

C: variabili e assegnamenti

Fondamenti di Programmazione

Dichiarazione di variabili

- E' il nome (etichetta) di un'area di memoria che il nostro programma potrà sucessivamente utilizzare.
- Dichiarazione delle variabili: definizione delle aree di memoria che saranno utilizzate nel programma.
- Assegnazione di un **nome** (identificatore) e attribuzione del **tipo**.
- Il tipo definisce le caratteristiche che regolano l'uso di una variabile.

```
tipo nomevariabile;
```

```
tipo nomevariabile1, nomevariabile2;
```

Dichiarazione di variabili

```
tipo nomevariabile1, nomevariabile2;
```

```
int i;  
int a,b;  
char c;
```

dove `int` e `char` rappresentano due diversi tipi

NB: il C è *case sensitive*!

NB: scelta del nome delle variabili

Dichiarazione di costanti

Viene assegnato un valore in maniera permanente

```
const float PiGreco = 3.14;  
const int N = 100;
```

```
#define N 100  
  
//direttiva al preprocessore  
//non e' scope-controlled
```

Differenze: vantaggi e svantaggi

Istruzione di assegnamento

Viene utilizzato il simbolo =

Permette di assegnare ad una variabile un valore o il risultato di una espressione (*o il valore restituito da una funzione*).

identificatore = valore

```
a=0;  
c='a';  
x=x+1; //Come si interpreta?
```

NB: da non confondere con il simbolo == (*uguale a*)!

Operatori aritmetici

+ addizione

– sottrazione

* moltiplicazione

/ divisione

% modulo (resto della divisione)

Istruzioni di input e output

Standard output: `printf()`

Standard input: `scanf()`

Istruzione printf

Stampa su video

```
printf("Ciao"); //stampa a video: Ciao  
printf("Ciao\n"); //stampa a video: Ciao
```

Necessita dell'inclusione della libreria `stdio.h`

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


Istruzione printf

Stampa su video

```
#include <stdio.h>

int main()
{
    int var=1;
    char c='a';
    printf("%d\n",var);
    //%d formato di stampa: intero sistema decimale
    printf("%c\n",c);
    //%c formato di stampa: carattere
    return 0;
}
```

Istruzione printf

```
#include <stdio.h>

int main()
{
    int base=2;
    int altezza=4;
    int area;

    printf("La base vale: %d\n",base);
    printf("L'altezza vale: %d\n",altezza);

    area = base * altezza;

    printf("L'area del rettangolo vale: %d\n",area);

    return 0;
}
```

Istruzione printf

```
#include <stdio.h>

int main()
{
    int base=2;
    int altezza=4;

    printf("Base: %d Altezza: %d\n",base,altezza);
    printf("L'area vale: %d\n",base * altezza);

    return 0;
}
```

Istruzione scanf

Inserimento di valori (da tastiera)

```
scanf("%d", &base);
```

Istruzione scanf

Inserimento di valori (da tastiera)

```
scanf("%d", &base);
```

Cosa indica il simbolo `&` ?

Perchè si usa il simbolo `&` ?

Cosa succede se non uso il simbolo `&` ?

Istruzione scanf

Inserimento di valori (da tastiera)

```
scanf("%d", &base);
```

& indica l'indirizzo di memoria della variabile `base`

& si usa perché la scanf si aspetta un puntatore (C: *passaggio per valore*)

se non utilizzo il simbolo & il programma si comporta in modo *anomalo*

! Esercizio 1

Scrivere un programma in C che permetta di calcolare l'area di un rettangolo. La base e l'altezza devono essere letti da tastiera.

Soluzione 1

```
#include <stdio.h>

int main()
{
    int base, altezza;

    printf("\nInserisci la base: ");
    scanf("%d", &base);

    printf("\nInserisci l'altezza: ");
    scanf("%d", &altezza);

    printf("\nL'area vale: %d", base * altezza);

    return 0;
}
```


! Esercizio 2

Scrivere un programma in C che esegua la somma di due numeri interi

Soluzione 2

```
#include <stdio.h>

int main()
{
    int n, m, somma;

    printf("\nInserisci i numeri: ");
    scanf("%d%d", &n,&m);

    somma=n+m;

    printf("\n%d + %d: %d",n,m, somma);

    return 0;
}
```

Espressioni aritmetiche e precedenze

```
#include <stdio.h>

int main()
{
    int n1=4, n2=2;
    float f1=4, f2=2;

    printf("1 + n1 / 2 * n2 = %d", 1 + n1 / 2 * n2); //5
    printf("\n");
    printf("1 + f1 / 2 * f2 = %f", 1 + f1 / 2 * f2); //5.0
    printf("\n");

    return 0;
}
```

Istruzioni composte

Unione di operatori aritmetici e assegnazione:

```
operatore_aritmetico= (" += ")
```

```
count += 10;
```

somma l'espressione a destra con l'espressione a sinistra e
assegna alla variabile a sinistra

```
count = count + 10;
```

! Esercizio 3

Calcolare diametro, circonferenza e area di un cerchio a partire dal valore del raggio.

r = Raggio

π = Pi Greco

Circonferenza = $2 * \pi * r$

Area = $r^2 * \pi$

Area = $\frac{\text{Circonferenza} * \text{Raggio}}{2}$



✓ Soluzione 3

```
#include <stdio.h>

int main()
{
    const float Pi=3.14;
    float raggio, diametro, circ, area;

    printf("Inserisci raggio: ");
    scanf("%f",&raggio);

    diametro=2*raggio;
    circ=di diametro*Pi;
    area=Pi*raggio*raggio;

    printf("Diametro: %f ",diametro);
    printf("Circonferenza: %f ",circ);
    printf("Area: %f ",area);

    return 0;
}
```

Conversione

```
#include <stdio.h>
int main()
{
    float f1=3.7, f2;
    int i1,i2=-10;

    i1=f1; //conversione da float ad intero: 3
    printf("%f -> ad intero produce: %d\n", f1, i1);

    f1=i2; // conversione da intero a float: -10.000
    printf("%d -> a float produce: %f\n", i2, f1);

    f1=i2/3; // divisione tra interi: -3.000
    printf("%d diviso 3 produce: %f\n", i2, f1);

    f2=i2/3.0; // intero diviso float: -3.333
    printf("%d diviso 3.0 produce: %f\n",i2,f2);

    f2=(float) i2/3; //cast: -3.333
    printf("(float) %d diviso 3 produce: %f\n",i2,f2);
    return 0;
}
```