Laboratorio di Elementi di Architetture e Sistemi Operativi

Esercizi del 4 Aprile 2012

Esercizio 1.

1. *Scrivere un programma C chiamato* `ciaomondo.c` *che esegua le seguenti operazioni:*
   - *scrive il messaggio* `Ciao Mondo!` *sullo standard output,*
   - *stampa il valore di una variabile intera,*
   - *stampa il valore di una variabile floating point,*
   - *stampa il contenuto di una variabile di tipo* `char`*.*

```
#include <stdio.h>

main()
{
    int n = 5;
    float d = 3.25;
    char c = 'A';

    printf("Ciao Mondo!\n");
    printf("Il valore di n è: %d\n", n);
    printf("Il valore di d è: %f\n", d);
    printf("Il valore di c è: %c\n", c);
}
```

2. *Utilizzare il compilatore gcc per produrre il file eseguibile corrispondente a* `ciaomondo.c`*.*

```
$ gcc ciaomondo.c
```

3. *Quale effetto ha il comando* `gcc` *invocato con argomento* `ciaomondo.c`*? Cosa viene prodotto?*
4. *Che cosa rappresenta il file* `a.out`*? Eseguitelo e verificatene gli effetti.*

   Il comando compila il programma C, ne fa il linking e produce un file `a.out` che rappresenta l'eseguibile in linguaggio macchina corrispondente al programma. La sua esecuzione produce i seguenti effetti:

```
$ ./a.out
Ciao Mondo!
Il valore di n è: 5
Il valore di d è: 3.250000
Il valore di c è: A
```

5. *Perfezioniamo l'invocazione del compilatore: utilizzare l'opzione* `-o` *in modo che venga generato un file eseguibile di nome* `ciaomondo`

```
$ gcc -o ciaomondo ciaomondo.c
```

6. *Eseguire il nuovo file prodotto dal compilatore. Verificare che i caratteri emessi dalla funzione* `printf` *sono realmente diretti sullo standard output provando a ridirigere alternativamente lo standard output e lo standard error su* `/dev/null`*.*

   L'esecuzione di `ciaomondo` provoca gli stessi effetti dell'esecuzione di `a.out`. La redirezione dello standard output fa si che il comando non visualizzi nulla a schermo, mentre la redirezione dello standard error non provoca effetti visibili:

```
$ ./ciaomondo > /dev/null
$ ./ciaomondo 2> /dev/null
Ciao Mondo!
Il valore di n è: 5
Il valore di d è: 3.250000
Il valore di c è: A
```
Questo mostra che l'output di `printf` viene stampato sullo standard output.

Esercizio 2. *Scrivere un programma C che stampi sullo schermo tutte le lettere maiuscole dell'alfabeto assieme al codice ASCII corrispondente, una lettera per riga.*

```
#include <stdio.h>

main()
{
    char c;
    for(c='a'; c <= 'z' ; c++) {
        printf("%c\t%d\n", c, c);
    }
}
```

Esercizio 3. *Scrivere un programma C per risolvere l'Esercizio 3 della lezione scorsa (stampa di un quadrato sullo schermo). A differenza dello script, il programma deve leggere dalla tastiera la dimensione del quadrato da stampare.*

```
#include <stdio.h>

main()
{
    int n, i, j;
    printf("Dimensione del quadrato: ");
    scanf("%d",&n);
    if(n < 2 || n > 15) {
        printf("Inserire una dimensione tra 2 e 15!\n");
        return;
    }
    for(i=1; i <= n; i++) {
        for(j=1; j <= n; j++) {
            if(i==1 || i==n) {
                if(j==1 || j==n) {
                    printf("+");
                } else {
                    printf("-");
                }
            } else {
                if(j==1 || j==n) {
                    printf("|");
                } else {
                    printf(" ");
                }
```

```
            }
        }
        printf("\n");
    }
}
```

Esercizio 4. *Scrivere un programma C chiamato* `calcolatrice.c` *che esegua delle semplici operazioni aritmetiche. Il programma deve:*

- *permettere all'utente di inserire l'operazione da eseguire da tastiera, specificandola nel formato* `num1 op num2`*.* `op` *può essere uno tra* `+`*,* `-`*,* `*` *e* `/`*, mentre* `num1` *e* `num2` *sono numeri interi (p.es.* `32 + 15`*,* `17 / 3`*);*
- *eseguire l'operazione e stampare il risultato.*

Provare l'esecuzione del programma su alcuni esempi. Cosa succede se l'utente sbaglia ad inserire l'operazione da eseguire?

```
#include <stdio.h>

main()
{
    int a, b;
    char op;

    printf("Inserire l'operazione da eseguire: ");
    scanf("%d %c %d", &a, &op, &b);
    switch(op) {
        case '+':
            printf("%d + %d = %d\n", a, b, a + b);
            break;
        case '-':
            printf("%d - %d = %d\n", a, b, a - b);
            break;
        case '*':
            printf("%d * %d = %d\n", a, b, a * b);
            break;
        case '/':
            printf("%d / %d = %d\n", a, b, a / b);
            break;
        default:
            printf("Operazione sconosciuta!\n");
            break;
    }
}
```

Nella maggior parte dei casi in cui l'utente sbaglia ad inserire l'operazione il programma produce il messaggio d'errore `Operazione sconosciuta!`. Ci sono tuttavia alcuni casi in cui ciò non avviene:

- quando l'utente sbaglia solo il secondo operando, ma inserisce correttamente il primo operando e l'operazione. In questo caso l'operazione viene eseguita comunque, con un valore casuale per il secondo operando:

```
$ ./calcolatrice
Inserire l'operazione da eseguire: 23 + g
23 + 32767 = 32790
```

- Quando si effettua una divisione per zero viene prodotto il messaggio d'errore `Floating point exception`.

Notare come l'utente non sia forzato ad inserire l'operazione separando numeri ed operazione con un solo spazio. Altri modi validi di inserire l'operazione sono:

- scritta tutta attaccata: `3+5`
- scritta con più spazi tra numeri ed operazione: `3  +  5`
- scritta su più righe:

```
3
+
5
```