Politecnico di Milano
Informatica B, A.A. 2021/2022
Laboratorio 1

Luca Frittoli (luca.frittoli@polimi.it)
Mirko Salaris (mirko.salaris@polimi.it)

13 Ottobre 2021

1. **Riscaldamento** Si compili e si esegua il seguente programma, correggendolo (se necessario) seguendo le indicazioni del compilatore.

```
#include <stdio.h>
int main()
{
    int y = 0;
    int x;
    1=x;
    printf("La variabile x contiene: %d\n, x");
    printf("La variabile y contiene: %d\n", y);
    return 0
}
```

Soluzione

Il programma contiene tre errori:

- un assegnamento al contrario (`1=x` al posto di `x=1`)
- manca un punto e virgola dopo `return 0`
- la variabile `x` da inserire nella prima `printf` è scritta all'interno della stringa.

Il programma corretto è il seguente:

```
#include <stdio.h>
int main()
{
    int y = 0;
    int x;
    x=1;
    printf("La variabile x contiene: %d\n", x);
    printf("La variabile y contiene: %d\n", y);
    return 0;
}
```

2. **Acquisizione da tastiera** Si scriva un programma che prenda in ingresso due numeri interi e stampi a video la differenza tra il primo e il secondo.
Bonus: Si scriva il programma utilizzando una sola volta la funzione `scanf`.
Suggerimento: La `scanf` può essere utilizzata per leggere più valori contemporaneamente. Si ricordi che la stringa di formattazione specifica come ci si aspetta che l'utente inserisca i dati. Ad esempio, una stringa `"%d %d"` indica che l'input dovrà contenere un valore intero, uno spazio ed un altro valore intero. Una stringa `"%d,%d,%d"` indica che l'input dovrà contenere tre interi separati da virgola, e così via. Ovviamente le variabili di destinazione che seguono la stringa devono essere nel giusto ordine e numero, come per la `printf`.

Soluzione

(versione bonus)

```
#include <stdio.h>
int main()
{
    // Definiamo le variabili che conterranno i numeri inseriti
    int x, y, z;
    // Chiediamo all'utente di inserire i due numeri
    printf("Inserisci due numeri interi x,y:\n");// Leggiamo il numero inserito
    scanf("%d,%d", &x, &y);
    // Calcoliamo e stampiamo la differenza
    z = x - y;
    printf("%d - %d = %d", x, y, z);
    return 0;
}
```

3. **Orario lavorativo** Si scriva un programma che prenda in ingresso l'orario di ingresso e di uscita (in formato `hh:mm`) sul posto di lavoro di un dipendente, calcoli la durata della sua giornata lavorativa e la stampi in formato `hh:mm`. Si supponga che l'orario di uscita sia successivo a quello di entrata.

Soluzione

```
#include <stdio.h>
void main()
{
    // Chiediamo all'utente di inserire gli orari di ingresso e uscita
    int h1, m1, h2, m2;
    printf("Inserisci l'orario di ingresso hh:mm ");
    scanf("%d:%d", &h1, &m1);
    printf("Inserisci l'orario di uscita hh:mm ");
    scanf("%d:%d", &h2, &m2);
    // Per comodita' trasformiamo tutto in minuti
    int min1, min2;
    min1 = m1 + 60*h1;
    min2 = m2 + 60*h2;
    // calcoliamo la differenza in minuti
    int diff = min2 - min1;
    // stampiamo la differenza in formato hh:mm
```

```
    int hh, mm;
    hh = diff / 60;
    mm = diff % 60;
    printf("La giornata lavorativa e' durata: %d:%d\n", hh, mm);
}
```

4. **Registratore di cassa** Si scriva un programma che prenda in ingresso un prezzo in euro (intero) e restituisca il numero minimo di banconote utilizzando solo pezzi da 5, 10 e 50 euro. Indicare anche la moneta rimanente.
Suggerimenti:

   - Si ricordi che l'operatore modulo (resto della divisione intera) spesso permette di risolvere problemi molto velocemente senza l'utilizzo di costrutti condizionali o cicli.
   - Nel caso una variabile appaia sia nella parte sinistra che in quella destra di un'espressione (ad esempio `resto = resto % 10`), il programma calcolerà la parte destra usando il valore corrente della variabile e salverà il risultato nella variabile stessa (sovrascrivendo il valore precedente).

Soluzione

```
#include <stdio.h>
int main()
{
    // Definiamo le variabili che conterranno il prezzo inserito e il resto
    int prezzo, resto;
    // Chiediamo all'utente di inserire un prezzo e leggiamolo
    printf("Inserisci un prezzo:\n");
    scanf("%d", &prezzo);
    // Calcoliamo il numero di banconote da 50 euro necessarie
    int num_50 = prezzo / 50;
    // Calcoliamo il prezzo rimanente usando l'operatore modulo
    // (resto della divisione intera)
    resto = prezzo % 50;
    // Ripetiamo la procedura precedente per le banconote da 10 euro
    int num_10 = resto / 10; // Adesso il prezzo rimanente e'
                             // salvato nella variabile 'resto'
    resto = resto % 10;
    // Infine calcoliamo il numero di banconote da 5 euro per
    // coprire il prezzo rimanente
    int num_5 = resto / 5;
    resto = resto % 5; // Questa e' la moneta finale
    // Stampiamo il risultato
    printf("Il prezzo inserito e' %d euro\n", prezzo);
    printf("Ti servono %d banconote da 50 euro, %d da 10 euro e %d
    da 5 euro\n", num_50, num_10, num_5);
    printf("Rimangono %d euro da pagare in moneta\n", resto);
    return 0;
}
```

5. **Registratore di cassa – parte 2** Si modifichi il programma precedente in modo che funzioni correttamente anche quando il prezzo non è un numero intero.

Soluzione

```
#include <stdio.h>
int main()
{
    // Definiamo le variabili che conterranno il prezzo inserito e il resto
    float prezzo, resto;
    // Usiamo %f al posto di %d per leggere un valore float
    printf("Inserisci un prezzo:\n");
    scanf("%f", &prezzo);
    // questo calcolo rimane uguale: il risultato sara' intero
    int num_50 = prezzo / 50;
    // attenzione: non possiamo usare l'operatore % con i float
    resto = prezzo - num_50*50;
    // Ripetiamo la procedura precedente per le banconote da 10 euro
    int num_10 = resto / 10; // Adesso il prezzo rimanente e'
                             // salvato nella variabile 'resto'
    resto = resto - num_10*10;
    // Infine calcoliamo il numero di banconote da 5 euro per
    // coprire il prezzo rimanente
    int num_5 = resto / 5;
    resto = resto - num_5*5; // Questa e' la moneta finale
    // Stampiamo il risultato
    printf("Il prezzo inserito e' %d euro\n", prezzo);
    printf("Ti servono %d banconote da 50 euro, %d da 10 euro e %d
    da 5 euro\n", num_50, num_10, num_5);
    // usiamo %f per stampare il resto, che e' un float
    printf("Rimangono %f euro da pagare in moneta\n", resto);
    return 0;
}
```

6. **Equazioni di secondo grado** Si scriva un programma che chieda all'utente tre numeri a, b e c, che rappresentino i coefficienti dell'equazione: $ax^2 + bx + c = 0$, risolva l'equazione e ne stampi le soluzioni, supponendo che l'equazione sia effettivamente di secondo grado ($a \neq 0$) e non impossibile ($b^2 - 4ac \geq 0$). **Suggerimento:** per utilizzare la radice quadrata `sqrt` è necessario includere la libreria `<math.h>`.

Soluzione

```
#include <stdio.h>
#include <math.h>
void main()
{
    float a, b, c, x1, x2;
    printf("Inserisci i coefficienti dell'equazione ax^2+bx+c=0\n");
    printf("a: ");
    scanf("%f", &a);
```

```
    printf("b: ");
    scanf("%f", &b);
    printf("c: ");
    scanf("%f", &c);
    // calcoliamo il delta
    float delta = b * b - 4 * a * c;
    printf("delta = %.2f\n", delta);
    // calcoliamo le soluzioni
    x1 = (-b + sqrt(delta) / (2 * a);
    x2 = (-b - sqrt(delta) / (2 * a);
    printf("L'equazione e' soddisfatta per x = %.2f e x = %.2f\n", x1, x2);
}
```

7. **Tipi di dati** Si compili e si esegua il seguente programma, e si spieghi il motivo dei valori stampati.

```
#include <stdio.h>
int main()
{
int x = 1000;
char c = 'a';
float f = 10.25;
int a = f;
char b = x;

printf("La variabile x contiene %d\n", x);

printf("La variabile c contiene %c\n", c);
printf("La variabile c contiene %d\n", c);

printf("La variabile f contiene %f\n", f);
printf("La variabile f contiene %d\n", f);

printf("La variabile a contiene %d\n", a);
printf("La variabile b contiene %d\n", b);

return 0;
}
```

Soluzione

L'output del programma è:

```
La variabile x contiene 1000
La variabile c contiene a
La variabile c contiene 97
La variabile f contiene 10.250000
La variabile f contiene 323576480
La variabile a contiene 10
La variabile b contiene -24
```

- La prima `printf` stampa correttamente il valore intero contenuto in `x`.
- La seconda stampa correttamente il carattere contenuto in `c`.
- La terza stampa il codice ASCII del carattere contenuto in `c` (in questo caso il codice ASCII di 'a' è 97);
- La quarta stampa correttamente il valore `float` contenuto in `f`.
- La quinta prova a stampare il valore contenuto in `f` come un intero. Il valore stampato è ovviamente errato in quanto il sistema prova ad utilizzare una codifica intera per una variabile con codifica diversa (decimale a precisione singola in questo caso). Per questo motivo il compilatore dà un *warning*.
- La sesta stampa correttamente il valore di `f` troncato a un intero. La troncatura viene effettuata tramite l'assegnamento di `f` ad `a`, una variabile di tipo `int`.
- La settima prova a stampare il valore intero contenuto in `b`. Ovviamente ci si aspetterebbe che tale valore sia 1000 (ovvero il contenuto di `x`). Si ricordi però che la dimensione di una variabile `char` è 8 bit, ovvero i valori contenuti devono stare nell'intervallo $[-128, +127]$. In questo caso 1000 non appartiene a tale intervallo e, di conseguenza, il valore stampato è errato.

N.B.: valutare bene il tipo delle variabili da utilizzare in base alla natura dei valori che dovranno contenere. Un tipo errato potrebbe causare comportamenti indesiderati e difficili da identificare.

8. **Operazioni con i caratteri** Si compili e si esegua il seguente programma, e si spieghi il motivo dei valori stampati.

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

int main()
{
    char a = 'b';
    int b = 15;
    printf("output: %c\n", a+b);

    return 0;
}
```

Soluzione

L'output del programma è:

```
output: q
```

Interpretazione: la `printf` stampa il carattere corrispondente al codice ASCII ottenuto sommando il codice ASCII del carattere 'b' (contenuto nella variabile `a`) con il valore intero 15 (contenuto nella variabile `b`). Il codice ASCII di 'b' è 98, quindi il codice ASCII di `a+b` è 98+15=113, che corrisponde al carattere 'q' stampato. Di fatto, il programma stampa il 15-esimo carattere dopo 'b' secondo l'ordine alfabetico.

9. **Caratteri maiuscoli e minuscoli** Si scriva un programma che richieda all'utente un carattere minuscolo e stampi lo stesso carattere maiuscolo (supponendo che l'input sia effettivamente un carattere).
Suggerimento: Nel codice ASCII i caratteri sono ordinati seguendo l'alfabeto, con i caratteri maiuscoli prima di quelli minuscoli. Esiste quindi un offset fisso tra i codici ASCII di un carattere maiuscolo e il corrispondente carattere minuscolo. Questo offset è consultabile nella tabella ASCII (`http://www.asciitable.com`), o può essere calcolato facilmente.

Soluzione

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
    //dichiariamo una variabile per salvare l'input
    char c;
    // Calcoliamo l'offset tra due caratteri.
    int offset = 'A' - 'a';
    //chiediamo l'input all'utente (opzionale)
    printf("Inserisca un carattere minuscolo.");
    // Leggiamo l'input dalla linea di comando e salviamo
    // il valore nella variabile dichiarata
    scanf("%c",&c);
    //Stampiamo il risultato
    printf("Il carattere %c maiuscolo e %c minuscolo",c, c+offset);
    return 0;
}
```

10. **Ordine alfabetico** Si scriva un programma che richieda all'utente di inserire un carattere maiuscolo o minuscolo e lo stampi a video insieme alla sua posizione nell'alfabeto (A – 1, b – 2,...). Si supponga che l'input sia effettivamente un carattere.

Soluzione

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
    // dichiariamo una variabile per salvare l'input
    char c;
    // salviamo in una variabile la prima lettera 'A' e l'offset
    char a = 'A';
    int offset = 'A' - 'a';
    // chiediamo l'input all'utente
    printf("Inserisci un carattere: ");
    // Leggiamo l'input dalla linea di comando e salviamo
    // il valore nella variabile dichiarata
    scanf("%c",&c);
    // calcoliamo e stampiamo la posizione di c nell'alfabeto
    int pos = 1 + ((c-a) % offset);
```

```
        printf("%c -- %d\n", c, pos);
    }
```