Programmazione I - Laboratorio

Esercitazione 1 - Variabili, costanti, strutture di controllo e input/output

Gianluca Mezzetti[1] Paolo Milazzo[2]

1. Dipartimento di Informatica, Università di Pisa
http://www.di.unipi.it/∼mezzetti
mezzetti@di.unipi.it

2. Dipartimento di Informatica, Università di Pisa
http://www.di.unipi.it/∼milazzo
milazzo@di.unipi.it

Corso di Laurea in Informatica
A.A. 2012/2013

Hello, World (1)

Il primo programma C...

```
#include <stdio.h>

int main()
{
  printf("Hello, world!\n");
}
```

Quando lo si esegue scrive sullo schermo il messaggio *Hello, world!*

Vediamolo in dettaglio...

Hello, World (2)

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

E' una direttiva (da usare all'inizio del programma) che dice al compilatore che intendiamo usare le funzioni di input/output messe a disposizione dalla libreria standard del linguaggio C

```
int main()
```

Definisce la funzione principale del nostro programma (deve sempre esserci una funzione `main` da cui inizia l'esecuzione)

```
{
  printf("Hello, world!\n");
}
```

Chiama la funzione di libreria `printf` per stampare su schermo; `\n` indica il new line; le parentesi graffe indicano l'inizio e la fine del corpo della funzione `main`

Eseguire un programma C (in breve)

Per poter eseguire un programma C è necessario:

1. Scriverlo usando un qualunque editor di testi
   - Editor di testo per Linux: vim, emacs, gedit, ...
   - Editor di testo per Windows: blocco note, notepad++, ...
   - Editor di testo per MacOS: TextEdit, ...
2. Salvarlo come file con estensione `.c`
   - Esempio: `helloworld.c`
3. Trasformarlo in un file eseguibile (binario) tramite un compilatore C
   - Compilatore C per Linux: gcc, ...
   - Compilatore C per Windows: MinGW (include gcc), ...
   - Compilatore C per MacOS: X Code Tools (include gcc), ...
4. Eseguire il file generato dal compilatore
   - Su Linux e MacOS: `a.out`
   - Su Windows: `a.exe`

Inserire commenti nel codice di un programma (1)

All'interno dei programmi è possibile inserire brevi commenti utili per il programmatore

- Aiutano a comprendere il funzionamento del programma a chi lo legge
- I commenti saranno trascurati dal compilatore

Un commento è un frammento di testo che inizia con /* e termina con */

- Può consistere di più linee di testo
- Può essere usato per disabilitare temporaneamente un frammento di codice (trasformandolo in un commento)

Inserire commenti nel codice di un programma (2)

Aggiungiamo qualche commento al programma che stampa *Hello, world!*

```
/* inclusione della libreria standard */
#include <stdio.h>

/* funzione principale del programma */
int main()
{
  /*
   * utilizzo della funzione
   * di libreria printf
   */
  printf("Hello, world!\n");
}
```

Variabili (1)

Per vedere come si utilizzano le variabili in C scriviamo un programma che usa la formula $C = (5/9)(F - 32)$ per stampare la seguente tabella di traduzione Fahrenheit/Celsius

```
0     -17
20    -6
40    4
60    15
80    26
100   37
120   48
140   60
160   71
180   82
200   93
220   104
240   115
260   126
280   137
300   148
```

Variabili (2)

Il programma che stampa la tabella è il seguente:

```
/* inclusione della libreria standard */
#include <stdio.h>

/* stampa la tabella Fahrenheit-Celsius */
int main()
{
  int fahr, celsius;
  int lower, upper, step;

  lower = 0;          /* limite inferiore della tabella */
  upper = 300;        /* limite superiore della tabella */
  step = 20;          /* incremento */

  fahr = lower;
  while (fahr <= upper) /* ciclo sulle righe della tabella */
  {
    /* calcolo e stampa di una riga */
    celsius = 5 * (fahr-32) / 9;
    printf("%d\t%d\n", fahr, celsius);
    fahr = fahr + step;
  }
}
```

Variabili (3)

Le variabili devono essere dichiarate all'inizio di un blocco (area delimitata da parentesi graffe) indicando il tipo e uno (o più) nomi di variabili.

- I tipi di base sono: `int`, `float`, `char`, `short`, `long` e `double`

```
int fahr, celsius;
int lower, upper, step;
```

Alle variabili possono essere assegnati valori (del tipo opportuno), anche ottenuti da altre variabli o come risultato di una espressione

```
lower = 0;
upper = 300;
step = 20;
fahr = lower;
celsius = 5 * (fahr-32) / 9;
```

E' anche possibile (prestando attenzione) forzare il cambiamento di tipo di una espressione (cast)

```
int x=5; int y=10;
float f=((float) x) / ((float) y);
```

Cicli (1)

Continuando a seguire l'esempio vediamo i comandi per realizzare cicli nel linguaggio C

- `while ( `*cond*` ) { `*corpo*` }`
- `do { `*corpo*` } while ( `*cond*` )`
- `for ( `*com*1` ; `*cond*` ; `*com*2` ) { `*corpo*` }`

Il comando `while` ripete i comandi contenuti in *corpo* fintanto che la condizione booleana *cond* è vera (la forma `do...while` esegue sempre almeno una iterazione)

Il comando `for` esegue i comandi in *com*1 (tipicamente assegnamenti di variabili usati come contatori), poi ripete i comandi contenuti in *corpo* e i comandi in *com*2 (tipicamente aggiornamenti dei contatori) fintanto che la condizione booleana *cond* è vera

Cicli (2)

- Esempio di ciclo `while`:

```
fahr = lower;
while (fahr <= upper)
{
  celsius = 5 * (fahr-32) / 9;
  printf("%d\t%d\n", fahr, celsius);
  fahr = fahr + step;
}
```

- Esempio di ciclo `for` (equivalente al precedente ciclo `while`):

```
for (fahr=lower; fahr<=upper; fahr=fahr+step)
{
  celsius = 5 * (fahr-32) / 9;
  printf("%d\t%d\n", fahr, celsius);
}
```

Comandi di selezione condizionale (1)

I comandi di selezione condizionale in C sono i seguenti:

- `if ( cond ) { com1 } else { com2 }`
- `switch ( exp )`
  `{`
  `case val1 : com1`
  `case val2 : com2`
  `...`
  `default : com`
  `}`

Il comando `if` esegue alternativamente *com1* o *com2* a seconda che l'espressione *cond* sia rispettivamente vera o falsa

- Il ramo `else` può essere omesso
- Si possono eseguire selezioni successive usando `else if`

Esempio:

```
if (x<0) { printf("negativo"); }
else if (x>0) { printf("positivo"); }
     else { printf("zero"); }
```

Comandi di selezione condizionale (2)

Il comando `switch` è una struttura di scelta plurima che controlla se un'espressione *exp* assume un valore all'interno di un insieme di costanti intere (numeri decimali o singoli caratteri) *val*1, *val*2, ..., *valN*

Se il valore di *exp* è uno di quelli considerati, verrà eseguito il comando corrispondente, altrimenti verrà eseguito il comando associato al caso `default`

Per garantire che solo uno dei comandi associati ai casi venga eseguito è necessario utilizzare il comando `break` al termine di ogni caso

Comandi di selezione condizionale (3)

Esempi:

```
switch (day)
{
  case 1: printf("lunedi\n");
          break;
  case 2: printf("martedi\n");
          break;
  case 3: printf("mercoledi\n");
          break;
  case 4: printf("giovedi\n");
          break;
  case 5: printf("venerdi\n");
          break;
  case 6: printf("sabato\n");
          break;
  case 7: printf("domenica\n");
          break;
  default: printf("valore errato\n");
}
```

```
switch (c) {
  case 'a': case 'e': case 'i': case 'o': case 'u':
      printf("vocale minuscola\n"); break;
  case 'A': case 'E': case 'I': case 'O': case 'U':
      printf("vocale maiuscola\n"); break;
  default: printf("altro simbolo\n");
}
```

Costanti simboliche (1)

Nell'esempio di programma che genera la tabella Fahrenheit/Celsius alcune variabili (`lower`,`upper` e `step`) mantengono sempre lo stesso valore

- In effetti queste variabili rappresentano parametri del programma che rimangono costanti

Un modo per definire parametri costanti in C è tramite la direttiva `#define` da usare all'inizio del programma

Un altro modo per definire costanti in C è tramite il modificatore `const`, che non vedremo...

Costanti simboliche (2)

Esempio:

```
#define LOWER 0
#define UPPER 300
#define STEP  20
```

Le costanti `LOWER`, `UPPER` e `STEP` potranno essere usate ovunque nel programma come se fossero i corrispondenti valori a loro assegnati

- Il compilatore sostituirà a ogni occorrenza di una costante il corrispondente valore durante la traduzione del programma in file eseguibile

Costanti simboliche (3)

Modifichiamo ora il programma di traduzione Fahrenheit/Celsius usando costanti e un ciclo `for` al posto del `while`

```
#include <stdio.h>

#define LOWER   0       /* limite inferiore della tabella */
#define UPPER   300     /* limite superiore della tabella */
#define STEP    20      /* incremento */

/* stampa la tabella Fahrenheit-Celsius */
int main()
{
  int fahr, celsius;

  for (fahr = LOWER; fahr <= UPPER; fahr = fahr+STEP)
  {
    celsius = 5 * (fahr-32) / 9;
    printf("%d\t%d\n", fahr, celsius);
  }
}
```

Funzioni di input/output di base (1)

Il modo più semplice per scrivere messaggi di output è tramite la funzione `printf`, definita come segue:

```
int printf(const char *format, arg list .....)
```

Il primo argomento `format` è in sostanza la stringa da stampare. Per poter includere in essa i valori di alcune variabili bisogna utilizzare dei "segnaposto" i cui più comuni sono definiti come segue:

%d	numero decimale
%ld	numero decimale lungo (di tipo `long`)
%f	numero reale come numero con la virgola
%e	numero reale in notazone esponenziale
%x	numero in notazione esadecimale
%o	numero in notazione ottale
%c	singolo carattere
%s	stringa

Ad ogni segnaposto dovrà corrispondere una variabile (prese nell'ordine) in `arg list`

Funzioni di input/output di base (2)

Esempi di uso di `printf`

```
int x = 4;
char *s = "passi";
printf("facciamo %d %s",x,s);
```

Corrisponde al seguente output: `facciamo 4 passi`

```
float x = 0.000001;
printf("%f e' uguale a %e",x,x);
```

Corrisponde al seguente output: `0.000001 e' uguale a 1.000000e-06`

Funzioni di input/output di base (3)

Per formattare il testo dei messaggi i output si possono usare caratteri di escape

- `\n` manda a capo il testo (si usa spesso come ultimo carattere della stringa stampata)
- `\t` inserisce una tab usato per dare un aspetto tabellare all'output (come nell'esempio Fahrenheit/Celsius)

Esempio:

```
printf("%d\t%d\n", fahr, celsius);
```

Funzioni di input/output di base (4)

Inoltre, i segnaposto prevedono dei parametri (flags) di visualizzazione. Ad esempio:

- `%6d` stampa un intero decimale in un campo di almeno sei caratteri
- `%6f` stampa un numero reale in un campo di almeno sei caratteri
- `%.2f` stampa un numero reale con due cifre dopo la virgola
- `%6.2f` stampa un numero reale in un campo di almeno sei caratteri e con due cifre dopo la virgola

Esempio:

```
int a=1; int b=20; int c=300;
printf("| %3d | %3d | %3d |",a,b,c);
```

corrisponde al seguente output: `|   1 |  20 | 300 |`

Funzioni di input/output di base (5)

Un modo semplice per leggere un input da tastiera (standard input, stdin) è tramite la funzione `scanf`

E' simile a `printf`: richiede una stringa di formato e una lista di (indirizzi di) variabili in cui memorizzare i dati ricevuti

Esempi (notare la & prima dei nomi delle variabili):

- Leggere una data:

```
scanf("%2d/%2d/%4d", &giorno, &mese, &anno);
```

- Leggere un'operazione aritmetica:

```
scanf("%f %c %f", &num1, &simb, &num2);
```

ESERCIZI

Variabili, costanti, input/output

Esempio svolto - rettangolo.c

Scrivere un programma che chieda all'utente due valori che rappresentano i lati di un rettangolo, e stampi il perimetro e l'area del rettangolo risultante.

```
#include <stdio.h>

int main()
{
  int base,altezza;

  /* leggo la base */
  printf("Base: ");
  scanf("%d",&base);

  /* leggo l'altezza */
  printf("Altezza: ");
  scanf("%d",&altezza);

  /* stampo l'area */
  printf("Area: %d\n",base*altezza);

  return 0;
}
```

Esercizio 1 - bisestile.c

Un anno bisestile e' identificato da un intero maggiore di 1584 che sia divisibile per 4 ma non per 100, oppure che sia divisibile per 400. Scrivere un programma che chieda in input un anno e stampi BISESTILE se l'anno inserito è bisestile, e NON BISESTILE altrimenti.

Esercizio 2 - operazione.c

Scrivere un programma che legga un valore di tipo `float`, uno di tipo `char` e poi ancora uno di tipo `float` e infine:

- se il carattere letto e' una delle quattro operazioni aritmetiche allora calcoli e stampi il risultato della corrispondente operazione sui due valori numerici;
- se e' un altro carattere allora stampi un messaggio d'errore

ESERCIZI

Cicli (non usare array)

Esempio svolto - binaria.c (1)

Scrivere un programma che letta una sequenza di 0 e di 1 di dimensione prefissata K, stampi il numero intero la cui rappresentazione binaria su K cifre e' la sequenza letta.

Esempio di esecuzione :

```
Digitare una sequenza di 0 e 1 lunga 5: 0 1 1 1 0
Il numero intero corrispondente e': 14
```

Note:

- usare una costante per K
- stampare un messaggio di errore se l'input non è corretto

Esempio svolto - binaria.c (2)

```c
#include <stdio.h>

#define K 5

int main() {
  int i;
  int num=0, val;
  int errore=0;

  printf("Digitare %d cifre 0/1 separate da spazi:\n", K);

  /* leggo ed elaboro le cifre una per volta */
  for(i=0; i<K && !errore; i++) {
    scanf("%d", &val);

    if ((val!=0)&&(val!=1))
    {
      printf("Errore nella sequenza\n");
      errore=1;
    }
    else if(i==0) num = val;
    else num = num * 2 + val;
  }
  if (!errore) printf("Il numero intero e': %d\n", num);
  return 0;
}
```

Esercizio 3 - mediaponderata1.c

Scrivere un programma che chieda ad uno studente i voti degli esami e il loro peso in crediti, uno per volta. Lo studente dovra' inserire 0 per segnalare che ha terminato l'inserimento. Il programma quindi calcola e stampa la sua media pesata sui crediti.

Nota: si tenga conto che la votazione del singolo esame e il numero di crediti sono interi.

Inoltre sono votazioni valide per il superamento di un esame solo quelle comprese tra 18 e 30 (estremi inclusi).

Esercizio 4 - mediaponderata2.c

Modificare l'esercizio precedente in modo il programma chieda inizialmente allo studente quanti esami ha superato, in modo da non dover usare lo 0 alla fine dell'inserimento dei voti e crediti.

Esercizio 5 - radicequadrata.c

Realizzare una funzione che calcoli la radice quadrata di un numero

- La radice quadrata di un numero x è compresa tra 0 e x
- Se $(x/2)^2 > x$ allora la radice è tra 0 e $x/2$, altrimenti è tra $x/2$ e x
- Si può proseguire prendendo il punto centrale tra $x/2$ e x o 0 ed $x/2$

Esercizio 6 - ciframassima.c

Scrivere un programma che chieda all'utente di inserire un numero intero compreso tra 0 e 10000 (estremi inclusi) e fornisca in output la cifra più alta contenuta nel numero letto. Se il numero non è compreso nell'intervallo specificato deve essere visualizzato un messaggio di errore.

Note:

- usare una costante per 10000