



Programmazione I

Il Linguaggio C

Esercizi

Daniel Riccio

Università di Napoli, Federico II

26 ottobre 2022



Esercizio – Vocali e consonanti



Scrivere un programma in linguaggio C 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.

Il programma dovrà stampare su schermo le seguenti informazioni:

- 1) per ognuna delle lettere dell'alfabeto, il numero di volte che la lettera compare nella stringa
- 2) il numero di consonanti presenti nella stringa
- 3) il numero di vocali presenti nella stringa

Esercizio – Vocali e consonanti



```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define MAXDIM 100
#define NUMLETTERE 26
```

```
int main(){
char frase[MAXDIM +1] ;
int lung_stringa ;
int vocali, consonanti ;
int contatori[NUMLETTERE];
int posizione_alfabeto ;
int i ;
```

```
/* dimensione massima stringa di caratteri */
/* numero di lettere dell'alfabeto */
```

```
/* stringa di caratteri inserita */
/* lunghezza della stringa inserita */
/* contatori numero di vocali e di consonanti */
/* memorizza il numero di occorrenze per ogni lettera */
/* posizione nell'alfabeto di una lettera */
/* indice dei cicli */
```

Leggi la frase inserita da tastiera

```
printf ("Inserisci una frase di al massimo %d caratteri: ", MAXDIM) ;
gets(frase) ;
```

Esercizio – Vocali e consonanti



Calcola la lunghezza della frase

```
lung_stringa = strlen(frase) ;
```

Stampa la frase inserita

```
printf("La frase inserita e': ") ;
```

```
puts(frase) ;
```

```
printf("La frase contiene %d caratteri (inclusi gli spazi)\n", lung_stringa) ;
```

Azzera il vettore dei contatori. Ogni cella di questo vettore è associata a una lettera dell'alfabeto. La cella 0 alla lettera A, la cella 1 alla B e così via.

```
for ( i=0; i<NUMLETTERE; i++ )  
    contatori[i] = 0 ;
```

Analizza la frase lettera per lettera e aggiorna il vettore dei contatori

```
for ( i=0; i<lung_stringa; i++ ){  
    if ( frase[i] >= 'A' && frase[i] <= 'Z' ){  
        posizione_alfabeto = frase[i] - 'A' ;  
        contatori[posizione_alfabeto] ++ ;  
    }  
}
```

Esercizio – Vocali e consonanti



Analizza la frase lettera per lettera e aggiorna il vettore dei contatori

```
else{  
    if ( frase[i] >= 'a' && frase[i] <= 'z' ){  
        posizione_alfabeto = frase[i] - 'a' ;  
        contatori[posizione_alfabeto]++ ;  
    }  
}  
}
```

Stampa i contatori delle varie lettere

```
for ( i=0; i<NUMLETTERE; i=i+1 )  
    printf ("La lettera %c compare %d volte \n", 'A'+i , contatori[i]) ;
```

Calcola il numero di vocali. Somma il numero di occorrenze presenti nel vettore «contatori» nelle celle associate alle lettere A, E, I, O, U, Y

```
vocali = contatori['A'-'A'] + contatori['E'-'A'] + contatori['I'-'A'] + contatori['O'-'A'] + contatori['U'-'A'] + contatori['Y'-'A'] ;
```

Esercizio – Vocali e consonanti



Calcola il numero di consonanti. Il numero di consonanti si ottiene sottraendo il numero complessivo di vocali dal numero complessivo di occorrenze di tutte le lettere

```
consonanti = 0 ;
```

```
for ( i=0; i<NUMLETTERE; i=i+1 )
```

```
    consonanti = consonanti + contatori[i] ;
```

```
consonanti = consonanti - vocali ;
```

Stampa il numero di vocali e consonanti

```
printf ("Il numero di vocali e': %d\n", vocali) ;
```

```
printf ("Il numero di consonanti e': %d\n", consonanti) ;
```

```
exit(0) ;
```

```
}
```

Esercizio – Statistiche sul 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

```
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
```

Esercizio – Statistiche sul testo

L'inserimento termina quando l'utente inserisce la parola FINE su una riga da sola

```
#define MAXRIGHE 2000
#define LUN 80
char testo[MAXRIGHE][LUN] ;
int Nrighe ; /* righe inserite */
char riga[LUN*10] ;
int i, j ;
int caratteri, caralfa, parole ;
```

Quante righe sono state inserite dall'utente

```
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 ) ;

printf("L'utente ha inserito %d righe\n", Nrighe);
```


Esercizio – Statistiche sul testo



Quanti caratteri sono stati inseriti

```
caratteri = 0 ;  
for(i=0; i<Nrighe; i++)  
    caratteri = caratteri + strlen(testo[i]) ;  
  
printf("L'utente ha inserito %d caratteri\n", caratteri) ;
```

Quanti caratteri alfanumerici sono stati inseriti

```
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) ;
```

Esercizio – Statistiche sul testo



Quante parole sono state inserite

```
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) ;
```

Esercizio – Concorso di intelligenza



In un concorso di intelligenza, **N** giudici esprimono il loro giudizio su **K** candidati.

Il giudizio è un valore numerico tra **0** e **5**

Si scriva un programma in linguaggio C per determinare il candidato più intelligente, ed il giudice più severo

Esercizio – Concorso di intelligenza



Azioni:

- 1) Acquisisce il numero di candidati e di giudici
- 2) Acquisisce il voto di ciascun giudice per ciascun candidato
- 3) Somma tutti i voti ottenuti da un candidato
- 4) Somma tutti i voti dati da un giudice
- 5) Il candidato migliore è quello che ha ottenuto il punteggio massimo
- 6) Il giudice più severo è quello che ha dato i voti più bassi (somma)

Esercizio – Concorso di intelligenza



Esecuzione

```
✓ ↗ 📋
Quanti candidati ci sono? 3
Quanti giudici ci sono? 4
Immettere i giudizi per il candidato 1
Giudice 1, cosa pensi del candidato 1? 3
Giudice 2, cosa pensi del candidato 1? 2
Giudice 3, cosa pensi del candidato 1? 1
Giudice 4, cosa pensi del candidato 1? 5
Immettere i giudizi per il candidato 2
Giudice 1, cosa pensi del candidato 2? 1
Giudice 2, cosa pensi del candidato 2? 4
Giudice 3, cosa pensi del candidato 2? 3
Giudice 4, cosa pensi del candidato 2? 2
Immettere i giudizi per il candidato 3
Giudice 1, cosa pensi del candidato 3? 4
Giudice 2, cosa pensi del candidato 3? 2
Giudice 3, cosa pensi del candidato 3? 3
Giudice 4, cosa pensi del candidato 3? 5
Il vincitore e' il candidato numero 3
Il giudice piu' severo e' il numero 3
```

Esercizio – Concorso di intelligenza



		Giudici				
		0	1	2	j	N-1
Candidati	0					
	1					
	2					
	i					
	K-1					

Inseriamo i voti in una matrice Candidati (righe) \times Giudici (colonne) di dimensione **K** \times **N**

Il totale del **candidato i-esimo** è rappresentato dalla somma delle colonne alla riga **i-1**

Il totale dei voti assegnati dal **giudice j-esimo** è dato dalla somma delle righe alla colonna **j-1**

Candidato migliore \rightarrow massimo delle K somme sulle colonne

Giudice più severo \rightarrow minimo delle N somme sulle righe

Esercizio – Concorso di intelligenza



```
#include <stdio.h>
#include <stdlib.h>
#define MAXK 100 /* max n. candidati */
#define MAXN 10 /* max n. giudici */

int main()
{
    int voti[MAXK][MAXN] ;
    int tot[MAXK]={0}; /* somma dei voti per ogni candidato */
    int totg[MAXN]={0}; /* somma dei voti di ogni giudice */
    int K, N ;
    int i, j ;
    int min, max, posmin, posmax ;
```

Esercizio – Concorso di intelligenza



```
printf("Quanti candidati ci sono? ");
scanf("%d", &K) ;
printf("Quanti giudici ci sono? ");
scanf("%d", &N) ;

for (i=0; i<K; i++){
    printf("Immettere i giudizi per il candidato %d\n", i+1);

    for (j=0; j<N; j++){
        printf("Giudice %d, cosa pensi del candidato %d? ", j+1,i+1 );
        scanf("%d", & voti[i][j] ) ;
    }
}
```


Esercizio – Concorso di intelligenza



```
for (i=0; i<K; i++){  
    for (j=0; j<N; j++){  
        tot[i] = tot[i] + voti[i][j] ;  
        totg[j] = totg[j] + voti[i][j] ;  
    }  
}
```

Calcola:

Il voto totale del candidato i-esimo

(somma delle colonne alla riga i-1)

La somma dei voti assegnati da un giudice

(somma delle righe sulla colonna j-1)

```
max = tot[0] ;  
posmax = 0 ;  
  
for (i=1; i<K; i++){  
    if (tot[i]>max){  
        max = tot[i];  
        posmax = i ;  
    }  
}
```

Calcola:

Il candidato con voto massimo

(massimo del vettore **tot**)

```
printf("Il vincitore è il candidato numero %d\n", posmax+1);
```

Esercizio – Concorso di intelligenza



```
min = totg[0] ;  
posmin = 0 ;
```

```
for (i=1; i<N; i++){  
    if (totg[i]<min){  
        min = totg[i];  
        posmin = i ;  
    }  
}
```

Calcola:
Il giudice più severo
(minimo del vettore **totg**)

```
printf("Il giudice più severo è il numero %d\n", posmin+1);
```

```
return 0 ;
```

```
}
```

Esercizio – Concorso di intelligenza



Esecuzione

```
✓ ↗ 📋
Quanti candidati ci sono? 3
Quanti giudici ci sono? 4
Immettere i giudizi per il candidato 1
Giudice 1, cosa pensi del candidato 1? 3
Giudice 2, cosa pensi del candidato 1? 2
Giudice 3, cosa pensi del candidato 1? 1
Giudice 4, cosa pensi del candidato 1? 5
Immettere i giudizi per il candidato 2
Giudice 1, cosa pensi del candidato 2? 1
Giudice 2, cosa pensi del candidato 2? 4
Giudice 3, cosa pensi del candidato 2? 3
Giudice 4, cosa pensi del candidato 2? 2
Immettere i giudizi per il candidato 3
Giudice 1, cosa pensi del candidato 3? 4
Giudice 2, cosa pensi del candidato 3? 2
Giudice 3, cosa pensi del candidato 3? 3
Giudice 4, cosa pensi del candidato 3? 5
Il vincitore e' il candidato numero 3
Il giudice piu' severo e' il numero 3
```

Esercizi

In crittografia il cifrario di **Cesare** è uno dei più antichi algoritmi crittografici di cui si abbia traccia storica.

È un cifrario a sostituzione monoalfabetica in cui ogni lettera del testo in chiaro è sostituita nel testo cifrato dalla lettera che si trova un certo numero di posizioni dopo nell'alfabeto.

Scrivere un programma che prenda in input una **stringa** ed un valore intero **N**. Il programma esegue le seguenti operazioni:

- 1) Cifra il testo sostituendo ogni lettera in posizione **i** nell'alfabeto con la lettera in posizione **$i+N$**
- 2) Stampa la stringa cifrata
- 3) Decifra la stringa codificata con la sostituzione inversa
- 4) Stampa la stringa in chiaro



Esercizio – Cifrario di Cesare



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

```
#define MAX_LEN 100
#define MAX_ROW 50
```

```
typedef char Stringa[MAX_LEN];
```

```
Stringa Testo[MAX_ROW];
Stringa Cifrato[MAX_ROW];
Stringa Decifrato[MAX_ROW];
```

Esercizio – Cifrario di Cesare



```
int main (){  
    int nrow = 0;  
    int N;  
    int i,j;  
    char c;  
    Stringa riga;  
  
    do {  
        printf("Inserisci un valore intero N [1,25]: ");  
        scanf("%d", &N);  
    }while(N<1 || N>25);
```

Richiediamo un valore per N

Cicliamo fin quando il valore
di N non è compreso
nell'intervallo

Esercizio – Cifrario di Cesare



```
printf ("Inserisci il testo\n\"FINE\" per terminare\n");
```

```
do  
{  
    gets(riga);  
    if (strcmp(riga, "FINE") != 0){  
        strcpy(Testo[nrow], riga);  
        nrow++;  
    }  
}  
while (strcmp(riga, "FINE") != 0);
```

All'interno di un ciclo prendiamo in input una riga.

Se il contenuto di riga è diverso da "FINE", il contenuto di riga viene copiato in Testo

Esercizio – Cifrario di Cesare



```
// Cifriamo il testo
printf("Cifratura del testo...\n");
for(i=0; i<nrow; i++){ // cicliamo su ogni riga
    j=0;
    while(Testo[i][j] != '\0'){ // scandiamo la riga fino al terminatore '\0'
        c = toupper(Testo[i][j]); // convertiamo il carattere in maiuscolo
        if(c>='A' && c<='Z') // se si tratta di una lettera
            Cifrato[i][j] = (c - 'A' + N)%26 + 'A'; // sostituiamo il carattere
        else
            Cifrato[i][j] = c; // copiamo il carattere senza modificarlo

        ++j;
    }
    Cifrato[i][j] = '\0'; // terminiamo la riga
}
```


Esercizio – Cifrario di Cesare



```
// Decifriamo il testo
printf("\nDecifratura del testo...\n");
for(i=0; i<nrow; i++){ // cicliamo su ogni riga
    j=0;
    while(Cifrato[i][j] != '\0'){ // scandiamo la riga fino al terminatore '\0'
        c = Cifrato[i][j];
        if(c>='A' && c<='Z'){ // se si tratta di una lettera
            c = Cifrato[i][j] - N; // sostituiamo il carattere
            if(c < 'A') // se sottraendo N il carattere ottenuto precede la 'A' sommiamo 26
                Decifrato[i][j] = c + 26;
            else
                Decifrato[i][j] = c;
        }
        else
            Decifrato[i][j] = c;

        ++j;
    }
    Decifrato[i][j] = '\0';
}
```

Esercizio – Cifrario di Cesare

```
// stampiamo il testo Cifrato
printf("\nTesto cifrato:\n");
for(i=0; i<nrow; i++)
    printf("%s", Cifrato[i]);

// stampiamo il testo Decifrato
printf("\nTesto decifrato:\n");
for(i=0; i<nrow; i++)
    printf("%s", Decifrato[i]);
}
```



Programmazione I

Il Linguaggio C

Esercizi

Daniel Riccio

Università di Napoli, Federico II

09 dicembre 2022



Quesito 1-a

Qs. 1 (a) – (5 punti): si risponda alle seguenti domande riportando su un foglio bianco il numero della domanda e al lato di ciascuna numero una V se si considera la risposta vera o una F se si ritiene che la risposta sia falsa.

N.	Domanda	V/F
1	Una variabile di tipo char ammette valori nell'intervallo [0,255]	F
2	La funzione strncat(a,b) confronta i primi n caratteri delle due stringhe a e b	F
3	if(N=0){...} non esegue mai le istruzioni nelle parentesi graffe	V
4	Un record è una collezione eterogenea di variabili	V
5	La definizione di un record non può contenere altri record annidati	F
6	Nell'aritmetica dei puntatori il prodotto e la divisione sono operazioni non ammesse	F
7	Una stringa è una sequenza di caratteri terminata da un '\0'	V
8	Il nodo sentinella termina una lista doppiamente concatenata	F
9	I puntatori hanno sempre la dimensione di una parola macchina	V
10	Uno stack è una struttura ricorsiva di tipo FIFO	F
11	Il ciclo do{...}while(1); è un ciclo infinito	V
12	Il Bubble Sort ha complessità $O(n)$ nel caso medio e $O(n^2)$ nel caso peggiore	F
13	Per i dati signed, shift logico e aritmetico non è equivalente	V
14	Il tipo long int occupa 8 byte su una macchina a 32 bit	V
15	La ricorsione con stack esplicito è meno efficiente di quella basata sullo stack di sistema	F
16	La ricerca binaria non può essere applicata su una stringa di caratteri anche se ordinata	F
17	La sequenza di fibonacci si calcola come $f(n)=f(n-1) + f(n-2)$, con $f(1)=1$ e $f(0)=1$	F
18	La funzione malloc() restituisce un puntatore a intero	F
19	L'inserimento di un nodo in una lista single-linked richiede due puntatori nella ricerca della posizione	V
20	Il nome di un vettore è un puntatore costante	V



Qs. 1 (b) – (3 punti): sia dato il vettore $V=\{81, 1, 21, 11, 25, 40\}$. Si mostri lo stato del vettore ad ogni passo dell'algoritmo

Quesito 1-b

Qs. 1 (b) – (3 punti): sia dato il vettore $V=\{81, 1, 21, 11, 25, 40\}$. Si mostri lo stato del vettore ad ogni passo dell'algoritmo Insertion Sort.

Iterazione	V[0]	V[1]	V[2]	V[3]	V[4]	V[5]
1	81	1	21	11	25	40
2	1	81	21	11	25	40
3	1	21	81	11	25	40
4	1	11	21	81	25	40
5	1	11	21	25	81	40
6	1	11	21	25	40	81
7						
8						
9						
10						

Quesito 2



Qs. 2 – (5 punti): si indichi l'output del seguente programma.

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

```
int V[5]={1,2,3};
```

```
int main(){
```

```
    int *p=V;
```

```
    int i;
```

```
    for(i=0; *p>0; p++);
```

```
    p[i] = *(p-1) + *(p-2);
```

```
    printf("%ld %d %d\n", sizeof(p), *p, V[i]);
```

```
    return 0;
```

```
}
```

Output:

4 5 1

Quesito 3



Qs. 3 – (5 punti): si indichino gli errori, qualora presenti, nelle seguenti istruzioni.

```
#include "stdio.h";
```

```
int main()
```

```
int c=32;
```

```
char *pc=&c;
```

```
pc=pc*1;
```

```
for((i=0); (i<5), (i++))
```

```
c = *pc + i;
```

```
printf(%d\n, *pc);
```

```
return &i;
```

punto e virgola

Manca la parentesi graffa

corretto

Tipi incompatibili

corretto

virgola

corretto

Mancano i doppi apici

int * non è un intero

Manca la parentesi graffa

Quesito 4



Qs. 5 – (12 punti): Si implementi in linguaggio C il seguente programma

Scrivere un programma C che:

- 1) Prende da linea di comando una 10 numeri e li inserisce in un vettore V. Inoltre, chiede in input un numero x mediante una scanf()**
- 2) Implementa la funzione ricorsiva int Cerca_Valore (int V[], int n, int x) che restituisce 1 se x compare nel vettore e 0 altrimenti.**

Input e Output

Input: 10 1 23 5 8 12 7 11 33 9

Output:

Inserisci il carattere da cercare: 8

Il carattere 8 compare nel vettore

Soluzione



```
int cerca_valore(int V[], int n, int x) {  
    if (n == 0)  
        return (V[n] == x);  
    else {  
        if (V[n] == x)  
            return 1 + cerca_valore(V, n - 1, x);  
        else  
            return cerca_valore(V, n - 1, x);  
    }  
}
```

Soluzione

```
int main(int argc, char *argv[])
{
    int i;
    int x = 0;
    int posizione = 0;
    int V[10];

    for (i = 0; i < 10; i++)
        V[i] = atoi(argv[i]);

    printf("Inserisci il valore da cercare: ");
    scanf("%d", &x);

    posizione = cerca_valore(V, 10, x);

    if (posizione > 0)
        printf("Il valore %d compare nel vettore\n", x);
    else
        printf("Il valore %d non compare nel vettore\n", x);

    return 0;
}
```



Traccia 012

Scrivere un programma C che:

- 1) Prende dalla linea di comando un numero generico di interi (es.: 2 3 7 10)
- 2) Inserisce i valori in un vettore di interi allocato dinamicamente
- 3) Implementa la funzione ricorsiva `int SommaPari(int *v, int n)` che somma solo i valori pari all'interno dell'array

Input e Output

Input: 2 3 7 10

La somma degli elementi nel vettore è: 12

Soluzione



```
int SommaPari(int *A, int n) {  
    if (n == 0)  
        if ((A[0] % 2 == 0))  
            return A[0];  
        else  
            return 0;  
    else if (A[n] % 2 == 0)  
        return Somma(A, n - 1) + A[n];  
    else return Somma(A, n - 1);  
}
```

Soluzione



```
int main(int argc, char *argv[])
{
    int *V;
    int i = 0;
    int somma = 0;

    V = (int *)malloc((argc - 1) * sizeof(int));

    for (i = 1; i < argc; i++)
        V[i - 1] = atoi(argv[i]);

    somma = SommaPari(V, argc - 2);

    printf("La somma degli elementi nel vettore è: %d", somma);

    free(V);

    return 0;
}
```



Traccia 008

Scrivere un programma C che:

- 1) Definisce un nuovo tipo di variabile Polinomio come record (grado n, n+1 coefficienti float).
- 2) Scrivere una funzione float polydev(Polinomio p, Polinomio *pdev) che inserisce in pdev la derivata prima di p.

Input e Output

Inserisci il grado del polinomio: 3

Inserisci il coefficiente per x^0: 2

Inserisci il coefficiente per x^1: 1.8

Inserisci il coefficiente per x^2: 2.6

Inserisci il coefficiente per x^3: 4.5

Il polinomio di grado 3 è: $4.5x^3 + 2.6x^2 + 1.8x^1 + 2$

Il polinomio derivato è di grado 2 ed è: $13.5x^2 + 5.2x^1 + 1.8$

Soluzione

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

```
typedef struct Polinomio {
    int grado;
    float c[64];
}Polinomio;
```

```
void polydev(Polinomio p, Polinomio *pdev);
```

Soluzione



```
int main(int argc, char* argv[])
{
    int i = 0;
    Polinomio p, pdev;

    printf("Inserisci il grado del polinomio: ");
    scanf("%d", &(p.grado));

    for (i = 0; i < p.grado + 1; i++) {
        printf("Inserisci il coefficiente per x^%d:", i);
        scanf("%f", &(p.c[i]));
    }

    printf("Il polinomio di grado %d è: ", p.grado);
    for (i = p.grado; i > 0; i--)
        printf("%gx^%d +", p.c[i], i);
    printf("%g\n", p.c[0]);

    polydev(p, &pdev);
    printf("Il polinomio derivato è di grado %d ed è: ", pdev.grado);

    for (i = pdev.grado; i > 0; i--)
        printf("%gx^%d +", pdev.c[i], i);
    printf("%g\n", pdev.c[0]);
    return 0;
}
```


Soluzione

```
void polydev(Polinomio p, Polinomio *pdev) {
```

```
    int i = 0;
```

```
    pdev->grado = p.grado - 1;
```

```
    for (i = p.grado; i > 0; i--)  
        pdev->c[i - 1] = i * p.c[i];
```

```
    return;
```

```
}
```

RICCIO DOMANDE ORALI

24/01/2023

Disclaimer: portare carta e penna, possibili domande sulla traccia scritta, alcune domande uguali sono state scritte più volte, ho deciso di lasciarle così in modo da lasciar capire su quali argomenti il prof si sia soffermato maggiormente all'orale. Le domande principali sono segnate dai numeri come elenco puntato (1,2,3 ecc) le domande correlate a quelle "principali" sono indicate da "-".

1. Ricorsione (definizione e utilizzi)
 - Come funziona la funzione ricorsiva di Fibonacci
2. Complessità di un problema, cos'è e qual è
3. Cos'è una struttura ricorsiva
4. Dimensione di un record
5. Cosa sono i campi di un Bit
6. Counting Sort
 - complessità del Counting Sort
7. Come si definisce la complessità O-grande
8. Definizione della mutazione O-grande e numero di passi
9. Come funziona l'inserimento in coda di una lista SL
10. Cos'è il nodo sentinella
11. Come funzionano Malloc e Calloc
12. Aritmetica dei puntatori
13. Scrittura della matrice
14. Ricerca binaria
15. Cos'è una stringa

16. Shift a sinistra e a destra
17. Codice binario
18. Static variabili e costanti
19. Definizione di una variabile Static

- perché le variabili NON static cambiano

- come si comporta una variabile static all'interno del programma

20. Chiamate a funzione
21. Cosa viene creato alla chiamata di una funzione
22. Cos'è uno Stack
23. Cos'è un Record
24. Differenza tra Stack a capacità limitata e illimitata
25. Tipi di inserimenti in lista
26. Cos'è O-grande e Omega (chiede anche come si comporta il grafico, grafico presente nelle slides)
27. Union perché usarla
28. Cos'è una struct
29. Puntatori a void
30. Sizeof
31. Come funziona string compare
32. Dimensione di un puntatore a void
33. Algoritmi di ordinamento

-complessità degli algoritmi di ordinamento (ha fatto fare anche ordinamenti su carta)

34. La torre di Anoi

- complessità del problema

35. Pile stack e inserimenti

- complessità delle pile stack e quanto costa ogni inserimento

36. Rimozione in testa e in coda

- 37. Bubble sort
- 38. Shift a destra e sinistra
- 39. Come funziona l'insertion sort

- costo dell'insertion sort

- 40. Funzioni Push e Pop
- 41. Switch case
- 42. Le stringhe

- stringhe dentro stringhe

- 43. String compare
- 44. Come scorrere una lista
- 45. Malloc e Calloc in array
- 46. Ricorsione

- come funziona il meccanismo di ricorsione

- 47. Casting esplicito
- 48. Come scrivere in forma ricorsiva la somma dei primi N numeri (e rispettiva complessità del problema)
- 49. Cosa succede se dichiaro una variabile static in una funzione
- 50. Cosa succede se inserisco la variabile const
- 51. Puntatore a funzione
- 52. Inserimento
- 53. Counting sort
- 54. Come si ordina un array di stringhe

- string compare

- 55. Com'è fatta una lista circolare

- a cosa serve il nodo sentinella

- 56. Com'è fatta una funzione ricorsiva per calcolare un N fattoriale
- 57. Puntatore a funzione

- 58. Aritmetica dei puntatori
- 59. Cosa accade se ci sono più variabili con lo stesso nome
- 60. Sistema numerico posizionale
- 61. Come implementare uno stack con capacità limitata

RICCIO DOMANDE ORALI

28/02/23

1) Cosa sono i cicli

- quali sono
- a cosa servono
- come sono strutturati

2) Come funziona la ricerca binaria

- complessità
- da cosa deriva il "log n" formalmente

3) Strutture dati

- Com'è fatta una lista
- Quante tipologie di liste esistono e la relativa organizzazione ed elementi dei nodi
- Operazioni sulle liste

4) Come funziona l'inserimento in una lista

- come scriverlo

5) Come funziona lo switch-case

- quando lo applichiamo
- struttura
- scrittura del codice

6) Struttura della serie di fibonacci

7) Funzioni ricorsive e iterative

- pro e contro

- utilizzi

8) Come funzionano le chiamate a funzione

9) Complessità di Fibonacci

- ricorsivamente e iterativamente

10) Aritmetica dei puntatori

- cos'è

- come funziona

- Operazioni che possono essere svolte

10) Puntatori a matrici

11) Selection sort

- come opera

- complessità

12) Cos'è una struttura ricorsiva

13) Come scrivere il nodo di una lista

14) Scrittura di una struttura ricorsiva

15) Tipologie di liste

16) Code

17) Cos'è lo scope di una variabile

18) Dichiarazioni di variabili con lo stesso nome

19) Scrittura della funzione "fattoriale di un numero"

20) Fattoriale di un numero, calcolo tramite ricorsione

21) Ricorsione nello stack esplicito

22) Counting sort

23) Come funziona la ricerca binaria

24) Torre di hanoi

25) Definizione di stringa

26) Ordinamento bubble sort

27) O- grande

28) Definizione matematica di complessità

- 29) Strstr
- 30) Come implementare uno stack con array
- 31) Come allocare una matrice dinamicamente
- 32) Puntatore a funzione
- 33) Differenze tra lista e coda
- 34) Come funziona la funzione ricorsiva
- 35) Complessità di una funzione (è stata scritta al momento dal professore) e relativi complessità della forma iterativa
- 36) Cos'è il codice ASCII
- 37) Come stampo il corrispondente ascii di un carattere
- 38) Come funziona il selection sort
- 39) Cosa sono le variabili static
- 40) Cos'è il casting esplicito ed implicito