La Programmazione

Cos’è la programmazione?

Concetti preliminari

Sommario

- La programmazione, questa sconosciuta
- Programmiamo Macchine Astratte
- Linguaggi di basso e alto livello e loro implementazione
- Esempio: C

Programmi

- Nonostante l'abbondanza di programmi disponibili esiste sempre l'esigenza di scrivere programmi nuovi o di migliorarne di già esistenti
- In ogni caso esistono dei limiti a quello che un computer può calcolare: ci sono dei problemi per cui non esiste nessun **algoritmo** risolutivo
- Un algoritmo è il cuore di un programma: è il procedimento di calcolo che il programma deve seguire

Programmare

- Per poter programmare una certa macchina abbiamo bisogno di:
  - Un algoritmo che calcola ciò che il programma deve calcolare
  - Un linguaggio per specificare l'algoritmo. La macchina su cui vogliamo far girare il programma deve saper capire questo linguaggio
- In genere il programma che si scrive è una serie di istruzioni

Es: Una macchina fisica

- Operazioni Primitive
  - Operazioni aritmetico-logiche
  - Operazioni di manipolazione di stringhe di bit
  - Lettura/Scrittura di celle di memoria e registri
  - Input/output
- Controllo di Sequenza
  - salti
  - condizionali
  - chiamate e ritorni dai sottoprogrammi

Programmare in un linguaggio

- Conoscere un linguaggio di programmazione corrisponde a conoscere tutti i suoi costrutti e come questi vengono eseguiti dalla relativa macchina astratta
- Avendo queste conoscenze si possono scrivere programmi nel linguaggio scelto e si può anche non conoscere per niente il tipo di implementazione della macchina astratta del linguaggio

Linguaggi di alto livello

- Richiedono una implementazione per poter essere poi eseguiti in una macchina fisica ospite
- Esistono diversi modi per implementarli:
  - Compilazione o traduzione
  - Interpretazione
  - Varie combinazioni delle due

Compilazione

- Un compilatore è un programma che si occupa di tradurre un altro programma scritto in un linguaggio L in un programma **equivalente** scritto in L0, linguaggio di una macchina fisica ospite

Compilazione

Programma scritto in L

```
int main() {
double x;
int z = 10;
while (z >0)
......
```

Compilatore

Programma equivalente in L0

```
MOV l1, R1
MOVF l3, R2
....
```

Compilazione

- Molti linguaggi sono compilati: C, C++, Pascal, FORTRAN, COBOL, Java (parzialmente)
- Scrivere un compilatore è un compito difficile, ma una volta fatto esso permette di far eseguire tutti i programmi del linguaggio L per cui è stato scritto da macchine che funzionano con il linguaggio L0

Interpretazione

- Il linguaggio L è implementato su una macchina che funziona su L0 tramite una simulazione software
- Un programma in L0 (generalmente chiamato interprete) si occupa di prelevare nel giusto ordine le istruzioni di un programma dato in L, di simulare la loro esecuzione e di restituirne i risultati.
- Esempi di linguaggi interpretati: PROLOG, LISP, Java bytecode

Interpretazione

Programma scritto in L

```
int main() {
double x;
int z = 10;
while (z >0)
.....
```

Interprete

simulazione

Esempio: linguaggio C

- Ad alto livello
- Compilato
- Il codice compilato può essere eseguito sulla macchina ospite con il supporto di alcune librerie che implementano diverse funzionalità (supporto a tempo di esecuzione – *run time library*)

Codifica degli algoritmi in un linguaggio di alto livello

Elementi -e terminologia- essenziali

- **Standard Input, Standard Output**
- **Stringa**
- **Celle di memoria o variabili**
- **Identificatori simbolici.**
- **Identificatori predefiniti e riservati**
- **Parole chiave**

Struttura *sintattica* di un programma C

- Un programma C è composto da:
  - un'intestazione seguita da
  - una sequenza di istruzioni racchiusa tra i simboli **{** e **}**.
- L'**intestazione** è costituita dall'identificatore predefinito main seguito da una coppia di parentesi ( ) (per il momento vuote)
- Le **istruzioni** sono *frasi* del linguaggio di programmazione; ognuna di esse termina con il simbolo ';'.

Le principali istruzioni del C (1)

1. **Istruzione di assegnazione (si usano variabili e/o costanti)**

```
x = 23;
w = 'a';
y = z;
r3 = (alfa*43–xgg)*(delta–32*ijj);
x = x+1;
```

2. **Istruzioni di ingresso e uscita (**scanf o printf) (si noti che sono chiamate di libreria)

3. **Istruzioni composte**

   3.1 Istruzione condizionale

   3.2 Istruzione iterativa (ciclo o loop).

Le principali istruzioni del C (2)

Le istruzioni composte

3. Istruzioni composte

3.1 Istruzione condizionale

```
if(x == 0) z = 5; else y = z + w*y;
if(x == 0) {z = 5;} else {y = z + w*y;}
if ((x+y)*(z-2) > (23+v)) {z = x + 1; y = 13 + x;}
if ((x == y && z >3) || w != y) z = 5; else {y = z + w*y; x = z;}
```

Istruzioni scorrette:

```
if (x == 0) else y = z; y = 34;
if (x == 0) a; else b + c;
```

3.2 Istruzione iterativa (ciclo o loop)

```
while (x >= 0) x = x – 1;
while (z != y) {y = z – x; x = x*3;}
```

Gli array

- Un array (o vettore) viene identificato come qualsiasi altra variabile
- Però anche i suoi elementi sono variabili
- Ad essi si accede mediante un *indice*:
- esempi:
  - scanf(s[2]);
  - a[3] = s[1] + x;
  - **if** (a[4] > s[1] + 3) s[2] = a[2] + a[1];
  - x = a[i];
  - a[i] = a[i+1];
    a[i*x] = s[a[j+1]–3]*(y – a[y]);

- in C il primo elemento di ogni array è sempre lo 0-esimo

Esecuzione di programmi C su macchine reali

Indice

Che cosa manca per poter “far girare i programmi”

- La struttura di un programma C
  - La parte dichiarativa
  - L’ I/O

Un “vero” programma C

```
/* Programma SommaSequenza */

#include <stdio.h>
main()
{
        int  numero, somma;
        somma = 0;
        scanf("%d", &numero);
        while (numero != 0)
        {
                somma = somma + numero;
                scanf("%d", &numero);
        }
        printf("La somma dei numeri digitati è: %d\n", somma);
}
```

La struttura dei programmi C

- Un programma C deve contenere, nell'ordine:
  - Una parte contenente **direttive** per il compilatore. Per il momento trascuriamo questa parte.
  - L'identificatore predefinito main seguito dalle parentesi ()
  - Due parti, racchiuse dalle parentesi **{}**:
    - la **parte dichiarativa**;
    - la **parte esecutiva**.
- La parte dichiarativa elenca tutti gli elementi che fanno parte del programma, con le loro principali caratteristiche.
- La parte esecutiva consiste in una successione di istruzioni come già descritto in generale.

La parte dichiarativa (1)

- Tutto ciò che viene usato va dichiarato. In prima istanza:
  - Dichiarazione delle costanti;
  - Dichiarazione delle variabili.
- Perché questa fatica … inutile?
  - Aiuta la **diagnostica** (ovvero *segnalazione di errori*):

    ```
    x = alfa;
    alba = alfa + 1;
    ```

  - Senza dichiarazione, alba è una nuova variabile!

- Principio importante:
  meglio un po' più di fatica nello scrivere un programma che nel leggerlo -e capirlo!

La parte dichiarativa (2)

- Una dichiarazione di variabile consiste in:
  - Uno **specificatore di tipo**, seguito da
  - Una lista di uno o più identificatori di variabili separati da una virgola
  - Ogni dichiarazione termina con ';'
- Importanza del concetto di *tipo* di dato

La parte dichiarativa (3)

- Esempi di dichiarazioni di variabili:

```
float   x,y;
int     i,j;
char    simb;
```

- Equivalenti a:

```
float   x;
int     i,j;
char    simb;
float   y;
```

La parte dichiarativa (4)

- La dichiarazione di costanti:

```
const   float   PiGreco   = 3.14;
const   float   PiGreco   = 3.1415, e = 2.718;
const   int     N = 100, M = 1000;
const   char    CAR1 = 'A', CAR2 = 'B';
```

- Un' eventuale assegnazione a una costante sarebbe segnalata come errore dal compilatore.

- AreaCerchio = PiGreco*RaggioCerchio*RaggioCerchio;

  è equivalente a:

  AreaCerchio = 3.14*RaggioCerchio*RaggioCerchio;

  (se si fa riferimento alla prima dichiarazione di PiGreco)

Le istruzioni di I/O

- Istruzione per l'output:

  printf (*stringa di controllo*, *elementi da stampare*);

- Stringa di controllo
  - caratteri di conversione
  - caratteri di formato

- L'insieme degli elementi da stampare è una lista di variabili, di costanti o di espressioni composte con variabili e costanti

Le istruzioni di I/O

- Istruzione per l'input:

  scanf (*stringa di controllo*, *elementi da leggere*);

- I nomi delle variabili da leggere sono preceduti dall'operatore unario *&*

- Esempio 4.3:

```
scanf("%c%c%c%d%f", &c1, &c2, &c3, &i, &x);
```

La direttiva #include

- *Ogni programma che utilizza al suo interno le funzioni* printf e scanf *deve dichiarare l'uso di tali funzioni nella parte direttiva che precede il programma principale:*

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

- È una direttiva data a una parte del compilatore, chiamata *preprocessore*,

Primi esempi… “che girano” (1)

- Il primo programma C:

```
                                        /* PrimoProgrammaC */
#include <stdio.h>
main()
{
    printf("Questo è il mio primo programma in C\n");
}
```

- NB: niente dichiarazioni!

Primi esempi… “che girano” (2)

```
/* Programma SommaDueInteri */
#include <stdio.h>
main()
{
    int  a, b, somma;
    printf (“inserisci come valore dei due addendi due numeri interi\n”);
    scanf("%d%d", &a, &b);
    somma = a + b;
    printf("La somma di a+b è:\n%d \nArrivederci!\n", somma);
}
```

- Se vengono inseriti i dati 3 e 5, l’effetto dell’esecuzione del programma sullo Standard Output è il seguente:

  ```
  La somma di a+b è:
  8
  Arrivederci!
  ```

- Se fossero stati omessi i primi due simboli \n nella stringa di controllo?