Laboratorio di algoritmi e strutture dati*

Docente: Violetta Lonati

Primi programmi - esercizi da svolgere a casa

Nota: **siete invitati a svolgere gli esercizi, e in particolare quelli di comprensione dei codice, lavorando in coppia con un compagno o una compagna.** Nel sito del laboratorio di algoritmi potete scaricare tutto il codice sorgente che trovate in questa scheda.

Nota: ricordate le opzioni principali del comando `gcc` (per eventuali dubbi, consultate il manuale on-line `man gcc`):

- `-o` per salvare l'*output* in un file specificato,
- `-c` per arrestare la compilazione prima della fase di *link*,
- `-s` per arrestare la compilazione prima della fase di *assemble*,
- `-E` per arrestare la compilazione dopo la fase di *preprocessing*,
- `-W`, con parametro `all`, per la segnalazione di avvertimenti (warning),
- `-l`, con parametro `m`, per linkare le librerie matematiche.

Utilizzate un editor (ad esempio `gedit`) per visualizzare o creare i file sorgenti secondo quanto indicato dagli esercizi seguenti.

1 Conversione temperatura

Considerate il seguente programma `gradi.c`. Il programma legge una temperatura in gradi Farenheit e stampa l'equivalente in gradi centigradi.

```
1  #include <stdio.h>
2
3  #define FATTORE_SCALA ( 5.0f / 9.0f )
4  #define ZERO 32.0f
5
6  int main( void ){
7          float fahr, cels;
8          printf( "Inserisci la temperatura in gradi Fahrenheit: " );
9          scanf( "%f", &fahr );
10         cels = ( fahr - ZERO ) * FATTORE_SCALA;
11         printf( "Gradi Celisus equivalenti: %.1f\n", cels );
12         return 0;
13 }
```

*Last update: 3 ottobre 2019

Analizzate il codice sorgente e rispondete, possibilmente per iscritto, alle seguenti domande. Se avete dubbi, potete testare il programma eseguendolo su casi di input significativi e modificandolo.

1. A cosa serve l'istruzione alla riga 8?
2. A cosa serve l'istruzione alla riga 9?
3. A cosa serve la variabile `cels`? E' davvero necessaria? Provate a modificare il programma in modo da non usarla (e senza sostituirla con un'analoga variable con altro nome!).
4. Nella definizione di riga 3, a cosa servono le parentesi? Possono essere omesse?
5. Nella definizione di riga 3, è possibile sostituire `5.0` con `5`? E' possibile sostituire `9.0` con `9`? E' possibile sostituirli entrambi contemporaneamente? Giustificate la risposta.

2 Lettere

Considerate il seguente programma `lettere.c`.

```
1  #include <stdio.h>
2  #include <ctype.h>
3
4  int main( void ) {
5
6    char c;
7
8    printf( "Inserisci un carattere: " );
9    scanf( "%c", &c );
10
11   if ( !isalpha (c) )
12     printf( "%c non e` una lettera dell'alfabeto italiano\n", c );
13   else {
14     switch ( tolower( c ) ) {
15       case 'a': case 'e': case 'i': case 'o': case 'u':
16           printf( "%c e` una vocale dell'alfabeto italiano\n", c );
17           break;
18
19       case 'x': case 'y': case 'w': case 'j': case 'k':
20         printf( "%c non e` una lettera dell'alfabeto italiano\n", c );
21         break;
22
23       default:
24         printf( "%c e` una consonante dell'alfabeto italiano\n", c );
25         break;
26     }
27   }
28
29   return 0;
30 }
```

Analizzate il codice sorgente e rispondete, possibilmente per iscritto, alle seguenti domande. Se avete dubbi, potete testare il programma eseguendolo su casi di input significativi e modificandolo.

1. Cosa si aspetta in input il programma?
2. A cosa serve la direttiva nella riga 2?
3. Cosa stampa il programma se l'input è 'a'? Cosa se l'input è 'b'? Cosa se è 'y'
4. Cosa stampa se l'input è un numero compreso tra 0 e 9?
5. Cosa stampa se l'input è un segno di punteggiatura?
6. Descrivete cosa fa la funzione `isalpha` invocata nella riga 11.

7. Riassumete il comportamento del programma con una frase del tipo: “Il programma legge da standard input.... e stabilisce se.... stampando su standard output....”. La frase deve tenere conto dei vari casi che si possono verificare.

3 Classificazione triangoli

Considerate il seguente programma `triangoli-base.c`. Il programma riceve da standard input le lunghezze dei lati di un triangolo e, in caso affermativo, classifica il triangolo come equilatero, isoscele o scaleno.

```c
#include <stdio.h>

int main( void ) {
  float a, b, c;

  printf( "Inserisci tre numeri separati da spazi: " );
  scanf( "%f %f %f", &a, &b, &c );

  if ( a == b && a == c )
    printf( "Il triangolo e` equilatero\n" );

  else if ( a == b || a == c || b == c )
    printf ( "Il triangolo e` isoscele\n" );

  else
    printf ( "Il triangolo e` scaleno\n" );

  return 0;
}
```

Analizzate il programma e modificatelo affinché prima di procedere con la classificazione verifichi la questa proprietà fondamentale: in un triangolo, ogni lato è più corto della somma degli altri due.

4 Numeri

Considerate il seguente programma `numeri.c`.

```c
#include <stdio.h>

int main( void ) {
  int n, x = 0;
  printf( "Inserisci una serie di numeri: " );

  do {
    scanf( "%d", &n );
    x = x + n;
  } while ( n != 0 );

  printf( "%d\n", x );
  return 0;

}
```

Analizzate il codice sorgente e rispondete, possibilmente per iscritto, alle seguenti domande. Se avete dubbi, potete testare il programma eseguendolo su casi di input significativi e modificandolo.

1. Che input si aspetta il programma?
2. Cosa deve avvenire affinché il programma termini?
3. Cosa rappresenta la variabile `x`?
4. Come sono memorizzati i numeri letti da standard input? Quanto è lo spazio di memoria occupato dal programma?
5. Riassumete il comportamento del programma con una frase del tipo: "Il programma legge da standard input ..., calcola ... e stampa ...". La frase deve tenere conto dei vari casi che si possono verificare.

Infine, modificate il programma in modo che termini dopo aver letto esattamente 10 numeri diversi da 0. Può essere utile l'utilizzo delle istruzioni `continue` e/o `break`.

5 Media di numeri

Modificate il programma precedente in modo da calcolare la media dei numeri inseriti.

6 Massimo tra due numeri

Considerate il seguente programma `ternario.c`.

```c
#include <stdio.h>

int main( void ) {
  int a, b;

  printf( "Inserisci due numeri interi: " );
  scanf( "%d%d", &a, &b );
  printf( "Il ... tra %d e %d e' %d.\n", a, b, a > b ? a : b );

  return 0;
}
```

Il programma fa uso dell'operatore condizionale `expr1 ? expr2 : expr3`.

Cosa fa il programma? Modificate il messaggio di stampa in modo che sia più esplicativo.

7 Frase

Considerate il seguente programma `maiuscolo.c`.

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

int main( void ) {
  char c;

  while ( ( c = getchar() ) != '.' ) {
    if ( isalpha( c ) )
      putchar( toupper( c ) );
    else
      putchar( c );
  }

  printf ( "\n" );
  return 0;
}
```

Analizzate il codice sorgente e rispondete, possibilmente per iscritto, alle seguenti domande. Se avete dubbi, potete testare il programma eseguendolo su casi di input significativi e modificandolo.

1. Che input si aspetta il programma?
2. Cosa deve avvenire affinché il programma termini?
3. Cosa cambia se l'input viene inserito tutto insieme oppure carattere per carattere premendo "invio" dopo ogni carattere?
4. Per cosa è usato il costrutto `if` nella riga 8?
5. Come sono memorizzati i caratteri letti da standard input? Quanto è lo spazio di memoria occupato dal programma?
6. Ci sono limiti sul numero di caratteri che possono essere letti dal programma?
7. Riassumete il comportamento del programma con una frase del tipo: "Il programma legge da standard input ... e stampa ...". La frase deve tenere conto dei vari casi che si possono verificare.

8 Il cifrario di Cesare

Svetonio nella *Vita dei dodici Cesari* racconta che Giulio Cesare usava per le sue corrispondenze riservate un codice di sostituzione molto semplice, nel quale la lettera chiara veniva sostituita dalla lettera che la segue di tre posti nell'alfabeto: la lettera A è sostituita dalla D, la B dalla E e così via fino alle ultime lettere che sono cifrate con le prime.

Più in generale si dice codice di Cesare un codice nel quale la lettera del messaggio chiaro viene spostata di un numero fisso k di posti, non necessariamente tre.

Partendo dal programma dell'esercizio precedente `maiuscolo.c`, scrivere un programma che legga da input il numero k (chiave di cifratura) e il testo da cifrare, sotto forma di una sequenza di caratteri terminata da `.` e che emetta in output il testo cifrato; il programma deve cifrare solo le lettere dell'alfabeto, mantenendo minuscole le minuscole, e maiuscole le maiuscole, mentre deve lasciare inalterati gli altri simboli.

Si consiglia di usare `scanf(%d)` per leggere k e la funzione `getchar()` per leggere uno a uno i caratteri del testo da cifrare.

Notate che non c'è motivo per tenere in memoria tutti i caratteri del testo da cifrare, è sufficiente tenere in memoria l'ultimo letto!

Per provare se il programma funziona, cifrate un messaggio con una certa chiave k e poi applicate al risultato una nuova cifratura con chiave $26 - k$: il risultato dovrebbe essere la stringa originale.

Esempio di funzionamento

```
5 mamma li Turchi
rfrrf qn Yzwhmn
```

```
21 rfrrf qn Yzwhmn
mamma li Turchi
```