47

Realizzazione (1/2)

- Esistenza: $\exists x : P(x)$
 - Inizializzo flag $F = 0$
 - Ciclo su tutte le x
 - Se $P(x)$ è vera
 - Pongo $F = 1$
 - Se $F = 1$, l'esistenza è dimostrata
 - Se $F = 0$, l'esistenza è negata
- Universalità: $\forall x : P(x)$
 - Inizializzo flag $F = 1$
 - Ciclo su tutte le x
 - Se $P(x)$ è falsa
 - Pongo $F = 0$
 - Se $F = 1$, l'universalità è dimostrata
 - Se $F = 0$, l'universalità è negata

48

Realizzazione (2/2)

► Esistenza: $\exists x : \text{not } P(x)$ ► Universalità: $\forall x : \text{not } P(x)$

● Inizializzo flag $F = 0$

● Inizializzo flag $F = 1$

● Ciclo su tutte le x

- Se $P(x)$ è falsa
- Pongo $F = 1$

● Se $F = 1$, l'esistenza è dimostrata

● Se $F = 0$, l'esistenza è negata

● Ciclo su tutte le x

- Se $P(x)$ è vera
- Pongo $F = 0$

● Se $F = 1$, l'universalità è dimostrata

● Se $F = 0$, l'universalità è negata

49

Esempio 1

► Verificare che tutti i dati inseriti dall'utente siano positivi

```
int positivi ;
...
positivi = 1 ;
i = 0 ;
while( i<n )
{
    ...
    if( dato <= 0 )
        positivi = 0 ;
    ...
    i = i + 1 ;
}
if( positivi == 1 )
    printf("Tutti positivi\n");
```

50

Esempio 2

► Determinare se una sequenza di dati inseriti dall'utente è crescente

```
int crescente ;
...
crescente = 1 ;
precedente = INT_MIN ;
i = 0 ;
while( i<n )
{
    ...
    if( dato < precedente )
        crescente = 0 ;
    precedente = dato ;
    ...
    i = i + 1 ;
}
```

51

Esempio 3

► Due numeri non sono primi tra loro se hanno almeno un divisore comune

```
int A, B ;
int nonprimi ;
...
nonprimi = 0 ;
i = 2 ;
while( i<=A )
{
    ...
    if( (A%i==0) && (B%i==0) )
        nonprimi = 1 ;
    ...
    i = i + 1 ;
}
```

52

Esempio 4

► Un poligono regolare ha tutti i lati di lunghezza uguale

```
int rego ;
...
rego = 1 ;
precedente = INT_MIN ;
i = 0 ;
while( i<n )
{
    ...
    if( lato != precedente )
        rego = 0 ;
    precedente = lato ;
    ...
    i = i + 1 ;
}
```

53

Errore frequente

► Resetare il flag al valore di inizializzazione, dimenticando di fatto eventuali condizioni incontrate in precedenza

```
trovato = 0 ;
i = 0 ;
while( i<n )
{
    ...
    if( dato == 100 )
        trovato = 1 ;
    else
        trovato = 0 ;
    ...
    i = i + 1 ;
}
```

```
trovato = 0 ;
i = 0 ;
while( i<n )
{
    ...
    if( dato == 100 )
        trovato = 1 ;
    ...
    i = i + 1 ;
}
```



Errore frequente

- Passare ai fatti non appena trovato il primo elemento che soddisfa la proprietà

```
trovato = 0 ;  
i = 0 ;  
while( i<n )  
{  
    ...  
    if( dato == 100 )  
    {  
        trovato = 1 ;  
        printf("W!\n");  
    }  
    ...  
    i = i + 1 ;  
}
```

```
trovato = 0 ;  
i = 0 ;  
while( i<n )  
{  
    ...  
    if( dato == 100 )  
        trovato = 1 ;  
    ...  
    i = i + 1 ;  
}  
if(trovato==1)  
    printf("W!\n");
```



Errore frequente

- Pensare che al primo fallimento si possa determinare che la proprietà è falsa

```
trovato = 0 ;  
i = 0 ;  
while( i<n )  
{  
    ...  
    if( dato == 100 )  
        trovato = 1 ;  
    else  
        printf("NO!\n");  
    ...  
    i = i + 1 ;  
}
```

```
trovato = 0 ;  
i = 0 ;  
while( i<n )  
{  
    ...  
    if( dato == 100 )  
        trovato = 1 ;  
    ...  
    i = i + 1 ;  
}  
if(trovato==0)  
    printf("NO!\n");
```

- ▶ Sintassi dell'istruzione
- ▶ Operatori di autoincremento
- ▶ Cicli for annidati

Cicli ed iterazioni

Istruzione for

2

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

#define MAXPAROLA 30
#define MAXIGRA 80

int main(int argc, char *argv[])
{
    int lungMAXPAROLA; /* valore di controllo
    char lungMAXIGRA; /* valore di controllo
    char riga[MAXIGRA];
    int i, indice, lungparola;
    FILE *f;

    if(argc != 2)
    {
        printf("Errore: l'unico argomento deve essere il nome del file (*.c)\n");
        exit(1);
    }

    i = fopen(argv[1], "r");
    if(i == NULL)
    {
        printf("Errore: impossibile aprire il file %s", argv[1]);
        exit(1);
    }

    while(fgets(riga, MAXIGRA, i) != NULL)
```

Istruzione for

Sintassi dell'istruzione

4

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>
```

Istruzione for

- ▶ L'istruzione fondamentale è **while**
 - La condizione solitamente valuta una variabile di controllo
 - Occorre ricordarsi l'inizializzazione della variabile di controllo
 - Occorre ricordarsi di aggiornare (incrementare, ...) la variabile di controllo
- ▶ L'istruzione **for** rende più semplice ricordare queste cose

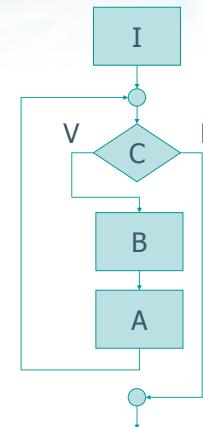
```
int lungMAXIGRA; /* valore di controllo
                    per le iterazioni nel ciclo for */

char riga[MAXIGRA]; /* valore di controllo
                      per le iterazioni nel ciclo for */

int i, indice, lungparola;
```

Istruzione for

```
for ( I; C; A )
{
    B ;
}
```



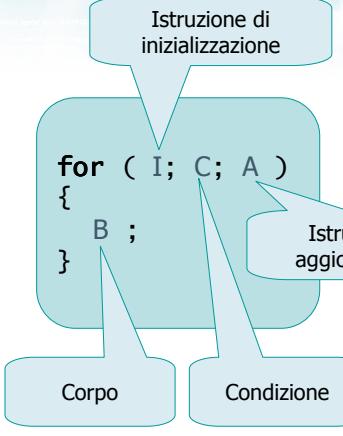
5

```
int lungMAXIGRA; /* valore di controllo
                    per le iterazioni nel ciclo for */

char riga[MAXIGRA]; /* valore di controllo
                      per le iterazioni nel ciclo for */

int i, indice, lungparola;
```

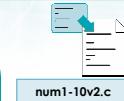
Istruzione for



6

Esempio

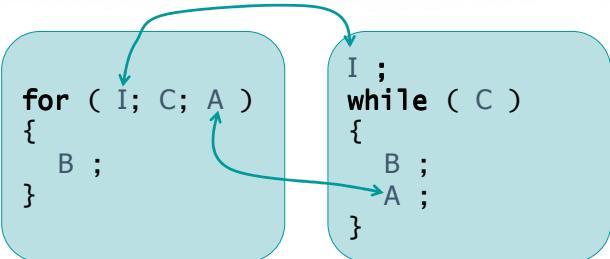
```
int i ;
for ( i=1; i<=10; i=i+1 )
{
    printf("Numero = %d\n", i) ;
}
```



num1-10v2.c

7

Equivalenza for↔while



8

Esempio

```
int i ;
for ( i=1; i<=10; i=i+1 )
{
    printf("%d\n", i) ;
}
```

```
int i ;
i = 1 ;
while ( i <= 10 )
{
    printf("%d\n", i) ;
    i = i + 1 ;
}
```

9

Utilizzo prevalente (1/2)

- » Le istruzioni di inizializzazione **I** e di aggiornamento **A** possono essere qualsiasi
- » Solitamente **I** viene utilizzata per inizializzare il contatore di controllo del ciclo, e quindi è del tipo
 - **i = 0**
- » Solitamente **A** viene utilizzata per incrementare (o decrementare) il contatore, e quindi è del tipo
 - **i = i + 1**

```
for ( I; C; A )
{
    B ;
}
```

10

Utilizzo prevalente (2/2)

- » L'istruzione **for** può sostituire un qualsiasi ciclo **while**
- » Solitamente viene utilizzata, per maggior chiarezza, nei cicli con numero di iterazioni noto a priori

11

Cicli for con iterazioni note

```
int i ;
for ( i=0; i<N; i=i+1 )
{
    .....
}
```

```
int i ;
for ( i=1; i<=N; i=i+1 )
{
    .....
}
```

```
int i ;
for ( i=N; i>0; i=i-1 )
{
    .....
}
```

```
int i ;
for ( i=N-1; i>=0; i=i-1 )
{
    .....
}
```

Casi particolari (1/6)

- Se non è necessario inizializzare nulla, si può omettere l'istruzione **I**
 - for(; i != 0 ; i = i - 1)**
- La condizione **C** viene comunque valutata prima della prima iterazione, pertanto le variabili coinvolte dovranno essere inizializzate prima dell'inizio del ciclo
- Il simbolo **;** è sempre necessario

```
for ( I; C; A )
{
    B ;
}
```

13

Casi particolari (2/6)

- Se l'aggiornamento viene fatto nel ciclo, si può omettere l'istruzione **A**
 - for(dato = INT_MIN; dato != 0 ;)**
- La responsabilità di aggiornare la variabile di controllo (**dato**) è quindi del corpo **B** del ciclo
- Il simbolo **;** è sempre necessario

```
for ( I; C; A )
{
    B ;
}
```

14

Casi particolari (3/6)

- Se occorre inizializzare più di una variabile, è possibile farlo separando le varie inizializzazioni con un simbolo ,
 - for(i=0, j=0; i<N; i=i+1)**
 - Soltamente uno solo è il contatore del ciclo, gli altri saranno altri contatori, accumulatori o flag

```
for ( I; C; A )
{
    B ;
}
```

15

Casi particolari (4/6)

- Se occorre aggiornare più di una variabile, è possibile farlo separando i vari aggiornamenti con un simbolo ,
 - for(i=0; i<N; i=i+1, k=k-1)**

```
for ( I; C; A )
{
    B ;
}
```

16

Casi particolari (5/6)

- Nel caso in cui si ometta sia **I** che **A**, il ciclo **for** degenera nel ciclo **while** equivalente
 - for(; i<N;)**
 - while(i<N)**

```
for ( I; C; A )
{
    B ;
}
```

17

Casi particolari (6/6)

- È possibile omettere la condizione **C**, in tal caso viene considerata come sempre vera
 - for(i=0; ; i=i+1)**
 - Questo costrutto genera un ciclo infinito. È necessario che il ciclo venga interrotto con un altro meccanismo (**break**, **return**, **exit**)
 - Talvolta si incontra anche un ciclo infinito "puro"
 - for(; ;)**

```
for ( I; C; A )
{
    B ;
}
```

18

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

#define MAXPAROLA 30
#define MAXIGRA 80

int main(int argc, char *argv[])
{
    int freqMAXPAROLA; /* vettore di confronto delle frequenze delle lunghezze delle parole */
    char parolaAXINGAL;
    int i, indice, lunghezza;
    FILE *fp;

    fp=fopen(argv[1], "r");
    if(fp==NULL)
    {
        fprintf(stderr, "ERRORE: non posso aprire il file %s", argv[1]);
        exit(1);
    }

    fscanf(fp, "%c", &parolaAXINGAL);
    if(isalpha(parolaAXINGAL))
    {
        lunghezza=1;
        indice=0;
        freqMAXPAROLA[0]=1;
    }
    else
    {
        lunghezza=0;
        indice=-1;
        freqMAXPAROLA[0]=0;
    }

    while(fscanf(fp, "%c", &parolaAXINGAL) != EOF)
    {
        if(isalpha(parolaAXINGAL))
        {
            freqMAXPAROLA[lunghezza]++;
            lunghezza++;
        }
        else
        {
            freqMAXIGRA[indice]++;
            indice--;
        }
    }

    printf("lunghezza parola: %d\n", lunghezza);
    printf("frequenza parola: %d\n", freqMAXIGRA[0]);
    printf("frequenza lunghezza: %d\n", freqMAXPAROLA[0]);
}
```

Istruzione for

Operatori di autoincremento

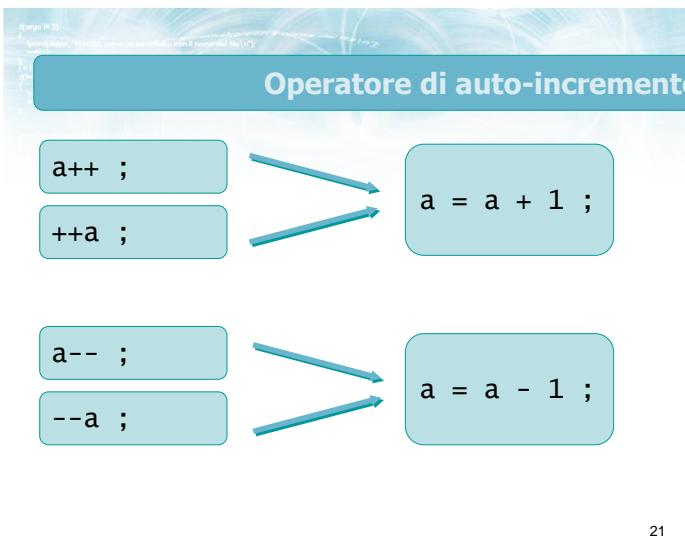


Istruzione di aggiornamento

- Nella maggioranza dei casi, l'istruzione di aggiornamento **A** consiste in un incremento
 - $i = i + 1$
- oppure in un decremento
 - $i = i - 1$
- Il linguaggio **C** dispone di operatori specifici per semplificare la sintassi di queste operazioni frequenti

```
for ( I; C; A )
{
    B ;
}
```

20



21

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

#define MAXPAROLA 30
#define MAXIGRA 80

int main(int argc, char *argv[])
{
    int freqMAXPAROLA; /* vettore di confronto delle frequenze delle lunghezze delle parole */
    char parolaAXINGAL;
    int i, indice, lunghezza;
    FILE *fp;

    fp=fopen(argv[1], "r");
    if(fp==NULL)
    {
        fprintf(stderr, "ERRORE: non posso aprire il file %s", argv[1]);
        exit(1);
    }

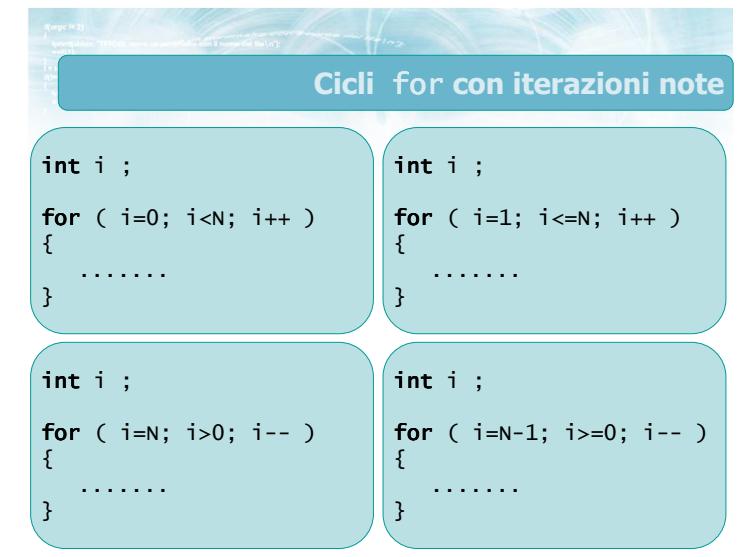
    fscanf(fp, "%c", &parolaAXINGAL);
    if(isalpha(parolaAXINGAL))
    {
        lunghezza=1;
        indice=0;
        freqMAXPAROLA[0]=1;
    }
    else
    {
        lunghezza=0;
        indice=-1;
        freqMAXPAROLA[0]=0;
    }

    while(fscanf(fp, "%c", &parolaAXINGAL) != EOF)
    {
        if(isalpha(parolaAXINGAL))
        {
            freqMAXPAROLA[lunghezza]++;
            lunghezza++;
        }
        else
        {
            freqMAXIGRA[indice]++;
            indice--;
        }
    }

    printf("lunghezza parola: %d\n", lunghezza);
    printf("frequenza parola: %d\n", freqMAXIGRA[0]);
    printf("frequenza lunghezza: %d\n", freqMAXPAROLA[0]);
}
```

Istruzione for

Cicli for annidati



Cicli for con iterazioni note

- int i ;**
- for (i=0; i<N; i++)**
- int i ;**
- for (i=1; i<=N; i++)**
- int i ;**
- for (i=N; i>0; i--)**
- int i ;**
- for (i=N-1; i>=0; i--)**



Annidamento di cicli

- Come sempre, all'interno del corpo **B** di un ciclo (**for** o **while**) è possibile annidare altri cicli (**for** o **while**)
- Non vi è limite al livello di annidamento
- I cicli più interni sono sempre eseguiti "più velocemente" dei cicli più esterni

24

Esempio

```
for( i=0; i<N; i++ )
{
    for( j=0; j<N; j++ )
    {
        printf("i=%d - j=%d\n", i, j);
    }
}
```



conta99v2.c

25

Esercizio

▶ Si scriva un programma in linguaggio C che

- acquisisca da tastiera 10 numeri
- per ciascuno di tali numeri determini se è un numero primo, stampando immediatamente un messaggio opportuno
- al termine, se nessuno tra i numeri inseriti era un numero primo, stampi un messaggio opportuno

26

Analisi (1/2)

```
Prompt dei comandi

TROVA PRIMI
Inserisci 4 numeri interi

Inserisci dato 1: 6
Inserisci dato 2: 3
E' un numero primo
Inserisci dato 3: 4
Inserisci dato 4: 5
E' un numero primo
```

27

Analisi (2/2)

```
Prompt dei comandi

TROVA PRIMI
Inserisci 4 numeri interi

Inserisci dato 1: 4
Inserisci dato 2: 6
Inserisci dato 3: 8
Inserisci dato 4: 9

Non c'erano numeri primi
```

28

Numero primo

```
primo = 1 ;
for( j=2; j<dato; j++)
{
    if( dato%j == 0 )
        primo = 0 ;
}
```



10primi.c

29

Stampa se non ci sono primi

```
for( i=0; i<n; i++ )
{
    scanf("%d", &dato);

    ....determina se è un numero primo.....

    if( primo == 1 )
    {
        printf("E' un numero primo\n");
    }
}
```



10primi.c

30

Stampa se è un primo

```

trovato = 0 ;
for( i=0; i<n; i++ )
{
    ....acquisisci dato e determina se è un numero primo.....
    if( primo == 1 )
    {
        printf("E' un numero primo\n");
        trovato = 1 ;
    }
}
if( trovato == 0 )
printf("Non ci sono primi\n") ;

```

10primi.c

31

Vista d'insieme

```

trovato = 0 ;
for( i=0; i<n; i++ )
{
    scanf("%d", &dato);

    primo = 1 ;
    for( j=2; j<dato; j++ )
    {
        if( dato%j == 0 )
            primo = 0 ;
    }

    if( primo == 1 )
    {
        printf("E' un numero primo\n");
        trovato = 1 ;
    }
}

if( trovato == 0 )
printf("Non c'erano numeri primi\n");

```

10primi.c

32

Vista d'insieme

```

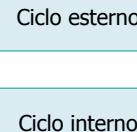
trovato = 0 ;
for( i=0; i<n; i++ )
{
    scanf("%d", &dato);

    primo = 1 ;
    for( j=2; j<dato; j++ )
    {
        if( dato%j == 0 )
            primo = 0 ;
    }

    if( primo == 1 )
    {
        printf("E' un numero primo\n");
        trovato = 1 ;
    }
}

if( trovato == 0 )
printf("Non c'erano numeri primi\n");

```



33

Vista d'insieme

```

trovato = 0 ;
for( i=0; i<n; i++ )
{
    scanf("%d", &dato);

    primo = 1 ;
    for( j=2; j<dato; j++ )
    {
        if( dato%j == 0 )
            primo = 0 ;
    }

    if( primo == 1 )
    {
        printf("E' un numero primo\n");
        trovato = 1 ;
    }
}

if( trovato == 0 )
printf("Non c'erano numeri primi\n");

```

Flag interno:
numero primo



34

Vista d'insieme

```

trovato = 0 ;
for( i=0; i<n; i++ )
{
    scanf("%d", &dato);

    primo = 1 ;
    for( j=2; j<dato; j++ )
    {
        if( dato%j == 0 )
            primo = 0 ;
    }

    if( primo == 1 )
    {
        printf("E' un numero primo\n");
        trovato = 1 ;
    }
}

if( trovato == 0 )
printf("Non c'erano numeri primi\n");

```



35

- » Istruzione do-while
- » Istruzione break
- » Istruzione continue

Cicli ed iterazioni

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

#define MAXPAROLA 30
#define MAXSIGA 80

int main(int argc, char *argv[])
{
    int lung(MAXPAROLA); /* valore di condizione
                           della frequenza delle parole */
    char testo[MAXSIGA]; /* stringa testo da leggere dalla tastiera */
    char sig(MAXSIGA);
    int i, indice, lungparola;
    i = 0;
    indice = 0;
    lungparola = 0;
    if(argc > 1)
        strcpy(testo, argv[1]);
    else
        gets(testo);
    while((lungparola = ignora_daga(sig, MAXSIGA, i)) != NULL)
```

Approfondimenti

Approfondimenti

2

```
(Merge de 2)
Salvo che non "FECCHE" serve un particolare caso è normale che l'utente inserisca un numero non compreso tra 1 e 10.
```

Confronto

- » Istruzione while
 - Condizione valutata prima di ogni iterazione
 - Numero minimo di iterazioni: 0
 - Per uscire: condizione falsa

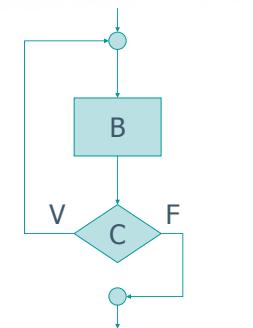
- » Istruzione do-while
 - Condizione valutata al termine di ogni iterazione
 - Numero minimo di iterazioni: 1
 - Per uscire: condizione falsa

5

```
(Merge de 2)
Salvo che non "FECCHE" serve un particolare caso è normale che l'utente inserisca un numero non compreso tra 1 e 10.
```

Esempio

- » Acquisire un numero compreso tra 1 e 10 da tastiera
- » Nel caso in cui l'utente non inserisca il numero correttamente, chiederlo nuovamente



4

Soluzione

```
printf("Numero tra 1 e 10\n");
do {
    scanf("%d", &n) ;
} while ( n<1 || n>10 ) ;
```

7

Soluzione migliore

```
printf("Numero tra 1 e 10\n");
do {
    scanf("%d", &n) ;
    if( n<1 || n>10 )
        printf("Errore: ripeti\n");
} while ( n<1 || n>10 ) ;
```

8

Esempio

- » Si scriva un programma in C che calcoli la somma di una sequenza di numeri interi
- » La sequenza termina quando l'utente inserisce il dato 9999

9

Soluzione

```
somma = 0 ;
do {
    scanf("%d", &dato) ;
    if( n != 9999 )
        somma = somma + dato ;
} while ( dato != 9999 ) ;
```

10

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

#define MAXPAROLA 30
#define MAXIGRA 80

int main(int argc, char *argv[])
{
    int lenMAXPAROLA; /* numero di caratteri
                        massimo compreso nel lunghezza delle parole */
    char strMAXIGRA[ ]; /* stringa massima */
    int i, indice, lunghezza;
    FILE *f;

    if(argc > 1)
        f = fopen(argv[1], "r");
    else
        f = stdin;
    if(f == NULL)
        printf("ERRORE: non è possibile aprire il file %s", argv[1]);
    else
        fgets(strMAXIGRA, MAXIGRA, f);
    if(strstr(strMAXIGRA, "\n") != NULL)
        strMAXIGRA[strlen(strMAXIGRA)-1] = '\0';
    else
        strMAXIGRA[strlen(strMAXIGRA)] = '\0';

    printf("Parola: %s\n", strMAXIGRA);

    if(argc > 1)
        fclose(f);
}
```

Approfondimenti

Istruzione break

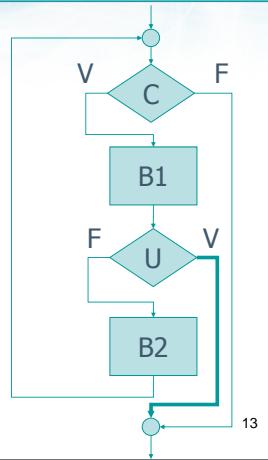
Interruzione dei cicli

- » Di norma, un ciclo termina quando la condizione di controllo diventa falsa
 - Necessario arrivare al termine del corpo per poter valutare la condizione
- » Talvolta potrebbe essere comodo interrompere prematuramente l'esecuzione di un ciclo
 - A seguito di condizioni di errore
 - Quando è stato trovato ciò che si cercava

12

Istruzione break

```
while ( C )
{
    B1 ;
    if ( U )
        break ;
    B2 ;
}
```



Funzionamento

Quando viene eseguita l'istruzione break

- Viene interrotta l'esecuzione del corpo del ciclo
- Il flusso di esecuzione passa all'esterno del ciclo che contiene il break
- Si esce dal ciclo anche se la condizione di controllo è ancora vera
- In caso di cicli annidati, si esce solo dal ciclo più interno

Funziona con cicli while, for, do-while

14

Esempio

- » Si scriva un programma in C che calcoli la somma di una sequenza di numeri interi
- » La sequenza termina quando l'utente inserisce il dato 9999

15

Esempio

- » Si scriva un programma in C che determini se un numero inserito da tastiera è primo

17

```
somma = 0 ;
do {

    scanf("%d", &dato) ;

    if( dato == 9999 )
        break;

    somma = somma + dato ;

} while ( 1 ) ;
```

16

Soluzione

```
scanf("%d", &dato) ;
primo = 1 ;
for ( i=2; i<dato; i++ )
{
    if( dato%i == 0 )
    {
        primo = 0 ;
        break ; /* inutile continuare */
    }
}
```

18

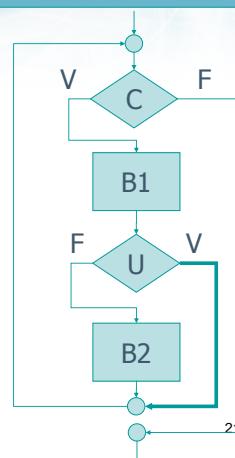
- ▶ L'istruzione break crea programmi non strutturati: usare con cautela
 - ▶ Non è possibile uscire da più cicli annidati contemporaneamente

Approfondimenti

Istruzione continue

1

Istruzione continua



```
while ( C )
{
    B1 ;
    if ( U )
        continue ;
    B2 ;
}
```

Funzionamento

- Quando viene eseguita l'istruzione **continue**
 - Viene interrotta l'esecuzione del corpo del ciclo
 - Il flusso di esecuzione passa al termine del corpo
 - Nel caso di cicli **for**, viene eseguita l'istruzione di aggiornamento
 - Viene nuovamente valutata la condizione
 - Il ciclo continua normalmente
 - In caso di cicli annidati, si esce solo dal ciclo più interno
 - Funziona con cicli **while**, **for**, **do-while**

22

Esempio

```
somma = 0 ;
do {
    scanf("%d", &dato) ;
    if( dato == 9999 )
        continue ; /* non considerarlo */
    somma = somma + dato ;
} while ( dato != 9999 ) ;
```

2

Avvertenze

- ▶ L'istruzione continue è oggettivamente poco utilizzata
 - ▶ Crea un "salto" poco visibile: accompagnarla sempre con commenti evidenti

24

Esercizi proposti

- ▶ Esercizio “Decimale-binario”
- ▶ Esercizio “Massimo Comun Divisore”
- ▶ Esercizio “Triangolo di Floyd”

Cicli ed iterazioni

Esercizi proposti

2

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>
```

```
#define MAXPAROLA 30
#define MAXDIGA 80
```

```
int main(int argc, char *argv[])
{
    int lung(MAXPAROLA); /* valore di controllo delle frequenze delle parole */
    char test[MAXPAROLA]; /* stringa testo inserita dalla tastiera */
    char digit[MAXDIGA]; /* stringa dei numeri inseriti dalla tastiera */
    int i, j, k;
```

```
for(i=0; i<MAXPAROLA; i++)
    lung[i]=0;

for(i=0; i<MAXDIGA; i++)
    digit[i]=0;

if(argc > 1)
{
    strcpy(test, argv[1]);
    if(strlen(argv[1]) > lung)
        lung = strlen(argv[1]);
}

for(j=0; j<lung; j++)
{
    i=0;
    while(test[i] != ' ' && test[i] != '\0')
    {
        if(isalpha(test[i]))
            test[i] = toupper(test[i]);
        else
            test[i] = ' ';
        i++;
    }
}
```

Esercizi proposti

Esercizio “Decimale-binario”

4

Esercizio “Decimale-binario”

- ▶ Si realizzi un programma in C in grado di
 - Leggere un numero naturale n
 - Convertire tale numero dalla base 10 alla base 2
 - Visualizzare il risultato, a partire dalla cifra meno significativa

```
{if(argc > 1)
    strcpy(test, argv[1]);
else
    test[0] = '0'};

if(strlen(test) > lung)
    lung = strlen(test);
```

Analisi

```
os: Prompt dei comandi
DECIMALE - BINARIO
Inserisci un numero intero positivo: 12
Numero binario: 0 0 1 1
```

5

```
{if(argc > 1)
    strcpy(test, argv[1]);
else
    test[0] = '0'};

if(strlen(test) > lung)
    lung = strlen(test);
```

Divisioni successive

- ▶ $n = 12$
- ▶ $n \% 2 = 0 \rightarrow$ cifra 0
- ▶ $n = n / 2 = 6$
- ▶ $n \% 2 = 0 \rightarrow$ cifra 0
- ▶ $n = n / 2 = 3$
- ▶ $n \% 2 = 1 \rightarrow$ cifra 1
- ▶ $n = n / 2 = 1$
- ▶ $n \% 2 = 1 \rightarrow$ cifra 1
- ▶ $n = n / 2 = 0 \rightarrow$ STOP

N	N % 2
12	0
6	0
3	1
1	1
0	

6

Soluzione

```

while( n!=0 )
{
    if( n%2 == 1 )
        printf("1 ");
    else
        printf("0 ");

    n = n / 2 ;
}

```

bin-dec.c

```

#include <stdio.h>
#include <string.h>
#include <ctype.h>

#define MAXPAROLA 30
#define MAXIGA 80

int main(int argc, char *argv[])
{
    int freqMAXPAROLA; /* numero di caratteri
    delle frequenze delle lunghezze delle parole */
    char strAXIGA[ ]; /* stringa AXIGA */
    int i, indice, lunghezza; /* iindice */

    for(i=0; i<MAXPAROLA; i++)
        freq[i] = 0;

    if(argc < 2)
    {
        printf("ERRORE: non è possibile calcolare il massimo del binario\n");
        exit(1);
    }
    else
    {
        if(argv[1][0] != 'f')
        {
            printf("ERRORE: l'argomento deve essere un file binario\n");
            exit(1);
        }
        else
        {
            strcpy(strAXIGA, argv[1]);
            freq[0]++;
            indice = 1;
            lunghezza = strlen(strAXIGA);
            for(i=1; i<lunghezza; i++)
            {
                if(strAXIGA[i] == '0')
                    freq[0]++;
                else
                    freq[1]++;
            }
            for(i=2; i<lunghezza; i++)
            {
                if(strAXIGA[i] == '0')
                    freq[i]++;
                else
                    freq[i+1]++;
            }
        }
    }
    printf("Il massimo del binario %s è %d\n", strAXIGA, freq[0]);
}

```

Esercizi proposti

Esercizio "Massimo Comun Divisore"

Esercizio "Massimo Comun Divisore"

- » Si scriva un programma in C in grado di calcolare il massimo comun divisore (MCD) di due numeri interi.
- » Il MCD è definito come il massimo tra i divisori comuni ai due numeri.

7

Algoritmo

bin-dec.c

- » $k_{max} = 0$
- » $\text{for } k = \text{da 1 a } N_1$
 - se k è un divisore di N_1
 - se k è un divisore di N_2
 - aggiorna $k_{max} = k$
- » $\text{MCD} = k_{max}$

9

```

#include <stdio.h>
#include <string.h>
#include <ctype.h>

#define MAXPAROLA 30
#define MAXIGA 80

int main(int argc, char *argv[])
{
    int freqMAXPAROLA; /* numero di caratteri
    delle frequenze delle lunghezze delle parole */
    char strAXIGA[ ]; /* stringa AXIGA */
    int i, indice, lunghezza; /* iindice */

    for(i=0; i<MAXPAROLA; i++)
        freq[i] = 0;

    if(argc < 2)
    {
        printf("ERRORE: non è possibile calcolare il massimo del binario\n");
        exit(1);
    }
    else
    {
        if(argv[1][0] != 'f')
        {
            printf("ERRORE: l'argomento deve essere un file binario\n");
            exit(1);
        }
        else
        {
            strcpy(strAXIGA, argv[1]);
            freq[0]++;
            indice = 1;
            lunghezza = strlen(strAXIGA);
            for(i=1; i<lunghezza; i++)
            {
                if(strAXIGA[i] == '0')
                    freq[0]++;
                else
                    freq[1]++;
            }
            for(i=2; i<lunghezza; i++)
            {
                if(strAXIGA[i] == '0')
                    freq[i]++;
                else
                    freq[i+1]++;
            }
        }
    }
    printf("Il massimo del binario %s è %d\n", strAXIGA, freq[0]);
}

```

Esercizi proposti

Analisi

- » Diciamo N_1 e N_2 i numeri inseriti dall'utente
- » Il MCD di N_1 e N_2 è il **massimo** tra i numeri che sono **divisori** sia di N_2 , sia di N_1 .
 - Troviamo i divisori di N_1 ...
 - ... tra quelli che sono anche divisori di N_2 ...
 - ... calcoliamo il massimo

10

Esercizio "Triangolo di Floyd"

11

Esercizio "Triangolo di Floyd"

- » Scrivere un programma C per la rappresentazione del triangolo di Floyd.
- » Il programma riceve da tastiera un numero interno N.
- » Il programma visualizza le prime N righe del triangolo di Floyd.

```
1  
2 3  
4 5 6  
7 8 9 10  
11 12 13 14 15
```

N=5

13

- » Occorre stampare i primi numeri interi in forma di triangolo
- » La riga k-esima ha k elementi

```
1  
2 3  
4 5 6  
7 8 9 10  
11 12 13 14 15
```

N=5

14

Algoritmo



- » cont = 1
- » for riga = da 1 a N
 - for colonna = da 1 a riga
 - stampa cont
 - cont++
 - vai a capo

```
1  
2 3  
4 5 6  
7 8 9 10  
11 12 13 14 15
```

N=5

15

```

#include <stdio.h>
#include <ctype.h>

#define MAXPARIOLA 30
#define MAXRGA 80

int main(int argc, char *argv[])
{
    int freqMAXPARIOLA; /* valore di controllo delle frequenze delle lunghezze delle parole */
    int i, min, lunghezza;
    FILE *fp;

    if(argc != 2)
    {
        fprintf(stderr, "ERRORE: non è possibile aprire il file %s", argv[1]);
        exit(1);
    }

    fp = fopen(argv[1], "r");
    if(fp == NULL)
    {
        fprintf(stderr, "ERRORE: impossibile aprire il file %s", argv[1]);
        exit(1);
    }

    for(i=0; i<MAXPARIOLA; i++)
        freqMAXPARIOLA[i] = 0;
    min = MAXRGA + 1;
    lunghezza = 0;
    while(fgets(argv[i], MAXRGA, fp) != NULL)
    {
        if(strlen(argv[i]) > lunghezza)
            lunghezza = strlen(argv[i]);
        if(argv[i][0] == 'A' || argv[i][0] == 'E' || argv[i][0] == 'I' || argv[i][0] == 'O' || argv[i][0] == 'U')
            freqMAXPARIOLA[lunghezza]++;
        else
            freqMAXPARIOLA[lunghezza] += 2;
        if(argv[i][0] < min)
            min = argv[i][0];
    }
    for(i=0; i<MAXPARIOLA; i++)
        if(freqMAXPARIOLA[i] > 0)
            printf("%c\t%d\n", min, freqMAXPARIOLA[i]);
}

```

Cicli ed iterazioni

Argomenti trattati

- Ripetizione del flusso di esecuzione
- Inizializzazione, Condizione, Aggiornamento, Corpo
- Istruzione while
- Istruzione for
- Cicli annidati

Sommario

2

Tecniche di programmazione

- Cicli con numero di iterazioni note o ignote
- Contatori
- Accumulatori
- Flag

3

Schemi ricorrenti

- Calcolo di somme, medie, ...
- Calcolo di max, min
- Ricerca di esistenza
- Ricerca di universalità
- Controllo dei dati in input

4

Suggerimenti

- Ricordare di verificare sempre le 4 parti del ciclo
 - Inizializzazione, Condizione, Corpo, Aggiornamento
- Le complicazioni nascono da
 - Cicli annidati
 - Condizioni if annidate in cicli
 - Annidamento di flag o ricerche
- Procedere sempre per gradi
 - Pseudo-codice o flow chart
 - Identificare chiaramente il ruolo dei diversi cicli

5

Materiale aggiuntivo

- Sul CD-ROM
 - Testi e soluzioni degli esercizi trattati nei lucidi
 - Scheda sintetica
 - Esercizi risolti
 - Esercizi proposti
- Esercizi proposti da altri libri di testo

6

```
#include <stdio.h>
#include <ctype.h>
#define MAXPARI 30
#define MAXIMPARI 30

int main(int argc, char *argv[])
{
    int freqMAXPAROLA; /* valore di confronto delle frequenze delle parole */
    int i, indice, lunghezza;
    FILE *f;

    if(argc != 2)
    {
        fprintf(stderr, "ERRORE: non è possibile aprire il file %s", argv[1]);
        exit(1);
    }

    f = fopen(argv[1], "r");
    if(f == NULL)
    {
        fprintf(stderr, "ERRORE: impossibile aprire il file %s", argv[1]);
        exit(1);
    }

    /* legge la parola più frequente */
    indice = 0;
    lunghezza = 0;
    freqMAXPAROLA = 0;
    while(fgets(indice, lunghezza, f) != NULL)
    {
        if(freqMAXPAROLA <= lunghezza)
        {
            freqMAXPAROLA = lunghezza;
            strcpy(indice, indice);
        }
    }

    /* legge tutte le altre parole */
    for(i = 1; i < lunghezza; i++)
    {
        if(strcmp(indice, fgets(i, lunghezza - i, f)) == 0)
        {
            freqMAXPAROLA++;
        }
    }

    /* stampa risultato */
    printf("La parola più frequente è %s (%d)\n", indice, freqMAXPAROLA);
}
```

Programmazione in C

Unità Vettori

2

- » Strutture dati complesse
- » I vettori in C
- » Operazioni elementari sui vettori
- » Esercizi guidati sui vettori
- » Sommario

Riferimenti al materiale

» Testi

- Kernighan & Ritchie: capitoli 1 e 5
- Cabodi, Quer, Sonza Reorda: capitolo 5
- Dietel & Dietel: capitolo 6

» Dispense

- Scheda: "Vettori in C"

3

```
#include <stdio.h>
#include <math.h>
#include <ctype.h>

#define MAXPAROLA 30
#define MAXIGA 80

int main(int argc, char *argv[])
{
    int lungMAXPAROLA; /* valore di costante
        delle frequenze delle lunghezze delle parole */
    int lungMAXIGA; /* valore di costante
        delle frequenze delle lunghezze delle parole */
    int i, indice, lungMAXI;

    if(argc != 2)
    {
        fprintf(stderr, "ERRORE: non è possibile aprire il file %s", argv[1]);
        exit(1);
    }

    if(indice = fopen(argv[1], "r")) // apre il file
    {
        fscanf(indice, "%d", &l lungMAXPAROLA); // legge la lunghezza massima delle parole
        fscanf(indice, "%d", &l lungMAXIGA); // legge la lunghezza massima delle righe
        fscanf(indice, "%c", &l lungMAXI); // legge la lunghezza massima delle righe
    }
    else
    {
        fprintf(stderr, "ERRORE: impossibile aprire il file %s", argv[1]);
        exit(1);
    }

    while(fgets(diga, MAXIGA, i) != NULL)
    {
        // qui va il codice per analizzare la riga
    }
}
```

Vettori

Strutture dati complesse

```
#include <stdio.h>
#include <math.h>
#include <ctype.h>

#define MAXPAROLA 30
#define MAXIGA 80

int main(int argc, char *argv[])
{
    int lungMAXPAROLA; /* valore di costante
        delle frequenze delle lunghezze delle parole */
    int lungMAXIGA; /* valore di costante
        delle frequenze delle lunghezze delle parole */
    int i, indice, lungMAXI;

    if(argc != 2)
    {
        fprintf(stderr, "ERRORE: non è possibile aprire il file %s", argv[1]);
        exit(1);
    }

    if(indice = fopen(argv[1], "r")) // apre il file
    {
        fscanf(indice, "%d", &l lungMAXPAROLA); // legge la lunghezza massima delle parole
        fscanf(indice, "%d", &l lungMAXIGA); // legge la lunghezza massima delle righe
        fscanf(indice, "%c", &l lungMAXI); // legge la lunghezza massima delle righe
    }
    else
    {
        fprintf(stderr, "ERRORE: impossibile aprire il file %s", argv[1]);
        exit(1);
    }

    while(fgets(diga, MAXIGA, i) != NULL)
    {
        // qui va il codice per analizzare la riga
    }
}
```

Strutture dati complesse

Tipi di dato strutturati

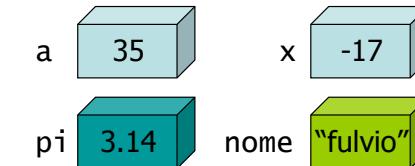


- Tipi di dato strutturati
- Introduzione ai vettori
- Caratteristiche dei vettori

2

Tipi di dato strutturati

- Finora abbiamo utilizzato dei tipi di dato semplici
 - **int, float**
 - Ogni variabile può contenere **un solo valore**
- Il linguaggio C permette di definire tipi di dato complessi, aggregando più variabili semplici



4

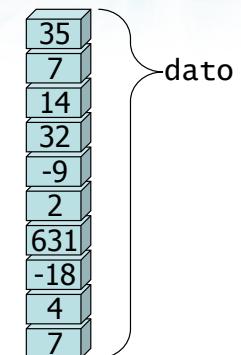
Esigenze

- Leggere da tastiera 10 numeri, e stamparli in ordine inverso
- Calcolare e stampare la tavola pitagorica
- Calcolare l'area del triangolo date le coordinate dei vertici
- Per ogni auto calcolare il tempo medio e minimo sul giro

5

Esigenze

- Leggere da tastiera 10 numeri, e stamparli in ordine inverso
- Calcolare e stampare la tavola pitagorica
- Calcolare l'area del triangolo date le coordinate dei vertici
- Per ogni auto calcolare il tempo medio e minimo sul giro



6

Esigenze

- ▶ Leggere da tastiera 10 numeri, e stamparli in ordine inverso
- ▶ Calcolare e stampare la tavola pitagorica
- ▶ Calcolare l'area del triangolo date le coordinate dei vertici
- ▶ Per ogni auto calcolare il tempo medio e minimo sul giro

pitagora				
1	2	3	4	5
2	4	6	8	10
3	6	9	12	15
4	8	12	16	20
5	10	15	20	25
6	12	18	24	30

7

Esigenze

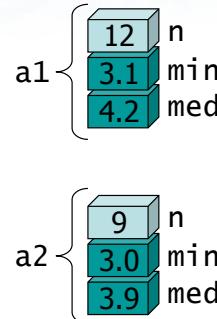
- ▶ Leggere da tastiera 10 numeri, e stamparli in ordine inverso
- ▶ Calcolare e stampare la tavola pitagorica
- ▶ Calcolare l'area del triangolo date le coordinate dei vertici
- ▶ Per ogni auto calcolare il tempo medio e minimo sul giro

A	{	3.1	x
		0.2	y
B	{	4.0	x
		-2.1	y
C	{	7.5	x
		3.3	y

8

Esigenze

- ▶ Leggere da tastiera 10 numeri, e stamparli in ordine inverso
- ▶ Calcolare e stampare la tavola pitagorica
- ▶ Calcolare l'area del triangolo date le coordinate dei vertici
- ▶ Per ogni auto calcolare il tempo medio e minimo sul giro



9

Dati strutturati (1/2)

- ▶ Raggruppare più variabili semplici in un'unica struttura complessa
- ▶ Diversi meccanismi di aggregazione
 - Vettore
 - Matrice
 - Struttura

10

Dati strutturati (2/2)

- ▶ Una sola variabile, di tipo complesso, memorizza **tutti** i dati della struttura complessa
 - Vettore_di_int dato
 - Matrice_di_int pitagora
 - Struttura_xy A, B, C
 - Struttura_auto a1, a2

```

#include <iostream.h>
#include <string.h>
#include <ctype.h>

#define MAXPAROLA 30
#define MAXIGRA 80

int main(int argc, char *argv[])
{
    int freqMAXPAROLA; /* variabile di controllo delle frequenze delle parole */
    char lineaAXIGRA[ ]; /* linea AXIGRA */
    int i, indice, lunghezza; /* iindice */
    FILE *f; /* file */

    for(i=0; i<MAXPAROLA; i++)
        freq[i]=0;

    f=fopen(argv[1], "r");
    if(f==NULL)
    {
        cout<<"ERRORE: impossibile aprire il file "<<argv[1]<<endl;
        exit(1);
    }

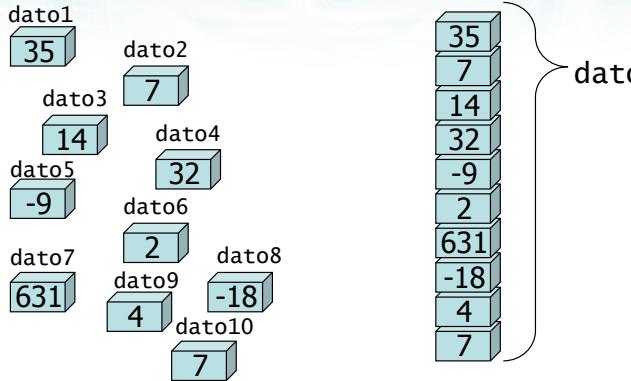
    while(fgets(lineaAXIGRA, MAXIGRA, f)!=NULL)
    {
        for(i=0; lineaAXIGRA[i]; i++)
            lineaAXIGRA[i]=tolower(lineaAXIGRA[i]);
    }
}
  
```

Strutture dati complesse

Introduzione ai vettori

11

Variabili e vettori



13

Da evitare...

```
int main(void)
{
    int dato1, dato2, dato3, dato4, dato5 ;
    int dato6, dato7, dato8, dato9, dato10 ;
    . . .
    scanf("%d", &dato1) ;
    scanf("%d", &dato2) ;
    scanf("%d", &dato3) ;
    . . .
    scanf("%d", &dato10) ;

    printf("%d\n", dato10) ;
    printf("%d\n", dato9) ;
    printf("%d\n", dato8) ;
    . . .
    printf("%d\n", dato1) ;
```



14

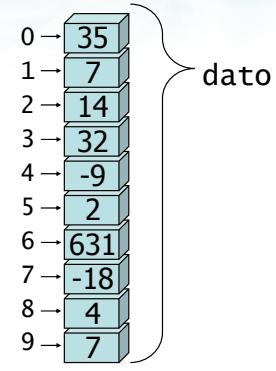
Vettori

- » Meccanismo di strutturazione più semplice
- » Utilizzato per aggregare serie di dati
- » Permette di memorizzare tutti i dati acquisiti o calcolati, e potervi accedere
 - In qualsiasi momento
 - In qualsiasi ordine
- » I singoli dati sono distinti dal loro numero d'ordine
 - Primo, secondo, ..., ultimo dato
 - Ciascun dato può avere un valore diverso

15

Vettori

- » Sequenza lineare di dati elementari
- » Elementi tutti dello stesso tipo
- » Numero di elementi fisso (N)
- » Elementi identificati dal proprio indice
 - da 0 a $N-1$



16

...così è meglio!

```
int main(void)
{
    int dato[10] ;
    for( i=0; i<10; i++)
        scanf("%d", &dato[i]) ;
    for( i=9; i>=0; i--)
        printf("%d\n", dato[i]) ;
}
```



17

Insieme o separati?

- » I singoli dati vengono tenuti insieme in un'unica variabile di tipo vettore
 - `int dato[10]`
- » Le operazioni (lettura, scrittura, elaborazione) avvengono però sempre un dato alla volta
 - `scanf("%d", &dato[i])`
 - `printf("%d\n", dato[i])`
- » Ogni singolo dato è identificato da
 - Nome del vettore
 - Posizione all'interno del vettore

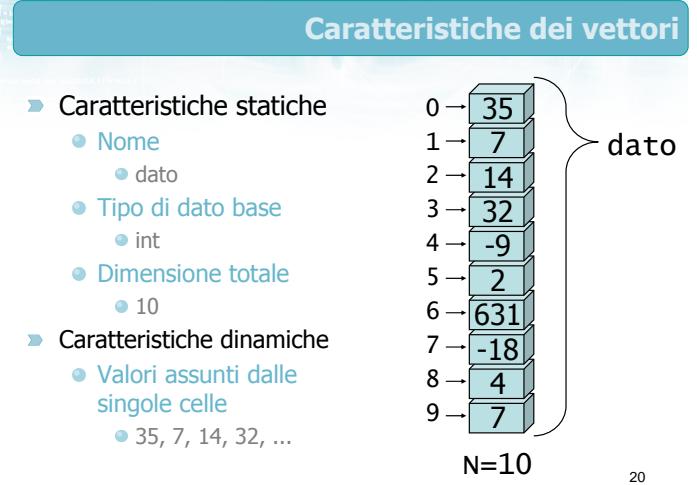
18

```
#include <stdio.h>
#include <ctype.h>
#define MAXPOLA 30
#define MAXRGA 60
```

int main(int argc, char *argv[])
{
 int freq[MAXPOLA]; /* vettore di contenuto delle frequenze delle piazze */
 int i, min, max;
 if (argc != 2) {
 fprintf(stderr, "ERRORE: inserire un percorso valido il numero del file (*.txt)\n");
 exit(1);
 }
 if (fopen(argv[1], "r") == NULL) {
 fprintf(stderr, "ERRORE: impossibile aprire il file %s", argv[1]);
 exit(1);
 }
 min = freq[0];
 max = freq[0];
 for (i = 1; i < MAXPOLA; i++) {
 if (freq[i] < min)
 min = freq[i];
 if (freq[i] > max)
 max = freq[i];
 }
 printf("min=%d, max=%d, i=%d\n", min, max, i);
}

Strutture dati complesse

Caratteristiche dei vettori



20

Esempio

- Leggere da tastiera 10 numeri e stamparli in ordine inverso
 - Tipo base: int
 - Dimensione: 10
 - Lettura delle celle da 0 a 9
 - Stampa delle celle da 9 a 0

21

Accesso alle celle

- Ciascuna cella è identificata dal proprio **indice**
- Gli indici sono sempre numeri **interi**
 - In C, gli indici partono da **0**
- Ogni cella è a tutti gli effetti una **variabile** il cui tipo è pari al tipo base del vettore
- Ogni cella, indipendentemente dalle altre
 - deve essere **inizializzata**
 - può essere **letta/stampata**
 - può essere **aggiornata** da istruzioni di **assegnazione**
 - può essere usata in espressioni aritmetiche

23

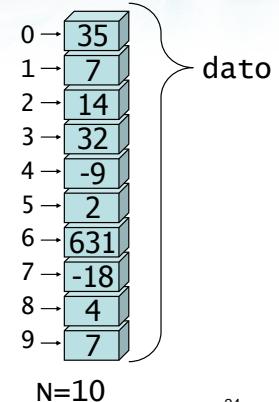
Vincoli

- Tutte le celle di un vettore avranno lo **stesso nome**
- Tutte le celle di un vettore devono avere lo **stesso tipo base**
- La dimensione del vettore è **fissa** e deve essere determinata al momento della sua definizione
 - La dimensione è **sempre un numero intero**
- Ogni cella ha **sempre** un valore
 - Impossibile avere celle "vuote"
 - Le celle non inizializzate contengono valori ignoti

22

Errore frequente

- Non confondere mai l'indice con il contenuto
 - dato[5] = 2
 - dato[9] == dato[1]
 - dato[i]>dato[j]
 - i>j

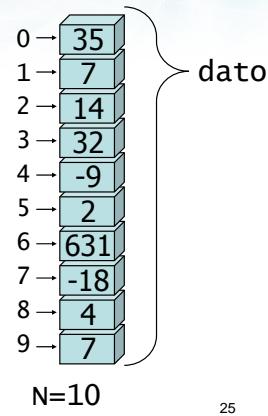


24



Errore frequente

- » **Non si può**
effettuare alcuna
operazione sul
vettore
 - `dato = 0`
 - `printf("%d", dato)`
- » Occorre operare sui
singoli elementi
 - Solitamente
all'interno di un ciclo
`for`



25

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

#define MAXPAROLA 30
#define MAXSIGA 80

int main(int argc, char *argv[])
{
    int lung(MAXPAROLA); /* vettore di costante
                           delle frequenze delle lunghezze delle parole */
    char sig(MAXSIGA); /* vettore di costante
                           delle frequenze delle sigle */
    char dpm[MAXSIGA];
    int i, lunga;
    FILE *fptr;

    if(argc != 2)
    {
        fprintf(stderr, "ERRORE: non è possibile aprire il file %s", argv[1]);
        exit(1);
    }

    if((fptr = fopen(argv[1], "r")) == NULL)
    {
        fprintf(stderr, "ERRORE: impossibile aprire il file %s", argv[1]);
        exit(1);
    }

    for(i=0; i<MAXSIGA; i++)
        dpm[i] = '\0';

    while(fgets(sig, MAXSIGA, fptr) != NULL)
    {
        if(sig[0] == 'A' || sig[0] == 'E' || sig[0] == 'I' || sig[0] == 'O' || sig[0] == 'U')
            lunga++;
        else
            lunga = 0;
        if(lunga > lung)
            lung = lunga;
    }

    printf("lung: %d\n", lung);
    printf("sig: %s", sig);

    exit(0);
}
```

Vettori

I vettori in C

2

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

#define MAXPAROLA 30
#define MAXSIGA 80

int main(int argc, char *argv[])
{
    int lung(MAXPAROLA); /* vettore di costante
                           delle frequenze delle lunghezze delle parole */
    char sig(MAXSIGA); /* vettore di costante
                           delle frequenze delle sigle */
    char dpm[MAXSIGA];
    int i, lunga;
    FILE *fptr;

    if(argc != 2)
    {
        fprintf(stderr, "ERRORE: non è possibile aprire il file %s", argv[1]);
        exit(1);
    }

    if((fptr = fopen(argv[1], "r")) == NULL)
    {
        fprintf(stderr, "ERRORE: impossibile aprire il file %s", argv[1]);
        exit(1);
    }

    for(i=0; i<MAXSIGA; i++)
        dpm[i] = '\0';

    while(fgets(sig, MAXSIGA, fptr) != NULL)
    {
        if(sig[0] == 'A' || sig[0] == 'E' || sig[0] == 'I' || sig[0] == 'O' || sig[0] == 'U')
            lunga++;
        else
            lunga = 0;
        if(lunga > lung)
            lung = lunga;
    }

    printf("lung: %d\n", lung);
    printf("sig: %s", sig);

    exit(0);
}
```

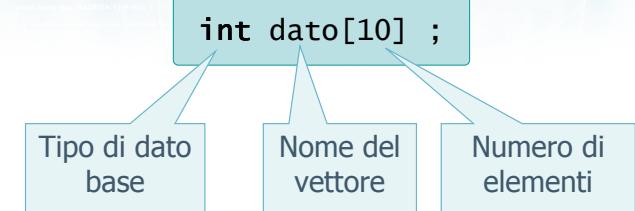
I vettori in C

Sintassi della definizione

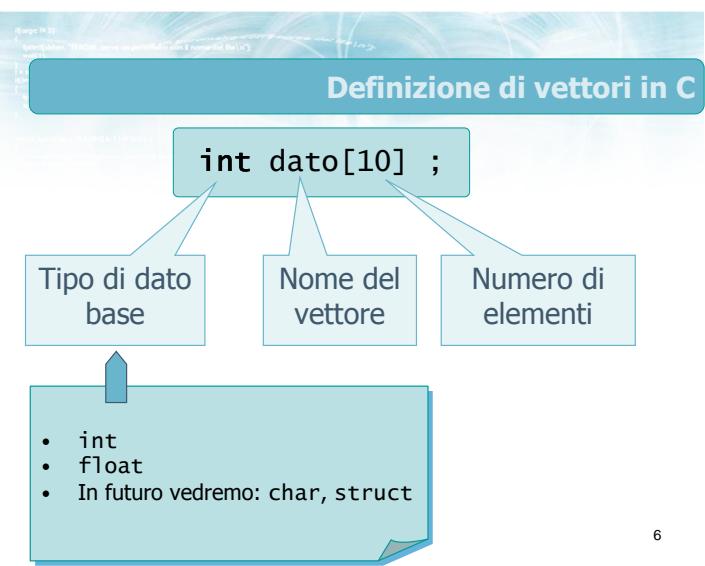
4

- Sintassi della definizione
- Definizione di costanti
- Operazioni di accesso

Definizione di vettori in C



5



6

Definizione di vettori in C

```
int dato[10] ;
```

Tipo di dato base Nome del vettore Numero di elementi

- Intero positivo
- Costante nota a tempo di compilazione
- Impossibile cambiarla in seguito

7

Esempi

```
int dato[10] ;  
float lati[8] ;  
int numeriprimi[100] ;  
int is_primo[1000] ;
```

8

Esempi

```
int dato[10] ;  
float lati[8] ;  
int numeriprimi[100] ;  
int is_primo[1000] ;
```

0	1
1	2
2	3
3	5
4	7
5	11
6	13
7	17
8	19
9	23
10	29
11	31
12	37
...	...

9

Esempi

```
int dato[10] ;  
float lati[8] ;  
int numeriprimi[100] ;  
int is_primo[1000] ;
```

0	0
1	1
2	1
3	1
4	0
5	0
6	0
7	1
8	0
9	0
10	0
11	1
12	0
...	...

10



Errore frequente

- Dichiare un vettore usando una variabile anziché una costante

```
int N = 10 ;  
int dato[N] ;
```

11

Errore frequente

- Dichiare un vettore usando una variabile non ancora inizializzata

```
int N ;  
int dato[N] ;  
...  
scanf("%d", &N) ;
```

12



Errore frequente

- Dichiarare un vettore usando il nome dell'indice

```
int i ;
int dato[i] ; // Errore: dato è un indice, non una costante

for(i=0; i<10; i++)
    scanf("%d", &dato[i]) ;

int dato[10] ; // Soluzione: dichiarare la dimensione del vettore
```

13

I vettori in C

Definizione di costanti

- La dimensione di un vettore deve essere specificata utilizzando una costante intera positiva

- Costante** = valore numerico già noto al momento della compilazione del programma

```
int dato[10] ;
```

15

Costanti

```
int i ;
int dato[10] ; // Errore: dato è un indice, non una costante

. . .

for(i=0; i<10; i++)
    scanf("%d", &dato[i]) ;

for(i=0; i<10; i++)
    printf("%d\n", dato[i]) ;
```

17

Se volessi lavorare con 20 dati dovrei modificare in tutti questi punti.

```
int i ;
int dato[10] ; // Errore: dato è un indice, non una costante

. . .

for(i=0; i<10; i++)
    scanf("%d", &dato[i]) ;

for(i=0; i<10; i++)
    printf("%d\n", dato[i]) ;
```

16

Problemi

Devono essere tutte uguali.
Chi lo garantisce?

```
int i ;
int dato[10] ; // Errore: dato è un indice, non una costante

. . .

for(i=0; i<10; i++)
    scanf("%d", &dato[i]) ;

for(i=0; i<10; i++)
    printf("%d\n", dato[i]) ;
```

18

Problemi

- Per risolvere i problemi visti si può ricorrere alle **costanti simboliche**

- Associamo un nome simbolico ad una costante
- Nel programma usiamo sempre il nome simbolico
- Il compilatore si occuperà di sostituire, ad ogni occorrenza del nome, il valore numerico della costante

Costanti simboliche

Costrutto #define

- Metodo originario, in tutte le versioni del C
- Usa una sintassi particolare, diversa da quella del C
- Definisce costanti valide su tutto il file
- Non specifica il tipo della costante

Modificatore const

- Metodo più moderno, nelle versioni recenti del C
- Usa la stessa sintassi di definizione delle variabili
- Specifica il tipo della costante

19

Costrutto #define

```
#define N 10  
int main(void)  
{  
    int dato[N];  
    . . .  
}
```

Definizione della costante

Uso della costante

20

Particularità (1/2)

- » La definizione non è terminata dal segno ;
- » Tra il nome della costante ed il suo valore vi è solo uno spazio (non il segno =)
- » Le istruzioni #define devono essere una per riga
- » Solitamente le #define vengono poste subito dopo le #include

```
#define N 10
```

21

Particularità (2/2)

- » Non è possibile avere una #define ed una variabile con lo stesso nome
- » Per convenzione, le costanti sono indicate da nomi TUTTI_MAIUSCOLI

```
#define N 10
```

22

Esempio

```
#define MAX 10  
int main(void)  
{  
    int i;  
    int dato[MAX];  
    . . .  
    for(i=0; i<MAX; i++)  
        scanf("%d", &dato[i]);  
    for(i=0; i<MAX; i++)  
        printf("%d\n", dato[i]);  
}
```

23

Modificatore const

```
int main(void)  
{  
    const int N = 10;  
  
    int dato[N];  
    . . .  
}
```

Definizione della costante

Uso della costante

24

Sintassi

- » Stessa sintassi per dichiarare una variabile
- » Parola chiave **const**
- » Valore della costante specificato dal segno =
- » Definizione terminata da segno ;
- » Necessario specificare il tipo (es. **int**)
- » Il valore di N non si può più cambiare

```
const int N = 10 ;
```

25

26

```
int main(void)
{
    const int MAX = 10 ;
    int i ;
    int dato[MAX] ;

    . . .

    for(i=0; i<MAX; i++)
        scanf("%d", &dato[i]) ;

    for(i=0; i<MAX; i++)
        printf("%d\n", dato[i]) ;
}
```



Suggerimenti

- » Utilizzare **sempre** il modificatore **const**
- » Permette maggiori controlli da parte del compilatore
- » Gli eventuali messaggi d'errore sono più chiari
- » Aggiungere **sempre** un commento per indicare lo scopo della variabile
- » Utilizzare la convenzione di assegnare nomi **TUTTI_MAIUSCOLI** alle costanti

27

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

#define MAXPAROLA 30
#define MAXIGA 80

int main(int argc, char *argv[])
{
    int lungMAXPAROLA; /* variezza di parola */
    const int lungMAXIGA; /* lunghezza max stringa */
    char riga[MAXIGA];
    int i, j, lungMAX;
    FILE *f;

    f=fopen(argv[1], "r");
    if(f==NULL)
    {
        printf("ERRORE, non esiste il file %s\n", argv[1]);
        exit(1);
    }

    while(fgets(riga, MAXIGA, f) != NULL)
    {
        i=0;
        while(riga[i] != '\n' && i<lungMAXPAROLA)
        {
            j=i;
            while(isalpha(riga[j])) /* legge una parola */
                j++;
            if(j-i>lungMAXPAROLA)
                break;
            else
                printf("%s\n", &riga[i]);
            i=j;
        }
    }
}
```

I vettori in C

Operazioni di accesso

Accesso ai valori di un vettore

- » Ciascun elemento di un vettore è equivalente ad una singola variabile avente lo stesso tipo base del vettore
- » È possibile accedere al contenuto di tale variabile utilizzando l'operatore di **indicizzazione**: []

dato[0]	→	35
dato[1]	→	7
dato[2]	→	14
dato[3]	→	32
dato[4]	→	-9
dato[5]	→	2
dato[6]	→	631
dato[7]	→	-18
dato[8]	→	4
dato[9]	→	7

```
int dato[10] ;
```

29

Sintassi

nomevettore[valoreindice]

Come nella dichiarazione

Valore intero compreso tra 0 e (dimensione del vettore - 1)

Costante, variabile o espressione aritmetica con valore intero

30

Vincoli

- » Il valore dell'indice deve essere compreso tra 0 e N-1. La responsabilità è del programmatore
- » Se l'indice non è un numero intero, viene automaticamente troncato
- » Il nome di un vettore può essere utilizzato solamente con l'operatore di indicizzazione

31

Uso di una cella di un vettore

- » L'elemento di un vettore è utilizzabile come una qualsiasi variabile:
 - utilizzabile all'interno di un'espressione
 - tot = tot + dato[i] ;
 - utilizzabile in istruzioni di assegnazione
 - dato[0] = 0 ;
 - utilizzabile per stampare il valore
 - printf("%d\n", dato[k]) ;
 - utilizzabile per leggere un valore
 - scanf("%d\n", &dato[k]) ;

32

Esempi

- » if (dato[i]==0)
 - se l'elemento contiene zero
- » if (dato[i]==dato[i+1])
 - due elementi consecutivi uguali
- » dato[i] = dato[i] + 1 ;
 - incrementa l'elemento i-esimo
- » dato[i] = dato[i+1] ;
 - copia un dato dalla cella successiva

33

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

#define MAXPAROLA 30
#define MAXRIGA 80

int main(int argc, char *argv[])
{
    int lungMAXPAROLA; /* vettore di costante
        delle frequenze delle lunghezze delle parole */
    char parola[MAXPAROLA];
    int i, indice, lunghezza;
    FILE *f;

    f=fopen(argv[1], "r");
    if(f==NULL)
    {
        fprintf(stderr, "ERRORE: non è possibile aprire il file %s", argv[1]);
        exit(1);
    }

    for(i=0; i<MAXPAROLA; i++)
        lungMAXPAROLA[i]=0;

    while(fgets(parola, MAXRIGA, f) != NULL)
    {
        lunghezza = strlen(parola);
        if(lunghezza > MAXPAROLA)
            lungMAXPAROLA[lunghezza-1]++;

        for(i=0; i<lunghezza; i++)
            if(isalpha(parola[i]))
                lungMAXPAROLA[parola[i]-65]++;
    }

    if(argc > 2)
    {
        if(strcmp(argv[2], "lunghezza") == 0)
        {
            for(i=0; i<MAXPAROLA; i++)
                printf("%d\t%d\n", i, lungMAXPAROLA[i]);
        }
        else if(strcmp(argv[2], "freq") == 0)
        {
            for(i=0; i<MAXPAROLA; i++)
                if(lungMAXPAROLA[i] > 0)
                    printf("%c\t%d\n", i+65, lungMAXPAROLA[i]);
        }
        else
            fprintf(stderr, "ERRORE: impreciso operazione %s\n", argv[2]);
    }
    else
        printf("lunghezza\tfreq\n");

    fclose(f);
}
```

Vettori

Operazioni elementari sui vettori

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

#define MAXPAROLA 30
#define MAXRIGA 80

int main(int argc, char *argv[])
{
    int lungMAXPAROLA; /* vettore di costante
        delle frequenze delle lunghezze delle parole */
    char parola[MAXPAROLA];
    int i, indice, lunghezza;
    FILE *f;

    f=fopen(argv[1], "r");
    if(f==NULL)
    {
        fprintf(stderr, "ERRORE: non è possibile aprire il file %s", argv[1]);
        exit(1);
    }

    for(i=0; i<MAXPAROLA; i++)
        lungMAXPAROLA[i]=0;

    while(fgets(parola, MAXRIGA, f) != NULL)
    {
        lunghezza = strlen(parola);
        if(lunghezza > MAXPAROLA)
            lungMAXPAROLA[lunghezza-1]++;

        for(i=0; i<lunghezza; i++)
            if(isalpha(parola[i]))
                lungMAXPAROLA[parola[i]-65]++;
    }

    if(argc > 2)
    {
        if(strcmp(argv[2], "lunghezza") == 0)
        {
            for(i=0; i<MAXPAROLA; i++)
                printf("%d\t%d\n", i, lungMAXPAROLA[i]);
        }
        else if(strcmp(argv[2], "freq") == 0)
        {
            for(i=0; i<MAXPAROLA; i++)
                if(lungMAXPAROLA[i] > 0)
                    printf("%c\t%d\n", i+65, lungMAXPAROLA[i]);
        }
        else
            fprintf(stderr, "ERRORE: impreciso operazione %s\n", argv[2]);
    }
    else
        printf("lunghezza\tfreq\n");

    fclose(f);
}
```

Operazioni elementari sui vettori

Definizioni

```
const int N = 10 ;
/* dimensioni dei vettori */

int v[N] ; /* vettore di N interi */

float r[N] ;
/* vettore di N reali */

int i, j ;
/* indici dei cicli */
```

Definizioni (2/2)

vettori.c

5

```
(Argomento: Operazioni sui vettori - Definizioni)
Operazioni sui vettori - Definizioni
```

Operazioni elementari sui vettori

- ▶ Definizioni
- ▶ Stampa di un vettore
- ▶ Lettura di un vettore
- ▶ Copia di un vettore
- ▶ Ricerca di un elemento
- ▶ Ricerca del massimo o minimo
- ▶ Vettori ad occupazione variabile

2

```
(Argomento: Operazioni sui vettori - Definizioni)
Operazioni sui vettori - Definizioni
```

Definizioni (1/2)

```
const int N = 10 ;
/* dimensioni dei vettori */

int v[N] ; /* vettore di N interi */

float r[N] ;
/* vettore di N reali */

int i, j ;
/* indici dei cicli */
```

vettori.c

4

```
(Argomento: Operazioni sui vettori - Definizioni)
Operazioni sui vettori - Definizioni
```

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

#define MAXPAROLA 30
#define MAXRIGA 80

int main(int argc, char *argv[])
{
    int lungMAXPAROLA; /* vettore di costante
        delle frequenze delle lunghezze delle parole */
    char parola[MAXPAROLA];
    int i, indice, lunghezza;
    FILE *f;

    f=fopen(argv[1], "r");
    if(f==NULL)
    {
        fprintf(stderr, "ERRORE: non è possibile aprire il file %s", argv[1]);
        exit(1);
    }

    for(i=0; i<MAXPAROLA; i++)
        lungMAXPAROLA[i]=0;

    while(fgets(parola, MAXRIGA, f) != NULL)
    {
        lunghezza = strlen(parola);
        if(lunghezza > MAXPAROLA)
            lungMAXPAROLA[lunghezza-1]++;

        for(i=0; i<lunghezza; i++)
            if(isalpha(parola[i]))
                lungMAXPAROLA[parola[i]-65]++;
    }

    if(argc > 2)
    {
        if(strcmp(argv[2], "lunghezza") == 0)
        {
            for(i=0; i<MAXPAROLA; i++)
                printf("%d\t%d\n", i, lungMAXPAROLA[i]);
        }
        else if(strcmp(argv[2], "freq") == 0)
        {
            for(i=0; i<MAXPAROLA; i++)
                if(lungMAXPAROLA[i] > 0)
                    printf("%c\t%d\n", i+65, lungMAXPAROLA[i]);
        }
        else
            fprintf(stderr, "ERRORE: impreciso operazione %s\n", argv[2]);
    }
    else
        printf("lunghezza\tfreq\n");

    fclose(f);
}
```

Operazioni elementari sui vettori

Stampa di un vettore

Stampa di un vettore

- Occorre stampare un elemento per volta, all'interno di un ciclo `for`
- Ricordare che
 - gli indici del vettore variano tra 0 e $N-1$
 - gli utenti solitamente contano tra 1 e N
 - $v[i]$ è l'elemento $(i+1)$ -esimo

7

Stampa vettore di interi

```
printf("vettore di %d interi\n", N) ;  
  
for( i=0; i<N; i++ )  
{  
    printf("Elemento %d: ", i+1) ;  
    printf("%d\n", v[i]) ;  
}
```

8

Stampa vettore di interi

```
printf("vettore di %d interi\n", N) ;  
  
for( i=0; i<N; i++ )  
{  
    printf("Elemento %d: ", i+1) ;  
    printf("%d\n", v[i]) ;  
}
```

Pronto dei comandi

```
Stampa di un vettore di 10 interi  
Elemento 1: 3  
Elemento 2: 4  
Elemento 3: 7  
Elemento 4: 5  
Elemento 5: 3  
Elemento 6: -1  
Elemento 7: -3  
Elemento 8: 2  
Elemento 9: 7  
Elemento 10: 3
```

9

Stampa in linea

```
printf("vettore di %d interi\n", N) ;  
  
for( i=0; i<N; i++ )  
{  
    printf("%d ", v[i]) ;  
}  
printf("\n") ;
```

10

Stampa in linea

```
printf("vettore di %d interi\n", N) ;  
  
for( i=0; i<N; i++ )  
{  
    printf("%d ", v[i]) ;  
}  
printf("\n") ;
```

Pronto dei comandi

```
Stampa di un vettore di 10 interi  
3 4 7 5 3 -1 -3 2 7 3
```

11

Stampa vettore di reali

```
printf("vettore di %d reali\n", N) ;  
  
for( i=0; i<N; i++ )  
{  
    printf("Elemento %d: ", i+1) ;  
    printf("%f\n", r[i]) ;  
}
```

12

Avvertenze

- » Anche se il vettore è di reali, l'indice è sempre intero
- » Separare sempre i vari elementi almeno con uno spazio (o un a capo)

Operazioni elementari sui vettori

Lettura di un vettore

13

Lettura di un vettore

- » Occorre leggere un elemento per volta, all'interno di un ciclo for
- » Ricordare l'operatore & nella scanf

15

Lettura vettore di interi

```
printf("Lettura di %d interi\n", N) ;  
for( i=0; i<N; i++ )  
{  
    printf("Elemento %d: ", i+1) ;  
    scanf("%d", &v[i]) ;  
}
```

16

Lettura vettore di interi

```
printf("Lettura di %d interi\n", N) ;  
for( i=0; i<N; i++ )  
{  
    printf("Elemento %d: ", i+1) ;  
    scanf("%d", &v[i]) ;  
}  
  
Lettura di un vettore di 10 interi  
Elemento 1: 3  
Elemento 2: 4  
Elemento 3: 7  
Elemento 4: 5  
Elemento 5: 3  
Elemento 6: -1  
Elemento 7: -3  
Elemento 8: 2  
Elemento 9: 7  
Elemento 10: 3
```

17

Lettura vettore di reali

```
printf("Lettura di %d reali\n", N) ;  
for( i=0; i<N; i++ )  
{  
    printf("Elemento %d: ", i+1) ;  
    scanf("%f", &r[i]) ;  
}
```

18

Avvertenze

- » Anche se il vettore è di reali, l'indice è sempre intero
- » Fare precedere sempre ogni lettura da una printf esplicativa

```
#include <stdio.h>
#include <math.h>
#include <ctype.h>

#define MAXPAROLA 80
#define MAXSIG 60

int main(int argc, char *argv[])
{
    int lungMAXPAROLA; /* valore di costante
        detta frequenza delle lunghezze delle parole */
    int lungMAXSIG; /* lunghezza di
        un singolo carattere */

    if(argc != 2)
    {
        fprintf(stderr, "ERRORE: non sono presenti dati di input dal file %s\n", argv[0]);
        exit(1);
    }
    lungMAXPAROLA = atoi(argv[1]);
    lungMAXSIG = 1;
    for(int i = 0; i < lungMAXSIG; i++)
    {
        if(isupper(argv[1][i])) /* se è maiuscolo */
            lungMAXSIG++;
    }
    if(lungMAXSIG > 1)
        fprintf(stderr, "ERRORE: impossibile aprire il file %s (%c)\n", argv[1], argv[1][1]);
    exit(1);
}
```

Operazioni elementari sui vettori

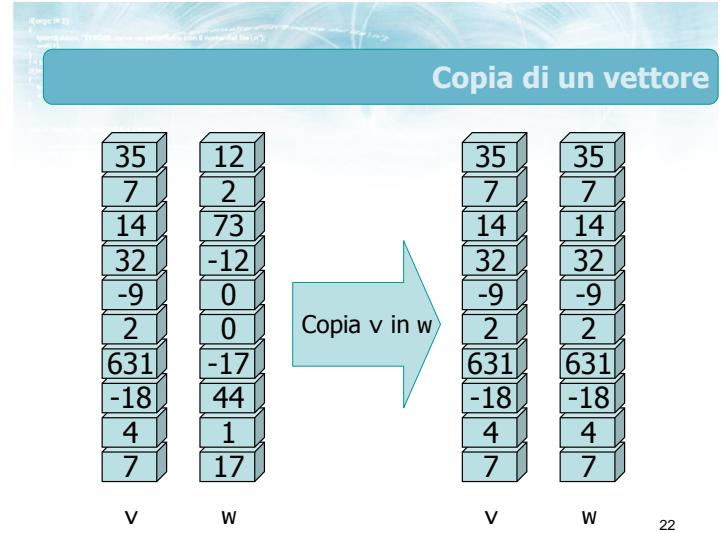
Copia di un vettore

19

Copia di un vettore

- » Più correttamente, si tratta di copiare **il contenuto** di un vettore in un altro vettore
- » Occorre copiare un elemento per volta dal vettore "sorgente" al vettore "destinazione", all'interno di un ciclo for
- » I due vettori devono avere lo stesso numero di elementi, ed essere dello stesso tipo base

21



22

Copia di un vettore

```
/* copia il contenuto di v[] in w[] */
for( i=0; i<N; i++ )
{
    w[i] = v[i] ;
}
```

23

Avvertenze

- » Nonostante siano coinvolti **due** vettori, occorre **un solo ciclo for**, e **un solo indice** per accedere agli elementi di entrambi i vettori
- » Assolutamente non tentare di fare la copia in una sola istruzione!

```
w = v ;
w[] = v[] ;
w[N] = v[N] ;
w[1,N] = v[1,N] ;
```

24

```
#include <stdio.h>
#include <ctype.h>
#define MAXPAGOLA 30
#define MAXRGA 80
```

```
int main(int argc, char *argv[])
{
    int freq[MAXPAGOLA]; /* vettore di dimensione
    delle frequenze delle lunghezze delle pagine */
    int i, min, max, l;
    float spazio_disco, pagine;
    if(argc != 2)
    {
        fprintf(stderr, "ERRORE: non sono state inserite le pagine del file\n");
        exit(1);
    }
    if((l = fopen(argv[1], "r")) == NULL)
    {
        perror("ERRORE: impossibile aprire il file %s", argv[1]);
        exit(1);
    }
    if(fread(freq, sizeof(int), MAXPAGOLA, l) != NULL)
```

Operazioni elementari sui vettori

Ricerca di un elemento

26

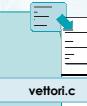
[Merge 14/3] VETTORI: ricerca un particolare dato il numero del file (n°3)

Ricerca di un elemento

- Dato un valore numerico, verificare
 - se almeno uno degli elementi del vettore è uguale al valore numerico
 - in caso affermativo, dire dove si trova
 - in caso negativo, dire che non esiste

- Si tratta di una classica istanza del problema di "ricerca di esistenza"

Ricerca di un elemento (1/3)



```
int dato ; /* dato da ricercare */
int trovato ; /* flag per ricerca */
int pos ; /* posizione elemento */

...
printf("Elemento da ricercare? ");
scanf("%d", &dato);
```

27

Ricerca di un elemento (2/3)

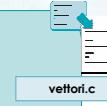


```
trovato = 0 ;
pos = -1 ;

for( i=0 ; i<N ; i++ )
{
    if( v[i] == dato )
    {
        trovato = 1 ;
        pos = i ;
    }
}
```

28

Ricerca di un elemento (3/3)



```
if( trovato==1 )
{
    printf("Elemento trovato "
           "alla posizione %d\n", pos+1) ;
}
else
{
    printf("Elemento non trovato\n");
```

29

Varianti

Altri tipi di ricerche

- Contare quante volte è presente l'elemento cercato
- Cercare se esiste almeno un elemento maggiore (o minore) del valore specificato
- Cercare se esiste un elemento approssimativamente uguale a quello specificato
- ...

30

- Dato un vettore (di interi o reali), determinare
 - quale sia l'elemento di valore massimo
 - quale sia la posizione in cui si trova tale elemento
- Conviene applicare la stessa tecnica per l'identificazione del massimo già vista in precedenza
 - Conviene inizializzare il max al valore del primo elemento

Operazioni elementari sui vettori

Ricerca del massimo o minimo

32

Ricerca del massimo (1/2)

```
float max ; /* valore del massimo */
int posmax ; /* posizione del max */

...
max = r[0] ;
posmax = 0 ;

for( i=1 ; i<N ; i++ )
{
    if( r[i]>max )
    {
        max = r[i] ;
        posmax = i ;
    }
}
```

33

Ricerca del massimo (2/2)

```
printf("Il max vale %f e si ", max) ;
printf("trova in posiz. %d\n", posmax) ;
```

34

Operazioni elementari sui vettori

Vettori ad occupazione variabile

36

Occupazione variabile

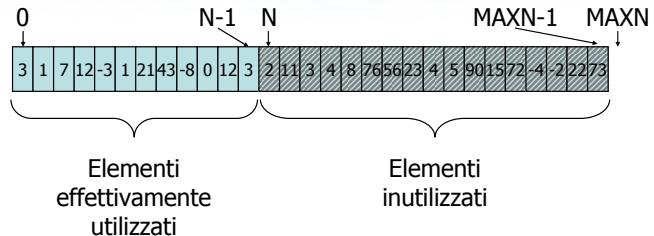
- La principale limitazione dei vettori è la loro dimensione fissa, definita come costante al tempo di compilazione del programma
- Molto spesso non si conosce l'effettivo numero di elementi necessari fino a quando il programma non andrà in esecuzione
- Occorre identificare delle tecniche che ci permettano di lavorare con vettori di dimensione fissa, ma occupazione variabile

Tecnica adottata

- Dichiarare un vettore di dimensione sufficientemente ampia da contenere il massimo numero di elementi nel caso peggiore
 - Esempio: MAXN
- La parte iniziale del vettore sarà occupata dagli elementi, la parte finale rimarrà inutilizzata
- Dichiarare una variabile che tenga traccia dell'effettiva occupazione del vettore
 - Esempio: N

37

Tecnica adottata



38

Esempio

```
/* dimensione massima */  
const int MAXN = 100 ;  
  
int v[MAXN] ; /* vettore di dim. max. */  
int N ; /* occupazione effettiva  
del vettore */  
  
...  
  
N = 0 ; /* inizialmente "vuoto" */
```

39

Regole di utilizzo

- All'inizio del programma si inizializza N al numero effettivo di elementi
 - Esempio: `scanf("%d", &N);`
- Verificare sempre che $N \leq MAXN$
- Durante l'esecuzione, utilizzare sempre N, e mai MAXN
 - Esempio: `for(i=0; i<N; i++)`
- Gli elementi da $v[N]$ a $v[MAXN-1]$ vengono ignorati (costituiscono memoria "sprecata")

40

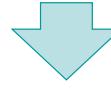
Crescita del vettore

- Un vettore ad occupazione variabile può facilmente crescere, aggiungendo elementi "in coda"

```
v[N] = nuovo_elemento ;  
N++ ;
```

41

Esempio

- Acquisire da tastiera una serie di numeri reali, e memorizzarli in un vettore.
 - La serie di numeri è terminata da un valore uguale a 0
- 
- Il valore di N non è noto all'inizio del programma, ma viene aggiornato via via che si leggono gli elementi

42

Soluzione (1/3)

```
const int MAXN = 100 ;  
  
float v[MAXN] ;  
float dato ;  
  
int N ;  
int i ;  
  
printf("Inserisci gli elementi\n") ;  
printf("(per terminare 0)\n") ;
```

leggi0.c

43

Soluzione (2/3)

```
N = 0 ; /* vettore inizialm. vuoto */  
  
/* leggi il primo dato */  
i = 0 ;  
printf("Elemento %d: ", i+1) ;  
scanf("%f", &dato) ;  
i++ ;  
  
/* aggiungi al vettore */  
if( dato != 0.0 )  
{  
    v[N] = dato ;  
    N++ ;  
}
```

leggi0.c

44

Soluzione (3/3)

```
while(dato!= 0.0)  
{  
    /* leggi il dato successivo */  
    printf("Elemento %d: ", i+1) ;  
    scanf("%f", &dato) ;  
    i++ ;  
  
    /* aggiungi al vettore */  
    if( dato != 0.0 )  
    {  
        v[N] = dato ;  
        N++ ;  
    }  
}
```

leggi0.c

45

Esercizio “Positivi e Negativi”

- ▶ Si realizzi un programma che legga da tastiera una sequenza di numeri interi (terminata da 0), e che successivamente stampi
 - tutti i numeri positivi presenti nella sequenza, nello stesso ordine
 - tutti i numeri negativi presenti nella sequenza, nello stesso ordine

46

Analisi

```
Prompt dei comandi  
INSERISCI UNA SEQUENZA (0 per terminare)  
Elemento: 3  
Elemento: -4  
Elemento: 1  
Elemento: 2  
Elemento: -3  
Elemento: 0  
  
Numeri positivi:  
3 1 2  
Numeri negativi:  
-4 -3
```

47

Approccio risolutivo

- ▶ Definiamo 3 vettori ad occupazione variabile:
 - seq, di occupazione N, che memorizza la sequenza iniziale
 - pos, di occupazione Np, che memorizza i soli elementi positivi
 - neg, di occupazione Nn, che memorizza i soli elementi negativi
- ▶ Il programma inizialmente acquisirà da tastiera il vettore seq, in seguito trascriverà ciascun elemento nel vettore più opportuno

48

Soluzione (1/4)

```
const int MAXN = 100 ;  
  
int seq[MAXN] ;  
int pos[MAXN] ;  
int neg[MAXN] ;  
  
int N, Np, Nn ;  
  
int i ;  
int dato ;  
  
/* vettori inizialmente vuoti */  
N = 0 ;  
Np = 0 ;  
Nn = 0 ;
```

posneg.c

49

Soluzione (2/4)

```
/* LETTURA SEQUENZA INIZIALE */  
/* vedi esempio precedente ... */
```

posneg.c

50

Soluzione (3/4)

```
for( i=0 ; i<N ; i++ )  
{  
    if(seq[i] > 0)  
    {  
        /* positivo => in pos[] */  
        pos[Np] = seq[i] ;  
        Np++ ;  
    }  
    else  
    {  
        /* negativo => in neg[] */  
        neg[Nn] = seq[i] ;  
        Nn++ ;  
    }  
}
```

posneg.c

51

Soluzione (4/4)

```
printf("Numeri positivi:\n") ;  
for(i=0; i<Np; i++)  
    printf("%d ", pos[i]) ;  
printf("\n");  
  
printf("Numeri negativi:\n") ;  
for(i=0; i<Nn; i++)  
    printf("%d ", neg[i]) ;  
printf("\n");
```

posneg.c

52

Vettori

Esercizi guidati sui vettori

2

```

#include <stdio.h>
#include <string.h>
#include <ctype.h>

#define MAXPAROLA 30
#define MAXBROA 80

int main(void argc, char *argv[])
{
    int lungMAXPAROLA; // > valore di costante
    char MAXBROA[ ]; // > stringa di dimensione n.
    char parola[MAXPAROLA];
    char broa[MAXBROA];
    int i, j, k, l, m, n, lungMAXparola;
    float x, y, z;

    for(;;MAXPAROLA++);
        broa[0] = '\0';

    if(argc < 1)
    {
        printf("Inserire il nome del file da leggere: ");
        scanf("%s", MAXBROA);
        if(strcmp(MAXBROA, "quit") == 0)
            exit(1);
    }

    if(lungMAXparola > 0) {
        for(i=0; i<lungMAXparola; i++) {
            if(argv[1][i] == 'a')
                parola[i] = 'A';
            else if(argv[1][i] == 'A')
                parola[i] = 'a';
            else if(argv[1][i] == 'E')
                parola[i] = 'e';
            else if(argv[1][i] == 'e')
                parola[i] = 'E';
            else if(argv[1][i] == 'I')
                parola[i] = 'i';
            else if(argv[1][i] == 'i')
                parola[i] = 'I';
            else if(argv[1][i] == 'O')
                parola[i] = 'o';
            else if(argv[1][i] == 'o')
                parola[i] = 'O';
            else if(argv[1][i] == 'U')
                parola[i] = 'u';
            else if(argv[1][i] == 'u')
                parola[i] = 'U';
            else if(argv[1][i] == 'Y')
                parola[i] = 'y';
            else if(argv[1][i] == 'y')
                parola[i] = 'Y';
            else
                parola[i] = argv[1][i];
        }
    }
}

```

Esercizi guidati sui vettori

Esercizio “Elementi comuni”

4

Esercizio “Elementi comuni” (2/2)

- ▶ In particolare, in una prima fase il programma acquisisce le disponibilità dei due colleghi
 - Per ciascun collega il programma acquisisce un elenco di numeri interi (supponiamo compresi tra e 31), che indicano i giorni del mese in cui essi sono disponibili. L'immissione dei dati termina inserendo 0.
 - ▶ Nella seconda fase, il programma identificherà i giorni in cui entrambi i colleghi sono disponibili, e li stamperà a video



Esercizi guidati sui vettori

- ▶ Esercizio “Elementi comuni”
 - ▶ Esercizio “Ricerca duplicati”
 - ▶ Esercizio “Sottosequenza”
 - ▶ Esercizio “Poligono”

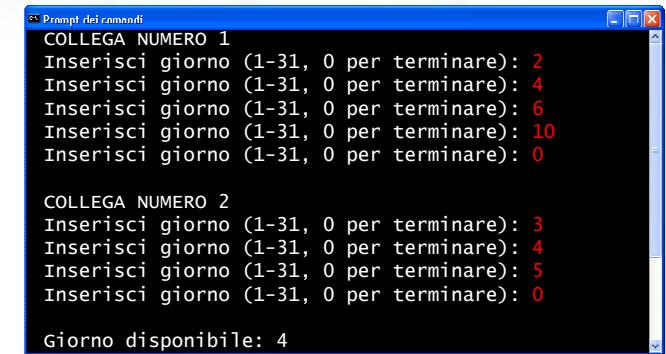


Esercizio “Elementi comuni” (1/2)

- ▶ Due colleghi intendono fissare una riunione, pertanto devono identificare dei giorni nei quali sono entrambi liberi da impegni. A tale scopo, essi realizzano un programma C che permetta a ciascuno di immettere le proprie disponibilità, e che identifichi i giorni nei quali entrambi sono liberi



Analisi (1/3)



6

Analisi (2/3)

```
collega numero 1  
Inserisci giorno (1-31, 0 per terminare): 2  
Inserisci giorno (1-31, 0 per terminare): 4  
Inserisci giorno (1-31, 0 per terminare): 6  
Inserisci giorno (1-31, 0 per terminare): 10  
Inserisci giorno (1-31, 0 per terminare): 0  
  
collega numero 2  
Inserisci giorno (1-31, 0 per terminare): 3  
Inserisci giorno (1-31, 0 per terminare): 5  
Inserisci giorno (1-31, 0 per terminare): 7  
Inserisci giorno (1-31, 0 per terminare): 0  
  
Purtroppo non vi e' NESSUN giorno disponibile
```

7

Analisi (3/3)

```
collega numero 1  
Inserisci giorno (1-31, 0 per terminare): 2  
Inserisci giorno (1-31, 0 per terminare): 4  
Inserisci giorno (1-31, 0 per terminare): 6  
Inserisci giorno (1-31, 0 per terminare): 0  
  
collega numero 2  
Inserisci giorno (1-31, 0 per terminare): 2  
Inserisci giorno (1-31, 0 per terminare): 3  
Inserisci giorno (1-31, 0 per terminare): 4  
Inserisci giorno (1-31, 0 per terminare): 0  
  
Giorno disponibile: 2  
Giorno disponibile: 4
```

8

Algoritmo

- » Acquisisci le disponibilità del collega 1
 - Vettore `giorni1[]` di `N1` elementi
- » Acquisisci le disponibilità del collega 2
 - Vettore `giorni2[]` di `N2` elementi
- » Verifica se vi sono elementi di `giorni1[]` che siano anche elementi di `giorni2[]`
 - Se sì, stampa tali elementi
 - Se no, stampa un messaggio

9

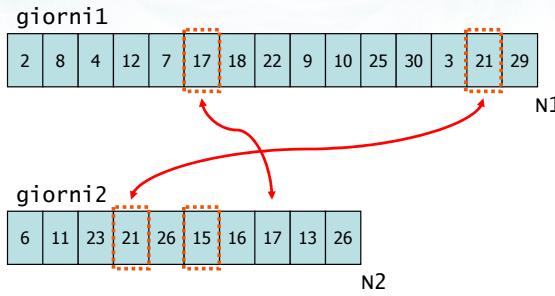
Ricerca elementi comuni

giorni1	N1
2 8 4 12 7 17 18 22 9 10 25 30 3 21 29	

giorni2	N2
6 11 23 21 26 15 16 17 13 26	

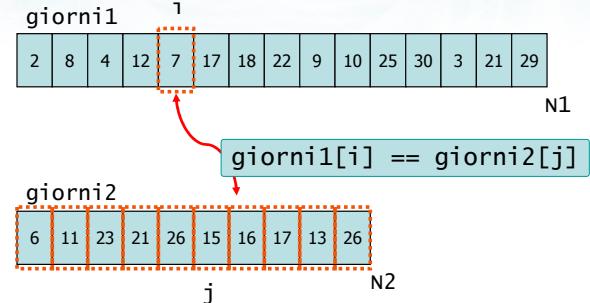
10

Ricerca elementi comuni



11

Ricerca elementi comuni



12

Soluzione (1/5)

```
const int MAXN = 100 ;  
  
int N1, N2 ;  
int giorni1[MAXN] ; /* giorni collega 1 */  
int giorni2[MAXN] ; /* giorni collega 2 */  
  
int giorno ;  
int i, j ;  
int trovato ;  
/* flag: giorni1[i] in giorni2[]? */  
int fallito ;  
/* flag: trovato almeno un giorno? */
```

riunione.c

13

Soluzione (2/5)

```
/* DISPONIBILITA' COLLEGA 1 */  
printf("COLLEGA NUMERO 1\n");  
N1 = 0 ;  
printf("Inserisci giorno (1-31): ");  
scanf("%d", &giorno) ;  
  
while( giorno != 0 )  
{  
    giorni1[N1] = giorno ;  
    N1++ ;  
  
    printf("Inserisci giorno (1-31): ");  
    scanf("%d", &giorno) ;  
}  
printf("Collega 1 ha inserito %d giorni\n",  
      N1);
```

riunione.c

14

Soluzione (3/5)

```
/* DISPONIBILITA' COLLEGA 2 */  
printf("COLLEGA NUMERO 2\n");  
N2 = 0 ;  
printf("Inserisci giorno (1-31): ");  
scanf("%d", &giorno) ;  
  
while( giorno != 0 )  
{  
    giorni2[N2] = giorno ;  
    N2++ ;  
  
    printf("Inserisci giorno (1-31): ");  
    scanf("%d", &giorno) ;  
}  
printf("Collega 2 ha inserito %d giorni\n",  
      N2);
```

riunione.c

15

Soluzione (4/5)

```
/* RICERCA DEGLI ELEMENTI COMUNI */  
fallito = 1 ;  
/* Per ogni giorno del collega 1... */  
for( i=0 ; i<N1; i++ )  
{  
    /* ...verifica se è disponibile  
     * il collega 2... */  
  
    /* ...in caso affermativo stampalo */  
    if( trovato == 1 )  
    {  
        printf("Giorno disponibile: %d\n",  
               giorni1[i]) ;  
        fallito = 0 ;  
    }  
}
```

riunione.c

16

Soluzione (4/5)

```
/* RICERCA DEGLI ELEMENTI COMUNI */  
fallito = 1 ;  
/* Per ogni giorno del collega 1... */  
for( i=0 ; i<N1; i++ )  
{  
    /* ...verifica se è disponibile  
     * il collega 2... */  
  
    /* ...in caso affermativo stampalo */  
    if( trovato == 1 )  
    {  
        printf("Giorno disponibile: %d\n",  
               giorni1[i]) ;  
        fallito = 0 ;  
    }  
}
```

riunione.c

17

Soluzione (5/5)

```
/* Se non ne ho trovato nessuno,  
   stampo un messaggio */  
if(fallito==1)  
    printf("NESSUN giorno disponibile\n");
```

riunione.c

18

```

#include <stdio.h>
#include <string.h>
#define MAXPAROLA 30
#define MAXIGA 80

int main(int argc, char *argv[])
{
    int freq[MAXPAROLA]; /* vettore di dimensione
                           delle frequenze delle lunghezze delle parole */
    int i, max, lunghezza;
    FILE *fp;

    fp=fopen(argv[1], "r");
    if(fp==NULL)
    {
        fprintf(stderr, "ERRORE: non è possibile aprire il file %s", argv[1]);
        exit(1);
    }

    for(i=0; i<MAXIGA; i++)
        freq[i]=0;

    while(fgets(argv[2], MAXIGA, fp))
    {
        lunghezza = strlen(argv[2]);
        freq[lunghezza]++;
    }

    max=freq[0];
    for(i=1; i<MAXIGA; i++)
        if(freq[i]>max)
            max=freq[i];

    printf("Parola più frequente: %s (%d)\n", argv[2], max);
}

```

Esercizi guidati sui vettori

Esercizio "Ricerca duplicati"

20

[Merge 14.3]

Esercizio "Ricerca duplicati" (1/2)

- La società organizzatrice di un concerto vuole verificare che non ci siano biglietti falsi. A tale scopo, realizza un programma in linguaggio C che acquisisce i numeri di serie dei biglietti e verifica che non vi siano numeri duplicati

```

#include <stdio.h>
#include <string.h>
#define MAXN 5
#define MAXIGA 80

int main()
{
    int serie[MAXN];
    int i, totale, n, num, pos;
    FILE *fp;

    fp=fopen("biglietti.txt", "r");
    if(fp==NULL)
    {
        fprintf(stderr, "ERRORE: non è possibile aprire il file %s", "biglietti.txt");
        exit(1);
    }

    totale=0;
    for(i=0; i<MAXN; i++)
        serie[i]=-1;

    while(fgets(fp, MAXIGA, fp))
    {
        num=atoi(fp);
        if(num<0 || num>MAXIGA)
            continue;

        if(serie[num]==-1)
            serie[num]=num;
        else
            pos=num;
    }

    totale=0;
    for(i=0; i<MAXN; i++)
        if(serie[i]!=-1)
            totale++;

    if(totale==n)
        printf("Tutto regolare\n");
    else
        printf("ATTENZIONE: biglietto %d duplicato!\n", pos);
}

```

Esercizio "Ricerca duplicati" (2/2)

- In particolare, il programma acquisisce innanzitutto il numero di biglietti venduti, N, ed in seguito acquisisce i numeri di serie degli N biglietti
- Al termine dell'acquisizione, il programma stamperà "Tutto regolare" se non si sono riscontrati duplicati, altrimenti stamperà il numero di serie dei biglietti duplicati

21

```

[Merge 14.3]
[Analisi]
[Analisi (1/2)]

```

[x] Prompt dei comandi

RICERCA BIGLIETTI DUPLICATI

Numero totale di biglietti: 5
 Numero di serie del biglietto 1: 1234
 Numero di serie del biglietto 2: 4321
 Numero di serie del biglietto 3: 1423
 Numero di serie del biglietto 4: 1242
 Numero di serie del biglietto 5: 3321
 Tutto regolare

22

```

[Analisi (2/2)]

```

23

```

[Analisi (2/2)]
[Algoritmo]

```

- Acquisizione del valore di N
- Lettura dei numeri di serie
 - Utilizziamo un vettore: `int serie[MAXN]`
- Ricerca dei duplicati**
- Stampa dei messaggi finali

24

Ricerca dei duplicati

- » Prendi un elemento per volta
 - `elem = serie[i] ;`
- » Cerca se **altri** elementi del vettore sono **uguali** a tale elemento
 - uguali \Rightarrow (`elem == serie[j]`)
 - altri \Rightarrow (`i != j`)
- » Si tratta di una ricerca di esistenza per ogni elemento considerato

25

Soluzione (1/5)

```
const int MAXN = 100 ;
int N ; /* num tot biglietti */
int serie[MAXN] ; /* numeri serie */

int elem ;
int i, j ;

/* flag: trovato almeno un dupl. ? */
int dupl ;

/* flag per ricerca di esistenza */
int trovato ;
```

26

Soluzione (2/5)

```
printf("RICERCA DUPLICATI\n") ;
printf("\n");

/* ACQUISIZIONE VALORE DI N */
do{
    printf("Num tot di biglietti: ") ;
    scanf("%d", &N) ;
    if (N<2 || N>MAXN)
        printf("N=%d non valido\n", N) ;
} while(N<2 || N>MAXN) ;
```

biglietti.c

27

Soluzione (3/5)

```
/* LETTURA DEI NUMERI DI SERIE */
for( i=0 ; i<N ; i++ )
{
    printf("Numero serie biglietto %d: ", i+1) ;
    scanf("%d", &serie[i]) ;
}
```

28

Soluzione (4/5)

```
/* RICERCA DEI DUPLICATI */
dupl = 0 ;
for( i=0 ; i<N ; i++ )
{
    /* verifica se serie[i] e' duplicato */
    elem = serie[i] ;

    /* => ricerca esistenza di elem
       all'interno di serie[] */

    if(trovato == 1)
    {
        printf("ATTENZIONE: %d duplicato\n",
               elem) ;
        dupl = 1 ;
    }
}
```

biglietti.c

29

Soluzione (4/5)

```
/* RICERCA DEI DUPLICATI */
dupl = 0 ;
for( i=0 ; i<N ; i++ )
{
    /* verifica se serie[i] e' duplicato */
    elem = serie[i] ;

    /* => ricerca esistenza di elem
       all'interno di serie[] */

    if(trovato == 1)
    {
        printf("A el")
        dupl = 1 ;
    }
}

trovato = 0 ;
for( j=0 ; j<N ; j++ )
{
    if( (i!=j) &&
        (elem == serie[j]) )
        trovato = 1 ;
}
```

30

Soluzione (5/5)

```
/* STAMPA DEI MESSAGGI FINALI */
if(dupl==0)
    printf("Tutto regolare\n");
```

bigletti.c

31

#include <stdio.h>
#include <string.h>
#include <ctype.h>

#define MAXPAROLA 30
#define MAXSIGA 60

```
int main(int argc, char *argv[])
{
    int freq[MAXPAROLA]; /* vettore di contatori
                           delle frequenze delle lunghezze delle parole */
    int i, maxfreq = 0;
    int l, minf, larghezza = 0;
    int k = 0;

    for(i=0; i<MAXSIGA; i++)
        freq[i] = 0;

    if(argc != 2)
        fprintf(stderr, "ERRORE: non è possibile indicare il nome del file (*.txt)\n");
    else {
        FILE *fp;
        fp = fopen(argv[1], "r");
        if(fp == NULL) {
            perror(argv[1]);
            exit(1);
        }
    }

    while(fgets(argv[k], MAXSIGA, fp) != NULL) {
        l = strlen(argv[k]);
        if(l > larghezza)
            larghezza = l;
        for(i=0; i<l; i++) {
            if(isalpha(argv[k][i])) {
                freq[l-1]++;
                break;
            }
        }
    }

    for(i=0; i<MAXSIGA; i++) {
        if(freq[i] > maxfreq)
            maxfreq = freq[i];
    }

    if(maxfreq >= larghezza)
        printf("Sottosequenza\n");
    else
        printf("Sottosequenza impossibile\n");
}
```

printf("lunghezza della sottosequenza: %d\n", larghezza);



Esercizi guidati sui vettori

Esercizio "Sottosequenza"

Esercizio "Sottosequenza" (1/2)

- In un esercizio di telepatia, un sensitivo scommette di essere in grado di indovinare almeno 3 numeri consecutivi, in una sequenza di 100 numeri pensati da uno spettatore
- Per garantire l'oggettività dell'esperimento, viene realizzato un programma in C per la verifica dell'avvenuta telepatia
 - Per maggior generalità, il programma viene realizzato in modo da controllare sequenze di almeno K numeri consecutivi, all'interno di sequenze di N numeri.

33

Esercizio "Sottosequenza" (2/2)

- In particolare, il programma acquisisce innanzitutto la sequenza di N numeri pensati dallo spettatore. Si ipotizza che tali numeri siano interi positivi, compresi tra 1 e 10000
- In seguito, il programma acquisisce dal sensitivo una sequenza di K numeri
- Il programma verifica se esiste, nella sequenza di N numeri, una sottosequenza di K numeri esattamente uguale a quella inserita dal sensitivo
- I valori di N e K sono introdotti dall'utente all'inizio del programma

34

Analisi

```
Prompt dei comandi
Lunghezza della sequenza complessiva: 6
Lunghezza della sequenza da indovinare: 3
Inserire la sequenza complessiva
Elemento 1: 3
Elemento 2: 4
Elemento 3: 5
Elemento 4: 6
Elemento 5: 7
Elemento 6: 8
Inserire la sequenza da indovinare telepaticamente
Elemento 1: 5
Elemento 2: 6
Elemento 3: 7
Complimenti! Hai ottime capacita' telepatiche
```

35

Algoritmo (1/2)

- Chiamiamo

- seq[] la sequenza di N elementi
- tele[] la sottosequenza di K elementi ($K < N$)

36

Algoritmo (2/2)

➤ Verifichiamo se

- i primi K elementi di seq[] sono uguali ai K elementi di tele[]
- i K elementi di seq[] con indice da 1 a K sono uguali ai K elementi di tele[]
- i K elementi di seq[] con indice da 2 a K+1 sono uguali ai K elementi di tele[]
- i K elementi di seq[] con indice da 3 a K+2 sono uguali ai K elementi di tele[]
- ...

37

Sottosequenze

seq

2	8	4	12	7	21	18	22	9	10	25	30	3	17	29
														N

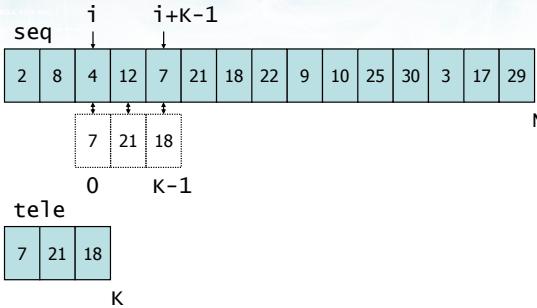
tele

7	21	18
K		

K

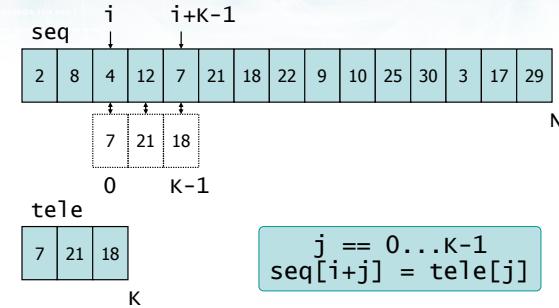
N

Sottosequenze



39

Sottosequenze



N

40

Soluzione (1/4)

```
const int MAXN = 100 ;
const int MAXK = 10 ;

int N ; /* lunghezza seq. completa */
int K ; /* lunghezza sottosequenza */
int seq[MAXN] ; /* seq. completa */
int tele[MAXK] ; /* seq. telepatica */

int i, j ;
int trovato ; /* flag: ricorda se ha
    trovato una sottosequenza uguale */
int errore ; /* flag: verifica che
    TUTTI gli elementi della
    sottosequenza siano uguali */
```

41

sensitivo.c

Soluzione (2/4)

```
printf("ESPERIMENTO DI TELEPATIA\n") ;
printf("\n");

/* ACQUISIZIONE LUNGHEZZA SEQUENZE */
...leggi da tastiera i valori di N e K...

/* ACQUISIZIONE SEQUENZA COMPLESSIVA */
...leggi da tastiera il vettore seq[] di N elementi...

/* ACQUISIZIONE SEQ. DA INDOVINARE */
...leggi da tastiera il vettore tele[] di K elementi...
```

42

Soluzione (3/4)

```

trovato = 0 ;
/* considera tutti i possibili
punti di partenza (i) */
for( i=0; i<N-K; i++ )
{
    /* verifica se seq[] nelle
    posizioni da (i) a (i+K-1) e'
    uguale a tele[] nelle posizioni
    da (0) a (K-1) */

    /* la sottosequenza era corretta? */
    if(errore==0)
        trovato=1 ;
}

```

sensitivo.c

43

Soluzione (3/4)

```

trovato = 0 ;
/* considera tutti i possibili
punti di partenza (i) */
for( i=0; i<N-K; i++ )
{
    /* verifica se seq[] nelle
    posizioni da (i) a (i+K-1) e'
    uguale a tele[] nelle posizioni
    da (0) a (K-1) */

    /* la sottosequenza era corretta? */
    if(errore==0)
        trovato=1 ;
}

```

sensitivo.c

44

Soluzione (4/4)

```

/* STAMPA RISULTATO DELLA VERIFICA */
if( trovato==1 )
    printf("Complimenti!\n");
else
    printf("Esperimento non riuscito\n");

```

sensitivo.c

45

#include <stdio.h>
#include <string.h>
#include <ctype.h>

#define MAXPAROLA 30
#define MAXRIGA 80

```

int main(int argc, char *argv[])
{
    int lungMAXPAROLA; /* numero di caratteri massimi per una parola */
    int lungMAXRIGA; /* numero massimo di righe */
    int i, j, k, lungRiga;
    FILE *f;

    f=fopen(argv[1], "r");
    if(f==NULL)
    {
        printf("ERRORE: impossibile aprire il file %s", argv[1]);
        exit(1);
    }

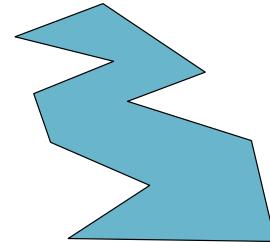
    while(fgets(sigla, MAXRIGA, f)!=NULL)
    {
        if(strlen(sigla)>lungMAXRIGA)
            lungRiga=lungMAXRIGA;
        else
            lungRiga=strlen(sigla);
        for(k=0; k<lungMAXRIGA; k++)
        {
            if(sigla[k]=='.')
                break;
            if(isalpha(sigla[k]))
                lungMAXPAROLA++;
        }
    }
}
```

Esercizi guidati sui vettori

Esercizio “Poligono”

Esercizio “Poligono” (1/2)

- Uno studente di geometria deve misurare il perimetro di una serie di poligoni irregolari, di cui conosce le coordinate cartesiane (x,y) dei vertici. Per far ciò realizza un programma in C



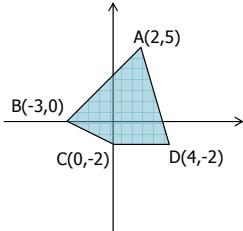
47

- In particolare, il programma innanzitutto acquisisce il numero di vertici di cui è composto il poligono. Chiamiamo N tale numero
- In seguito, il programma acquisisce le N coppie (x,y) corrispondenti agli N vertici
- Il programma infine stampa la lunghezza complessiva del perimetro del poligono irregolare:
 - comando delle lunghezze di ciascun lato
 - lato determinato dalle coordinate dei vertici

Esercizio “Poligono” (2/2)

48

Analisi (1/2)



- » $AB = \sqrt{(2-(-3))^2 + (5-0)^2}$
- » $BC = \sqrt{(-3-0)^2 + (0-(-2))^2}$
- » $CD = \sqrt{(0-4)^2 + (-2-(-2))^2}$
- » $DA = \sqrt{(4-2)^2 + (-2-5)^2}$

- » Perimetro =
• $= AB + BC + CD + DA$

49

Analisi (2/2)

```
es> Prompt dei comandi
CALCOLO DEL PERIMETRO

Numero di vertici: 4
Inserire le coordinate dei vertici
vertice 1: x = 2
y = 5
vertice 2: x = -3
y = 0
vertice 3: x = 0
y = -2
vertice 4: x = 4
y = -2

Lunghezza del perimetro: 21.956729
```

50

Struttura dati

- » In questo problema i dati da memorizzare non sono semplici numeri, ma coppie di numeri
- » Possiamo utilizzare due vettori
 - Vettore $x[]$, contenente le ascisse dei punti
 - Vettore $y[]$, contenente le ordinate dei punti
- » Esempio:
 - Punto A(2,5) $\Rightarrow x[0]=2$; $y[0]=5$;
- » In tutte le elaborazioni, i due vettori verranno usati con **uguale valore dell'indice**
 - Vettori "paralleli"

51

Soluzione (1/3)

```
const int MAXN = 10 ;
int N ; /* numero di vertici */
/* vettori "paralleli" */
float x[MAXN] ;
float y[MAXN] ;
int i ;
float lato ;
float perimetro ;
```

52

Soluzione (2/3)

```
/* ACQUISIZIONE NUMERO VERTICI */
...leggi da tastiera il valore di N...

/* ACQUISIZIONE COORDINATE VERTICI */
printf("Inserire coordinate\n");
for( i=0; i<N; i++ )
{
    printf("vertice %d: x = ", i+1) ;
    scanf("%f", &x[i]) ;
    printf("                y = ") ;
    scanf("%f", &y[i]) ;
}
```

53

```
perimetro = 0 ;
for( i=0; i<N-1; i++ )
{
    /* (x[i],y[i])-(x[i+1],y[i+1]) */
    lato = sqrt( (x[i]-x[i+1])*(x[i]-x[i+1])
                 + (y[i]-y[i+1])*(y[i]-y[i+1]) ) ;
    perimetro = perimetro + lato ;
}
/* ultimo lato:
   (x[N-1],y[N-1])-(x[0],y[0]) */
lato = sqrt( (x[N-1]-x[0])*(x[N-1]-x[0])
                 + (y[N-1]-y[0])*(y[N-1]-y[0]) ) ;
perimetro = perimetro + lato ;
```

54

```
#include <stdio.h>
#include <ctype.h>
#define MAXPARI 30
#define MAXIMA 80

int main(int argc, char *argv[])
{
    int freq[MAXPARI]; /* vettore di frequenza delle frequenze delle parole */
    int i, max, longhezza;
    FILE *fp;

    if(argc != 2)
    {
        fprintf(stderr, "ERRORE: serve un parametro che è il nome del file (*.txt)\n");
        exit(1);
    }

    fp = fopen(argv[1], "r");
    if(fp == NULL)
    {
        fprintf(stderr, "ERRORE: impossibile aprire il file %s", argv[1]);
        exit(1);
    }

    for(i=0; i<MAXPARI; i++)
        freq[i] = 0;

    longhezza = 0;
    max = 0;
    while(fgets(fp, 100, fp) != NULL)
    {
        for(i=0; i<strlen(fp); i++)
        {
            if(isalpha(fp[i]))
            {
                fp[i] = toupper(fp[i]);
                freq[fp[i] - 'A']++;
            }
        }
        if(strlen(fp) > max)
            max = strlen(fp);
        longhezza += strlen(fp);
    }

    printf("Parole con la più lunga parola: %d\n", max);
    printf("Lunghezza media delle parole: %.2f\n", (float)longhezza / freq[0]);
    printf("Parole con la più alta frequenza: %d\n", freq[0]);
    printf("Frequenza delle parole: \n");
    for(i=0; i<MAXPARI; i++)
        if(freq[i] != 0)
            printf("%c: %d\n", i + 'A', freq[i]);
}

int main()
{
    if(freq[0] == 0)
        printf("Nessuna parola nel file\n");
}
```

Vettori

Sommario

2



- » La struttura dati vettoriale
- » Dichiarazione di vettori in C
- » Accesso agli elementi del vettore
- » Vettori con occupazione variabile

Tecniche di programmazione

- » Lettura e scrittura di vettori
- » Ricerca di elementi
- » Ricerche di duplicati, di sottosequenze, di elementi comuni, ...
- » Vettori paralleli

3

Vettori e cicli

- » Per elaborare il contenuto di un vettore sono spesso necessari dei cicli
 - Operazioni semplici: scansione del vettore per stampa, lettura, ricerca, ...
 - Operazioni complesse: possono richiedere più cicli annidati

4



- » Al vettore **non è associato alcun indice** particolare, per scandirne gli elementi
- » Lo stesso vettore può essere usato con indici diversi
- » Lo stesso indice può essere applicato a vettori diversi

5



- » Non è detto che ad **ogni vettore** corrisponda un **ciclo**
- » Controesempio: nella ricerca di elementi duplicati, vi è un solo vettore, ma due cicli annidati
- » Controesempio: nel calcolo del perimetro, vi sono due vettori, ma un solo ciclo

6



Suggerimenti

- Tenere separate le operazioni di lettura/scrittura dalle elaborazioni vere e proprie
- Procedere per gradi, in modalità top-down, cercando di riconoscere ove possibile le strutture note
 - ricerca di un elemento
 - verifica di esistenza
 - verifica di universalità
- Non confondere i vari “flag” utilizzati in caso di cicli annidati

7

Avvertenze

- La combinazione di vettori e cicli crea un **fortissimo incremento nella complessità** dei programmi realizzabili
 - Questo è il punto più ripido nella curva di apprendimento
- Prima di procedere oltre, allenarsi con molti esercizi di programmazione

8

Materiale aggiuntivo

- Sul CD-ROM
 - Testi e soluzioni degli esercizi trattati nei lucidi
 - Scheda sintetica
 - Esercizi risolti
 - Esercizi proposti
- Esercizi proposti da altri libri di testo

9

```
#include <stdio.h>
#include <ctype.h>
#define MAXPAGINA 80
```

```
int main(int argc, char *argv[])
{
```

```
    int freq[MAXPAGINA]; /* vettore di contenuto delle frequenze delle parole */
    FILE *fp; /* file da leggere */
    int i, inicio, lunghezza;
```

```
    inicio = inicioMAXPAGINA - 1;
    lunghezza = MAXPAGINA;
```

```
    fp = fopen(argv[1], "r");
    if (fp == NULL)
```

```
        fprintf(stderr, "ERRORE: non è possibile aprire il file %s", argv[1]);
```

```
    else
```

```
        for (i = inicio; i >= 0; i--) freq[i] = 0;
```

```
    while (fscanf(fp, "%c", &char) != EOF)
```

```
        if (char >= 'A' & char <= 'Z')
```

```
            freq[char - 'A']++;
```

```
    for (i = inicio; i >= 0; i--) printf("%c\t%d\n", i + 'A', freq[i]);
```

```
    fclose(fp);
}
```

Programmazione in C

Unità Caratteri e stringhe

Caratteri e stringhe

- » Dati testuali
- » Il tipo char
- » Vettori di caratteri
- » Operazioni elementari sulle stringhe
- » Funzioni di libreria
- » Esercizi proposti
- » Sommario

2

Riferimenti al materiale

» Testi

- Kernighan & Ritchie: capitoli 1 e 5
- Cabodi, Quer, Sonza Reorda: capitolo 5
- Dietel & Dietel: capitolo 8

» Dispense

- Scheda: "Caratteri e stringhe in C"

3

- Tipi di dato testuali
- Caratteri
- Stringhe

Caratteri e stringhe

Dati testuali

5

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

#define MAXPAROLA 30
#define MAXRIGA 80

int main(int argc, char *argv[])
{
    int lungMAXPAROLA; /* valore di costante
    cioè numero massimo di caratteri nella parola */
    char rigaMAXRIGA; /* valore di costante
    cioè numero massimo di caratteri della riga */
    int i, indice, lungRiga;
    FILE *fptr;

    if(argc != 2)
    {
        fprintf(stderr, "ERRORE: non è possibile aprire il file %s", argv[1]);
        exit(1);
    }

    if((fptr = fopen(argv[1], "r")) == NULL)
    {
        perror("ERRORE: impossibile aprire il file %s", argv[1]);
        exit(1);
    }

    while(fgets(riga, MAXRIGA, fptr) != NULL)
    {
        if(strlen(riga) > lungMAXRIGA)
        {
            printf("ERRORE: la riga %s supera il limite massimo (%d)\n", riga, lungMAXRIGA);
            exit(1);
        }
        else
        {
            for(i=0; riga[i] != '\0'; i++)
            {
                if(isalpha(riga[i]))
                {
                    printf("%c", riga[i]);
                }
                else
                {
                    printf("\\%c", riga[i]);
                }
            }
            printf("\n");
        }
    }
    fclose(fptr);
}
```

Dati testuali

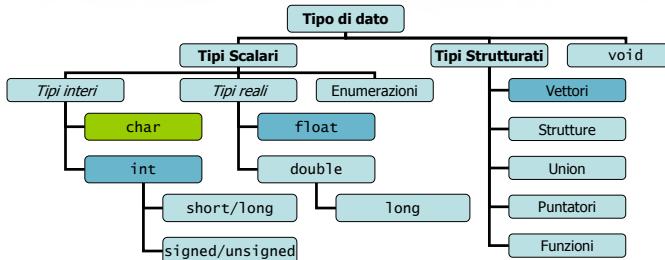
Tipi di dato testuali

7

Tipi di dato testuali

- I programmi visti finora erano in grado di elaborare esclusivamente informazioni numeriche
 - Numeri interi (`int`), numeri reali (`float`)
 - Variabili singole o vettori
- In molti casi è necessario elaborare informazioni di tipo testuale
 - Vuoi continuare (s/n)?
 - Conta le parole di un testo scritto
 - Gestisci una rubrica di nomi e numeri di telefono
 - ...

Il sistema dei tipi C



8

Rappresentazione dei testi

- Il calcolatore è in grado di rappresentare i caratteri alfabetici, numerici ed i simboli speciali di punteggiatura
- Ad ogni diverso carattere viene assegnato, **convenzionalmente**, un codice numerico corrispondente
- Il programma in C lavora sempre con i codici numerici
- Le funzioni di input/output sono in grado di accettare e mostrare i caratteri corrispondenti

9

Codice ASCII

Dec	Hex	Oct	Utf8	Chr	Dec	Hex	Oct	Utf8	Chr	Dec	Hex	Oct	Utf8	Chr			
0	0 000	NUL (null)			32	20 040	#32;	Space		64	40 100	#64;		96	60 140	#96;	
1	1 001	SOH (start of heading)			33	21 041	#33;			65	41 101	#65;	A	97	61 141	#97;	a
2	2 002	STX (start of text)			34	22 042	#34;			66	42 102	#66;	B	98	62 142	#98;	b
3	3 003	ETX (end of text)			35	23 043	#35;	#		67	43 103	#67;	C	99	63 143	#99;	c
4	4 004	ETB (end of transmission)			36	24 044	#36;			68	44 104	#68;	D	100	64 144	#100;	d
5	5 005	ENQ (enquiry)			37	25 045	#37;			69	45 105	#69;	E	101	65 145	#101;	e
6	6 006	ACK (acknowledge)			38	26 046	#38;			70	46 106	#70;	F	102	66 146	#102;	f
7	7 007	BEL (bell)			39	27 047	#39;			71	47 107	#71;	G	103	67 147	#103;	g
8	8 010	BS (backspace)			40	28 050	#40;			72	48 110	#72;	H	104	68 148	#104;	h
9	9 011	TAB (horizontal tab)			41	29 051	#41;			73	49 111	#73;	I	105	69 151	#105;	i
10	10 012	LF (newline, feed, new line)			42	30 052	#42;			74	50 112	#74;	J	106	70 152	#106;	j
11	11 013	VT (vertical tab)			43	28 053	#43;+			75	48 113	#75;	K	107	68 153	#107;	k
12	12 014	FF (NP form feed, new page)			44	2C 054	#44;			4C	114	#76;	L	108	69 154	#108;	l
13	13 015	CR (carriage return)			45	2D 055	#45;			76	49 115	#77;	M	109	60 155	#109;	m
14	14 016	SO (shift out)			46	2E 056	#46;			77	4D 115	#77;	M	109	60 155	#109;	m
15	15 017	SI (shift in)			47	2F 057	#47;			78	4C 116	#78;	N	110	6F 156	#110;	n
16	16 020	DLE (data link escape)			48	30 060	#48;	0		80	50 120	#80;	P	112	70 160	#112;	p
17	17 021	DC1 (device control 1)			49	31 061	#49;	1		81	51 121	#81;	Q	113	71 161	#113;	q
18	18 022	DC2 (device control 2)			50	32 062	#50;	2		82	52 122	#82;	R	114	72 162	#114;	r
19	19 023	DC3 (device control 3)			51	33 063	#51;	3		83	53 123	#83;	S	115	73 163	#115;	s
20	20 024	DC4 (device control 4)			52	34 064	#52;	4		84	54 124	#84;	T	116	74 164	#116;	t
21	21 025	NAK (negative acknowledge)			53	35 065	#53;	5		85	55 125	#85;	U	117	75 165	#117;	u
22	22 026	SYN (synchronous idle)			54	36 066	#54;	6		86	56 126	#86;	V	118	76 166	#118;	v
23	23 027	ETB (end of trans. block)			55	37 067	#55;	7		87	57 127	#87;	W	119	77 167	#119;	w
24	24 030	CAN (cancel)			56	38 068	#56;	8		88	58 128	#88;	X	120	78 168	#120;	x
25	25 031	EN (end of medium)			57	39 071	#57;	9		89	59 129	#89;	Y	121	79 169	#121;	y
26	26 032	SUB (substitute)			58	3A 072	#58;			90	5A 132	#90;	Z	122	7A 172	#122;	z
27	27 033	ESC (escape)			59	3B 073	#59;			91	5B 133	#91;		123	7B 173	#123;	
28	28 034	FS (file separator)			60	3C 074	#60;	<		92	5C 134	#92;		124	7C 174	#124;	
29	29 035	GS (group separator)			61	3D 075	#61;	=		93	5D 135	#93;		125	7D 175	#125;	
30	30 036	RS (record separator)			62	3E 076	#62;	-		94	5E 136	#94;		126	7E 176	#126;	
31	31 037	US (unit separator)			63	3F 077	#63;	~		95	5F 137	#95;		127	7F 177	#127;	DEL

10

Source: www.lookupables.com

- Il codice ASCII permette di rappresentare un singolo carattere

y 7 w ! %

- Nelle applicazioni pratiche spesso serve rappresentare sequenze di caratteri: stringhe

F	u	1	v	i	o	0	6	A	Z	N
0	1	1	-	5	6	4	6	3	3	2

11

Dualità caratteri - numeri

- Ogni carattere è rappresentato dal suo codice ASCII

y 7 w ! %

121

55

87

33

37

- Ogni stringa è rappresentata dai codici ASCII dei caratteri di cui è composta

F	u	1	v	i	o	0	6	A	Z	N
70	117	108	118	105	111	48	54	65	90	78
0	1	1	-	5	6	4	6	3	3	2
48	49	49	45	53	54	52	54	51	51	50

12

Source: www.lookupables.com

Caratteri in C

- Ogni carattere viene rappresentato dal proprio codice ASCII
- Sono sufficienti 7 bit per rappresentare ciascun carattere
 - Il C usa variabili di 8 bit (1 byte)
- Non sono previste le lettere accentate né altri simboli diacritici
 - Richiedono estensioni speciali e librerie specifiche

14

Dec	Hex	Oct	Utf8	Chr	Dec	Hex	Oct	Utf8	Chr	Dec	Hex	Oct	Utf8	Chr			
0	0 000	NUL (null)			32	20 040	#32;	Space		64	40 100	#64;		96	60 140	#96;	
1	1 001	SOH (start of heading)			33	21 041	#41;			65	41 101	#65;	A	97	61 141	#97;	a
2	2 002	STX (start of text)			34	22 042	#34;			66	42 102	#66;	B	98	62 142	#98;	b
3	3 003	ETX (end of text)			35	23 043	#35;#			67	43 103	#67;	C	99	63 143	#99;	c
4	4 004	ETB (end of transmission)			36	24 044	#36;			68	44 104	#68;	D	100	64 144	#100;	d
5	5 005	ENQ (enquiry)			37	25 045	#35;			69	45 105	#69;	E	101	65 145	#101;	e
6	6 006	ACK (acknowledge)			38	26 046	#38;			70	46 106	#70;	F	102	66 146	#102;	f
7	7 007	BEL (bell)			39	27 047	#39;			71	47 107	#71;	G	103	67 147	#103;	g
8	8 010	BS (backspace)			40	28 050	#40;			72	48 110	#72;	H	104	68 148	#104;	h
9	9 011	TAB (horizontal tab)			41	29 051	#41;			73	49 111	#73;	I	105	69 151	#105;	i
10	10 012	LF (newline, feed, new line)			42	30 052	#42;			74	50 112	#74;	J	106	60 152	#106;	j
11	11 013	VT (vertical tab)			43	28 053	#43;+			75	48 113	#75;	K	107	68 153	#107;	k
12	12 014	FF (NP form feed, new page)			44	2C 054	#44;			76	49 114	#76;	L	108	69 154	#108;	l
13	13 015	CR (carriage return)			45	2D 055	#45;			77	50 115	#77;	M	109	60 155	#109;	m
14	14 016	SO (shift out)			46	2E 056	#46;			78	48 116	#78;	N	110	6F 156	#110;	n
15	15 017	SI (shift in)			47	2F 057	#47; /			79	47 117	#79;	O	111	6F 157	#111;	o
16	16 020	DLE (data link escape)			48	30 060	#48;	0		80	50 120	#80;	P	112	70 160	#112;	p
17	17 021	DC1 (device control 1)			49	31 061	#49;	1		81	51 121	#81;	Q	113	71 161	#113;	q
18	18 022	DC2 (device control 2)			50	32 062	#50;	2		82	52 122	#82;	R	114	72 162	#114;	r
19	19 023	DC3 (device control 3)			51	33 063	#51;	3		83	53 123	#83;	S	115	73 163	#115;	s
20	20 024	DC4 (device control 4)			52	34 064	#52;	4		84	54 124	#84;	T	116	74 164	#116;	t
21	21 025	NAK (negative acknowledge)			53	35 065	#53;	5		85	55 125	#85;	U	117	75 165	#117;	u
22	22 026	SYN (synchronous idle)			54	36 066	#54;	6		86	56 126	#86;	V	118	76 166	#118;	v
23	23 027	ETB (end of trans. block)			55	37 067	#57;	7		87	57 127	#87;	W	119	77 167	#119;	w
24	24 030	CAN (cancel)			56	38 070	#56;	8		88	58 130	#88;	X	120	78 168	#120;	x
25	25 031	EN (end of medium)			57	39 071	#57;	9		89	59 131	#89;	Y	121	79 171	#121;	y
26	26 032	SUB (substitute)			58	3A 072	#58;			90	5A 132	#90;	Z	122	7A 172	#122;	z
27	27 033	ESC (escape)			59	3B 073	#59;			91	5B 133	#91;		123	7B 173	#123;	
28	28 034	FS (file separator)			60	3C 074	#60;	<		92	5C 134	#92;		124	7C 174	#124;	
29	29 035	GS (group separator)			61	3D 075	#61;	=		93	5D 135	#93;		125	7D 175	#125;	
30	30 036	RS (record separator)			62	3E 076	#62;	-		94	5E 136	#94;		126	7E 176	#126;	
31	31 037	US (unit separator)			63	3F 077	#63;	~		95	5F 137	#95;		127	7F 177	#127;	DEL

Source: www.lookupables.com

Caratteri e stringhe

- Il codice ASCII permette di rappresentare un singolo carattere

Codice ASCII

Dec	Hx	Oct	Char	Dec	Hx	Oct	Char
0	\000	000	NUL (null)	41	6#		
1	\001	001	SOH (start of header)	42	6#		
2	\002	002	STX (start of text)	43	6#		
3	\003	003	ETX (end of text)	44	6#		
4	\004	004	(end of transmission)	36	24	044	6#

Valore decimale
(tra 0 e 127)

Simbolo corrispondente

16

Source: www.lookuptables.com

Codice ASCII

Dec	Hx	Oct	Html	Chr
96	60	140	`	
97	61	141	a	à
98	62	142	b	à
99	63	143	c	à
100	64	144	d	à
101	65	145	e	à
102	66	146	f	à
103	67	147	g	à
104	68	150	h	à
105	69	151	i	à
106	70	152	j	à
107	71	153	k	à
108	72	154	l	à
109	73	155	m	à
110	74	156	n	à
111	75	157	o	à
112	76	160	p	à
113	77	161	q	à
114	78	162	r	à

Lettere minuscole

Lettere maiuscole

17

Source: www.lookuptables.com

Codice ASCII

Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
Space	64	40	100	@	à	96	60	140	`	à
0	65	41	101	A	à	97	61	141	a	à
1	66	42	102	B	à	98	62	142	b	à
2	67	43	103	C	à	99	63	143	c	à
3	68	44	104	D	à	100	64	144	d	à
4	69	45	105	E	à	101	65	145	e	à
5	70	46	106	B	à	102	66	146	f	à
6	71	47	107	C	à	103	67	147	g	à
7	72	48	108	D	à	104	68	148	h	à
8	73	49	109	E	à	105	69	149	i	à
9	74	50	110	<	à	106	70	150	j	à
;	75	51	111	=	à	107	71	151	k	à
,	76	52	112	>	à	108	72	152	l	à
:	77	53	113	?	à	109	73	153	m	à
;	78	54	114	@	à	110	74	154	n	à
;	79	55	115	A	à	111	75	155	o	à
;	80	56	116	B	à	112	76	156	p	à
;	81	57	117	C	à	113	77	157	q	à
;	82	58	118	D	à	114	78	158	r	à
;	83	59	119	E	à	115	79	159	s	à
;	84	60	120	<	à	116	80	160	t	à
;	85	61	121	=	à	117	81	161	u	à
;	86	62	122	>	à	118	82	162	v	à
;	87	63	123	?	à	119	83	163	w	à
;	88	64	124	@	à	120	84	164	n	à
;	89	65	125	A	à	121	85	165	o	à
;	90	66	126	B	à	122	86	166	p	à
;	91	67	127	C	à	123	87	167	q	à
;	92	68	128	D	à	124	88	168	r	à
;	93	69	129	E	à	125	89	169	s	à
;	94	70	130	<	à	126	90	170	t	à
;	95	71	131	=	à	127	91	171	u	à
;	96	72	132	>	à	128	92	172	v	à
;	97	73	133	?	à	129	93	173	w	à
;	98	74	134	\	à	130	94	174	|	à
;	99	75	135]	à	131	95	175	}	à
;	100	76	136	^	à	132	96	176	~	à
;	101	77	137	_	à	133	97	177		à
;	102	78	138	`	à	134	98	178	€	à
;	103	79	139	a	à	135	99	179		à
;	104	80	140	b	à	136	100	180	x	à
;	105	81	141	c	à	137	101	181	y	à
;	106	82	142	Z	à	138	102	182	z	à
;	107	83	143	[à	139	103	183	{	à
;	108	84	144	\	à	140	104	184	|	à
;	109	85	145]	à	141	105	185	}	à
;	110	86	146	^	à	142	106	186	~	à
;	111	87	147	_	à	143	107	187		à
;	112	88	148	`	à	144	108	188	€	à
;	113	89	149	a	à	145	109	189		à
;	114	90	150	b	à	146	110	190	x	à
;	115	91	151	c	à	147	111	191	y	à
;	116	92	152	Z	à	148	112	192	z	à
;	117	93	153	[à	149	113	193	{	à
;	118	94	154	\	à	150	114	194	|	à
;	119	95	155]	à	151	115	195	}	à
;	120	96	156	^	à	152	116	196	~	à
;	121	97	157	_	à	153	117	197		à
;	122	98	158	`	à	154	118	198	€	à
;	123	99	159	a	à	155	119	199		à
;	124	100	160	b	à	156	120	200	x	à
;	125	101	161	c	à	157	121	201	y	à
;	126	102	162	Z	à	158	122	202	z	à
;	127	103	163	[à	159	123	203	{	à
;	128	104	164	\	à	160	124	204	|	à
;	129	105	165]	à	161	125	205	}	à
;	130	106	166	^	à	162	126	206	~	à
;	131	107	167	_	à	163	127	207		à
;	132	108	168	`	à	164	128	208	€	à
;	133	109	169	a	à	165	129	209		à
;	134	110	170	b	à	166	130	210	x	à
;	135	111	171	c	à	167	131	211	y	à
;	136	112	172	Z	à	168	132	212	z	à
;	137	113	173	[à	169	133	213	{	à
;	138	114	174	\	à	170	134	214	|	à
;	139	115	175]	à	171	135	215	}	à
;	140	116	176	^	à	172	136	216	~	à
;	141	117	177	_	à	173	137	217		à
;	142	118	178	`	à	174	138	218	€	à
;	143	119	179	a	à	175	139	219		à
;	144	120	180	b	à	176	140	220	x	à
;	145	121	181	c	à	177	141	221	y	à
;	146	122	182	Z	à	178	142	222	z	à
;	147	123	183	[à	179	143	223	{	à
;	148	124	184	\	à	180	144	224	|	à
;	149	125	185]	à	181	145	225	}	à
;	150	126	186	^	à	182	146	226	~	à
;	151	127	187	_	à	183	147	227		à
;	152	128	188	`	à	184	148	228	€	à
;	153	129	189	a	à	185	149	229		à
;	154	130	190	b	à	186	150	230	x	à
;	155	131	191	c	à	187	151	231	y	à
;	156	132	192	Z	à	188	152	232	z	à
;	157	133	193	[à	189	153	233	{	à
;	158	134	194	\	à	190	154	234	|	à
;	159	135	195]	à	191	155	235	}	à
;	160	136	196	^	à	192	156	236	&	



Errore frequente

- Non confondere il carattere ASCII che rappresenta una cifra numerica con il valore decimale associato a tale cifra

int
7

char
7
55

- Per chiarezza useremo gli apici per indicare i caratteri

char
'7'
55

22



Errore frequente

- Pensare che un singolo carattere possa memorizzare più simboli

char
Fulvio

char F
55
char u
117
char l
108

23

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

#define MAXPAROLA 30
#define MAXSIGLA 80

int main(int argc, char *argv[])
{
    int lunghezzaParola; /* /> valore di parola */
    char testoParola; /* /> stringa inserita dalla tastiera */
    char siglaMAXSIGLA; /* /> sigla, lunghezza: */
    int i, j;
    if(argc > 1)
    {
        lunghezzaParola = strlen(argv[1]);
        testoParola = argv[1];
        if(lunghezzaParola > MAXPAROLA)
            printf("ERRORE, imposta la parola da %d caratteri\n", MAXPAROLA);
        else
        {
            if(isupper(testoParola))
                printf("ERRORE, imposta la parola da maiuscolo\n");
            else
                printf("ERRORE, imposta la parola da minuscolo\n");
        }
    }
    else
        printf("ERRORE, non inserisci nulla\n");
    return 0;
}
```

Dati testuali

Stringhe

25

Caratteristiche delle stringhe

- Memorizzate come singoli caratteri, ma il loro significato è dato dall'intera sequenza di caratteri
- Lunghezza variabile
- Mix di lettere/cifre/punteggiatura/spazi
- Solitamente non contengono caratteri di controllo

F	u	1	v	i	o
70	117	108	118	105	111

0	6	A	Z	N
48	54	65	90	78

0	1	1	-	5	6	4	6	3	3	2
48	49	49	45	53	54	52	54	51	51	50

26

Manipolazione delle stringhe

- Occorre trattare l'insieme di caratteri memorizzato nel vettore come un'unica "variabile"
- Ogni operazione elementare sulle stringhe coinvolgerà tipicamente dei cicli che scandiscono il vettore
- Molte funzioni di libreria sono già disponibili per compiere le operazioni più frequenti ed utili

27



Errore frequente

- Non confondere una stringa composta da cifre numeriche con il valore decimale associato a tale sequenza

int	char
137	1 3 7
49 51 55	

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

#define MAXPAROLA 30
#define MAXSIGA 80

int main(int argc, char *argv[])
{
    int freqMAXPAROLA; /* valore di controllo delle frequenze delle lunghezze delle parole */
    char freq[MAXSIGA]; /* frequenze delle lunghezze */
    int i, indice, lunghezza;
    FILE *fptr;
    char riga[100];
    int nlinee;
    int spazio;
    int c;
    int posizione;
    int maxlunghezza;
    int maxfreq;
    int i1, i2, i3;
    int i4, i5, i6;
    int i7, i8, i9;
    int i10, i11, i12;
    int i13, i14, i15;
    int i16, i17, i18;
    int i19, i20, i21;
    int i22, i23, i24;
    int i25, i26, i27;
    int i28, i29, i30;
    int i31, i32, i33;
    int i34, i35, i36;
    int i37, i38, i39;
    int i39, i40, i41;
    int i42, i43, i44;
    int i45, i46, i47;
    int i47, i48, i49;
    int i49, i50, i51;
    int i51, i52, i53;
    int i53, i54, i55;
    int i55, i56, i57;
    int i57, i58, i59;
    int i59, i60, i61;
    int i61, i62, i63;
    int i63, i64, i65;
    int i65, i66, i67;
    int i67, i68, i69;
    int i69, i70, i71;
    int i71, i72, i73;
    int i73, i74, i75;
    int i75, i76, i77;
    int i77, i78, i79;
    int i79, i80, i81;
    int i81, i82, i83;
    int i83, i84, i85;
    int i85, i86, i87;
    int i87, i88, i89;
    int i89, i90, i91;
    int i91, i92, i93;
    int i93, i94, i95;
    int i95, i96, i97;
    int i97, i98, i99;
    int i99, i100, i101;
    int i101, i102, i103;
    int i103, i104, i105;
    int i105, i106, i107;
    int i107, i108, i109;
    int i109, i110, i111;
    int i111, i112, i113;
    int i113, i114, i115;
    int i115, i116, i117;
    int i117, i118, i119;
    int i119, i120, i121;
    int i121, i122, i123;
    int i123, i124, i125;
    int i125, i126, i127;
    int i127, i128, i129;
    int i129, i130, i131;
    int i131, i132, i133;
    int i133, i134, i135;
    int i135, i136, i137;
    int i137, i138, i139;
    int i139, i140, i141;
    int i141, i142, i143;
    int i143, i144, i145;
    int i145, i146, i147;
    int i147, i148, i149;
    int i149, i150, i151;
    int i151, i152, i153;
    int i153, i154, i155;
    int i155, i156, i157;
    int i157, i158, i159;
    int i159, i160, i161;
    int i161, i162, i163;
    int i163, i164, i165;
    int i165, i166, i167;
    int i167, i168, i169;
    int i169, i170, i171;
    int i171, i172, i173;
    int i173, i174, i175;
    int i175, i176, i177;
    int i177, i178, i179;
    int i179, i180, i181;
    int i181, i182, i183;
    int i183, i184, i185;
    int i185, i186, i187;
    int i187, i188, i189;
    int i189, i190, i191;
    int i191, i192, i193;
    int i193, i194, i195;
    int i195, i196, i197;
    int i197, i198, i199;
    int i199, i200, i201;
    int i201, i202, i203;
    int i203, i204, i205;
    int i205, i206, i207;
    int i207, i208, i209;
    int i209, i210, i211;
    int i211, i212, i213;
    int i213, i214, i215;
    int i215, i216, i217;
    int i217, i218, i219;
    int i219, i220, i221;
    int i221, i222, i223;
    int i223, i224, i225;
    int i225, i226, i227;
    int i227, i228, i229;
    int i229, i230, i231;
    int i231, i232, i233;
    int i233, i234, i235;
    int i235, i236, i237;
    int i237, i238, i239;
    int i239, i240, i241;
    int i241, i242, i243;
    int i243, i244, i245;
    int i245, i246, i247;
    int i247, i248, i249;
    int i249, i250, i251;
    int i251, i252, i253;
    int i253, i254, i255;
    int i255, i256, i257;
    int i257, i258, i259;
    int i259, i260, i261;
    int i261, i262, i263;
    int i263, i264, i265;
    int i265, i266, i267;
    int i267, i268, i269;
    int i269, i270, i271;
    int i271, i272, i273;
    int i273, i274, i275;
    int i275, i276, i277;
    int i277, i278, i279;
    int i279, i280, i281;
    int i281, i282, i283;
    int i283, i284, i285;
    int i285, i286, i287;
    int i287, i288, i289;
    int i289, i290, i291;
    int i291, i292, i293;
    int i293, i294, i295;
    int i295, i296, i297;
    int i297, i298, i299;
    int i299, i300, i301;
    int i301, i302, i303;
    int i303, i304, i305;
    int i305, i306, i307;
    int i307, i308, i309;
    int i309, i310, i311;
    int i311, i312, i313;
    int i313, i314, i315;
    int i315, i316, i317;
    int i317, i318, i319;
    int i319, i320, i321;
    int i321, i322, i323;
    int i323, i324, i325;
    int i325, i326, i327;
    int i327, i328, i329;
    int i329, i330, i331;
    int i331, i332, i333;
    int i333, i334, i335;
    int i335, i336, i337;
    int i337, i338, i339;
    int i339, i340, i341;
    int i341, i342, i343;
    int i343, i344, i345;
    int i345, i346, i347;
    int i347, i348, i349;
    int i349, i350, i351;
    int i351, i352, i353;
    int i353, i354, i355;
    int i355, i356, i357;
    int i357, i358, i359;
    int i359, i360, i361;
    int i361, i362, i363;
    int i363, i364, i365;
    int i365, i366, i367;
    int i367, i368, i369;
    int i369, i370, i371;
    int i371, i372, i373;
    int i373, i374, i375;
    int i375, i376, i377;
    int i377, i378, i379;
    int i379, i380, i381;
    int i381, i382, i383;
    int i383, i384, i385;
    int i385, i386, i387;
    int i387, i388, i389;
    int i389, i390, i391;
    int i391, i392, i393;
    int i393, i394, i395;
    int i395, i396, i397;
    int i397, i398, i399;
    int i399, i400, i401;
    int i401, i402, i403;
    int i403, i404, i405;
    int i405, i406, i407;
    int i407, i408, i409;
    int i409, i410, i411;
    int i411, i412, i413;
    int i413, i414, i415;
    int i415, i416, i417;
    int i417, i418, i419;
    int i419, i420, i421;
    int i421, i422, i423;
    int i423, i424, i425;
    int i425, i426, i427;
    int i427, i428, i429;
    int i429, i430, i431;
    int i431, i432, i433;
    int i433, i434, i435;
    int i435, i436, i437;
    int i437, i438, i439;
    int i439, i440, i441;
    int i441, i442, i443;
    int i443, i444, i445;
    int i445, i446, i447;
    int i447, i448, i449;
    int i449, i450, i451;
    int i451, i452, i453;
    int i453, i454, i455;
    int i455, i456, i457;
    int i457, i458, i459;
    int i459, i460, i461;
    int i461, i462, i463;
    int i463, i464, i465;
    int i465, i466, i467;
    int i467, i468, i469;
    int i469, i470, i471;
    int i471, i472, i473;
    int i473, i474, i475;
    int i475, i476, i477;
    int i477, i478, i479;
    int i479, i480, i481;
    int i481, i482, i483;
    int i483, i484, i485;
    int i485, i486, i487;
    int i487, i488, i489;
    int i489, i490, i491;
    int i491, i492, i493;
    int i493, i494, i495;
    int i495, i496, i497;
    int i497, i498, i499;
    int i499, i500, i501;
    int i501, i502, i503;
    int i503, i504, i505;
    int i505, i506, i507;
    int i507, i508, i509;
    int i509, i510, i511;
    int i511, i512, i513;
    int i513, i514, i515;
    int i515, i516, i517;
    int i517, i518, i519;
    int i519, i520, i521;
    int i521, i522, i523;
    int i523, i524, i525;
    int i525, i526, i527;
    int i527, i528, i529;
    int i529, i530, i531;
    int i531, i532, i533;
    int i533, i534, i535;
    int i535, i536, i537;
    int i537, i538, i539;
    int i539, i540, i541;
    int i541, i542, i543;
    int i543, i544, i545;
    int i545, i546, i547;
    int i547, i548, i549;
    int i549, i550, i551;
    int i551, i552, i553;
    int i553, i554, i555;
    int i555, i556, i557;
    int i557, i558, i559;
    int i559, i560, i561;
    int i561, i562, i563;
    int i563, i564, i565;
    int i565, i566, i567;
    int i567, i568, i569;
    int i569, i570, i571;
    int i571, i572, i573;
    int i573, i574, i575;
    int i575, i576, i577;
    int i577, i578, i579;
    int i579, i580, i581;
    int i581, i582, i583;
    int i583, i584, i585;
    int i585, i586, i587;
    int i587, i588, i589;
    int i589, i590, i591;
    int i591, i592, i593;
    int i593, i594, i595;
    int i595, i596, i597;
    int i597, i598, i599;
    int i599, i600, i601;
    int i601, i602, i603;
    int i603, i604, i605;
    int i605, i606, i607;
    int i607, i608, i609;
    int i609, i610, i611;
    int i611, i612, i613;
    int i613, i614, i615;
    int i615, i616, i617;
    int i617, i618, i619;
    int i619, i620, i621;
    int i621, i622, i623;
    int i623, i624, i625;
    int i625, i626, i627;
    int i627, i628, i629;
    int i629, i630, i631;
    int i631, i632, i633;
    int i633, i634, i635;
    int i635, i636, i637;
    int i637, i638, i639;
    int i639, i640, i641;
    int i641, i642, i643;
    int i643, i644, i645;
    int i645, i646, i647;
    int i647, i648, i649;
    int i649, i650, i651;
    int i651, i652, i653;
    int i653, i654, i655;
    int i655, i656, i657;
    int i657, i658, i659;
    int i659, i660, i661;
    int i661, i662, i663;
    int i663, i664, i665;
    int i665, i666, i667;
    int i667, i668, i669;
    int i669, i670, i671;
    int i671, i672, i673;
    int i673, i674, i675;
    int i675, i676, i677;
    int i677, i678, i679;
    int i679, i680, i681;
    int i681, i682, i683;
    int i683, i684, i685;
    int i685, i686, i687;
    int i687, i688, i689;
    int i689, i690, i691;
    int i691, i692, i693;
    int i693, i694, i695;
    int i695, i696, i697;
    int i697, i698, i699;
    int i699, i700, i701;
    int i701, i702, i703;
    int i703, i704, i705;
    int i705, i706, i707;
    int i707, i708, i709;
    int i709, i710, i711;
    int i711, i712, i713;
    int i713, i714, i715;
    int i715, i716, i717;
    int i717, i718, i719;
    int i719, i720, i721;
    int i721, i722, i723;
    int i723, i724, i725;
    int i725, i726, i727;
    int i727, i728, i729;
    int i729, i730, i731;
    int i731, i732, i733;
    int i733, i734, i735;
    int i735, i736, i737;
    int i737, i738, i739;
    int i739, i740, i741;
    int i741, i742, i743;
    int i743, i744, i745;
    int i745, i746, i747;
    int i747, i748, i749;
    int i749, i750, i751;
    int i751, i752, i753;
    int i753, i754, i755;
    int i755, i756, i757;
    int i757, i758, i759;
    int i759, i760, i761;
    int i761, i762, i763;
    int i763, i764, i765;
    int i765, i766, i767;
    int i767, i768, i769;
    int i769, i770, i771;
    int i771, i772, i773;
    int i773, i774, i775;
    int i775, i776, i777;
    int i777, i778, i779;
    int i779, i780, i781;
    int i781, i782, i783;
    int i783, i784, i785;
    int i785, i786, i787;
    int i787, i788, i789;
    int i789, i790, i791;
    int i791, i792, i793;
    int i793, i794, i795;
    int i795, i796, i797;
    int i797, i798, i799;
    int i799, i800, i801;
    int i801, i802, i803;
    int i803, i804, i805;
    int i805, i806, i807;
    int i807, i808, i809;
    int i809, i810, i811;
    int i811, i812, i813;
    int i813, i814, i815;
    int i815, i816, i817;
    int i817, i818, i819;
    int i819, i820, i821;
    int i821, i822, i823;
    int i823, i824, i825;
    int i825, i826, i827;
    int i827, i828, i829;
    int i829, i830, i831;
    int i831, i832, i833;
    int i833, i834, i835;
    int i835, i836, i837;
    int i837, i838, i839;
    int i839, i840, i841;
    int i841, i842, i843;
    int i843, i844, i845;
    int i845, i846, i847;
    int i847, i848, i849;
    int i849, i850, i851;
    int i851, i852, i853;
    int i853, i854, i855;
    int i855, i856, i857;
    int i857, i858, i859;
    int i859, i860, i861;
    int i861, i862, i863;
    int i863, i864, i865;
    int i865, i866, i867;
    int i867, i868, i869;
    int i869, i870, i871;
    int i871, i872, i873;
    int i873, i874, i875;
    int i875, i876, i877;
    int i877, i878, i879;
    int i879, i880, i881;
    int i881, i882, i883;
    int i883, i884, i885;
    int i885, i886, i887;
    int i887, i888, i889;
    int i889, i890, i891;
    int i891, i892, i893;
    int i893, i894, i895;
    int i895, i896, i897;
    int i897, i898, i899;
    int i899, i900, i901;
    int i901, i902, i903;
    int i903, i904, i905;
    int i905, i906, i907;
    int i907, i908, i909;
    int i909, i910, i911;
    int i911, i912, i913;
    int i913, i914, i915;
    int i915, i916, i917;
    int i917, i918, i919;
    int i919, i920, i921;
    int i921, i922, i923;
    int i923, i924, i925;
    int i925, i926, i927;
    int i927, i928, i929;
    int i929, i930, i931;
    int i931, i932, i933;
    int i933, i934, i935;
    int i935, i936, i937;
    int i937, i938, i939;
    int i939, i940, i941;
    int i941, i942, i943;
    int i943, i944, i945;
    int i945, i946, i947;
    int i947, i948, i949;
    int i949, i950, i951;
    int i951, i952, i953;
    int i953, i954, i955;
    int i955, i956, i957;
    int i957, i958, i959;
    int i959, i960, i961;
    int i961, i962, i963;
    int i963, i964, i965;
    int i965, i966, i967;
    int i967, i968, i969;
    int i969, i970, i971;
    int i971, i972, i973;
    int i973, i974, i975;
    int i975, i976, i977;
    int i977, i978, i979;
    int i979, i980, i981;
    int i981, i982, i983;
    int i983, i984, i985;
    int i985, i986, i987;
    int i987, i988, i989;
    int i989, i990, i991;
    int i991, i992, i993;
    int i993, i994, i995;
    int i995, i996, i997;
    int i997, i998, i999;
    int i999, i1000, i1001;
    int i1001, i1002, i1003;
    int i1003, i1004, i1005;
    int i1005, i1006, i1007;
    int i1007, i1008, i1009;
    int i1009, i1010, i1011;
    int i1011, i1012, i1013;
    int i1013, i1014, i1015;
    int i1015, i1016, i1017;
    int i1017, i1018, i1019;
    int i1019, i1020, i1021;
    int i1021, i1022, i1023;
    int i1023, i1024, i1025;
    int i1025, i1026, i1027;
    int i1027, i1028, i1029;
    int i1029, i1030, i1031;
    int i1031, i1032, i1033;
    int i1033, i1034, i1035;
    int i1035, i1036, i1037;
    int i1037, i1038, i1039;
    int i1039, i1040, i1041;
    int i1041, i1042, i1043;
    int i1043, i1044, i1045;
    int i1045, i1046, i1047;
    int i1047, i1048, i1049;
    int i1049, i1050, i1051;
    int i1051, i1052, i1053;
    int i1053, i1054, i1055;
    int i1055, i1056, i1057;
    int i1057, i1058, i1059;
    int i1059, i1060, i1061;
    int i1061, i1062, i1063;
    int i1063, i1064, i1065;
    int i1065, i1066, i1067;
    int i1067, i1068, i1069;
    int i1069, i1070, i1071;
    int i1071, i1072, i1073;
    int i1073, i1074, i1075;
    int i1075, i1076, i1077;
    int i1077, i1078, i1079;
    int i1079, i1080, i1081;
    int i1081, i1082, i1083;
    int i1083, i1084, i1085;
    int i1085, i1086, i1087;
    int i1087, i1088, i1089;
    int i1089, i1090, i1091;
    int i1091, i1092, i1093;
    int i1093, i1094, i1095;
    int i1095, i1096, i1097;
    int i1097, i1098, i1099;
    int i1099, i1100, i1101;
    int i1101, i1102, i1103;
    int i1103, i1104, i1105;
    int i1105, i1106, i1107;
    int i1107, i1108, i1109;
    int i1109, i1110, i1111;
    int i1111, i1112, i1113;
    int i1113, i1114, i1115;
    int i1115, i1116, i1117;
    int i1117, i1118, i1119;
    int i1119, i1120, i1121;
    int i1121, i1122, i1123;
    int i1123, i1124, i1125;
    int i1125, i1126, i1127;
    int i1127, i1128, i1129;
    int i1129, i1130, i1131;
    int i1131, i1132, i1133;
    int i1133, i1134, i1135;
    int i1135, i1136, i1137;
    int i1137, i1138, i1139;
    int i1139, i1140, i1141;
    int i1141, i1142, i1143;
    int i1143, i1144, i1145;
    int i1145, i1146, i1147;
    int i1147, i1148, i1149;
    int i1149, i1150, i1151;
    int i1151, i1152, i1153;
    int i1153, i1154, i1155;
    int i1155, i1156, i1157;
    int i1157, i1158, i1159;
    int i1159, i1160, i1161;
    int i1161, i1162, i1163;
    int i1163, i1164, i1165;
    int i1165, i1166, i1167;
    int i1167, i1168, i1169;
    int i1169, i1170, i1171;
    int i1171, i1172, i1173;
    int i1173, i1174, i1175;
    int i1175, i1176, i1177;
    int i1177, i1178, i1179;
    int i1179, i1180, i1181;
    int i1181, i1182, i1183;
    int i1183, i1184, i1185;
    int i1185, i1186, i1187;
    int i1187, i1188, i1189;
    int i1189, i1190, i1191;
    int i1191, i1192, i1193;
    int i1193, i1194, i1195;
    int i1195, i1196, i1197;
    int i1197, i1198, i1199;
    int i1199, i1200, i1201;
    int i1201, i1202, i1203;
    int i1203, i1204, i1205;
    int i1205, i1206, i1207;
    int i1207, i1208, i1209;
    int i1209, i1210, i1211;
    int i1211, i1212, i1213;
    int i1213, i1214, i1215;
    int i1215, i1216, i1217;
    int i1217, i1218, i1219;
    int i1219, i1220, i1221;
    int i1221, i1222, i1223;
    int i1223, i1224, i1225;
    int i1225, i1226, i1227;
    int i1227, i1228, i1229;
    int i1229, i1230, i1231;
    int i1231, i1232, i1233;
    int i1233, i1234, i1235;
    int i1235, i1236, i1237;
    int i1237, i1238, i1239;
    int i1239, i1240, i1241;
    int i1241, i1242, i1243;
    int i1243, i1244, i1245;
    int i1245, i1246, i1247;
    int i1247, i1248, i1249;
    int i1249, i1250, i1251;
    int i1251, i1252, i1253;
    int i1253, i1254, i1255;
    int i1255, i1256, i1257;
    int i1257, i1258, i1259;
    int i1259, i1260, i1261;
    int i1261, i1262, i1263;
    int i1263, i1264, i1265;
    int i1265, i1266, i1267;
    int i1267, i1268, i1269;
    int i1269, i1270, i1271;
    int i1271, i1272, i1273;
    int i1273, i1
```

Esempi

```
int i ;  
char c ;  
  
c = 'A' ;
```

7

Esempi

```
int i ;  
char c ;  
  
c = 'A' ;  
c = 65 ; /* equivalente! */
```

8

Esempi

```
int i ;  
char c ;  
  
c = 'A' ;  
c = 65 ; /* equivalente! */  
i = c ; /* i sarà 65 */
```

9

Esempi

```
int i ;  
char c ;  
  
c = 'A' ;  
c = 65 ; /* equivalente! */  
i = c ; /* i sarà 65 */  
c = c + 1 ; /* c sarà 66 = 'B' */
```

10

Esempi

```
int i ;  
char c ;  
  
c = 'A' ;  
c = 65 ; /* equivalente! */  
i = c ; /* i sarà 65 */  
c = c + 1 ; /* c sarà 66 = 'B' */  
c = c * 2 ; /* non ha senso... */
```

11

Esempi

```
int i ;  
char c ;  
  
c = 'A' ;  
c = 65 ; /* equivalente! */  
i = c ; /* i sarà 65 */  
c = c + 1 ; /* c sarà 66 = 'B' */  
c = c * 2 ; /* non ha senso... */  
if (c == 'Z') ...
```

12

```

int i ;
char c ;

c = 'A' ;
c = 65 ; /* equivalente! */
i = c ; /* i sarà 65 */
c = c + 1 ; /* c sarà 66 = 'B' */
c = c * 2 ; /* non ha senso... */
if (c == 'z') ...
for( c='A'; c<='Z'; c++) ...

```

13

- Per alcuni caratteri di controllo il linguaggio C definisce una particolare **sequenza di escape** per poterli rappresentare

C	ASCII	Significato
'\n'	LF - 10	A capo
'\t'	TAB - 9	Tabulazione
'\b'	BS - 8	Backspace – cancella ultimo car.
'\a'	BEL - 7	Emette un "bip"
'\r'	CR - 13	Torna alla prima colonna

14

Punteggiatura speciale in C

- Alcuni caratteri hanno un significato particolare dentro gli apici. Per poterli inserire come carattere esistono apposite sequenze di escape

C	ASCII	Significato
'\\'	\	Immette un backslash
'\''	'	Immette un apice singolo
'\"'	"	Immette un apice doppio
'\ooo'	ooo	Immette in carattere ASCII con codice (ottale) ooo
'\xhh'	hh	Immette in carattere ASCII con codice (esadecimale) hh

Input/output di char

- Esistono due insiemi di funzioni che permettono di leggere e stampare variabili di tipo char:
 - Le funzioni printf/sccanf, usando lo specificatore di formato "%c"
 - Le funzioni putchar e getchar
- In entrambi i casi è sufficiente includere la libreria <stdio.h>
- È possibile mescolare liberamente le due famiglie di funzioni

17

```

#include <stdio.h>
#include <string.h>
#include <ctype.h>

#define MAXPAROLA 30
#define MAXIGRA 80

int main(int argc, char *argv[])
{
    int lungMAXPAROLA; /* variezza di parola */
    char s[MAXPAROLA]; /* stringa contenente la parola */
    char riga[MAXIGRA];
    int i, j, lungriga;
    i=0; j=0;

    for(i=0; i<MAXPAROLA; i++)
        riga[i]=0;

    if(argc > 1)
    {
        strcpy(riga, argv[1]);
        lungriga = strlen(argv[1]);
        if(lungriga > MAXIGRA)
            lungriga = MAXIGRA;
        i=0;
        while((riga[i] != '\0') && (i < lungriga))
            s[i] = riga[i];
        s[i] = '\0';
    }
    else
        s[0] = '\0';

    printf("La parola inserita è: %s\n", s);
}

```

Il tipo char

Input/output di char

```
char ch ;
```

```
printf("%c", ch) ;
```

Stampa di caratteri

```

char ch ;
putchar(ch) ;

```

18

Lettura di caratteri

```
char ch ;  
scanf("%c", &ch) ;
```

```
char ch ;  
ch = getchar() ;
```

19

Suggerimenti (1/2)

- La funzione printf è più comoda quando occorre stampare altri caratteri insieme a quello desiderato
 - printf("La risposta e': %c\n", ch) ;
 - printf("Codice: %c%d\n", ch, num) ;
- La funzione putchar è più comoda quando occorre stampare semplicemente il carattere
 - for(ch='a'; ch<='z'; ch++)
putchar(ch) ;

20

Suggerimenti (2/2)

- La funzione getchar è generalmente più comoda in tutti i casi
 - printf("\vuoi continuare (s/n)? ");
ch = getchar() ;

21

Bufferizzazione dell'input-output

- Tutte le funzioni della libreria <stdio.h> gestiscono l'input-output in modo **bufferizzato**
 - Per maggior efficienza, i caratteri non vengono trasferiti immediatamente dal programma al terminale (o viceversa), ma solo a gruppi
 - È quindi possibile che dopo una putchar, il carattere **non compaia immediatamente sullo schermo**
 - Analogamente, la getchar **non restituisce il carattere finché l'utente non preme invio**

22

Conseguenza pratica

```
char ch,ch2 ;  
printf("Dato: ");  
ch = getchar() ;  
ch2 = getchar() ;
```

Dato: _

Il programma stampa l'invito ad inserire un dato

23

Conseguenza pratica

```
char ch,ch2 ;  
printf("Dato: ");  
ch = getchar() ;  
ch2 = getchar() ;
```

Dato: _

getchar blocca il programma in attesa del dato

24

Conseguenza pratica

```
char ch,ch2 ;  
printf("Dato: ");  
ch = getchar() ;  
ch2 = getchar() ;
```

Dato: a_

L'utente immette 'a', il programma non lo riceve

25

Conseguenza pratica

```
char ch,ch2 ;  
printf("Dato: ");  
ch = getchar() ;  
ch2 = getchar() ;
```

Dato: a

L'utente immette Invio, il programma prosegue

26

Conseguenza pratica

```
char ch,ch2 ;  
printf("Dato: ");  
ch = getchar() ;  
ch2 = getchar() ;
```

Dato: a

_

Ora ch='a', il programma fa un'altra getchar()

27

Conseguenza pratica

```
char ch,ch2 ;  
printf("Dato: ");  
ch = getchar() ;  
ch2 = getchar() ;
```

Dato: a

_

C'era già un carattere pronto: Invio! ch2='\n'

29

Conseguenza pratica

```
char ch,ch2 ;  
printf("Dato: ");  
ch = getchar() ;  
ch2 = getchar() ;
```

Dato: a

_

Il programma **non** si blocca in attesa dell'utente

28

Consigli pratici

- Ricordare che l'utente deve sempre premere Invio, anche se il programma richiede un singolo carattere
- Ricordare che, se l'utente inserisce più di un carattere, questi verranno restituiti uno ad uno nelle getchar successive
- Ricordare che l'Invio viene letto come tutti gli altri caratteri

30

Soluzione proposta

```
char ch, temp ;  
  
printf("Dato: ");  
  
ch = getchar() ; /* leggi il dato */  
  
/* elimina eventuali caratteri successivi  
ed il \n che sicuramente ci sarà */  
do {  
    temp = getchar() ;  
} while (temp != '\n') ;
```

31

Soluzione proposta

```
char ch, temp ;  
  
printf("Dato: ");  
  
ch = getchar() ; /* leggi il dato */  
  
/* elimina eventuali caratteri successivi  
ed il \n che sicuramente ci sarà */  
do {  
    temp = getchar() ;  
} while (temp != '\n') ;
```

32



Il tipo char

Operazioni sui char

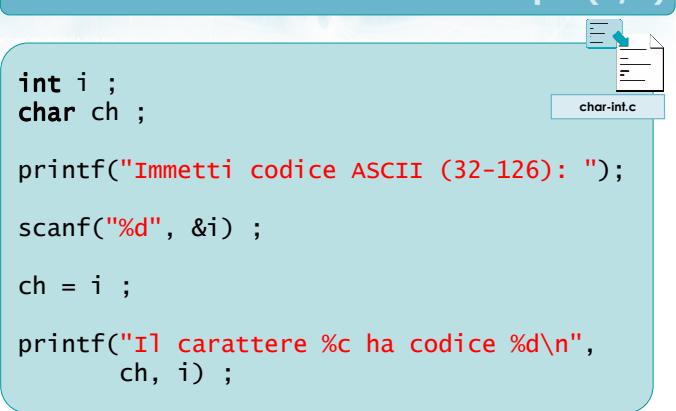
34



Conversione ASCII-Carattere

- Una variabile di tipo char è allo stesso tempo
 - Il valore numerico del codice ASCII del carattere
 - printf("%d", ch) ;
 - i = ch ;
 - ch = j ;
 - ch = 48 ;
 - Il simbolo corrispondente al carattere ASCII
 - printf("%c", ch) ;
 - putchar(ch) ;
 - ch = 'Z' ;
 - ch = '4' ;

35



Esempio (1/3)

```
int i ;  
char ch ;  
  
printf("Immetti codice ASCII (32-126): ");  
  
scanf("%d", &i) ;  
  
ch = i ;  
  
printf("Il carattere %c ha codice %d\n",  
      ch, i) ;
```

36

Esempio (2/3)

```
printf("Immetti un carattere: ") ;
ch = getchar() ;

while( getchar() != '\n' )
/**/ ;

i = ch ;

printf("Il carattere %c ha codice %d\n",
ch, i) ;
```



char-int.c

37

Esempio (3/3)

es> Prompt dei comandi

```
Immetti un codice ASCII (32-126): 44
Il carattere , ha codice ASCII 44
```

```
Immetti un carattere: $
Il carattere $ ha codice ASCII 36
```



char-int.c

38

Scansione dell'alfabeto

- ▶ È possibile generare tutte le lettere dell'alfabeto, in ordine, grazie al fatto che nella tabella ASCII esse compaiono consecutive e ordinate

```
char ch ;

for( ch = 'A' ; ch <= 'z' ; ch++ )
    putchar(ch) ;

putchar('\n') ;
```

39

Verifica se è una lettera

- ▶ Per sapere se un carattere è alfabetico, è sufficiente verificare se cade nell'intervallo delle lettere (maiuscole o minuscole)

```
if( ch>='A' && ch<='Z' )
    printf("%c lettera maiuscola\n", ch) ;

if( ch>='a' && ch<='z' )
    printf("%c lettera minuscola\n", ch) ;

if( (ch>='A' && ch<='Z') || 
    (ch>='a' && ch<='z') )
    printf("%c lettera\n", ch) ;
```

Verifica se è una cifra

- ▶ Per sapere se un carattere è numerico ('0'-'9'), è sufficiente verificare se cade nell'intervallo delle cifre

```
if( ch>='0' && ch<='9' )
    printf("%c cifra numerica\n", ch) ;
```

41

Valore di una cifra

- ▶ Conoscere il valore decimale di un carattere numerico ('0'-'9'), è sufficiente calcolare la "distanza" dalla cifra '0'

```
if( ch>='0' && ch<='9' )
{
    printf("%c cifra numerica\n", ch) ;
    val = ch - '0' ;
    printf("Il suo valore e': %d", val) ;
}
```

42

Da minuscolo a maiuscolo (1/2)

- I codici ASCII delle lettere maiuscole e delle minuscole differiscono solamente per una costante:
 - 'A' = 65 ... 'z' = 90
 - 'a' = 97 ... 'Z' = 122
- Se ch è una lettera minuscola
 - ch - 'a' è la sua posizione nell'alfabeto
 - (ch - 'a') + 'A' è la corrispondente lettera maiuscola

43

Da minuscolo a maiuscolo (2/2)

- Possiamo interpretare la conversione come una traslazione della quantità ('A' - 'a')

```
if( ch>='a' && ch<='z' )  
{  
    printf("%c lettera minuscola\n", ch) ;  
    ch2 = ch + ('A'-'a') ;  
    printf("La maiuscola e': %c\n", ch2) ;  
}
```

44

Confronto alfabetico

- Se due caratteri sono **entrambi maiuscoli** (o entrambi minuscoli) è sufficiente confrontare i rispettivi codici ASCII

```
if( ch < ch2 )  
    printf("%c viene prima di %c", ch, ch2) ;  
else  
    printf("%c viene prima di %c", ch2, ch) ;
```

45

```
#include <stdio.h>  
#include <string.h>  
#include <ctype.h>  
  
#define MAXPAROLA 30  
#define MAXRIGA 80  
  
int main(int argc, char *argv[]){  
    int lungMAXPAROLA; /* variezza di parola */  
    char maxParola[MAXPAROLA]; /* stringa che contiene la parola più grande */  
    char riga[MAXRIGA]; /* riga da leggere */  
    int i, j, k, lungRiga; /* contatori */  
    FILE *f; /* file */  
  
    if(argc < 2){  
        printf("Usage: %s <file>\n", argv[0]);  
        exit(1);  
    }  
    if((f=fopen(argv[1], "r"))==NULL){  
        perror("Error opening file");  
        exit(1);  
    }  
    while(fgets(riga, MAXRIGA, f)!=NULL){  
        if(strlen(maxParola)<=strlen(riga)){  
            strcpy(maxParola, riga);  
        }  
        else{  
            if(strcmp(maxParola, riga)>0){  
                strcpy(maxParola, riga);  
            }  
        }  
    }  
    printf("The longest word is: %s\n", maxParola);  
    exit(0);  
}
```

Il tipo char

Esercizio “Quadrati di lettere”

- Si scriva un programma in linguaggio C che stampi su video una serie di quadrati, composti dalle successive lettere dell'alfabeto, di dimensioni sempre crescenti:
 - Un quadrato 1x1 di lettere A
 - Un quadrato 2x2 di lettere B
 - Un quadrato 3x3 di lettere C
 - ...eccetera

47

Esercizio “Quadrati di lettere”

Analisi

```
PS: Prompt dei comandi  
Quanti quadrati vuoi stampare? 4  
A  
BB  
BB  
CCC  
CCC  
CCC  
DDDD  
DDDD  
DDDD  
DDDD
```

48

Soluzione (1/2)

```
int i, N ;
int riga, col ;
char ch ;

printf("Quanti quadrati? ") ;
scanf("%d", &N) ;

while(N<1 || N>26)
{
    printf("Deve essere tra 1 e 26\n");
    printf("Quanti quadrati? ") ;
    scanf("%d", &N) ;
}
```

quadrati.c

49

Soluzione (2/2)

```
for( i=0; i<N; i++ )
{
    /* stampa un quadrato
       di dimensione (i+1) */

    ch = i + 'A' ;

    for(riga=0; riga<i+1; riga++)
    {
        for(col=0; col<i+1; col++)
            putchar(ch);
        putchar('\n') ;
    }
    putchar('\n') ;
}
```

quadrati.c

50

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

#define MAXPAROLA 30
#define MAXSIGA 80

int main(int argc, char *argv[])
{
    int lung(MAXPAROLA); /* valore di costante
    dove lung è la lunghezza massima della parola */
    int i, indice, lungparola;
    char digit[MAXSIGA];
    FILE *file;

    if(argc != 2)
    {
        printf("ERRORE: non è possibile aprire il file %s", argv[1]);
        exit(1);
    }

    if(indice = fopen(argv[1], "r")) // apre il file
    {
        if(fscanf(indice, "%s", digit) == 1) // legge la parola
        {
            lungparola = strlen(digit); // calcola la lunghezza della parola
            for(i=0; i<lungparola; i++)
            {
                if(isalpha(digit[i])) // se è un carattere alfabetico
                {
                    digit[i] = toupper(digit[i]); // lo trasforma in maiuscolo
                }
            }
            printf("%s", digit);
        }
        else
        {
            printf("ERRORE: impossibile aprire il file %s", argv[1]);
            exit(1);
        }
    }
    else
    {
        printf("ERRORE: non è possibile aprire il file %s", argv[1]);
        exit(1);
    }
}
```

Caratteri e stringhe

Vettori di caratteri

2

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

#define MAXPAROLA 30
#define MAXSIGA 80

int main(int argc, char *argv[])
{
    int lung(MAXPAROLA); /* valore di costante
    dove lung è la lunghezza massima della parola */
    int i, indice, lungparola;
    char digit[MAXSIGA];
    FILE *file;

    if(argc != 2)
    {
        printf("ERRORE: non è possibile aprire il file %s", argv[1]);
        exit(1);
    }

    if(indice = fopen(argv[1], "r")) // apre il file
    {
        if(fscanf(indice, "%s", digit) == 1) // legge la parola
        {
            lungparola = strlen(digit); // calcola la lunghezza della parola
            for(i=0; i<lungparola; i++)
            {
                if(isalpha(digit[i])) // se è un carattere alfabetico
                {
                    digit[i] = toupper(digit[i]); // lo trasforma in maiuscolo
                }
            }
            printf("%s", digit);
        }
        else
        {
            printf("ERRORE: impossibile aprire il file %s", argv[1]);
            exit(1);
        }
    }
    else
    {
        printf("ERRORE: non è possibile aprire il file %s", argv[1]);
        exit(1);
    }
}
```

Vettori di caratteri

Il tipo stringa

4

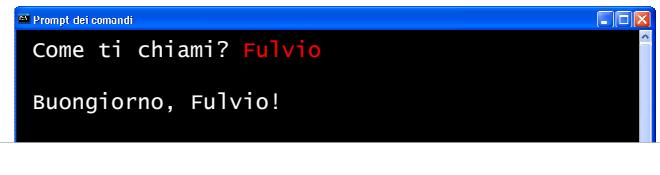
`char saluto[10] ;`

B u o n g i o r n o

```
(f) lung(4,3)
Spiegazione: "ERRORE: non è possibile aprire il nome del file (%s)".
Salvo che...
1. lung(4,3)
2. lung(4,3)
3. lung(4,3)
```

Esempio

- Si realizzi un programma in linguaggio C che acquisisca da tastiera il nome dell'utente (una stringa di max 20 caratteri), e stampi a video un saluto per l'utente stesso



5

```
(f) lung(4,3)
Spiegazione: "ERRORE: non è possibile aprire il nome del file (%s)".
Salvo che...
1. lung(4,3)
2. lung(4,3)
3. lung(4,3)
```

Soluzione (1/3)

```
const int MAX = 20 ;
char nome[MAX] ;
int N ;
char ch ;
int i ;

printf("Come ti chiami? ") ;
N = 0 ;
```



soluti.c

6

Soluzione (2/3)

```
ch = getchar() ;
while( ch != '\n' && N<MAX )
{
    nome[N] = ch ;
    N++ ;
    ch = getchar() ;
}
```

7

Soluzione (3/3)

```
printf("Buongiorno, ") ;
for(i=0; i<N; i++)
    putchar( nome[i] ) ;
printf("!\n") ;
```

8

Commenti (1/2)

- » Qualsiasi operazione sulle stringhe si può realizzare agendo opportunamente su vettori di caratteri, gestiti con occupazione variabile
- » Così facendo, però vi sono alcuni svantaggi
 - Per ogni vettore di caratteri, occorre definire un'opportuna variabile che ne indichi la lunghezza
 - Ogni operazione, anche elementare, richiede l'uso di cicli `for/while`

9

Commenti (2/2)

- » Alcune convenzioni ci possono aiutare
 - Gestire in modo standard i vettori di caratteri usati per memorizzare stringhe
 - Apprendere le tecniche solitamente utilizzate per compiere le operazioni più frequenti
- » Molte funzioni di libreria seguono queste convenzioni
 - Conoscere le funzioni di libreria ed utilizzarle per accelerare la scrittura del programma

10

```
#include "stdlib.h"
#include "string.h"
#include "ctype.h"

#define MAXPAROLA 30
#define MAXRIGA 80

int main(int argc, char *argv[])
{
    int lungMAXPAROLA; /* vettore di dimensione
        lunghezza massima della parola */
    char nomeMAXRIGA[ ]; /* vettore di dimensione
        lunghezza massima della riga */
    int i, indice, lunghezza;
    FILE *f;

    if(argc < 2)
    {
        printf("Usage: %s file\n", argv[0]);
        exit(1);
    }

    f = fopen(argv[1], "r");
    if(f == NULL)
    {
        perror("Error opening file");
        exit(1);
    }

    fgets(nomeMAXRIGA, MAXRIGA, f);
    lungMAXPAROLA = strlen(nomeMAXRIGA);
    nomeMAXRIGA[lungMAXPAROLA] = '\0';
    nomeMAXRIGA[lungMAXPAROLA+1] = '\0';

    for(i=0; i<lungMAXPAROLA; i++)
    {
        if(isalpha(nomeMAXRIGA[i]))
            indice++;
        else
            break;
    }

    if(indice > lungMAXPAROLA)
    {
        printf("Error: too many characters in the word\n");
        exit(1);
    }
}
```

Vettori di caratteri

Terminatore nullo

Lunghezza di una stringa

- » Vi sono due tecniche per determinare la lunghezza di una stringa
 1. utilizzare una variabile intera che memorizzi il numero di caratteri validi

```
char nome[10] ;
int lungh_nome ;
```

6

12

Lunghezza di una stringa

- Vi sono due tecniche per determinare la lunghezza di una stringa

- utilizzare una variabile intera che memorizzi il numero di caratteri validi

```
char nome[10] ; F u l v i o z ! $ .  
int lungh_nome ; 6
```

- utilizzare un carattere "speciale", con funzione di terminatore, dopo l'ultimo carattere valido

```
char nome[10] ; F u l v i o 0 ! $ .
```

13

Carattere terminatore

- Il carattere "terminatore" deve avere le seguenti caratteristiche

- Fare parte della tabella dei codici ASCII
 - Deve essere rappresentabile in un char
- Non comparire mai nelle stringhe utilizzate dal programma
 - Non deve confondersi con i caratteri "normali"

- Inoltre il vettore di caratteri deve avere una posizione libera in più, per memorizzare il terminatore stesso

14

Terminatore standard in C

- Per convenzione, in C si sceglie che tutte le stringhe siano rappresentate mediante un carattere terminatore
- Il terminatore corrisponde al carattere di codice ASCII pari a zero

- nome[6] = 0 ;
- nome[6] = '\0' ;

```
F u l v i o 0 ! $ .
```

15

Vantaggi

- Non è necessaria un'ulteriore variabile intera per ciascuna stringa
- L'informazione sulla lunghezza della stringa è interna al vettore stesso
- Tutte le funzioni della libreria standard C rispettano questa convenzione
 - Si aspettano che la stringa sia terminata
 - Restituiscono sempre stringhe terminate

16

Svantaggi

- Necessario 1 byte in più
 - Per una stringa di N caratteri, serve un vettore di N+1 elementi
- Necessario ricordare di aggiungere sempre il terminatore
- Impossibile rappresentare stringhe contenenti il carattere ASCII 0

17

Esempio

- Si realizzi un programma in linguaggio C che acquisisca da tastiera il nome dell'utente (una stringa di max 20 caratteri), e stampi a video un saluto per l'utente stesso

```
Prompt dei comandi  
Come ti chiami? Fulvio  
Buongiorno, Fulvio!
```

18

Soluzione (1/3)

```
const int MAX = 20 ;
char nome[MAX+1] ;
char ch ;
int i ;

printf("Come ti chiami? ") ;

i = 0 ;
```

salutIO.c

19

Soluzione (2/3)

```
i = 0 ;

ch = getchar() ;

while( ch != '\n' && i<MAX )
{
    nome[i] = ch ;
    i++ ;
    ch = getchar() ;
}
/* aggiunge terminatore nullo */
nome[i] = '\0' ;
```

salutIO.c

20

Soluzione (3/3)

```
printf("Buongiorno, ") ;

for(i=0; nome[i]!='\0'; i++)
    putchar( nome[i] ) ;

printf("\n") ;
```

salutIO.c

21

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

#define MAXPAROLA 30
#define MAXRIGA 80

int main(int argc, char *argv[])
{
    int lungMAXPAROLA; /* variezza di parola */
    char nome[MAXRIGA]; /* stringa che contiene il nome */
    char singlMAXRIGA; /* singola riga */
    int i, j, lunghezza;
    FILE *f;

    f=fopen(argv[1], "r");
    if(f==NULL)
    {
        printf("ERRORE: non è possibile aprire il file %s\n", argv[1]);
        exit(1);
    }

    /* legge singlRIGA */
    singlMAXRIGA=fgets(singlMAXRIGA, MAXRIGA, f);
    if(singlMAXRIGA==NULL)
    {
        printf("ERRORE: impossibile leggere il file %s\n", argv[1]);
        exit(1);
    }

    while(fgets(nome, MAXRIGA, f)!=NULL)
```

Vettori di caratteri

Input/output di stringhe

I/O di stringhe

- Diamo per scontato di utilizzare la convenzione del terminatore nullo
- Si possono utilizzare
 - Funzioni di lettura e scrittura carattere per carattere
 - Come nell'esercizio precedente
 - Funzioni di lettura e scrittura di stringhe intere
 - scanf e printf
 - gets e puts

23

Lettura di stringhe con scanf

- Utilizzare la funzione scanf con lo specificatore di formato "%s"
- La variabile da leggere deve essere il nome di un vettore di caratteri
 - Non utilizzare le parentesi quadre
 - Non utilizzare la &
- Legge ciò che viene immesso da tastiera, fino al primo spazio o fine linea (esclusi)
 - Non adatta a leggere nomi composti (es. "Pier Paolo")

24

Esempio

```
const int MAX = 20 ;  
char nome[MAX+1] ;  
  
printf("Come ti chiami? ") ;  
scanf("%s", nome) ;
```

25

Esempio

```
const int MAX = 20 ;  
char nome[MAX+1] ;  
  
printf("Come ti chiami? ") ;  
gets(nome) ;
```

27

Esempio

```
printf("Buongiorno, ") ;  
printf("%s", nome) ;  
printf("\n") ;  
  
printf("Buongiorno, %s!\n", nome) ;
```

29

Lettura di stringhe con gets

- La funzione gets è pensata appositamente per acquisire una stringa
- Accetta un parametro, che corrisponde al nome di un vettore di caratteri
 - Non utilizzare le parentesi quadre
- Legge ciò che viene immesso da tastiera, fino al fine linea (escluso), e compresi eventuali spazi
 - Possibile leggere nomi composti (es. "Pier Paolo")

26

Scrittura di stringhe con printf

- Utilizzare la funzione printf con lo specificatore di formato "%s"
- La variabile da stampare deve essere il nome di un vettore di caratteri
 - Non utilizzare le parentesi quadre
- È possibile combinare la stringa con altre variabili nella stessa istruzione

28

Scrittura di stringhe con puts

- La funzione puts è pensata appositamente per stampare una stringa
- La variabile da stampare deve essere il nome di un vettore di caratteri
 - Non utilizzare le parentesi quadre
- Va a capo automaticamente
 - Non è possibile stampare altre informazioni sulla stessa riga

30

Esempio

```
printf("Buongiorno, ") ;  
puts(nome) ;  
/* No!! printf("\n") ; */
```

31

Conclusione

- Utilizzare sempre la convenzione del terminatore nullo
- Ricordare di allocare un elemento in più nei vettori di caratteri
- Utilizzare quando possibile le funzioni di libreria predefinite
 - In lettura, prediligere gets
 - In scrittura
 - printf è indicata per messaggi composti
 - puts è più semplice se si ha un dato per riga

32

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

#define MAXPAROLA 30
#define MAXSIGA 80

int main(int argc, char *argv[])
{
    int lung(MAXPAROLA); /* valore di controllo della frequenza delle lunghezze delle parole */
    int i, indice, lunghezza;
    char s[MAXSIGA];
    char s1[MAXSIGA];
    int lun;

    if(argc != 2)
    {
        printf("Errore: l'unico argomento deve essere il nome del file (*.c)\n");
        exit(1);
    }
    i = fopen(argv[1], "r");
    if(i == NULL)
    {
        printf("Errore: impossibile aprire il file %s", argv[1]);
        exit(1);
    }

    while(fgets(s, MAXSIGA, i) != NULL)
    {
        if(s[lung] == '\0')
        {
            printf("Lunghezza: %d\n", lung);
            lung++;
        }
        else
        {
            if(lung > MAXPAROLA)
            {
                printf("Lunghezza: %d\n", MAXPAROLA);
                lung = 1;
            }
            else
            {
                s[lung] = '\0';
                lung++;
            }
        }
    }
    fclose(i);
}
```

Caratteri e stringhe

Operazioni elementari sulle stringhe

- ▶ Lunghezza
- ▶ Copia di stringhe
- ▶ Concatenazione di stringhe
- ▶ Confronto di stringhe
- ▶ Ricerca di sotto-stringhe
- ▶ Ricerca di parole

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

#define MAXPAROLA 30
#define MAXSIGA 80

int main(int argc, char *argv[])
{
    int lung(MAXPAROLA); /* valore di controllo della frequenza delle lunghezze delle parole */
    int i, indice, lunghezza;
    char s[MAXSIGA];
    char s1[MAXSIGA];
    int lun;

    if(argc != 2)
    {
        printf("Errore: l'unico argomento deve essere il nome del file (*.c)\n");
        exit(1);
    }
    i = fopen(argv[1], "r");
    if(i == NULL)
    {
        printf("Errore: impossibile aprire il file %s", argv[1]);
        exit(1);
    }

    while(fgets(s, MAXSIGA, i) != NULL)
    {
        if(s[lung] == '\0')
        {
            printf("Lunghezza: %d\n", lung);
            lung++;
        }
        else
        {
            if(lung > MAXPAROLA)
            {
                printf("Lunghezza: %d\n", MAXPAROLA);
                lung = 1;
            }
            else
            {
                s[lung] = '\0';
                lung++;
            }
        }
    }
    fclose(i);
}
```

Operazioni elementari sulle stringhe

Lunghezza

Lunghezza di una stringa

- ▶ La lunghezza di una stringa si può determinare ricercando la posizione del terminatore nullo

```
char s[MAX+1] ;
int lun ;
```

s	S	a	1	v	e	Ø	3	r	w	t
0	1	2	3	4	5					

4

Calcolo della lunghezza

```
const int MAX = 20 ;
char s[MAX+1] ;
int lun ;
int i ;

... /* lettura stringa */

for( i=0 ; s[i] != 0 ; i++ )
/* Niente */ ;

lun = i ;
```

5

La funzione strlen

- ▶ Nella libreria standard C è disponibile la funzione `strlen`, che calcola la lunghezza della stringa passata come parametro
- ▶ Necessario includere `<string.h>`

```
const int MAX = 20 ;
char s[MAX+1] ;
int lun ;

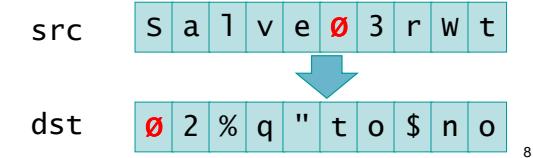
... /* lettura stringa */

lun = strlen(s) ;
```

6

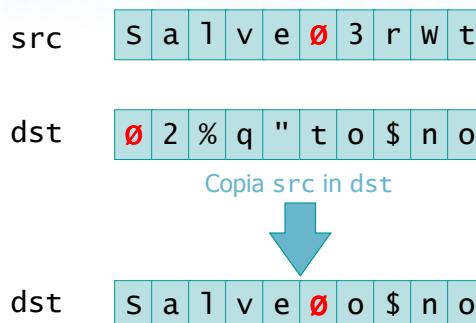
Operazioni elementari sulle stringhe

Copia di stringhe



8

Risultato della copia



```
const int MAXS = 20, MAXD = 30 ;
char src[MAXS+1] ;
char dst[MAXD+1] ;
int i ;
... /* lettura stringa src */
for( i=0 ; src[i] != 0 ; i++ )
    dst[i] = src[i] ; /* copia */
dst[i] = 0 ; /* aggiunge terminatore */
```

10

La funzione strcp

- ▶ Nella libreria standard C, includendo `<string.h>`, è disponibile la funzione `strcpy` che effettua la copia di stringhe
 - Primo parametro: stringa destinazione
 - Secondo parametro: stringa sorgente

```
const int MAXS = 20, MAXD = 30
char src[MAXS+1] ;
char dst[MAXD+1] ;

... /* lettura stringa src */

strcpy(dst, src) ;
```

Avvertenze

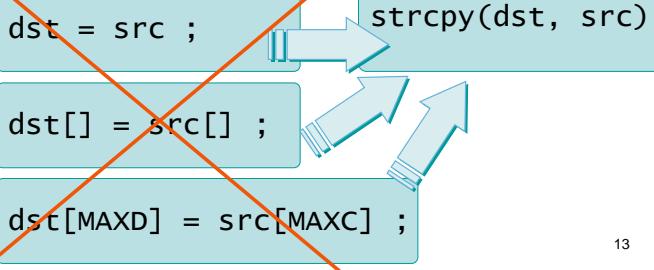
- Nella stringa destinazione vi deve essere un numero sufficiente di locazioni libere
 - `MAXD+1 >= strlen(src)+1`
 - Il contenuto precedente della stringa destinazione viene perso
 - La stringa sorgente non viene modificata
 - Il terminatore nullo
 - Deve essere aggiunto in coda a dst
 - La `strncpy` pensa già autonomamente a farlo

12



Errore frequente

- Per effettuare una copia di stringhe **non** si può assolutamente utilizzare l'operatore =
- Necessario usare `strcpy`



13

Operazioni elementari sulle stringhe

Concatenazione di stringhe

- L'operazione di concatenazione corrisponde a creare una nuova stringa composta dai caratteri di una prima stringa, **seguiti** dai caratteri di una seconda stringa

sa	S	a	l	v	e	Ø	3	r	w	t
----	---	---	---	---	---	---	---	---	---	---

sb	m	o	n	d	o	Ø	o	\$	n	o
----	---	---	---	---	---	---	---	----	---	---

Concatenazione di sa con sb

S	a	l	v	e	Ø	w	z	3	w	7	w	1	Q	r
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

15

Esempio

sa	S	a	l	v	e	Ø	w	z	3	w	7	w	1	Q	r
----	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

sb	m	o	n	d	o	Ø	h	!	L	.	2	x	y	E	P
----	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

Concatenazione di sa con sb

sa	S	a	l	v	e	Ø	w	z	3	w	7	w	1	Q	r
----	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

sb	m	o	n	d	o	Ø	h	!	L	.	2	x	y	E	P
----	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

17

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

#define MAXPAROLA 30
#define MAXSIGLA 60

int main(int argc, char *argv[])
{
    int lunghezzaParola; /* *vorice di controllo */
    int lunghezza e delle lunghezze delle parole */
    int lunghezzaSigla; /* *orifici, lunghezza */
    int i, j, k;
    char parola[MAXPAROLA];
    char sigla[MAXSIGLA];
    int posizioneParola = 0;
    int posizioneSigla = 0;

    if(argc != 2)
    {
        fprintf(stderr, "ERRORE: non sono presenti dati da inserire nel file\n");
        exit(1);
    }

    if((lunghezzaParola = strlen(argv[1])) > MAXPAROLA)
    {
        fprintf(stderr, "ERRORE: impossibile aprire il file %s", argv[1]);
        exit(1);
    }

    if((lunghezzaSigla = strlen(argv[2])) > MAXSIGLA)
    {
        fprintf(stderr, "ERRORE: impossibile aprire il file %s", argv[2]);
        exit(1);
    }

    posizioneParola = 0;
    posizioneSigla = 0;
```

Operazioni elementari sulle stringhe

Concatenazione di stringhe

- Per maggior semplicità, in C l'operazione di concatenazione scrive il risultato **nello stesso vettore** della prima stringa
- Il valore precedente della prima stringa viene così perso
- Per memorizzare altrove il risultato, o per non perdere la prima stringa, è possibile ricorrere a stringhe temporanee ed alla funzione `strcpy`

Semplificazione

16

Esempio

sa	S	a	l	v	e	Ø	w	z	3	w	7	w	1	Q	r
----	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

sb	m	o	n	d	o	Ø	h	!	L	.	2	x	y	E	P
----	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

Concatenazione di sa con sb

Algoritmo di concatenazione

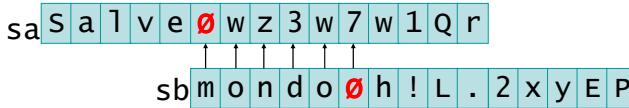
- Trova la fine della prima stringa

sa	S	a	l	v	e	Ø	w	z	3	w	7	w	1	Q	r
----	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

18

Algoritmo di concatenazione

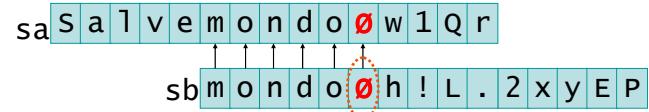
- ▶ Trova la fine della prima stringa
- ▶ Copia la seconda stringa nel vettore della prima, a partire dalla posizione del terminatore nullo (sovrascrivendolo)



19

Algoritmo di concatenazione

- ▶ Trova la fine della prima stringa
- ▶ Copia la seconda stringa nel vettore della prima, a partire dalla posizione del terminatore nullo (sovrascrivendolo)
- ▶ Termina la copia non appena trovato il terminatore della seconda stringa



20

Concatenazione

```
const int MAX = 20 ;
char sa[MAX] ;
char sb[MAX] ;
int la ;
int i ;

/* lettura stringhe */

la = strlen(sa) ;
for( i=0 ; sb[i] != 0 ; i++ )
    sa[la+i] = sb[i] ; /* copia */
sa[la+i] = 0 ; /* terminatore */
```

21

La funzione strcat

- ▶ Nella libreria standard C, includendo `<string.h>`, è disponibile la funzione `strcat`, che effettua la concatenazione di stringhe
 - Primo parametro: prima stringa (destinazione)
 - Secondo parametro: seconda stringa

```
const int MAX = 20 ;
char sa[MAX] ;
char sb[MAX] ;

/* lettura stringhe */

strcat(sa, sb) ;
```

22

Avvertenze (1/2)

- ▶ Nella prima stringa vi deve essere un numero sufficiente di locazioni libere
 - `MAX+1 >= strlen(sa)+strlen(sb)+1`
- ▶ Il contenuto precedente della prima stringa viene perso
- ▶ La seconda stringa non viene modificata
- ▶ Il terminatore nullo
 - Deve essere aggiunto in coda alla prima stringa
 - La `strcat` pensa già autonomamente a farlo

23

Avvertenze (2/2)

- ▶ Per concatenare 3 o più stringhe, occorre farlo due a due:
 - `strcat(sa, sb);`
 - `strcat(sa, sc);`
- ▶ È possibile concatenare anche stringhe costanti
 - `strcat(sa, "!");`

24

- Il confronto di due stringhe (es.: sa e sb), mira a determinare se:

- Le due stringhe sono uguali
 - hanno uguale lunghezza e sono composte dagli stessi caratteri nello stesso ordine
- Le due stringhe sono diverse
- La stringa sa precede la stringa sb
 - secondo l'ordine lessicografico imposto dal codice ASCII
 - parzialmente compatibile con l'ordine alfabetico
- La stringa sa segue la stringa sb

26

Operazioni elementari sulle stringhe

Confronto di stringhe

Confronto di uguaglianza

- Ogni carattere di sa deve essere uguale al carattere corrispondente di sb
- Il terminatore nullo deve essere nella stessa posizione
- I caratteri successivi al terminatore vanno ignorati

sa [S a 1 v e 0 o 4 d 1 0 w 1 Q r]
 sb [S a 1 v e 0 h ! L . 2 x y E P]

27

Confronto di uguaglianza

```
const int MAX = 20 ;
char sa[MAX] ;
char sb[MAX] ;
int uguali ;
int i ;
...
uguali = 1 ;
for( i=0 ; sa[i]!=0 && sb[i]!=0 ; i++ )
{
    if(sa[i]!=sb[i])
        uguali = 0 ;
}
if(sa[i]==0 || sb[i]==0)
    uguali = 0 ;
```

29

Confronto di uguaglianza

Confronto di uguaglianza

```
const int MAX = 20 ;
char sa[MAX] ;
char sb[MAX] ;
int uguali ;
int i ;
...
uguali = 1 ;
for( i=0 ; sa[i]!=0 && sb[i]!=0 ; i++ )
{
    if(sa[i]!=sb[i])
        uguali = 0 ;
}
if(sa[i]==0 || sb[i]==0)
    uguali = 0 ;
```

28

Confronto di uguaglianza

Confronto di uguaglianza

```
const int MAX = 20 ;
char sa[MAX] ;
char sb[MAX] ;
int uguali ;
int i ;
...
uguali = 1 ;
for( i=0 ; sa[i]!=0 && sb[i]!=0 ; i++ )
{
    if(sa[i]!=sb[i])
        uguali = 0 ;
}
if(sa[i]==0 || sb[i]==0)
    uguali = 0 ;
```

30

Confronto di uguaglianza

```
const int MAX = 20 ;
char sa[MAX] ;
char sb[MAX] ;
int uguali ;
int i ;
...
uguali = 1 ;
for( i=0 ; sa[i]!=0
{
    if(sa[i]!=sb[i])
        uguali = 0 ;
}
if(sa[i]==0 || sb[i]==0)
    uguali = 0 ;
```

Verifica che tutti i caratteri incontrati siano uguali.
Se no, poni a 0 il flag uguali.

31

Confronto di uguaglianza

```
const int MAX = 20 ;
char sa[MAX] ;
char sb[MAX] ;
int uguali ;
int i ;
...
uguali = 1 ;
for( i=0 ; sa[i]!=0
{
    if(sa[i]!=sb[i])
        uguali = 0 ;
}
if(sa[i]==0 || sb[i]==0)
    uguali = 0 ;
```

In questo punto sicuramente una delle due stringhe è arrivata al terminatore. Se non lo è anche l'altra, allora non sono uguali!

32

Confronto di ordine

- » Verifichiamo se sa "è minore di" sb. Partiamo con i=0
- » Se sa[i]<sb[i], allora sa è minore
- » Se sa[i]>sb[i], allora sa non è minore
- » Se sa[i]=sb[i], allora bisogna controllare i caratteri successivi (i++)
- » Il terminatore nullo conta come "minore" di tutti

sa S a 1 v e 0 o 4 d 1 0 w 1 Q r

sb S a 1 u t e 0 ! L . 2 x y E P

33

Confronto di ordine (1/2)

```
const int MAX = 20 ;
char sa[MAX] ;
char sb[MAX] ;
int minore ;
int i ;
...
minore = 0 ;
for( i=0 ; sa[i]!=0 && sb[i]!=0
    && minore==0; i++ )
{
    if(sa[i]<sb[i])
        minore = 1 ;
    if(sa[i]>sb[i])
        minore = -1 ;
}
```

34

Confronto di ordine (2/2)

```
if(minore==0 && sa[i]==0 && sb[i]==0)
    minore=1 ;
if(minore==1)
    printf("%s e' minore di %s\n",
           sa, sb ) ;
```

35

Commenti

```
minore = 0 ;
for( i=0 ; sa[i]!=0 && sb[i]!=0
    && minore==0; i++ )
{
    if(sa[i]<sb[i])
        minore = 1 ;
    if(sa[i]>sb[i])
        minore = -1 ;
}

if(minore==0 && sa[i]==0 && sb[i]==0)
    minore=1 ;

if(minore==1)
    ...
```

Ricerca di esistenza della condizione sa[i]<sb[i].

36

Commenti

```
minore = 0 ;
for( i=0 ; sa[i]!=0 && sb[i]!=0
    && minore==0; i++ )
{
    if(sa[i]<sb[i])
        minore
    if(sa[i]>sb[i])
        minore
}
if(minore==0 &&
    minore==1 ;
if(minore==1)
    ...
```

Cicla fino al primo terminatore nullo, oppure fino a che non si "scopre" chi è minore.
In altre parole, continua a ciclare solo finché le stringhe "sembrano" uguali.

37

```
minore = 0 ;
for( i=0 ; sa[i]!=0 && minore==0;
    && minore!=1; i++ )
{
    if(sa[i]<sb[i])
        minore = 1 ;
    if(sa[i]>sb[i])
        minore = -1 ;
}
if(minore==0 && sa[i]==0 && sb[i]==0)
    ...
Se flag minore==0 continua a ciclare
...
Sicuramente sa è minore di sb
Flag: minore = 1
Sicuramente sa non è minore di sb
Flag: minore = -1
```

38

Commenti

```
minore = 0 ;
for( i=0 ; sa[i]!=0 && sb[i]!=0
    && minore==0; i++ )
{
    if(sa[i]<sb[i])
        minore = 1
    if(sa[i]>sb[i])
        minore = -1
}
if(minore==0 && sa[i]==0 && sb[i]==0)
    minore=1 ;
if(minore==1)
    ...
...  
Se finora erano uguali, ma sa è più corta di sb, allora sa è minore
```

39

La funzione strcmp

- Nella libreria standard C, includendo `<string.h>`, è disponibile la funzione `strcmp`, che effettua il confronto di stringhe
 - Primo parametro: prima stringa
 - Secondo parametro: seconda stringa
 - Valore restituito:
 - <0 se la prima stringa è minore della seconda
 - ==0 se le stringhe sono uguali
 - >0 se la prima stringa è maggiore della seconda

40

Confronti vari

```
const int MAX = 20 ;
char sa[MAX] ;
char sb[MAX] ;
int ris ;
...
ris = strcmp(sa, sb) ;
if(ris<0)
    printf("%s minore di %s\n", sa, sb);
if(ris==0)
    printf("%s uguale a %s\n", sa, sb);
if(ris>0)
    printf("%s maggiore di %s\n", sa, sb);
```



Suggerimento

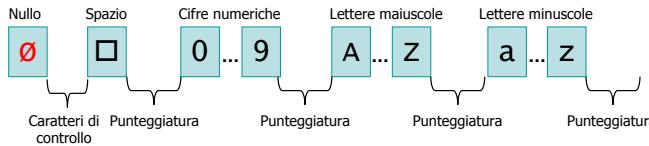
- Per ricordare il significato del valore calcolato da `strcmp`, immaginare che la funzione faccia una "sottrazione" tra le due stringhe
 - `sa - sb`
 - Negativo: sa minore
 - Positivo: sa maggiore
 - Nullo: uguali

```
const int MAX = 20 ;
char sa[MAX] ;
char sb[MAX] ;
int ris ;
...
ris = strcmp(sa, sb) ;
```

42

Ordinamento delle stringhe

- ▶ La funzione `strcmp` lavora confrontando tra loro i codici ASCII dei caratteri
 - ▶ Il criterio di ordinamento è quindi dato dalla posizione dei caratteri nella tabella ASCII



4

Conseguenze

- ▶ Ogni lettera maiuscola precede ogni lettera minuscola
 - *Ciao* precede *ciao*
 - *Zulu* precede *apache*
 - ▶ Gli spazi contano, e precedono le lettere
 - *Qui Quo Qua* precede *QuiQuoQua*
 - ▶ I simboli di punteggiatura contano, ma non vi è una regola intuitiva
 - ▶ L'ordinamento che si ottiene è lievemente diverso da quello “standard” alfabetico

44

```

#include <iostream.h>
#include <string.h>
#include <ctype.h>

#define MAXPAROLA 30
#define MAXIGA 80

int main(void, char *argv[])
{
    int lung(MAXPAROLA); // -> vertice di condizione
    char parola[MAXPAROLA]; // -> parola inserita da tastiera
    char riga(MAXIGA); // -> riga inserita da tastiera
    int i, j, k, l, m, n, pos, lunghezza;
    lung = 1;

    for(i=0; i<MAXPAROLA; i++)
        parola[i] = '\0';

    if(argc > 1)
    {
        for(j=1; j<argc; j++)
        {
            if(strcmp(argv[j], "-f") == 0)
                lung = 0;
            else if(strcmp(argv[j], "-r") == 0)
                lung = 1;
        }
    }

    if(lung == 1)
    {
        cout << "Inserire una parola da inserire & premere Invio: ";
        cin >> parola;
    }
    else
    {
        cout << "Inserire un file da leggere: ";
        cin >> riga;
        if(riga[0] != '#')
            lung = 0;
        else
            lung = 1;
    }

    if(lung == 0)
        getchar();
    else
        system("stty sane");
}

main( legge(dico, MAXIGA, 1) != NULL )

```

Operazioni elementari sulle stringhe

Ricerca di sotto-stringhe

46

Ricerca in una stringa

- ▶ È possibile concepire diversi tipi di ricerche da compiersi all'interno di una stringa:
 - Determinare se un determinato carattere compare all'interno di una stringa data
 - Determinare se una determinata stringa compare integralmente all'interno di un'altra stringa data, in una posizione arbitraria

Ricerca di un carattere (1/2)

Ricerca di un carattere (1/2)

- Detti:
 - *s* una stringa arbitraria
 - *ch* un carattere qualsiasi
 - Determinare se la stringa *s* contiene (una o più volte) il carattere *ch* al suo interno, in qualsiasi posizione

s S a 1 v e Ø o 4 d 1 a w 1 Q r

ch a

4

```
const int MAX = 20 ;
char s[MAX] ;
char ch ;
int trovato ;
int i ;

...
trovato = 0 ;
for( i=0 ; s[i]!=0 && trovato==0; i++ )
{
    if( s[i]==ch )
        trovato = 1 ;
}
```

48

La funzione strchr (1/2)

- » Nella libreria standard C, includendo `<string.h>`, è disponibile la funzione `strchr`, che effettua la ricerca di un carattere
 - Primo parametro: stringa in cui cercare
 - Secondo parametro: carattere da cercare
 - Valore restituito:
 - `!=NULL` se il carattere c'è
 - `==NULL` se il carattere non c'è

49

La funzione strchr (2/2)

```
const int MAX = 20 ;
char s[MAX] ;
char ch ;

...

if(strchr(s, ch)!=NULL)
    printf("%s contiene %c\n", s, ch) ;
```

50

Ricerca di una sotto-stringa

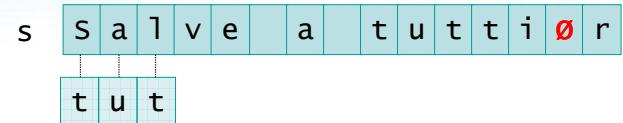
- » Detti:
 - `s` una stringa arbitraria
 - `r` una stringa da ricercare
- » Determinare se la stringa `s` contiene (una o più volte) la stringa `r` al suo interno, in qualsiasi posizione

s 

r 

51

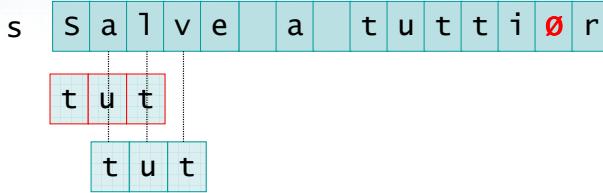
Esempio



r 

52

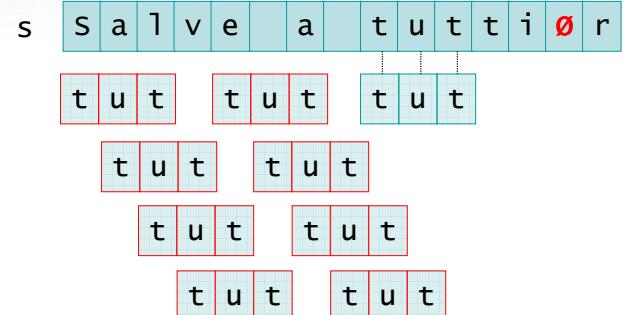
Esempio



r 

53

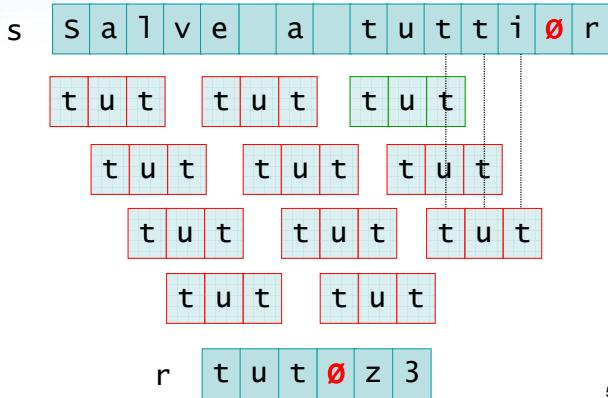
Esempio



r 

54

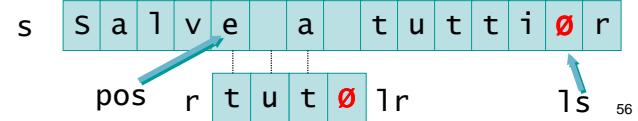
Esempio



55

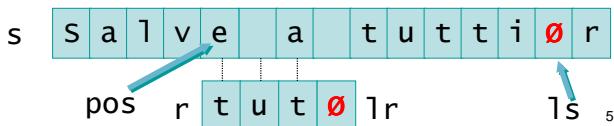
Algoritmo di ricerca

- lr = strlen(r); ls = strlen(s)
- trovato = 0
- Per ogni posizione possibile di r all'interno di s:
pos = 0...ls-1r (compresi)



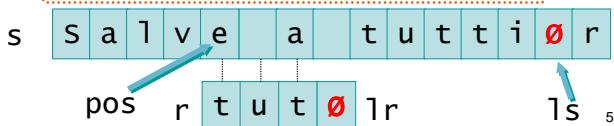
Algoritmo di ricerca

- lr = strlen(r); ls = strlen(s)
- trovato = 0
- Per ogni posizione possibile di r all'interno di s:
pos = 0...ls-1r (compresi)
 - Controlla se i caratteri di r, tra 0 e lr-1, coincidono con i caratteri di s, tra pos e pos+lr-1
 - Se sì, trovato = 1



Algoritmo di ricerca

- lr = strlen(r); ls = strlen(s)
- trovato = 0
- Per ogni posizione possibile di r all'interno di s:
pos = 0...ls-1r
 - Controlla se i caratteri di r, tra 0 e lr-1, coincidono con i caratteri di s, tra pos e pos+lr-1
 - Se sì, trovato = 1



Ricerca di una sotto-stringa (1/2)

```
const int MAX = 20 ;
char s[MAX] ;
char r[MAX] ;
int lr, ls, pos ;
int i ;
int trovato, diversi ;

...
ls = strlen(s);
lr = strlen(r);
```



60

Ricerca di una sotto-stringa (2/2)

```
trovato = 0 ;
for(pos=0; pos<=ls-1r; pos++)
{
    /* confronta r[0...lr-1] con s[pos...pos+lr-1] */
    diversi = 0 ;
    for(i=0; i<lr; i++)
        if(r[i]!=s[pos+i])
            diversi = 1 ;

    if(diversi==0)
        trovato=1 ;
}

if(trovato==1)
    printf("Trovato!\n");
```



substr.c

61

La funzione strstr (1/2)

- Nella libreria standard C, includendo <string.h>, è disponibile la funzione **strstr**, che effettua la ricerca di una sottostringa
 - Primo parametro: stringa in cui cercare
 - Secondo parametro: sotto-stringa da cercare
 - Valore restituito:
 - !=NULL se la sotto-stringa c'è
 - ==NULL se la sotto-stringa non c'è

62

La funzione strstr (2/2)

```
const int MAX = 20 ;
char s[MAX] ;
char r[MAX] ;

...
if(strstr(s, r)!=NULL)
    printf("Trovato!\n");
```

63

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

#define MAXPAROLA 30
#define MAXIGRA 80

int main(int argc, char *argv[])
{
    int lungMAXPAROLA; /* variezza di parola */
    const char *parola; /* parola da riconoscere nella stringa */
    char riga[MAXIGRA];
    int i, j, lungMAX;
    i=0;

    for(i=0; i<MAXPAROLA; i++)
        lungMAX=0;

    if(argc > 1)
    {
        parola=argv[1];
        if(strlen(parola) > MAXPAROLA)
            printf("ERRORE: parola troppo lunga\n");
        else
            i=0;
    }
    else
        i=0;

    while(1)
    {
        if(strstr(riga, parola, i)==NULL)
            break;
        else
            i+=strlen(parola);
    }
}
```

Operazioni elementari sulle stringhe

Ricerca di parole

Ricerca di parole

- Talvolta non interessa trovare una qualsiasi sotto-stringa, ma solamente verificare se una **parola completa** è presente in una stringa

s1	c	i	a	o	n	n	n	n	o	Ø	t	2	"	r
s2	o	g	g	i	n	n	n	c	'	e	'	Ø	4	

r	n	o	n	Ø	z	3
---	---	---	---	---	---	---

65

Definizioni (1/2)

- Lettera:** carattere ASCII facente parte dell'alfabeto maiuscolo ('A'...'Z') o minuscolo ('a'...'z')
- Parola:** insieme consecutivo di lettere, separato da altre parole mediante spazi, numeri o simboli di punteggiatura

66

Definizioni (2/2)

➤ **Inizio di parola:** lettera, prima della quale non vi è un'altra lettera

- Non vi è un altro carattere (inizio stringa)
- Vi è un altro carattere, ma non è una lettera

➤ **Fine di parola:** lettera, dopo la quale non vi è un'altra lettera

- Non vi è un altro carattere (fine stringa)
- Vi è un altro carattere, ma non è una lettera

67

Algoritmo di ricerca

➤ Per ogni possibile posizione pos di r all'interno di s

- Se il carattere in quella posizione, s[pos], è un inizio di parola

- Controlla se i caratteri di r, tra 0 e 1r-1, coincidono con i caratteri di s, tra pos e pos+1r-1
- Controlla se s[pos+1r-1] è una fine di parola
- Se entrambi i controlli sono ok, allora trovato=1

68

Algoritmo di ricerca

➤ Per ogni possibile posizione pos di r all'interno di s

- Se il carattere in quella posizione, s[pos], è un inizio di parola

- Controlla se i caratteri di r, tra 0 e 1r-1, coincidono con i caratteri di s, tra pos e pos+1r-1
- Controlla se s[pos+1r-1] non è una lettera
- Se entrambi i `if(pos == 0 || s[pos-1] non è una lettera)`

69

Algoritmo di ricerca

➤ Per ogni possibile posizione pos di r all'interno di s

- Se il carattere in quella posizione, s[pos], è un inizio di parola

- Controlla se i caratteri di r, tra 0 e 1r-1, coincidono con i caratteri di s, tra pos e pos+1r-1
- Controlla se s[pos+1r-1] non è una lettera
- Se entrambi i `if(pos == 0 || s[pos-1] non è una lettera)`

```
if( pos == 0 ||  
    !( (s[pos-1]>='a' && s[pos-1]<='z') ||  
        (s[pos-1]>='A' && s[pos-1]<='Z') )  
)
```

70

Algoritmo di ricerca

➤ Per ogni possibile posizione pos di r all'interno di s

- Se il carattere in quella posizione, s[pos], è un inizio di parola

- Controlla se i caratteri di r, tra 0 e 1r-1, coincidono con i caratteri di s, tra pos e pos+1r-1
- Controlla se s[pos+1r-1] è una fine di parola
- Se entrambi i controlli sono ok, allora trovato=1

71

Algoritmo di ricerca

➤ Per ogni possibile posizione pos di r all'interno di s

- Se il carattere in quella posizione, s[pos], è un inizio di parola

- Controlla se i caratteri di r, tra 0 e 1r-1, coincidono con i caratteri di s, tra pos e pos+1r-1
- Controlla se s[pos+1r-1] non è una lettera
- Se entrambi i controlli sono ok, allora trovato=1

72

Ricerca di una parola (1/2)

```
trovato = 0 ;  
for(pos=0; pos<=ls-1r; pos++)  
{  
    if( pos==0 ||  
        !( s[pos-1]>='a' &&  
            s[pos-1]<='z') ||  
        (s[pos-1]>='A' &&  
            s[pos-1]<='Z') ) )  
    {  
        diversi = 0 ;  
        for(i=0; i<lr; i++)  
            if(r[i]!=s[pos+i])  
                diversi = 1 ;  
    }  
}
```

parola.c

73

Ricerca di una parola (2/2)

```
if( diversi==0 &&  
    ( pos == ls-1r ||  
        !( s[pos+1r]>='a' &&  
            s[pos+1r]<='z') ||  
        (s[pos+1r]>='A' &&  
            s[pos+1r]<='Z') ) )  
{  
    trovato=1 ;  
}  
}
```

parola.c

74

La funzione strparola

- » Nella libreria standard C non esiste alcuna funzione che svolga automaticamente la ricerca di una parola intera!!!
- » Occorre identificare, ogni volta, se il compito da svolgere è riconducibile ad una o più funzioni di libreria
- » Eventualmente si combinano tra loro più funzioni di libreria diverse
- » In alcuni casi occorre però ricorrere all'analisi carattere per carattere

75

- ▶ Introduzione
- ▶ Lunghezza di stringhe
- ▶ Classificazione di caratteri
- ▶ Trasformazione di caratteri
- ▶ Copia e concatenazione
- ▶ Confronto di stringhe
- ▶ Ricerca in stringhe
- ▶ Conversione numero-stringa

2

Caratteri e stringhe

Funzioni di libreria

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>
```

```
#define MAXPAROLA 30
#define MAXSIGA 80
```

```
int main(int argc, char *argv[])
{
    int lung(MAXPAROLA); /* valore di controllo delle frequenze delle parole */
    char test[MAXSIGA]; /* stringa testo da analizzare */
    char sig(MAXSIGA);
    int i, lunghezza;
    FILE *fptr;
```

```
for(i=0; i<MAXPAROLA; i++)
    lung[i]=0;
for(i=0; i<MAXSIGA; i++)
    test[i]=0;
if((fptr=fopen(argv[1], "r"))==NULL)
    {
        printf("ERRORE, impossibile aprire il file %s", argv[1]);
        exit(1);
    }

```

Funzioni di libreria

Introduzione

4

```
(Argomenti)
```

Librerie sulle stringhe

- ▶ La libreria standard C dispone di molte funzioni predisposte per lavorare su caratteri e stringhe
- ▶ Tali funzioni si trovano prevalentemente in due librerie:
 - **<ctype.h>** funzioni operanti su caratteri
 - **<string.h>** funzioni operanti su stringhe
- ▶ Tutte le funzioni di libreria accettano e generano stringhe correttamente terminate



Suggerimenti

- ▶ Quando possibile, utilizzare sempre le funzioni di libreria
 - Sono più veloci
 - Sono maggiormente collaudate
- ▶ In ogni caso, ricordare che è sempre possibile effettuare le operazioni direttamente:
 - Sui caratteri, ricorrendo alla codifica ASCII
 - Sulle stringhe, ricorrendo alla rappresentazione come vettori di caratteri

5

```
(Argomenti)
```

Rappresentazione

Nome funzione	strlen
Libreria	#include <string.h>
Parametri in ingresso	s : stringa
Valore restituito	int : la lunghezza della stringa
Descrizione	Calcola la lunghezza della stringa s
Esempio	lun = strlen(s) ;

6

Convenzioni

- Assumiamo che nel seguito di questa lezione siano valide le seguenti definizioni

```
const int MAX = 20 ;
char s[MAX] ;
char s1[MAX] ;
char s2[MAX] ;
char r[MAX] ;
int lun ;
int n ;
char ch ;
float x ;
```

7

Funzioni di libreria

Lunghezza di stringhe

- Definite in `<string.h>`
- Determina la lunghezza di una stringa data

● `strlen`

9

strlen

Nome funzione	strlen
Libreria	<code>#include <string.h></code>
Parametri in ingresso	<code>s : stringa</code>
Valore restituito	<code>int : la lunghezza della stringa</code>
Descrizione	Calcola la lunghezza della stringa <code>s</code>
Esempio	<code>lun = strlen(s) ;</code>

10

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

#define MAXPAROLA 30
#define MAXIGRA 80

int main(int argc, char *argv[])
{
    int lungMAXPAROLA; /* valore di confronto
                        delle frequenze delle lunghezze delle parole */
    char parolaMAXIGRA[ ];
    int i, indice, lunghezza;
    FILE *f;

    if(argc > 1)
        lungMAXPAROLA = atoi(argv[1]);
    else
        lungMAXPAROLA = 10;

    f = fopen(argv[0], "r");
    if(f == NULL)
    {
        perror("ERRORE: impossibile aprire il file %s", argv[0]);
        exit(1);
    }

    printf("Lunghezza di MAXIGRA: %d\n", lungMAXIGRA);

    for(i=0; fgetchar(f); i++)
    {
        if(indice < lungMAXIGRA)
            parolaMAXIGRA[indice] = fgetc(f);
        else
            break;
    }
    parolaMAXIGRA[lungMAXIGRA] = '\0';

    printf("Parola inserita: %s\n", parolaMAXIGRA);
}
```

Funzioni di libreria

Classificazione di caratteri

- Definite in `<cctype.h>`
- Analizzano un singolo carattere, identificandone la tipologia
 - Lettera
 - Maiuscola
 - Minuscola
 - Cifra
 - Punteggiatura

- `isalpha`
- `isupper`
- `islower`
- `isdigit`
- `isalnum`
- `isxdigit`
- `ispunct`
- `isgraph`
- `isprint`
- `isspace`
- `iscntrl`

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

int main()
{
    char c;
    int i, lungMAXIGRA;
    FILE *f;

    lungMAXIGRA = 10;
    f = fopen("test.txt", "r");
    if(f == NULL)
    {
        perror("ERRORE: impossibile aprire il file test.txt");
        exit(1);
    }

    for(i=0; fgetchar(f); i++)
    {
        if(indice < lungMAXIGRA)
            parolaMAXIGRA[indice] = fgetc(f);
        else
            break;
    }
    parolaMAXIGRA[lungMAXIGRA] = '\0';

    printf("Parola inserita: %s\n", parolaMAXIGRA);
}
```

Classificazione di caratteri

12

isalpha

Nome funzione	isalpha
Libreria	#include <ctype.h>
Parametri in ingresso	ch : carattere
Valore restituito	Vero/falso
Descrizione	Ritorna "vero" se il carattere ch è una lettera maiuscola o minuscola (A...Z, a...z), "falso" altrimenti
Esempio	if(isalpha(ch)) { ... }

13

isupper

Nome funzione	isupper
Libreria	#include <ctype.h>
Parametri in ingresso	ch : carattere
Valore restituito	Vero/falso
Descrizione	Ritorna "vero" se il carattere ch è una lettera maiuscola (A...Z), "falso" altrimenti
Esempio	if(isupper(ch)) { ... }

14

islower

Nome funzione	islower
Libreria	#include <ctype.h>
Parametri in ingresso	ch : carattere
Valore restituito	Vero/falso
Descrizione	Ritorna "vero" se il carattere ch è una lettera minuscola (a...z), "falso" altrimenti
Esempio	if(islower(ch)) { ... }

15

isdigit

Nome funzione	isdigit
Libreria	#include <ctype.h>
Parametri in ingresso	ch : carattere
Valore restituito	Vero/falso
Descrizione	Ritorna "vero" se il carattere ch è una cifra numerica (0...9), "falso" altrimenti
Esempio	if(isdigit(ch)) { ... }

16

isalnum

Nome funzione	isalnum
Libreria	#include <ctype.h>
Parametri in ingresso	ch : carattere
Valore restituito	Vero/falso
Descrizione	Ritorna "vero" se il carattere ch è una lettera oppure una cifra numerica, "falso" altrimenti. Equivalenti a <code>isalpha(ch) isdigit(ch)</code>
Esempio	if(isalnum(ch)) { ... }

17

isxdigit

Nome funzione	isxdigit
Libreria	#include <ctype.h>
Parametri in ingresso	ch : carattere
Valore restituito	Vero/falso
Descrizione	Ritorna "vero" se il carattere ch è una cifra numerica oppure una lettera valida in base 16 (a...f, A...F), "falso" altrimenti.
Esempio	if(isxdigit(ch)) { ... }

18

ispunct

Nome funzione	ispunct
Libreria	#include <ctype.h>
Parametri in ingresso	ch : carattere
Valore restituito	Vero/falso
Descrizione	Ritorna "vero" se il carattere ch è un simbolo di punteggiatura (!"#\$%&'()*+,-./;:<=>?@[\]^_`{ }~), "falso" altrimenti.
Esempio	if(ispunct(ch)) { ... }

19

isgraph

Nome funzione	isgraph
Libreria	#include <ctype.h>
Parametri in ingresso	ch : carattere
Valore restituito	Vero/falso
Descrizione	Ritorna "vero" se il carattere ch è un qualsiasi simbolo visibile (lettera, cifra, punteggiatura), "falso" altrimenti.
Esempio	if(isgraph(ch)) { ... }

20

isprint

Nome funzione	isprint
Libreria	#include <ctype.h>
Parametri in ingresso	ch : carattere
Valore restituito	Vero/falso
Descrizione	Ritorna "vero" se il carattere ch è un qualsiasi simbolo visibile oppure lo spazio, "falso" altrimenti.
Esempio	if(isprint(ch)) { ... }

21

isspace

Nome funzione	isspace
Libreria	#include <ctype.h>
Parametri in ingresso	ch : carattere
Valore restituito	Vero/falso
Descrizione	Ritorna "vero" se il carattere ch è invisibile (spazio, tab, a capo), "falso" altrimenti.
Esempio	if(isspace(ch)) { ... }

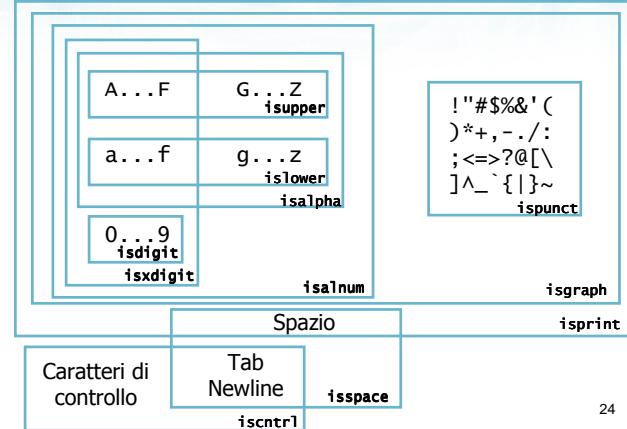
22

iscntrl

Nome funzione	iscntrl
Libreria	#include <ctype.h>
Parametri in ingresso	ch : carattere
Valore restituito	Vero/falso
Descrizione	Ritorna "vero" se ch è un carattere di controllo (ASCII 0...31, 127), "falso" altrimenti.
Esempio	if(iscntrl(ch)) { ... }

23

Vista d'insieme



24

```

#include <stdio.h>
#include <string.h>
#include <ctype.h>

#define MAXPAROLA 30
#define MAXIGRA 80

int main(int argc, char *argv[])
{
    int freqMAXPAROLA; /* valore di confronto delle frequenze delle parole */
    char parolaAXINGAL;
    int i, indice, lunghezza;
    FILE *fp;

    if(argc != 2)
    {
        fprintf(stderr, "ERRORE: non è possibile aprire il file %s", argv[1]);
        exit(1);
    }

    fp = fopen(argv[1], "r");
    if(fp == NULL)
    {
        perror("ERRORE: impossibile aprire il file %s", argv[1]);
        exit(1);
    }

    printf("Aperto file: %s\n", MAXIGRA, i) == NULL;
}

```

Funzioni di libreria

Trasformazione di caratteri

Trasformazione di caratteri

- Definite in `<ctype.h>`
- Convertono tra lettere maiuscole e lettere minuscole
- toupper
- tolower

26

toupper

Nome funzione	toupper
Libreria	#include <ctype.h>
Parametri in ingresso	ch : carattere
Valore restituito	char : carattere maiuscolo
Descrizione	Se ch è una lettera minuscola, ritorna l'equivalente carattere maiuscolo, se no ritorna ch stesso
Esempio	for (i=0; s[i]!=0; i++) s[i] = toupper(s[i]) ;

27

tolower

Nome funzione	tolower
Libreria	#include <ctype.h>
Parametri in ingresso	ch : carattere
Valore restituito	char : carattere maiuscolo
Descrizione	Se ch è una lettera minuscola, ritorna l'equivalente carattere maiuscolo, se no ritorna ch stesso
Esempio	for (i=0; s[i]!=0; i++) s[i] = tolower(s[i]) ;

28

```

#include <stdio.h>
#include <string.h>
#include <ctype.h>

#define MAXPAROLA 30
#define MAXIGRA 80

int main(int argc, char *argv[])
{
    int freqMAXPAROLA; /* valore di confronto delle frequenze delle parole */
    char parolaAXINGAL;
    int i, indice, lunghezza;
    FILE *fp;

    if(argc != 2)
    {
        fprintf(stderr, "ERRORE: non è possibile aprire il file %s", argv[1]);
        exit(1);
    }

    fp = fopen(argv[1], "r");
    if(fp == NULL)
    {
        perror("ERRORE: impossibile aprire il file %s", argv[1]);
        exit(1);
    }

    printf("Aperto file: %s\n", MAXIGRA, i) == NULL;
}

```

Funzioni di libreria

Copia e concatenazione

Copia e concatenazione

- Definite in `<string.h>`
- Trasferiscono il contenuto di una stringa in un'altra
 - Sostituendolo
 - Accodandolo
- strcpy
- strncpy
- strcat
- strncat

30

strcpy

Nome funzione	strcpy
Libreria	#include <string.h>
Parametri in ingresso	dst : stringa src : stringa
Valore restituito	nessuno utile
Descrizione	Copia il contenuto della stringa src all'interno della stringa dst (che deve avere lunghezza sufficiente).
Esempio	strcpy(s1, s2) ; strcpy(s, "") ; strcpy(s1, "ciao") ;

31

strncpy

Nome funzione	strncpy
Libreria	#include <string.h>
Parametri in ingresso	dst : stringa src : stringa n : numero max caratteri
Valore restituito	nessuno utile
Descrizione	Copia il contenuto della stringa src (massimo n caratteri) all'interno della stringa dst.
Esempio	strncpy(s1, s2, 20) ; strncpy(s1, s2, MAX) ;

32

strcat

Nome funzione	strcat
Libreria	#include <string.h>
Parametri in ingresso	dst : stringa src : stringa
Valore restituito	nessuno utile
Descrizione	Accoda il contenuto della stringa src alla fine della stringa dst (che deve avere lunghezza sufficiente).
Esempio	strcat(s1, s2) ; strcat(s1, " ") ;

33

strncat

Nome funzione	strncat
Libreria	#include <string.h>
Parametri in ingresso	dst : stringa src : stringa n : numero max caratteri
Valore restituito	nessuno utile
Descrizione	Accoda il contenuto della stringa src (massimo n caratteri) alla fine della stringa dst.
Esempio	strncat(s1, s2) ;

34

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

#define MAXPAROLA 30
#define MAXIGRA 80

int main(int argc, char *argv[])
{
    int lenMAXPAROLA; /* vettore di dimensione massima delle parole */
    char strMAXPAROLA[MAXPAROLA]; /* stringa MAXPAROLA */
    int i, indice, lunghezza;
    i = 1;

    for(i=0; i<MAXPAROLA; i++)
        lunghezza=0;
    if(argc>1)
    {
        strcpy(strMAXPAROLA, argv[1]);
        lenMAXPAROLA = strlen(strMAXPAROLA);
        for(i=0; i<lenMAXPAROLA; i++)
            if(strMAXPAROLA[i] == ' ')
                lunghezza++;
        if(lunghezza>0)
            strMAXPAROLA[lunghezza] = '\0';
    }
    else
        strMAXPAROLA[0] = '\0';
    printf("Parola inserita: %s\n", strMAXPAROLA);
}

int main()
{
    char strMAXIGRA[MAXIGRA];
    int i, indice, lunghezza;
    i = 1;
    for(i=0; i<MAXIGRA; i++)
        lunghezza=0;
    if(argc>1)
    {
        strcpy(strMAXIGRA, argv[1]);
        lenMAXIGRA = strlen(strMAXIGRA);
        for(i=0; i<lenMAXIGRA; i++)
            if(strMAXIGRA[i] == ' ')
                lunghezza++;
        if(lunghezza>0)
            strMAXIGRA[lunghezza] = '\0';
    }
    else
        strMAXIGRA[0] = '\0';
    printf("Parola inserita: %s\n", strMAXIGRA);
}
```

Funzioni di libreria

Confronto di stringhe

Confronto di stringhe

- Definite in <string.h>
- Confrontano due stringhe sulla base dell'ordine lessicografico imposto dalla tabella dei codici ASCII
- strcmp
- strncmp

36

strcmp

Nome funzione	strcmp
Libreria	#include <string.h>
Parametri in ingresso	s1 : stringa s2 : stringa
Valore restituito	int : risultato confronto
Descrizione	Risultato <0 se s1 precede s2 Risultato ==0 se s1 è uguale a s2 Risultato >0 se s1 segue s2
Esempio	if (strcmp(s, r)==0) {...} while (strcmp(r, "fine")!=0) {...}

3

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

#define MAXPAROLA 30
#define MAXBGA 80

int main(int argc, char *argv[])
{
    int lung(MAXPAROLA); // -> vettore di caratteri
    char s[MAXPAROLA]; // -> stringa da inserire nel vettore
    char s2[MAXBGA]; // -> stringa da inserire nella parola
    int i, n, lungBGA; // -> variabili intere
    i = 0;
    n = 0;
    lungBGA = 0;

    for(i=0; i<MAXPAROLA; i++)
        s[i] = '0';

    if(argc < 2)
    {
        printf("ERRORE: non hai specificato il nome del file da aprire\n");
        exit(1);
    }

    if(fopen(argv[1], "r") == NULL)
    {
        printf("ERRORE: impossibile aprire il file %s", argv[1]);
        exit(1);
    }

    while(fgets(s, MAXPAROLA, f) != NULL)
    {
        if(sscanf(s, "%s", s2) == 1)
        {
            if(strlen(s2) > lungBGA)
                lungBGA = strlen(s2);
        }
    }

    printf("La parola con la maggiore lunghezza e' %s\n", s2);
    printf("La sua lunghezza e' %d\n", lungBGA);
}
```

Funzioni di libreria

Ricerca in stringhe

Ricerca

- Definite in `<string.h>`
 - Ricercano all'interno di una stringa data
 - Se compare un carattere
 - Se compare una sotto-stringa
 - Se compare una sequenza qualsiasi composta di caratteri dati

40

strchr

Nome funzione	strchr
Libreria	#include <string.h>
Parametri in ingresso	s : stringa ch : carattere
Valore restituito	==NULL oppure !=NULL
Descrizione	Risultato !=NULL se il carattere ch compare nella stringa. Risultato ==NULL se non compare.
Esempio	if(strchr(s, '.'))!=NULL ... if(strchr(s, ch)==NULL) ...

4

strstr

Nome funzione	strstr
Libreria	#include <string.h>
Parametri in ingresso	s : stringa r : stringa
Valore restituito	==NULL oppure !=NULL
Descrizione	Risultato !=NULL se la sotto-stringa r compare nella stringa s. Risultato ==NULL se non compare.
Esempio	if(strstr(s, "xy")!=NULL)... if(strstr(s, s1)==NULL)...

42

strspr

Nome funzione	strspn
Libreria	#include <string.h>
Parametri in ingresso	s : stringa r : stringa
Valore restituito	int : lunghezza sequenza iniziale
Descrizione	Calcola la lunghezza della parte iniziale di s che è composta esclusivamente dei caratteri presenti in r (in qualsiasi ordine).
Esempio	lun = strspn(s, " ") ; lun = strspn(s, " :,.;") ;

4

strcspn

Nome funzione	strcspn
Libreria	#include <string.h>
Parametri in ingresso	s : stringa r : stringa
Valore restituito	int : lunghezza sequenza iniziale
Descrizione	Calcola la lunghezza della parte iniziale di s che è composta esclusivamente da caratteri non presenti in r (in qualsiasi ordine).
Esempio	lun = strcspn(s, " ") ; lun = strcspn(s, " :,.;") ;

44

```
#include <iostream.h>
#include <string.h>
#include <math.h>
#include <conio.h>

#define MAXPAROLA 30
#define MAXIGA 80

int main()
{
    char *argv[] = {
        "inf", MAXPAROLA, "/f", "vertice di confine",
        "inf", MAXPAROLA, "/t", "vertice di tangente",
        "inf", MAXIGA, "/l", "lunghezza",
        NULL
    };

    for (int i = 0; i < 4; i++)
        cout << argv[i] << endl;

    if (argc != 5)
    {
        cout << "ERRORE: non sono comparsi tutti e 5 argomenti." << endl;
        exit(1);
    }

    if (strcmp(argv[1], "inf") != 0)
    {
        cout << "Argomento 1 deve essere 'inf'." << endl;
        exit(1);
    }

    if (strcmp(argv[2], "/f") != 0)
    {
        cout << "Argomento 2 deve essere '/f'." << endl;
        exit(1);
    }

    if (strcmp(argv[3], "/t") != 0)
    {
        cout << "Argomento 3 deve essere '/t'." << endl;
        exit(1);
    }

    if (strcmp(argv[4], "/l") != 0)
    {
        cout << "Argomento 4 deve essere '/l'." << endl;
        exit(1);
    }

    if (strcmp(argv[5], NULL) != 0)
    {
        cout << "Argomento 5 deve essere nullo." << endl;
        exit(1);
    }
}
```

Funzioni di libreria

Conversione numero-stringa

atoi

ato-

Nome funzione	atoi
Libreria	#include <stdlib.h>
Parametri in ingresso	s : stringa
Valore restituito	int : valore estratto
Descrizione	Analizza la stringa s ed estraе il valore intero in essa contenuto (a partire dai primi caratteri).
Esempio	n = atoi(s) ; n = atoi("232abc") ;

6

atof

Nome funzione	atof
Libreria	#include <stdlib.h>
Parametri in ingresso	s : stringa
Valore restituito	double/float : valore estratto
Descrizione	Analizza la stringa s ed estrae il valore reale in essa contenuto (a partire dai primi caratteri).
Esempio	x = atof(s) ; x = atof("2.32abc") ;

49

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

#define MAXPAROLA 30
#define MAXSIGA 80
```

Caratteri e stringhe

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

#define MAXPAROLA 30
#define MAXSIGA 80

int main(int argc, char *argv[])
{
    int lungMAXPAROLA; /* valore di controllo delle frequenze delle parole */
    char lungMAXSIGA; /* valore di controllo delle lunghezze delle parole */
    int i, indice, lungParola;
    FILE *fp;

    fp=fopen(argv[1], "r");
    lungMAXSIGA=0;
    lungMAXPAROLA=0;
    for(i=0; i<MAXSIGA; i++)
        lungMAXSIGA+=lungMAXSIGA;
    for(i=0; i<MAXPAROLA; i++)
        lungMAXPAROLA+=lungMAXPAROLA;
    if(argc > 1)
    {
        fp=fopen(argv[1], "r");
        if(fp==NULL)
            printf("ERRORE, non esiste un file chiamato %s\n", argv[1]);
        else
        {
            i=0;
            while(fgets(sigla, MAXSIGA, fp) != NULL)
            {
                lungParola=0;
                for(indice=0; sigla[indice] != '\n'; indice++)
                    lungParola++;
                if(lungParola > lungMAXSIGA)
                    lungMAXSIGA=lungParola;
                if(lungParola > lungMAXPAROLA)
                    lungMAXPAROLA=lungParola;
            }
        }
    }
    printf("lungMAXSIGA = %d\n", lungMAXSIGA);
    printf("lungMAXPAROLA = %d\n", lungMAXPAROLA);
    fclose(fp);
}
```

Esercizi proposti

Esercizi proposti

Esercizi proposti

- Esercizio “Parola palindroma”
- Esercizio “Iniziali maiuscole”
- Esercizio “Alfabetto farfallino”

2

Esercizio “Parola palindroma”

Esercizio “Parola palindroma”

- Sia data una parola inserita da tastiera.
- Si consideri che la parola può contenere sia caratteri maiuscoli che caratteri minuscoli, e complessivamente al massimo 30 caratteri
- Il programma deve svolgere le seguenti operazioni:
 - Visualizzare la parola inserita
 - Aggiornare la parola in modo che tutti i caratteri siano minuscoli, e visualizzarla
 - Verificare se la parola è palindroma

4

Palindromia

- Una parola è detta **palindroma** se può essere letta indifferentemente da sinistra verso destra e da destra verso sinistra

- Esempi:

o	t	t	o
---	---	---	---

m	a	d	a	m
---	---	---	---	---

5

Analisi

- Acquisisci parola
- Stampa parola
- Converti in minuscolo
- Stampa minuscolo
- Verifica se è palindroma
- Stampa se è palindroma

6

Analisi

- » Acquisisci parola
- » Stampa parola
- » Converti in minuscolo
- » Stampa minuscule
- » Verifica se è palindroma
- » Stampa se è palindroma

```
const int MAX = 30 ;  
char parola[MAX+1] ;  
printf("Inserisci parola: ") ;  
scanf("%s", parola) ;
```

7

8

Analisi

- » Acquisisci parola
- » Stampa parola
- » Converti in minuscolo
- » Stampa minuscule
- » Verifica se è palindroma
- » Stampa se è palindroma

```
char minusc[MAX+1] ;  
int i ;  
  
strcpy(minusc, parola) ;  
  
for(i=0; minusc[i]!=0; i++)  
{  
    minusc[i] = tolower( minusc[i] ) ;  
}
```

- » Acquisisci parola
- » Stampa parola
- » Converti in minuscolo
- » Stampa minuscule
- » Verifica se è palindroma
- » Stampa se è palindroma

10

Analisi

- » Acquisisci parola
- » Stampa parola
- » Converti in minuscolo
- » Stampa minuscule
- » Verifica se è palindroma
- » Stampa se è palindroma

```
int i, j, palin, lun ;  
i = 0 ;  
j = strlen( minusc ) - 1 ;  
palin = 1 ;  
  
while( i<j && palin==1 )  
{  
    if(minusc[i] != minusc[j])  
        palin = 0 ;  
    i++ ; j-- ;  
}
```

- » Acquisisci parola
- » Stampa se è palindroma
- » Converti in minuscolo
- » Stampa minuscule
- » Verifica se è palindroma
- » Stampa se è palindroma

12

Soluzione

```
Quincy 2005
Parola palindroma
Inserisci una parola: Otto
La parola inserita e': Otto
La parola in minuscolo e': otto
La parola e' palindroma
Press Enter to return to Quincy...
```

palindroma.c

13

Esercizi proposti

Esercizio "Iniziali maiuscole"

Esercizio "Iniziali maiuscole" (1/2)

- Scrivere un programma che legga una frase introdotta da tastiera
 - La frase è terminata dall'introduzione del carattere di invio
 - La frase contiene sia caratteri maiuscoli che caratteri minuscoli, e complessivamente al più 100 caratteri

15

Esercizio "Iniziali maiuscole" (2/2)

- Il programma deve svolgere le seguenti operazioni:
 - Visualizzare la frase inserita
 - Costruire una nuova frase in cui il primo carattere di ciascuna parola nella frase di partenza è stato reso maiuscolo. Tutti gli altri caratteri devono essere resi minuscoli
 - Visualizzare la nuova frase

16

Esempio

cHe bElla gIORnATA

Che Bella Giornata

17

Analisi

- La frase inserita può contenere degli spazi: occorrerà usare `gets` e non `scanf`
- Ogni lettera iniziale di parola va convertita in maiuscolo
- Ogni lettera non iniziale di parola va convertita in minuscolo
- Ogni altro carattere va lasciato immutato

18

Conversione delle lettere

```
for(i=0; frase[i]!=0; i++)
{
    if( isalpha(frase[i]) &&
        ( i==0 || !isalpha(frase[i-1]) ) )
    {
        frase[i] = toupper( frase[i] ) ;
    }
    else
    {
        frase[i] = tolower( frase[i] ) ;
    }
}
```

iniziali.c



iniziali.c

19

```
#include <stdio.h>
#include <ctype.h>
#include <string.h>

#define MAXPAROLA 30
#define MAXSIGA 60

int main(int argc, char *argv[])
{
    int lunghezzaParola; /* valore di controllo della frequenza delle lunghezze delle parole */
    int lunghezzaSiga; /* lunghezza della parola */
    int i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z;
    char frase[MAXPAROLA]; /* frase inserita dall'utente */
    char siga[MAXSIGA]; /* sigla inserita dall'utente */
    char output[MAXSIGA]; /* sigla ottenuta dalla conversione */

    if(argc != 2)
    {
        fprintf(stderr, "ERRORE: inserire un parametro sulla linea di comando che sia l'input");
        exit(1);
    }

    if((frase = fopen(argv[1], "r")) == NULL)
    {
        fprintf(stderr, "ERRORE: impossibile aprire il file %s", argv[1]);
        exit(1);
    }

    if((siga = fopen("sighe.txt", "r")) == NULL)
    {
        fprintf(stderr, "ERRORE: impossibile aprire il file sighe.txt");
        exit(1);
    }

    if((output = fopen("output.txt", "w")) == NULL)
    {
        fprintf(stderr, "ERRORE: impossibile aprire il file output");
        exit(1);
    }
```

Esercizi proposti

Esercizio “Alfabeto farfallino”

Esercizio “Alfabeto farfallino” (1/2)

- » Scrivere un programma che legga una frase introdotta da tastiera
- La frase è terminata dall'introduzione del carattere di invio
 - La frase contiene sia caratteri maiuscoli che caratteri minuscoli, e complessivamente al più 100 caratteri

21

Esercizio “Alfabeto farfallino” (2/2)

- » Il programma deve svolgere le seguenti operazioni:
- Visualizzare la frase inserita
 - Costruire una nuova frase nel cosiddetto “alfabeto farfallino”
 - Visualizzare la nuova frase

22

L’alfabeto farfallino

- » La traduzione nell’alfabeto farfallino di una parola segue le seguenti regole:
- Tutte le consonanti sono tradotte in modo identico
 - Ogni vocale è tradotta uguale a se stessa, seguita da altri due caratteri:
 - la lettera ‘f’ (se la vocale è minuscola) o la lettera ‘F’ (se la vocale è maiuscola)
 - una copia della vocale stessa
- » Esempi:
- a → afa
 - con → cofon

23

Vacanze di NATALE

vafacafanzefe difi NAFATAFALEFE

24

Approccio risolutivo

- » Copiamo la stringa frase in una nuova stringa farfa
- » lun=0
- » Per ogni carattere frase[i]
 - Se non è una vocale, va accodato a farfa[lun]
 - Se è una vocale, occorre accodare a farfa[lun] i 3 caratteri: frase[i], poi 'f' o 'F', poi ancora frase[i]
 - Incrementare lun (di 1 oppure 3)
- » Infine aggiungere a farfa[lun] il terminatore nullo

25

Conversione alfabeto (1/2)

```
lun = 0 ;  
for(i=0; frase[i]!=0; i++)  
{  if( isalpha(frase[i]) )  
    { /* lettera alfabetica */  
      ch = tolower(frase[i]) ;  
      if( ch=='a' || ch=='e' || ch=='i'  
          || ch=='o' || ch=='u' )  
      { /* vocale: trasforma */  
        1 farfa[lun] = frase[i] ;  
        if(isupper(frase[i]))  
          2 farfa[lun+1] = 'F' ;  
        else farfa[lun+1] = 'f' ;  
        3 farfa[lun+2] = frase[i] ;  
  
        lun = lun + 3 ;  
      }  
    }  
  }  
}
```

26

Conversione alfabeto (2/2)

```
else  
{  
  /* consonante: copia */  
  farfa[lun] = frase[i] ;  
  lun++ ;  
}  
}  
else  
{  
  /* altro carattere: copia */  
  farfa[lun] = frase[i] ;  
  lun++ ;  
}  
}  
farfa[lun] = 0 ; /* terminatore */
```

27

```
#include <stdio.h>
#include <ctype.h>
#define MAXPAROLA 30
#define MAXIGRA 80

int main(int argc, char *argv[])
{
    int freq[MAXPAROLA]; /* vettore di contatori delle frequenze delle lunghezze delle parole */
    int i, max, lunghezza; /* i: indice, lunghezza: */

    if(argc > 1)
    {
        for(i = 0; i < argc; i++)
        {
            if(strcmp(argv[i], "MAXIGRA") == 0)
                freq[0] = 1;
            else if(strcmp(argv[i], "MAXPAROLA") == 0)
                freq[1] = 1;
            else if(strcmp(argv[i], "lunghezza") == 0)
                freq[2] = 1;
            else if(strcmp(argv[i], "max") == 0)
                freq[3] = 1;
            else if(strcmp(argv[i], "parola") == 0)
                freq[4] = 1;
            else if(strcmp(argv[i], "stringa") == 0)
                freq[5] = 1;
            else if(strcmp(argv[i], "vettori") == 0)
                freq[6] = 1;
            else if(strcmp(argv[i], "errore") == 0)
                freq[7] = 1;
            else if(strcmp(argv[i], "esempio") == 0)
                freq[8] = 1;
            else if(strcmp(argv[i], "classificazione") == 0)
                freq[9] = 1;
            else if(strcmp(argv[i], "ricerca") == 0)
                freq[10] = 1;
            else if(strcmp(argv[i], "concatenazione") == 0)
                freq[11] = 1;
            else if(strcmp(argv[i], "conversione") == 0)
                freq[12] = 1;
        }
    }
    else
        printf("Nessun argomento inserito\n");
}
```

Caratteri e stringhe

Sommario

Argomenti trattati

- Caratteri e stringhe
- Il tipo char
- Vettori di char
- Stringhe: parole, frasi
- Operazioni fondamentali sulle stringhe
 - Classificazione
 - Ricerca
 - Copia e concatenazione
 - Conversione

2

```
#include <string.h>
#include <ctype.h>
```

Tecniche di programmazione

- Terminatore nullo
- Librerie <string.h> e <ctype.h>
- Manipolazione di stringhe come vettori di caratteri
- Manipolazione di stringhe attraverso le funzioni di libreria
- Identificazione di parole all'interno di frasi

3

```
#include <string.h>
#include <ctype.h>
```

Errore frequente

- L'input/output nelle stringhe è spesso problematico
- Utilizzando la funzione scanf, in presenza di spazi interni alla stringa rimarranno dei caratteri "non letti", che daranno fastidio alle successive scanf
- Quando possibile, ricorrere alla funzione gets per la lettura di una stringa

5

```
#include <string.h>
#include <ctype.h>
```

Stringhe e vettori

- Molte operazioni sulle stringhe sono ricondotte ad analoghe operazioni sui vettori
- Molti problemi "astratti" su vettori numerici assumono forma più "concreta" nel caso delle stringhe
- I cicli sono solitamente governati dal controllo del terminatore di fine stringa

4

```
#include <string.h>
#include <ctype.h>
```

Errore frequente

- Le stringhe hanno lunghezza variabile; i vettori che le contengono hanno lunghezza fissa
- È possibile, con una chiamata a strcpy o strcat, scrivere oltre la dimensione del vettore
 - Grave errore di programmazione, che può portare alla corruzione di dati in altre variabili
- Abituarsi a verificare la lunghezza prima di copiare le stringhe

```
if(strlen(a) + strlen(b) + 1 <= MAX)
    strcat(a,b) ;
else
    ERRORE!!!
```

6

Materiale aggiuntivo

» Sul CD-ROM

- Testi e soluzioni degli esercizi trattati nei lucidi
- Scheda sintetica
- Esercizi risolti
- Esercizi proposti

» Esercizi proposti da altri libri di testo

```
#include <stdio.h>
#include <ctype.h>
#define MAXFAROLA 30
#define MAXCIGA 60
```

```
int main(int argc, char *argv[])
{
    int freq[MAXFAROLA]; /* vettore di contenuto delle frequenze delle parole */
    int i, maxf, length; /* i = indice, length = lunghezza */

    if(argc != 2)
    {
        fprintf(stderr, "ERRORE: non è possibile aprire il file %s", argv[1]);
        exit(1);
    }
    if((length = fopen(argv[1], "r")) == NULL)
```

Programmazione in C

Unità Matrici - Vettori di stringhe

Matrici – Vettori di stringhe

- » Matrici
- » Definizione di matrici in C
- » Operazioni elementari sulle matrici
- » Vettori di stringhe
- » Esercizi proposti
- » Sommario

2

Riferimenti al materiale

» Testi

- Kernighan & Ritchie: capitoli 1 e 5
- Cabodi, Quer, Sonza Reorda: capitolo 5
- Dietel & Dietel: capitolo 6

» Dispense

- Scheda: "Matrici e Vettori di stringhe in C"

3

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

#define MAXPAROLA 30
#define MAXIGA 80

int main(int argc, char *argv[])
{
    int lung(MAXPAROLA); /* vettore di contenuto delle frequenze delle parole */
    char lung(MAXIGA); /* stringa che contiene la parola */
    int i, indice, lunghezza;
    FILE *fptr;

    if(argc != 2)
    {
        printf("ERRORE: non è possibile aprire il file %s", argv[1]);
        exit(1);
    }

    fptr = fopen(argv[1], "r");
    if(fptr == NULL)
    {
        printf("ERRORE: impossibile aprire il file %s", argv[1]);
        exit(1);
    }

    while(fgets(lung, MAXIGA, fptr) != NULL)
    {
        lunghezza = strlen(lung);
        for(i=0; i<lunghezza; i++)
        {
            if(isalpha(lung[i]))
            {
                lung[i] = toupper(lung[i]);
            }
        }
        if(strcmp(lung, MAXPAROLA) == 0)
        {
            indice++;
            lung[MAXPAROLA] = '\0';
            lung[0] = '\0';
            lung[1] = '\0';
            lung[2] = '\0';
            lung[3] = '\0';
            lung[4] = '\0';
            lung[5] = '\0';
            lung[6] = '\0';
            lung[7] = '\0';
            lung[8] = '\0';
            lung[9] = '\0';
            lung[10] = '\0';
            lung[11] = '\0';
            lung[12] = '\0';
            lung[13] = '\0';
            lung[14] = '\0';
            lung[15] = '\0';
            lung[16] = '\0';
            lung[17] = '\0';
            lung[18] = '\0';
            lung[19] = '\0';
            lung[20] = '\0';
            lung[21] = '\0';
            lung[22] = '\0';
            lung[23] = '\0';
            lung[24] = '\0';
            lung[25] = '\0';
            lung[26] = '\0';
            lung[27] = '\0';
            lung[28] = '\0';
            lung[29] = '\0';
        }
        else
        {
            lung[0] = lung[1];
            lung[1] = lung[2];
            lung[2] = lung[3];
            lung[3] = lung[4];
            lung[4] = lung[5];
            lung[5] = lung[6];
            lung[6] = lung[7];
            lung[7] = lung[8];
            lung[8] = lung[9];
            lung[9] = lung[10];
            lung[10] = lung[11];
            lung[11] = lung[12];
            lung[12] = lung[13];
            lung[13] = lung[14];
            lung[14] = lung[15];
            lung[15] = lung[16];
            lung[16] = lung[17];
            lung[17] = lung[18];
            lung[18] = lung[19];
            lung[19] = lung[20];
            lung[20] = lung[21];
            lung[21] = lung[22];
            lung[22] = lung[23];
            lung[23] = lung[24];
            lung[24] = lung[25];
            lung[25] = lung[26];
            lung[26] = lung[27];
            lung[27] = lung[28];
            lung[28] = lung[29];
        }
    }
    printf("%d", indice);
}
```

Matrici – Vettori di stringhe

Matrici

5

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

#define MAXPAROLA 30
#define MAXIGA 80

int main(int argc, char *argv[])
{
    int lung(MAXPAROLA); /* vettore di contenuto delle frequenze delle parole */
    char lung(MAXIGA); /* stringa che contiene la parola */
    int i, indice, lunghezza;
    FILE *fptr;

    if(argc != 2)
    {
        printf("ERRORE: non è possibile aprire il file %s", argv[1]);
        exit(1);
    }

    fptr = fopen(argv[1], "r");
    if(fptr == NULL)
    {
        printf("ERRORE: impossibile aprire il file %s", argv[1]);
        exit(1);
    }

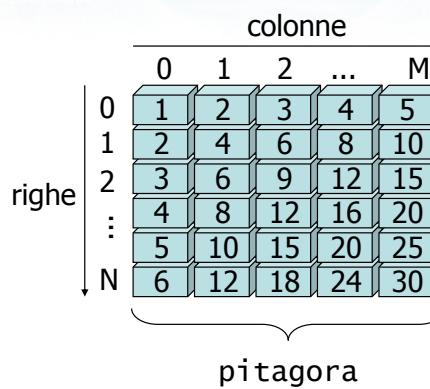
    while(fgets(lung, MAXIGA, fptr) != NULL)
    {
        lunghezza = strlen(lung);
        for(i=0; i<lunghezza; i++)
        {
            if(isalpha(lung[i]))
            {
                lung[i] = toupper(lung[i]);
            }
        }
        if(strcmp(lung, MAXPAROLA) == 0)
        {
            indice++;
            lung[MAXPAROLA] = '\0';
            lung[0] = '\0';
            lung[1] = '\0';
            lung[2] = '\0';
            lung[3] = '\0';
            lung[4] = '\0';
            lung[5] = '\0';
            lung[6] = '\0';
            lung[7] = '\0';
            lung[8] = '\0';
            lung[9] = '\0';
            lung[10] = '\0';
            lung[11] = '\0';
            lung[12] = '\0';
            lung[13] = '\0';
            lung[14] = '\0';
            lung[15] = '\0';
            lung[16] = '\0';
            lung[17] = '\0';
            lung[18] = '\0';
            lung[19] = '\0';
            lung[20] = '\0';
            lung[21] = '\0';
            lung[22] = '\0';
            lung[23] = '\0';
            lung[24] = '\0';
            lung[25] = '\0';
            lung[26] = '\0';
            lung[27] = '\0';
            lung[28] = '\0';
            lung[29] = '\0';
        }
        else
        {
            lung[0] = lung[1];
            lung[1] = lung[2];
            lung[2] = lung[3];
            lung[3] = lung[4];
            lung[4] = lung[5];
            lung[5] = lung[6];
            lung[6] = lung[7];
            lung[7] = lung[8];
            lung[8] = lung[9];
            lung[9] = lung[10];
            lung[10] = lung[11];
            lung[11] = lung[12];
            lung[12] = lung[13];
            lung[13] = lung[14];
            lung[14] = lung[15];
            lung[15] = lung[16];
            lung[16] = lung[17];
            lung[17] = lung[18];
            lung[18] = lung[19];
            lung[19] = lung[20];
            lung[20] = lung[21];
            lung[21] = lung[22];
            lung[22] = lung[23];
            lung[23] = lung[24];
            lung[24] = lung[25];
            lung[25] = lung[26];
            lung[26] = lung[27];
            lung[27] = lung[28];
            lung[28] = lung[29];
        }
    }
    printf("%d", indice);
}
```

Matrici

Matrici bidimensionali

7

Matrice bidimensionale



8

```
(fargc 14 3)
Spiegazione: "TUTTO" serve un particolare solo il numero del file (%s)
```

```
(fargc 14 3)
Spiegazione: "TUTTO" serve un particolare solo il numero del file (%s)
```

- Matrici bidimensionali
- Matrici come vettori di vettori
- Matrici pluridimensionali

```
(fargc 14 3)
Spiegazione: "TUTTO" serve un particolare solo il numero del file (%s)
```

Il concetto di matrice

- La **matrice (array)** è un'estensione logica del concetto di vettore
 - Vettore = Sequenza uni-dimensionale di valori
 - Tutti dello stesso tipo
 - Identificati da un indice intero
 - Dimensione fissa
 - Matrice = Schiera bi- (o n-) dimensionale di valori
 - Tutti dello stesso tipo
 - Identificati da 2 (o n) indici interi
 - Dimensioni fisse

```
(fargc 14 3)
Spiegazione: "TUTTO" serve un particolare solo il numero del file (%s)
```

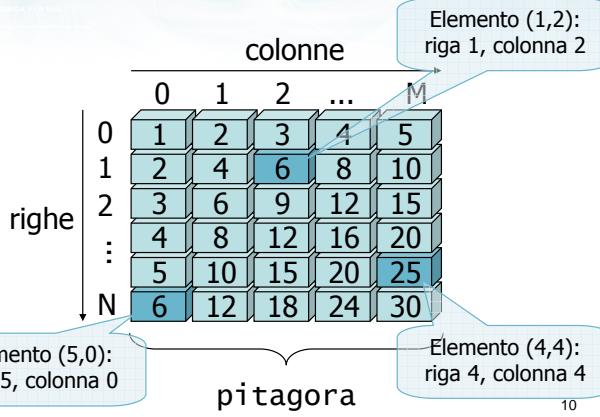
```
(fargc 14 3)
Spiegazione: "TUTTO" serve un particolare solo il numero del file (%s)
```

Caratteristiche

- Una matrice bi-dimensionale è caratterizzata da
 - Nome : pitagora
 - Numero di righe : N
 - Numero di colonne : M
 - Tipo degli elementi : int
- Le righe sono numerate da 0 ad N-1
- Le colonne sono numerate da 0 ad M-1
- In totale ci sono N×M elementi
- In generale M≠N; per matrici quadrate, M=N

9

Identificazione degli elementi



Lavorare con le matrici

- Ogni operazione su una matrice deve essere svolta lavorando singolarmente su ciascuno degli elementi
- Ciò solitamente significa dover ricorrere a due cicli annidati

```
for(i=0; i<N; i++) /* righe */
{
    for(j=0; j<M; j++) /* colonne */
    {
        somma = somma + matrice[i][j] ;
    }
}
```

11



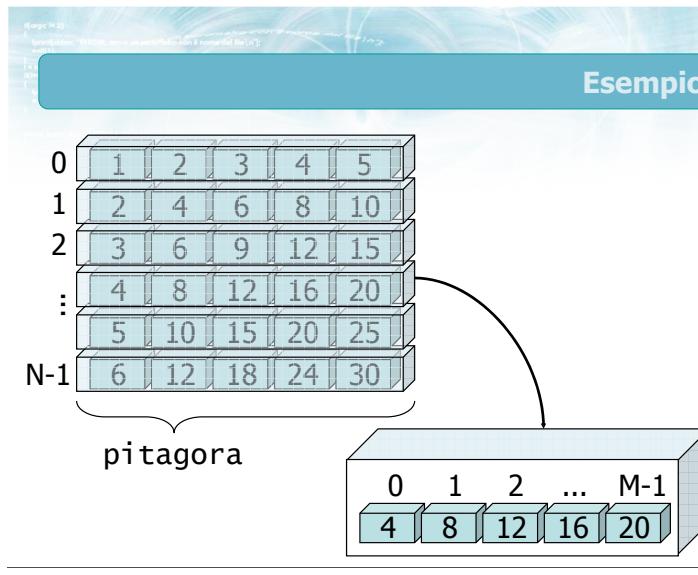
Matrici

Matrici come vettori di vettori

13

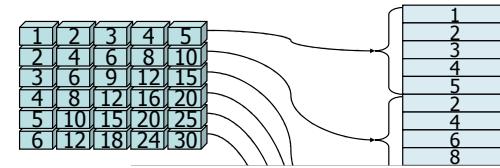
Vettori di vettori

- Un altro modo di vedere le matrici è concepirle come vettori di vettori:
- Un vettore di N oggetti (righe)
- Ciascun oggetto (riga) è composto da M elementi (colonne)
- Questa prende il nome di **rappresentazione "per righe"** di una matrice



Codifica

- Nella realtà, poiché la memoria di un calcolatore è uni-dimensionale, le matrici vengono effettivamente memorizzate "per righe"



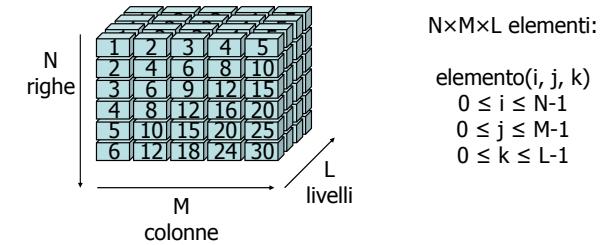
15

Matrici

Matrici pluridimensionali

Matrici con più dimensioni

- Il concetto di matrice può essere generalizzato anche a più di 2 dimensioni
 - Non vi sono, a priori, limiti sul numero di dimensioni



17

Il(ore) (#3)
Scrivere il nome del file con il nome del file ("n") e
wall(1).

Caratteristiche

- ▶ Anche le matrici a più dimensioni condividono vincoli di base dell'intera famiglia degli array:
 - **Tipo di elementi uniforme**
 - **Dimensioni fissate a priori**
 - **Indici interi a partire da 0**
 - ▶ In pratica è molto raro utilizzare più di 3 dimensioni

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

#define MAXPAROLA 30
#define MAXSIGA 80

int main(int argc, char *argv[])
{
    int lung(MAXPAROLA); /* valore di costante
                           cioè numero massimo di caratteri della parola */
    char nome[MAXSIGA];
    int i, indice, lunghezza;
    FILE *fptr;

    if(argc != 2)
    {
        fprintf(stderr, "ERRORE: non è possibile aprire il file %s", argv[1]);
        exit(1);
    }

    if((fptr = fopen(argv[1], "r")) == NULL)
    {
        fprintf(stderr, "ERRORE: impossibile aprire il file %s", argv[1]);
        exit(1);
    }

    while(fgets(nome, MAXSIGA, fptr) != NULL)
    {
        for(i=0; nome[i] != '\0'; i++)
            lung++;
        if(lung > lunghezza)
            lunghezza = lung;
    }

    fclose(fptr);

    printf("Nome della parola con più caratteri: %s\n", nome);
}
```

Matrici – Vettori di stringhe

Definizione di matrici in C

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

#define MAXPAROLA 30
#define MAXSIGA 80

int main(int argc, char *argv[])
{
    int lung(MAXPAROLA); /* valore di costante
                           cioè numero massimo di caratteri della parola */
    char nome[MAXSIGA];
    int i, indice, lunghezza;
    FILE *fptr;

    if(argc != 2)
    {
        fprintf(stderr, "ERRORE: non è possibile aprire il file %s", argv[1]);
        exit(1);
    }

    if((fptr = fopen(argv[1], "r")) == NULL)
    {
        fprintf(stderr, "ERRORE: impossibile aprire il file %s", argv[1]);
        exit(1);
    }

    while(fgets(nome, MAXSIGA, fptr) != NULL)
    {
        for(i=0; nome[i] != '\0'; i++)
            lung++;
        if(lung > lunghezza)
            lunghezza = lung;
    }

    fclose(fptr);

    printf("Nome della parola con più caratteri: %s\n", nome);
}
```

Definizione di matrici in C

Sintassi della definizione

(Argomento: Definizione di matrici in C)

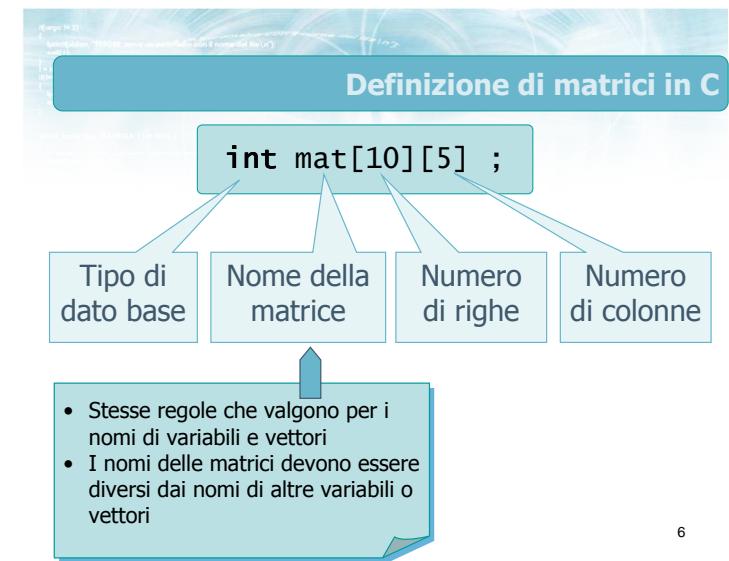
Definizione di matrici in C

- Sintassi della definizione
- Operazioni di accesso

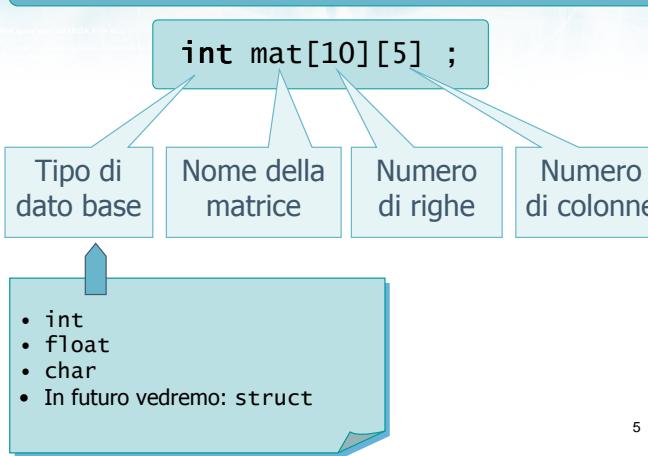
2



4

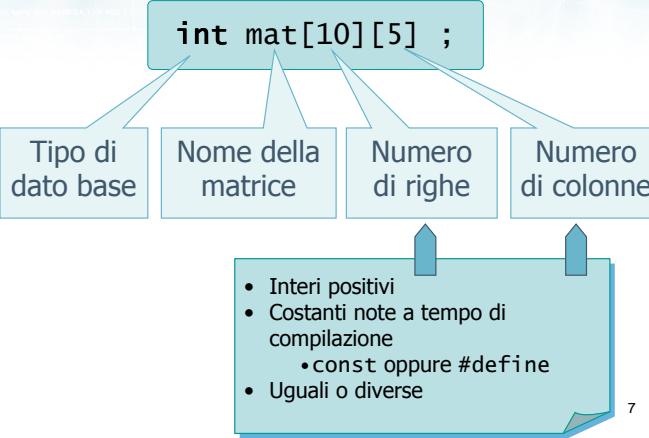


5



6

Definizione di matrici in C



Esempi

» int pitagora[10][10] ;

0	1	2	3	4	5	6	7	8	9	10
1	2	4	6	8	10	12	14	16	18	20
2	3	6	9	12	15	18	21	24	27	30
3	4	8	12	16	20	24	28	32	36	40
4	5	10	15	20	25	30	35	40	45	50
5	6	12	18	24	30	36	42	48	54	60
6	7	14	21	28	35	42	49	56	63	70
7	8	16	24	32	40	48	56	64	72	80
8	9	18	27	36	45	54	63	72	81	90
9	10	20	30	40	50	60	70	80	90	100

8

Esempi

» int pitagora[10][10] ;
 » char tris[3][3] ;

0	1	2
1	.	.
2	.	.

0	1	2
1	.	X
2	.	.

0	O	.
1	.	X
2	.	.

9

Esempi

» int pitagora[10][10] ;
 » char tris[3][3] ;
 » float rot[2][2] ;

0	1
1	-0.707

0	0.500
1	-0.866



10

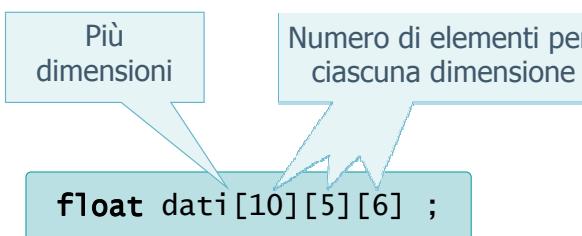
0	0	1
1	cos β	sin β
0	-sin β	cos β

0	0	1
1	0.000	1.000
0	-1.000	0.000



Matrici a più dimensioni

int mat[10][5] ;



11

Errore frequente

» Dichiarare una matrice usando variabili anziché costanti per le dimensioni

int N = 10 ;
 int mat[N][N] ;

int mat[10][10] ;

const int N = 10 ;
 int mat[N][N] ;



Errore frequente

- Dichiarare una matrice usando il nome degli indici

```
int i, j;
int mat[i][j];
```

```
. for(i=0; i<10; i++)
    for(j=0; j<10; j++)
        scanf("%d", &mat[i][j]);
```

```
const int N = 10 ;
int mat[N][N] ;
```

14

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

#define MAXPAROLA 30
#define MAXSIGA 80

int main(int argc, char *argv[])
{
    int maxParola, /* valore di punteggio
    const int maxSiga, /* massimo numero di righe */
    char siga[MAXSIGA];
    int i, j, lunghezza;
    FILE *f;

    if(argc != 2)
    {
        printf("Usage: %s <file>\n", argv[0]);
        exit(1);
    }
    if((f=fopen(argv[1], "r")) == NULL)
    {
        perror("Error, impossibile aprire il file %s", argv[1]);
        exit(1);
    }

    fgets(sigा, MAXSIGA, f) == NULL;
```

Definizione di matrici in C



Operazioni di accesso

16



Errore frequente

- Dichiarare una matrice usando il simbolo di "virgola"

```
const int N = 10 ;
const int M = 20 ;
```

```
int mat[N,M] ;
```

```
int mat[N][M] ;
```

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>
```

Sintassi

```
nomematrice[ valoreindice1 ][ valoreindice2 ]
```

Costante, variabile o
espressione aritmetica
con valore intero

17

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>
```

Sintassi

```
nomematrice[ valoreindice1 ][ valoreindice2 ]
```

Costante, variabile o
espressione aritmetica
con valore intero

Valore intero compreso
tra 0 e numero di righe -1

Valore intero compreso
tra 0 e numero di
colonne -1

18

Esempi

pitagora[1][2]					
0	1	2	3	4	
0	1	2	3	4	5
1	2	4	6	8	10
2	3	6	9	12	15
3	4	8	12	16	20
4	5	10	15	20	25
5	6	12	18	24	30

int pitagora[6][5] ;

19

Vincoli (1/2)

- In una matrice NxMxKx..., il valore dell'indice deve essere compreso tra 0 e N-1/M-1/K-1/....
 - La responsabilità è del programmatore
- Se qualche indice non è un numero intero, viene automaticamente troncato

```
pitagora[i][j] = (i+1)*(j+1) ;
x = pitagora[1][2] ;
```

20

Vincoli (2/2)

- Una variabile di tipo matrice può essere utilizzata solamente mediante l'operatore di indicizzazione
 - Occorre agire individualmente sui singoli elementi
 - Non è possibile agire sull'intera matrice in una sola istruzione

```
pitagora[i][j] = (i+1)*(j+1) ;
x = pitagora[1][2] ;
```

21

Uso di una cella di un vettore

- L'elemento di una matrice è utilizzabile come una qualsiasi variabile:
 - Utilizzabile all'interno di un'espressione
 - tot = tot + mat[i][j] ;
 - Utilizzabile in istruzioni di assegnazione
 - mat[0][0] = 0 ;
 - Utilizzabile per stampare il valore
 - printf("%d\n", mat[z][k]) ;
 - Utilizzabile per leggere un valore
 - scanf("%d\n", &mat[k][z]) ;

22

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

#define MAXPAROLA 30
#define MAXRIGA 80

int main(int argc, char *argv[])
{
    int freqMAXPAROLA; /* valore di controllo delle frequenze delle parole */
    char parola[MAXPAROLA];
    int i, indice, lunghezza;
    FILE *fptr;

    if(argc != 2)
    {
        fprintf(stderr, "ERRORE: non è possibile calcolare il numero del file\n");
        exit(1);
    }

    fptr = fopen(argv[1], "r");
    if(fptr == NULL)
    {
        fprintf(stderr, "ERRORE: impossibile aprire il file %s", argv[1]);
        exit(1);
    }

    while(fgets(parola, MAXRIGA, fptr) != NULL)
```

Matrici – Vettori di stringhe

Operazioni elementari sulle matrici

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

#define MAXPAROLA 30
#define MAXRIGA 80

int main(int argc, char *argv[])
{
    int freqMAXPAROLA; /* valore di controllo delle frequenze delle parole */
    char parola[MAXPAROLA];
    int i, indice, lunghezza;
    FILE *fptr;

    for(i=0; i<MAXPAROLA; i++)
        lunghezza[i] = 0;

    if(argc != 2)
    {
        fprintf(stderr, "ERRORE: non è possibile calcolare il numero del file\n");
        exit(1);
    }

    fptr = fopen(argv[1], "r");
    if(fptr == NULL)
    {
        fprintf(stderr, "ERRORE: impossibile aprire il file %s", argv[1]);
        exit(1);
    }

    while(fgets(parola, MAXRIGA, fptr) != NULL)
```

Operazioni elementari sulle matrici

Definizioni

(Argomento: Operazioni elementari sulle matrici)

Operazioni elementari sulle matrici

- ▶ Definizioni
- ▶ Stampa di una matrice
- ▶ Lettura di una matrice
- ▶ Copia di una matrice
- ▶ Somme di riga o di colonna
- ▶ Ricerca di un elemento
- ▶ Ricerca del massimo o del minimo

2

(Argomento: Operazioni elementari sulle matrici)

Definizioni (1/2)

```
const int N = 10 ;
const int M = 5 ; /* dimensioni massime */

float mat[N][M] ; /* matrice 10x5 di reali */
float mat2[N][M] ; /* uguali dimensioni */

float sr[N] ; /* somma per righe */
float sc[M] ; /* somma per colonne */

int i, j ; /* indici dei cicli */
```

matrici.c

matrici.c

4

(Argomento: Operazioni elementari sulle matrici)

Definizioni (2/2)

```
int trovato ; /* flag */
int riga, col ; /* risultati ricerca */

float dato ; /* elemento da ricercare */

float somma, sommar, sommac ; /* per calcolo di somme */

float maxr, maxc ; /* massimi */
```



matrici.c

5

(Argomento: Operazioni elementari sulle matrici)

Operazioni elementari sulle matrici

Stampa di una matrice

(Argomento: Operazioni elementari sulle matrici)

Operazioni elementari sulle matrici

Definizioni

Stampa di una matrice

Lettura di una matrice

Copia di una matrice

Somme di riga o di colonna

Ricerca di un elemento

Ricerca del massimo o del minimo

3

(Argomento: Operazioni elementari sulle matrici)

Operazioni elementari sulle matrici

Definizioni

Stampa di una matrice

Lettura di una matrice

Copia di una matrice

Somme di riga o di colonna

Ricerca di un elemento

Ricerca del massimo o del minimo

2

Stampa di matrici

- Occorre stampare un elemento per volta, all'interno di cicli `for`
- Sono necessari due cicli annidati
 - Il ciclo esterno per scandire le righe (da 0 a $N-1$)
 - Il ciclo interno per scandire ciascuna colonna (da 0 a $M-1$) della riga data
- Si può stampare "per righe" (caso normale) o "per colonne" (trasposta)

7

Stampa per righe matrice di reali

```
printf("Matrice: %d x %d\n", N, M);
```

```
for(i=0; i<N; i++)
```

```
{     Stampa la riga i-esima
```

```
}
```

matrici.c

8

Stampa per righe matrice di reali

```
printf("Matrice: %d x %d\n", N, M);  
  
for(i=0; i<N; i++) /* Stampa la riga i-esima */  
{  
    for(j=0; j<M; j++)  
    {  
        printf("%f ", mat[i][j]);  
    }  
    printf("\n");  
}
```

matrici.c

9

Esempio

Prompt dei comandi

```
Matrice: 10 righe, 5 colonne  
1.000000 0.500000 0.333333 0.250000 0.200000  
2.000000 1.000000 0.666667 0.500000 0.400000  
3.000000 1.500000 1.000000 0.750000 0.600000  
4.000000 2.000000 1.333333 1.000000 0.800000  
5.000000 2.500000 1.666667 1.250000 1.000000  
6.000000 3.000000 2.000000 1.500000 1.200000  
7.000000 3.500000 2.333333 1.750000 1.400000  
8.000000 4.000000 2.666667 2.000000 1.600000  
9.000000 4.500000 3.000000 2.250000 1.800000  
10.000000 5.000000 3.333333 2.500000 2.000000
```

10

Stampa per colonne matrice di reali

```
printf("Matrice: %d x %d\n", N, M);  
  
for(j=0; j<M; j++)  
{  
    for(i=0; i<N; i++)  
    {  
        printf("%f ", mat[i][j]);  
    }  
    printf("\n");  
}
```

matrici.c

11

Esempio

Prompt dei comandi

```
Matrice: 10 righe, 5 colonne  
1.00 2.00 3.00 4.00 5.00 6.00 7.00 8.00 9.00 10.00  
0.50 1.00 1.50 2.00 2.50 3.00 3.50 4.00 4.50 5.00  
0.33 0.67 1.00 1.33 1.67 2.00 2.33 2.67 3.00 3.33  
0.25 0.50 0.75 1.00 1.25 1.50 1.75 2.00 2.25 2.50  
0.20 0.40 0.60 0.80 1.00 1.20 1.40 1.60 1.80 2.00
```

12

- Occorre leggere un elemento per volta
- Si procede per righe (o per colonne)
- Si utilizzano solitamente due cicli annidati

Operazioni elementari sulle matrici

Lettura di una matrice

14

Lettura per righe matrice di reali

```
printf("Immetti matrice %d x %d\n", N, M) ;
for(i=0; i<N; i++)
{
    printf("Riga %d:\n", i+1) ;
    for(j=0; j<M; j++)
    {
        printf("Elemento (%d,%d): ", i+1, j+1) ;
        scanf("%F", &mat[i][j]) ;
    }
}
```



15

Esempio

```
cat Prompt dei comandi
Immetti una matrice 10 x 5
Riga 1:
Elemento (1,1): 3.2
Elemento (1,2): 1
Elemento (1,3): -12.4
Elemento (1,4): 2.112
Elemento (1,5): 23
Riga 2:
Elemento (2,1): 23.1
Elemento (2,2): 2.11
Elemento (2,3): .22
Elemento (2,4): 3.14
Elemento (2,5): 2.71
```

16

Operazioni elementari sulle matrici

Copia di una matrice

18

- L'operazione di copia di una matrice "sorgente" in una "destinazione" richiede che ciascun elemento venga copiato individualmente
- La matrice destinazione deve avere dimensioni uguali o superiori a quelle della sorgente
- L'operazione di copia avviene ovviamente a livello del singolo elemento

Copia di matrici (2/2)

```
for(i=0; i<N; i++)
    for(j=0; j<M; j++)
        mat2[i][j] = mat[i][j] ;
```

matrici.c

19

Operazioni elementari sulle matrici

Somme di riga o di colonna

Sommatorie in matrici

- Il calcolo di totali sui dati contenuti in una matrice può corrispondere a tre diverse operazioni:
 - Somma degli elementi di ciascuna riga (totali di riga)
 - Somma degli elementi di ciascuna colonna (totali di colonna)
 - Somma di tutti gli elementi della matrice

21

Esempio

```
float mat[N][M] ;
```

1.00	0.50	0.33	0.25	0.20
2.00	1.00	0.67	0.50	0.40
3.00	1.50	1.00	0.75	0.60
4.00	2.00	1.33	1.00	0.80
5.00	2.50	1.67	1.25	1.00
6.00	3.00	2.00	1.50	1.20
7.00	3.50	2.33	1.75	1.40
8.00	4.00	2.67	2.00	1.60
9.00	4.50	3.00	2.25	1.80
10.00	5.00	3.33	2.50	2.00

22

Esempio

```
float mat[N][M] ;
float sr[N] ;
```

1.00	0.50	0.33	0.25	0.20	2.28
2.00	1.00	0.67	0.50	0.40	4.56
3.00	1.50	1.00	0.75	0.60	6.85
4.00	2.00	1.33	1.00	0.80	9.13
5.00	2.50	1.67	1.25	1.00	11.41
6.00	3.00	2.00	1.50	1.20	13.70
7.00	3.50	2.33	1.75	1.40	15.98
8.00	4.00	2.67	2.00	1.60	18.26
9.00	4.50	3.00	2.25	1.80	20.55
10.00	5.00	3.33	2.50	2.00	22.83

55.00	27.50	18.33	13.75	11.00	float sc[M] ;
-------	-------	-------	-------	-------	---------------

23

```
#include <stdio.h>
#include <math.h>
#include <ctype.h>

#define MAXPAROLA 30
#define MAXSIGA 60

int main(int argc, char *argv[])
{
    int freq[MAXPAROLA]; /* valore di controllo delle frequenze delle parole */
    int l, maxfreq; /* l: indice, maxfreq: */

    if(argc != 2)
    {
        fprintf(stderr, "ERRORE: non sono stati inseriti i parametri del file\n");
        exit(1);
    }
    if((l = fopen(argv[1], "rt")) == NULL)
    {
        perror(argv[1]);
        exit(1);
    }

    for(l = 0; l < MAXPAROLA; l++)
        freq[l] = 0;

    while(fgets(argv[1], 1000, l) != NULL)
    {
        for(l = 0; l < MAXSIGA; l++)
            if(isalpha(argv[1][l]))
                freq[l]++;
    }

    for(l = 0; l < MAXPAROLA; l++)
        if(freq[l] > maxfreq)
            maxfreq = freq[l];
}
```

Operazioni elementari sulle matrici

Somma per righe

```
for(i=0 ; i<N ; i++)
```

```
{
    somma = 0.0 ;
    for(j=0; j<M; j++)
        somma = somma + mat[i][j] ;
    sr[i] = somma ;
}
```

```
for(i=0; i<N; i++)
    printf("Somma riga %d = %f\n",
           i+1, sr[i]) ;
```

24

Somma per colonne

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

#define MAXPAROLA 30
#define MAXRIGA 80

int main(int argc, char *argv[])
{
    int lungMAXPAROLA; /* valore di parola */
    char mat[MAXRIGA][MAXPAROLA]; /* matrice delle parole */
    char datoMAXRIGA;
    int i, j, lungRiga;
    int k=1;

    for(i=0; i<MAXRIGA; i++)
        lungRiga=0;

    if(argc > 1)
    {
        lungMAXPAROLA = strlen(argv[1]);
        if(lungMAXPAROLA > MAXPAROLA)
            lungMAXPAROLA = MAXPAROLA;
        else
            lungMAXPAROLA = lungMAXPAROLA;
    }
    else
        lungMAXPAROLA = MAXPAROLA;

    for(i=0; i<lungMAXPAROLA; i++)
        datoMAXRIGA = 'A'+i;

    if(argc > 1)
    {
        for(i=0; i<lungMAXPAROLA; i++)
            mat[0][i] = datoMAXRIGA;
    }
    else
        mat[0][0] = 'A';

    for(i=1; i<MAXRIGA; i++)
        mat[i][0] = '\0';

    printf("Inserire le righe della matrice: \n");
    for(i=1; i<MAXRIGA; i++)
    {
        for(j=0; j<lungMAXPAROLA; j++)
        {
            if(j==lungRiga)
                mat[i][j] = '\0';
            else
                mat[i][j] = mat[0][j];
        }
    }

    for(i=1; i<MAXRIGA; i++)
        lungRiga=0;
}

int main()
{
    int i, j, somma;
    float sc[100];
    double mat[100][100];

    for(i=0; i<100; i++)
        for(j=0; j<100; j++)
            mat[i][j] = 0.0;

    for(i=0; i<100; i++)
        sc[i] = 0.0;

    for(i=0; i<100; i++)
    {
        for(j=0; j<100; j++)
        {
            if(mat[i][j] > 0.0)
                sc[i] = sc[i] + mat[i][j];
        }
    }

    for(i=0; i<100; i++)
        printf("Somma colonna %d = %f\n", i+1, sc[i]);
}
```

matrici.c

25

Somma complessiva

```
somma = 0.0 ;
for(i=0 ; i<N ; i++)
{
    for(j=0; j<M; j++)
        somma = somma + mat[i][j] ;
}

printf("Somma complessiva = %f\n",
       somma) ;
```

matrici.c

26



Operazioni elementari sulle matrici

Ricerca di un elemento

```
printf("Data da ricercare: ");
scanf("%f", &dato) ;

trovato = 0 ;
riga = -1 ;
col = -1 ;

for(i=0; i<N && trovato==0; i++)
    for(j=0; j<M && trovato==0; j++)
        if( mat[i][j]==dato )
        {
            trovato=1 ;
            riga = i ;
            col = j ;
        }
```

matrici.c

29

Ricerca di elementi

- Dato un valore dato, ricercare se esso esiste (almeno una volta) nella matrice
- In caso affermativo, specificare la riga e la colonna
- Si utilizzano i soliti due cicli annidati

28

Ricerca elemento (1/2)

```
if(trovato==1)
    printf("Data %f presente: (%d,%d)\n",
           dato, riga, col) ;
else
    printf("Data %f non presente\n",
           dato) ;
```

matrici.c

30

```

#include <stdio.h>
#include <ctype.h>

#define MAXFAROLA 30
#define MAXRIGA 60

int main(int argc, char *argv[])
{
    int freq[MAXFAROLA]; /* vettore di contenuto delle frequenze delle lunghezze delle parole */
    int i, minf, lunghezza; /* i è indice, lunghezza è il valore */
    FILE *fp;

    fp=fopen(argv[1], "r");
    if(fp==NULL)
    {
        fprintf(stderr, "ERRORE: impossibile aprire il file %s", argv[1]);
        exit(1);
    }
    /* legge le parole da file */
    for(i=0; i<MAXRIGA; i++)
    {
        fscanf(fp, "%s", freq+i);
        if(freq[i]==NULL)
            break;
    }
    /* stampa le parole */
    for(i=0; i<minf; i++)
        printf("%s\n", freq+i);
    /* chiude il file */
    fclose(fp);
}

```

Operazioni elementari sulle matrici

Ricerca del massimo o del minimo

32

- Per quanto riguarda il calcolo di massimi e minimi si possono distinguere i 3 scenari:
 - Massimo/minimo degli elementi di ciascuna riga
 - Massimo/minimo degli elementi di ciascuna colonna
 - Massimo/minimo tra tutti gli elementi della matrice

Massimo per righe

```

for(i=0; i<N; i++)
{
    col = 0 ;
    maxr = mat[i][0] ;
    for(j=1; j<M; j++)
        if( mat[i][j] > maxr )
        {
            maxr = mat[i][j] ;
            col = j ;
        }

    printf("Max di riga %d vale %f
          e si trova nella colonna %d\n",
          i+1, maxr, col+1) ;
}

```

33



Osservazioni

- Le operazioni qui citate sono gli elementi fondamentali dell'elaborazione di matrici
- Nei problemi concreti si osserveranno delle combinazioni di tali operazioni
- Occorre analizzare il problema, scomporlo nei suoi sottoproblemi e combinare opportunamente le varie tecniche

35

Massimo per colonne

```

for(j=0; j<M; j++)
{
    riga = 0 ;
    maxc = mat[0][j] ;
    for(i=1; i<N; i++)
        if( mat[i][j] > maxc )
        {
            maxc = mat[i][j] ;
            riga = i ;
        }

    printf("Max di colonna %d vale %f
          e si trova nella riga %d\n",
          j+1, maxc, riga+1) ;
}

```

34



Esercizio “Max Sum Abs”

- Data una matrice NxN, determinare la riga in cui la somma dei valori assoluti degli elementi sia massima

$$r = \max_i \left(\sum_j |M_{ij}| \right)$$

36

Soluzione 1

- » Inizializza max
- » Per ogni riga i :
 - Calcola la somma sommar dei valori assoluti di tale riga
 - Confronta sommar con il max corrente, ed eventualmente aggiorna il max
- » Stampa max

37

Soluzione 1

```
max = -1.0 ;  
  
for(i=0; i<N; i++)  
{  
    sommar = 0.0 ;  
    for(j=0; j<M; j++)  
    {  
        sommar = sommar +  
                  fabs(mat[i][j]) ;  
    }  
  
    if(sommar>max)  
        max = sommar ;  
}  
printf("R = %f\n", max) ;
```

38

Soluzione 2

- » Calcola un vettore di appoggio, di N elementi, contenente le sommatorie per ogni riga
- » Trova il max all'interno di questo vettore di appoggio
- » Soluzione più lunga dal punto di vista del codice, ma più semplice da concepire e realizzare

39

Soluzione 2 (1/2)

```
for(i=0; i<N; i++)  
{  
    sommar = 0.0 ;  
    for(j=0; j<M; j++)  
    {  
        sommar = sommar +  
                  fabs(mat[i][j]) ;  
    }  
    sr[i] = sommar ;  
}
```

40

Soluzione 2 (2/2)

```
max = -1.0 ;  
  
for(i=0; i<N; i++)  
    if(sr[i]>max)  
        max = sr[i] ;  
  
printf("R = %f\n", max) ;
```

41

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

#define MAXPAROLA 30
#define MAXSIGA 80
```

Matrici – Vettori di stringhe

Vettori di stringhe

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

#define MAXPAROLA 30
#define MAXSIGA 80

int main(int argc, char *argv[])
{
    int lungMAXPAROLA; /* vettore di caratteri
    come lunghezza della parola */
    char siglaMAXSIGA; /* vettore di caratteri
    come lunghezza della parola */
    int i, indice, lungparola;
    FILE *fptr;

    fptr=fopen(argv[1], "r");
    lungMAXPAROLA=1;
    lungMAXSIGA=1;
    for(i=0; i<MAXPAROLA; i++)
        lungMAXPAROLA++;
    for(i=0; i<MAXSIGA; i++)
        lungMAXSIGA++;

    if(argc < 2)
    {
        printf("Errore: non è possibile aprire il file %s", argv[1]);
        exit(1);
    }
    else
    {
        if(fopen(argv[1], "r") == NULL)
        {
            printf("Errore: impossibile aprire il file %s", argv[1]);
            exit(1);
        }
    }

    while(fgets(sigla, MAXSIGA, fptr) != NULL)
```

Vettori di stringhe

Matrici di caratteri

(Argomento: Vettori di stringhe)

Vettori di stringhe

- ▶ Matrici di caratteri
- ▶ Vettori di stringhe
- ▶ I/O di vettori di stringhe

2

(Argomento: Matrici di caratteri)

Esercizio “Verifica Sudoku”

- ▶ Si realizzi un programma in C che verifichi la corretta soluzione della griglia di un “Sudoku”
- ▶ Il programma acquisisce da tastiera la griglia 9x9, in cui ciascun elemento è un carattere tra 1 e 9
- ▶ Il programma deve verificare se il Sudoku è stato correttamente risolto

(Argomento: Matrici di caratteri)

Matrici di caratteri

- ▶ Nel definire una matrice, è ovviamente possibile usare il tipo base **char**
- ▶ Permette di memorizzare una tabella NxM di caratteri ASCII
- ▶ Ogni posizione [i][j] deve contenere **un carattere**
 - Non può essere vuota
 - Non può contenere più di un carattere

char tris[3][3] ;

	0	1	2
0	O	X	.
1	.	X	.
2	.	.	.

4

(Argomento: Esempio)

Esempio

5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9

5

6

Analisi

- » Tutti i valori devono essere singoli caratteri tra '1' e '9'
- » Su ciascuna delle 9 righe, non devono esserci valori ripetuti
- » Su ciascuna delle 9 colonne, non devono esserci valori ripetuti
- » In ciascuno dei 9 blocchi 3x3, non devono esserci valori ripetuti

7

Soluzione (1/10)

```
const int N = 9 ;  
char sudoku[N][N] ;  
  
int i, j, k ; /* indici dei cicli */  
int r1, c1, r2, c2 ;  
char ch ;  
int err, good ; /* flag */  
  
printf("verifica Sudoku\n") ;
```

sudoku.c

8

Soluzione (2/10)

```
for(i=0; i<N; i++)  
{  
    printf("Riga %d:\n", i+1) ;  
    for(j=0; j<N; j++)  
    {  
        Acquisisci un carattere  
        ch tra '1' e '9'  
        sudoku[i][j] = ch ;  
    }  
}
```

sudoku.c

Acquisisci un carattere
ch tra '1' e '9'

9

Soluzione (2/10)

```
do {  
    printf(" Colonna %d: ", j+1) ;  
    ch = getchar() ;  
    if( ch<'1' || ch>'9' )  
        printf("Errata - ripeti\n") ;  
  
    /* elimina fino fine linea */  
    while( getchar() !='\n')  
        /*nop*/ ;  
} while( ch<'1' || ch>'9') ;
```

sudoku.c

10

Soluzione (3/10)

```
/* Stampa il tabellone */  
for(i=0; i<9; i++)  
{  
    for(j=0; j<9; j++)  
    {  
        printf("%c ", sudoku[i][j]) ;  
        if(j==2 || j==5)  
            printf(" ") ;  
    }  
    printf("\n") ;  
    if(i==2 || i==5)  
        printf("\n") ;  
}
```

sudoku.c

11

Soluzione (4/10)

```
good = 1 ; /* flag generale */  
  
/* Verifica le righe */  
for(i=0; i<N; i++)  
{  
    printf("Riga %d: ", i+1) ;  
    err = 0 ;  
  
    /* ricerca duplicati su col. j,k */  
    for(j=0; j<N; j++)  
        for(k=j+1; k<N; k++)  
            if(sudoku[i][j]==  
                sudoku[i][k])  
                err = 1 ;
```

sudoku.c

12

Soluzione (5/10)

```

if(err==0)
    printf("OK\n");
else
{
    printf("Errore!\n");
    good = 0 ;
}
}

```



13

Soluzione (6/10)

```

for(i=0; i<N; i++) /* colonne */
{
    printf("Colonna %d: ", i+1) ;
    err = 0 ;
    /* ricerca dupl. su righe j,k */
    for(j=0; j<N; j++)
        for(k=j+1; k<N; k++)
            if(sudoku[j][i]==sudoku[k][i])
                err = 1 ;

    if(err==0) printf("OK\n");
    else
    {
        printf("Errore!\n");
        good = 0 ;
    }
}

```



14

Ricerca per blocchi

5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9

15

Ricerca per blocchi

	j	j+2							
i									
i+2									
	5	3	4	6	7	8	9	1	2
	6	7	2	1	9	5	3	4	8
	1	9	8	3	4	2	5	6	7
	8	5	9	7	6	1	4	2	3
	4	2	6	8	5	3	7	9	1
	7	1	3	9	2	4	8	5	6
	9	6	1	5	3	7	2	8	4
	2	8	7	4	1	9	6	3	5
	3	4	5	2	8	6	1	7	9

16

Ricerca per blocchi

	j	j+2							
i									
i+2									
	5	3	4	6	7	8	9	1	2
	6	7	2	1	9	5	3	4	8
	1	9	8	3	4	2	5	6	7
	8	5	9	7	6	1	4	2	3
	4	2	6	8	5	3	7	9	1
	7	1	3	9	2	4	8	5	6
	9	6	1	5	3	7	2	8	4
	2	8	7	4	1	9	6	3	5
	3	4	5	2	8	6	1	7	9

17

Soluzione (7/10)

```

for(i=0; i<N; i=i+3)
{
    for(j=0; j<N; j=j+3)
    {
        printf("Blocco (%d,%d)-(%d,%d): ",
               i+1, j+1, i+3, j+3) ;
        /* ricerca duplicati nel blocco */
        /* Confronta [r1][c1]
           con [r2][c2] */

        err = 0 ;
    }
}

```



18

Soluzione (8/10)

```

for(r1=i; r1<i+3; r1++)
for(c1=j; c1<j+3; c1++)
{
    /* elemento [r1][c1]... */
    for(r2=i; r2<i+3; r2++)
    for(c2=j; c2<j+3; c2++)
    {
        /* ..rispetto a [r2][c2] */
        if( ((r1!=r2)|| (c1!=c2)) &&
            sudoku[r1][c1]==
                sudoku[r2][c2] )
        {
            err = 1 ;
        }
    } /*r2,c2*/
} /*r1,c1*/

```

sudoku.c

19

Soluzione (9/10)

```

if(err==0)
    printf("OK\n");
else
{
    printf("Errore!\n");
    good = 0 ;
}
} /*i*/

```

sudoku.c

20

Soluzione (10/10)

```

if(good==1)
    printf("Sudoku valido!\n");
else
    printf("Sudoku errato...\n");

```

sudoku.c

21

#include <stdio.h>
#include <string.h>
#include <ctype.h>

#define MAXPAROLA 30
#define MAXRIGA 80

```

int main(int argc, char *argv[])
{
    int lungMAXPAROLA; /* dimensione massima di una parola */
    int lungMAXRIGA; /* dimensione massima di una riga */
    int i, j, k, lungparola;
    FILE *f;
    char s[MAXRIGA];
    char s1[MAXPAROLA];
    fgets(s, MAXRIGA, f);
    lungparola = strlen(s1);
    if(lungparola > lungMAXPAROLA)
    {
        printf("ERRORE, parola troppo lunga\n");
        exit(1);
    }
    if(lungparola < lungMAXPAROLA)
    {
        printf("ERRORE, parola troppo corta\n");
        exit(1);
    }
    if(argv[1] == NULL)
    {
        printf("ERRORE, non è stato specificato il nome del file\n");
        exit(1);
    }
    while(fgets(s, MAXRIGA, f) != NULL)
    {
        if(strlen(s) > lungMAXRIGA)
        {
            printf("ERRORE, riga troppo lunga\n");
            exit(1);
        }
    }
}
```

Vettori di stringhe

Vettori di stringhe

Vettori di stringhe

- » Una matrice di caratteri può anche essere vista come:
 - Un vettore di vettori di caratteri, cioè
 - Un vettore di stringhe
- » Si tratta di un metodo diverso di interpretare la stessa struttura dati

23

Esempio

```
char tris[3][3] ;
```

0	1	2
O	X	.
.	X	.

```
char nomi[5][10] ;
```

	0	1	2	3	4	5	6	7	8	9
0	F	u	l	v	i	o	\0	x	!	w
1	A	n	t	o	n	i	\0	.	z	
2	C	r	i	s	t	i	n	a	\0	u
3	E	l	e	n	a	\0	5	g	r	d
4	D	a	v	i	d	e	\0	\$	2	e

24

Caratteristiche

5 stringhe di lunghezza variabile
Lunghezza max 9 caratteri
Terminatore nullo in ogni stringa (riga)

```
char nomi[5][10] ;
```

0	1	2	3	4	5	6	7	8	9
0	F	u	l	v	i	o	\0	x	!
1	A	n	t	o	n	i	\0	.	Z
2	C	r	i	s	t	i	n	a	\0
3	E	l	e	n	a	\0	5	g	r
4	D	a	v	i	d	e	\0	\$	2

25

Sintassi

```
char vett[MAX][LUN] ;
```

Nome del vettore di stringhe

Numero di stringhe

Lunghezza massima di ciascuna stringa (compreso terminatore nullo)

26

Sintassi

Definizione

```
char vett[MAX][LUN] ;
```

Accesso al singolo carattere

```
vett[i][j]
```

Carattere (j+1)-esimo della stringa
(i+1)-esima, da usarsi per l'elaborazione carattere-per-carattere

27

Sintassi

Definizione

```
char vett[MAX][LUN] ;
```

Accesso

Stringa (i+1)-esima, da usarsi con le funzioni di libreria

Accesso all'intera stringa

```
vett[i]
```

28

Esempio 1

- Dato un vettore di stringhe, determinare quante volte è presente la lettera 'A' (maiuscola o minuscola)

```
cont = 0 ;
for(i=0; i<MAX; i++)
{
    for(j=0; vett[i][j]!=0; j++)
    {
        if(toupper(vett[i][j])=='A')
            cont++ ;
    }
}
```

29

Esempio 2

- Dato un vettore di stringhe, determinare se esistono due stringhe identiche

```
uguali = 0 ;
for(i=0; i<MAX; i++)
{
    for(k=i+1; k<MAX; k++)
    {
        if(strcmp(vett[i], vett[k])==0)
            uguali=1 ;
    }
}
```

30

Occupazione variabile

- In un vettore di stringhe, ogni riga (ogni stringa) è intrinsecamente un vettore di caratteri ad occupazione variabile
 - Terminatore nullo per indicare la lunghezza effettiva
- Il numero di stringhe effettivamente memorizzato potrebbe non riempire l'intero vettore
 - Variabile intera che indica l'effettiva occupazione del vettore

31

```
const int MAX = 5 ;  
const int LUN = 9 ;
```

```
char nomi[MAX][LUN+1] ;  
int N ;
```

	0	1	2	3	4	5	6	7	8	9
0	F	u	l	v	i	o	\0	x	!	w
1	A	n	t	o	n	i	\0	.	z	
2	C	r	i	s	t	a	\0	u		
3	e	4	1)	a	\0	5	g	r	d
4	1	%	<	d	d	e	g	\$	2	e

N=3

32



Errore frequente

- Confondere una stringa (vettore di caratteri) con un vettore di stringhe (matrice di caratteri)

```
char s[LUN+1] ;
```

```
char v[MAX][LUN+1] ;
```

- $s[i]$ è un singolo carattere
- s è l'intera stringa

- $v[i][j]$ è il singolo carattere
- $v[i]$ è un'intera stringa
- v è l'intera matrice

33

```
#include <stdio.h>  
#include <string.h>  
#include <ctype.h>
```

```
#define MAXPAROLA 30  
#define MAXIGRA 80
```

```
int main(int argc, char *argv[])
{
    int lungMAXPAROLA; /* vettore di punteggio
    controlla se un singolo carattere è un terminatore */
    char s[MAXPAROLA];
    int lung, lungmax;
    FILE *f;

    f=fopen(argv[1], "r");
    if(f==NULL)
    {
        printf("ERRORE, impossibile aprire il file %s\n", argv[1]);
        exit(1);
    }

    while(fgets(s, MAXIGRA, f)!=NULL)
```

Vettori di stringhe

I/O di vettori di stringhe

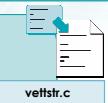
Stampa (1/2)

- La stampa del contenuto di un vettore di stringhe si ottiene semplicemente stampando ciascuno degli elementi
 - Si può utilizzare `puts` oppure `printf`

35

```
for(i=0; i<N; i++)
{
    puts(vett[i]) ;
}
```

Stampa (2/2)



vettstr.c

36

Lettura

- » Acquisire da tastiera un vettore di stringhe
- » Un ciclo per ciascuna delle stringhe da leggere
 - Lunghezza nota a priori
 - Lunghezza determinata dalla lettura di un certo dato (es.: "FINE")
- » Acquisizione, nel ciclo, di ciascuna delle stringhe
 - Utilizzo della funzione `gets`
 - Eventualmente, lettura in una stringa d'appoggio per la verifica di correttezza, prima di ricopiare nel vettore destinazione

37

Dimensione nota a priori (1/2)

```
char vett[MAX][LUN+1] ;  
char s[250] ;  
...  
do {  
    printf("Quante stringhe? ") ;  
    gets(s) ;  
    N = atoi(s) ;  
    if(N<1 || N>MAX)  
        printf("Valore errato: deve  
                essere tra 1 e %d\n", MAX) ;  
} while(N<1 || N>MAX) ;
```

38

Dimensione nota a priori (2/2)

```
for(i=0; i<N; i++)  
{    printf("Stringa %d: ", i+1) ;  
    gets(s) ;  
    if (strlen(s)==0)  
    {        printf("Vuota - ripeti\n");  
        i-- ;  
    }  
    else if(strlen(s)>LUN)  
    {        printf("Troppo lunga -  
                    max %d chr\n", LUN) ;  
        i-- ;  
    }  
    else  
        strcpy(vett[i], s) ;  
}
```

39

Lettura terminata da "FINE"

```
N = 0 ;  
end = 0 ;  
do {  
    printf("Stringa %d: ", N+1) ;  
    gets(s) ;  
    if (strlen(s)==0)  
        printf("Vuota - ripeti\n");  
    else if(strlen(s)>LUN)  
        printf("Troppo lunga\n") ;  
    else if(strcmp(s, "FINE")==0)  
        end = 1 ;  
    else  
    {        strcpy(vett[N], s) ;  
        N++ ;  
    }  
} while(end==0) ;
```

40

- Esercizio "Statistiche testo"
- Esercizio "Magazzino"

Matrici – Vettori di stringhe

Esercizi proposti

2

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>
```

```
#define MAXPAROLA 30
#define MAXRIGA 80
```

```
int main(int argc, char *argv[])
{
    int lungMAXPAROLA; /* valore di costante
        cioè numero massimo di caratteri per parola */
    char riga[MAXRIGA];
    int i, indice, lungparola;
```

```
for(i=0; i<MAXRIGA; i++)
    riga[i]=0;

if(argc > 1)
{
    lungMAXPAROLA = strlen(argv[1]); /* lunghezza della parola */
    if(lungMAXPAROLA > MAXRIGA)
        lungMAXPAROLA = MAXRIGA;
    else if(lungMAXPAROLA < 1)
        lungMAXPAROLA = 1;
}

for(i=0; i<lungMAXPAROLA; i++)
    riga[i] = toupper(argv[1][i]);

if(strcmp(riga, "FINE") == 0)
    break;

printf("FINE, inserisci opere o fini\n", argv[1]);
}
```

Esercizi proposti

Esercizio "Statistiche testo"

4

Esercizio "Statistiche testo"

- Un utente inserisce una serie di frasi da tastiera, su più righe
- L'inserimento termina quando l'utente inserisce la parola FINE su una riga da sola
- Il programma deve determinare:
 - 1) Quante righe sono state inserite dall'utente
 - 2) Quanti caratteri sono stati inseriti
 - 3) Quanti caratteri alfanumerici sono stati inseriti
 - 4) Quante parole sono state inserite

Analisi

Prompt dei comandi

```
Testo: Nel mezzo del cammin di nostra vita
Testo: mi ritrovai per una selva oscura
Testo: che la diritta via era smarrita.
Testo: FINE
L'utente ha inserito 3 righe
L'utente ha inserito 99 caratteri
L'utente ha inserito 82 caratteri alfanumerici
L'utente ha inserito 19 parole
```

5

Soluzione (1/5)



```
const int MAXRIGHE = 2000 ;
const int LUN = 80 ;

char testo[MAXRIGHE][LUN] ;
int Nrighe ; /* righe inserite */
char riga[LUN*10] ;
int i, j ;
int caratteri, caralfa, parole ;
```

6

Soluzione (2/5)

```
Nrighe = 0 ;  
do {  
    printf("Testo: ");  
    gets(riga) ;  
  
    if( strcmp(riga, "FINE")!=0 )  
    {  
        /*copia riga in testo[Nrighe] ;*/  
        strcpy( testo[Nrighe] , riga ) ;  
  
        Nrighe++ ;  
    }  
} while( strcmp(riga, "FINE")!=0 ) ;
```

statistichesto.c

7

Soluzione (3/5)

```
printf("L'utente ha inserito"  
      " %d righe\n", Nrighe);  
  
caratteri = 0 ;  
for(i=0; i<Nrighe; i++)  
    caratteri = caratteri +  
        strlen(testo[i]) ;  
  
printf("L'utente ha inserito"  
      " %d caratteri\n", caratteri) ;
```

statistichesto.c

8

Soluzione (4/5)

```
caralfa = 0 ;  
for(i=0; i<Nrighe; i++)  
{  
    for(j=0; testo[i][j]!=0; j++)  
    {  
        if( isalnum(testo[i][j] ) )  
            caralfa++ ;  
    }  
}  
  
printf("L'utente ha inserito "  
      "%d caratteri alfanumerici\n",  
      caralfa) ;
```

statistichesto.c

9

Soluzione (5/5)

```
parole = 0 ;  
for(j=0; j<Nrighe; j++)  
{  
    for(i=0; testo[j][i]!=0; i++)  
    {  
        if( isalpha(testo[j][i]) &&  
            (i==0 || !isalpha(testo[j][i-1])))  
        {  
            parole ++ ;  
        }  
    }  
}  
  
printf("L'utente ha inserito "  
      "%d parole\n", parole) ;
```

statistichesto.c

10

```
#include <stdio.h>  
#include <string.h>  
#include <ctype.h>  
  
#define MAXPAROLA 30  
#define MAXRIGA 80  
  
int main(int argc, char *argv[]){  
    int lungMAXPAROLA; /* vettore di dimensione  
    lunghezza massima delle parole */  
    char parolaMAXPAROLA[ ];  
    int i, indice, lungParola ;  
    FILE *f;  
  
    f=fopen(argv[1], "r");  
    if(f==NULL)  
    {  
        printf("ERRORE: non posso aprire il file %s\n", argv[1]);  
        exit(1);  
    }  
  
    fgets(parolaMAXPAROLA, MAXRIGA-1);  
    lungMAXPAROLA = strlen(parolaMAXPAROLA);  
  
    for(i=0; i<lungMAXPAROLA; i++)  
    {  
        if(isalpha(parolaMAXPAROLA[i]))  
            indice++;  
        else  
            if(indice>0)  
                lungParola++;  
            indice=0;  
    }  
  
    printf("lungMAXPAROLA = %d\n", lungMAXPAROLA);  
    printf("lungParola = %d\n", lungParola);  
}
```

Esercizi proposti

Esercizio "Magazzino"

Esercizio "Magazzino" (1/2)

- Un'azienda deve tenere traccia dei beni presenti in un magazzino
- L'utente inserisce da tastiera dei "comandi" nel seguente formato:
 - bene EU quantitàdove:
 - bene è il nome di un bene
 - EU è la lettera ' E' per entrata, ' U' per uscita
 - quantità è la quantità di bene entrata o uscita

12

Esercizio "Magazzino" (2/2)

- L'utente termina il caricamento inserendo un comando pari a "FINE". In tal caso il programma deve stampare le quantità di beni presenti a magazzino

13

```
es> Prompt dei comandi
Comando: viti E 10
Comando: dadi E 50
Comando: viti U 5
Comando: viti E 3
Comando: FINE
viti 8
dadi 50
```

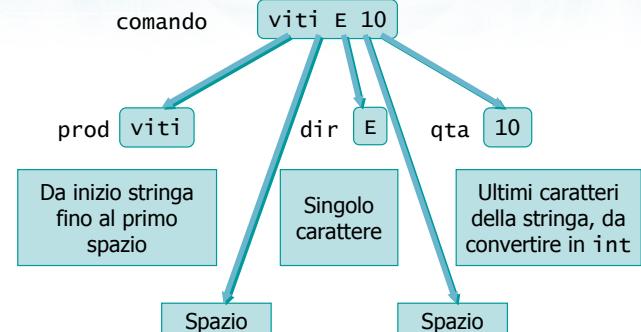
14

Problemi

- Estrazione dei 3 parametri dal comando immesso dall'utente
 - Nome prodotto: stringa **viti E 10**
 - Direzione: carattere
 - Quantità: intero
- Memorizzazione di nomi di prodotti e quantità relative
 - Vettori paralleli
- Aggiornamento delle giacenze

15

Stringa di comando



16

Analisi del comando (1/2)

```
i = 0 ;
while(comando[i]!=' ')
{
    prod[i] = comando[i] ;
    i++ ;
}
prod[i] = 0 ;

/* salta lo spazio */
i++ ;

dir = comando[i] ; /* 'E' o 'U' */
```



17

Analisi del comando (2/2)

```
/* salta lo spazio */
i++ ;

/* da qui a fine: la quantità */
j = 0 ;
while(comando[i]!=0)
{
    temp[j] = comando[i] ;
    i++ ;
    j++ ;
}
temp[j] = 0 ;
qta = atoi(temp) ;
```



18

Osservazione

- » L'analisi di una stringa di comando composta da più campi è sempre un'operazione complessa
- » Nel caso in cui si dovessero gestire delle condizioni anomale (es. più spazi consecutivi) o di errore (es. manca la quantità), il codice diverrebbe estremamente articolato
- » Vedremo più avanti la funzione `sscanf` che può aiutare in questi casi

19

Rappresentazione del magazzino

- » Occorre memorizzare, per ciascun prodotto
 - Il nome
 - La quantità corrente
- » Si possono usare due vettori "paralleli"

```
char prodotti[MAX][LUN+1] ;
int quantita[MAX] ;

int N ; /* occupazione effettiva
          vettori prodotti e quantita */
```

20

Inserimento di un prodotto

- » Determinare se il prodotto è già in magazzino
 - Ricerca del nome del prodotto nel vettore `prodotto`
- » Se c'è già, incrementa la quantità
- » Se non c'è ancora, aggiungi una riga ai vettori

21

Inserimento di un prodotto

```
/* trova la posizione del prodotto */
trovato = -1 ;
for(i=0; i<N; i++)
    if(strcmp(prodotti[i], prod)==0)
        trovato = i ;

if( trovato != -1 ) /* esiste già */
    quantita[trovato] =
        quantita[trovato] + qta ;

else /* prodotto nuovo */
{
    strcpy(prodotti[N], prod) ;
    quantita[N] = qta ;
    N++ ;
}
```

22

Eliminazione di un prodotto

- » Determinare se il prodotto è già in magazzino
 - Ricerca del nome del prodotto nel vettore `prodotto`
- » Se c'è già, decrementa la quantità
- » Se non c'è ancora, errore

23

Eliminazione di un prodotto

```
/* trova la posizione del prodotto */
trovato = -1 ;
for(i=0; i<N; i++)
    if(strcmp(prodotti[i], prod)==0)
        trovato = i ;

if( trovato == -1 )
    printf("Prodotto %s non "
          "trovato in magazzino\n", prod);
else
    quantita[trovato] =
        quantita[trovato] - qta ;
```

24

- ▶ Matrici bi-dimensionali e pluri-dimensionali
- ▶ Matrici di numeri interi e reali
 - Definizione
 - Operazioni frequenti
- ▶ Matrici di caratteri
- ▶ Vettori di stringhe
 - Caso particolare di matrici di caratteri
 - Operazioni frequenti

2

Matrici – Vettori di stringhe

Sommario

Tecniche di programmazione

- ▶ Usare matrici per memorizzare schiere di dati numerici
- ▶ Usare vettori di stringhe per memorizzare stringhe di testo di lunghezza variabile
- ▶ Compire operazioni di ricerca nei vettori di stringhe

3

Materiale aggiuntivo

- ▶ Sul CD-ROM
 - Testi e soluzioni degli esercizi trattati nei lucidi
 - Scheda sintetica
 - Esercizi risolti
 - Esercizi proposti
- ▶ Esercizi proposti da altri libri di testo

4

```
#include <stdio.h>
#include <ctype.h>

#define MAXPARIOLA 30
#define MAXRGA 60

int main(int argc, char *argv[])
{
    int freqMAXPARIOLA; /* valore di confronto delle frequenze delle parola */
    int freqRGA; /* valore di confronto delle frequenze delle parole */
    int i, indice, lunghezza;
    FILE *f;

    if(argc != 2)
    {
        fprintf(stderr, "ERRORE: non sono stati inseriti i parametri necessari\n");
        exit(1);
    }

    f = fopen(argv[1], "r");
    if(f == NULL)
    {
        fprintf(stderr, "ERRORE: impossibile aprire il file %s", argv[1]);
        exit(1);
    }

    /* legge la parola da inserire nel dizionario */
    fscanf(f, "%s", MAXPARIOLA);
    lunghezza = strlen(MAXPARIOLA);
    indice = lunghezza - 1;
    freqMAXPARIOLA = 0;
    freqRGA = 0;
    for(i = 0; i < lunghezza; i++)
    {
        if(isupper(argv[1][i]))
            freqMAXPARIOLA++;
        else
            freqRGA++;
    }
    if(freqMAXPARIOLA > freqRGA)
        printf("Parola inserita nel dizionario\n");
    else
        printf("Parola non inserita nel dizionario\n");

    fclose(f);
}
```

Programmazione in C

Unità Funzioni

2

- » Tipi di dato
- » Funzioni in C
- » Modifica dei parametri
- » Parametri “by reference”
- » La funzione main()
- » Esercizi proposti
- » Sommario

Riferimenti al materiale

» Testi

- Kernighan & Ritchie: capitoli 2 e 4
- Cabodi, Quer, Sonza Reorda: capitoli 3 e 7
- Dietel & Dietel: capitolo 5

» Dispense

- Scheda: “Tipi di dato in C”
- Scheda: “Funzioni in C”

3

- I tipi scalari in C
- Input/output dei tipi scalari
- Conversioni di tipo

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

#define MAXPAROLA 30
#define MAXIGA 80

int main(int argc, char *argv[])
{
    int lungMAXPAROLA; /* valore di controllo
    delle frequenze delle parole */
    int i, indice, lungMAXIGA;
    char riga[MAXIGA];
    FILE *fptr;

    if(argc != 2)
    {
        fprintf(stderr, "ERRORE, non è possibile aprire il file %s", argv[1]);
        exit(1);
    }

    if((fptr = fopen(argv[1], "r")) == NULL)
    {
        perror("ERRORE, impossibile aprire il file %s", argv[1]);
        exit(1);
    }

    while(fgets(riga, MAXIGA, fptr) != NULL)
    {
        for(i=0; riga[i] != '\0'; i++)
        {
            if(isalpha(riga[i]))
                riga[i] = toupper(riga[i]);
            else if(riga[i] == ' ')
                riga[i] = '\0';
        }
        if(strlen(riga) > lungMAXPAROLA)
            riga[lungMAXPAROLA] = '\0';

        if(strcmp(riga, MAXPAROLA) > lungMAXPAROLA)
            lungMAXPAROLA = strlen(riga);
    }

    printf("lungMAXPAROLA = %d\n", lungMAXPAROLA);
    printf("lungMAXIGA = %d\n", lungMAXIGA);
    printf("lungMAXIGA = %d\n", lungMAXIGA);
}
```

Funzioni

Tipi di dato

5

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

#define MAXPAROLA 30
#define MAXIGA 80

int main(int argc, char *argv[])
{
    int lungMAXPAROLA; /* valore di controllo
    delle frequenze delle parole */
    int i, indice, lungMAXIGA;
    char riga[MAXIGA];
    FILE *fptr;

    if(argc != 2)
    {
        fprintf(stderr, "ERRORE, non è possibile aprire il file %s", argv[1]);
        exit(1);
    }

    if((fptr = fopen(argv[1], "r")) == NULL)
    {
        perror("ERRORE, impossibile aprire il file %s", argv[1]);
        exit(1);
    }

    while(fgets(riga, MAXIGA, fptr) != NULL)
    {
        for(i=0; riga[i] != '\0'; i++)
        {
            if(isalpha(riga[i]))
                riga[i] = toupper(riga[i]);
            else if(riga[i] == ' ')
                riga[i] = '\0';
        }
        if(strlen(riga) > lungMAXPAROLA)
            riga[lungMAXPAROLA] = '\0';

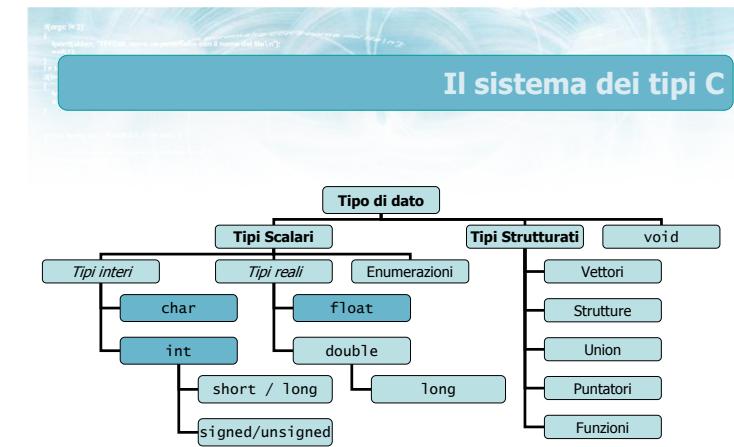
        if(strcmp(riga, MAXPAROLA) > lungMAXPAROLA)
            lungMAXPAROLA = strlen(riga);
    }

    printf("lungMAXPAROLA = %d\n", lungMAXPAROLA);
    printf("lungMAXIGA = %d\n", lungMAXIGA);
    printf("lungMAXIGA = %d\n", lungMAXIGA);
}
```

Tipi di dato

I tipi scalari in C

7



I tipi interi in C

Tipo	Descrizione	Esempi
char	Caratteri ASCII	'a' '7' '!'
int	Interi...	+2 -18 0 +24221
short int	... con meno bit	
long int	... con più bit	
unsigned int	Interi senza segno...	0 1 423 23234
unsigned short int	... con meno bit	
unsigned long int	... con più bit	

8

Quanti bit?

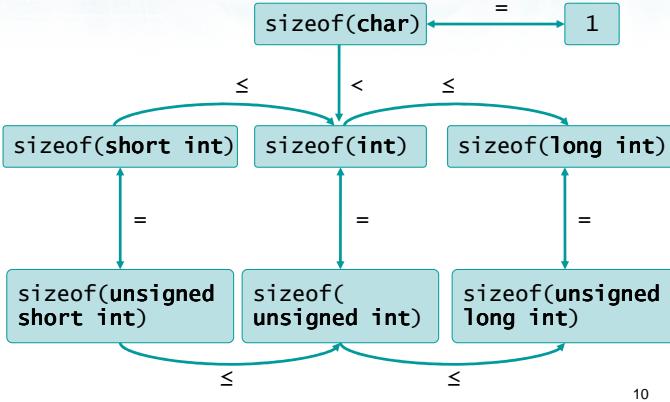
- Lo standard C non specifica l'ampiezza, in bit, dei tipi di dato fondamentali
- L'operatore `sizeof` può essere usato per ricavare una costante pari al numero di byte occupato da ciascun tipo

`sizeof(char)`

`sizeof(int)`

9

Specifiche del C



1

Intervallo di rappresentazione

Tipo	Min	Max
char	CHAR_MIN	CHAR_MAX
int	INT_MIN	INT_MAX
short int	SHRT_MIN	SHRT_MAX
long int	LONG_MIN	LONG_MAX
unsigned int	0	UINT_MAX
unsigned short int	0	USHRT_MAX
unsigned long int	0	ULONG_MAX

```
#include <limits.h>
```

11

Compilatori a 32 bit

Tipo	N. Bit	Min	Max
char	8	-128	127
int	32	-2147483648	2147483647
short int	16	-32768	32767
long int	32	-2147483648	2147483647
unsigned int	32	0	4294967295
unsigned short int	16	0	65536
unsigned long int	32	0	4294967295

I tipi reali in C

Tipo	Descrizione
float	Numeri reali in singola precisione
double	Numeri reali in doppia precisione
long double	Numeri reali in massima precisione

$$A = \underbrace{\pm 1.mmmmm}_{\text{mantissa}} \times 2^{\underbrace{\pm eeeeeee}_{\text{esponente}}}$$

13

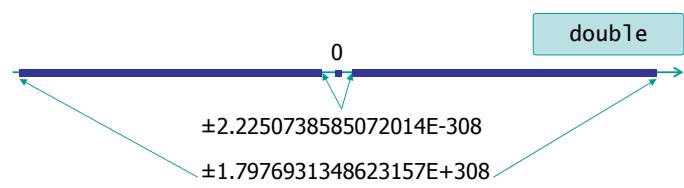
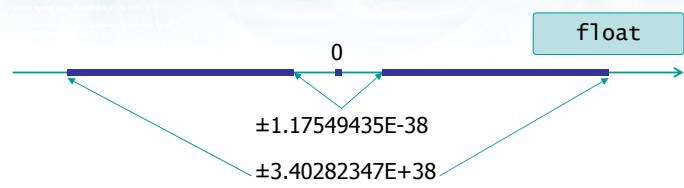
Numero di bit

Tipo	Dimensione	Mantissa	Esponente
float	32 bit	23 bit	8 bit
double	64 bit	53 bit	10 bit
long double		≥ double	

$$A = \pm \underbrace{1.mmmmm}_{\text{mantissa}} \times 2^{\overbrace{+eeeeee}^{\text{esponente}}}$$

1

Intervallo di rappresentazione



15

- I diversi tipi scalari visti sono utilizzabili con le normali funzioni scanf/printf, adottando degli specifici indicatori di formato
- Utilizzando la funzione gets per l'input, si possono usare le funzioni di conversione ato...

Tipi di dato

Input/output dei tipi scalari

17

Specificatori di formato

Tipo	scanf	printf
char	%c %[...]	%c %d
int	%d	%d
short int	%hd	%hd %d
long int	%ld	%ld
unsigned int	%u %o %x	%u %o %x
unsigned short int	%hu	%hu
unsigned long int	%lu	%lu
float	%f	%f %g
double	%lf	%f %g

18

Funzioni di conversione

```
char line[80] ;
int x ;

gets(line) ;
x = atoi(line) ;
```

```
char line[80] ;
float x ;

gets(line) ;
x = atof(line) ;
```

```
char line[80] ;
long int x ;

gets(line) ;
x = atol(line) ;
```

```
char line[80] ;
double x ;

gets(line) ;
x = atof(line) ;
```

19

Tipi di dato

Conversioni di tipo

21

- Nel linguaggio C è possibile combinare, nella stessa espressione, variabili di tipo diverso
 - I due operandi di un operatore aritmetico possono avere tipi diversi

```
int a ;
long int b, c ;
c = b + a ;
```

```
double prod ;
float v[N] ;
prod = prod * v[i] ;
```

Conversioni di tipo (2/2)

- La variabile di destinazione di un'assegnazione può avere tipo diverso dal tipo dell'espressione

```
int a ;  
long int b ;  
b = a ;
```

```
double prod ;  
float v[N] ;  
prod = v[0] ;
```

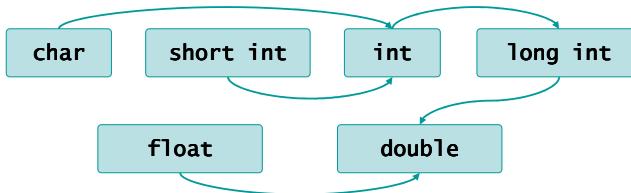
22

Tipologie di conversioni

- Per calcolare tali tipi di espressioni, il linguaggio C applica tre tipi di conversioni:
 - Conversioni "automatiche" verso il tipo più capiente, basate sul principio di **promozione del tipo**
 - Arrotondamenti e troncamenti, in caso di assegnazioni "forzate" a tipi meno capienti
 - Conversioni "esplicite", basate sull'operatore di **typecasting**

Promozione del tipo

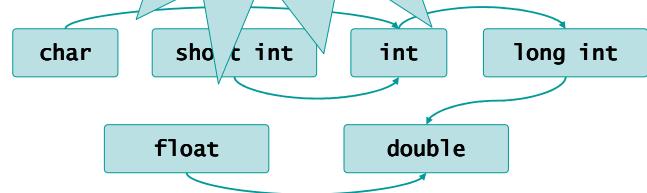
- Se i due operandi di un operatore aritmetico hanno tipo diverso, l'operando del tipo più limitato viene convertito al tipo dell'operando più esteso



24

Promozione del tipo

- Se i due operandi di un operatore aritmetico hanno tipo diverso, l'operando del tipo più limitato viene convertito al tipo dell'operando più esteso.



25

Troncamento del risultato

- Nell'operatore di assegnazione ci possono essere 3 casi:
 - La variabile destinazione ha lo stesso tipo dell'espressione calcolata
 - La variabile destinazione ha un tipo più ampio del tipo dell'espressione calcolata
 - Si promuove il tipo dell'espressione al tipo della variabile destinazione
 - La variabile destinazione ha un tipo più ristretto del tipo dell'espressione calcolata
 - Si approssima il tipo dell'espressione al tipo della variabile destinazione, perdendo precisione

26

(nuovotipo)expr

- Converte l'espressione expr dal suo tipo nativo, al tipo desiderato nuovotipo
 - Più capiente
 - Meno capiente: troncamento o approssimazione

27

Esempio 1

```
double media ;  
int somma, N ;  
  
media = somma / N ; /* no */  
media = (double)somma / N ;
```

28

Esempio 2

```
int voto ;  
float parte1, parte2, parte3 ;  
  
voto = (int) ((parte1 +  
parte2 + parte3)/3) ;
```

29

Esempio 3

```
int voto ;  
float parte1, parte2, parte3 ;  
float media ;  
  
/* arrotondamento all'intero  
più vicino */  
  
media = (parte1 +  
parte2 + parte3)/3 ;  
  
voto = (int) (media + 0.5) ;
```

30

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

#define MAXPAROLA 30
#define MAXSIGA 80

int main(int argc, char *argv[])
{
    int lung(MAXPAROLA); /* valore di controllo delle frequenze delle lunghezze delle parole */
    char str[MAXSIGA]; /* stringa che contiene la parola inserita */
    int i, indice, lunghezza;
    FILE *fp;

    fp=fopen(argv[1], "r");
    if(fp==NULL)
    {
        fprintf(stderr, "ERRORE, non è possibile aprire il file %s", argv[1]);
        exit(1);
    }

    fgets(str, MAXSIGA, fp);
    while(fgets(str, MAXSIGA, fp)!=NULL)
    {
        lunghezza=strlen(str);
        if(lunghezza>lung)
            lung=lunghezza;
    }
    fclose(fp);

    fp=fopen(argv[1], "w");
    if(fp==NULL)
    {
        fprintf(stderr, "ERRORE, non è possibile aprire il file %s", argv[1]);
        exit(1);
    }

    for(i=0;i<lung;i++)
    {
        lung(MAXPAROLA);
        fgets(str, MAXSIGA, fp);
        if(str[i]=='.')
            str[i]='\0';
        else
        {
            str[i]=str[i];
            str[i+1]=str[i+1];
        }
        fprintf(fp, "%c", str[i]);
    }
    fclose(fp);
}
```

Funzioni

Funzioni in C

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

#define MAXPAROLA 30
#define MAXSIGA 80

int main(int argc, char *argv[])
{
    int lung(MAXPAROLA); /* valore di controllo delle frequenze delle lunghezze delle parole */
    char str[MAXSIGA]; /* stringa che contiene la parola inserita */
    int i, indice, lunghezza;
    FILE *fp;

    fp=fopen(argv[1], "r");
    if(fp==NULL)
    {
        fprintf(stderr, "ERRORE, non è possibile aprire il file %s", argv[1]);
        exit(1);
    }

    fgets(str, MAXSIGA, fp);
    while(fgets(str, MAXSIGA, fp)!=NULL)
    {
        lunghezza=strlen(str);
        if(lunghezza>lung)
            lung=lunghezza;
    }
    fclose(fp);

    fp=fopen(argv[1], "w");
    if(fp==NULL)
    {
        fprintf(stderr, "ERRORE, non è possibile aprire il file %s", argv[1]);
        exit(1);
    }

    for(i=0;i<lung;i++)
    {
        lung(MAXPAROLA);
        fgets(str, MAXSIGA, fp);
        if(str[i]=='.')
            str[i]='\0';
        else
        {
            str[i]=str[i];
            str[i+1]=str[i+1];
        }
        fprintf(fp, "%c", str[i]);
    }
    fclose(fp);
}
```

Funzioni in C

Il concetto di funzione



- Il concetto di funzione
- Parametri formali e attuali
- Il valore di ritorno
- Definizione e chiamata di funzioni
- Passaggio dei parametri
- Corpo della funzione

2

Strategie di programmazione

- Riuso di codice esistente
 - Funzionalità simili in programmi diversi
 - Funzionalità ripetute all'interno dello stesso programma
 - Minore tempo di sviluppo
 - Frammenti di codice già verificati
 - Utilizzo di parti di codice scritte da altri
 - Funzioni di libreria
 - Sviluppo collaborativo

4

Come riusare il codice? (1/3)

- Copia-e-incolla
 - Semplice, ma poco efficace
 - Occorre adattare il codice incollato, ritoccando i nomi delle variabili e costanti utilizzati
 - Se si scopre un errore, occorre correggerlo in tutti i punti in cui è stato incollato
 - Nel listato finale non è evidente che si sta riutilizzando la stessa funzionalità
 - Occorre disporre del codice originario
 - Occorre capire il codice originario
- Definizione di funzioni
 - Dichiarazione esplicita che una certa funzionalità viene utilizzata in più punti
 - Si separa la definizione della funzionalità rispetto al punto in cui questa viene utilizzata
 - La stessa funzione può essere usata più volte con parametri diversi

5

Come riusare il codice? (2/3)

- Copia-e-incolla
 - Semplice, ma poco efficace
 - Occorre adattare il codice incollato, ritoccando i nomi delle variabili e costanti utilizzati
 - Se si scopre un errore, occorre correggerlo in tutti i punti in cui è stato incollato
 - Nel listato finale non è evidente che si sta riutilizzando la stessa funzionalità
 - Occorre disporre del codice originario
 - Occorre capire il codice originario
- Definizione di funzioni
 - Dichiarazione esplicita che una certa funzionalità viene utilizzata in più punti
 - Si separa la definizione della funzionalità rispetto al punto in cui questa viene utilizzata
 - La stessa funzione può essere usata più volte con parametri diversi

6

Come riusare il codice? (3/3)

- Ogni miglioramento o correzione è automaticamente disponibile in tutti i punti in cui la funzione viene usata
- Nel listato finale è evidente che si sta riutilizzando la stessa funzionalità
- Non occorre disporre del codice originario
- Non occorre capire il codice originario

7

Principio di funzionamento (1/3)

```
int main(void)
{
    int x, y;

    /* leggi un numero
     * tra 50 e 100 e
     * memorizzalo
     * in x */
    /* leggi un numero
     * tra 1 e 10 e
     * memorizzalo
     * in y */

    printf("%d %d\n",
        x, y);
}
```

8

Principio di funzionamento (2/3)

```
int main(void)
{
    int x, y;
    x = leggi(50, 100);
    y = leggi(1, 10);
    printf("%d %d\n",
        x, y);
}
```

9

Principio di funzionamento (3/3)

```
int main(void)
{
    int x, y;
    x = leggi(50, 100);
    y = leggi(1, 10);
    printf("%d %d\n",
        x, y);
}
```

```
int leggi(int min,
          int max)
{
    int v;
    do {
        scanf("%d", &v);
    } while( v<min || v>max );
    return v;
}
```

10

Principio di funzionamento (3/3)

```
int main(void)
{
    int x, y;
    x = leggi(50, 100);
    y = leggi(1, 10);
    printf("%d %d\n",
        x, y);
}
```

```
min=50   max=100
int leggi(int min,
          int max)
{
    int v;
    do {
        scanf("%d", &v);
    } while( v<min || v>max );
    return v;
}
```

11

Principio di funzionamento (3/3)

```
int main(void)
{
    int x, y;
    x = leggi(50, 100);
    y = leggi(1, 10);
    printf("%d %d\n",
        x, y);
}
```

```
min=50   max=100
int leggi(int min,
          int max)
{
    int v;
    do {
        scanf("%d", &v);
    } while( v<min || v>max );
    return v;
}
```

Chiamante

Chiamato

12

Principio di funzionamento (3/3)

```
int main(void)
{
    int x, y ;
    x = leggi(50, 100) ;
    y = leggi(1, 10) ;
    printf("%d %d\n",
        x, y ) ;
}
```

```
min=1      max=10
int leggi(int min,
          int max)
{
    int v ;
    do {
        scanf("%d", &v) ;
    } while( v<min || v>max) ;
    return v ;
}
```

13

Sommario

- La definizione di una **funzione** delimita un frammento di codice riutilizzabile più volte
- La funzione può essere **chiamata** più volte
- Può ricevere dei **parametri** diversi in ogni chiamata
- Può restituire un **valore di ritorno** al **chiamante**
- Istruzione **return**

14

Miglioramento della funzione

```
int leggi(int min, int max)
{
    char riga[80] ;
    int v ;
    do {
        gets(riga) ;
        v = atoi(riga) ;
        if(v<min) printf("Piccolo: min %d\n", min) ;
        if(v>max) printf("Grande: max %d\n", max) ;
    } while( v<min || v>max) ;
    return v ;
}
```

15

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

#define MAXPAROLA 30
#define MAXRIGA 80

int main(int argc, char *argv[])
{
    int lungMAXPAROLA ; /* variabile di controllo
                           che indica la lunghezza massima */
    int lung, lungRiga ; /* lunghezza */
    int i, j, k ;

    for(i=0; i<MAXPAROLA; i++)
        lung=0 ;
    for(j=0; j<MAXRIGA; j++)
        lungRiga=0 ;
    if(argc > 1) {
        if(strcmp(argv[1], "-h") == 0) {
            printf("Uso: leggi [lungMAXPAROLA] [lungMAXRIGA]\n");
            exit(1);
        }
        if(argc > 2) {
            lungMAXPAROLA = atoi(argv[1]);
            lungMAXRIGA = atoi(argv[2]);
        }
        if(lungMAXPAROLA < 1) {
            printf("ERRORE: Impossibile aprire il file %s", argv[1]);
            exit(1);
        }
        if(lungMAXRIGA < 1) {
            printf("ERRORE: Impossibile aprire il file %s", argv[2]);
            exit(1);
        }
    }
    while(fgets(riga, MAXRIGA, f) != NULL)
```

Funzioni in C

Parametri formali e attuali

Parametri di una funzione

- Le funzioni possono ricevere dei parametri dal proprio chiamante
- Nella funzione:
 - **Parametri formali**
 - Nomi "interni" dei parametri

```
int leggi(int min,
          int max)
{
    ...
}
```

17

- Le funzioni possono ricevere dei parametri dal proprio chiamante
- Nella funzione:
 - **Parametri formali**
 - Nomi "interni" dei parametri
- Nel chiamante:
 - **Parametri attuali**
 - Valori effettivi (costanti, variabili, espressioni)

```
int leggi(int min,
          int max)
{
    ...
}
```

```
int main(void)
{
    ...
    y = leggi(1, 10) ;
    ...
}
```

18

Parametri formali (1/2)

- » Uno o più parametri
- » Tipo del parametro
 - Tipo scalare
 - Vettore o matrice
- » Nome del parametro

```
int leggi(int min,  
          int max)  
{  
    ...  
}
```

- » Nel caso in cui la funzione non abbia bisogno di parametri, si usa la parola chiave **void**

```
int stampa_menu(void)  
{  
    ...  
}
```

19

Parametri formali (2/2)

- » Per parametri vettoriali esistono 3 sintassi alternative
 - **int v[]**
 - **int v[MAX]**
 - **int *v**

- » Per parametri matriciali
 - **int m[RIGHE][COL]**
 - **int m[][COL]**

```
int leggi(int v[])  
{  
    ...  
}
```

```
int sudoku(int m[9][9])  
{  
    ...  
}
```

20

Avvertenza (1/2)

- » Il valore della dimensione del vettore (es. MAX)
 - Viene totalmente ignorato dal meccanismo di chiamata
 - Non sarebbe comunque disponibile alla funzione chiamata
 - Meglio per chiarezza ometterlo
 - Si suggerisce di passare un ulteriore parametro contenente l'occupazione del vettore

21

Avvertenza (2/2)

- » Nel caso di matrici
 - Il secondo parametro (es. COL) è obbligatorio e deve essere una costante
 - Il primo parametro viene ignorato e può essere omesso
 - Per matrici pluri-dimensional, occorre specificare tutti i parametri tranne il primo

22

Parametri attuali (1/2)

- » Uno o più valori, in esatta corrispondenza con i parametri formali
- » Tipi di dato compatibili con i parametri formali
- » È possibile usare
 - Costanti
 - Variabili
 - Espressioni

```
int main(void)  
{  
    y = leggi(1, 10);  
}
```

```
int main(void)  
{  
    y = leggi(a, b);  
}
```

```
int main(void)  
{  
    y = leggi(a+b+1,  
              c*c);  
}
```

22

Parametri attuali (2/2)

- » I nomi delle variabili usate non hanno alcuna relazione con i nomi dei parametri formali

- » Le parentesi sono sempre necessarie, anche se non vi sono parametri

```
int main(void)  
{  
    ...  
    y = leggi(a, b);  
    min=a  
    max=b  
}
```

```
int main(void)  
{  
    ...  
    y = stampa_menu();  
    ...  
}
```

24

Funzioni in C

Il valore di ritorno

Valore di ritorno

- Ogni funzione può ritornare un valore a proprio chiamante
 - Il tipo del valore di ritorno deve essere **scalare**
 - L'istruzione **return**
 - Termina l'esecuzione della funzione
 - Rende disponibile il valore al chiamante

```
int leggi(int min,
          int max)
{
    int v ;
    scanf("%d", &v) ;
    return v ;
}

int main(void)
{
    y = leggi(a, b) ;
}
```

(Kongrelerin 23. Eskişehir İl Genel Meclisi 2012-2013 Dönemine Ait İstihdam ve İşverenlik Raporu) 10

Tipo del valore di ritorno

- Valore scalare
 - `char`, `int` (o varianti), `float`, `double`
 - Tipi avanzati
 - Puntatori, `struct`
 - Nessun valore ritornato
 - `void`

```
double sqrt(double a  
{  
    ...  
}  
  
void stampa_err(int  
{  
    ...  
}
```

Kategorie: 33
Spendername: "Kinder- und Jugendärzte für Kinder & Jugendliche Berlin e.V."
Spender-ID: 10000000000000000000000000000000

L'istruzione return (1/2)

- Restituisce il valore
 - Costante
 - Variabile
 - Espressione
 - Il tipo deve essere compatibile con il tipo dichiarato per il valore di ritorno
 - Sintassi
 - `return x ;`
 - `return(x) ;`
 - L'esecuzione della funzione viene interrotta

L'istruzione return (2/2)

L'istruzione return (2/2)

- Per funzioni che non ritornano valori (void)
 - `return ;`
 - Il raggiungimento della fine del corpo della funzione } equivale ad un'istruzione `return` senza parametri
 - Permesso solo per funzioni void
 - Per funzioni non-void, è obbligatorio che la funzione ritorni **sempre** un valore

Nel chiamante...

Nel chiamante...

- Il chiamante può:
 - Ignorare il valore ritornato
 - `scanf("%d", &x) ;`
 - Memorizzarlo in una variabile
 - `y = sin(x) ;`
 - Utilizzarlo in un'espressione
 - `if (sqrt(x*x+y*y)>z) ...`

Convenzioni utili

- Le funzioni di tipo matematico ritornano sempre un valore `double`
- Le funzioni che non devono calcolare un valore (ma effettuare delle operazioni, per esempio) ritornano solitamente un valore `int`
 - Valore di ritorno == 0 ⇒ tutto ok
 - Valore di ritorno != 0 ⇒ si è verificato un errore
- Molte eccezioni importanti: `strcmp`, `scanf`, ...

31

Funzioni in C

Definizione e chiamata di funzioni

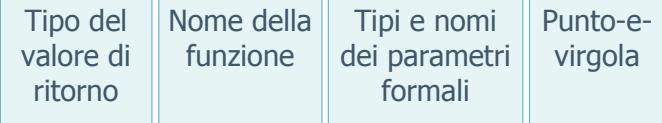
Sintassi C per le funzioni

- Il linguaggio C prevede 3 distinti momenti :
 - La dichiarazione (*prototipo o function prototype*)
 - L'interfaccia della funzione
 - Solitamente: prima del `main()`
 - La definizione
 - L'implementazione della funzione
 - Solitamente: al fondo del file, dopo il `main()`
 - La chiamata
 - L'utilizzo della funzione
 - Solitamente: dentro il corpo del `main()` o di altre funzioni

33

Dichiarazione o prototipo

```
int leggi(int min, int max);
```



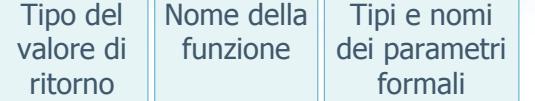
34

Scopo del prototipo

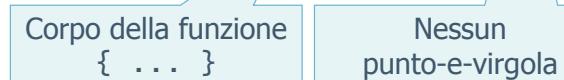
- Dichiare che esiste una funzione con il nome dato, e dichiarare i tipi dei parametri formali e del valore di ritorno
- Dal prototipo in avanti, si può chiamare tale funzione dal corpo di qualsiasi funzione (compreso il `main()`)
- Non è errore se la funzione non viene chiamata
- I file `.h` contengono centinaia di prototipi delle funzioni di libreria
- I prototipi sono posti solitamente ad inizio file

35

Definizione o implementazione



```
int leggi(int min, int max)
{
    ... codice della funzione ...
}
```

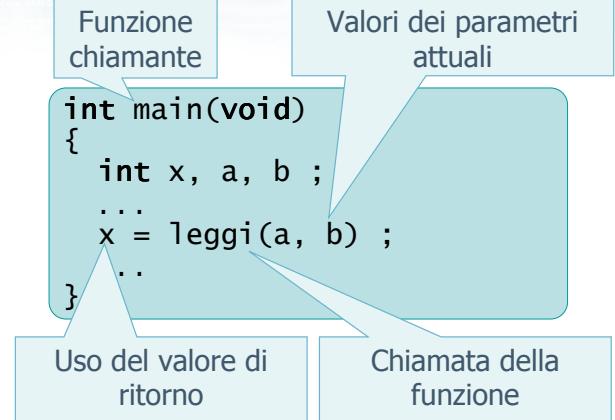


Scopo della definizione

- » Contiene il codice vero e proprio della funzione
- » Può essere posizionata ovunque nel file (al di fuori del corpo di altre funzioni)
- » Il nome della funzione ed i tipi dei parametri e del valore di ritorno devono coincidere con quanto dichiarato nel prototipo
- » Il corpo della funzione può essere arbitrariamente complesso, e si possono chiamare altre funzioni

37

Chiamata o invocazione



Meccanismo di chiamata

- » Le espressioni corrispondenti ai parametri attuali vengono valutate (e ne viene calcolato il valore numerico)
 - Compiano le variabili del chiamante
- » I valori dei parametri attuali vengono copiati nei parametri formali
- » La funzione viene eseguita
- » All'istruzione `return`, il flusso di esecuzione torna al chiamante
- » Il valore di ritorno viene usato o memorizzato

39

Riassumendo...

```
int leggi(int min, int max) ;  
  
int main(void)  
{  
    int x, a, b ;  
    ...  
    x = leggi(a, b) ;  
    ...  
}  
  
int leggi(int min, int max)  
{  
    ... codice della funzione ...  
}
```

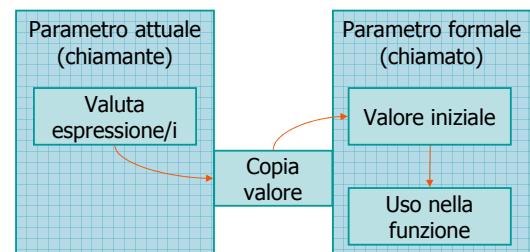
40

```
#include <stdio.h>  
#include <string.h>  
#include <ctype.h>  
  
#define MAXPAROLA 30  
#define MAXIGRA 80  
  
int main(argc, argv){  
    char *arg0; //  
    int freqMAXPAROLA; //  
    char frase[MAXIGRA]; // lunghezza delle parole  
    char parolaMAXIGRA; //  
    int i, inicio, lunghezza ;  
    inicio = 0;  
  
    for(i=0; i<MAXIGRA; i++)  
        frase[i] = '\0';  
  
    arg0 = argv[0];  
    freqMAXPAROLA = 0;  
    inicio = 0;  
    lunghezza = 0;  
    parolaMAXIGRA = 0;  
  
    while(1){  
        if(fgets(frase, MAXIGRA, stdin) == NULL){  
            printf("ERRORE: impossibile aprire il file %s", arg0);  
            exit(1);  
        }  
  
        if(frase[inicio] == '\0')  
            inicio = 0;  
        else{  
            if(frase[inicio] >='A' & frase[inicio] <='Z')  
                frase[inicio] = frase[inicio] - 'A' + 'a';  
            inicio++;  
        }  
        if(frase[inicio] == ' ')  
            inicio++;  
        else{  
            if(lunghezza > parolaMAXIGRA)  
                parolaMAXIGRA = lunghezza;  
            lunghezza++;  
        }  
    }  
    printf("Parole: %d\n", MAXIGRA - inicio);  
}
```

Funzioni in C

Passaggio dei parametri

- » Ogni volta che viene chiamata una funzione, avviene il trasferimento del valore corrente dei parametri attuali ai parametri formali



42

Caratteristiche

- Le variabili locali sono accessibili solo dall'interno della funzione
- Le variabili locali sono indipendenti da eventuali variabili di ugual nome definite nel `main`
 - In ogni caso, dal corpo della funzione è impossibile accedere alle variabili definite nel `main`
- Le variabili locali devono avere nomi diversi dai parametri formali

Istruzioni eseguibili

- Il corpo di una funzione può contenere qualsiasi combinazione di istruzioni eseguibili
- Ricordare l'istruzione `return`

```
int leggi(int min,
          int max)
{
    int v ;
    scanf("%d", &v) ;
    return v ;
}
```

```

#include <stdio.h>
#include <string.h>
#include <ctype.h>

#define MAXPAROLA 30
#define MAXSIGA 80

int main(int argc, char *argv[])
{
    int lungMAXPAROLA; /* valore di contenuto delle frequenze della parola */
    char lungMAXSIGA; /* valore di contenuto della lunghezza della parola */
    int i, indice, lungParola;
    FILE *fp;
    fp=fopen(argv[1], "r");
    if(fp==NULL)
    {
        printf("ERRORE, non è possibile aprire il file %s", argv[1]);
        exit(1);
    }
    fscanf(fp, "%d", &lungMAXPAROLA);
    fscanf(fp, "%d", &lungMAXSIGA);
    fscanf(fp, "%s", lungParola);
    if(lungParola!=0)
    {
        printf("ERRORE, impossibile aprire il file %s", argv[1]);
        exit(1);
    }
    while(fgets(lungParola, MAXSIGA, fp)!=NULL)
    {
        i=0;
        indice=0;
        lungParola[i]=0;
        while(lungParola[i]!='\n')
        {
            if(isalpha(lungParola[i]))
            {
                lungParola[i]=tolower(lungParola[i]);
                lungParola[i]=lungParola[i]-97;
                lungParola[i]=lungParola[i]*100;
                lungParola[i]=lungParola[i]+97;
            }
            else
            {
                lungParola[i]=0;
                break;
            }
            i++;
        }
        if(indice<lungMAXSIGA)
        {
            lungParola[indice]=lungParola[i];
            indice++;
        }
        else
        {
            printf("ERRORE, impossibile aprire il file %s", argv[1]);
            exit(1);
        }
    }
    fclose(fp);
}

```

Funzioni

Parametri “by reference”

```

#include <stdio.h>
#include <string.h>
#include <ctype.h>

#define MAXPAROLA 30
#define MAXSIGA 80

int main(int argc, char *argv[])
{
    int lungMAXPAROLA; /* valore di contenuto delle frequenze della parola */
    char lungMAXSIGA; /* valore di contenuto della lunghezza della parola */
    int i, indice, lungParola;
    FILE *fp;
    fp=fopen(argv[1], "r");
    if(fp==NULL)
    {
        printf("ERRORE, non è possibile aprire il file %s", argv[1]);
        exit(1);
    }
    fscanf(fp, "%d", &lungMAXPAROLA);
    fscanf(fp, "%d", &lungMAXSIGA);
    fscanf(fp, "%s", lungParola);
    if(lungParola!=0)
    {
        printf("ERRORE, impossibile aprire il file %s", argv[1]);
        exit(1);
    }
    while(fgets(lungParola, MAXSIGA, fp)!=NULL)
    {
        i=0;
        indice=0;
        lungParola[i]=0;
        while(lungParola[i]!='\n')
        {
            if(isalpha(lungParola[i]))
            {
                lungParola[i]=tolower(lungParola[i]);
                lungParola[i]=lungParola[i]-97;
                lungParola[i]=lungParola[i]*100;
                lungParola[i]=lungParola[i]+97;
            }
            else
            {
                lungParola[i]=0;
                break;
            }
            i++;
        }
        if(indice<lungMAXSIGA)
        {
            lungParola[indice]=lungParola[i];
            indice++;
        }
        else
        {
            printf("ERRORE, impossibile aprire il file %s", argv[1]);
            exit(1);
        }
    }
    fclose(fp);
}

```

Parametri “by reference”

Introduzione

- Il passaggio “by value” risulta inefficiente qualora le quantità di dati da passare fossero notevoli
 - Nel caso del passaggio di vettori o matrici, il linguaggio C non permette il passaggio “by value”, ma copia solamente l’indirizzo di partenza
 - Esempio: `strcmp`
- Talvolta è necessario o utile poter modificare il valore di una variabile nel chiamante
 - Occorre adottare un meccanismo per permettere tale modifica
 - Esempio: `scanf`

Problemi

Parametri “by reference”

- Introduzione
- Operatori & e *
- Passaggio “by reference”
- Passaggio di vettori
- Esercizio “strcpy”

Passaggio dei parametri

- Il linguaggio C prevede il passaggio di parametri “by value”
 - Il chiamato non può modificare le variabili del chiamante
 - Il parametro formale viene inizializzato con una copia del valore del parametro attuale

Soluzione

- La soluzione ad entrambi i problemi è la stessa:
 - Nel passaggio di vettori, ciò che viene passato è solamente l’indirizzo
 - Per permettere di modificare una variabile, se ne passa l’indirizzo, in modo che il chiamato possa modificare direttamente il suo contenuto in memoria
- Viene detto passaggio “by reference” dei parametri
 - Definizione impropria, in quanto gli indirizzi sono, a loro volta, passati “by value”

```
#include <stdio.h>
#include <ctype.h>

#define MAXFAROLA 30
#define MAXIGA 80

int main(int argc, char *argv[])
{
    int freqMAXFAROLA; /* valori di conversione delle frequenze delle lunghezze della parola */
    int i, media, lunghezza;
    FILE *fptr;

    fptr=fopen(argv[1], "r");
    if(fptr==NULL)
    {
        printf("Errore: non è possibile aprire il file %s", argv[1]);
        exit(1);
    }

    fscanf(fptr, "%d", &freqMAXFAROLA);
    fscanf(fptr, "%d", &media);
    fscanf(fptr, "%d", &lunghezza);

    printf("Media: %d\n", media);
    printf("Lunghezza: %d\n", lunghezza);
    printf("Frequenza massima: %d\n", freqMAXFAROLA);
    printf("Argomento: %s\n", argv[1]);
}
```

Parametri "by reference"

Operatori & e *

- L'operatore **indirizzo-di** restituisce l'indirizzo di memoria della variabile a cui viene applicato
 - `&a` è l'indirizzo 1012
 - `&b` è l'indirizzo 1020

int a	→ 1012	37
	1000	
	1004	
	1008	
	1016	
int b	→ 1020	-4
	1020	
	1024	
	1028	

9

- Per gestire il passaggio "by reference" dei parametri occorre
 - Conoscere l'indirizzo di memoria di una variabile
 - Operatore &
 - Accedere al contenuto di una variabile di cui si conosce l'indirizzo ma non il nome
 - Operatore *
- Prime nozioni della aritmetica degli indirizzi, che verrà approfondita in Unità successive

8

Osservazioni

- L'indirizzo di una variabile viene deciso dal compilatore
- L'operatore & si può applicare solo a variabili singole, non ad espressioni
 - Non ha senso `&(a+b)`
 - Non ha senso `&(3)`
- Conoscere l'indirizzo di una variabile permette di leggerne o modificarne il valore senza conoscerne il nome

10

Variabili "puntatore"

- Per memorizzare gli indirizzi di memoria, occorre definire opportune variabili di tipo "indirizzo di..."
- Nel linguaggio C si chiamano **puntatori**
- Un puntatore si definisce con il simbolo *
 - `int *p ; /* puntatore ad un valore intero */`
 - `float *q ; /* puntatore ad un valore reale */`

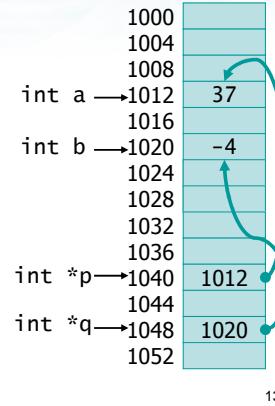
11

- int main(void)
 {
 int a, b ;
 int *p, *q ;
 a = 37 ;
 b = -4 ;
 p = &a ;
 /* p "punta a" a */
 q = &b ;
 /* q "punta a" b */
 }
- | | | |
|--------|--------|------|
| int a | → 1012 | 37 |
| | 1000 | |
| | 1004 | |
| | 1008 | |
| int b | → 1020 | -4 |
| | 1016 | |
| | 1020 | |
| | 1024 | |
| | 1028 | |
| | 1032 | |
| | 1036 | |
| int *p | → 1040 | 1012 |
| | 1044 | |
| int *q | → 1048 | 1020 |
| | 1052 | |

12

Operatore

- ▶ L'operatore di **accesso indiretto** permette di accedere, in lettura o scrittura, al valore di una variabile di cui si conosce l'indirizzo
 - `*p` equivale ad `a`
 - `*q` equivale a `b`
 - `*p = 0 ;`
 - `if(*q > 0) ...`



1

Costrutti frequenti

Costrutto	Significato
int x ;	x è una variabile intera
int *p ;	p è un puntatore a variabili intere
p = &x ;	p punta ad x
*p = 0 ;	Azzera la variabile puntata da p (cioè x)
b = *p ;	Leggi il contenuto della variabile puntata da p e copialo in b

```

#include <stdio.h>
#include <string.h>
#include <ctype.h>

#define MAXPAROLA 30
#define MAXIGRA 80

int main(int argc, char *argv[])
{
    int kmp[MAXPAROLA]; // vettore di corrispondenze
    char parola[MAXIGRA];
    int i, nero, lunghezza;
    FILE *f;

    for(i=0; i<MAXPAROLA; i++)
        kmp[i] = -1;

    if(argc != 2)
    {
        printf("Usage: %s \n", argv[0]);
        exit(1);
    }

    f = fopen(argv[1], "r");
    if(f == NULL)
    {
        printf("Error: file %s non esiste\n", argv[1]);
        exit(1);
    }

    while(fgets(parola, MAXIGRA, f) != NULL)
    {
        lunghezza = strlen(parola);
        if(lunghezza > MAXIGRA)
            lunghezza = MAXIGRA;
        if(parola[lunghezza-1] == '\n')
            parola[lunghezza-1] = '\0';
        if(parola[lunghezza-1] == '\r')
            parola[lunghezza-1] = '\0';

        if(kmp[0] == -1)
        {
            for(i=0; i<lunghezza; i++)
                if(parola[i] != ' ')
                    kmp[i] = i;
        }
        else
        {
            i = kmp[0];
            if(parola[i] == ' ')
                i++;
            if(parola[i] == '\0')
                break;
            if(parola[i] != parola[0])
                i = -1;
            else
                i = kmp[i];
        }
        if(i != -1)
            printf("%s\n", parola);
    }
}


```

Parametri “by reference”

Passaggio "by reference"

- Obiettivo: passare ad una funzione una variabile, in modo tale che la funzione la possa modificare
 - Soluzione:
 - Definire un parametro attuale di tipo puntatore
 - Al momento della chiamata, passare l'indirizzo della variabile (anziché il suo valore)
 - All'interno del corpo della funzione, fare sempre accesso indiretto alla variabile di cui è noto l'indirizzo

16

Esempio: "Azzera"

- ▶ Scrivere una funzione `azzeri`, che riceve un parametro di tipo intero (by reference) e che azzeri il valore di tale parametro

1

Soluzione

```
void azzera( int *v ) ;
```

```
int main( void )
{
    int x ;
    ...
    azzer(&x) ;
    ...
}
```

```
void azzera( int *v )
{
    *v = 0 ;
}
```

18

Esempio: "Scambia"

- Scrivere una funzione `scambia`, che riceve due parametri di tipo intero (by reference) e che scambia tra di loro i valori in essi contenuti

19

Soluzione

```
void scambia( int *p, int *q ) ;  
  
int main( void )  
{  
    int a,b ;  
    ...  
    scambia(&a, &b) ;  
    ...  
}  
  
void scambia( int *p, int *q )  
{  
    int t ;  
    t = *p ;  
    *p = *q ;  
    *q = t ;  
}
```

20

Osservazione

- Il meccanismo di passaggio by reference spiega (finalmente!) il motivo per cui nella funzione `scanf` è necessario specificare il carattere `&` nelle variabili lette
- Le variabili vengono passate by reference alla funzione `scanf`, in modo che questa possa scrivervi dentro il valore immesso dall'utente

21

```
#include <stdio.h>  
#include <string.h>  
#include <ctype.h>  
  
#define MAXPAROLA 30  
#define MAXIGRA 80  
  
int main(int argc, char *argv[]){  
    int lungMAXPAROLA; /* variezza di parola */  
    char riga[MAXIGRA]; /* riga inserita */  
    char *p, *q, *t;  
    int i, j, lungmax;  
    i=0;  
  
    for(i=0; i<MAXPAROLA; i++)  
        riga[i]=0;  
  
    if(argc > 1){  
        /* se c'è un argomento, leggono la parola */  
        p=argv[1];  
        if(*p=='\0')  
            lungMAXPAROLA=0;  
        else{  
            /* legge la parola */  
            while(*p!=0){  
                if(isalpha(*p))  
                    lungMAXPAROLA++;  
                p++;  
            }  
            /* salta la parola */  
            p+=lungMAXPAROLA+1;  
        }  
    }  
  
    /* legge la riga */  
    q=p; /* punta alla fine della parola */  
    if(lungMAXPAROLA==0)  
        q=riga; /* se non c'è parola, punta alla riga */  
    else  
        q+=lungMAXPAROLA;  
  
    /* legge la riga */  
    while(*q!=0){  
        if(isalpha(*q))  
            lungmax++;  
        q++;  
    }  
    /* salta la riga */  
    q+=lungmax+1;  
    /* stampa la riga */  
    if(lungmax>0)  
        printf("Riga: %s\n", riga);  
    else  
        printf("Riga: %s\n", riga+lungMAXPAROLA);  
    /* salta la riga */  
    q+=lungmax+1;  
    /* stampa la parola */  
    if(lungMAXPAROLA>0)  
        printf("Parola: %s\n", riga);  
    else  
        printf("Parola: %s\n", riga+lungMAXPAROLA);  
    /* salta la parola */  
    q+=lungMAXPAROLA+1;  
    /* stampa la riga */  
    if(lungmax>0)  
        printf("Riga: %s\n", riga);  
    else  
        printf("Riga: %s\n", riga+lungMAXPAROLA);  
}
```

Parametri "by reference"

Passaggio di vettori

Passaggio di vettori e matrici

- Nel linguaggio C, il nome di un array (vettore o matrice) è automaticamente sinonimo del puntatore al suo primo elemento

```
int main(void)  
{  
    int v[10] ;  
    int *p ;  
    p = & v[0] ;  
}
```

p e v sono del tutto equivalenti

23

Conseguenze

- Quando il parametro di una funzione è di tipo array (vettore o matrice)
 - L'array viene passato direttamente "by reference"
 - Non è necessario l'operatore `&` per determinare l'indirizzo
 - È sufficiente il nome del vettore
 - Non è necessario l'operatore `*` per accedere al contenuto
 - È sufficiente l'operatore di indicizzazione `[]`
 - Non è possibile, neppure volendolo, passare un array "by value"

24

Esercizio "Duplicati"

- Scrivere una funzione che, ricevendo due parametri
 - Un vettore di double
 - Un intero che indica l'occupazione effettiva di tale vettorepossa determinare se vi siano valori duplicati in tale vettore
- La funzione ritornerà un intero pari a 1 nel caso in cui vi siano duplicati, pari a 0 nel caso in cui non ve ne siano

25

Soluzione (1/3)

```
int duplicati(double v[], int N) ;  
/*  
Riceve in ingresso il vettore v[] di double  
che contiente N elementi (da v[0] a v[N-1])  
  
Restituisce 0 se in v[] non vi sono duplicati  
Restituisce 1 se in v[] vi sono duplicati  
  
Il vettore v[] non viene modificato  
*/
```

26

Soluzione (2/3)

```
int duplicati(double v[], int N)  
{  
    int i, j ;  
  
    for(i=0; i<N; i++)  
    {  
        for(j=i+1; j<N; j++)  
        {  
            if(v[i]==v[j])  
                return 1 ;  
        }  
    }  
    return 0 ;  
}
```

27

Soluzione (3/3)

```
int main(void)  
{  
    const int MAX = 100 ;  
    double dati[MAX] ;  
    int Ndati ;  
    int dupl ;  
  
    ...  
    dupl = duplicati(dati, Ndati) ;  
    ...  
}
```

28



Errore frequente

- Nel passaggio di un vettore occorre indicarne solo il nome

```
dupl = duplicati(dati, Ndati) ;
```

~~```
dupl = duplicati(dati[], Ndati) ;
dupl = duplicati(dati[MAX], Ndati) ;
dupl = duplicati(dati[Ndati], Ndati) ;
dupl = duplicati(&dati, Ndati) ;
```~~

29

## Osservazione

- Nel caso dei vettori, il linguaggio C permette solamente il passaggio by reference
  - Ciò significa che il chiamato ha la possibilità di modificare il contenuto del vettore
- Non è detto che il chiamato effettivamente ne modifichi il contenuto
  - La funzione duplicati analizza il vettore senza modificarlo
  - Esplicitarlo sempre nei commenti di documentazione

30

```
#include <stdio.h>
#include <ctype.h>

#define MAXFAROLA 30
#define MAXIGA 60

int main(int argc, char *argv[])
{
 int freqMAXFAROLA; /* vettore di contenuto delle frequenze delle lunghezze delle parole */
 int i, indice, lunghezza;
 FILE *fp;

 fp=fopen(argv[1], "r");
 if(fp==NULL)
 {
 fprintf(stderr, "ERRORE: non è possibile aprire il file %s", argv[1]);
 exit(1);
 }

 for(i=0; i<MAXFAROLA; i++)
 freqMAXFAROLA[i]=0;

 while(fgets(argv[2], MAXIGA, fp) != NULL)
 {
 indice=0;
 lunghezza=0;
 for(i=0; argv[2][i] != '\0'; i++)
 {
 if(isalpha(argv[2][i])) /* se è una lettera */
 indice++;
 else
 break;
 }
 if(indice > MAXFAROLA)
 indice=MAXFAROLA;
 freqMAXFAROLA[indice]++;
 lunghezza=i;
 }

 fclose(fp);
}
```

## Parametri "by reference"

### Esercizio "strcpy"

32

- Si implementi, sotto forma di funzione, la ben nota funzione di libreria **strcpy** per la copia di due stringhe

### Soluzione (1/2)

```
void strcpy(char *dst, char *src) ;
/*
Copia il contenuto della stringa src
nella stringa dst

Assume che src sia 0-terminata, e restituisce
dst in forma 0-terminata.

Assume che nella stringa dst vi sia spazio
sufficiente per la copia.

La stringa src non viene modificata.
*/
```

33

### Soluzione (2/2)

```
void strcpy(char *dst, char *src)
{
 int i ;
 for(i=0; src[i]!=0; i++)
 {
 dst[i] = src[i] ;
 }
 dst[i] = 0 ;
}
```

34

### Osservazione

- La funzione può essere dichiarata in due modi:
  - void strcpy(char \*dst, char \*src)**
  - void strcpy(char dst[], char src[])**
- Sono forme assolutamente equivalenti
- Tutte le funzioni di libreria che lavorano sulle stringhe accettano dei parametri di tipo **char \***

35

```

#ifndef _SHELL_H_
#define _SHELL_H_
#include <sys/types.h>
#include <sys/conf.h>
#include <sys/conf.h>
#include <sys/conf.h>
#include <sys/conf.h>

#define MAXPARMA 20
#define MAXBIG 20

int main(int argc, char *argv[])
{
 int len=MAXPARMA; /* > vector */
 char big[MAXBIG]; /* big string */
 char *bigstr=NULL;
 int i, ratio, length;
 FILE *f;
 if(argc<2)
 return 1;
 if((f=fopen(argv[1], "r"))==NULL)
 return 1;
 if(fscanf(f, "%d %d", &i, &ratio)==2)
 {
 if(i>len)
 i=len;
 if(ratio>100)
 ratio=100;
 if(ratio<1)
 ratio=1;
 if((bigstr=(char*)malloc(i))!=NULL)
 {
 for(j=0;j

```

## Funzioni

## La funzione main()

```

#include <stdio.h>
#include <string.h>
#include <ctype.h>

#define MAXPAROLA 30
#define MAXGIGA 80

int main(int argc, char *argv[])
{
 int i;
 long long int maxGiga = 0;
 char str[MAXGIGA];
 int len, length;
 int n = 0;

 for(i=0; i<MAXPAROLA; i++)
 str[i] = '0';

 if(argc > 1)
 {
 for(i=0; i<argc-1; i++)
 {
 if(strcmp(argv[i], "-n") == 0)
 n = 1;
 else
 length = strlen(argv[i]);
 }
 if(n == 1)
 length = 0;
 if(length > MAXGIGA)
 length = MAXGIGA;
 if(length > 0)
 strcpy(str, argv[argc-1]);
 else
 str[0] = '\0';
 maxGiga = strtoll(str, NULL, 10);
 }
 else
 maxGiga = strtoll("0", NULL, 10);

 printf("%ld\n", maxGiga);
}

```

## La funzione main()

#### **Interfaccia con il sistema operativo**

Korgi (n. 2) spiegherò, "KORG, non ti preoccupare con il nome del Ma-

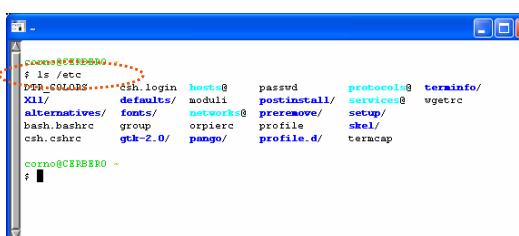
## La funzione main()

- Interfaccia con il sistema operativo
  - Argomenti sulla linea di comando
  - Parametri argc e argv
  - Valore di ritorno del programma
  - La funzione exit
  - Esercizio “Calcolatrice”

2

## Interfaccia in modalità “console”

- ▶ Ricordiamo che il linguaggio C si è evoluto con interfacce “a caratteri”
  - ▶ L’attivazione di un programma avviene digitandone il nome in una finestra di comando



1

l'origine [n. 2] {  
↳ spazio di dati, "EXCEL serve un percorso di calcolo con il nome del file  
word11".  
}

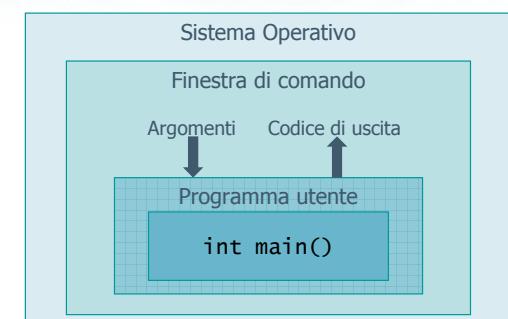
## La funzione main()

- La funzione `main()`, presente in tutti i programmi C, è una funzione come tutte le altre
  - Unica particolarità: viene chiamata automaticamente dal Sistema Operativo, appena il programma viene avviato
    - Non esiste mai una chiamata esplicita a `main()`
    - L'interfaccia della funzione viene definita dalle caratteristiche del sistema operativo

4

**Il modello “console”**

## Il modello “console”



6

## Interfaccia del programma

- La finestra di comando permette di passare al programma una serie di **argomenti**
  - Zero, una o più stringhe di testo
  - Utilizzate dal programma come dati in ingresso
- Al termine dell'esecuzione, il programma restituisce un **codice di uscita**
  - Numero intero
  - Indica eventuali condizioni di errore
- Durante l'esecuzione, il programma può accedere all'input (tastiera) e all'output (schermo)

7

## La funzione main()

### Argomenti sulla linea di comando

## La linea di comando (1/2)

- L'attivazione di un programma avviene digitando il suo nome in una finestra di comando

```
Microsoft Windows XP [Versione 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\corno>cd \progr
C:\progr>quadrato

***_
***_
***_
C:\progr>_
```

9

## La linea di comando (2/2)

- È possibile passare dei parametri all'attivazione del programma

```
C:\progr>quadrato 5

***_
***_
***_
C:\progr>_
```

```
C:\progr>quadrato 5 K
KKKKK
KKKKK
KKKKK
KKKKK
C:\progr>_
```

## Nomenclatura

- Parametri sulla linea di comando
  - **Command line parameters**
- Argomenti del programma
  - **Program arguments**
- Parametri di attivazione
- Parametri del main
- Argomenti del main
- Opzioni [sulla linea di comando]

11

## Caratteristiche

- Numero variabile di parametri
  - Anche nessuno
- Tipo variabile
  - Numeri
  - Caratteri o Stringhe
- Il chiamante (sistema operativo) non ha modo di sapere quanti parametri servono al programma né di che tipo
  - Verranno trattati in modo standardizzato

12

## Standardizzazione dei parametri

- Gli argomenti sulla linea di comando vengono trattati come un **vettore di stringhe**
  - Il programma riceve
    - Una copia del vettore di stringhe
    - Un valore numerico che indica quante stringhe sono presenti

1

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

#define MAXPAROLA 30
#define MAXBROGA 80

int main(int argc, char *argv[])
{
 int lungj(MAXPAROLA); /* valore di controllo
 per la lunghezza della parola */
 int lungb(MAXBROGA); /* valore di controllo
 per la lunghezza della broga */
 char digitj[MAXPAROLA];
 char digitb[MAXBROGA];
 int i=0, j=0;

 fgets(digitj, MAXPAROLA, stdin);
 lungj = strlen(digitj);

 fgets(digitb, MAXBROGA, stdin);
 lungb = strlen(digitb);

 if(lungj < 1) {
 fprintf(stderr, "ERRORE: Impossibile avere una parola vuota\n");
 exit(1);
 }

 if(lungb < 1) {
 fprintf(stderr, "ERRORE: Impossibile avere una broga vuota\n");
 exit(1);
 }

 while (digitj[i] >='0' & digitj[i] <='9' & digitb[j] >='0' & digitb[j] <='9') {
 if(digitj[i] > digitb[j]) {
 printf("Digitj > digitb\n");
 }
 else if(digitj[i] < digitb[j]) {
 printf("Digitj < digitb\n");
 }
 else {
 printf("Digitj = digitb\n");
 }
 i++;
 j++;
 }
}
```

## La funzione main()

## Parametri argc e argv

1

## Esempi

C:\progr>quadrato

- ## ➤ Numero argomenti = 0

C:\progr>quadrato 5

- Numero argomenti = 1
  - Argomento 1 = "5"

C:\progr>quadrato 5 K

- » Numero argomenti = 2
  - » Argomento 1 = "5"
  - » Argomento 2 = "K"

14

## Parametri formali del main

- ▶ I parametri sulla linea di comando sono disponibili al `main` attraverso i suoi parametri formali
  - ▶ La definizione completa della funzione `main` è:

```
int main(int argc, char *argv[]) ;
```

| Argument count | Argument values |
|----------------|-----------------|
|----------------|-----------------|

## Argument count

## Argument values

16

## Il parametro argo

- ▶ `int argc`
  - ▶ Numero di parametri sulla linea di comando
    - Incrementato di uno, in quanto il nome del programma viene considerato come un parametro
  - ▶ Se non vi sono argomenti effettivi, vale 1
  - ▶ Se vi sono k argomenti effettivi, vale k+1

1

## Il parametro argv

- ▶ `char *argv[]`
  - ▶ È un vettore di stringhe
  - ▶ Ogni stringa è un parametro del programma
  - ▶ Vi sono `argc` diverse stringhe
  - ▶ La prima stringa, `argv[0]`, è il nome del programma
  - ▶ La seconda stringa, `argv[1]`, è il primo argomento (se esiste)
  - ▶ ...
  - ▶ L'ultimo argomento è in `argv[argc-1]`

18

## Esempi

c:\progr>**quadrato**

- » argc==1
- » argv[0]=="quadrato"

c:\progr>**quadrato 5**

- » argc==2
- » argv[0]=="quadrato"
- » argv[1]=="5"

c:\progr>**quadrato 5 K**

- » argc==3
- » argv[0]=="quadrato"
- » argv[1]=="5"
- » argv[2]=="K"

19

## Per capire meglio...

```
#include <stdio.h>

int main(int argc, char *argv[])
{
 int i ;

 printf("argc = %d\n", argc) ;
 for(i=0; i<argc; i++)
 {
 printf("argv[%d] = \"%s\"\n",
 i, argv[i]) ;
 }
}
```



20

## Osservazione

- » Il vettore argv contiene i dati sotto forma esclusivamente di stringa
- » Qualora uno dei dati richiesti sia di tipo numerico (int o double), occorre effettuare la conversione da stringa a numero
  - i = atoi(argv[1]) ;
  - r = atof(argv[1]) ;
- » Se il parametro è invece una stringa, conviene copiarlo in una variabile mediante strcpy

21

```
#include <stdlib.h>
#include <string.h>
#include <ctype.h>

#define MAXPAROLA 30
#define MAXIGA 80

int main(int argc, char *argv[])
{
 int lungMAXPAROLA; /* vettore di controllo
 che indica se ogni parola inserita è
 una parola corretta */
 char s[MAXIGA]; /* stringa che contiene la parola inserita */
 int i, j, lungMAX; /* i è l'indice della parola inserita
 j è l'indice del carattere
 lungMAX è la lunghezza massima
 di una parola inserita */

 for(i=0; i<MAXPAROLA; i++)
 lungMAX++; /* lungMAX è la lunghezza massima
 di una parola inserita */

 if(argc > 1)
 {
 /* legge la parola inserita */
 strcpy(s, argv[1]);
 /* controlla se la parola inserita è una parola corretta */
 lungMAX = lungMAX - strlen(argv[1]);
 if(strlen(argv[1]) > lungMAX)
 printf("ERRORE: Impossibile aprire il file \"%s\", argv[1].\n", argv[1]);
 else
 /* legge la parola inserita */
 if(strlen(argv[1]) == lungMAX)
 /* se la parola inserita è una parola corretta */
 if(strcmp(argv[1], s) == 0)
 printf("Parola inserita corretta.\n");
 else
 printf("Parola inserita errata.\n");
 else
 printf("Parola inserita errata.\n");
 }
 while(fgets(s, MAXIGA, f) != NULL)
 /* legge la parola inserita */
 if(strlen(s) == lungMAX)
 /* se la parola inserita è una parola corretta */
 if(strcmp(s, argv[1]) == 0)
 printf("Parola inserita corretta.\n");
 else
 printf("Parola inserita errata.\n");
 else
 printf("Parola inserita errata.\n");
}
```

## La funzione main()

## Valore di ritorno del programma

## Valore di ritorno

- » Al termine dell'elaborazione il programma restituisce un numero intero al sistema operativo
- » Tale valore viene spesso ignorato, ma in caso di esecuzione "batch" è possibile interrogarlo a livello di sistema operativo
  - in MS-DOS, tramite la variabile ERRORLEVEL
    - echo %errorlevel%
  - in sistemi Unix, mediante la macro \$?
    - echo \$?

23

## Convenzioni

- » Il valore di ritorno è un int, ma per compatibilità si preferisce ritornare degli "interi positivi piccoli"
- » Convenzionalmente
  - Il valore di ritorno pari a 0 indica "programma terminato correttamente"
  - Il valore di ritorno diverso da 0 indica "programma terminato anormalmente a causa di un errore"
    - Il valore specifico ritornato (1, 2, 3, ...) può indicare la causa dell'errore

24

## Esempio

```
corn0n@CERBERO ~
$ grep corno /etc/passwd
corno:unused_by_nt/2000/xp:1004:513:corno,U-CERBERO/corno,S-1-5-21-4189850523-2
89800181-2619313026-1004:/home/corno/bin/bash

corn0n@CERBERO ~
$ echo ?
0

corn0n@CERBERO ~
$ grep cornozzz /etc/passwd

corn0n@CERBERO ~
$ echo ?
1

corn0n@CERBERO ~
$ grep corno /etc/passwdzzz
grep: /etc/passwdzzz: No such file or directory

corn0n@CERBERO ~
$ echo ?
2

corn0n@CERBERO ~
$ [
```

1

## La funzione main()

## La funzione exit

## **Restituzione del valore di ritorno**

- Quando il programma termina, deve restituire il valore di ritorno
    - ==0, se tutto OK
    - !=0, se errore
  - Il modo più semplice per restituire tale valore è di utilizzare l'istruzione `return` all'interno della funzione `main`
    - L'elaborazione viene immediatamente interrotta
    - Il valore ritornato viene passato al sistema operativo

1

```
#include <stdio.h>

int main(int argc, char *argv[])
{
 . . .
 . . .
 . . .
 return 0 ;
}
```

28

## La funzione exit

- Esiste inoltre la funzione di libreria `exit`, dichiarata in `<stdlib.h>`, che assolve alla stessa funzione
    - Interrompe l'esecuzione del programma
    - Ritorna il valore specificato
  - Il vantaggio rispetto all'istruzione `return` è che può essere usata all'interno di qualsiasi funzione, non solo del `main`

```
void exit(int value)
```

1

```
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[])
{
 . . .
 . . .
 . . .
 exit(0) ;
}
```

30



## Suggerimento

- » Ricordare sempre di ritornare un valore
- » Mettere come ultima istruzione del main: `exit(0);`
- » Per eventuali condizioni di errore (parametri assenti, valori illegali, ...) che non possono essere corrette dal programma, restituire un valore positivo: `exit(1)` ;
  - Tali errori possono essere controllati dall'interno di qualsiasi funzione: la `exit` interrompe comunque l'intero programma

31

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define MAXPAROLA 30
#define MAXSIG 60

int main(int argc, char *argv[])
{
 int lung(MAXPAROLA); /* valore di controllo
 della frequenza delle lunghezze delle parole */
 int i, j, MAXLNG;
 int l, inf, lungMAX;
 int n = 0;
 char op[20];
 FILE *fptr;
 fptr = fopen("TEST100", "r");
 if (fptr == NULL)
 {
 printf("Error opening file\n");
 exit(1);
 }
 fscanf(fptr, "%s", op);
 lungMAX = strlen(op);
 lung = lungMAX;
 MAXLNG = lungMAX;
 for (i = 1; i < argc; i++)
 {
 if (lung(MAXPAROLA) > lung)
 {
 lung = lung(MAXPAROLA);
 lungMAX = lung;
 MAXLNG = lungMAX;
 }
 else if (lung(MAXPAROLA) < lung)
 {
 lung = lung(MAXPAROLA);
 lungMAX = lung;
 MAXLNG = lungMAX;
 }
 else if (lung(MAXPAROLA) == lung)
 {
 lung = lung(MAXPAROLA);
 lungMAX = lung;
 MAXLNG = lungMAX;
 }
 }
 printf("lung: %d\n", lung);
 printf("lungMAX: %d\n", lungMAX);
 printf("MAXLNG: %d\n", MAXLNG);
 printf("lung: %d\n", lung);
 printf("lungMAX: %d\n", lungMAX);
 printf("MAXLNG: %d\n", MAXLNG);
}
```

## La funzione main()

### Esercizio “Calcolatrice”

### Esercizio “Calcolatrice”

- » Si scriva un programma da utilizzarsi come semplice calcolatrice sulla linea di comando
- » Il programma, denominato `calcola`, accetta 3 parametri sulla linea di comando
  - Il primo ed il terzo parametro sono degli operandi, espressi come numeri reali
  - Il secondo parametro è un operatore, scelto tra `+`, `-`, `*` e `/`
- » Il programma stampa il risultato corrispondente all'operazione

33

```
c:\>calcola 3 + 2
Risultato: 5.0000
c:\>calcola 3 * 2
Risultato: 6.0000
c:\>calcola 3 $ 2
Errore: operatore non riconosciuto
c:\>calcola 3 +
Errore: operando mancante
```

### Analisi

### Soluzione (1/5)

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main(int argc, char *argv[])
{
 double v1, v2 ;
 char op[20] ;
```

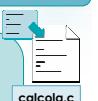


calcola.c

35

```
if(argc!=4)
{
 printf("Errore: numero parametri
 insufficiente\n");
 exit(1) ;
}

v1 = atof(argv[1]) ;
strcpy(op, argv[2]) ;
v2 = atof(argv[3]) ;
```



calcola.c

36

### Soluzione (2/5)

## Soluzione (3/5)

```
if(strcmp(op, "+")==0)
 printf("Risultato: %f\n", v1 + v2) ;
else if(strcmp(op, "-")==0)
 printf("Risultato: %f\n", v1 - v2) ;
else if(strcmp(op, "*")==0)
 printf("Risultato: %f\n", v1 * v2) ;
```

calcola.c

37

## Soluzione (4/5)

```
else if(strcmp(op, "/")==0)
{
 if(v2==0)
 {
 printf("Errore: divisione per zero\n");
 exit(2) ;
 }
 else
 printf("Risultato: %f\n", v1 / v2) ;
```

calcola.c

38

## Soluzione (5/5)

```
else
{
 printf("Errore: operatore
 non riconosciuto\n") ;
 exit(3) ;
}
exit(0) ;
```

calcola.c

39

## Esercizi proposti

- Esercizio "Confronto tra date"
- Esercizio "Quadrato"
- Esercizio "Indovina numero"

## Funzioni

### Esercizi proposti

2

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>
```

```
#define MAXPAROLA 30
#define MAXRIGA 80
```

```
int main(int argc, char *argv[])
{
 int lungMAXPAROLA; /* vettore di dimensione
 delle frequenze delle lunghezze delle parole */
 int lungMAXRIGA; /* vettore di dimensione delle
 righe del file */
 int i, indice, lungParola;
```

```
char lungMAXRIGA[];
char lungMAXPAROLA[];
```

```
for(i=0; i<lungMAXRIGA; i++)
 lungMAXRIGA[i] = 0;
```

```
for(i=0; i<lungMAXPAROLA; i++)
 lungMAXPAROLA[i] = 0;
```

```
if(argc > 1)
 {
 if(strcmp(argv[1], "TESTFILE", strlen("TESTFILE")) == 0)
 {
 FILE *fp;
 fp = fopen(argv[1], "r");
 if(fp != NULL)
 {
 fscanf(fp, "%d", &l lungMAXRIGA);
 fscanf(fp, "%d", &l lungMAXPAROLA);
 fscanf(fp, "%s", lungMAXRIGA);
 fscanf(fp, "%s", lungMAXPAROLA);
 }
 else
 {
 printf("ERRORE: impossibile aprire il file %s", argv[1]);
 }
 }
 else
 {
 printf("ERRORE: impossibile aprire il file %s", argv[1]);
 }
 }
 else
 {
 printf("ERRORE: non è possibile aprire il nome del file (%s)", argv[1]);
 }
}
```

## Esercizi proposti

### Esercizio "Confronto tra date"

4

### Esercizio "Confronto tra date"

- Si scriva un programma che chieda all'utente di inserire due date (giorno, mese, anno) e determini:
  - Se le date solo uguali
  - Se la prima data **precede** la seconda
  - Se la prima data **segue** la seconda
- Il programma dovrà porre particolare attenzione a **non accettare** date non valide
  - Esempio: 30/02/1984
  - Esempio: 10/14/2001

Prompt dei comandi

### Confronto tra date

Inserisci la PRIMA data

Giorno: 15

Mese: 3

Anno: 2007

Inserisci la SECONDA data

Giorno: 26

Mese: 4

Anno: 1967

La prima data 15/3/2007 SEGUE

La seconda data 26/4/1967

## Analisi

## Controlli

- $1 \leq \text{giorno} \leq 31$
- $1 \leq \text{mese} \leq 12$
- $1900 < \text{anno} < 2100$
- Se  $\text{mese} = 4, 6, 9, 11$ , allora  $\text{giorno} \leq 30$
- Se  $\text{mese} = 2$ , allora  $\text{giorno} \leq 29$
- Se  $\text{mese} = 2$  e l'anno non è bisestile, allora  $\text{giorno} \leq 28$ 
  - L'anno è bisestile se è multiplo di 4

## Soluzione

- Scrivere una funzione per la lettura della data, che comprenda al suo interno tutti i controlli:

```
void leggidata(int *giorno,
 int *mese, int *anno) ;
```

- La funzione restituisce, nelle 3 variabili passate *by reference*, le componenti (giorno, mese, anno) della data  
➤ La funzione **garantisce** che la data restituita è **corretta**

7

## Programma principale

```
int main(void)
{
 int g1, m1, a1 ;
 int g2, m2, a2 ;

 printf("Confronto tra date\n\n") ;
 printf("Inserisci la PRIMA data\n") ;
 leggidata(&g1, &m1, &a1) ;

 printf("Inserisci la SECONDA data\n") ;
 leggidata(&g2, &m2, &a2) ;

 /* Confronto delle date */

} /* main */
```

date.c

## Confronto delle date

```
if(g1 == g2 && m1 == m2 && a1 == a2)
 printf("Le date sono uguali\n") ;
else if(a1<a2 ||
 (a1==a2 && m1<m2) ||
 (a1==a2 && m1==m2 && g1<g2))
{
 printf("La prima data %d/%d/%d "
 "PRECEDE la seconda %d/%d/%d\n",
 g1, m1, a1, g2, m2, a2) ;
}
else
{
 printf("La prima data %d/%d/%d "
 "SEGUE la seconda %d/%d/%d\n",
 g1, m1, a1, g2, m2, a2) ;
}
```

date.c

9

## Funzione leggidata (1/5)

```
void leggidata(int *giorno, int *mese,
 int *anno)
{
 int g, m, a ;
 int ok ;

 do {
 do {
 printf("Giorno: ") ;
 scanf("%d", &g) ;

 if(g<1 || g>31)
 printf("Giorno non valido\n");
 } while(g<1 || g>31) ;
```

date.c

10

## Funzione leggidata (2/5)

```
do {
 printf("Mese: ") ;
 scanf("%d", &m) ;

 if(m<1 || m>12)
 printf("Mese non valido\n");
} while(m<1 || m>12) ;

do {
 printf("Anno: ") ;
 scanf("%d", &a) ;

 if(a<=1900 || a>=2100)
 printf("Anno non valido\n") ;
} while(a<=1900 || a>=2100) ;
```

date.c

11

## Funzione leggidata (3/5)

```
ok = 1 ;
if(g>30 && (m==4 || m==6 ||
 m==9 || m==11))
{
 ok = 0 ;
 printf("Il mese %d non "
 "ha %d giorni\n", m, g);
}
else if(g>29 && m==2)
{
 ok = 0 ;
 printf("Il mese %d non "
 "ha %d giorni\n", m, g);
}
```

date.c

## Funzione leggidata (4/5)

```
else if (g==29 && m==2 && a%4!=0)
{
 ok = 0 ;
 printf("Il mese %d non "
 "ha %d giorni perche' "
 "l'anno %d non e' bisestile\n",
 m, g, a);
}
```

date.c

```
#include <stdlib.h>
#include <string.h>
#include <ctype.h>

#define MAXPAROLA 30
#define MAXRIGA 80

int main(int argc, char *argv[]){
 int lunghezzaParola; /* valore di controllo
 delle parole inserite dall'utente */
 char riga[MAXRIGA];
 int i, j, lunghezza; /*
 riga[i]: lunghezza */

 for(i=0; i<MAXRIGA; i++)
 riga[i]=0;

 if(argc > 1){
 lunghezzaParola = strlen(argv[1]);
 if(lunghezzaParola > 0){
 if(argv[1][0] == '#')
 i++;
 if(argv[1][0] != '#')
 lunghezzaParola++;
 if(argv[1][lunghezzaParola-1] == '#')
 i++;
 }
 }
 if(i > lunghezzaParola) {
 printf("ERRORE: Impossibile aprire il file %s", argv[1]);
 exit(1);
 }
 while(fgets(riga, MAXRIGA, i) != NULL)
 /*...*/
}
```

## Esercizi proposti

### Esercizio "Quadrato"

- Gli argomenti sono opzionali: se il carattere viene omesso, occorre stampare "\*". Se anche la dimensione viene omessa, si assume pari a 3

### Esercizio "Quadrato" (2/2)

## Funzione leggidata (5/5)

```
} while(ok==0) ;

*giorno = g ;
*mese = m ;
*anno = a ;

return ;
} /* leggidata */
```

date.c

14

## Esercizio "Quadrato" (1/2)

- Si scriva un programma, denominato quadrato, che stampi a video un quadrato composto di caratteri tutti uguali.
- Il programma riceve due argomenti sulla linea di comando:
  - Il primo argomento indica la **dimensione** del quadrato (ossia il numero di righe e colonne di cui è composto)
  - Il secondo argomento indica il **carattere** di cui è composto il quadrato

16

## Analisi

c:\progr>quadrato

➤ Quadrato 3x3 di "\*"

c:\progr>quadrato 5

➤ Quadrato 5x5 di "\*"

c:\progr>quadrato 5 K

➤ Quadrato 5x5 di "K"

17

18

## Dal punto di vista del main

C:\progr>quadrato

- » argc==1
- » argv[0]=="quadrato"

C:\progr>quadrato 5

- » argc==2
- » argv[0]=="quadrato"
- » argv[1]=="5"

C:\progr>quadrato 5 K

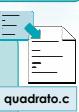
- » argc==3
- » argv[0]=="quadrato"
- » argv[1]=="5"
- » argv[2]=="K"

19

## Soluzione (1/4)

```
int main(int argc, char *argv[])
{
 int dim ;
 char ch ;
 int i, j ;

 if (argc==1)
 {
 dim = 3 ;
 ch = '*' ;
 }
```



20

## Soluzione (2/4)



quadrato.c

```
else if (argc==2)
{
 dim = atoi(argv[1]) ;
 if(dim<1 || dim>20)
 {
 printf("Dimens. non valida\n") ;
 exit(1) ;
 }
 ch = '*' ;
}
```

## Soluzione (3/4)

```
else if (argc==3)
{
 dim = atoi(argv[1]) ;
 if(dim<1 || dim>20)
 {
 printf("Dimens. non valida\n") ;
 exit(1) ;
 }
 ch = argv[2][0] ;
 if(strlen(argv[2])!=1)
 {
 printf("Carattere non valido\n") ;
 exit(1) ;
 }
}
```



quadrato.c

```
else
{
 printf("Numero argomenti "
 "non valido\n") ;
 exit(1) ;
}

for(i=0; i<dim; i++)
{
 for(j=0; j<dim; j++)
 putchar(ch) ;
 putchar('\n') ;
}

exit(0) ;
```

23

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

#define MAXPAROLA 30
#define MAXIGRA 80

int main(int argc, char *argv[])
{
 int lungMAXPAROLA; /* variabile di costante
 dove lungMAXPAROLA è la lunghezza delle parole */
 char lungMAXIGRA; /* variabile di costante
 dove lungMAXIGRA è la lunghezza delle righe */
 int i, indice, lunghezza ;
 FILE *f ;

 f=fopen("MAXIGRA.txt", "r");
 lungMAXIGRA=fscanf(f, "%d", &lungMAXIGRA);
 fscanf(f, "%d", &lungMAXPAROLA);
 fscanf(f, "%d", &indice);

 for(i=0; i<lungMAXIGRA; i++)
 {
 lunghezza=fscanf(f, "%s", lungMAXPAROLA);
 if(lunghezza==0)
 break;
 else
 {
 printf("%s", lungMAXPAROLA);
 printf("\n");
 }
 }
 printf("File MAXIGRA.txt è vuoto\n");
}
```

## Esercizi proposti

### Esercizio "Indovina numero"

## Esercizio "Indovina numero"

- Si realizzi un programma in C per permettere a due giocatori umani di giocare ad "indovina il numero"
- Il primo giocatore attiva il programma, denominato **segreto**, passandogli sulla linea di comando un numero intero tra 1 e 100
- Il secondo giocatore farà una serie di tentativi, immettendoli via tastiera
- Ad ogni tentativo il programma dirà se il numero tentato è più alto o più basso del numero da indovinare

25

## Soluzione (1/3)

```
int main(int argc, char *argv[])
{
 int secreto ;
 int cont, tent ;

 /* Acquisisci dalla linea
 di comando il numero "segreto" */
 if(argc != 2)
 {
 printf("Numero di parametri "
 "errato\n") ;
 exit(1) ;
 }
```



## Soluzione (2/3)

```
segreto = atoi(argv[1]) ;

if(secreto<1 || secreto>100)
{
 printf("Numero secreto "
 "errato\n") ;
 exit(1) ;
}

printf("INDOVINA IL NUMERO\n\n");
```



## Soluzione (3/3)

```
cont = 1 ;
do {
 printf("Tentativo %d: ", cont) ;
 scanf("%d", &tent) ;

 if(tent < secreto)
 printf("Basso...\n") ;
 else if (tent > secreto)
 printf("Alto...\n") ;

 cont++ ;
} while (tent != secreto) ;
cont-- ;

printf("Indovinato: %d tentativi\n",
 cont) ;
```



- Definizione di funzioni in C
  - Prototipo
  - Implementazione
- Passaggio di parametri alle funzioni
  - By value
  - By reference
  - Vettori
- Interfaccia del main
  - Parametri sulla linea di comando
  - Valore di ritorno

2

## Funzioni

### Sommario

## Tecniche di programmazione

- Decomporre il programma in più funzioni, implementando ciascuna individualmente
- Identificare le parti ripetitive e racchiuderle in apposite funzioni
- Permettere il passaggio di parametri al programma
- Analizzare i parametri passati dall'utente

3

## Materiale aggiuntivo

- Sul CD-ROM
  - Testi e soluzioni degli esercizi trattati nei lucidi
  - Scheda sintetica
  - Esercizi risolti
  - Esercizi proposti
- Esercizi proposti da altri libri di testo

4

```

#include <iostream>
#include <string>
#include <vector>
#include <algorithm>

std::string MAXPAROLA_30;
std::string MAXPAROLA_60;

int main(int argc, char *argv[])
{
 int lung(MAXPAROLA_1); /* -> valore di controllo
 delle caratteristiche lunghezza della parola */
 std::string s;
 std::vector<std::string> v;
 int i, resto, lunghezza;
 RESTO = 0;

 for(i=0; i<MAXPAROLA_1; i++)
 v.push_back(" ");

 cout << "Inserire la parola da inserire nel vettore: ";
 cin >> s;
 lung = s.length();
 if(lung > lung)
 {
 cout << "Lunghezza della parola inserita supera il limite imposto da MAXPAROLA_1. Ripetere l'operazione." << endl;
 exit(1);
 }
 else
 {
 if(lung == lung)
 v[resto] = s;
 else
 {
 cout << "Lunghezza della parola inserita supera il limite imposto da MAXPAROLA_1. Ripetere l'operazione." << endl;
 exit(1);
 }
 }
}

void leggi_file(string MAXPAROLA_1) { ... }

```

## Programmazione in C

Unità  
I/O Avanzato e File

I/O Avanzato e File

- ▶ Definizione di file
  - ▶ File di testo in C
  - ▶ Input robusto
  - ▶ Formattazione avanzata
  - ▶ Esercizi proposti
  - ▶ Sommario

## Riferimenti al materiale

## Riferimenti al materiale

## ➤ Testi

- Kernighan & Ritchie: capitolo 7, appendice B
  - Cabodi, Quer, Sonza Reorda: capitoli 3, 8
  - Dietel & Dietel: capitoli 9, 11

## Dispense

- Scheda: "I/O Avanzato in C"
  - Scheda: "Gestione dei file in C"

```

Includere MAXIPAROLA_30
Includere MAXIPAROLA_10

int main(int argc, char *argv[])
{
 int lung_maxiparola[10] = {0}; // <-- vettore di coordinate
 Dati frequenze delle lunghezze delle parole
 int i, j, k;
 int l, m, n, lunghezza;
 int RIS = 1;

 for(i=0; i<MAXIPAROLA_10; i++)
 lung_maxiparola[i] = -1;

 if(argc>1) {
 for(j=0; j<argc-1; j++) {
 lunghezza = strlen(argv[j+1]);
 for(k=0; k<lunghezza; k++) {
 if(argv[j][k] == ' ') {
 lunghezza--;
 break;
 }
 }
 if(lunghezza > RIS)
 RIS = lunghezza;
 else if(lunghezza < lung_maxiparola[l])
 lung_maxiparola[l] = lunghezza;
 l++;
 }
 }

 cout << "Lunghezza massima delle parole inserite è " << RIS << endl;
 cout << "I vettori sono: ";
 for(i=0; i<MAXIPAROLA_10; i++)
 cout << lung_maxiparola[i] << " ";
 cout << endl;
}

```

# I/O Avanzato e File

## Definizione di file

```

#include <stdio.h>
#include <string.h>
#include <ctype.h>

#define MAXAFORO 50
#define MAXCIGA 80

int main(argc, char *argv[])
{
 if(argc != 2)
 {
 printf("Usage: %s MAXAFORO\n", argv[0]);
 exit(1);
 }

 int maxaforo = atoi(argv[1]);
 if(maxaforo < 0 || maxaforo > 50)
 {
 printf("Error: maxaforo (%d) must be between 0 and %d\n",
 maxaforo, MAXAFORO);
 exit(1);
 }

 for(int i = 0; i < maxaforo; i++)
 {
 if(i == 0)
 printf("0");
 else
 printf("%d", i);
 if(i < maxaforo - 1)
 printf(" ");
 }
 printf("\n");
}


```

## Definizione di file

## Directory e file

```
(cargo: 36.2)
 System.out.println("The total memory required for the class is " + memory);
 System.out.println("The total memory required for the object is " + memory);
 System.out.println("The total memory required for the array is " + memory);
}
```

## Definizione

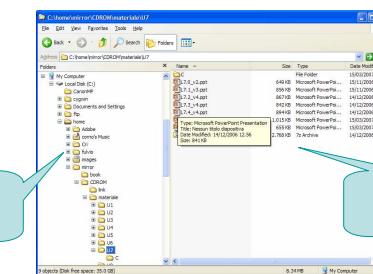
- **File:**
    - Una sequenza di byte
    - Memorizzata su un disco
    - Caratterizzata da uno specifico nome
    - Contenuta all'interno di una specifica directory
  - **Directory:**
    - Un contenitore di file e di altre directory
    - Caratterizzata da uno specifico nome
    - Contenuta all'interno di un'altra directory

**Definizione di file**

- ▶ Directory e file
  - ▶ File binari e file di test

Directory e file

- ▶ Tutti i sistemi operativi permettono di organizzare le informazioni su hard disk secondo la metafora di cartelle (directory) e file



Cartell

File

**Identificazione di un file**

- Per identificare un file occorre dunque conoscere:
    - Il nome assegnato al file
    - Il nome della directory in cui esso è memorizzato
  - Una volta identificato un file, è possibile accedere al suo contenuto
    - Leggere la sequenza di byte di cui è composto
    - Modificare la sequenza di byte di cui è composto



## Operazioni permesse sui file

### ► Operazioni generiche

- Cancellazione di un file esistente
- Rinominazione di un file esistente
- Copia di un file, creando un nuovo file con lo stesso nome in una diversa directory
- Spostamento di un file, equivalente alla copia con cancellazione dell'originale

### ► Operazioni specifiche

- Creazione di un nuovo file
- Modifica del contenuto del file

10

## Operazioni generiche o specifiche

- Le operazioni generiche sono solitamente svolte interagendo con il sistema operativo, e si possono applicare a qualsiasi file
- Le operazioni specifiche invece coinvolgono il contenuto del file, pertanto richiedono programmi specifici per ogni tipologia di file

11

## File = Sequenza di byte

| Address                          | +0 | +1 | +2 | +3 | +4 | +5 | +6 | +7 | +8 | +9 | +A | +B | +C | +D | +E | +F            | Dump               |                  |
|----------------------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---------------|--------------------|------------------|
| 00000000000000000000000000000010 | 00 | 01 | 00 | 00 | 0f | 00 | 80 | 00 | 03 | 00 | 70 | 4f | 53 | 2f | 32 | .....         | p0\$/{2            |                  |
| 00000000000000000000000000000020 | 80 | 72 | 61 | 4b | 00 | 00 | fc | 00 | 00 | 00 | 4e | 50 | 43 | 4c | 54 | 6rak...NFCII  |                    |                  |
| 00000000000000000000000000000030 | 12 | 8e | 1e | f7 | 00 | 00 | 01 | 4c | 00 | 00 | 00 | 36 | 56 | 44 | 4d | 58            | .2s...L...6VDMX    |                  |
| 00000000000000000000000000000040 | e8 | 2e | 98 | 52 | 00 | 00 | 01 | 84 | 00 | 00 | 17 | 70 | 63 | 6d | 61 | 70            | E4R....pemap       |                  |
| 00000000000000000000000000000050 | 24 | 1f | fa | d3 | 00 | 00 | 18 | f4 | 00 | 00 | 02 | fa | 63 | 76 | 74 | 20            | 81^...8...dcvt     |                  |
| 00000000000000000000000000000060 | a9 | 4e | 08 | 7d | 00 | 00 | 18 | f0 | 00 | 00 | 06 | 16 | 66 | 70 | 67 | 6d            | \$..G...8...ffpm   |                  |
| 00000000000000000000000000000070 | 01 | 4e | 08 | 7d | 00 | 00 | 22 | 18 | 00 | 00 | 05 | 0e | 67 | 62 | 79 | 66            | 82...1..."....gylf |                  |
| 00000000000000000000000000000080 | ae | 5e | 88 | 30 | 00 | 01 | 20 | 5c | 00 | 00 | 00 | 36 | 69 | 68 | 65 | 61            | 8<--1...8...6hess  |                  |
| 00000000000000000000000000000090 | 0b | 79 | 03 | 33 | 00 | 01 | 20 | 94 | 00 | 00 | 00 | 24 | 68 | 6d | 74 | 78            | 9y...,"...Shmxx    |                  |
| 000000000000000000000000000000a0 | 50 | 50 | 98 | 0b | 00 | 00 | 01 | 20 | b8 | 00 | 00 | 03 | 9e | 6e | 6f | 63            | 61                 | EP"...,...elcosa |
| 000000000000000000000000000000b0 | fd | 88 | 3c | 7d | 00 | 01 | 24 | 54 | 00 | 00 | 01 | d0 | 6d | 61 | 78 | 70            | y'<1...\$T...Bmaxp |                  |
| 000000000000000000000000000000c0 | 03 | de | 06 | 00 | 00 | 01 | 26 | 24 | 00 | 00 | 00 | 20 | 6e | 61 | 6d | 65            | .P...8\$...name    |                  |
| 000000000000000000000000000000d0 | 6a | 6b | bc | 34 | 00 | 01 | 26 | 44 | 00 | 00 | 05 | 44 | 70 | 6f | 73 | 74            | jk4h...4D...Dpost  |                  |
| 000000000000000000000000000000e0 | cb | 5d | 48 | b4 | 00 | 01 | 28 | 88 | 00 | 00 | 07 | 89 | 70 | 72 | 65 | 70            | EJ'H...+...kprep   |                  |
| 000000000000000000000000000000f0 | ae | 53 | bc | 38 | 00 | 01 | 33 | 14 | 00 | 00 | 05 | 81 | 00 | 00 | 04 | c7            | 8\$he...3...l...G  |                  |
| 000000000000000000000000000000g0 | 05 | 90 | 00 | 05 | 00 | 00 | 05 | 94 | 05 | 33 | 00 | 00 | 04 | 05 | 5a | ..J...8.S...8 |                    |                  |
| 000000000000000000000000000000h0 | 05 | 90 | 00 | 05 | 00 | 00 | 05 | 94 | 05 | 33 | 00 | 00 | 04 | 05 | 5a | ..J...8.S...8 |                    |                  |
| 000000000000000000000000000000i0 | 02 | 02 | 05 | 02 | 04 | 00 | 00 | 00 | 00 | 00 | 00 | 02 | 07 | 03 | 09 | 03            | ..3...8.f...       |                  |
| 000000000000000000000000000000j0 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 4d | 6f | 6e | 65 | 00 | 40 | f0 | 20 | f0            | 00                 |                  |
| 000000000000000000000000000000k0 | 04 | e7 | fe | 7e | 00 | 00 | 06 | a9 | 02 | 67 | 00 | 00 | 01 | 01 | 00 | 01            | 55                 | Mon...8\$        |
| 000000000000000000000000000000l0 | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20            | M...1...8...U      |                  |
| 000000000000000000000000000000m0 | ff | ff | ff | ff | ff | ff | ff | ff | ff | ff | ff | ff | ff | ff | ff | ff            | fffff              |                  |
| 000000000000000000000000000000n0 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 01 | 00 | 04 | 01 | 04 | 03 | 03 | 01 | 05            | .....              |                  |

12

## Conseguenza

- Non è possibile lavorare con file gestiti da altri programmi, a meno di non conoscere il formato del file
- **Eccezione:** se il formato del file è pubblicamente documentato, allora sarà possibile leggerlo e scriverlo interpretando correttamente i byte
  - Esempio: file di testo (ASCII)
  - La codifica ASCII è utilizzata in molti campi: testi (.txt), programmi in C (.c), pagine HTML (.html), ...
  - Esempio: file Acrobat (.pdf)
  - Struttura molto complessa, ma documentata

14

```
#include <iostream.h>
#include <string.h>
#include <ctype.h>

#define MAXPAROLA 30
#define MAXIGRA 80

int main(int argc, char *argv[])
{
 int lungMAXPAROLA; /* verifcare di conoscere
 delle frequenze delle parola */
 char MAXIGRA[AXIGRA];
 int i, indice, lungMAXI;
 FILE *f;

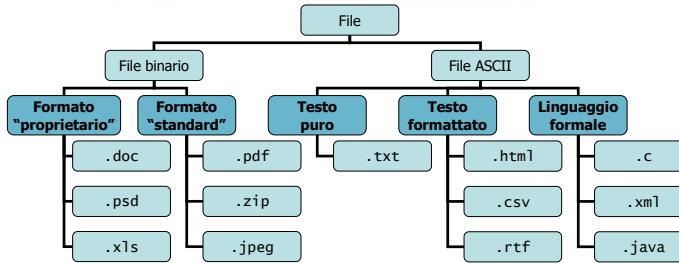
 f=fopen(argv[1], "r");
 if(f==NULL)
 cout<<"ERRORE: impossibile aprire il file "<<argv[1]<<endl;
 else
 cout<<"file aperto con successo!</n></n>";

 for(i=0; i<lungMAXPAROLA; i++)
 MAXIGRA[i]=NULL;
 lungMAXI=0;
 while(fgets(MAXIGRA, AXIGRA, f))
 lungMAXI++;
 cout<<"lunghezza della parola: "<<lungMAXI<<endl;
 cout<<MAXIGRA<<endl;
 fclose(f);
}
```

## Definizione di file

## File binari e file di testo

## Vista d'insieme dei formati di file



16

## Differenze operative

### A livello di programmazione, esistono

- Funzioni generali per accedere ai file
- Funzioni specifiche per la lettura e la scrittura del contenuto di file binari
- Funzioni specifiche per la lettura e la scrittura del contenuto di file ASCII

### Il corretto funzionamento di un programma dipende dalla perfetta conoscenza del formato del file stesso

17

## Scelta operativa

- In questo corso si tratteranno esclusivamente i file ASCII
  - Più semplici da comprendere
  - Facili da visualizzare e da creare con qualsiasi editor di testi
    - Notepad o molte sue alternative
    - L'ambiente di sviluppo in C
- Si userà indifferentemente la denominazione "file ASCII" o "file di testo"

18

## File di "testo puro"

```

Notepad++ C:\Program Files\Notepad++\readme.txt
File Edit Search View Format Language Settings Macro Run TextFX TextFX Quick TextFX Viz Plugins Window Z
readme.txt
1 Notepad++ release Note :
2
3 What is Notepad++?
4 ****
5
6 Notepad++ is a generic source editor (it tries to be anyway) and Notepad replacement written in C++ with the win32 API. The aim of Note
totally customizable GUI.
7
8 Why another source editor?
9 ****
10
11 I worked for a big smart card company as engineer developer, and I took charge of looking for an
alternative solution for an internal tool coded in Java. The internal tool needed an edit
component, and I found Scintilla (which allows me to develop in C++) via Internet. I began my
conception and development in this project.
12
13 Unfortunately the project is abandoned after 3 months. The reason is political : our company
will use a new environment of development, as well a property language, to re-develop all
internal tools. All the developers are forced to use this unstable and uncomfortable IDE and the
incoherent property language.
14
15 As a C++/Java developer, I decided to continue the project with my spare time. The prototype of
project was already done, I removed the components which depend on the specification of
abandoned project - It made a generic code editor. Then I made it available on.sourceforge and I
hope it will be useful for other people.
16
17
18
19
20
Normal text file rb char : 1770 ln:1 Col:1 Sel:0 DosWindows ANSI INS

```

19

## File di "testo puro"

```

Notepad++ C:\Program Files\Notepad++\readme.txt
File Edit Search View Format Language Settings Macro Run TextFX TextFX Quick TextFX Viz Plugins Window Z
readme.txt
1 Notepad++ release Note :
2
3 What is Notepad++?
4 ****
5
6 Notepad++ is a generic source editor (it tries to be anyway) and Notepad replacement written in C++
with the win32 API. The aim of Note
totally customizable GUI.
7
8 Why another source editor?
9 ****
10
11 I worked for a big smart card company as
alternative solution for an internal tool component, and I found Scintilla (which
conception and development in this project
was abandoned after 3 months. The reason is political : our company
will use a new environment of development, as well a property language, to re-develop all
internal tools. All the developers are forced to use this unstable and uncomfortable IDE and the
incoherent property language.
12
13 Unfortunately the project is abandoned after 3 months. The reason is political : our company
will use a new environment of development, as well a property language, to re-develop all
internal tools. All the developers are forced to use this unstable and uncomfortable IDE and the
incoherent property language.
14
15 As a C++/Java developer, I decided to continue the project with my spare time. The prototype of
project was already done, I removed the components which depend on the specification of
abandoned project - It made a generic code editor. Then I made it available on.sourceforge and I
hope it will be useful for other people.
16
17
18
19
20
Normal text file rb char : 1770 ln:1 Col:1 Sel:0 DosWindows ANSI INS

```

- Serie di più righe
- Ciascuna riga ha un numero variabile di caratteri
- Solitamente contengono del testo "libero" (ad es., in italiano)
- Concetti di: carattere, parola, frase, riga

## File di "testo formattato" (es: XML)

```

Notepad++ C:\Program Files\Notepad++\config.xml
File Edit Search View Format Language Settings Macro Run TextFX TextFX Quick TextFX Viz Plugins Window Z
config.xml
1 <?xml version="1.0" encoding="Windows-1252"?>
2 <NotepadPlus>
3 <GUIConfig>
4 <!-- 3 status : "large", "small" or "hide"-->
5 <GUIConfig name="ToolBar">standard</GUIConfig>
6 <!-- 2 status : "show" or "hide"-->
7 <GUIConfig name="StatusBar">show</GUIConfig>
8 <!-- For all attributes, 2 status : "yes" or "no"-->
9 <GUIConfig name="TabBar">dragAndDrop="yes" drawInactiveTab="yes" </GUIConfig>
10 <!-- 2 positions : "horizontal" or "vertical"-->
11 <GUIConfig name="ScintillaViewSplitter">vertical</GUIConfig>
12 <!-- For the attribut of position, 2 status : docked or undocked ; 2 status : "show" or
"hide"-->
13 <GUIConfig name="UserDefinedDig" position="docked"/></GUIConfig>
14 <GUIConfig name="TabSetting" size="4" replaceBySpace="no" />
15 <!-- For position-->
16 <GUIConfig name="AppPosition" x="195" y="101" width="1033" height="834" >
17 <!-- For the primary scintilla view,
18 2 status for Attribut lineNumberMargin, bookmarkMargin, indentGuideLine and
currentLineHilitingShow: "show" or "hide"
19 4 status for Attribut folderMarkStyle : "simple", "arrow", "circle" and "box" -->
20 <GUIConfig name="ScintillaPrimaryView" lineNumberMargin="show" bookmarkMargin="show" >
 folderMarkStyle="box" indentGuideLine="show" currentLineHilitingShow="show" wrap="no" edge="no" </GUIConfig>

```

## File di "testo formattato" (es: XML)

```

<?xml version="1.0" encoding="Windows-1252"?>
<!DOCTYPE NotepadPlus>
<NotepadPlus>
 <GUIConfig name="Large" status="true" />
 <GUIConfig name="ToolBar" stand="1" status="show" />
 <GUIConfig name="StatusBar" show="1" />
 <!-- For all attributes, 2 status -->
 <GUIConfig name="TabBar" dragAnd reduce="yes" />
 <!-- 2 positions : "horizontal" -->
 <GUIConfig name="ScintillaViews" hide="1" />
 <!-- For the attribute of position -->
 <GUIConfig name="UserDefinedig" <!-- For the primary scintilla -->
 <GUIConfig name="TabSetting" size="1" />
 <!-- App position-->
 <GUIConfig name="AppPosition" x="17" y="17" isMaximized="yes" />
 <!-- For the primary scintilla -->
 <GUIConfig name="Sci" currentLineHighlight="show" or="hide" fold="4" status="Attribut" folderMarkStyle="box" indentGuideline="1" isMaximized="yes" />
 <!-- For the primary scintilla -->
 <GUIConfig name="ScintillaPrima

```

- Linguaggio di programmazione basato su "tag"
- Possibile "annidamento" dei tag
- Complesso da analizzare, richiede strutture dati avanzate

## File di "testo formattato" (es: CSV)

```

"Unita","Lezione","Argomento","Tipologia","Materiale aggiuntivo","PPT","Sorgenti","Video"
".....",
"0. Introduzione al corso",,
"0.1.1. Introduzione",,
"0.1.2. Obiettivi del corso",,"Slide",,"Scheda sintetica",,,,
"1.1.1. Metodologie didattiche",,"Slide",,"",,,,
"1.1.2. Argomenti trattati",,"Slide",,"",,,,
"1.2.1. Prerequisiti",,"Slide",,"",,,,
"1.2.2. Materiale didattico",,,,
"1.2.3. Libri di testo",,"Slide",,"Bibliografia",,,,
"1.2.4. Strumenti software",,"Slide",,"Link utili",,,,
"1.2.5. Dispense e siti web",,"Slide",,"",,,,
"1.3.1. Primo programma in C",,,,
"1.3.2. Introduzione ai linguaggi C",,,,
"1.3.3. Introduzione storia",,"Slide",,,,
"1.3.4. Struttura minima di un file C",,,,
"1.3.5. Applicazioni C in modo "console",,"Slide",,,,
"1.3.6. Struttura del programma",,"Slide",,,,
"1.3.7. Commenti",,"Slide",,,,
"1.3.8. Directives "include",,,,
"1.3.9. Definizione di variabili",,"Slide",,,,
"1.3.10. Tipi di dati",,"Slide",,,,
"1.3.11. Sostanzialmente minimale di istruzioni",,,,
"1.3.12. I tipi int e float",,"Slide",,"Scheda #1: istruzioni C minimali",,"istruzioneiminime.c",,,,
"1.3.13. "

```

## File di "testo formattato" (custom)

```

"Unita","Lezione","Argomento","Tipologia","Materiale aggiuntivo","PPT","Sorgenti","Video"
".....",
"0. Introduzione al corso",,
"0.1.1. Introduzione",,
"0.1.2. Obiettivi del corso",,"Slide",,"Scheda sintetica",,,,
"1.1.1. Metodologie didattiche",,"Slide",,"",,,,
"1.1.2. Argomenti trattati",,"Slide",,"",,,,
"1.2.1. Prerequisiti",,"Slide",,"",,,,
"1.2.2. Materiale didattico",,,,
"1.2.3. Libri di testo",,"Slide",,"Bibliografia",,,,
"1.2.4. Strumenti software",,"Slide",,"Link utili",,,,
"1.2.5. Dispense e siti web",,"Slide",,"",,,,
"1.3.1. Primo programma in C",,,,
"1.3.2. Introduzione ai linguaggi C",,,,
"1.3.3. Introduzione storia",,"Slide",,,,
"1.3.4. Struttura minima di un file C",,,,
"1.3.5. Applicazioni C in modo "console",,"Slide",,,,
"1.3.6. Struttura del programma",,"Slide",,,,
"1.3.7. Commenti",,"Slide",,,,
"1.3.8. Directives "include",,,,
"1.3.9. Definizione di variabili",,"Slide",,,,
"1.3.10. Tipi di dati",,"Slide",,,,
"1.3.11. Sostanzialmente minimale di istruzioni",,,,
"1.3.12. I tipi int e float",,"Slide",,"Scheda #1: istruzioni C minimali",,"istruzioneiminime.c",,,,
"1.3.13. "

```

- Ogni riga è un dato
- Ogni dato è rappresentato da più campi
- I campi sono separati da un carattere specifico ( , o ; )
- I campi possono essere numerici o testuali
- Ragionevolmente semplice da analizzare, modificare e creare

## File di "testo formattato" (custom)

```

A201 2 20.00 23
A302 1 15.00 34
B200 1 0.85 35
B200 2 1.70 35
B200 10 8.50 35
A100 1 43.00 43
C000 1 12.50 44
A302 1 15.00 45
A302 2 30.00 46
B200 2 1.70 46
A201 1 10.00 47
A302 1 15.00 50

```

## File di "testo formattato" (custom)

```

A201 2 20.00 23
A302 1 15.00 34
B200 1 0.85 35
B200 2 1.70 35
B200 10 8.50 35
A100 1 43.00 43
C000 1 12.50 44
A302 1 15.00 45
A302 2 30.00 46
B200 2 1.70 46
A201 1 10.00 47
A302 1 15.00 50

```

- Formato "inventato" ad hoc per ciascun programma specifico
- Versione semplificata del CSV, dove il separatore è lo spazio e non vi sono virgolette delimitatrici
- È il più semplice da gestire

- ▶ Accesso ai file
- ▶ Funzioni fopen/fclose
- ▶ Funzioni fget\*/fput\*
- ▶ Funzioni fprintf/fscanf
- ▶ Condizione feof

## I/O Avanzato e File

### File di testo in C

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

#define MAXPAROLA 30
#define MAXSIGA 80

int main(int argc, char *argv[])
{
 int lungMAXPAROLA; /* valore di costante
 cioè numero massimo di caratteri nella parola */
 char siga[MAXSIGA];
 int i, indice, lungparola;
 FILE *fptr;

 if(argc != 2)
 {
 printf("Errore: \"%s\" non permette di aprire il file %s\n", argv[0], argv[1]);
 exit(1);
 }
 else if(isupper(argv[1][0]))
 {
 printf("Errore: \"%s\" non permette di aprire il file %s\n", argv[0], argv[1]);
 exit(1);
 }
 else
 {
 strcpy(siga, argv[1]);
 i = 0;
 lungparola = strlen(siga);
 while((siga[i] != '\0') && (i < lungparola))
 {
 if((siga[i] >='A') && (siga[i] <='Z'))
 siga[i] = (char)tolower(siga[i]);
 i++;
 }
 }
 if((fptr = fopen(siga, "r")) == NULL)
 {
 printf("Errore: non è possibile aprire il file %s\n", siga);
 exit(1);
 }
 else
 {
 fgets(sigia, MAXSIGA, fptr);
 lungMAXSIGA = strlen(sigia);
 if(lungMAXSIGA > lungparola)
 lungMAXSIGA = lungparola;
 for(indice = 0; indice < lungMAXSIGA; indice++)
 printf("%c", sigia[indice]);
 printf("\n");
 }
 fclose(fptr);
}
```

## File di testo in C

### Accesso ai file

### Accesso ai file (1/4)

- ▶ Un programma C può accedere ad alcuni file presenti sui dischi del calcolatore
  - File aperto: file al quale attualmente il programma ha accesso
  - File chiuso: file residente su disco al quale attualmente il programma non ha accesso

### Accesso ai file (2/4)

- ▶ All'atto dell'apertura di un file, il programma deve dichiarare la modalità di accesso
  - Modalità di **lettura**: il programma può leggere il contenuto del file, ma non modificarlo
  - Modalità di **scrittura**: il programma può riscrivere da zero il contenuto del file
  - Modalità di **aggiunta**: il programma può aggiungere nuove informazioni al file
  - Modalità di **lettura/scrittura**: tutte le precedenti
- ▶ I successivi accessi al file devono essere compatibili con la modalità di accesso dichiarata

### Accesso ai file (3/4)

- ▶ L'accesso ai file di testo è rigorosamente sequenziale
  - La lettura avviene dalla prima riga all'ultima, dal primo carattere all'ultimo
  - In scrittura, ogni riga o carattere scritto vengono posizionati dopo le righe o caratteri scritti in precedenza
    - A partire dal primo carattere, in modalità di scrittura
    - A partire dall'ultimo carattere esistente, in modalità di aggiunta

## Accesso ai file (4/4)

- All'atto dell'apertura di un file, il programma deve dichiarare se il file è di tipo binario oppure di testo
  - La differenza consiste solamente nel trattamento "speciale" del carattere '\n' nel caso dei file di testo
  - In questo corso useremo sempre la modalità testuale

7

## Stream associato ad un file

- In un programma C, esiste un tipo di dato specifico per rappresentare le informazioni relative ad un file aperto
  - Denominato: **file stream** (flusso associato ad un file)
  - Tipo di dato: **FILE \*** (definito in `<stdio.h>`)
- "Aprire" un file significa quindi creare un nuovo stream ed associarlo ad uno specifico file sul disco

8

## Significato di stream

- Una volta che il file è aperto, il suo stream rappresenta
  - Un "collegamento" mediante il quale poter compiere delle operazioni sul contenuto del file
  - Le modalità di accesso scelte (testo/binario, lettura/scrittura/...)
  - La posizione attuale a cui si è arrivati nello scrivere o nel leggere il file
- Ogni operazione sul file avviene chiamando una funzione che riceve lo stream come parametro

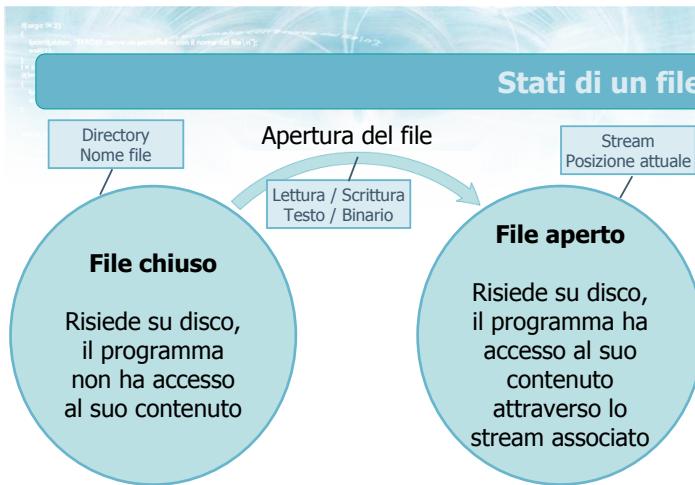
9

## Stati di un file



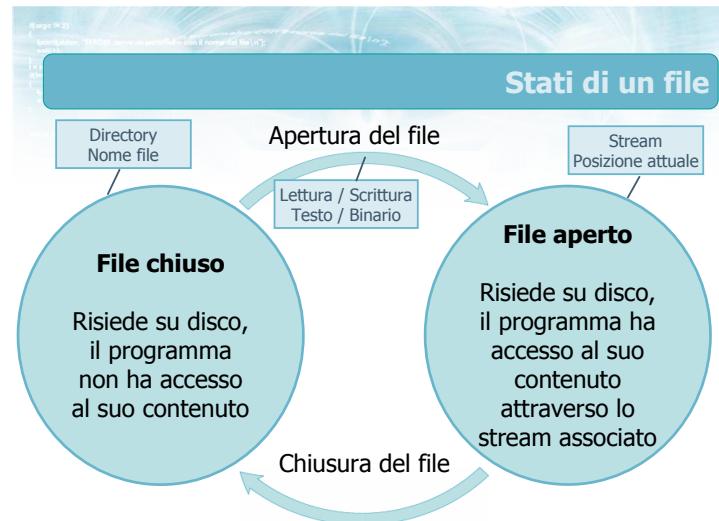
10

## Stati di un file



11

## Stati di un file



12

## Lettura di un file

Apertura del file

**File aperto in lettura**

Posizione iniziale  
(primo carattere)

Apertura del file

**File aperto in lettura**

Posizione iniziale  
(primo carattere)

**Leggi riga /  
Leggi carattere**

13

14

## Lettura di un file

Apertura del file

**File aperto in lettura**

Posizione iniziale  
(primo carattere)

**File aperto in lettura**

Posizione intermedia  
(n-esimo carattere)

**Leggi riga /  
Leggi carattere**

15

## Lettura di un file

Apertura del file

**File aperto in lettura**

Posizione iniziale  
(primo carattere)

**File aperto in lettura**

Posizione intermedia  
(n-esimo carattere)

**Leggi riga /  
Leggi carattere**

Condizione  
end-of-file

Chiusura del file

**Leggi riga /  
Leggi carattere**

16

## Scrittura di un file

Apertura del file

**File aperto in scrittura**

Posizione iniziale  
(primo carattere)

Apertura del file

**File aperto in scrittura**

Posizione iniziale  
(primo carattere)

## Scrittura di un file

**Scrivi riga /  
Scrivi carattere**

17

18

## Scrittura di un file



19

## Scrittura di un file



20

## Aggiunta ad un file



21

## Funzioni C

- Tutte le funzioni per l'accesso ai file sono contenute in `<stdio.h>`
- Funzioni per apertura e chiusura: `fopen`, `fclose`
- Funzioni per la lettura: `fgetc`, `fgets`, `fscanf`
- Funzioni per la scrittura: `fputc`, `fputs`, `fprintf`
- Funzioni per lo stato del file: `feof`

22



## File di testo in C

### Funzioni fopen/fclose



## Funzioni fopen e fclose

- Apertura del file (`fopen`):
  - La funzione `fopen` apre un file e restituisce una variabile stream
  - Richiede il nome del file e le modalità di apertura
  - Restituisce una nuova variabile di tipo `FILE *`
- Chiusura del file (`fclose`)
  - Quando il file non è più richiesto, si chiama la funzione `fclose` che chiude il file
  - Richiede come parametro lo stream, precedentemente restituito da `fopen`

24

## fopen: sintassi

```
FILE * f ;
...
f = fopen("nomefile", "modo") ;
```

Variabile stream  
di tipo FILE \*

Stringa  
contenente il  
nome del file

Modalità di  
apertura  
(stringa)

25

## Nome del file

```
f = fopen("nomefile", "modo") ;
f = fopen("dati.txt", "modo") ;
```

26

## Nome del file

```
f = fopen("nomefile", "modo") ;
f = fopen("dati.txt", "modo") ;
f = fopen("c:\\\\prog\\\\dati.txt",
"modo") ;
```

27

## Nome del file

```
f = fopen("nomefile", "modo") ;
f = fopen("dati.txt", "modo") ;
f = fopen("c:\\\\prog\\\\dati.txt",
"modo") ;

char nome[20] ;
gets(nome) ;
f = fopen(nome, "modo") ;
```

28

## Nome del file

```
f = fopen("nomefile", "modo") ;
f = fopen("dati.txt", "modo") ;
f = fopen("c:\\\\prog\\\\dati.txt",
"modo") ;

char nome[20] ;
gets(nome) ;
f = fopen(nome, "modo") ;

f = fopen(argv[1], "modo") ;
```

29

## Modalità di apertura

```
f = fopen("nomefile", "modo") ;
```

Modalità **lettura**, file di testo → "rt" "r"

Modalità **scrittura**, file di testo → "wt" "w"

Modalità **aggiunta**, file di testo → "at" "a"

30

## Effetto della fopen (1/3)

### » Modalità "r"

- Se il file esiste, viene aperto ed f punta allo stream relativo, posizionato in lettura al primo carattere
- Se il file non esiste, non viene creato nessuno stream e f==NULL

31

## Effetto della fopen (2/3)

### » Modalità "w"

- Se il file non esiste, viene creato da zero ed f punta allo stream relativo, posizionato in scrittura al primo carattere
- Se il file esiste già, viene innanzitutto cancellato e poi ricreato da zero, f punta allo stream relativo, posizionato in scrittura al primo carattere
- Se non è possibile creare il file (perché la directory non esiste, o il disco è protetto in scrittura, ...), non viene creato nessuno stream e f==NULL

32

## Effetto della fopen (3/3)

### » Modalità "a"

- Se il file non esiste, viene creato da zero ed f punta allo stream relativo, posizionato in scrittura al primo carattere
- Se il file esiste già, non viene modificato, f punta allo stream relativo, posizionato in scrittura dopo l'ultimo carattere
- Se non è possibile creare o modificare il file (perché la directory non esiste, o il disco è protetto in scrittura, ...), non viene creato nessuno stream e f==NULL

33

## Controllo dell'errore

```
FILE * f ;
...
f = fopen("nomefile" , "r") ;
if(f == NULL)
{
 printf("Impossibile aprire file\n");
 exit(1) ;
}
```

34

## Suggerimento

- » Ogniqualvolta viene chiamata la funzione fopen, è indispensabile subito dopo controllare se il valore ritornato non è NULL
- » È da considerarsi **errore** una chiamata ad fopen di cui non venga controllato il risultato
- » In caso di errore, solitamente conviene interrompere il programma segnalando un codice di errore
  - Esempio: exit(1) ;

35

## fclose: sintassi

```
FILE * f ;
...
f = fopen("nomefile" , "modo") ;
/* accesso al file */
fclose(f) ;
```

Variabile  
stream

36

## Avvertenze

- » La funzione `fclose` può essere chiamata solamente su stream correttamente aperti
  - Mai chiamare `fclose` se `f==NULL`
- » Dopo la chiusura del file, non è più possibile accedere allo stream
  - Eventualmente, ri-aprirlo nuovamente

37

## Controllo dell'errore

```
int ris ;
...
ris = fclose(f) ;
if(ris!=0)
{
 printf("Impossibile chiudere\n") ;
 exit(1) ;
}
```

38



## Suggerimento

- » Conviene definire due funzioni aggiuntive, che chiamino le funzioni di libreria `fopen` e `fclose` e addizionalmente compiano i controlli d'errore
- » Chiameremo `myfopen` e `myfclose` tali funzioni
- » Nei programmi chiameremo sempre `myfopen` e `myfclose`, e mai direttamente le funzioni di libreria

39

## Funzione myfopen

```
FILE * myfopen(char *name, char *mode)
{
 FILE * f ;

 f = fopen(name, mode) ;
 if (f==NULL)
 {
 printf("Impossibile aprire %s\n",
 name) ;
 exit(1) ;
 }
 return f ;
}
```



```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

#define MAXPAROLA 30
#define MAXIGRA 80

int main(int argc, char *argv){
 int lungMAXPAROLA; /* verifica di constanti
 delle frequenze e lunghezza delle parole */
 char c, AXIGRA[];
 int i, indice, lunghezza ;
 FILE *f;

 for(i=0;i<MAXPAROLA;i++)
 lungMAXPAROLA++;

 if(argc<2){
 printf("Usage: %s <file>\n", argv[0]);
 exit(1);
 }
 f=fopen(argv[1], "r");
 if(f==NULL){
 printf("Errore: impossibile aprire il file %s", argv[1]);
 exit(1);
 }
 while(fgets(AXIGRA, MAXIGRA, f))
 ...
}
```

## Funzione myfclose

```
int myfclose(FILE *f)
{ int ris ;

 if (f==NULL)
 { printf("ERRORE INTERNO\n") ;
 exit(1) ;
 }
 ris = fclose(f) ;
 if(ris!=0)
 { printf("Impossibile chiudere\n") ;
 exit(1) ;
 }
 return ris ;
}
```



## File di testo in C

## Funzioni fget\*/fput\*

## Lettura e scrittura su file

|                   | Lettura | Scrittura |
|-------------------|---------|-----------|
| Carattere singolo | fgetc   | fputc     |
| Riga intera       | fgets   | fputs     |

43

## Lettura e scrittura su file

|                   | Lettura | Scrittura |
|-------------------|---------|-----------|
| Carattere singolo | fgetc   | fputc     |
| Riga intera       | fgets   | fputs     |

Legge prossimo  
elemento, fino  
alla fine del file

Scrive o  
aggiunge

44

## Lettura e scrittura su file

|                   | Lettura | Scrittura |
|-------------------|---------|-----------|
| Carattere singolo | fgetc   | fputc     |
| Riga intera       | fgets   | fputs     |

Parametro:  
char

Legge prossimo  
elemento, fino  
alla fine del file

Scrive o  
aggiunge

Parametro:  
stringa

45

## fgetc: sintassi

```
int ch ;
ch = fgetc(f) ;
```

Stream aperto  
in lettura

Prossimo carattere del file;  
EOF se il file è finito

46

## fputc: sintassi

```
int ch ;
fputc(ch, f) ;
```

Stream aperto in  
scrittura o in aggiunta

Carattere da  
aggiungere al file

47

## fgets: sintassi

```
char str[80] ;
fgets(str, 79, f) ;
```

Max numero di  
caratteri letti

Stringa nella quale viene  
letta la prossima riga del file  
(fino al \n compreso)

Stream aperto  
in lettura

48

## Fine del file

- La funzione fgets restituisce un valore NULL se ha tentato di leggere oltre la fine del file

```
char str[80] ;

while(fgets(str, 79, f) != NULL)
{
 /* elabora str */
}
```

49

## fputs: sintassi

```
char str[80] ;
```

```
fputs(str, f) ;
```

Stream aperto in scrittura o in aggiunta

Stringa da aggiungere al file  
(solitamente termina con \n)

50

## Esercizio: "Frequenza lettere"

- Sia dato un file di testo, contenente dei brani scritti da un utente
- Si scriva un programma in C che acquisisca sulla linea di comando il nome di tale file, e che stampi le frequenze con cui compaiono le varie lettere dell'alfabeto
- Si considerino equivalenti le maiuscole e le minuscole, e si ignorino i caratteri di spaziatura e punteggiatura

51

## Analisi (1/2)

manzoni.txt

Quel ramo del lago di Como,  
che volge a mezzogiorno,  
tra due catene non interrotte di monti,  
tutto a seni e a golfi,  
a seconda dello sporgere e del  
rientrare di quelli, vien, quasi  
a un tratto, a ristringersi, e a  
prender corso e figura di fiume,  
tra un promontorio a destra,  
e un'ampia costiera dall'altra parte

52

## Analisi (2/2)

manzoni.txt

```
ex Prompt dei comandi
C:\prog>frequenza manzoni.txt
A : 26
B : 0
C : 6
D : 12
E : 32
F : 3
G : 7
H : 1
I : 21
J : 0
```

53

## Soluzioni possibili

- Occorre calcolare un vettore di frequenze, in cui ciascuna posizione rappresenti la frequenza di ciascuna delle lettere alfabetiche
- Ci sono due approcci possibili alla lettura del file:
  - Soluzione per caratteri:** il file viene letto un carattere alla volta, usando la funzione fgetc
  - Soluzione per righe:** il file viene letto una riga alla volta, usando la funzione fgets, e tale riga viene poi esaminata con un ciclo interno al programma

54

### Soluzione 1: per caratteri (1/3)

```
const int LETT = 26 ;
int freq[LETT] ; /* frequenze lettere */

FILE * f ;
int ch, i ;

if (argc!=2)
{
 printf("Numero argomenti errato\n") ;
 exit(1) ;
}

for(i=0; i<LETT; i++)
 freq[i] = 0 ;
```



### Soluzione 1: per caratteri (2/3)

```
f = myfopen(argv[1], "r") ;

ch = fgetc(f) ;
while(ch!=EOF)
{
 if(isalpha(ch))
 {
 i = toupper(ch)-'A' ;
 /* posizione 0..25 della lettera */
 freq[i]++ ;
 }
 ch = fgetc(f) ;
}
myfclose(f) ;
```



```
for(i=0; i<LETT; i++)
{
 printf("%c : %d\n", i+'A', freq[i]) ;
}

exit(0) ;
```



### Soluzione 2: per righe

```
const int LUN = 200 ;
char riga[LUN+1];
...

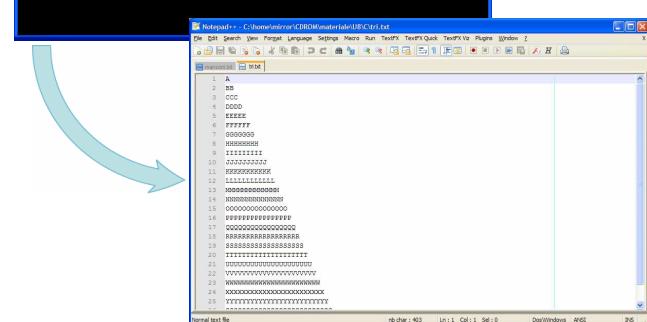
while(fgets(riga, LUN, f) != NULL)
{
 for(i=0; riga[i]!=0; i++)
 {
 if(isalpha(riga[i]))
 {
 freq[toupper(riga[i])-'A']++ ;
 }
 }
}
```



### Esercizio: "Triangolo alfabetico"

- Si realizzi un programma in C che crei un file di testo contenente tutte le lettere dell'alfabeto, con una disposizione "a triangolo"
  - La prima riga contiene una volta la lettera A
  - La seconda riga contiene 2 volte la lettera B
  - La terza riga contiene 3 volte la lettera C
  - ...
- Il nome del file viene passato come primo argomento sulla linea di comando

c:\prog>triangolo tri.txt



## Soluzione (1/2)

```
FILE * f ;
int i, j ;
char ch ;

if (argc!=2)
{
 printf("Numero argomenti errato\n") ;
 exit(1) ;
}

f = myfopen(argv[1], "w") ;
```



## Soluzione (2/2)

```
for(i=0; i<26; i++)
{
 ch = i+'A' ;
 for(j=0; j<=i; j++)
 fputc(ch, f) ;
 fputc('\n', f) ;
}
myfclose(f) ;
exit(0) ;
```



```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

#define MAXPAROLA 20
#define MAXIGRA 80

int main(int argc, char *argv[])
{
 int kmp[MAXPAROLA]; // -> vettore di costanti
 char parola[MAXPAROLA]; // -> parola da cercare
 char igra[MAXIGRA]; // -> parola, lunghezza :
 int i, j, n, m;

 for(i=0; i<MAXPAROLA; i++)
 kmp[i]=0;

 kmp[0]=0;

 if(argc != 2)
 {
 printf("Usage: %s stringa parola da cercare\n", argv[0]);
 exit(1);
 }

 if((n = fopen(argv[1], "r")) == NULL)
 exit(1);

 if((m = fopen(igra, "w")) == NULL)
 exit(1);

 if(fscanf(n, "%s", parola) != 1)
 fprintf(m, "ERRORE: impossibile aprire il file %s", argv[1]);
 else
 exit(1);

 while(fscanf(n, "%s", MAXIGRA) != NULL)
 {
 if(strstr(parola, igra))
 fprintf(m, "%s", igra);
 }
}


```

## File di testo in C

## Funzioni printf/fscanf

## fprintf: sintassi

```
FILE * f ;
fprintf(f, "formato", variabili) ;
```

Stream aperto in  
scrittura o in aggiunta

### Elenco delle variabili da scrivere

Formato dei dati da stampare,  
usando gli stessi specificatori  
validi per printf

## **Output formattato**

- ▶ Qualora sia necessario creare file con più campi nella stessa riga, è scomodo ricorrere alle funzioni `fputc/fputs`
  - ▶ È possibile utilizzare una variante della funzione `printf`, operante su uno stream aperto in scrittura
    - `fprintf(f, "formato", x, y, z) ;`

## Input formattato

- Qualora sia necessario leggere file con più campi nella stessa riga
    - È scomodo ricorrere alla funzione `fgetc`
    - Il risultato della funzione `fgets` deve successivamente essere analizzato
  - È possibile utilizzare una variante della funzione `scanf`, operante su uno stream aperto in lettura
    - `fscanf(f, "formato", &x, &y, &z) ;`

## fscanf: sintassi

```
FILE * f ;
fscanf(f, "formato", &variabili) ;

Stream aperto
in lettura

Puntatori alle
variabili da leggere

Formato dei dati da leggere,
usando gli stessi specificatori
validi per scanf
```

67

## fscanf: una funzione pericolosa

- Nonostante la funzione fscanf sia prevista dalla libreria standard C, è considerata una funzione **pericolosa** nella lettura di file in generale
- In particolare, qualora il file non sia nel formato corretto (file contenente errori), allora il meccanismo di funzionamento di fscanf rende **impossibile** acquisire i dati in modo **affidabile**
- Suggerimento: **non usare mai** fscanf
- Nella prossima lezione vedremo una **soluzione robusta** al problema

68



## File di testo in C

## Condizione feof

70

## Tentativi di lettura

- Se si tenta di leggere oltre l'End-of-File
  - fgets restituisce NULL
  - fgetc restituisce EOF
  - fscanf restituisce EOF
- È possibile controllare tali valori di ritorno per controllare la fine del file
  - In tali casi, l'errore è già avvenuto, e l'operazione di lettura non è andata a buon fine

71

## Funzione feof

- La funzione feof è specificatamente utile per verificare se uno stream f è già nella condizione di End-of-File **prima** di tentare operazioni di lettura
  - `if ( !feof( f ) ) { ... }`
- La funzione, partendo dallo stream f, restituisce:
  - **Vero**, se lo stream è già in End-of-File, e quindi le successive letture falliranno
  - **Falso**, se lo stream **non** è ancora in End-of-File, e quindi sono possibili ulteriori letture

72

## Esempio

```
ch = fgetc(f) ;
while(ch!=EOF)
{
 .../* elabora ch */
 ch = fgetc(f) ;
}
```

```
while(!feof(f))
{
 ch = fgetc(f) ;
 .../* elabora ch */
}
```

# I/O Avanzato e File

## Input robusto

2

## Input robusto

## Problemi nella lettura da file

4

**Largo (W 2)**  
Spedizione: 1000000 minuti passati a dormire nel letto di Max (V)  
+1  
I +  
R +

- ▶ Problemi nella lettura da file
  - ▶ Soluzione basata su fgetc
  - ▶ Funzione sscanf
  - ▶ Soluzione basata su fgets

## Lettura da file

- I file di testo contengono dati secondo un certo **formato**
  - È semplice scrivere un programma in grado di leggere un file formattato correttamente
  - Diviene molto complesso gestire eventuali **errori** di formato del file

Il logo (n° 2) spieghiamo: TECNO, invece di partecipare con il numero del filo "n";

## Esempio

- Un file di testo contiene i PIN dei bancomat dei membri di una famiglia
  - Il file di testo contiene sulla prima riga il numero di bancomat descritti nel file
  - Le righe successive contengono le informazioni su ciascun bancomat, uno per riga
  - Ciascuna riga contiene 3 campi separati da spazi:
    - Il nome del proprietario del bancomat
    - Il numero del bancomat
    - Il PIN segreto (5 cifre)

- Righe contenenti un numero errato di elementi
    - Elementi in eccesso
      - In fondo
      - All'inizio o in mezzo
    - Elementi in difetto
  - Tipi di dato errati
    - Caratteri, stringhe, interi, reali
  - Errori di coerenza interna

6

## Esempio di file corretto

bancomat.txt

```
3
Aldo 123456789 12762
Giovanni 334422445 97864
Giacomo 887868083 32552
```

7

## Definizioni

```
const char nomefile[] = "banco.txt" ;
const int MAX = 20 ;
/* numero max di bancomat */
const int LUN = 15 ;
/* lunghezza del nome */

int N ;
char nome[MAX][LUN+1] ;
int numero[MAX] ;
int pin[MAX] ;

FILE * f ;
int i ;
```

banco-bad.c

## Lettura del file (solo se corretto)

```
f = myfopen(nomefile, "r") ;
fscanf(f, "%d", &N) ;
for(i=0; i<N; i++)
{
 fscanf(f, "%s %d %d",
 nome[i], &numero[i], &pin[i]) ;
}
myfclose(f) ;
```



banco-bad.c

## Possibili errori nel file (1/3)

```
3
Aldo 123456789 12762
334422445 97864
Giacomo 887868083 32552
```

Campo mancante

```
3
Aldo 3212 123456789 12762
Giovanni 334422445 97864
Giacomo 887868083 32552
```

Campo extra nella  
riga

11

## Possibili errori nel file (1/3)

```
3
Aldo 123456789 12762
334422445 97864
Giacomo 887868083 32552
```

Campo mancante

10

## Possibili errori nel file (2/3)

```
3
Aldo 123456789 12762 3212
Giovanni 334422445 97864
Giacomo 887868083 32552
```

Campo extra a  
fine riga

12

## Possibili errori nel file (2/3)

```
3
Aldo 123456789 12762 3212
Giovanni 334422445 97864
Giacomo 887868083 32552
```

Campo extra a fine riga

```
3
Aldo 123456789 12762
Giovanni 334422445 97864
Giacomo A32Z4324 32552
```

Tipi di dato errati

13

## Possibili errori nel file (3/3)

```
3
Pier Aldo 123456789 12762
Giovanni 334422445 97864
Giacomo 887868083 32552
```

Spazi

```
3
Pier Aldo 123456789 12762
Giovanni 334422445 97864
Giacomo 887868083 32552
```

Spazi

```
3
Aldo 123456789 12762
Giacomo 887868083 32552
```

Incoerenza interna

15

## Osservazioni

- ▶ Di fronte a qualsiasi errore di formato, la funzione `fscanf`
  - Perde il “sincronismo” con le righe del file
  - “Blocca” la lettura in caso di stringhe incontrate quando si aspettava un numero
  - Non si “accorge” dell’End-of-File
- ▶ La funzione `fscanf` non è sufficientemente robusta per gestire la lettura da file di testo

16

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

#define MAXPAROLA 30
#define MAXRIGA 80

int main(int argc, char *argv[])
{
 int lungMAXPAROLA; /* vettore di dimensioni
 delle massime lunghezze delle parole */
 char parolaAXX[AXX];
 int i, indice, lungParola;
 FILE *f;

 if(argc > 1)
 lungMAXPAROLA = atoi(argv[1]);
 else
 lungMAXPAROLA = 10;

 f = fopen("dati.txt", "r");
 if(f == NULL)
 {
 printf("ERRORE: impossibile aprire il file dati\n");
 exit(1);
 }

 for(i=0; i<lungMAXPAROLA; i++)
 {
 lungParola = fscanf(f, "%s", parolaAXX);
 if(lungParola == 1)
 printf("%s\n", parolaAXX);
 else
 printf("ERRORE: impossibile leggere la parola %d\n", i+1);
 }

 printf("fine lettura dati\n");
 MAXRIGA = NULL;
}
```

## Input robusto

## Soluzione basata su `fgetc`

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

#define MAXRIGA 80
#define MAXPAROLA 30
```

## Lettura basata su `fgetc`

- ▶ Dovendo evitare l’utilizzo della funzione `fscanf`, si potrebbe optare per la funzione `fgetc`
- ▶ L’adozione di `fgetc` risolve i problemi di sincronizzazione e di lettura dei dati errati, ma introduce spesso una complessità eccessiva nel programma

18

## Soluzioni basate su fgetc (1/4)

- » Acquisizione di una stringa

```
char s[MAX] ;

i = 0 ;
ch = fgetc(f) ;
while(ch != EOF && ch != '\n'
 && ch != ' ' && i < MAX-1)
{
 s[i] = ch ;
 i++ ;
 ch = fgetc(f) ;
}

s[i] = 0 ; /* terminatore nullo */
```

## Soluzioni basate su fgetc (3/4)

- » Acquisizione di un intero positivo

```
char s[MAX] ;

i = 0 ;
ch = fgetc(f) ;
while(ch != EOF && isdigit(ch)
 && i < MAX-1)
{
 s[i] = ch ;
 i++ ;
 ch = fgetc(f) ;
}
s[i] = 0 ; /* terminatore nullo */

x = atoi(s) ; /* converti in int */
```

## Soluzioni basate su fgetc (2/4)

- » Saltare tutti gli spazi (ma non gli a-capo)

```
ch = fgetc(f) ;
while(ch != EOF && ch != '\n' &&
 ch == ' ')
{
 ch = fgetc(f) ;
}
```

## Soluzioni basate su fgetc (4/4)

- » Sono possibili tutti i controlli personalizzati, su
  - Lunghezza minima e massima dei campi
  - Tipo di caratteri permessi
- » Alcuni tipi di dati sono complessi da acquisire
  - Intero relativo: -124
  - Numero reale: -3.14e+21
- » Soluzione in generale completa, ma molto lavoro manuale
- » Rischio: dimenticare alcuni casi particolari

22

## Suggerimento

- » Utilizzare la funzione fgetc quando occorre leggere dei testi in "formato libero"
  - Esempio: statistiche sul testo
  - Esempio: file di una sola riga
- » Per file in formato custom, contenenti campi prefissati e/o di tipo numerico, occorre una soluzione più comoda

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

#define MAXPAROLA 30
#define MAXRIGA 80

int main(int argc, char *argv){
 int lungMAXPAROLA; /* valore di costante
 deve coincidere con lunghezza delle parole */
 char ciaoAXXIGAL;
 int i, indice, lunghezza ;
 FILE *f;

 for(i=0;i<MAXPAROLA;i++)
 lungMAXPAROLA++;

 f=fopen(argv[1],"r");
 if(f==NULL){
 perror("ERRORE: impossibile aprire il file %s", argv[1]);
 exit(1);
 }

 printf("%d\n", lungMAXPAROLA);
}
```

## Input robusto

## Funzione sscanf

## Funzione sscanf

- La risposta a molti dei problemi sollevati viene da una nuova funzione di libreria: sscanf
- Tale funzione si può usare per analizzare il contenuto di una stringa, estraendone vari campi e memorizzandoli in variabili distinte
- Ha tutta la funzionalità di scanf e fscanf, ma lavora soltanto all'interno dei caratteri contenuti in una stringa
  - Potente e sicura

25

## sscanf: sintassi

```
char str[80] ;
sscanf(str, "formato", &variabili) ;
```

Stringa di caratteri  
Puntatori alle variabili da leggere  
Formato dei dati da leggere, usando gli stessi specificatori validi per scanf

26

## Esempio

```
char str[80] ;
char nome[80] ;
int numero, pin ;

strcpy(str, "Aldo 91243213 1234\n");
sscanf(str, "%s %d %d",
 nome, &numero, &pin) ;
```

27

## Gestione degli errori

- La funzione sscanf non potrà mai leggere le righe successive di un file, in quanto la sua visibilità è confinata alla stringa passata
- Gli eventuali campi in eccesso a fine riga vengono quindi ignorati automaticamente
- Gli eventuali campi mancanti o di formato errato causano il mancato riconoscimento di quelli successivi
  - Condizione di errore facile da verificare analizzando il valore di ritorno di sscanf

28

## Valore di ritorno

- La funzione sscanf restituisce al chiamante un valore intero:
  - Il valore è pari al numero di argomenti (specificatori %) correttamente riconosciuti e memorizzati nelle rispettive variabili

```
r = sscanf(str, "%s %d %d",
 nome, &numero, &pin) ;
```

29

## Esempio

```
char str[80] ;
char nome[80] ;
int numero, pin ;
int r ;

strcpy(str, "Aldo 91243213 1234\n");

r = sscanf(str, "%s %d %d",
 nome, &numero, &pin) ;

if(r != 3)
{ ... errore ... }
```



## Suggerimenti

- » Utilizzare **sempre** `sscanf` per analizzare una stringa
- » Controllare **sempre** il valore di ritorno
- » Non utilizzare più la funzione `atoi`, sostituirla con `sscanf(... "%d" ...)`
- » Per acquisire dati da tastiera, combinare con `gets`
- » Per acquisire dati da file, combinare con `fgets`
- » Nella prossima lezione vedremo come "istruire" `sscanf` a riconoscere formati più complessi

31

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

#define MAXPAROLA 30
#define MAXRIGA 80

int main(int argc, char *argv[])
{
 int lunghezzaParola; /* > valore di controllo delle frequenze delle lunghezze delle parole */
 int lunghezzaRiga; /* > lunghezza riga */
 int i, lettura, lunghezza = 0;
 FILE *f;
 char parola[MAXPAROLA], riga[MAXRIGA];
 int nlinee = 0;

 if(argc != 2)
 {
 fprintf(stderr, "ERRORE: serve un parametro: nome del file\n");
 exit(1);
 }
 f = fopen(argv[1], "r");
 if(f == NULL)
 {
 fprintf(stderr, "ERRORE: impossibile aprire il file %s", argv[1]);
 exit(1);
 }

 while(fgets(riga, MAXRIGA, f) != NULL)
 {
 lunghezzaRiga = strlen(riga);
 if(lunghezzaRiga > lunghezza)
 lunghezza = lunghezzaRiga;
 for(i = 0; i < lunghezzaRiga; i++)
 {
 lettura = isalpha(riga[i]);
 if(lettura)
 parola[lunghezzaParola] = riga[i];
 else
 parola[lunghezzaParola] = '\0';
 lunghezzaParola++;
 if(lunghezzaParola == MAXPAROLA)
 break;
 }
 if(lunghezzaParola > lunghezza)
 lunghezza = lunghezzaParola;
 lunghezzaParola = 0;
 nlinee++;
 }
 printf("lunghezza max parola: %d\n", lunghezza);
 printf("nlinee: %d\n", nlinee);
 fclose(f);
}
```

## Input robusto

## Soluzione basata su fgets

## Input robusto da file di testo

- » Affidiamo diversi ruoli alle varie funzioni
- » `fgets`
  - Lettura del file riga per riga
  - Limite alla lunghezza max delle righe
  - Riconoscimento End-of-File
- » `sscanf`
  - Analisi dei campi presenti in una riga
  - Controllo della correttezza del formato
  - Trasferimento nelle variabili/vettori del programma

33

## Schema consigliato

```
const int LUNRIGA = 200 ;
int r, nr ;
char riga[LUNRIGA+1] ;

f = myfopen(nomefile, "r") ;
/* ciclo di lettura */
myfclose(f) ;
```

34

## Ciclo di lettura

```
nr = 0 ;
while(fgets(riga, LUNRIGA, f)!=NULL)
{
 r = sscanf(riga, "%s %d %d",
 nome, &numero, &pin) ;
 if(r == 3)
 {
 /* ...elabora la riga... */
 }
 else
 printf("Riga %d ignorata\n", nr+1);
 nr++ ;
}
```

35

## Soluzione corretta (1/6)

```
const char nomefile[]="banco.txt";
const int MAX = 20 ;
const int LUN = 15 ;
const int LUNRIGA = 200 ;

int N ;
char nome[MAX][LUN+1] ;
int numero[MAX] ;
int pin[MAX] ;

FILE * f ;
int i, r, nr ;
char riga[LUNRIGA+1] ;
```



banco-ok.c

## Soluzione corretta (2/6)

```
f = myfopen(nomefile, "r") ;
if(fgets(riga, LUNRIGA, f)==NULL)
{
 printf("Errore: file vuoto\n") ;
 exit(1) ;
}
r = sscanf(riga, "%d", &N) ;
if(r!=1)
{
 printf("Errore: La prima riga "
 "non contiene il numero\n");
 exit(1) ;
}
```

banco-ok.c

## Soluzione corretta (3/6)

```
if(N<1 || N>MAX)
{
 printf("Errore: Num. bancomat "
 "%d non valido\n", N) ;
 printf("Valori ammessi: "
 "da 1 a %d\n", MAX) ;
 exit(1) ;
}
```

banco-ok.c

## Soluzione corretta (4/6)

```
i = 0 ;
nr = 0 ;
while(fgets(riga, LUNRIGA, f)
 != NULL)
{
 if(i==N)
 {
 printf("Errore: troppe "
 "righe nel file\n") ;
 exit(1) ;
 }
}
```

banco-ok.c

## Soluzione corretta (5/6)

```
r = sscanf(riga, "%s %d %d",
 nome[i], &numero[i], &pin[i]) ;

if(r == 3)
 i++ ;
else
{
 printf("Riga %d ignorata\n",
 nr) ;
}
nr++ ;
```

banco-ok.c

## Soluzione corretta (6/6)

```
if(i != N)
{
 printf("Errore: poche righe "
 "nel file\n") ;
 exit(1) ;
}

myfclose(f) ;
```

banco-ok.c

## Conclusioni

- Prevedere tutti i possibili errori è difficile e pesante
  - La maggior parte delle linee di codice è dedicata alla gestione degli errori o delle anomalie
- Gli strumenti offerti dalla "coppia" fgets + sscanf sono validi ed efficaci

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

#define MAXPAROLA 30
#define MAXIGRA 80

int main(int argc, char *argv[])
{
 int freqMAXPAROLA; /* valore di controllo delle frequenze delle parole */
 char *freq[MAXIGRA]; /* frequenze delle parole */
 int i, indice, longhezza;
 FILE *fptr;

 if(argc > 1)
 {
 fptr=fopen(argv[1], "r");
 if(fptr==NULL)
 {
 fprintf(stderr, "ERRORE, non è possibile aprire il file %s", argv[1]);
 exit(1);
 }
 }
 else
 {
 fprintf(stderr, "ERRORE, impossibile aprire il file %s", argv[1]);
 exit(1);
 }

 while(fgets(freq[i], MAXIGRA, fptr) != NULL)
 {
 // ...
 }
}
```

## I/O Avanzato e File

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

#define MAXPAROLA 30
#define MAXIGRA 80

int main(int argc, char *argv[])
{
 int freqMAXPAROLA; /* valore di controllo delle frequenze delle parole */
 char *freq[MAXIGRA]; /* frequenze delle parole */
 int i, indice, longhezza;
 FILE *fptr;

 if(argc > 1)
 {
 fptr=fopen(argv[1], "r");
 if(fptr==NULL)
 {
 fprintf(stderr, "ERRORE, non è possibile aprire il file %s", argv[1]);
 exit(1);
 }
 }
 else
 {
 fprintf(stderr, "ERRORE, impossibile aprire il file %s", argv[1]);
 exit(1);
 }

 while(fgets(freq[i], MAXIGRA, fptr) != NULL)
 {
 // ...
 }
}
```

## Formattazione avanzata

```
%{argc}(%3)
%{width}(%3)
%{precision}(%3)
%{format}(%3)

int main()
{
 printf("TESTING, inserire un perimetro da calcolare il numero del file (%s)\n", argv[1]);
 if(argc > 1)
 {
 width = atoi(argv[1]);
 if(width < 0)
 {
 printf("ERRORE, inserire un perimetro positivo (%d)\n", width);
 exit(1);
 }
 }
 else
 {
 printf("ERRORE, inserire un perimetro (%d)\n", width);
 exit(1);
 }
}
```

## Specificatori di formato

| Tipo               | printf   |
|--------------------|----------|
| char               | %c %d    |
| int                | %d       |
| short int          | %hd %d   |
| long int           | %ld      |
| unsigned int       | %u %o %x |
| unsigned short int | %hu      |
| unsigned long int  | %lu      |
| float              | %f %e %g |
| double             | %f %e %g |
| char []            | %s       |

5



## Formattazione avanzata

- Modificatori di formato in output
- Modificatori di formato in input
- Stream predefiniti

2



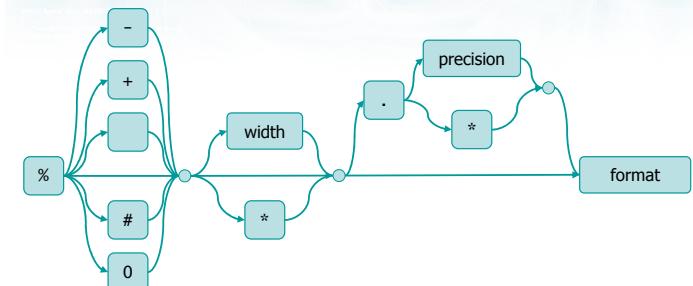
## Formattazione dell'output

- L'output (su schermo o su file) viene formattato solitamente mediante la funzione printf (o fprintf)
- Ogni dato viene stampato attraverso un opportuno specificatore di formato (codici %)
- Ciascuno di questi codici dispone di ulteriori opzioni per meglio controllare la formattazione
  - Stampa incolonnata
  - Numero di cifre decimali
  - Spazi di riempimento
  - ...

4

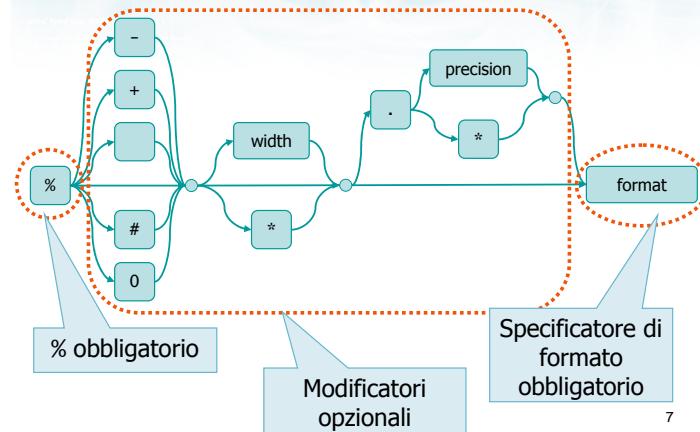


## Forma completa degli specificatori



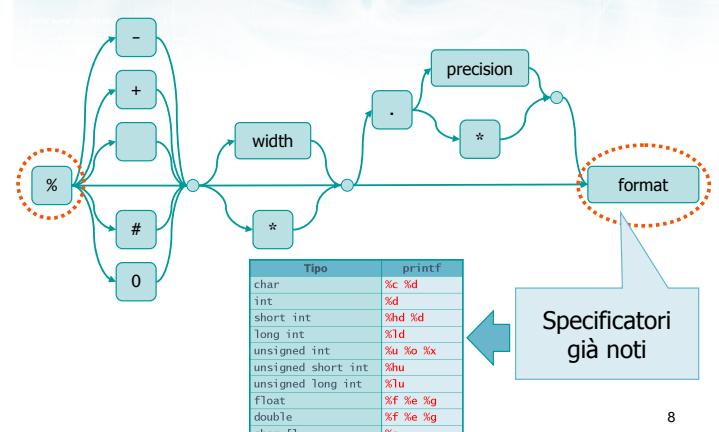
6

## Forma completa degli speciatori



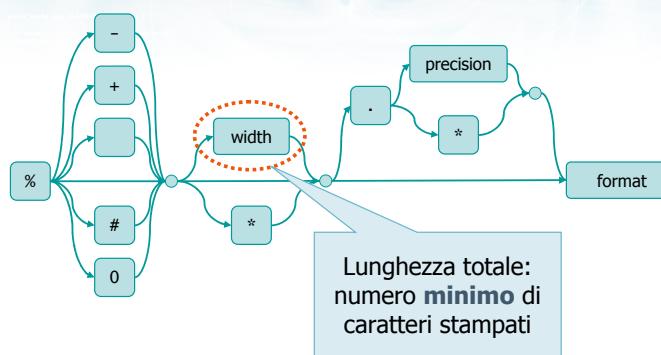
7

## Forma completa degli speciatori



8

## Forma completa degli speciatori



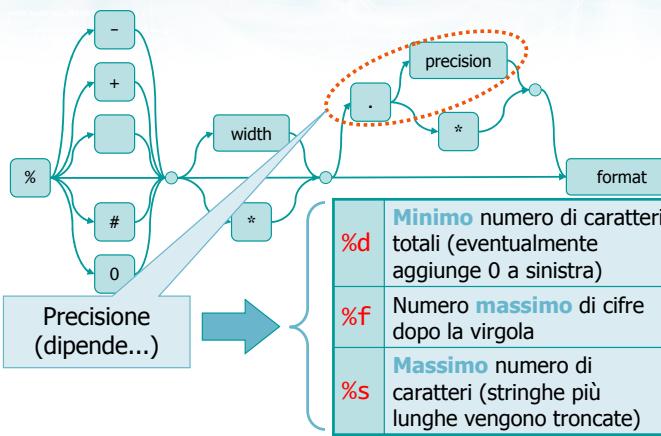
9

## Esempi

| Istruzione             | Risultato  |
|------------------------|------------|
| printf("%d", 13);      | 13         |
| printf("%1d", 13);     | 13         |
| printf("%3d", 13);     | _13        |
| printf("%f", 13.14);   | 13.140000  |
| printf("%6f", 13.14);  | 13.140000  |
| printf("%12f", 13.14); | _13.140000 |
| printf("%6s", "ciao"); | _ciao      |

10

## Forma completa degli speciatori



12

## Esempi (1/2)

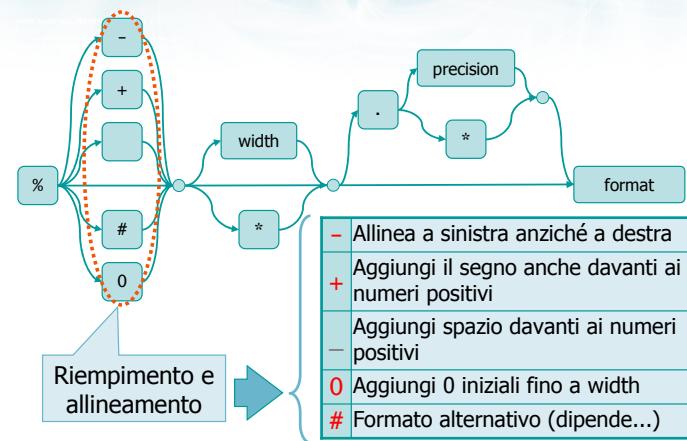
| Istruzione               | Risultato |
|--------------------------|-----------|
| printf("%.1d", 13);      | 13        |
| printf("%.4d", 13);      | 0013      |
| printf("%6.4d", 13);     | _0013     |
| printf("%4.6d", 13);     | 000013    |
| printf("%.2s", "ciao");  | ci        |
| printf("%.6s", "ciao");  | ciao      |
| printf("%6.3s", "ciao"); | _cia      |

## Esempi (2/2)

| Istruzione                | Risultato |
|---------------------------|-----------|
| printf("% .2f", 13.14) ;  | 13.14     |
| printf("% .4f", 13.14) ;  | 13.1400   |
| printf("%6 .4f", 13.14) ; | 13.1400   |
| printf("%9 .4f", 13.14) ; | _13.1400  |

13

## Forma completa degli specificatori



## Esempi (1/2)

| Istruzione               | Risultato |
|--------------------------|-----------|
| printf("%6d", 13) ;      | ___13     |
| printf("%-6d", 13) ;     | 13___     |
| printf("%06d", 13) ;     | 000013    |
| printf("%6s", "ciao") ;  | _ciao     |
| printf("%-6s", "ciao") ; | ciao_     |

15

## Esempi (2/2)

| Istruzione           | Risultato |
|----------------------|-----------|
| printf("%d", 13) ;   | 13        |
| printf("%d", -13) ;  | -13       |
| printf("%+d", 13) ;  | +13       |
| printf("%+d", -13) ; | -13       |
| printf("% d", 13) ;  | _13       |
| printf("% d", -13) ; | -13       |

16

```

#include <stdio.h>
#include <string.h>
#include <ctype.h>

#define MAXPAROLA 30
#define MAXIGRA 80

int main(int argc, char *argv[])
{
 int larg(MAXPAROLA); /* vettore di dimensione larghezza della parola */
 char str(MAXIGRA); /* vettore di dimensione massima delle parole */
 int i, indice, lunghezza;
 FILE *f;

 if(argc > 1)
 {
 f = fopen(argv[1], "r");
 if(f == NULL)
 {
 printf("ERRORE: impossibile aprire il file %s", argv[1]);
 exit(1);
 }
 }
 else
 {
 strcpy(str, "C");
 larg = 1;
 }

 for(i=0; str[i] != '\0'; i++)
 {
 if(isalpha(str[i]))
 {
 str[i] = toupper(str[i]);
 }
 }

 printf("%s\n", str);
}

```

## Formattazione avanzata

### Modificatori di formato in input

## Approfondimenti su scanf

- Tipologie di caratteri nella stringa di formato
- Modificatori degli specificatori di formato
- Valore di ritorno
- Specificatore %[]

18

## Stringa di formato (1/2)

### Caratteri stampabili:

- `scanf` si aspetta che tali caratteri compaiano esattamente nell'input
  - Se no, interrompe la lettura
- Spaziatura ("whitespace"):
- Spazio, tab, a capo
  - `scanf` "salta" ogni (eventuale) sequenza di caratteri di spaziatura
  - Si ferma al primo carattere non di spaziatura (o End-of-File)

19

## Stringa di formato (2/2)

### Specificatori di formato (%-codice):

- Se il codice non è %c, innanzitutto `scanf` "salta" ogni eventuale sequenza di caratteri di spaziatura
- `scanf` legge i caratteri successivi e *cerca* di convertirli secondo il formato specificato
- La lettura si interrompe al primo carattere che non può essere interpretato come parte del campo

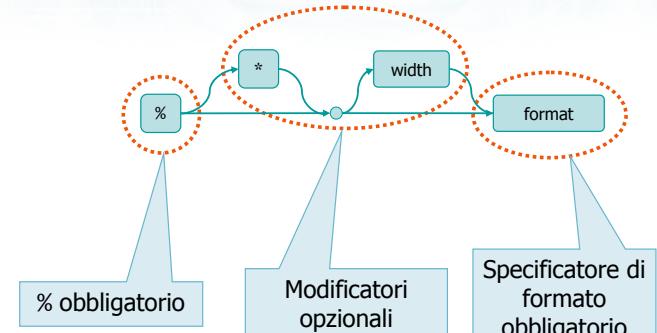
20

## Specificatori di formato

| Tipo               | <code>scanf</code> |
|--------------------|--------------------|
| char               | %c %[...]          |
| int                | %d                 |
| short int          | %hd                |
| long int           | %ld                |
| unsigned int       | %u %o %x           |
| unsigned short int | %hu                |
| unsigned long int  | %lu                |
| float              | %f                 |
| double             | %lf                |
| char []            | %s %[...]          |

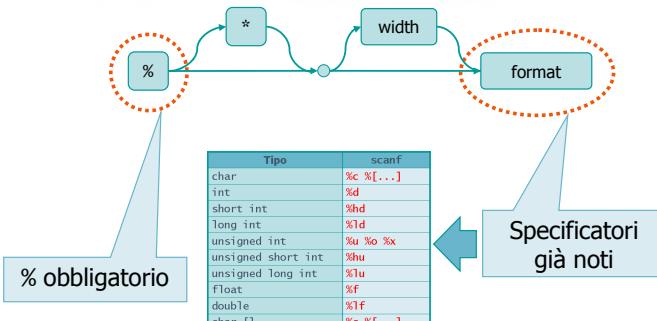
21

## Forma completa degli specificatori



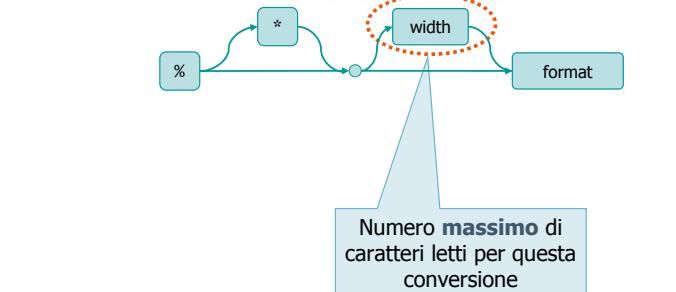
22

## Forma completa degli specificatori



23

## Forma completa degli specificatori



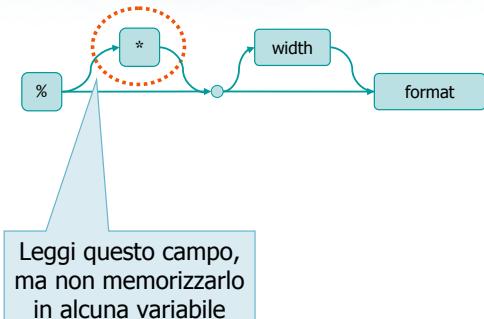
24

## Esempi

| Istruzione         | Input  | Risultato    |
|--------------------|--------|--------------|
| scanf("%d", &x) ;  | 134xyz | x = 134      |
| scanf("%2d", &x) ; | 134xyz | x = 13       |
| scanf("%s", v) ;   | 134xyz | v = "134xyz" |
| scanf("%2s", v) ;  | 134xyz | v = "13"     |

25

## Forma completa degli specificatori



26

## Esempi

| Istruzione              | Input    | Risultato                 |
|-------------------------|----------|---------------------------|
| scanf("%d %s", &x, v) ; | 10 Pippo | x = 10<br>v = "Pippo"     |
| scanf("%s", v) ;        | 10 Pippo | x immutato<br>v = "10"    |
| scanf("%*d %s", v) ;    | 10 Pippo | x immutato<br>v = "Pippo" |

27

## Valore di ritorno

- ▶ La funzione scanf ritorna un valore intero:
  - Numero di elementi (%) effettivamente letti
    - Non conta quelli "saltati" con %\*
    - Non conta quelli non letti perché l'input non conteneva i caratteri desiderati
    - Non conta quelli non letti perché l'input è finito troppo presto
      - End-of-File per fscanf
      - Fine stringa per sscanf
  - EOF se l'input era già in condizione End-of-File all'inizio della lettura

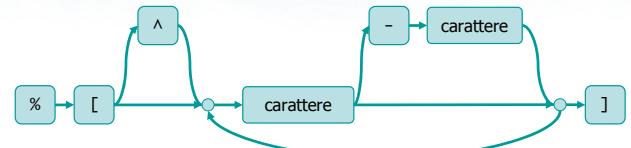
28

## Lettura di stringhe

- ▶ La lettura di stringhe avviene solitamente con lo specificatore di formato %s
  - Salta tutti i caratteri di spaziatura
  - Acquisisci tutti i caratteri seguenti, fermandosi al primo carattere di spaziatura (senza leggerlo)
- ▶ Qualora l'input dei separatori diversi da spazio, è possibile istruire scanf su quali siano i caratteri leciti, mediante lo specificatore %[pattern]

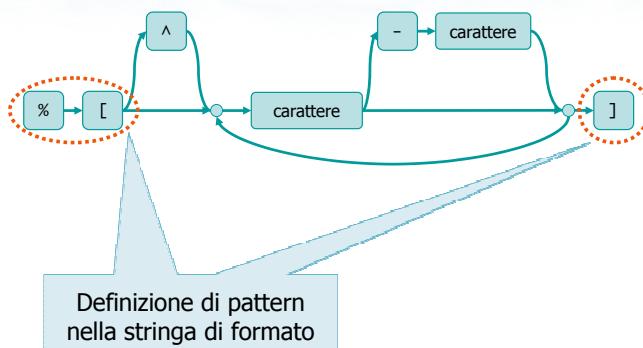
29

## Struttura di un pattern



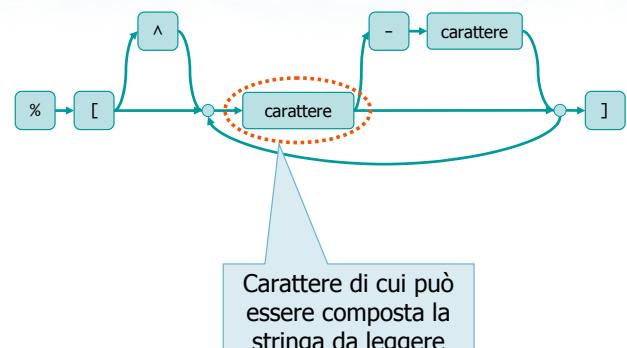
30

## Struttura di un pattern



31

## Struttura di un pattern



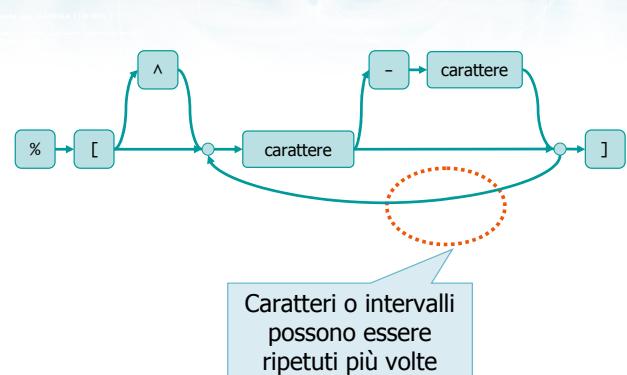
32

## Esempi

| Pattern | Effetto                    |
|---------|----------------------------|
| %[r]    | Legge solo sequenze di 'r' |

33

## Struttura di un pattern



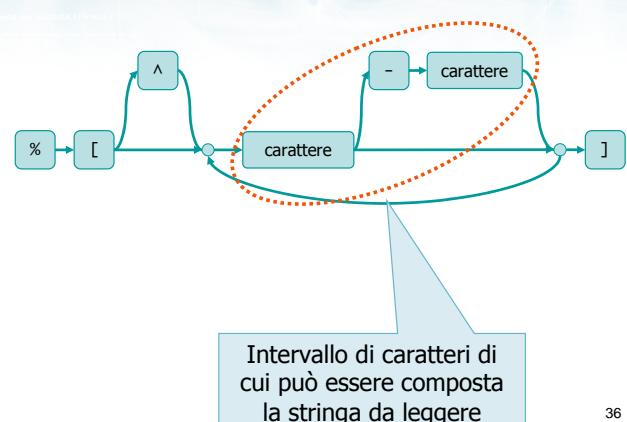
34

## Esempi

| Pattern   | Effetto                                                                                   |
|-----------|-------------------------------------------------------------------------------------------|
| %[r]      | Legge solo sequenze di 'r'                                                                |
| %[abcABC] | Legge sequenze composte da a, b, c, A, B, C, in qualsiasi ordine e di qualsiasi lunghezza |

35

## Struttura di un pattern



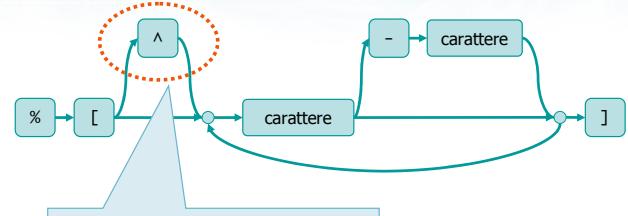
36

## Esempi

| Pattern      | Effetto                                                                                   |
|--------------|-------------------------------------------------------------------------------------------|
| %[r]         | Legge solo sequenze di ' r '                                                              |
| %[abcABC]    | Legge sequenze composte da a, b, c, A, B, C, in qualsiasi ordine e di qualsiasi lunghezza |
| %[a-cA-C]    | Idem come sopra                                                                           |
| %[a-zA-Z]    | Sequenze di lettere alfabetiche                                                           |
| %[0-9]       | Sequenze di cifre numeriche                                                               |
| %[a-zA-Z0-9] | Sequenze alfanumeriche                                                                    |

37

## Struttura di un pattern



Pattern "invertito": i caratteri specificati **non** devono comparire nella stringa

38

## Esempi

| Pattern      | Effetto                                                                                   |
|--------------|-------------------------------------------------------------------------------------------|
| %[r]         | Legge solo sequenze di ' r '                                                              |
| %[abcABC]    | Legge sequenze composte da a, b, c, A, B, C, in qualsiasi ordine e di qualsiasi lunghezza |
| %[a-cA-C]    | Idem come sopra                                                                           |
| %[a-zA-Z]    | Sequenze di lettere alfabetiche                                                           |
| %[0-9]       | Sequenze di cifre numeriche                                                               |
| %[a-zA-Z0-9] | Sequenze alfanumeriche                                                                    |
| %[^x]        | Qualunque sequenza che non contiene ' x '                                                 |
| %[^n]        | Legge fino a file riga                                                                    |
| %[^,;.!? ]   | Si ferma alla punteggiatura o spazio                                                      |

39

## Osservazioni

- Ricordare che i pattern devono sempre essere associati a dati di tipo stringa (vettori di caratteri)
- Il comune specificatore "%s" equivale al pattern "%[^ \t\n]"

40

## Esempio

- Il file /etc/passwd, presente in tutti i sistemi operativi derivati da Unix, contiene i dati degli utenti nel seguente formato:

```
corno:w3tce34:501:401:Fulvio Corno:/home/corno:/bin/bash
```

- Campi separati da ':'
- Nome utente, password: stringhe prive di spazi
- User ID, Group ID: interi
- Nome reale: stringa generica (con spazi)
- Home directory e shell: stringhe generiche

41

## Soluzione

```
f = myfopen("/etc/passwd", "r") ;
while(fgets(riga, MAX, f)!=NULL)
{
 sscanf(riga,
 "%[:]:[%:]:%d:%d:"
 "%[:]:%[:]:%[:]",
 login, pass, &uid, &gid,
 realname, home, shell) ;
 /* elabora i dati ... */
}
myfclose(f) ;
```

42

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/conf.h>
```

## Formattazione avanzata

### Stream predefiniti

### Stream predefiniti

- L'istruzione fopen permette di aprire nuovi stream, associati a file esistenti sui dischi dell'elaboratore
- All'avvio di un programma in C, sono già stati aperti in modo automatico 3 stream predefiniti
  - **stdin**
  - **stdout**
  - **stderr**

44

### stdin

- **stdin** è detto lo "standard input" di un programma
- Normalmente è associato al canale di input del terminale (o della console) nel quale il programma è avviato
  - In pratica, la tastiera del P.C.
- L'input può essere rediretto, prendendolo da un file anziché dalla tastiera, avviando il programma da linea di comando con l'operatore <
  - prog < file.txt

45

### stdout

- **stdout** è detto lo "standard output" di un programma
- Normalmente è associato al canale di output del terminale (o della console) nel quale il programma è avviato
  - In pratica, il video del P.C.
- L'output può essere rediretto, salvandolo su un file anziché su video, avviando il programma da linea di comando con l'operatore >
  - prog > file.txt

46

### stderr

- **stderr** è detto lo "standard error" di un programma
- Normalmente è associato al canale di output del terminale (o della console) nel quale il programma è avviato
  - In pratica, il video del P.C.
- È uno stream distinto ed indipendente da stdout
- Solitamente l'output di stderr non viene rediretto, per permettere ai messaggi di errore di essere sempre visti dall'utilizzatore

47

### Uso comune

- stdin viene usato per acquisire i dati
  - Può essere rediretto da un file
- stdout viene usato per presentare i risultati
  - Può essere rediretto su un file
- stderr viene usato esclusivamente per i messaggi di errore
  - Rimane visibile sulla console

48

## Equivalenze

| La funzione...          | è equivalente a...               |
|-------------------------|----------------------------------|
| scanf("formato", ...);  | fscanf(stdin, "formato", ...);   |
| printf("formato", ...); | fprintf(stdout, "formato", ...); |
| ch=getchar();           | ch=fgetc(stdin);                 |
| putchar(ch);            | fputc(ch, stdout);               |

49

## Stampa dei messaggi di errore

```
if(...condizione errore fatale...){
 fprintf(stderr,
 "Messaggio di errore\n") ;
 exit(1) ;
}
```

50



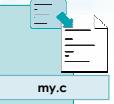
## Suggerimento

- » Tutti i messaggi di errore devono essere stampati sullo stream stderr
- » Conviene definire una funzione myerror
  - Stampa un messaggio di errore
  - Interrompe il programma

```
void myerror(char *message) ;
```

51

## Funzione myerror



```
int myerror(char *message){
 fputs(message, stderr) ;
 exit(1) ;
}
```

52

```

#include <stdio.h>
#include <string.h>
#include <ctype.h>

#define MAXPAROLA 30
#define MAXRIGA 80

int main(int argc, char *argv[])
{
 int lung(MAXPAROLA); /* valore di controllo delle frequenze delle lunghezze delle parole */
 int i, indice, lunghezza;
 FILE *fptr;
 char riga[MAXRIGA];
 char riga1[MAXRIGA];
 int nriga;
 int somma;
 if(argc != 2)
 {
 printf("Errore: l'unico argomento deve essere il nome del file da leggere\n");
 exit(1);
 }
 fptr = fopen(argv[1], "r");
 if(fptr == NULL)
 {
 printf("Errore: impossibile aprire il file %s", argv[1]);
 exit(1);
 }
 while(fgets(riga, MAXRIGA, fptr) != NULL)
 {
 if(strlen(riga) > lung)
 lung = strlen(riga);
 else if(strlen(riga) == lung)
 indice++;
 else if(strlen(riga) < lung)
 lung = strlen(riga);
 }
 printf("I/O Avanzato e File\n");
}

```

## I/O Avanzato e File

```

#include <stdio.h>
#include <string.h>
#include <ctype.h>

#define MAXPAROLA 30
#define MAXRIGA 80

int main(int argc, char *argv[])
{
 int lung(MAXPAROLA); /* valore di controllo delle frequenze delle lunghezze delle parole */
 int i, indice, lunghezza;
 FILE *fptr;
 char riga[MAXRIGA];
 char riga1[MAXRIGA];
 int nriga;
 int somma;
 if(argc != 2)
 {
 printf("Errore: l'unico argomento deve essere il nome del file da leggere\n");
 exit(1);
 }
 fptr = fopen(argv[1], "r");
 if(fptr == NULL)
 {
 printf("Errore: impossibile aprire il file %s", argv[1]);
 exit(1);
 }
 while(fgets(riga, MAXRIGA, fptr) != NULL)
 {
 if(strlen(riga) > lung)
 lung = strlen(riga);
 else if(strlen(riga) == lung)
 indice++;
 else if(strlen(riga) < lung)
 lung = strlen(riga);
 }
 printf("Esercizi proposti\n");
}

```

## Esercizi proposti

### Esercizi proposti

## Esercizi proposti

- Esercizio “Somma numeri”
- Esercizio “Bersagli”
- Esercizio “Consumi toner”

2

```

#include <stdio.h>
#include <string.h>
#include <ctype.h>

#define MAXPAROLA 30
#define MAXRIGA 80

int main(int argc, char *argv[])
{
 int lung(MAXPAROLA); /* valore di controllo delle frequenze delle lunghezze delle parole */
 int i, indice, lunghezza;
 FILE *fptr;
 char riga[MAXRIGA];
 char riga1[MAXRIGA];
 int nriga;
 int somma;
 if(argc != 2)
 {
 printf("Errore: l'unico argomento deve essere il nome del file da leggere\n");
 exit(1);
 }
 fptr = fopen(argv[1], "r");
 if(fptr == NULL)
 {
 printf("Errore: impossibile aprire il file %s", argv[1]);
 exit(1);
 }
 while(fgets(riga, MAXRIGA, fptr) != NULL)
 {
 if(strlen(riga) > lung)
 lung = strlen(riga);
 else if(strlen(riga) == lung)
 indice++;
 else if(strlen(riga) < lung)
 lung = strlen(riga);
 }
 printf("Esercizio \"Somma numeri\"");
}

```

### Esercizio “Somma numeri”

- Un file di testo contiene una serie di numeri interi (positivi o negativi), uno per riga
- Si scriva un programma C che:
  - Acquisisca da linea di comando il nome del file
  - Calcoli la somma di tutti i numeri presenti nel file
  - Stampa in output il valore di tale somma

4

### Esercizio “Somma numeri”

#### Analisi

```

#include <stdio.h>
#include <string.h>
#include <ctype.h>

FILE *fptr;
char nomefile[MAX] ;
char riga[MAX] ;
int r, num ;
int somma ;

int main()
{
 FILE *fptr;
 char nomefile[MAX];
 char riga[MAX];
 int r, num;
 int somma;
 if(argc != 2)
 {
 printf("Errore: l'unico argomento deve essere il nome del file da leggere\n");
 exit(1);
 }
 fptr = fopen(nomefile, "r");
 if(fptr == NULL)
 {
 printf("Errore: impossibile aprire il file %s", nomefile);
 exit(1);
 }
 while(fgets(riga, MAX, fptr) != NULL)
 {
 if(riga[0] == '-')
 num = -1 * atoi(riga + 1);
 else
 num = atoi(riga);
 somma += num;
 }
 printf("La somma vale: %d", somma);
}

```

#### Soluzione (1/4)

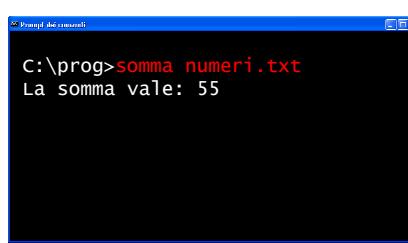
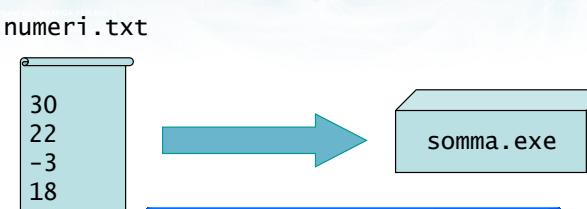
```

int main(int argc, char *argv[])
{
 const int MAX = 80 ;

 FILE * f ;
 char nomefile[MAX] ;
 char riga[MAX] ;
 int r, num ;

 int somma ;

```



5

## Soluzione (2/4)

```
if(argc != 2)
 myerror("Num. argomenti errato\n") ;

strcpy(nomefile, argv[1]) ;
f = myopen(nomefile, "rt") ;
somma = 0 ;
```

sommofile.c

## Soluzione (3/4)

```
while(fgets(riga, MAX, f) != NULL)
{
 r = sscanf(riga, "%d", &num) ;

 if(r==1)
 somma = somma + num ;
 else
 printf("Riga ignorata\n") ;
}
```

sommofile.c

## Soluzione (4/4)

```
myfclose(f) ;

printf("La somma vale: %d\n", somma) ;
exit(0) ;
}
```

sommofile.c

#include <stdio.h>
#include <string.h>
#include <ctype.h>

#define MAXPAROLA 30
#define MAXRIGA 80

```
int main(int argc, char *argv[])
{
 int lungMAXPAROLA; /* variezza di parola */
 char riga[MAXRIGA]; /* riga inserita dall'utente */
 char parola[MAXRIGA];
 int i, j, lungparola;
 FILE *f;

 for(;;) {
 if(argc > 1) {
 if(strcmp(argv[1], "STOP") == 0) {
 exit(0);
 }
 }
 if(fgets(riga, MAXRIGA, f) == NULL) {
 if(ferror(f)) {
 perror("ERRORE, impossibile aprire il file");
 exit(1);
 }
 }
 while((riga[lungparola] != '\n') && (lungparola < MAXRIGA - 1))
 lungparola++;
 if(lungparola >= MAXRIGA - 1) {
 printf("Riga troppo lunga\n");
 exit(1);
 }
 }
}
```

## Esercizi proposti

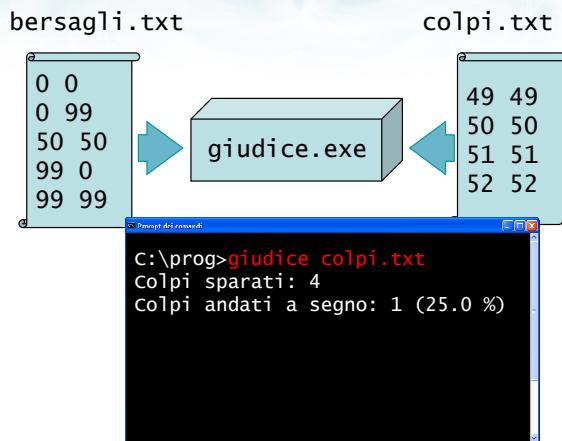
### Esercizio “Bersagli”

#### Esercizio “Bersagli” (1/2)

- Si desidera creare un programma in grado di calcolare il numero di colpi andati a segno in un'esercitazione di tiro
- I bersagli sono descritti tramite le coordinate cartesiane del punto in cui sono posizionati all'interno di una griglia  $100 \times 100$ . Le coordinate sono rappresentate solo da numeri interi, compresi tra 0 e 99. La posizione dei bersagli è contenuta nel file di testo bersagli1.txt: ogni riga di tale file contiene le coordinate X e Y di un singolo bersaglio

#### Esercizio “Bersagli” (2/2)

- I colpi sparati sono descritti anch'essi tramite le loro coordinate X e Y e sono memorizzati in un file di caratteri il cui nome è passato come primo parametro sulla linea di comando. Ogni riga di tale file contiene le coordinate X e Y del punto in cui è stato sparato un colpo
- Si scriva un programma che legga dai file succitati la posizione dei bersagli ed i colpi sparati e quindi calcoli il numero di colpi andati a segno, sia come valore assoluto sia come percentuale dei colpi sparati



13

- Acquisire dal file `bersagli.txt` tutte le coordinate, memorizzandole in due vettori paralleli `Bx[]` e `By[]`. Lunghezza dei vettori: `Nb`
- Acquisire dal file `argv[1]` le coordinate dei vari colpi `Cx, Cy`. Numero colpi: `Nc`
  - Per ciascun colpo, verificare se le coordinate coincidono con quelle di almeno un bersaglio
  - Se sì, incrementare `Ncc`
- Stampare `Ncc` e  $Ncc/Nc \times 100$

14

**Soluzione (1/4)**

```
int main(int argc, char *argv[])
{
 const int MAXB = 100 ;
 /* massimo numero di bersagli */
 const int MAX = 80 ;
 /* lunghezza riga del file */
 const char FILEB[] = "bersagli.txt";
 int Nb ; /* numero di bersagli */
 int Bx[MAXB], By[MAXB] ;
 /* coordinate dei bersagli */

 int NC ; /* numero colpi sparati */
 int Ncc ; /* numero di colpi centrati */
}
```

**Soluzione (3/4)**

```
/* 2: analisi coordinate dei colpi */
if(argc != 2)
 myerror("ERR: manca nome file\n");
f = myfopen(argv[1], "rt");
Nc = 0 ;
Ncc = 0 ;
while(fgets(riga, MAX, f) != NULL)
{
 r = sscanf(riga, "%d %d", &Cx, &Cy);
 if(r!=2) myerror("Formato errato\n");
 NC ++ ;
 /* Ricerca del bersaglio */
}
myfclose(f);
```

**Soluzione (2/4)**

```
FILE *f ;
char riga[MAX] ;
int Cx, Cy ;
int i, r, trovato ;

/* 1: acquisizione coordinate bersagli */
f = myfopen(FILEB, "rt");
Nb = 0 ;
while(fgets(riga, MAX, f) != NULL)
{
 r=sscanf(riga,"%d %d",&Bx[Nb],&By[Nb]);
 if(r!=2)
 myerror("Formato errato\n");
 Nb ++ ;
}
myfclose(f);
```

**Soluzione (3/4)**

```
/* 2: analisi coordinate dei colpi */
trovato = 0 ;
for(i=0; i<Nb && trovato==0; i++)
 if(Cx==Bx[i] && Cy==By[i])
 trovato = 1 ;
 if(trovato==1)
 Ncc ++ ;
 NC ++ ;
 /* Ricerca del bersaglio */
}
myfclose(f);
```



## Soluzione (4/4)

```
/* 3: stampa risultati */
printf("Colpi sparati: %d\n", Nc) ;
printf("Colpi andati a segno: %d ", Ncc);
if(Nc!=0)
 printf("%.2f%%", Ncc*100.0/Nc) ;
printf("\n");

exit(0) ;
```

bersagli.c

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

#define MAXPAROLA 30
#define MAXSIG 60

int main(int argc, char *argv[])
{
 int freq[MAXPAROLA]; /* vettore di contatori delle frequenze delle parole */
 int i, maxfreq = 0; /* i=indice, maxfreq=0 */
 FILE *fp; /* apertura del file */

 if(argc != 2)
 fprintf(stderr, "ERRORE: non è possibile indicare il nome del file (%s)\n", argv[0]);
 else {
 fp = fopen(argv[1], "r");
 if(fp == NULL)
 fprintf(stderr, "ERRORE: impossibile aprire il file %s (%s)\n", argv[1], strerror(errno));
 else {
 /* legge ogni riga da file */
 for(i = 0; i < MAXSIG; i++)
 freq[i] = 0;
 while(fgets(argv[i], MAXSIG, fp) != NULL) {
 /* pulisce la riga */
 argv[i] = strcchr(argv[i], '\n');
 if(argv[i] != NULL)
 *argv[i] = '\0';
 /* calcola la parola più frequente */
 for(i = 0; i < MAXSIG; i++)
 if(freq[i] < freq[i+1])
 freq[i] = freq[i+1];
 }
 /* stampa le parole più frequenti */
 for(i = 0; i < MAXSIG; i++)
 if(freq[i] > maxfreq)
 maxfreq = freq[i];
 for(i = 0; i < MAXSIG; i++)
 if(freq[i] == maxfreq)
 printf("%s\t%d\n", argv[i], freq[i]);
 }
 }
}
```

## Esercizi proposti

### Esercizio “Consumi toner”

#### Esercizio “Consumi toner” (1/3)

- » Si desidera analizzare la statistica dei consumi di toner di un’azienda per ottimizzare gli acquisti futuri
- » La quantità di cartucce di toner prelevate dal magazzino ogni giorno è riportata all’interno di un file di testo il cui nome è passato come primo parametro sulla riga di comando

21

#### Esercizio “Consumi toner” (2/3)

- » Il file contiene una riga per ogni giorno. Ogni riga contiene in sequenza:
  - Il nome del dipartimento che ha prelevato il toner (una stringa lunga al massimo 5 caratteri)
  - Un numero intero (valore minimo 1 e massimo 99) che indica la quantità di cartucce di toner prelevate in quel giorno da quel dipartimento
- » Non è noto il numero di righe presenti nel file

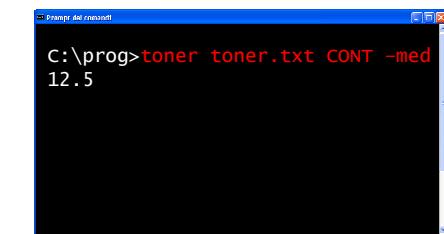
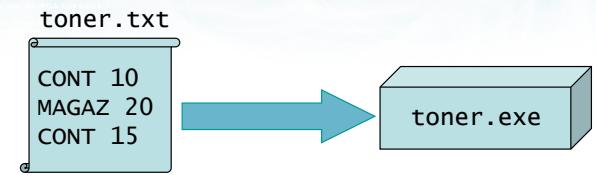
22

#### Esercizio “Consumi toner” (3/3)

- » Il programma riceve inoltre come secondo argomento sulla linea di comando il nome di un dipartimento per il quale calcolare l’indicatore statistico dato come terzo argomento sulla linea di comando secondo la seguente codifica:
  - -min indica che si desidera il valore minimo
  - -max indica che si desidera il valore massimo
  - -med indica che si desidera il valore medio (da stamparsi in output con un cifra dopo la virgola)

23

## Analisi



24

## Argomenti del programma

```
C:\prog>toner toner.txt CONT -med
```

argv[1]  
Nome del file  
contenente i  
consumi

argv[2]  
Dipartimento  
da analizzare

argv[3]  
Operazione  
statistica:  
-min  
-med  
-max

25

## Soluzione (1/4)

```
int main(int argc, char *argv[])
{
 const int LUNDIP = 5 ;
 const int MAX = 80 ;

 char dip[LUNDIP+1], dipf[LUNDIP+1] ;
 int stat ;
 /* tipo di statistica:
 1=min, 2=max, 3=med */
 FILE * f ;
 int qtaf, r ;
 int min, max, tot, cont ;
 char riga[MAX+1] ;
```

toner.c

## Soluzione (2/4)

```
if(argc!=4)
 myerror("Numero parametri errato\n");
/* Acquisisci il nome del dipartimento */
strcpy(dip, argv[2]) ;
/* Acquisisci tipo statistica */
if(strcmp(argv[3], "-min") == 0)
 stat = 1 ;
else if(strcmp(argv[3], "-max") == 0)
 stat = 2 ;
else if(strcmp(argv[3], "-med") == 0)
 stat = 3 ;
else
 myerror("Statistica sconosciuta\n");
```



toner.c

## Soluzione (3/4)

```
f = myfopen(argv[1], "rt") ;
tot = 0 ;
cont = 0 ;
min = 100 ;
max = 0 ;
while(fgets(riga, MAX, f) != NULL)
{
 r = sscanf(riga, "%s %d", dipf, &qtaf);
 if(r!=2)
 printf("Riga ignorata\n");
 else
 {
 /* Aggiorna statistiche */
 }
}
myfclose(f) ;
```

toner.c

## Soluzione (3/4)

```
if(strcmp(dip, dipf)==0)
{
 if(qtaf < min)
 min = qtaf ;
 if(qtaf > max)
 max = qtaf ;
 tot = tot + qtaf ;
 cont++ ;
}
else
{
 /* Aggiorna statistiche */
}
myfclose(f) ;
```



toner.c

## Soluzione (4/4)

```
/* Stampa il valore della statistica */
if(cont==0)
 printf("Nessun elemento\n");
else if(stat==1)
 printf("%d\n", min) ;
else if(stat ==2)
 printf("%d\n", max) ;
else if(stat==3)
 printf("%.1f\n", (float)tot/cont) ;
exit(0) ;
```

toner.c

```
#include <stdio.h>
#include <ctype.h>
#define MAXPOLA 30
#define MAXSIGA 80

int main(int argc, char *argv[])
{
 int freqMAXPOLA; /* valore di controllo delle frequenze delle lunghezze delle piazze */
 float f, m, l; /* frequenza, lunghezza e massa */
 FILE *fptr; /* puntatore al file */

 if(argc != 2)
 {
 fprintf(stderr, "ERRORE: non è possibile aprire il file %s", argv[1]);
 exit(1);
 }
 fptr = fopen(argv[1], "r");
 if(fptr == NULL)
 {
 fprintf(stderr, "ERRORE: impossibile aprire il file %s", argv[1]);
 exit(1);
 }

 /* legge le prime tre righe del file */
 fscanf(fptr, "%f %f %f", &freqMAXPOLA, &m, &l);

 /* legge le righe successive */
 while(fscanf(fptr, "%f %f %f", &f, &m, &l) == 3)
 {
 /* calcola la lunghezza della piazza */
 if(freqMAXPOLA <= f)
 printf("Lunghezza della piazza: %f\n", l);
 else
 printf("Lunghezza della piazza: %f\n", l * freqMAXPOLA);
 }

 /* chiude il file */
 fclose(fptr);
}
```

## I/O Avanzato e File

### Sommario

## Argomenti trattati (1/2)

### File

- File binari
  - File di testo
- Gestione dei file in C
- Apertura/chiusura
  - Lettura/scrittura
  - Gestione degli errori
  - Il problema degli errori di formattazione

2

## Argomenti trattati (2/2)

### ► Formattazione avanzata

- Funzione sscanf
  - Opzioni degli speciatori di formato
    - In output
    - In input
    - Pattern di input
  - Stream predefiniti
- Input robusto
- Utilizzo combinato di fgets e sscanf

3

## Tecniche di programmazione

- Gestire i file, in lettura e scrittura
- Verificare gli errori che possono incorrere nelle operazioni di I/O
- Utilizzare le funzioni myopen, myfclose, myerror
- Utilizzare sscanf per analizzare righe anche dal formato complesso
- Utilizzare printf/fprintf per controllare l'ampiezza dei campi di output

4

## Materiale aggiuntivo

### ► Sul CD-ROM

- Testi e soluzioni degli esercizi trattati nei lucidi
  - Scheda sintetica
  - Esercizi risolti
  - Esercizi proposti
- Esercizi proposti da altri libri di testo

5