06-10-2003
Il linguaggio C

Linguaggi di Programmazione I

Matricole 2-3

Sunto della scorsa lezione

- Un programma C è un insieme di funzioni.
  La funzione *main()* è sempre presente
- Una funzione è un insieme di istruzioni racchiuse tra **{ }**
  ogni istruzione termina con **;**
- Tutto quanto è racchiuso tra /* e */ viene considerato un commento e quindi non viene compilato
- Per ricordare valori non noti a priori, si usano le variabili
- Le *variabili* vanno dichiarate prima di essere usate, preferibilmente all'inizio del programma.
- I tipi numerici *int*, *float*, *double*.
- Bisogna fare attenzione ai *tipi* che appaiono in espressioni composte. Le conversioni di tipo possono creare effetti indesiderati (*bug*).

Un Programma con Errori

```
/* File: maratona.c
* Converti in chilometri la lunghezza della maratona
*/
#include <stdio.h>
main()
[
int miglia;
float km;
miglia = 26
yard = 385;
km = 1.609*(miglia + yard/1760);
printf ("\nUna maratona è lunga %f chilometri. \n", km);
]
```


Il preprocessore

- Agisce sul file sorgente prima del compilatore
- Le righe di codice che iniziano con # rappresentano le *direttive* al preprocessore
  - Possono trovarsi ovunque nel programma ma convenzionalmente sono poste all'inizio del programma
- #define definisce una costante simbolica
  - Esempio: #define PI 3.14159 dice al preprocessore di sostituire testualmente tutte le occorrenze di PI con 3.14159
- #include serve ad includere un file di intestazione
  -Esempio: #include "file.h" dice al preprocessore di includere una copia di file.h nel punto in cui appare la direttiva #include

I file di intestazione (header)

- contengono informazioni necessarie al programma in cui sono inclusi
- il sistema C fornisce alcuni file di intestazione standard
  - contengono informazioni sugli strumenti forniti dalla libreria standard
  - Esempio: stdio.h contiene macro, definizioni di tipo e prototipi delle funzioni di input/output forniti dalla libreria standard (contiene le dichiarazioni di printf e scanf)
- #include "file.h"
  - la ricerca di file.h comincia nella directory corrente e continua in altri punti dipendenti dal sistema
- #include <file.h>
  - la ricerca di file.h avviene esclusivamente negli altri punti

Il preprocessore

```
/* File: converti.h
*File di intestazione per
* converti.c
/*
#define MULT 1000
#include <stdio.h>
```

```
/* File: converti.c
* Converti da metri a chilometri
*/
#include "converti.h"
main()
{
float metri, km;
metri= 15600;
km = metri / MULT;
printf ("chilometri:%f \n", km);
}
```


Il preprocessore

```
/* File: converti.h
*File di intestazione per
* converti.c
/*
#define MULT 1000
#include <stdio.h>
```

```
/* File: converti.c
* Converti da metri a chilometri
*/
#include "converti.h"
main()
{
float metri, km;
metri= 15600;
km = metri / MULT;
printf("chilom=metri/MULT=%f \n",km);
}
```

```
[…]
void printf(…)
[…]
main()
{
float metri, km;
metri= 1560B0;
km = metri / 1000;
printf("chilom. = metri/MULT=%f \n", km);
}
```


Output: printf

- Funzione di libreria I/O
- Riceve un elenco di parametri suddiviso in:
  - *stringa di controllo*
  - *altri parametri*
- La stringa di controllo indica il formato dell'input
  - Alcune specifiche di conversione che possono essere contenute nella stringa di controllo: %d intero in notazione decimale, %f floating point, %c caratteri, %s stringhe
  - le specifiche di conversione vengono applicate, secondo l'ordine in cui appaiono, ai restanti parametri

Esempi dell'uso di printf

- printf("%d, %d", miglia, km);
  - la stringa di controllo è: "%d, %d"
  - stampa il valore di miglia e di yard in forma decimale
- printf ("\nUna maratona è lunga %f chilometri. \n", km);
  - la stringa di controllo è: "\nUna maratona è lunga %f chilometri. \n"
  - stampa il valore di km come numero in floating point (virgola mobile) nella posizione in cui si trova %f

Printf: I campi

- la posizione in cui viene stampato un parametro è detta *campo*
- il numeri di caratteri del campo è detta *ampiezza del campo*
  - l'ampiezza del campo viene specificata da un intero posto tra % e il carattere di conversione
- Esempio:

printf("%c%4c%8c", 'P','O','P');

stampa P O P

Input: scanf

- Funzione di libreria I/O
- Riceve un elenco di parametri suddiviso in:
  - *stringa di controllo*
  - *altri parametri*
- La stringa di controllo indica il modo di interpretare l'input
- Gli altri parametri sono *indirizzi*
  - Esempio: scanf("%d", &a); interpreta l'input come un valore decimale e lo memorizza all'indirizzo di a

Esempio dell'uso di Scanf

```
/* File: prova.c
Esempio dell'uso di printf e scanf
*/
#include <stdio.h>
main()
{
int x;
char y;
printf("Inserisci un intero e un carattere:");
scanf ("%d%c", &x, &y);
printf ("x =:%d e y=%c\n", x, y);
```

Flusso del Controllo
if e if-else

- le istruzioni *if* e *if-else* vengono usate per modificare il normale flusso sequenziale del controllo
- le istruzioni *if* e *if-else* permettono di scegliere tra azioni alternative
- richiedono la valutazione di *espressioni logiche,* cioè espressioni che possono assumere valore *true* (vero) o *false* (falso)
  - *true* : ogni valore diverso da 0
  - *false*: 0

Flusso del Controllo
if e if-else

- forma generale di un' istruzione di *if:*

```
if (expr)
  statement
```

  - Significato: se *expr* ha valore *true* (diverso da 0) allora statement viene eseguita
- forma generale di un' istruzione di *if-else:*

```
if (expr)
  statement1
else
  statement2
```

  - Significato: se *expr* ha valore *true* (diverso da 0) allora viene eseguita statement1 altrimenti viene eseguita statement2

Esempio dell'uso di if

```
/* File: es_if.c
Esempio dell'uso di if
*/
#include <stdio.h>
main()
{
int x,y;
printf("Inserisci un intero :");
scanf ("%d", &x);
y=5;
if (x==6)
  y=2;
printf ("%d \n", y);
```


Esempio dell'uso di if-else

```
/* File: es_ifelse.c
Esempio dell'uso di if -else*/
#include <stdio.h>
main()
{
int x,y;
printf("Inserisci un intero :");
scanf ("%d", &x);
if (x==6)
  y=2;
else
  y=5;
printf ("%d \n", y);
```

Un altro esempio dell'uso di if

Istruzioni Composte

```
/* File: es_if1.c
Esempio dell'uso di if */
#include <stdio.h>
main()
{
int x,y,z;
printf("Inserisci un intero :");
scanf ("%d", &x);
y=5;
z=3;
if (x==6)
  {
    y=2;
    z=7;
  }
printf ("%d %d\n", y, z);
```


while

- Permette di ripetere un'istruzione o un blocco di istruzioni fino a quando una certa condizione diventa *false* (uguale a 0)
- Il controllo del test viene effettuato prima dell'inizio di ogni ciclo
- Se all'inizio la condizione del test è *false* allora non viene eseguita nessuna istruzione del ciclo
- Forma generale di un' istruzione (ciclo) di *while:*

```
while (expr)
  statement
```

- Significato: se *expr* ha valore *true* (diverso da 0) allora viene eseguito statement e il controllo ritorna all'inizio del ciclo di *while*

Esempio dell'uso di while

```
/* File: fattoriale.c
*Stampa il prodotto degli
*interi da 1 a 10
*/
#include <stdio.h>
main()
{
int i,prod;
i=1;
prod=1;
while(i<=10)
  {
    prod=prod*i;
    i=i+1;
  }
printf ("%d \n", prod);
}
```

Espressione logica

- ha valore *vero* (1) se i è minore o uguale di 10
- *falso* (0) altrimenti

Istruzioni eseguite se l'espressione logica i<=10 è vera

Esempio dell'uso di while

```
/* File: fattoriale.c
Esempio dell'uso di while
*/
#include <stdio.h>
main()
{
int i,prod;
i=1;
while(i<=10)
  {
    prod=prod*i;
    i=i+1;
  }
printf ("%d \n", prod);
}
```

- Il ciclo del *while* viene ripetuto fino a che i non diventa maggiore di 10.
- Ad ogni esecuzione del ciclo di *while* i viene incrementata di 1.
- Poiché il valore iniziale di i è 1 allora il ciclo viene ripetuto 10 volte.

Ciclo di while infinito

- se la condizione è sempre *true* allora il ciclo di *while* è infinito
- per uscire da un ciclo infinito si utilizza un'istruzione di *break*
  - *break* causa l'uscita dal ciclo più interno che la contiene

Esempio di uscita da un ciclo di *while* infinito:

```
while (TRUE) {
  scanf("%d", &x);
  if (x<10)
    break;
  printf("%d\n", 3*x)
}
```

Se x è minore di 10 allora il controllo viene trasferito alla prima istruzione successiva al ciclo

Altre istruzioni di assegnamento

- un assegnamento del tipo:

  variabile = variabile op (expr)

  può essere sostituito con un assegnamento del tipo:

  variabile op = expr

- esempio:

  `prod = prod * i;` è equivalente a `prod *= i;`

Altre istruzioni di assegnamento

- l'istruzione di assegnamento

  `i = i + 1;`

  è equivalente all'istruzione

  `i++;`

- l'istruzione di assegnamento

  `i = i -1;`

  è equivalente all'istruzione

  `i--;`

for

- Permette di ripetere un' istruzione o un blocco di istruzioni fino a quando una certa condizione diventa *false*
- Forma generale di un' istruzione (ciclo) di *for:*

```
for (expr1;expr2;expr3)
    statement
```

- In genere expr1 è un assegnamento, expr2 è un test ed expr3 è un'operazione di incremento o decremento

-Significato:

- valuta expr1
- fino a che expr2 ha valore *true* (diverso da 0) viene prima eseguito statement e poi valutata expr3

for

```
for (i=1;i<6;i++)
        statement
```

è equivalente a:

```
i=1;
while (i<6)
{
  statement
  i++;
}
```


for

- alcune o tutte le espressioni di un *for* possono essere omesse:

- for(; i< 8; i++)

  non assegna il valore iniziale alla variabile i

- for(i=1; i< 8;)

  non incrementa il valore della variabile i

- for(i=1 ; ; i++)

  il test risulta sempre *true* e si ha un ciclo infinito

Esempio dell'uso di for

```
/* File: factorial.c
*Stampa il prodotto degli
*interi da 1 a 10
*/
#include <stdio.h>
main()
{
int i,prod;
prod=1;
for(i=1; i<=10;i++)
  prod=prod*i;
printf ("%d \n", prod);
}
```

Espressione logica

- ha valore *vero* (1) se i è minore o uguale di 10
- *falso* (0) altrimenti

Istruzione eseguita se l'espressione logica i<=10 è vera

Esempio dell'uso di for

```
/* File: factorial.c
*Stampa il prodotto degli
*interi da 1 a 10
*/
#include <stdio.h>
main()
{
int i,prod;
prod=1;
for(i=1; i<=10;i++)
 prod=prod*i;
printf ("%d \n", prod);
}
```

- Il ciclo del *for* viene ripetuto fino a che i non diventa maggiore di 10.
- Ad ogni esecuzione del ciclo di for i viene incrementata di 1.
- Poiché il valore iniziale di i è 1 allora il ciclo viene ripetuto 10 volte.

Cicli innestati

- un istruzione all'interno di un ciclo può essere un altro ciclo
  - in questo caso si parla di cicli *innestati*
  - esempio di cicli innestati:

```
x = 0;
while(x < 25) {
   for(j = 1; j < = 5; j++)
      x+=j;
}
printf("%d\n", x);
```

ciclo esterno: `while(x < 25) { ... }` — ciclo interno: `for(j = 1; j < = 5; j++) x+=j;`

Un esempio di for innestati

```
/* coppie.c
* Stampa coppie di interi */
#include<stdio.h>
main()
{
int i,j;
for(i=1;i<=4;i++)
  {
   for(j=1;j<=2;j++)
     printf("(%d,%d) ", i,j);
   printf("\n");
  }
 }
```

- all'interno del ciclo di *for* for(i=1;i<=4;i++) vengono eseguite l'istruzione di *for* for(j=1;j<=2;j++) e printf("\n");
- per ogni valore dell'indice i vengono scorsi tutti i valori dell'indice j

Il programma stampa :

(1,1) (1,2)
(2,1) (2,2)
(3,1) (3,2)
(4,1) (4,2)

Un altro esempio di for innestati

- vogliamo scrivere scrivere un programma che stampa la tavola pitagorica delle moltiplicazioni
- la tabella avrà 10 righe e 10 colonne
- avremo bisogno di in indice riga e di un indice colonna

```
1 2 3 4  5  6  7  8  9 10
2 4 6 8 10 12 14 16 18 20
: :
. .
```

- stamperemo la tabella riga per riga: per ciascun indice riga scorreremo tutti gli indici colonna