Introduzione al C

Lezione 1

Ing. Colazzo Sebastiano

I linguaggi di programmazione

- **Linguaggi macchina**: lingua naturale di un particolare computer, esso consiste di sequenze di numeri (1 o 0) e dipende dalla macchina

  0000111101
  1111110001
  0011001111

- **Linguaggi assembly**: abbreviazioni simili all'inglese, per rappresentare le operazioni elementari del computer

  ***LOAD LOAD** N*
  ***ADD ADD** X*

- **Linguaggi ad alto livello**: singole istruzioni contenenti notazioni matematiche utilizzate comunemente

  x = n + 2;

- ***Il C è un linguaggio di alto livello***

Programmi compilati vs. interpretati

- I linguaggi compilati
  - C, C++, Delphi, Visual Basic
  - si scrive il codice in un editor
  - il codice viene poi controllato per verificare che non ci siano errori e poi viene “compilato”
    - ogni istruzione viene trasformata nel corrispondente codice in linguaggio macchina
  - hanno il vantaggio di prestazioni migliori.
- I linguaggi intrepretati
  - PHP
  - il codice sorgente viene “interpretato” al volo e vengono, quindi, eseguite le istruzioni così come descritte nel codice sorgente
  - alta portabilità e immediatezza tra quello che si scrive e quello che viene presentato all'esecuzione del programma
  - rimangono dei problemi come la ricerca di errori nel codice sorgente o il carico di lavoro maggiore per il processore
- La via di mezzo – JAVA
  - è sia compilato che interpretato
  - il codice sorgente viene compilato in un formato intermedio (chiamato bytecode), il quale a sua volta viene interpretato dalla Java Virtual Machine (JVM) che ha il compito di interpretare "al volo" le istruzioni bytecode in istruzioni per il processore

Le peculiarità del C

- **Dimensioni del codice ridotte**
- **Dimensioni dell'eseguibile ridotte**
- **Efficienza dei programmi**
- **Può essere compilato su una vasta gamma di computer**
- **È un linguaggio di alto livello**
- **Può maneggiare attività di basso livello**
- **Implementazione dei puntatori**
- **Loose Typing**

La storia del C (1)

- La nascita del linguaggio C e' collegata a quella del sistema operativo UNIX
- UNIX nasce nel 1969 ad opera di Ken Thompson
- La prima versione di unix e' scritta in assembly PDP-7
- Il linguaggio C viene sviluppato da Dennis Ritchie nel 1971

- Obiettivi del C furono:
  - la facilita' di programmazione (rispetto all'assembly)
  - che fosse sufficientemente compatto da funzionare sui calcolatori dell'epoca

La storia del C (2)

- La seconda versione di Unix e' scritta in C nel 1973
- Il linguaggio e' documentato in Kernighan, Ritchie, *The C Programming Language*, 1978
- Il linguaggio C viene standardizzato da ANSI nel 1989

Il primo programma in C

```
/* Il mio primo programma in C */

   main ()
   {
     printf(“Welcome to C! \n ");
   }
```

Elementi fondamentali (1)

/* Il mio primo programma in C */

- E’ un commento
- In C esistono 2 metodi per inserire i commenti:
  - // - Tutto quello che sta a destra sulla medesima riga viene considerato commento
  - /* ... */ - Tutto quello che è compreso tra i due asterischi viene considerato commento

Elementi fondamentali (3)

```
{
/* statement */
}
```

- Servono per delimitare le istruzioni (o "statement“)
- Le istruzioni vengono eseguite in ordine, da quella più in alto, giù fino all'ultima

Elementi fondamentali (2)

main()

- E’ la funzione principale in un qualsiasi programma in C
- Può avere dei parametri
- E’ indispensabile ed unico
- Deve esserci sempre

Elementi fondamentali (4)

printf(“Welcome to C! \n ");

- È una *“funzione”*
- **printf** è adibita a stampare a video tutto quello che gli viene passato come *“argomento”*
- fa parte della libreria **<stdio.h>** (le vedremo in seguito)

NB. Il *“;”* serve per chiudere un'istruzione

Le sequenze di escape

- \n → newline
- \t → tabulazione orizzontale
- \r → ritorno carrello
- \a → allarme
- \\ → backslash
- \’ → apice singolo
- \” → virgolette

Il secondo programma in C

```
/* programma di addizione */
#include <stdio.h>
main()
{
    int integer1, integer2, sum; // dichiarazioni
    printf(“Enter first integer: \n”);
    scanf(“%d”, &integer1);
    printf(“Enter second integer: \n”);
    scanf(“%d”, &integer2);
    sum = integer1 + integer2;
    printf(“Sum is %d\n”, sum);
    return 0; // il programma è terminato in modo corretto
}
```

Le direttive di inclusione

#include <stdio.h>

- Le *direttive di inclusione* sono certamente le più comunemente usate, infatti vengono utilizzate per includere i file con le definizioni delle funzioni di libreria

- Alcuni dei file piu comunemente usati sono:
  - stdio.h
  - stdlib.h
  - string.h
  - math.h
  - time.h

- Esistono altri tipi di direttive:
  - Definizioni e macroistruzioni
  - Direttive condizionali

Le variabili

int integer1, integer2, sum;

- E’ una dichiarazione di “variabili”
- Le variabili non sono altro che dei contenitori, identificati da un “nome” o “identificatore” univoco (integer1,integer2,sum)
- Oltre che dal nome le variabili vengono definite da un “tipo” (int)
- Un identificatore è costituito da una o più lettere, cifre o caratteri e deve iniziare con una lettera o il carattere di sottolineatura (underscore "_")
- Il C è case-sensitive, quindi si fa distinzione tra lettere maiuscole e lettere minuscole.
- Il tipo della variabile indica quale tipo di valori può assumere il contenuto della variabile stessa.
  - Domanda: Che differenza c’è tra l’intero ‘7’ e il carattere ‘7’ ?
- Tutte le variabili, prima di essere utilizzate, devono essere “dichiarate” (una ed una sola volta) e successivamente devono essere “inizializzate”

I tipi delle variabili

Tipi di dichiarazione	Rappresentazione	N. di byte
char	Carattere	1 (8 bit)
int	Numero intero	2 (16 bit)
short	Numero intero "corto"	2 (16 bit)
long	Numero intero "lungo"	4 (32 bit)
float	Numero reale	4 (32 bit)
double	Numero reale "lungo"	8 (64 bit)

Domanda:

- Come è possibile fare quando vogliamo memorizzare una successione di caratteri?

Ottenere un valore da un utente

```
scanf("%d", &integer1);
…
scanf("%d", &integer2);
```

- Prende i dati in ingresso dallo standard input (di solito la tastiera)
- L'inserimento di un valore è confermato con il tasto invio
- Ha due argomenti:
  - **%d** → specifica di conversione, indica che il dato immesso deve essere un intero
  - **&integer1** → & si chiama "operatore di indirizzo", integer1 è il nome della variabile utilizzata per memorizzare il valore immesso

L’istruzione di assegnamento

sum = integer1 + integer2;

- “=“ si chiama “operatore di assegnamento”
- “+” e “=“ si chiamano “operatori binari”

Visualizzare il valore di una variabile

printf(“Sum is %d\n”, sum);

- printf è la stessa di prima ma questa volta ha due “parametri”
  - I caratteri da stampare e la specifica di conversione (%d)
  - La variabile di cui vogliamo stampare il valore

- Potevamo anche scrivere
  - printf(“La somma tra %d e %d è %d\n”, integer1,integer2,sum);

Le specifiche di conversione

Stringa di controllo	Cosa viene stampato
%d, %i	Intero decimale
%f	Valore in virgola mobile
%c	Un carattere
%s	Una stringa di caratteri
%o	Numero ottale
%x, %X	Numero esadecimale
%u	Intero senza segno
%f	Numero reale (float o double)
%e, %E	Formato scientifico
%%	Stampa il carattere %

- Possono essere utilizzate sia con printf che con scanf

Introduzione agli Operatori

Permettono di “lavorare” con le variabili:

- **Operatori aritmetici.** Comprendono somma, sottrazione, moltiplicazione, divisione intera, divisione con modulo ecc.
- **Operatori di confronto.** Operatori che permettono di verificare determinate condizioni, come ad esempio l'uguaglianza o la disuguaglianza.
- **Operatori logici.** Da utilizzare con le istruzioni condizionali ed iterative.

Operatori Aritmetici

Operazioni con gli int	Simbolo	Esempio
Addizione	+	4 + 27 = 31
Sottrazione	-	76 - 23 = 53
Moltiplicazione	*	4 * 7 = 28
Divisione intera	/	10 / 3 = 3 (3 è il n di volte divisibili senza resto)
Divisione con modulo	%	11 / 6 = 5 (5 è il resto della divisione)

Operazion con i double	Simbolo	Esempio
Addizione	+	2.5 + 14.3 = 16.8
Sottrazione	-	43.8 - 12.7 = 31.1
Moltiplicazione	*	7.5 * 3.0 = 22.5
Divisione	/	5.0 / 2.0 = 2.5

- In C esistono anche altri tipi di operatori (++,--) che vedremo in seguito

Operatori di Confronto

Simbolo	Significato	Utilizzo
==	uguale a	a == b
!=	diverso da	a != b
<	minore	a < b
>	maggiore	a > b
<=	minore o uguale	a <= b
>	maggiore o uguale	a >= b

Operatori Logici

Simbolo	Significato	Utilizzo
&&	AND logico	a && b
||	OR logico	a || b

- restituiscono 1 quando sono vere e 0 quando sono false
  - "a || b" == 1 se e solo se "a" o "b" valgono uno
  - "a || b“ == 0 se entrambi gli operandi valgono zero
  - "a && b“ == 0 se "a" o "b" valgono zero
  - “a && b” == 1 se entrambi valgano uno.
- && e || sono diversi da & e | che rappresentano il bitwise AND ed il bitwise OR

Controlli condizionali: If-Else

```
if (risultato_esame >=18)
{
   printf ("Complimenti hai superato l'esame");
   int passato = 1;
} else {
   printf ("Non hai superato l'esame");
   int passato = 0;
}
```

- in C non esiste il tipo booleano
  - quando deve essere *verificata una condizione* essa risulta falsa se vale zero e vera altrimenti.

Riferimenti

- Libro di testo
- Corso di su www.html.it