5) Elementi di C

- direttive del preprocessore
- simboli speciali
- parole chiave
- identificatori
- costanti

```
#include <stdio.h>
#define TF2TC

main() {
   double tc, tf, offset, conv;
   offset = 32.;

#ifdef TF2TC
   conv = 5. / 9.;
   printf("Valore in gradi Fahrenheit= ");
   scanf("%lf",&tf);
   tc = (tf - offset) * conv;
   printf("Valore in gradi celsius= %f\n",tc);
#endif

#ifndef TF2TC
   conv = 9. / 5.;
   printf("Valore in gradi celsius= ");
   scanf("%lf",&tc);
   tf = tc * conv + offset;
   printf("Valore in gradi Fahrenheit= %f\n",tf);
#endif
}
```

Identificatori

- Gli identificatori delle variabili e delle funzioni sono una sequenza continua di caratteri (massimo 32) appartenenti alla lista

  'a'..'z', 'A'..'Z', '0'..'9', '_'

  con il vincolo che il primo carattere non sia un numero

- Quali di questi sono identificatori validi?

  abc, ABC, a_1, a1, m/h, 5a, velocita, velocita'

- Risposta: abc,ABC,a_1,a1,velocita

Costanti

- Razionali: 0.1 3. 3.0 1.234 -12.5 0.0 0. 1e10 5e-3
- Interi: -1 0 1 –44456 +877
- Caratteri: 'A' ' ' '\n'
- Stringhe di caratteri: "qualunque cosa tra doppi apici"

Simboli speciali

- Commenti
  - **/*** commento ***/**
- Direttive del precompilatore
  - Tutte le righe che iniziano con il simbolo **#**
- Operatori
  - **+,-,*,/,% ...**
- Punteggiatura
  - Ogni istruzione C termina con ;
  - Ogni parte autoconsistente del programma è racchiusa tra parentesi graffe **{ }**

Dichiarazioni

- In C e C++ una variabile (ma anche una funzione) deve essere sempre dichiarata prima di essere usata: la dichiarazione permette al compilatore di riservare in memoria lo spazio necessario e di conoscere il tipo di rappresentazione da adottare.

- In C le dichiarazioni delle variabili all'interno di una funzione o di un blocco di codice devono essere poste all'inizio della funzione o del blocco in cui vengono utilizzate (non è così in C++).

Tipi

I tipi fondamentali di variabili sono:

- int, numeri interi che occupano una parola in memoria
- float, numeri razionali che occupano una parola (numeri a virgola mobile)
- double, numeri razionali che usano due parole (doppia precisione)
- char, caratteri (usano solo 8 bit)

E le loro modifiche (signed, unsigned,short, long...)

I files `limits.h` e `float.h` definiscono per un dato sistema l'intervallo di rappresentazione dei numeri.

Espressioni aritmetiche

- Le espressioni aritmetiche sono sequenze di identificatori, operatori e parentesi (tonde)

- Esempi:
  - `x+y`
  - `-x`
  - `x%2` `/* % è l'operatore modulo (resto della divisione tra due interi) */`
  - `x*(y-z)/(x+y)`

Precedenza degli operatori aritmetici

- Gli operatori *, / e % hanno precedenza su + e –
- Le espressioni vengono valutate da sinistra a destra
- Esempi:
  - 5-7*2+1 equivale a 5-(7*2)+1 e vale –8
  - 4.1/2-2 equivale a (4.1/2)-2 e vale 0.05
  - 9%5+1 equivale a (9%5)+1 e vale 5

Assegnazione

- Un'assegnazione altera il valore di una variabile:

  id_var = espressione;

  Il valore dell'espressione (*rValue: right value*) viene assegnato alla variabile (*lValue: left value*)

  var1 = 20.0 ;

- Le assegnazioni posso essere anche concatenate

  var1 = var2 = var3 = 20.0;

  (sconsigliato! Il programma risulta meno leggibile)

Esempi

- Incremento del valore di una variabile intera. Le seguenti espressioni forniscono tutte lo stesso nuovo valore della variabile **totale**.
    - **totale = totale + 1;**
    - totale += 1;
    - totale++;
    - ++totale;

- Decremento:
    - **totale = totale – 1;**
    - totale –= 1;
    - totale--;
    - --totale;

Esempi

- Modifica del valore di una variabile: anche in questo caso il valore finale di **costo** è lo stesso

    - costo = costo + 0.2 * costo;
    - costo += 0.2 * costo;
    - costo = costo * 1.2;
    - costo *= 1.2;

Conversioni di tipo

- Possiamo assegnare valori interi (int) a variabili razionali (float, double): viene usato il valore reale corrispondente (promozione)

```
double x;
int i;
i=100;
x=i;
```

- Quando invece assegniamo dei reali a degli interi perdiamo la parte frazionaria (troncamento)

```
int j = 1.234;
int k = -1.99;
printf("j= %d",j); /* stampa 1 ! */
printf("k= %d",k); /* stampa -1 ! */
```

Uso di / e di %

```
  int secondi,  tOre, tMinuti, tSecondi;
/* esempio di conversione da secondi a ore, minuti, secondi */
   secondi= 3726;              /* tempo totale in secondi */
   tOre = secondi / 3600;        /* tOre vale 1 */
/* NB per effettuare una divisione il calcolatore promuove
 entrambi gli operandi a numeri reali, quindi  secondi/3600
 è uguale a 3726./3600. */
   tMinuti = secondi % 3600 / 60;
/* il calcolatore valuta secondi % 3600 che vale 126, poi
 calcola 126/60, che vale 2.1 e lo converte nell'intero 2
 quando lo assegna a tMinuti */
   tSecondi = secondi % 3600 % 60;
/* il calcolatore valuta secondi % 3600 che vale ancora 126,
  poi 126 % 60 che vale 6 */
```

Arrotondamenti

- Il troncamento può essere sfruttato per effettuare l'arrotondamento di un numero con la precisione voluta.
- Consideriamo un numero reale r che vogliamo arrotondare con precisione p. Procederemo nel modo seguente:

```
float r,p,ar;
int temp;
temp=r/p + 0.5;
/* il troncamento è per difetto, sommando 0.5 si arrotonda
  all'intero più vicino */
ar=temp*p;
```

- esempio: r=10.52 , p=0.1 , 10.52/0.1 + 0.5 vale 105.2+0.5 ovvero 105.7, quindi temp vale 105 e ar vale 10.5
- esempio: r=10.56 , p =0.1, in questo caso otteniamo 105.7+0.5 ovvero 106.2, temp vale 106 e ar vale 10.6

Espressioni booleane

- Un'espressione logica (booleana) può assumere solo due valori: vero o falso (true o false).
- In C ogni espressione numerica può essere utilizzata come espressione booleana in quanto il C interpreta zero come **false** e ogni altro valore numerico come **true.**
- Le espressioni logiche si combinano tra loro mediante operatori booleani per costituire nuove espressioni logiche.
- un'espressione logica può essere il risultato dell'applicazione di operatori relazionali.

Operatori relazionali e booleani

<	minore di
<=	minore o uguale di
>	maggiore di
>=	maggiore o uguale di
==	uguale a (da non confondere con =)
!=	diverso da

Relazionali: producono una risposta booleana

Hanno precedenza su

&&	and logico
||	or logico
!	Not logico

Booleani

Esempi

- 4 < 5 vale 1 (true)
- 2 >= 3 vale 0 (false)
- 4 == 4 vale 1 (true)

double x = 1.5, y = -1.8;

- x > y vale 1 (true)
- x > y + 5 vale 0 (false)
- (4<5)&&7<6 vale 0 (false)

Esercizi

- Valutare le seguenti espressioni dati

  int j=7, k=3, m=12;

  - j - k / m
  - m * j % k
  - 2 * j - k
  - (j - k) * 2

Esercizi

valutare le seguenti espressioni booleane

!(4<5)

3 <= 4 && 5 < 7

2 < 1 || 6 < 8

!(5 ==5) && 3<7

!(x = -56) con x=56

Soluzioni

int j=7, k=3, m=12;

- j - k / m vale 7
- m * j % k vale 0
- 2 * j - k vale 11
- (j - k) * 2 vale 8

Soluzioni

- !(4<5) vale 0
- 3 <= 4 && 5 < 7 vale 1
- 2 < 1 || 6 < 8 vale 1
- !(5 ==5) && 3<7 vale 0
- !(x = -56) con x=56 vale 0 !

Altri operatori

- Il C fornisce un operatore che restituisce il numero di byte occupato in memoria da una espressione E o da un tipo di variabile T: sizeof(E) o sizeof(T).

- E' disponibile anche l'operatore *cast*, che effettua una conversione tra tipi lasciando invariata la variabile originale var: `(nuovo tipo) var`

```
double r = (double)1/(double)5;
```

Input/output (I/O)

- Il C prevede le seguente funzioni
  - printf: scrittura su schermo
  - fprintf: scrittura su file
  - scanf: lettura da tastiera
  - fscanf: lettura da file
- Il formato generico di printf (e fprintf) è

```
printf("Stringa di caratteri"[esp1,esp2....]);
```

- Il formato generico di scanf è

```
scanf("Stringa di caratteri",&v1[,&v2...]);
```

printf

- Esempi:

```
printf("Valore in gradi celsius=");
/* stampa la stringa tra doppi apici*/
printf("Valore in gradi celsius= %f\n,tc);
/* stampa la stringa tra doppi apici seguita dal valore della
   variabile razionale tc e va a capo*/
```

- Descrittori di formato:
  - %f, %e, %g per numeri razionali float e double
  - %d, %i per numeri interi
  - %c per caratteri
  - %s per stringhe
  - %x per rappresentazione esadecimale di numeri interi

scanf

- Legge il valore di una o più variabili dal canale di input
- Tra i doppi apici possono comparire solo i descrittori dei formati delle variabili da acquisire
- Gli identificatori delle variabili sono preceduti dal simbolo &
- Descrittori di formato:
  - %f per numeri razionali float e %lf per i double
  - %d, %i per numeri interi, %u per gli unsigned int
  - %c per caratteri
  - %s per stringhe
  - %x per rappresentazione esadecimale di numeri interi