Il linguaggio C

Breve richiamo

In questo corso si assume che lo studente abbia conoscenze pregresse della programmazione in linguaggio C/C++

Input/output

```c
#include <stdio.h>

int main()
{
    printf("Hello, world!\n");    // stampa su stdout la stringa "Hello, world!"
                                  // il carattere '\n' inserisce una nuova linea
    return 0;
}
```

Input/output

```c
#include <stdio.h>

int main()
{
    int a, b;
    printf("Leggi due interi: ");
    scanf("%d %d", &a, &b);             // legge due interi da stdin e
                                        // li assegna  alle variabili a e b
    printf("La somma di %d e %d e' %d\n", a, b, a+b);
    return 0;
}
```

If...else

```
#include <stdio.h>

int main()
{
    int a, b;
    printf("Leggi due interi: ");
    scanf("%d %d", &a, &b);
    if (a > b)
        printf("a e' maggiore di b\n");
    else if (a < b)
        printf("b e' maggiore di a\n");
    else
        printf("a e b sono uguali\n");
    return 0;
}
```

Cicli

```c
#include <stdio.h>

int main()
{
    int n, somma=0;
    printf("Inserisci dieci interi: ");
    int cont = 0;
    while (cont < 10) {
        scanf("%d", &n);
        somma += n;
        cont++;
    }
    printf("La somma e' %d\n", somma);
    return 0;
}
```

```c
#include <stdio.h>

int main()
{
    int n, somma=0;
    printf("Inserisci dieci interi: ");
    for (int cont = 0; cont < 10; cont++){
        scanf("%d", &n);
        somma += n;
    }
    printf("La somma e' %d\n", somma);
    return 0;
}
```


somma += n equivale a **somma = somma + n**

Array

```
#include <stdio.h>

int main()
{
  int a1[4] = {1,2,3,4};                    // inizializzazione di un array di 4 interi
  for (int i=0; i<4; i++)
     printf("a[%d] = %d\n", i, a[i]);       // accesso diretto agli elementi dell'array

  int a2[] = {1,2,3,4};                     // equivalente a a1
  int a3[100] = {1,2,3,4};                  // array di 100 elementi, solo 4 inizializzati

  char s1[] = {'h','e','l','l','o','\0'};   // array di caratteri: se termina con '\0' e
                                            // una stringa
  char s2[] = "hello";                      // equivalente a s1
  return 0;
}
```

Stringhe (1/2)

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

int main() {
  char str1[] = "hello\n";      // inizializza un array di 6 caratteri + '\0'
  char str2[] = "bye\n";        // inizializza un array di 4 caratteri + '\0'
  char str3[32];                // inizializza un array di 32 caratteri

  // lunghezza di una stringa
  printf("lunghezza di str1: %d\n", strlen(str1));   // 6
  printf("dimensione di str1: %d\n, sizeof(str1));   // 7

  ...
```


Una stringa è un **array di caratteri**, il cui ultimo elemento è il carattere '\0'.
Il carattere '\0' è detto «terminatore di stringa» o «carattere nullo». Da NON confondere con il carattere '\n' (fine linea).

Quando si inizializza un array di caratteri come **`char`** `str[] = "stringa"`, il carattere nullo viene sempre aggiunto implicitamente di seguito all'ultimo carattere.

Stringhe (2/2)

```
...

// confronto di stringhe
if (strcmp(str1,str2) == 0)
   printf("str1 e str2 sono uguali\n");
else
   printf("str1 e str2 sono diverse\n");

// copia di una stringa
strcpy(str3, str1);
printf("str1: %s, str3: %s\n, str1, str3);        // str1: hello, str3: hello

return 0;
}
```


La funzione strcmp() restituisce un numero:

- Uguale a zero quando le due stringhe sono uguali;
- Maggiore di zero quando str1 viene dopo str2 seguendo l'ordine alfanumerico;
- Minore di zero quando str1 viene prima di str2 seguendo l'ordine alfanumerico.

str3 = str1; → **NO!**
Non usare MAI (se non quando si sa perfettamente cosa si sta facendo) l'operatore di assegnamento per copiare il contenuto di una stringa dentro un'altra stringa.
L'assegnamento di una stringa effettua la **copia del puntatore** al primo carattere della stringa.

Struct

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

struct elem {
    int num;
    char str[32];
};

int main ()
{
    struct elem s;
    s.num = 10;
    strcpy(s.str, "hello");       // non usare assegnamento s.str = "hello"
    return 0;
}
```


Le strutture devono sempre essere riferite con la parola chiave `struct`

Puntatori

```c
#include <stdio.h>
int main ()
{
    int a = 10;
    int* p = &a;
    printf("a = %d, p = %d, valp = %d \n", a, p, *p);    // a = 10,
                                                         // p = 247659820,
                                                         // valp = 10
    return 0;
}
```

Un puntatore è una **variabile che contiene l'indirizzo di memoria di un'altra variabile**.
Nell'esempio, la variabile p contiene il valore dell'indirizzo della variabile a, ovvero p è un puntatore ad a.

Per accedere al contenuto dell'oggetto puntato da p, dobbiamo **dereferenziare il puntatore** → *p

Funzioni

- Passaggio dei parametri solo **per valore**

```c
#include <stdio.h>

int fun(int a) {
  a *= 10;
  return a;
}
int main()
{
  int n = 2;
  int m = fun(n);
  printf("n=%d, m=%d \n", n, m);
  return 0;
}
```

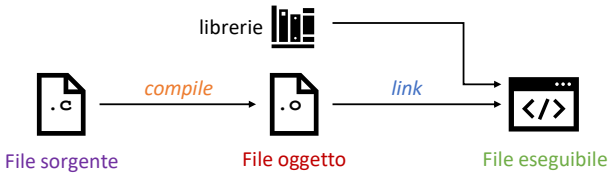
```c
#include <stdio.h>

int fun(int* a) {
  (*a) *= 10;
  return *a;
}
int main()
{
  int n = 2;
  int m = fun(&n);
  printf("n=%d, m=%d \n", n, m);
  return 0;
}
```


Entrambi gli esempi mostrano un passaggio di paramero per valore.
Nel secondo caso, infatti, la variabile a (puntatore a intero) conterrà una copia dell'indirizzo della variabile n. Dereferenziando il puntatore a, si può modificare il valore della variabile n.

Nel primo esempio, il programma stamperà n=2, m=20.
Nel secondo esempio, il programma stamperà n=20, m=20.

Compilare e eseguire un programma

- **gcc** è il compilatore C dei sistemi Unix
- Effettua anche il collegamento (linking)




```
$ gcc -c helloworld.c
$ gcc helloworld.o -o helloworld
$ ./helloworld
```

1. Compilazione
2. Collegamento (linking)
3. Esecuzione

oppure

```
$ gcc helloworld.c -o helloworld
$ ./helloworld
```

1. Compilazione e collegamento
2. Esecuzione

Il file sorgente deve essere tradotto in linguaggio macchina comprensibile dalla macchina. L'operazione è eseguita dal programma **compilatore**, che genera il file oggetto.
Il file oggetto deve essere collegato con eventuali altri moduli (altri file oggetto) che compongono il programma e le librerie per rendere il programma eseguibile. L'operazione è eseguita dal programma **collegatore** (o **linker**), che genera il file eseguibile.

gcc è un programma che effettua sia la compilazione che il collegamento.
L'opzione –o serve a specificare il nome del file di output generato dal comando