Programmazione in C

La struttura del programma

Variabili, espressioni, operazioni

Struttura del programma

```
// Area quadrato circoscritto
// circonferenza
/* Author:          Lorenzo
   Course:          info
*/
#include <stdio.h>

int main ( void ) {
     int r, area;
     do {
          scanf("%d", &r);
     } while (r < 0);
     area = 4 * r * r;
     printf("%d\n}", area);
     return 0;
}
```

- commenti. Su una riga o su più righe.
- #include: chiede al preprocessore di caricare il contenuto di un file.
- Ogni programma in c contiene una o più funzioni ma solo una si chiama main. E' la funzione dalla quale inizia l'esecuzione del programma.
- la funzione è di tipo intero, cioè la funzione restituisce un intero
- return indica il valore restituito dalla funzione. 0 significa tutto ok!
- { } indica un blocco. Il corpo di ogni funzione è racchiuso in un blocco.

printf

In c non esistono istruzioni di input/output.

printf è una funzione di libreria: durante la compilazione, quando viene richiamata il **linker** la ricerca nella libreria e carica il codice nell'obj.

Tutto ciò che compare tra "" viene visualizzato ma alcune sequenze, introdotte dal carattere '\' (esc) producono azioni …

Escape Sequence	Description
\n	Newline. Position the cursor at the beginning of the next line.
\t	Horizontal tab. Move the cursor to the next tab stop.
\a	Alert. Sound the system bell.
\\	Backslash. Insert a backslash character in a string.
\"	Double quote. Insert a double quote character in a string.

variabili

variabile: spazio di memoria nel quale può essere inserito un valore. La variabile è il contenitore.

Ogni variabile è caratterizzata dal tipo, la forma del contenitore.

Il tipo stabilisce lo spazio occupato dalla variabile, la rappresentazione utilizzata e le operazioni permesse.

Il nome è una sequenza di lettere, numeri, _ , ma non deve iniziare con un numero.

Il C è *case sensitive*

variabili

Data types	printf conversion specifications	scanf conversion specifications
long double	%Lf	%Lf
double	%f	%lf
float	%f	%f
unsigned long int	%lu	%lu
long int	%ld	%ld
unsigned int	%u	%u
int	%d	%d
short	%hd	%hd
char	%c	%c

Nella tabella sono indicati i tipi di dato a disposizione in C in ordine gerarchico, dal basso all'alto. Ognuno occupa uno spazio di memoria prefissato, misurato in byte, ed utilizza una particolare codifica.

scanf

Funzione di input. Permette all'utente,all'operatore, di inserire un valore in una variabile scrivendo i dati sulla tastiera e terminando l'inserimento con RETURN.

Tra "" compare il carattere di formattazione preceduto dal simbolo %.

Il simbolo & posto davanti al nome della variabile specifica che si fa riferimento al contenitore e non al contenuto della variabile. Indica la zona di memoria dove si trova la variabile.

scanf - printf

Permettono input – output formattato.

- Input: il valore inserito dall'utente viene convertito in base a quanto indicato.
- Output: il valore che deve essere visualizzato viene considerato del tipo indicato dal carattere di formattazione e convertito opportunamente.

scanf - printf

```
int main() {
    int n=123456789;
    char x='a';
    float y=-3.5123456789;

    printf("Int %d float %f\n", n,
    n);
    printf("Character %c int
    %d\n", x, x);
    printf("float %f int %d\n", y,
    y);
    return 0;
}
```

Risultato prodotto:

Int 123456789 float 0.000000
Character a int 97
float -3.512346 int -1073741824

assegnamento

area = 4 * r * r; // area ⟵ 4*r*r

Nella variabile area, sulla sinistra dell'operatore di assegnamento, viene inserito il risultato dell'espressione sulla destra.

Sinistra: variabile.

Destra: espressione che viene valutata.

L'operazione produce come risultato il valore assegnato.

Il contenuto di area viene modificato, il contenuto di r rimane inalterato.

Assegnamento: esempi

```
int main() {
    int x=2, y=3;
    printf("assegna %d\n", x=y);
    printf("x = %d y = %d\n", x, y);
    return 0;
}
```

Il risultato prodotto sarà:

```
assegna 3
x = 3 y = 3
```

Operatori aritmetici

C operation	Arithmetic operator	Algebraic expression	C expression
Addition	+	$f + 7$	`f + 7`
Subtraction	-	$p - c$	`p - c`
Multiplication	*	bm	`b * m`
Division	/	x / y	`x / y`
Modulus	%	$r \bmod s$	`r % s`

Il risultato di un'espressione aritmetica dipende dai tipi di dato coinvolti.

Se ci sono più operatori dello stesso tipo la valutazione procede da sinistra a destra.

*, /, % hanno la precedenza su + , -

( ) permettono di cambiare l'ordine di valutazione

Operatori relazionali

Standard algebraic equality operator or relational operator	C equality or relational operator	Example of C condition	Meaning of C condition
Equality Operators			
=	==	x == y	x is equal to y
≠	!=	x != y	x is not equal to y
Relational Operators			
>	>	x > y	x is greater than y
<	<	x < y	x is less than y
>=	>=	x >= y	x is greater than or equal to y
<=	<=	x <= y	x is less than or equal to y

Bisogna prestare attenzione a come si compongono più operatori.

In matematica 3 < x < 8 significa x compreso tra 3 e 8

In c la stessa scrittura da come risultato 1, cioè vero, a prescindere dal valore x

Precedenze degli operatori

Operators				Associativity
*	/	%		left to right
+	-			left to right
<	<=	>	>=	left to right
==	!=			left to right
=				right to left

Gli operatori vengono valutati da sinistra a destra escluso l'assegnamento che procede da destra a sinistra.

int a=3, b=4, c=5, d=6;

a=b=c=d;

Alla fine dell'operazione a, b,c,d conterranno 6.

Operatori di assegnamento

a+=2;	a = a + 2;
a-=4;	a = a – 4;
a*=3;	a = a * 3;
a/=5;	a = a / 5;
a%=3;	a = a % 3;

Operatori di incremento e decremento

++ operatore di incremento

a++; a = a + 1;

-- operatore di decremento

a--; a = a – 1;

Pre incremento, pre decremento:

++a; --a;

Il risultato cambia se utilizzati in una espressione.

Operatori di incremento e decremento: esempio

```
int main() {
    int x=2, y=3, z;
    int a=2, b=3, c;

    z = x++ + y++;
    c = ++a + ++b;
    printf("z = %d\nx = %d\ny = %d\n", z, x, y);
    printf("c = %d\na = %d\nb = %d\n", c, a, b);

    return 0;
}
```

Risultato

```
z = 5
x = 3
y = 4

c = 7
a = 3
b = 4
```