



Ambienti di Programmazione per il Software di Base

Linguaggio C
Puntatori e Vettori



Esempio main

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

main(int argc, char **argv){
    int i=0;

    while( i < argc ){
        printf("argv[%d] = %s\n", i, argv[i]);
        i++;
    }
}
```

Esempio sui puntatori

```
#include <stdio.h>
#include <stdlib.h>                                Esempio1.c

// esempio sui puntatori

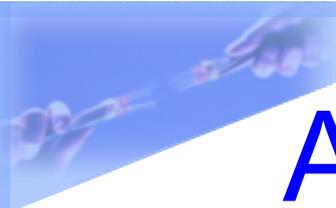
int main(int argc, char **argv){
    int b=1,c=2; /* definisco 2 interi */
    int *a; /* definisco a come puntatore a intero */

    b=10;

    a=&b; /* assegno ad a l'indirizzo di memoria di b */

    c=*a; /* assegno a c il valore contenuto all'indirizzo
    di memoria che e specificato da a quindi siccome a
    contiene l'indirizzo in memoria di b c sara' uguale a b */

    printf("a = %d\n", *a);
    printf("b = %d\n",b);
    printf("c = %d\n",c);
    return 0;
}
```



Allocazione della memoria

Il linguaggio C consente di allocare la memoria anche in fase di esecuzione del programma (= a *run-time*)

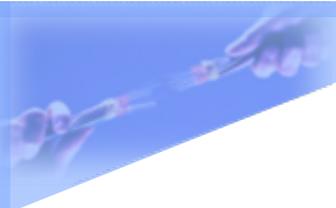
L'operazione è effettuata mediante una funzione della standard library: `malloc()`

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

```
void * malloc( size_t numeroDiBytes )
```

La funzione malloc()

- malloc() alloca un' area di memoria della dimensione specificata con il parametro numeroDiBytes
 - Si usa l' operatore **sizeof** per restituire il numeroDiBytes per un dato tipo
 - **NOTA: sizeof è un operatore, non una funzione**
 - Non occorre includere nessuna libreria
 - Il tipo restituito da sizeof è **size_t**
 - è in sostanza un tipo intero senza segno)



La funzione malloc()

- malloc() ritorna il puntatore alla prima locazione del blocco di memoria allocato dal sistema operativo
 - Il puntatore è di tipo void e dovrà essere effettuato un cast per restituire il puntatore al tipo desiderato
 - Se non c'è memoria disponibile viene restituito un puntatore NULL

La funzione free()

- Quando l'area di memoria non è più necessaria la si può restituire al sistema operativo (“rilasciare”) mediante la funzione free()
 - **ATTENZIONE:** per evitare *memory leak* ogni variabile allocata dinamicamente andrebbe rilasciata prima della fine del programma!

```
#include <stdlib.h>

void free( void * address )
```

Esempio

```
int * p;  
p = (int *) malloc( sizeof(int) );  
if( p == NULL ) {  
    printf("MEMORIA NON DISPONIBILE! \n");  
    return 1;  
}  
*p = 100;  
printf("p = %x\n \*p = %d\n", p, *p);  
free(p);
```

Array

```
int * p;
int dimensione=5;
p = (int *) malloc( dimensione * sizeof(int) );
if( p == NULL ) {
    printf("MEMORIA NON DISPONIBILE! \n");
    return 1;
}

p[0] = 100;
int i;
for(i=0; i< dimensione; i++)
    p[i]=i;
free(p);
```

Esempio sui puntatori

Esempio3.c

```
#include <stdio.h>
#include <string.h>
// concatenazione di stringhe e stampa del contenuto della memoria tramite puntatori
int main(int argc, char **argv){
    char s1[8];
    char s2[5];

    strcpy(s1,"via\0");
    strcpy(s2,"vai\0");

    int i;
    for(i=0;i<20; i++)
        printf("%d %c\n",i, *(&s1[0]+i));

    printf("[%s] - [%s]\n", s1, s2);
    printf("-----\n");

    strcat(s1, " ");
    strcat(s1,s2);

    for(i=0;i<20; i++)
        printf("%d %c\n",i, *(&s1[0]+i));

    printf("[%s] - [%s]\n", s1, s2);
}
```

Matrici

```
int ** mat;
int c=5, r=3;
mat = (int **) malloc( r * sizeof(int*) );
if( mat == NULL ) {
    printf("MEMORIA NON DISPONIBILE! \n");
    return 1;
}
int i, j;
for (i=0; i<r; i++) {
    mat[i]=(int*) malloc (c*sizeof(int));
    if (mat[i]==NULL) { ... }
} //for malloc
```

Esempio matrici.c

do free...

mat[i][j]



Esercizi

Si sviluppi un programma in linguaggio C che riceva in ingresso due vettori di interi, ciascuno di 10 elementi. Supponendo che i due vettori siano inseriti già ordinati in modo crescente, il programma deve creare e stampare un terzo vettore che rappresenti la “fusione” dei due vettori acquisiti, ovvero che contenga tutti i 20 elementi ordinati tra loro in modo crescente.

Ad esempio, se il primo vettore contiene gli elementi

2 5 9 14 15 20 25 27 30 32

e il secondo vettore contiene gli elementi

3 5 10 11 12 22 23 24 26 27

Il programma crea e stampa un vettore contenente i seguenti elementi:

2 3 5 5 9 10 11 12 14 15 20 22 23 24 25 26 27 27 30 32



Esercizi

1. Scrivere un programma che inizializza un vettore monodimensionale di interi e poi copia il vettore in un altro vettore della stessa dimensione.
 2. Scrivere un programma che crei casualmente un array di 13 interi, lo stampi a video, lo ordini con ordinamento crescente e quindi lo stampi a video nuovamente
 3. Scrivere un programma che legga una matrice da tastiera, la stampa a video e libera la memoria
 4. Scrivere un programma che calcola il prodotto tra due vettori
 5. Scrivere un programma che calcola il prodotto tra due matrici
-



Esercizi

Scrivere un programma C che:

1. richiede all'utente, in ordine strettamente crescente, l'inserimento di una serie di numeri interi (al massimo 19), salvandoli nel vettore num1. L'acquisizione termina dopo l'inserimento del diciannovesimo numero, o dopo che l'utente inserisce un numero non ordinato (questo numero non deve essere salvato);
 2. stampa a video il vettore acquisito;
 3. acquisisce dall'utente un numero intero Ne lo inserisce nel vettore, nella posizione corretta (il vettore deve rimanere in ordine crescente);
 4. stampa a video il vettore ottenuto.
-