Analisi del programma quadrato

```
/*
   calcolare l'area del quadrato
*/

#include <stdio.h>

main(){
  int lato, area;

  printf("lato ");
  scanf("%d", &lato);
  if(lato < 0) printf("Errore nei dati \n");
  else {
   area = lato * lato;
   printf("Area = %d \n", area);
  }
}
```

- Commenti:

  /* questo è l'inizio del commento
  e questa è la fine ***/**

- #include <stdio.h>

  Direttiva al preprocessore di inserire il file
  stdio.h

- main(){
  int lato, area;

  main è il nome di funzione. Si capisce che è una funzione per la presenza delle parentesi

  int è una parola chiave
  lato area sono due identificatori

- printf
  scanf

  sono due identificatori. La presenza delle parentesi indica che sono identificatori di funzioni

- “lato “
  Una sequenza di caratteri racchiusa da doppi apici è una costante stringa.

- & lato

  & è l’operatore indirizzo.

- area = lato * lato

  = e * sono operatori

- if else

  sono parole chiave

- “,” “;” “{“ “}” sono simboli di interpunzione

Operatori aritmetici

Sono:

```
+   -   *   /   %
```

Assegnamento

Per modificare il contenuto di una variabile si usa l'operatore assegnamento =
Ad esempio

```
a = b + c;
```

= è un operatore che produce un risultato, che è il valore assegnato:

```
a = b = c + d;
```

si calcola c + d. il risultato viene assegnato alla variabile b, che produce come risultato il valore assegnato, che viene rassegnato ad a.

```
a = 1;
b = 2;
c = a + b;
```

oppure

```
c = (a = 1) + (b = 2);
```

Operatori di assegnamento

Sono:

```
=   +=  -=  *=  /=  %=  >>=    <<=    ^=     |=
```

Incremento e decremento

++ operatore di incremento

-- operatore di decremento

```
int i;

    i++;        incrementa i
    ++i;        incrementa i
```

sono equivalenti

```
        i = i+1;
```

Utilizzo.

```
        int a=0, b=0, c=0, d=0;

        c = a++;
        d = ++b;
```

quanto vale c, d?

Libreria standard

```
/*
    genera n numeri casuali e li scrive

*/

#include <stdio.h>
#include <stdlib.h>


main() {
    int i, n;

    printf("Generatore di numeri casuali.\n");
    printf("Quanti numeri vuoi generare? ");
    scanf("%d", &n);

    for(i=0; i<n; ++i){
        if( i % 5 == 0) printf("\n");
        else printf(" ");
        printf("%10d", rand() );
    }
}
```

i file

```
        stdio.h
        stdlib.h
```

contengono i prototipi delle funzioni per input/output e le funzioni di libreria. Sono file di intestazione (header file) e per convenzione terminano con .h

Non bisogna confonderli con le librerie: la libreria standard contiene il codice delle funzioni, compilate in precedenza. I file di intestazione **non** contengono codice compilato.

I tipi di dato presenti in C sono:

char	short int	int	long int
unsigned char	unsigned short int	unsigned int	unsigned long int
signed char	float	double	long double

Il seguente programma stampa per ogni tipo lo spazio di memoria occupato.

```
/*

    stampa memoria occupata per ogni tipo di dato

*/

#include <stdio.h>

main() {
  printf("memoria occupata dai tipi di dato \n");

  printf("       char:%3d bytes\n", sizeof(char));
  printf("      short:%3d bytes\n", sizeof(short));
  printf("        int:%3d bytes\n", sizeof(int));
  printf("       long:%3d bytes\n", sizeof(long));
  printf("   unsigned:%3d bytes\n", sizeof(unsigned));
  printf("      float:%3d bytes\n", sizeof(float));
  printf("     double:%3d bytes\n", sizeof(double));
  printf("long double:%3d bytes\n", sizeof(long double));
}
```

Operatori relazionali

Gli operatori logici sono utilizzati nelle espressioni e forniscono come risultato un *int* di valore 0 per falso e 1 per vero. Gli operatori sono:

operatori relazionali	<	minore
	>	maggiore
	<=	minore uguale
	>=	maggiore uguale
operatori di uguaglianza	==	uguale
	!=	diverso
operatori logici	!	negazione
	&&	and logico
	||	or logico

L'unico operatore unario è la negazione (!); tutti gli altri sono binari.

Per illustrare le precedenze consideriamo la seguente tabella.

```
char c = 'w';
int i = 1, j = 2, k = -7;
double z = 7e+33, y = 0.001 ;
```

Espressione	Espressione equivalente	Risultato
'b'+1<c	('b'+1)<c	1
x<x+y	x<(x+y)	1
3<k<5	(3<k)<5	1
i!=j	!(i==j)	1
!!5	!(!5)	1
!5-3	(!5)-3	-3
x/!y	x/(!y)	errore
i && j && k	(i && j) && k	0
x && i || k	(x && i) || k	1

Alcuni risultati potrebbero risultare sorprendenti.

```
3<k<5      con k = -7.
```

Il risultato è 1, cioè vero perché non vengono seguite le regole della matematica ma quelle dell'associatività da sinistra a destra.

La scrittura corretta è

```
3 < k && k < 7
```

Attenzione: nel caso in cui si utilizza la seguente scrittura

```
expr1 && expr2
```

con expr1 = 0 (falso) expr2 non viene valutata.

Mentre

```
expr1 || expr2
```

non viene valutata expr2 se expr1 è diversa da 0 (vera).

Istruzione composta

Sequenza di dichiarazioni ed istruzioni racchiusa tra parentesi grafe.

```
{
    x = 2;
    {
        y = 3;
        z = 5;
    }
}
```

Istruzione vuota

E’ formata da un puntoevirgola

```
;
```

Può essere utile quando la sintassi richiede la presenza di una istruzione.

Istruzione condizionale.

istruzione if*:* ha la forma

```
if ( espressione )

    istruzione;
```

dove istruzione può anche essere un'istruzione composta.

Se il risultato di espressione è un valore diverso da 0 allora viene eseguita istruzione.

istuzione if - else: ha la forma

```
if ( espressione )
   istruzione1;
else
   istuzione2;
```

dove sia istruzone1 che istruzione2 possono anche essere istruzioni composte.

Se espressione ha come risultato un valore diverso da 0 viene eseguita l'istruzione 1 altrimenti viene eseguita l'istruzione 2.

- Assegnare a min il valore minore tra a e b.

```
if( a<=b ) min=a;
else min=b;
```

- Determinare il maggiore tra tre numeri.

```
if(a>b ) max=a;
else max=b;
if( max< c ) max=c;
```

- Dati due numeri max e min ordinarli.

```
if( max<min ) {
    tmp = max; max=min; min=tmp;
 }
```

- Istruzione probabilmente inutile:

```
if( a<b );
```

Istruzioni di ciclo.

istruzione while: ha la forma

```
while( espressione )
    istruzione;
```

dove istruzione può essere un'istruzione composta.

Mentre espressione è diversa da zero viene eseguita istruzione.

- Eseguire una sequenza di azioni 5 volte;

```
int n=5;
while( n-- >0 ) {
    /* esegui azioni */
}
```

oppure

```
int n=5;
while( n-- ) {
    /* esegui azioni */
}
```

- un ciclo infinito.

```
while( 1 )
    /* azione */
```

istruzione do : ha la forma

```
do
   istruzione
while( espressione );
```

dove istruzione può essere un'istruzione composta. È una variante dell'istruzione while dove il test di controllo viene eseguito dopo le istruzioni. Le istruzioni vengono ripetute se l'espressione è diversa da zero.

Spesso per semplificare la lettura del codice, anche se l'istruzione è singola si scrive:

```
do {
   istruzione;
} while( espressione );
```

- ripetere un ciclo 5 volte.

```
int n=5;
do {
   /* istruzioni */
} while( --n );
```

istruzione for : ha la forma

```
for( espressione1; espressione2; espressione3 )
    istruzione;
```

dove istruzione può essere anche l'istruzione composta.

Questo ciclo è equivalente al seguente:

```
espressione1;
while( espressione2 ) {
    istruzioni;
    espressione3;
}
```

- Espressione1 rappresenta l'operazione che deve essere effettuata prima del ciclo;

- espressione2 è il controllo che viene effettuato per decidere se eseguire le istruzioni del ciclo: se il risultato è diverso da zero il ciclo viene eseguito.

- Espressione3 rappresenta le operazioni che vengono eseguite dopo l'esecuzione di istruzioni.

- ripetere 5 volte un'istruzione si scrive:

```
for(i=1; i<=5; i++)
   /* istruzioni del ciclo */
```

- In un ciclo for tutte e tre le espressioni possono anche essere l'istruzione vuota. Con un ciclo for possiamo simulare anche un ciclo while nel seguente modo:

```
for( ; espressione; )
```

- Per avere una ciclo infinito si scrive

```
for(; 1 ;)
```

- Ripetere un ciclo n volte.

```
for( i=0; i<n; i++) {
      istruzioni;
}
```

oppure

```
for(; --n; ) {
   istruzioni;
}
```

- Sommare i numeri compresi tra m ed n, con m minore di n.

```
for(somma=0, i=m; i<=n; i++) somma+=i;
```

oppure

```
for( somma=0, i=m; i<=n; somma+=i, i++);
```