

Strutture e file

Dott.ssa Veronica Rossano
rossano@di.uniba.it
<http://www.di.uniba.it/~rossano>

Laboratorio di Programmazione - Veronica
Rossano

1

Tipi definiti dall'utente: i record

- Un record è una collezione di informazioni riguardanti uno specifico oggetto
- La struttura del record è determinata dalla struttura e dalla natura dell'oggetto che si vuole rappresentare
- Per definire una struttura è necessario specificare tutti i singoli elementi che la compongono

Laboratorio di Programmazione - Veronica Rossano

2

Il tipo struttura

- Il C consente di definire un record utilizzando la **structure type definition** che consentirà di definire variabili con una determinata struttura
- La sintassi è la seguente

```
typedef struct {  
    tipo1 comp1;  
    tipo2 comp2;  
    ...  
} nome_struttura;
```

Laboratorio di Programmazione - Veronica Rossano

3

Esempi

- Definiamo un record che contenga i dati identificativi di uno studente

```
typedef struct {  
    char cognome[20];  
    char nome[20];  
    char matricola[6];  
    char corso_di_laurea[20];  
} studente_t;
```

- Definiamo la struttura di un numero complesso

```
typedef struct {  
    double parte_reale;  
    double parte_immaginaria;  
} complesso_t;
```

Laboratorio di Programmazione - Veronica Rossano

4

Riferirsi ad una componente

- Per manipolare ogni singola componente della struttura si utilizza l'operatore di selezione `.` che divide il nome della struttura dal nome della componente

```
studente_t stud;
stud.nome /* si riferisce al nome */
stud.cognome /* si riferisce al cognome */
stud.matricola /* si riferisce alla matricola */

complesso_t numero;
numero.parte_reale /* si riferisce alla parte reale */
numero.parte_immaginaria /* parte immaginaria */
```

Riferirsi all'intera struttura

- Se necessario è possibile riferirsi all'intera struttura semplicemente utilizzando il nome della variabile dichiarata del tipo struttura
- Istruzioni del tipo seguente creano una copia della struttura che può essere manipolata indipendentemente dalla struttura originaria

```
studente_t stud, stud2;
stud2=stud;
complesso_t numero, numero2;
numero2=numero;
```

Strutture e sottoprogrammi

- È possibile passare un tipo struttura come parametro di input/output di un sottoprogramma
 - Per default il passaggio avviene per **valore**
 - I valori di tutte le componenti del parametro attuale sono copiate nelle componenti del parametro formale
 - Se il passaggio deve essere fatto per **indirizzo** si utilizza come di consueto il puntatore alla struttura
 - Gli operatori `*` e `&` devono essere applicati come di consueto

Restituire una struttura come risultato di una funzione

- Al contrario di quanto avviene per gli array è possibile restituire una struttura come se fosse un dato elementare del C
- Una funzione che restituisce una struttura è definita esattamente come una funzione che restituisce un dato elementare
- La funzione restituisce il valore della struttura non un indirizzo

Tipo Struttura.c

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

Definizione della
struttura studente_t

```
typedef struct
{
    char cognome[20];
    char nome[20];
    char matricola[6];
    char corso_di_laurea[20];
} studente_t;
```

Parametro
formale

```
void inserisci_dettagli(void);
void stampa_dettagli(studente_t stud);
```

```
int
main ()
```

```
{
    studente_t studente;
    printf("*****\n");
    printf("*** Inserisci i dati identificativi dello studente **\n");
    printf("*****\n");
    studente=inserisci_dettagli();
    printf("\n\n*****\n");
    printf("*** I dati identificativi dello studente inseriti sono: **\n");
    printf("*****\n");
```

Chiamata della
procedura

Chiamata della
funzione

```
    stampa_dettagli(studente);
    system("pause");
    return(0);
}
```

```
studente_t inserisci_dettagli(void)
```

```
{
```

```
    studente_t stud;
    printf("\n\n      COGNOME  --> " );
    scanf("%s", stud.cognome);
    printf("\n\n      NOME    --> " );
    scanf("%s", stud.nome);
    printf("\n\n      MATRICOLA --> " );
    scanf("%s", stud.matricola);
    printf("\n\n      CORSO DI LAUREA IN  --> " );
    scanf("%s", stud.corso_di_laurea);
    return(stud);
}
```

Uso del
parametro
attuale

```
void stampa_dettagli(studente_t stud)
```

```
{
```

```
    printf("\n\n      COGNOME  --> %s", stud.cognome);
    printf("\n\n      NOME    --> %s", stud.nome);
    printf("\n\n      MATRICOLA --> %s", stud.matricola);
    printf("\n\n      CORSO DI LAUREA IN  --> %s\n\n", stud.corso_di_laurea);
}
```

```
int
main ()
```

```
{
    studente_t studente;
    printf("*****\n");
    printf("*** Inserisci i dati identificativi dello studente **\n");
    printf("*****\n");
    inserisci_dettagli(&studente);
    printf("\n\n*****\n");
    printf("*** I dati identificativi dello studente inseriti sono: **\n");
    printf("*****\n");
```

Passaggio del parametro per
indirizzo

```
    stampa_dettagli(studente);
    system("pause");
    return(0);
}
```

Tipo Struttura Puntatore.c

```
void inserisci_dettagli(studente_t *stud)
{
    printf("\n\n      COGNOME  --> " );
    scanf("%s", (*stud).cognome);
    printf("\n\n      NOME    --> " );
    scanf("%s", (*stud).nome);
    printf("\n\n      MATRICOLA --> " );
    scanf("%s", (*stud).matricola);
    printf("\n\n      CORSO DI LAUREA IN  --> " );
    scanf("%s", (*stud).corso_di_laurea);
}

void stampa_dettagli(studente_t stud)
{
    printf("\n\n      COGNOME  --> %s", stud.cognome);
    printf("\n\n      NOME    --> %s", stud.nome);
    printf("\n\n      MATRICOLA --> %s", stud.matricola);
    printf("\n\n      CORSO DI LAUREA IN  --> %s\n\n", stud.corso_di_laurea);
}
```

Riferimento ad una componente
della struttura puntata dal
puntatore

Operatore di selezione indiretta delle componenti

- Quando si utilizza un puntatore ad una struttura il riferimento alle singole componenti è più complesso
- Si utilizza l'operatore di selezione indiretta -> che consente di selezionare i valori delle singole componenti di una struttura puntata da un puntatore
- Le seguenti istruzioni sono equivalenti

(*stud).cognome
stud->cognome

Struttura Puntatore 2.c

```

void inserisci_dettagli(studente_t *stud)
{
    printf("\n\n      COGNOME   --> " );
    scanf("%s", stud->cognome);
    printf("\n\n      NOME     --> " );
    scanf("%s", stud->nome);
    printf("\n\n      MATRICOLA --> " );
    scanf("%s", stud->matricola);
    printf("\n\n      CORSO DI LAUREA IN   --> " );
    scanf("%s", stud->corso_di_laurea);
}

```

Esercizio

- Scrivere un programma che definisca una struttura per la memorizzazione e la visualizzazione dei dati di un libro (autore, ISBN, titolo, disponibilità di magazzino)
 - NB: ricordate che le stringhe in C non possono contenere spazi

Struttura Libro.c

```

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

typedef struct
{
    char autore[20];
    char titolo[50];
    char isbn[11];
    int disponibilita;
} libro_t;

void inserisci_dettagli(libro_t *book);
void stampa_dettagli(libro_t book);

int
main ()
{
    libro_t libro;
    printf("*****\n" );
    printf("***          Inserisci i dati del libro          **\n" );
    printf("*****\n" );
    inserisci_dettagli(&libro);
    printf("\n\n*****\n" );
    printf("***          I dati del libro inseriti sono:          **\n" );
    printf("*****\n" );

    stampa_dettagli(libro);
    system("pause");
    return(0);
}

```

```

void inserisci_dettagli(libro_t *book)
{
    printf("\n\n      AUTORE   --> " );
    scanf("%s", book->autore);
    printf("\n\n      TITOLO   --> " );
    scanf("%s", book->titolo);
    printf("\n\n      ISBN    --> " );
    scanf("%s", book->isbn);
    printf("\n\n      DISPONIBILITA' --> " );
    scanf("%d", &book->disponibilita);
}

void stampa_dettagli(libro_t book)
{
    printf("\n\n      AUTORE   --> %s", book.autore);
    printf("\n\n      TITOLO   --> %s", book.titolo);
    printf("\n\n      ISBN    --> %s", book.isbn);
    printf("\n\n      DISPONIBILITA' --> %d \n\n", book.disponibilita);
}

```

Array di strutture

- È possibile combinare la definizione di un array con la definizione di una struttura per poter utilizzare delle collezioni di dati che contengano elementi simili e a loro volta composti da componenti differenti.

```
typedef struct
{
    char cognome[20];
    char nome[20];
} studente_t;
studente_t studente[MAX];
```

Laboratorio di Programmazione - Veronica Rossano

17

Esercizio

- Costruire un programma che memorizzi e visualizzi tutti i risultati delle varie prove dell'esame di programmazione degli studenti presenti in quest'aula. Per ciascuno studente memorizzare e visualizzare il voto finale calcolato come la parte intera della media tra i tre voti.

Laboratorio di Programmazione - Veronica Rossano

18

```
typedef struct
{
    char cognome[20];
    char nome[20];
    int laboratorio;
    int scritto;
    int orale;
    int media;
} studente_t;

studente_t inserisci_dettagli(void);
void stampa_dettagli(studente_t stud);

int
main ()
{
    studente_t studente[MAX];
    int n,i;
    printf("Inserire il numero degli studenti (Massimo 60) -->");
    scanf("%d", &n);
    for(i=0; i<n; ++i)
    {
        printf("\n\n**   Inserisci i dati del %d studente   **", i+1 );
        studente[i]=inserisci_dettagli();
    }
    for(i=0; i<n; ++i)
    {
        printf("\n\n**   I voti del %d studente sono   **", i+1 );
        stampa_dettagli(studente[i]);
    }
    system("pause");
    return(0);
}
```

19

```
studente_t inserisci_dettagli()
{
    studente_t stud;
    printf("\n\n      COGNOME    --> " );
    scanf("%s", stud.cognome);
    printf("\n\n      NOME      --> " );
    scanf("%s", stud.nome);
    printf("\n\n      VOTO DELLA PROVA DI LABORATORIO    --> " );
    scanf("%d", &stud.laboratorio);
    printf("\n\n      VOTO DELLA PROVA SCRITTA    --> " );
    scanf("%d", &stud.scritto);
    printf("\n\n      VOTO DELLA PROVA ORALE    --> " );
    scanf("%d", &stud.orale);
    stud.media=(stud.laboratorio+stud.orale+stud.scritto)/3;
    return(stud);
}

void stampa_dettagli(studente_t stud)
{
    printf("\n\n      COGNOME    --> %s", stud.cognome);
    printf("\n\n      NOME      --> %s", stud.nome);
    printf("\n\n      PROVA DI LABORATORIO    --> %d", stud.laboratorio);
    printf("\n\n      PROVA SCRITTA    --> %d", stud.scritto);
    printf("\n\n      PROVA ORALE    --> %d\n\n", stud.orale);
    printf("\n\n      VOTO FINALE --> %d\n\n", stud.media);
}
```

I file...

- Il C consente di utilizzare file di testo
- Un file di testo è una collezione di caratteri salvati in memoria secondaria
- Un file non ha una dimensione fissa, la fine del file è indicata con un carattere speciale di *end of file* denotato con EOF
- Il ritorno a capo in C è identificato dai caratteri \n

...I file

- Per utilizzare un file è necessario definire una variabile puntatore al file
- Il sistema deve preparare il file per ricevere l'input e l'output prima di permettere l'accesso
- Le funzioni che consentono di accedere ai file si trovano nella libreria <stdio.h>

Definire e aprire un file

- La sintassi per la definizione di un file è la seguente

```
FILE *nome_puntatore;
```

- FILE *input_file;
- FILE *studenti;

- La sintassi per l'apertura di un file è la seguente

```
nome_puntatore= fopen("nome_file.txt", "opzione");
```

Dove opzione indica l'operazione che si intende elaborare sul file

Aprire un file...

- La funzione fopen
 - Consente di aprire un file
 - Il primo argomento indica il nome del file da elaborare
 - Il secondo indica quale operazione si intende compiere sul file
 - Ritorna un puntatore ad un FILE
 - Se l'operazione non va a buon fine il puntatore assume il valore NULL

... Aprire un file

■ Le opzioni di apertura di un file

- r lettura
- w scrittura
- a appendere in coda
- r+ lettura/scrittura
- w+ creazione lettura/scrittura
- a+ appende o crea un file per lettura/scrittura

Chiudere un file

- Quando un programma non deve più utilizzare un file è necessario chiuderlo per liberare la memoria allocata al momento della sua apertura
- La funzione `fclose` chiude il file e impedisce ogni altro accesso al file

Leggere un file

■ La funzione `getc` consente di leggere un carattere alla volta all'interno di un file

- Prende in input un puntatore ad un file
- Restituisce un carattere
- Alla fine del file restituisce EOF

```

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

int
main ()
{
    FILE *primo_file;
    char ch;
    primo_file=fopen("CiaoMondo.txt", "r");
    if (primo_file==NULL)
    {
        printf("Impossibile aprire il file\n");
    }
    else
    {
        for(ch=getc(primo_file); ch!=EOF; ch=getc(primo_file))
            printf("%c", ch);
        fclose(primo_file);
    }
    printf("\n");

    system("pause");
    return(0);
}

```

Leggi File.c

Definizione del FILE

Apertura del file

Legge un carattere alla volta

Scrivere un file

- La funzione **putc** consente di scrivere un carattere alla volta all'interno di un file
 - Prende in input un puntatore ad un file e il carattere da scrivere

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

int
main ()
{
    FILE *primo_file;
    char ch[60];
    int i, lung;
    strcpy(ch, "Questo e' un nuovo file\nE' la prima prova di un file di testo");
    lung=strlen(ch);
    primo_file=fopen("CiaoMondo.txt", "w+");
    if (primo_file==NULL)
    {
        printf("Impossibile aprire il file");
    }
    else
    {
        for(i=0; i<=lung; i++)
            putc(ch[i], primo_file);
        printf("Il file e' stato creato con successo.\n");
        fclose(primo_file);
    }
    system("pause");
    return(0);
}
```

Leggi File.c

Stringa da scrivere nel file

Apertura del file

Scrittura di un carattere nel file

fprintf e fscanf

- Per inserire dati in un file sequenziale:
 - in cui inserire dati di diverso tipo
 - senza dover scandire carattere per carattere

possono essere usate le funzioni
fprintf e **fscanf**

```
int
main ()
{
    FILE *primo_file;
    char insegnamento[60];
    int voto;
    primo_file=fopen("Esami.txt", "a+"); // appende le informazioni alla fine del file
    printf("Il programma memorizza in un file Esami.txt le votazioni riportate \n da un s");
    if (primo_file==NULL)
        printf("\n\n***** Impossibile aprire il file***** \n\n");
    else
    {
        printf("Inserire gli insegnamenti e le rispettive votazioni riportate (Es. Programmazione)\n");
        scanf("%s %d", insegnamento, &voto);
        while (!feof (stdin))
        {
            fprintf(primo_file, "%s %d\n", insegnamento, voto);
            scanf("%s %d", insegnamento, &voto);
        }
        printf("File creato\n\n");
        fclose(primo_file);
    }

    system("pause");
    return(0);
}
```

Fprintf.c


```

int
main ()
{
    FILE *primo_file;
    char insegnamento[60];
    int voto;
    primo_file=fopen("Esami.txt", "r"); // appende le informazioni alla fine del file
    printf("Il programma memorizza in un file Esami.txt le votazioni riportate \n da un s
    if (primo_file==NULL)
        printf("\n\n\n***** Impossibile aprire il file***** \n\n");
    else
    {
        printf(" %-20s %-6s\n\n", "Insegnamento", "Voto");
        fscanf(primo_file,"%s%d", insegnamento, &voto);
        while (!feof (primo_file))
        {
            printf(" %-20s %d\n\n", insegnamento, voto);
            fscanf(primo_file,"%s %d", insegnamento, &voto);
        }
        printf("*****File Terminato*****\n\n");
        fclose(primo_file);
    }

    system("pause");
    return(0);
}

```

Fscanf.c

File ad accesso casuale

- In C è possibile creare file ad accesso casuale e quindi file di record
- La funzione **fwrite** trasferisce in un file un numero specificato di byte partendo da una data posizione in memoria
- La funzione **fseek** sistema il puntatore di posizione del file su un byte specifico
- La funzione **fread** legge uno specifico numero di byte da un file

Fwrite

File Accesso Casuale.c

```

do
{
    printf("Inserisci i dati dello studente: \n");
    scanf(stdin,"%s%s%s", stud.cognome, stud.nome,
        stud.matricola, stud.corso_di_laurea );
    fwrite(&stud, sizeof(studente_t), 1, primo_file);
    printf("\nVuoi Continuare? (s/n)\n");
    scanf("%c", &risp);
}while ((risp=='s') || (risp=='S'));

```

Fseek...

```
fseek(puntatoreFile,sizeof(struttura), posizionePartenza);
```

- Dove:
 - **puntatoreFile** è il puntatore al file su cui si intende operare
 - **Struttura** è il nome della struttura che si intende scrivere nel file
 - **PosizionePartenza** può assumere i seguenti valori:
 - **SEEK_SET** indica l'inizio del file
 - **SEEK_CUR** indica la posizione corrente
 - **SEEK_END** indica la fine del file

...Fseek

File Casuale Fseek.c

```

do
{
    printf("Inserire i dati dello studente (-1 nel codice per terminare) -->");
    printf("\n\n      CODICE    --> " );
    scanf("%d", &stud.codice);
    if (stud.codice!=-1)
    {
        printf("      COGNOME    --> " );
        scanf("%s", stud.cognome);
        printf("      NOME      --> " );
        scanf("%s", stud.nome);
        printf("      MATRICOLA   --> " );
        scanf("%s", stud.matricola);
        printf("\n      CORSO DI LAUREA IN    --> " );
        scanf("%s", stud.corso_di_laurea);
        fseek(primo_file, (stud.codice-1)*sizeof(studente_t), SEEK_SET);
        fwrite(&stud, sizeof(studente_t), 1, primo_file);
    }

    }while (stud.codice!=-1);

```

Fread...

```

fread(IndirizzoStruttura, sizeof(struttura), numElementi,
puntatoreFile);

```

■ Dove:

- **IndirizzoStruttura** è l'indirizzo della struttura in cui memorizzare i dati della struttura che si intende leggere dal file
- **numElementi** è il numero di elementi da leggere dal file
- **puntatoreFile** è il puntatore al file su cui si intende operare

...Fread

```

while (!feof(primo_file))
{
    fread(&stud, sizeof(studente_t), 1, primo_file);
    if (!feof(primo_file))
        printf(" %-10s %-10s %-10s %-10s\n\n", stud.cognome,
            stud.nome, stud.matricola,
            stud.corso_di_laurea );
}

```

Librerie personali

- È possibile creare le proprie librerie personali che contengano funzioni già sviluppate e pronte per essere riutilizzate
 - la direttiva del preprocessore **#include** consente di richiamare i file di libreria
 - I file di libreria devono avere estensione **.h**

Header file...

- Un header file è un file di testo che contiene tutte le informazioni necessarie al compilatore durante la fase di compilazione di un programma che utilizza funzioni definite nella libreria
- La struttura di un file header prevede
 - Un blocco di commento che definisca l'obiettivo della libreria
 - Le direttive che definiscono le costanti
 - Le eventuali definizioni di tipi
 - I corpi delle varie funzioni

...Header file

- Quando si usa un header file di sistema la sintassi è

`#include <stdio.h>`

- Quando si usa un header file personale che si trova nella stessa directory del file che stiamo realizzando la sintassi è

`#include "array.h"`

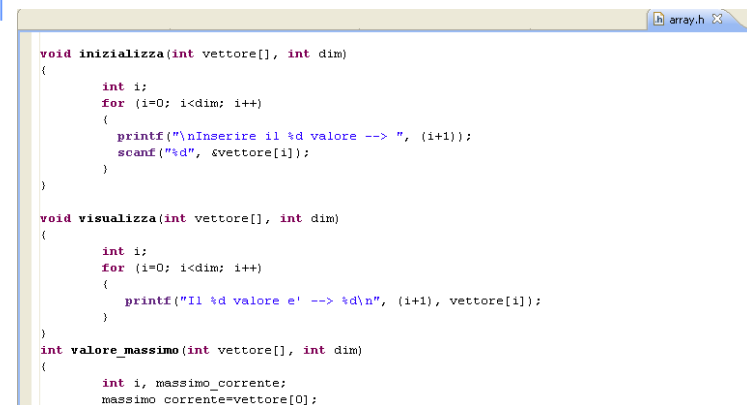
```
#include <stdio.h>
#include <stdlib.h>
#include "array.h"
#define MAX 100

int cerca_in_vettore(int vettore[], int dim, int valore);

int
main (void)
{
    int n, valore, pos;
    int vett[MAX];
    printf("Inserire il numero degli elementi del vettore (Massimo 100) -->");
    scanf("%d", &n);
    inizializza(vett, n);
    printf("\nInserire il numero da cercare -->");
    scanf("%d", &valore);
    pos=cerca_in_vettore(vett, n, valore);
    if (pos==-1)
        printf("\nElemento cercato non esiste\n\n");
    else
        printf("\nElemento cercato si trova in posizione %d\n\n", (pos+1));
    system("pause");
    return(0);
}

int cerca_in_vettore(int vettore[], int dim, int valore)
{
    int i, posizione;
    posizione=-1;
    i=0;
    while ((i<dim) && (posizione==-1))
    {
        if (vettore[i]==valore)
            posizione=i;
        ++i;
    };
    return (posizione);
}
```

formatica



```
void inizializza(int vettore[], int dim)
{
    int i;
    for (i=0; i<dim; i++)
    {
        printf("\nInserire il %d valore --> ", (i+1));
        scanf("%d", &vettore[i]);
    }
}

void visualizza(int vettore[], int dim)
{
    int i;
    for (i=0; i<dim; i++)
    {
        printf("Il %d valore e' --> %d\n", (i+1), vettore[i]);
    }
}

int valore_massimo(int vettore[], int dim)
{
    int i, massimo_corrente;
    massimo_corrente=vettore[0];
```

```
#include <stdio.h>
#include <stdlib.h>
#include "array.h"
#define MAX 100

void ordina_vettore(int vettore[], int dim);

int
main (void)
{
    int n;
    int vett[MAX];
    printf("Inserire il numero degli eleme");
    scanf("%d", &n);
    inizializza(vett, n);
    ordina_vettore(vett, n);
    visualizza(vett, n);
    system("pause");
    return (0);
}
```

File
header

```
void inizializza(int vettore[], int dim)
{
    int i;
    for (i=0; i<dim; i++)
    {
        printf("\nInserire il %d valore --> ", (i+1));
        scanf("%d", &vettore[i]);
    }
}

void visualizza(int vettore[], int dim)
{
    int i;
    for (i=0; i<dim; i++)
    {
        printf("Il %d valore e' --> %d\n", (i+1), vettore[i]);
    }
}

int valore_massimo(int vettore[], int dim)
{
    int i, massimo_corrente;
    massimo_corrente=vettore[0];
    for (i=1; i<dim; i++)
    {
        if (vettore[i]>massimo_corrente)
            massimo_corrente=vettore[i];
    };
    return(massimo_corrente);
}
```

Laboratorio di Programmazione

Altre librerie utili

- Math.h
 - Contiene funzioni matematiche
 - Sin (seno), cos (coseno), sqrt (radice quadrata),...
- Time.h
 - Contiene funzioni per il calcolo di tempo e data
 - Strftime (formato tempo), clock (tempo di esecuzione),...