

# Laboratorio di algoritmi e strutture dati\*

Docente: Violetta Lonati

## Primi programmi - esercizi da svolgere in laboratorio

### 1 Cerchio

Scrivete un programma che legga (in una variabile di tipo `float`) il raggio di un cerchio e ne stampi l'area.

Tenete presente che il file `math.h` della libreria standard contiene la definizione della costante `M_PI`, pari al valore di  $\pi$  (il rapporto tra la circonferenza ed il diametro). Quando compilate, usate l'opzione `-lm`.

### 2 Equazione di secondo grado

Scrivete un programma che legga tre coefficienti  $a, b, c$  e calcoli le soluzioni (complesse) dell'equazione  $ax^2 + bx + c$ . Può essere utile includere il file di intestazione `math.h` della libreria standard, contenente la funzione `sqrt` per il calcolo della radice quadrata. Quando compilate, usate l'opzione `-lm`.

### 3 Ordine alfabetico

Scrivete un programma che legga due lettere maiuscole e stampi la loro distanza nell'ordine alfabetico. Ad esempio, su ingresso **A C**, il programma deve stampare 3, su ingresso **F B**, il programma deve stampare 5. Ricordate che i caratteri (`char`) in C sono rappresentati come (piccoli) interi e osservate che, secondo la codifica ASCII, non c'è soluzione di continuità e non ci sono altri caratteri tra la A e la Z (cosa di cui potete convincervi con il comando `man ascii`).

### 4 Quadrati perfetti

Scrivete un programma che stampi la sequenza crescente dei primi 10 quadrati perfetti (ossia numeri  $x$  tali che  $x = y^2$  per qualche numero naturale  $y$ ). Usate una macro di valore 10 per rendere il programma più facile da modificare!

### 5 Divisori e numeri primi

1. Scrivete un programma che stampi la sequenza decrescente dei numeri divisori di  $n$ , dove  $n$  è un numero inserito in input. Modificate il programma in modo che stampi anche il numero di divisori di  $n$ .

---

\*Ultima modifica 30 settembre 2020

2. Scrivete un altro programma che, usando un ciclo `for`, stabilisca se un numero  $n$  è primo (ovvero ha per divisori solo 1 e se stesso) oppure no. Cercate di ridurre il numero di istruzioni da eseguire! Scrivete una nuova versione del programma precedente usando un ciclo `while`.
3. Scrivete un programma che scomponga in fattori primi un numero dato in input.

## 6 Operatore `sizeof` e `limits.h`

L'operatore unario `sizeof` applicato ad una variabile ha per valore la dimensione della variabile a cui è applicato espressa in byte. Così, ad esempio (sull'architettura delle macchine presenti in laboratorio), se la variabile  $x$  è di tipo `int`, allora l'espressione `sizeof(x)` ha valore 4.

Il file di intestazione `limits.h` contiene (tra le altre cose) le definizioni (ottenute grazie alla direttiva `#define`) dei valori limite, per l'architettura corrente, dei tipi interi e carattere. Ad esempio, il seguente programma stampa l'intervallo di valori possibili per le variabili di tipo intero con segno.

---

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

int main( void ) {
    printf( "%d..%d\n", INT_MIN, INT_MAX );

    return 0;
}
```

---

Le definizioni dei limiti per alcuni degli altri tipi fondamentali si chiamano: `SCHAR_MIN`, `SCHAR_MAX` e `UCHAR_MAX` per il tipo carattere con e senza segno, `SHRT_MIN`, `SHRT_MAX` e `USHRT_MAX` per il tipo intero breve, con e senza segno.

Scrivete un programma che dichiari una variabile per ciascuno dei tipi fondamentali e delle sue rispettive varianti `long` e `short` (qualora ammissibili), e ne stampi la dimensione in byte ottenuta tramite l'operatore `sizeof` e l'intervallo di valori possibili per tali tipi.

## 7 Indovina il numero

Il gioco *Indovina il numero* funziona come segue: un giocatore  $A$  pensa a un numero intero  $x$  (con  $0 \leq x \leq 1000$ ), e l'altro giocatore,  $B$ , lo deve indovinare. Per farlo,  $B$  pone domande del tipo "Il numero è  $y$ ?", cui  $A$  può rispondere = (per indicare che il numero è stato indovinato), oppure < (per indicare che  $x$  è minore del numero  $y$ ), oppure > (per indicare che  $x$  è maggiore del numero  $y$ ).

Considerate il seguente programma `indovinaNumero.c`. Il programma agisce come il giocatore  $B$ , l'utente è il giocatore  $A$ .

---

```
1 #include <stdio.h>
2 #define MAX 1000
3
4 int main( void ) {
5     int min = 0, max = MAX, n;
6     char risposta;
7
8     for ( ; ; ) {
9         n = min + ( max - min ) / 2;
10        printf( "Il numero e' %d? ", n );
11        scanf( " %c", &risposta );
12        switch ( risposta ) {
13            case '<': max = n - 1; break;
```

```

14     case '>': min = n + 1; break;
15     case '=': return 0;
16 }
17 /* printf( "max = %d, min = %d\n", max, min ); */
18 }
19
20 return 0;
21 }

```

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. Quale strategia usa il programma per indovinare il numero? Provate a descriverla a parole.
2. Cosa cambierebbe nell'esecuzione se il programma venisse modificato sostituendo la riga 11 con la seguente?

```
scanf( "%c", &risposta );
```

3. Cosa cambierebbe invece se il programma venisse modificato sostituendo la riga 11 con la seguente?

```
scanf( "%c ", &risposta );
```

4. Date una spiegazione che giustifichi questi comportamenti.

NOTA: L'algoritmo implementato dal programma funziona con ogni sequenza ordinata, non solo numerica, ed è chiamato algoritmo di *ricerca dicotomica*.

## 8 Intervallo di tempo

Scrivete un programma che calcoli l'intervallo di tempo compreso tra due orari. Assumete che sia gli orari che l'intervallo di tempo siano rappresentati nel formato a 24 ore hh:mm:ss. E' possibile usare short int invece che int?

## 9 Conversione orario

Scrivete un programma che, dato un orario in formato a 24 ore hh:mm, fornisca il corrispondente orario nel formato AM/PM e viceversa. Ricordate che secondo il formato AM/PM, le 24 ore del giorno sono suddivise in due periodi chiamati AM (ante meridiem) e PM (post meridiem): ogni periodo consiste di 12 ore numerate con 12 (usato come 0), 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11. L'orario 12:00 AM indica la mezzanotte, l'orario 12:00 PM indica il mezzogiorno.

## 10 Calendario

Scrivete un programma che stampi un calendario mensile. L'utente deve specificare il nome del mese e il giorno della settimana in cui comincia il mese. Per semplicità considerate solo anni non bisestili...

### Esempio di funzionamento:

```

Inserisci il numero del mese (1 = gennaio, ..., 12 = dicembre): 2
Inserisci il giorno di inizio mese (1 = lunedì, ..., 7 = domenica): 4
 L  M  M  G  V  S  D
      1  2  3  4
 5  6  7  8  9 10 11
12 13 14 15 16 17 18
19 20 21 22 23 24 25
26 27 28

```

## 11 Disegni

Questo esercizio richiede di scrivere dei programmi che stampino delle "figure", secondo gli schemi sotto definiti. L'utente dovrà inserire un intero  $n$  che definisce la dimensione della figura da disegnare.

Il primo programma è fornito come esempio. Gli altri programmi possono essere costruiti modificandolo.

### 11.1 Righe alternate

Considerate il seguente programma disegno.c. Il programma produce delle figure in cui si alternano righe costituite solo da + e righe costituite solo da o, come nei due esempi qui sotto.

$n = 3$ + + + o o o + + +	$n = 5$ + + + + + o o o o o + + + + + o o o o o + + + + +
------------------------------------	--

```
1 #include <stdio.h>
2
3 int main( void ) {
4     int n, i, j;
5     char simbolo;
6
7     scanf( "%d", &n );
8
9     for ( i = 0; i < n; i++ ) {
10        simbolo = ( i % 2 ) ? 'o' : '+';
11        for ( j = 0; j < n; j++ )
12            printf( "%c ", simbolo );
13        printf( "\n" );
14    }
15
16    return 0;
17 }
```

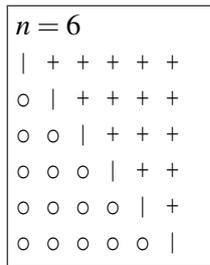
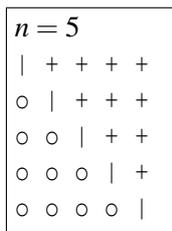
### 11.2 Caratteri alternati

In questo caso la figura è ottenuta alternando caratteri o e +:

$n = 5$ o + o + o + o + o + o + o + o + o + o + o + o + o	$n = 6$ o + o + o + + o + o + o o + o + o + + o + o + o o + o + o + + o + o + o
--	---

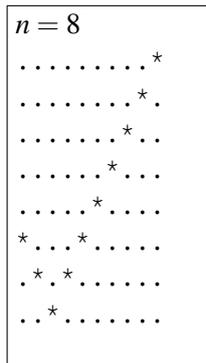
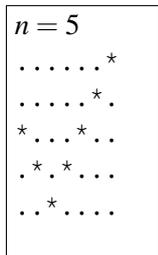
### 11.3 Triangolo

Il triangolo sotto la diagonale con direzione alto/sx verso basso/dx è formato da o, il triangolo sopra la diagonale da +, la diagonale da |.



### 11.4 Spunta

La figura riproduce il simbolo di spunta su uno sfondo di puntini (il ramo di sinistra è formato da 3 asterischi e parte dal bordo sinistro dello schermo, il ramo di destra è formato da  $n$  asterischi):



## 12 Espressioni ben parentesizzate

Una successione di caratteri è un'espressione ben parentesizzata se, per ogni prefisso della successione stessa (cioè, per ogni possibile segmento iniziale della successione), il numero di parentesi aperte "(" è maggiore o uguale al numero di parentesi chiuse ")", e se, complessivamente, il numero di parentesi aperte è uguale al numero di parentesi chiuse. Questo è ciò che avviene, per esempio, nelle espressioni aritmetiche ben formate.

Scrivete un programma che legga (mediante la funzione `getchar()` inclusa in `stdio.h`) una sequenza di caratteri terminata da `.` e determini se essa è un'espressione ben parentesizzata. In caso negativo, il programma dovrà stampare in quale posizione ha identificato un errore, e il tipo di errore.

### Esempio di funzionamento:

Stringa: <code>( (1) abb (3 (2a) 4 (b) ) 5) .</code> La stringa è un'espressione ben parentesizzata
Stringa: <code>( (1) abb (3 (2a) ) ) 4 (b5) .</code> La stringa non è un'espressione ben parentesizzata: Carattere 19: mancano parentesi chiuse alla fine!
Stringa: <code>( (1) abb (3) ) (2) ) a 4 (b) ) 5) .</code> La stringa non è un'espressione ben parentesizzata: Carattere 15: troppe parentesi chiuse!