Linguaggio C

Generalità

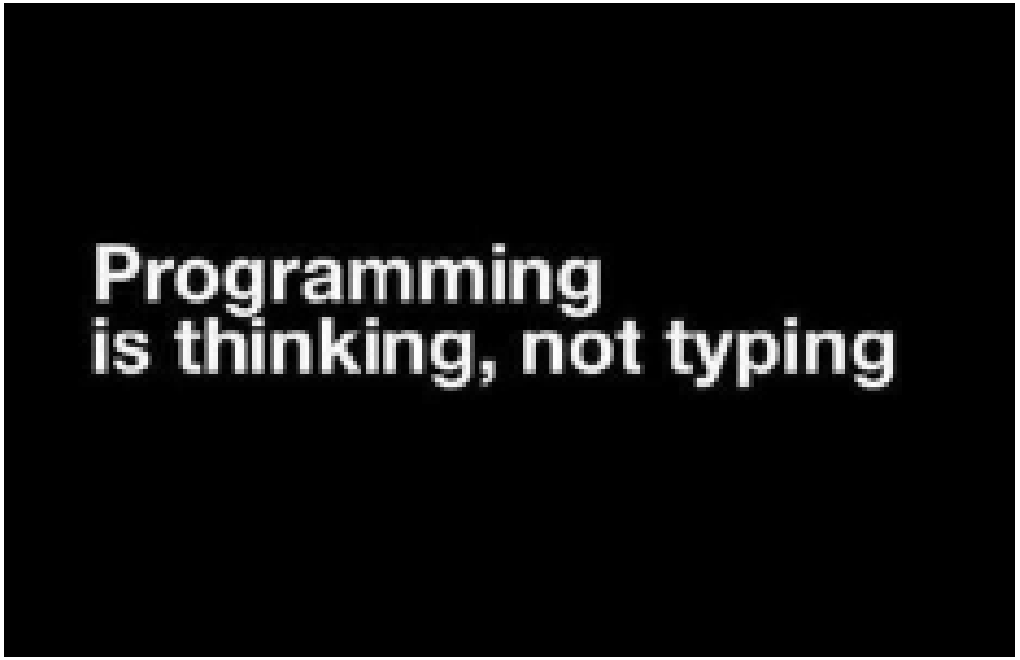
Il linguaggio C è un linguaggio:

- **procedurale**: è possibile definire funzioni e procedure, e richiamarle poi per nome.
- **dichiarativo**: prima di usare una variabile o funzione è necessario dichiararla.
- **function-centered:** al termine della fase di progettazione, viene individuato un insieme di funzioni, ognuna delle quali risolve una piccola parte del problema di programmazione
- **compilato:….**

Generalità

Il linguaggio C è un linguaggio:

"io queste cose già le so!"



Compilazione

Compilazione

Editing: scrittura del codice sorgente, file **.c** (in un IDE o con un semplice file di testo)

Compilazione

Pre-processing: Eseguito prima del compilatore, risolve direttive che iniziano con #. Ad esempio:

- **#define,** utilizzata per definire costanti
- **#include,** utilizzata per includere codice scritto in un file diverso da quello che si sta compilando.
  - La direttiva #include <nomefile.h> indica al preprocessore di cercare il file in una directory speciale, definita dal SO, dove sono contenuti i file che vengono normalmente inclusi da tutti i programmi utente.
  - La direttiva #include “nome_file.h” indica di cercare nella directory del file sorgente e, quando non reperito, seguendo il percorso classico

Compilazione

Compiling: Verifica la correttezza sintattica del codice sorgente e costruisce un file intermedio, detto " oggetto" (con estensione **.o**) che viene salvato su disco.

Compilazione

Linking: Collega i vari file oggetto costruiti dal compilatore e unisce eventuali librerie esterne**,** al fine di generare il file eseguibile.

- Il linker è generalmente un programma distinto dal compilatore: in alcuni ambienti il programma di link deve essere lanciato separatamente.
- Sebbene il codice sorgente ed oggetto può essere organizzato su diversi file, il codice eseguibile risiede in un unico file.

Compilazione

Loading: Carica in memoria e lancia l'eseguibile compilato.

•Il processo è preso in carico dalla CPU che esegue sequenzialmente le istruzioni ed eventualmente alloca memoria per creare variabili, file su disco, etc.

Esempio

```
#include <stdio.h>
/* function main begins program execution */

int main()
{
    int integer1; /* first number to be input by user */
    int integer2; /* second number to be input by user */
    int sum; /* variable in which sum will be stored */
    printf( "Enter first integer\n" ); /* prompt */
    scanf( "%d", &integer1 ); /* read an integer */
    printf( "Enter second integer\n" ); /* prompt */
    scanf( "%d", &integer2 ); /* read an integer */
    sum = integer1 + integer2; /* assign total to sum */
    printf( "Sum is %d\n", sum ); /* print sum */
    return 0; /* indicate that program ended successfully */

} /* end function main */
```

```
#include <stdio.h>
/* function main begins program execution */

int main()
{
     int integer1; /* first number to be input by
user */
     int integer2; /* second number to be input by
user */
     int sum; /* variable in which sum will be
stored */
     printf( "Enter first integer\n" ); /* prompt */
     scanf( "%d", &integer1 ); /* read an integer */
     printf( "Enter second integer\n" ); /* prompt
*/
     scanf( "%d", &integer2 ); /* read an integer */
     sum = integer1 + integer2; /* assign total to
sum */
     printf( "Sum is %d\n", sum ); /* print sum */
     return 0; /* indicate that program ended
successfully */

} /* end function main */
```

-direttive al preprocessore. Aggiunge le funzioni per gestire i flussi di *input/output*
-il *main()* è la funzione principale

-dichiariamo *variabili* di tipo intero

- stampa di una stringa, lettura di un valore e memorizzazione in una variabile in una locazione di memoria, quindi serve l'operatore *&* per referenziare l'indirizzo di quella variabile

-somma aritmetica, assegnazione di un valore a auna nuova variabile e stampa del valore. Lo specificatore di conversione *%d* serve a indicare che la variabile è di tipo intero.

Programmazione in C

- L'esempio riportato segue la organizzazione standard adottata in tutti i programmi scritti in C:
  - definizione e inizializzazione delle variabili
  - elaborazione dei dati
  - visualizzazione in output dei risultati
- C è uno dei linguaggi emblematici in cui i programmi sono scritti basandosi sulla Programmazione Strutturata (Th. Bohm-Jacopini), ovvero ricorrendo a tre **strutture di controllo** principali:
  - **Sequenza**, nativa in C. Le istruzioni sono eseguite sequenzialmente.
  - **Selezione**. In C ci sono tre costrutti di selezione *if, if-then, switch.*
  - **Iterazione**. In C ci sono tre costrutti iterativi *for, while-do, do-while.*

Esempio

- *Scrivere un programma che stampi un programma che stampi un messaggio diverso a seconda che l'utente abbia un voto d'esame maggiore o minore di 18.*
- Per ideare una soluzione per un problema si può procedere rispondendo alle seguenti:
    - Input?
    - Output?
    - Quale struttura di controllo usare?

Esempio

- *Scrivere un programma che stampi un programma che stampi un messaggio diverso a seconda che l'utente abbia un voto d'esame maggiore o minore di 18.*
- Per ideare una soluzione per un problema si può procedere rispondendo alle seguenti:
  - Input?
    - Voto dell'utente, memorizzata in una variabile di tipo intero
  - Output?
    - Messaggio, diverso a seconda che il voto sia maggiore o minore di 18
  - Quale struttura di controllo usare?
    - Struttura di selezione. Perché il programma sulla base del valore inserito può sviluppare due diverse alternative.

Esempio

- Quale struttura di controllo usare?
  - Struttura di selezione. Perché il programma sulla base del valore inserito può sviluppare due diverse alternative.

If Se il voto dello studente è maggiore di 18 Stampa “Promosso”

If…else Se il voto dello studente è maggiore di 18 Stampa “Promosso” Altrimenti Stampa “Bocciato”

Esempio

- Quale struttura di controllo usare?
  - Struttura di selezione. Perché il programma sulla base del valore inserito può sviluppare due diverse alternative.

If Se il voto dello studente è maggiore di 18 Stampa “Promosso”

```
if ( voto>= 18 )
   puts( "Promosso\n" );
```

Esempio

- Quale struttura di controllo usare?
  - Struttura di selezione. Perché il programma sulla base del valore inserito può sviluppare due diverse alternative.

If…else Se il voto dello studente è maggiore di 18 Stampa "Promosso" Altrimenti Stampa "Bocciato"

```
if ( voto>= 18 )
{
   puts( "Promosso\n" );
}
else
   puts( "Bocciato\n" );
```

Esempio

- Quale struttura di controllo usare?
  - Struttura di selezione. Perché il programma sulla base del valore inserito può sviluppare due diverse alternative.

Switch Se il voto dello studente è maggiore di 18 Stampa "Promosso" Altrimenti Stampa "Bocciato"

```
switch(voto) {
case 0:    case 1:    case 2:
case 3:    case 4:    case 5:
case 6:    case 7:    case 8:
case 9:    case 10:    case 11:
case 12:    case 13:    case 14:
case 15:    case 16:    case 17:
   puts("Bocciato\n");
   break;
default:
   puts("Promosso\n");
}
```

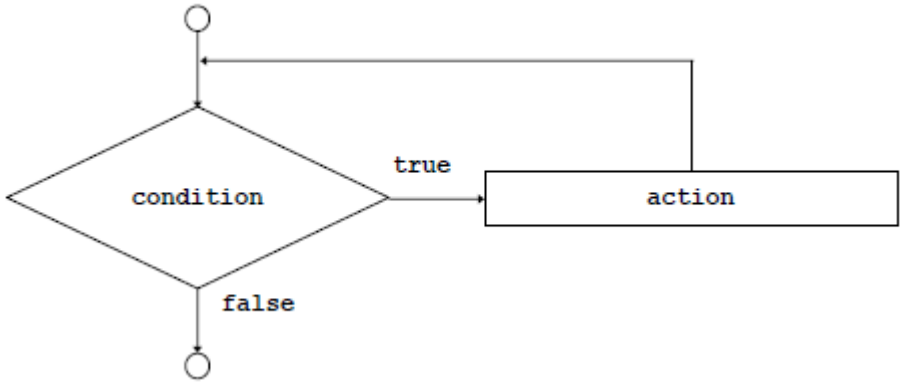
Esempio

- *Scrivere un programma che sommi il costo totale dei prodotti in un carrello che contiene cinque prodotti.*

    - Input?
        - Costo singoli prodotti
    - Output?
        - Costo totale della spesa
    - Quale struttura di controllo usare?
        - Struttura di iterazione. Il programma deve iterativamente sommare il costo di un prodotto al costo corrente.

Esempio

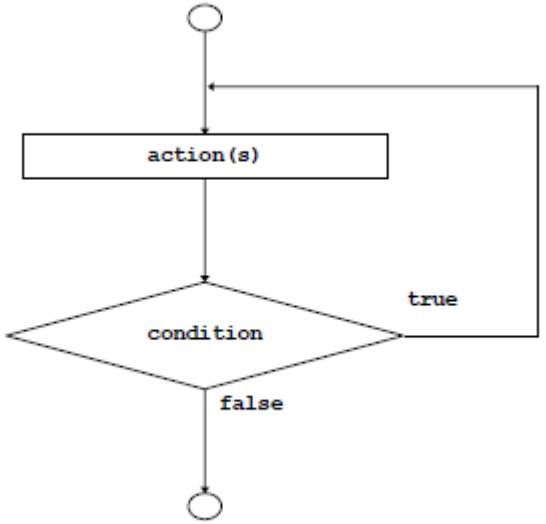
- A proposito di strutture di iterazione, ricordiamo che



  - **while-do**
  - Ripete le action finchè la condition è TRUE.

Esempio

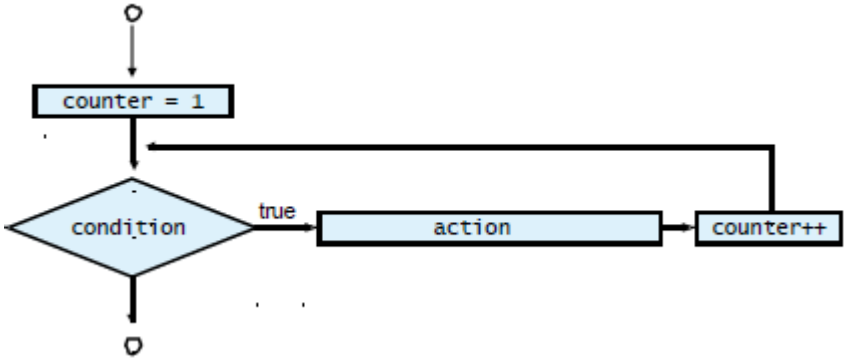
- A proposito di strutture di iterazione, ricordiamo che



  - **do-while**
  - Ripete le action (almeno una volta) finchè la condition è TRUE.

Esempio

- A proposito di strutture di iterazione, ricordiamo che



  - **for**
  - Come while-do, ripete le action (in un numero indicato dal contatore) finchè la condition è TRUE.

Esempio

- Quale struttura di iterazione usare?

while-do

```
finchè (numero_prodotti<5)
        leggi costo
        aggiungi  al costo totale
```

do-while

```
ripeti
    leggi costo
    aggiungi  al costo totale
finchè (numero_prodotti<5)
```

for

```
finchè (numero_prodotti<5)
    leggi costo
    aggiungi  al costo totale
```

Esempio

- Quale struttura di iterazione usare?

while-do finchè (numero_prodotti<5)
leggi costo
aggiungi al costo totale

```
while(products<5){
   scanf("%d",&costo);
   totale= totale + costo;
   products++;
}
```

Esempio

- Quale struttura di iterazione usare?

do-while ripeti
leggi costo
aggiungi al costo totale
finchè (numero_prodotti<5)

```
do {
   scanf("%d",&costo);
   totale= totale + costo;
   products++;
} while(products<5);
```

Esempio

- Quale struttura di iterazione usare?

for

finchè (numero_prodotti<5)
    leggi costo
    aggiungi al costo totale

```
for(products=0; products<5;products++) {
   scanf("%d",&costo);
   totale= totale + costo;
}
```

Esempio

- Quale struttura di iterazione usare?
- COMPLETARE: STUDENTI AL LAVORO!

- Vi sono casi limite? Quali?

Esempio

- Quale struttura di iterazione usare? Con while-do

```c
#include <stdio.h>

int main() {
 unsigned int costo = 0; // variabile che memorizza il costo
 unsigned int totale = 0; // variabile che memorizza il totale
 unsigned int counter = 0; // variabile contatore

 while (counter < 5) { // ciclo
   printf("Inserisci il costo del prodotto: "); // valori input
   scanf("%d", &costo); // lettura valori input

   totale = totale + costo; //aggiorna il totale, si può anche esprimere come "totale += costo";
   counter++; // incrementa il contatore, altrimenti non usciremmo mai dal ciclo
 }

 printf("Il costo totale è %d euro", totale); // stampa output
```

Esempio

- Quale struttura di iterazione usare? Con do-while

```c
#include <stdio.h>

int main() {
 unsigned int costo = 0; // variabile che memorizza il costo
 unsigned int totale = 0; // variabile che memorizza il totale
 unsigned int counter = 0; // variabile contatore

 // ciclo
 do{
   printf("Inserisci il costo del prodotto: "); // valori input
   scanf("%d", &costo); // lettura valori input

   totale = totale + costo; //aggiorna il totale, si può anche esprimere come "totale += costo";
   counter++; // incrementa il contatore, altrimenti non usciremmo mai dal ciclo
 } while (counter < 5) ;

}

 printf("Il costo totale è %d euro", totale); // stampa output
```

Esempio

- Quale struttura di iterazione usare? Con for

```c
#include <stdio.h>

int main() {
 unsigned int costo = 0; // variabile che memorizza il costo
 unsigned int totale = 0; // variabile che memorizza il totale
 unsigned int counter = 0; // variabile contatore

 // ciclo
 for(counter = 0; counter < 5; counter++) {
   printf("Inserisci il costo del prodotto: "); // valori input
   scanf("%u", &costo); // lettura valori input
   totale = totale + costo; //aggiorna il totale, si può anche esprimere come "totale += costo";
 }

 printf("Il costo totale è %u euro", totale); // stampa output
}
```

Esempio

- *Scrivere un programma che conteggi il costo totale dei prodotti in un carrello, di cui non conosciamo il numero di prodotti.*

  - Input?
    - Costo singoli prodotti
  - Output?
    - Costo totale della spesa
  - Quale struttura di controllo usare?
    - Struttura di iterazione. Il programma deve iterativamente sommare il costo di un prodotto al costo corrente, quando il numero di prodotti è sconosciuto.

Esempio

- *Scrivere un programma che conteggi il costo totale dei prodotti in un carrello, di cui non conosciamo il numero di prodotti.*

  - Input?
    - Costo singoli prodotti
  - Output?
    - Costo totale della spesa
  - Quale struttura di controllo usare?
    - Struttura di iterazione. L'iterazione sarà **non controllata,** cioè non c'è un limite massimo pre-definito, ma un valore **sentinella**: se il valore-sentinella viene letto, l'iterazione finisce.

Esempio

- Quale struttura di iterazione usare?

while-do

```
finchè (sentinella=TRUE)
        leggi costo
        aggiungi  al costo totale
```

do-while

```
ripeti
        leggi costo
        aggiungi  al costo totale
    finchè (sentinella=TRUE)
```

for

```
finchè (sentinella=TRUE)
    leggi costo
    aggiungi  al costo totale
```

Esempio

- Quale struttura di iterazione usare?

while-do finchè (sentinella=TRUE)
leggi costo
aggiungi al costo totale

```
while(products<5){
    scanf("%d",&costo);
    totale= totale + costo;
    products++;
}
```

```
while(costo !=-1){
   scanf("%d",&costo);
   totale= totale + costo;
}
```

- Il valore sentinella va nella condizione.
- Si tendono ad utilizzare valori non validi per quella variabile

Esempio

- Quale struttura di iterazione usare?

for finchè (sentinella=TRUE)
leggi costo
aggiungi al costo totale

```
for(; costo !=-1;) {
   scanf("%d",&costo);
   totale= totale + costo;
}
```

- Soluzione difforme dalla forma standard della struttura for

Esempio

- Quale struttura di iterazione usare?

do-while ripeti
leggi costo
aggiungi al costo totale
finchè (sentinella=TRUE)

```
do{
   scanf("%d",&costo);
   totale= totale + costo;
}
```

- While-do e do-while sono preferiti per iterazioni con valori sentinella.

Esempio

- Quale struttura di iterazione usare?
- COMPLETARE: STUDENTI AL LAVORO!

- Come implementare la sentinella?

Esempio

- Quale struttura di iterazione usare?

```c
#include <stdio.h>

int main() {
 unsigned int costo = 0; // variabile che memorizza il costo
 unsigned int totale = 0; // variabile che memorizza il totale
 //unsigned int counter = 0; // non serve più il contatore

  printf("Inserisci il costo del prodotto: "); // valori input
  scanf("%d", &costo); // lettura valori input

 // ciclo
 while(costo != -1) {
   totale = totale + costo; //aggiorna il totale, si può anche esprimere come "totale += costo";
   printf("Inserisci il costo del prodotto: "); // valori input
   scanf("%d", &costo); // lettura valori input
 }

 printf("Il costo totale è %d euro", totale); // stampa output
}
```

Perché?

Operatori di abbreviazione aritmetici

- int c = 3, d = 5, e = 4, f = 6, g = 12;
- Sono più efficienti perché riducono gli accessi in memoria non creando oggetti temporanei

Operatore di assegnamento	Esempio di espressione	Spiegazione	Assegna
+=	c += 7	c = c + 7	10 a c
-=	d -= 4	d = d - 4	1 a d
*=	e *= 5	e = e * 5	20 a e
/=	f /= 3	f = f / 3	2 a f
%=	g %= 9	g = g % 9	3 a g

Priorità tra gli operatori

Priorità	Operatore	Simbolo	Associatività
1 (max)	chiamate a funzione selezioni	`()` `[]  ->  .`	a sinistra

2	operatori unari:
	op. negazione
	op. aritmetici unari
	op. incr. / decr.
	op. indir. e deref.
	op. sizeof
		
		`!  ~`
		`+  -`
		`++  --`
		`&  *`
		`sizeof`
			a destra

3	op. moltiplicativi	`*  /  %`	a sinistra
4	op. additivi	`+  -`	a sinistra

Priorità tra gli operatori

Priorità	Operatore	Simbolo	Associatività
5	op. di shift	`>>` `<<`	a sinistra
6	op. relazionali	`<` `<=` `>` `>=`	a sinistra
7	op. uguaglianza	`==` `!=`	a sinistra
8	op. di AND bit a bit	`&`	a sinistra
9	op. di XOR bit a bit	`^`	a sinistra
10	op. di OR bit a bit	`|`	a sinistra
11	op. di AND logico	`&&`	a sinistra
12	op. di OR logico	`||`	a sinistra
13	op. condizionale	`?...:`	a destra
14	op. assegnamento e sue varianti	`=` `+=` `-=` `*=` `/=` `%=` `&=` `^=` `|=` `<<=` `>>=`	a destra
15 (min)	op. concatenazione	`,`	a sinistra

Esercizi

- *Scrivere un programma che stampi un messaggio diverse a seconda che l'utente sia maggiorenne/minorenne o pensionato/lavoratore.*
- *Scrivere un programma che consenta di svolgere una delle 4 operazioni aritmetiche fondamentali tra due dati in input sulla base della scelta di menù operata dall'utente.*

- STUDENTI AL LAVORO!