

# 12BHD INFORMATICA – a.a. 2019/20

## Esercitazione di Laboratorio 1

---

### Obiettivi dell'esercitazione

- Disegnare flow-chart
- Prendere confidenza con l'ambiente di sviluppo, compilazione e debug.
- Eseguire programmi scritti in linguaggio C per acquisire da tastiera, manipolare e visualizzare a video valori numerici interi

### Contenuti tecnici

- Definizione della funzione *main* in un programma C
- Definizione di variabili intere (*int*) e loro utilizzo
- Uso di strutture elementari nei flow chart.

---

### Da risolvere preferibilmente in laboratorio

Esercizio 1. Utilizzando l'ambiente di sviluppo, scrivere, compilare ed eseguire il seguente programma in linguaggio C, verificando che non ci siano *errori* né *warning* in fase di compilazione

```
#include <stdio.h>
int main(void)
{
    int x , y, z;

    printf("Introduci un numero intero: ");
    scanf("%d", &x);
    y = 3;
    z = x/y;

    printf("%d/%d=%d\n", x, y, z);
    return 0 ;
}
```

Dopo averlo eseguito, esercitarsi con l'esecuzione passo a passo osservando il valore delle variabili x, y e z tramite i ‘watch’, provare con diversi valori: 0,9,15,20.

Esercizio 2. Disegnare il flow-chart per il calcolo del modulo (valore assoluto) di un numero; in particolare il programma dovrà:

- a) Acquisire da input un valore intero, positivo o negativo, e memorizzarlo in una variabile opportunamente definita.
- b) Stabilire utilizzando la struttura elementare *if-then(-else)* se tale variabile contiene un valore negativo e, in questo caso, trasformarlo nel corrispondente valore positivo
- c) Inviare in output il valore finale, ovvero il modulo del valore acquisito

Esercizio 3. Utilizzando l’ambiente di sviluppo, scrivere, compilare ed eseguire il seguente programma in linguaggio C, verificando che non ci siano *errori* né *warning* in fase di compilazione. Si disegni infine il relativo diagramma di flusso.

```
#include <stdio.h>

int main (void)
{
    int x;

    printf("Inserisci un numero: ");
    scanf("%d", &x);

    if (x>0)
        printf("Il valore %d e' positivo\n", x);
    else
        printf("Il valore %d e' negativo o pari a 0\n", x);

    return 0;
}
```

Dopo averlo eseguito, esercitarsi con l’esecuzione passo a passo osservando il valore della variabile x tramite ‘watch’, provare con diversi valori: 10, -10, 0, 9, -15.

#### Da risolvere a casa

Esercizio 4. Disegnare il flow-chart per stabilire se un numero è primo; in particolare, la soluzione dovrà:

- Acquisire da input un valore intero da tastiera
- Utilizzare le strutture elementari *while-do* o *do-while* per stabilire tramite un ciclo se il numero è primo o meno
- A seconda dei casi, inviare in output un opportuno messaggio.

Nota1) Sforzarsi di realizzare un algoritmo “ben strutturato”, senza salti: in questo caso l’utilizzo di una variabile logica (in gergo “flag”, bandierina) può aiutare.

Nota2) Adottare sempre il metodo dei raffinamenti successivi: ad esempio, nella stesura iniziale considero come divisori tutti i numeri che precedono il numero dato. Successivamente analizzo nel dettaglio il problema, per vedere se posso limitare il numero di cicli.

Esercizio 5. Scrivere un programma che definisca 3 variabili intere chiamate *operand1*, *operand2* e *result*, e:

- Acquisisca da tastiera il valore di *operand1* e *operand2* tramite la funzione *scanf*
- Ne calcoli la somma e la salvi nella variabile *result*

c) Visualizzi a video il valore della variabile *result* utilizzando la funzione *printf*  
Consiglio: prima di dare il run, eseguire sempre il programma passo-passo, controllando le variabili con il watch. In particolare, verificare sempre che i valori introdotti da tastiera e i valori letti nelle variabili coincidano, e così per i valori in output (test di correttezza per gli specificatori di formato e altro...). Inoltre verificare se ci sono casi in cui l’operazione di somma sia errata (overflow).

Esercizio 6. Scrivere un programma C in grado di risolvere un'equazione di primo grado espressa nella forma  $ax+b=0$ ; in particolare il programma dovrà:

- a) Definire due variabili intere (*int*),  $a$  e  $b$  per memorizzare i coefficienti dell'equazione
- b) Definire una variabile intera chiamata  $x$  in cui memorizzare il risultato dell'equazione
- c) Acquisire da tastiera il valore dei coefficienti  $a$  e  $b$
- d) Calcolare il valore di  $x$  e visualizzarlo a video

Approfondimento: cosa succede nel caso in cui il valore di  $a$  è uguale a 0?

Esercizio 7. <sup>1</sup> Realizzare un programma per la conversione da gradi Celsius a gradi Fahrenheit e viceversa. Il programma deve permettere di inserire una lettera 'C' o 'F' che indica la scala (Celsius o Fahrenheit) dei gradi; il valore dei gradi verrà inserito in seguito. Il programma deve calcolare e visualizzare i gradi Fahrenheit o Celsius rispettivamente. Se la relazione tra gradi Celsius e gradi Fahrenheit non è nota, cercarla in Internet.

---

<sup>1</sup> Questo esercizio sarà svolto in modo multimediale e inserito sul Portale, tra il materiale comune, nelle settimane successive.

# 12BHD INFORMATICA, A.A. 2019/2020

## Esercitazione di Laboratorio 2

---

### Obiettivi dell'esercitazione

- Definire variabili in linguaggio C
- Realizzare semplici calcoli basati sugli operatori aritmetici di base
- Utilizzare le opzioni di debug per il controllo delle variabili durante l'esecuzione di un programma
- Sperimentare l'uso dei costrutti condizionali per prendere decisioni per risolvere semplici problemi matematici

### Contenuti tecnici

- Assegnazione e debug di valori numerici tramite le funzioni di debug
- Uso di espressioni aritmetiche
- Uso preliminare dei costrutti condizionali *if* e *if-then-else*

---

### Da risolvere preferibilmente in laboratorio

Esercizio 1. Scrivere un programma che definisca:

- a. 1 variabile di tipo intero chiamata *int\_var* ed 1 variabile di tipo reale chiamata *float\_var*
- b. 1 variabile di tipo intero senza segno chiamata *uint\_var* ed 1 variabile di tipo reale rappresentata con doppia precisione chiamata *double\_var* ed inoltre,
- c. assegna i seguenti valori alle variabili definite:
  - *int\_var*  $\leftarrow -3$
  - *float\_var*  $\leftarrow 2.5$
  - *uint\_var*  $\leftarrow 50$
  - *double\_var*  $\leftarrow 2/3$

Visualizzare il contenuto delle variabili mediante l'uso della opzione di debug: Debug → Debugging windows → Watches

Esercizio 2. Scrivere un programma che definisca 3 variabili intere chiamate *var1*, *var2* e *tmp*. Il programma dovrà:

- a. Assegnare i seguenti valori alle variabili definite:
  - *var1*  $\leftarrow -3$
  - *var2*  $\leftarrow 12$

b. Utilizzando la variabile *tmp*, scambiare i valori di *var1* e *var2*.

Visualizzare il contenuto delle variabili mediante l'uso della opzione di debug: Debug → Debugging windows → Watches

Esercizio 3. Scrivere un programma che definisca 3 variabili reali (*float*) chiamate *length*, *width* e *perimeter*, corrispondenti a base, altezza e perimetro di un rettangolo:

- a. Inizializzi le variabili *length* e *width* usando dei valori scelti dal programmatore
- b. Partendo da questi dati, calcoli il perimetro del rettangolo e lo salvi nella variabile *perimeter*

Visualizzare il contenuto delle variabili mediante l'uso della opzione di debug: Debug → Debugging windows → Watches

---

#### Da risolvere a casa

- Esercizio 4. Scrivere un programma che definisca 2 variabili intere chiamate *var1*, *var2*. Il programma dovrà:
- a. Assegnare i seguenti valori alle variabili definite:
    - *var1* ← 25
    - *var2* ← -53
  - b. Senza utilizzare una variabile di appoggio, scambiare i valori delle variabili *var1* e *var2*.

Visualizzare il contenuto delle variabili mediante l'uso della opzione di debug: Debug → Debugging windows → Watches

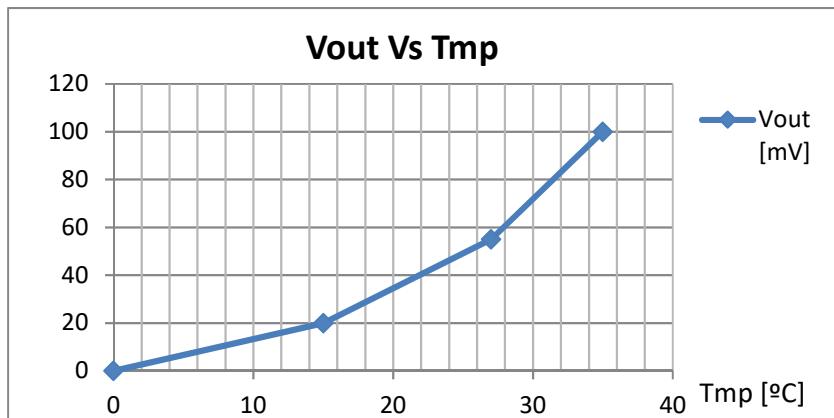
- Esercizio 5. Scrivere un programma che definisca 3 variabili reali (*float*) chiamate *price*, *tax* e *receipt*, e:
- a. Assegnare valori scelti dal programmatore per *price* e *tax*
  - b. Partendo da questi dati, calcoli il prezzo comprensivo delle tasse (*price+price\*tax/100*) e lo salvi nella variabile *receipt*

Visualizzare il contenuto delle variabili mediante l'uso della opzione di debug: Debug → Debugging windows → Watches

- Esercizio 6. Partendo dal flow-chart disegnato per l'esercizio 2 della settimana 1, si scriva un programma C per il calcolo del modulo (valore assoluto) di un numero; in particolare il programma dovrà:
- a. Acquisire da tastiera un valore intero, positivo o negativo, e memorizzarlo in una variabile opportunamente definita
  - b. Stabilire, utilizzando per la prima volta in modo sperimentale il costrutto condizionale *if*, se tale variabile contiene un valore negativo e, in questo caso, trasformarlo nel corrispondente valore positivo
  - c. Stampare a video il valore finale, ovvero il modulo del valore acquisito.

Esercizio 7. <sup>1</sup>Si vuole linearizzare il valore letto da un sensore di temperatura, che fornisce in uscita ( $V_{out}$ ) un valore tra 0 e 100 mV, mediante una spezzata. Sperimentalmente sono stati trovati i seguenti punti di ginocchio della spezzata:

Tmp [°C]	$V_{out}$ [mV]
0	0
15	20
27	55
35	100



Realizzare un programma che permetta di introdurre da tastiera un valore di tensione intero in mV fornito dal sensore, calcoli e stampi il valore di temperatura espresso in °C. Il programma deve usare in modo sperimentale il costrutto *if-then-else*.

---

<sup>1</sup> Questo esercizio sarà svolto in modo multimediale e inserito sul Portale, tra il materiale comune, nelle settimane successive.

# INFORMATICA, A.A. 2019/2020

## Esercitazione di Laboratorio 3

---

### Obiettivi

- Risolvere problemi gestendo input-output

### Contenuti tecnici

- Uso di *scanf* e *printf*
- Uso della direttiva *#define*
- Uso base di espressioni aritmetiche
- Uso operatori relazionali
- Uso operatori logici

---

### Da risolvere preferibilmente in laboratorio

Esercizio 1. Scrivere un programma che:

- a) Definisca 2 variabili di tipo intero: int\_1 e int\_2
- b) Definisca 2 variabili di tipo reale: float\_1 e float\_2
- c) Tramite la funzione *scanf* acquisisca da tastiera i valori per dette variabili
- d) Visualizzi su schermo usando la funzione *printf* il valore assunto dalle 4 variabili con il seguente formato:
  - i. *int\_1* e *int\_2* sulla stessa riga, occupando per ciascuno 5 colonne (incolonnamento per costruire tabelle),
  - ii. *float\_1* occupando almeno 5 spazi e con una precisione di 2 posizioni dopo il punto decimale,
  - iii. *float\_2* con una precisione di 3 posizioni dopo il punto decimale.

Esempio: valori acquisiti da tastiera 12 321 3.5 73.125

Variable	Value
int_1	12
int_2	321
float_1	3.50
float_2	73.125

- e) Si provi il programma con i seguenti valori: -3 e -3.5, 1000 e 1000.4567, 1 e 1.01

Approfondimento: si modifichi il programma in modo che acquisisca esclusivamente un valore reale da tastiera tramite la funzione *scanf*, e lo assegni a tutte le 4 variabili: verificare col watch delle variabili cosa succede.

Esercizio 2. Definire e assegnare dei valori iniziali alle variabili intere A, B e C (ad esempio: A=3, B=5 o A=7, B=7). Se eseguo la seguente istruzione:  
*C = (A==B)*

qual è il valore di C? Si ripeta l'esperimento con gli altri operatori relazionali:  
!=, <=, >=, al posto di ==

Approfondimento 1: si calcoli e visualizzi (usando la funzione *printf*) il valore di C per tutte le combinazioni di 0 e 1 come valore delle variabili A e B (A=0 B=0, A=0 B=1, A=1

B=0, A=1 B=1) nella seguente espressione

$$C = ( (A \&\& B) || (!B) ) \&\& (!A)$$

Approfondimento 2: si definiscano le variabili intere  $A$ ,  $B$ ,  $C$  e  $X$  e si attribuiscano loro dei valori opportuni per verificare se l'espressione logica

$$C = A < X < B$$

ha nel linguaggio C lo stesso significato che la relazione ha in matematica ( $X$  compreso tra  $A$  e  $B$ ). Qual è la forma corretta per esprimere in linguaggio C la relazione matematica?

Esercizio 3. Si scriva un programma per determinare la soluzione della seguente equazione:  
$$ax + bcx + dK = 0$$

In particolare:

- Si definisca una costante K tramite `#define`, e gli si assegni un valore a piacere, (es. `#define K 10`) (nota: con la define non si mette né l'=' né il ;)
- Si definiscano quattro variabili intere chiamate a, b, c, d corrispondenti ai parametri dell'equazione ed un'ulteriore variabile reale x
- Si acquisisca da tastiera il valore di a, b, c, e d
- Si calcoli il valore di x
- Si stampi il risultato a video.

---

#### Da risolvere a casa

Esercizio 4. Desidero acquistare un cellulare usato. La cifra massima che voglio spendere è:

- 100 euro come prezzo base.
- 40 euro aggiuntivi per ognuna delle caratteristiche a cui sono interessato
- 20 euro in meno per ogni mese in cui il cellulare è stato posseduto dal precedente proprietario.

Si realizzi un programma C che:

- Definisca tramite `#define` i valori che compongono il prezzo
- Definisca le variabili intere *price*, *features*, *months* e *years*
- Acquisisca da tastiera la quantità di caratteristiche possedute dal cellulare (*features*) e di anni per cui il cellulare è stato posseduto (*years*)
- Calcoli il numero di mesi per cui il cellulare è stato posseduto (*months*)
- Calcoli il prezzo massimo spendibile per il cellulare (*price*)
- Si stampi il risultato a video.

Esercizio 5. <sup>1</sup>Disegnare il flow-chart che classifichi un triangolo date le lunghezze dei suoi lati.

L'algoritmo deve implementare le seguenti funzionalità:

- Ricevere da tastiera 3 numeri interi corrispondenti alle lunghezze dei lati
- Stabilire se il triangolo è valido, degenere o non valido

---

<sup>1</sup> Questo esercizio sarà svolto in modo multimediale e inserito sul Portale, tra il materiale comune, nelle settimane successive.

c) In caso sia valido, stabilire se si tratta di un triangolo equilatero, isoscele o scaleno.

Suggerimento: un triangolo è valido se ogni lato è strettamente minore della somma degli altri due, è degenere se un lato è uguale alla somma degli altri due.

Approfondimento: stabilire se il triangolo è anche rettangolo, ovvero se rispetta il teorema di Pitagora ( $\text{Cateto}_1^2 + \text{Cateto}_2^2 = \text{Ipotenusa}^2$ ).

Esercizio 6. Si scriva un programma che acquisisca due numeri interi e ne calcoli la media.  
Il programma dovrà:

- Sommare i valori (positivi o negativi) acquisiti in una variabile somma opportunamente definita
- Calcolare la media aritmetica
- Visualizzare il risultato sullo schermo.

Si controlli il risultato per le seguenti copie: di valori (1,1) (0,8) (2,5) (-5,0) (-3,3).

Esercizio 7. Si scriva un programma che acquisisca quattro numeri interi positivi minori di 1000. Il programma dovrà:

- controllare che i valori siano contenuti nell'intervallo definito  $[0, 1000]$ . In caso contrario dovrà assegnare 0 al valore acquisito e indicare l'errore all'utente
- calcolare la massima differenza fra i valori acquisiti (in valore assoluto)
- stampare il valore della massima differenza come risultato.

Per esempio, se il programma riceve: 25, 115, 380, 213

Il programma dovrà stampare il valore: 355 che corrisponde alla differenza fra 380 e 25.

# **12BHD INFORMATICA, A.A. 2019/2020**

## Esercitazione di Laboratorio 4

---

### Obiettivi dell'esercitazione

- Risolvere problemi che implichino scelte logiche
- Sperimentare il concetto di iterazione

### Contenuti tecnici

- Uso dei costrutti *if-then-else* e *switch*
  - Uso dei costrutti iterativi *while*, *do-while* e *for*
  - Introduzione all'uso degli operatori *cast* e *sizeof*
- 

### Da risolvere preferibilmente in laboratorio

Esercizio 1. Si scriva un programma in linguaggio C in grado di determinare se l'equazione di secondo grado ( $ax^2 + bx + c = 0$ ) ha soluzioni reali.

In particolare:

- a. Si definiscano tre variabili chiamate *a*, *b* e *c*, corrispondenti ai parametri dell'equazione
- b. Si acquisisca da tastiera il valore di *a*, *b* e *c*
- c. Si calcoli il cosiddetto *discriminante* della formula risolutiva
  - i. In caso il delta sia positivo, visualizzare il seguente messaggio “L'equazione ha due soluzione REALI distinte”
  - ii. In caso il delta sia nullo, visualizzare il seguente messaggio “L'equazione ha due soluzione REALI coincidenti”
  - iii. Altrimenti stampare a video un messaggio per segnalare che l'equazione non ha soluzioni reali

Esercizio 2. Si scriva un programma in linguaggio C che, dato un numero intero tra 1 e 12 che rappresenta il mese corrente, utilizzi il costrutto *switch* per stampare il nome del mese per esteso (1→“Gennaio”, 2→“Febbraio”, 3→“Marzo”, ..., 12 → “Dicembre”).

Il programma gestisca anche le situazioni di inserimento di valori non compresi nell'intervallo 1-12.

Approfondimento: modificare il programma in modo che accetti come input una data nella forma gg/mese/anno (esempio: 23/3/2012) e stampi la stessa data con il mese per esteso (esempio: 23 marzo 2012). Si consiglia l'utilizzo del costrutto *switch*.

Esercizio 3. Si scriva un programma C che acquisisca numeri interi da tastiera finché non viene inserito il valore 0.

Suggerimento: si utilizzi il costrutto iterativo *while* oppure *do-while*.

Approfondimento: modificare il programma accumulando (ovvero continuando a sommare) in una variabile *i* valori inseriti prima dell'immissione del numero 0; al termine dell'acquisizione il programma stampi a video il valore calcolato.

### Da risolvere a casa

Esercizio 4. Scrivere un programma C che acquisisca in input da tastiera un valore intero positivo  $N \leq 40$  corrispondente alla base di un triangolo rettangolo e isoscele, e che riproduca a video tale triangolo utilizzando il carattere “\*”.

Esempio: se il valore letto da tastiera è 3, a video dovrà essere visualizzata la seguente serie di caratteri:

```
*  
**  
***
```

Approfondimento: Si scriva un programma in linguaggio C che letto un valore intero positivo dispari  $N$  disegni forme geometriche alternative, quali il triangolo isoscele, il quadrato... ecc.

Per esempio, provare a disegnare la seguente figura geometrica:

```
*****  
*****  
****  
***  
*
```

In questo caso il valore  $N = 9$ .

Esercizio 5. Scrivere un programma in linguaggio C che visualizzi i primi 20 numeri della serie di Fibonacci.

Suggerimento: ecco i primi numeri appartenenti alla serie 0 1 1 2 3 5 8 ... In modo formale la serie si costruisce considerando la seguente relazione:

$X_i = X_{i-1} + X_{i-2}$ , con  $X_0 = 0$  e  $X_1 = 1$ ;

Approfondimento: si modifichi la serie come segue:

$X_i = X_{i-1} * X_{i-2}$ , con  $X_0 = 1$  e  $X_1 = 2$ ; quanti sono gli elementi di questa serie rappresentabili con variabili di tipo intero?

Esercizio 6. <sup>1</sup>Scrivere un programma C che legga in input da tastiera un numero reale  $N$  e successivamente numeri interi finché entrambe le seguenti condizioni sono rispettate  
a. La media dei numeri acquisiti è superiore al valore di  $N$   
b. Sono stati acquisiti meno di 10 numeri.

Esercizio 7. Si scriva un programma C con lo scopo di calcolare il massimo valore positivo memorizzabile in variabili di tipo *int*, *long* e *unsigned int*.

Suggerimento: seguendo la traccia riportata nel seguito, eseguire le istruzioni del programma con il passo-passo del debug e osservare il risultato delle varie assegnazioni.

a) Verificare che non è una via percorribile provare ad assegnare valori via via più grandi: se ad esempio si scrive l'istruzione *value = 3000000000*, il compilatore non segnala errore, e al più segnala warning. Cosa si osserva in *value* con il watch, dopo l'esecuzione dell'istruzione?

b) Provare a ottenere questi valori in modo “empirico”, ovvero acquisendoli tramite la funzione *scanf* e ristampandoli opportunamente usando la *printf*. Verificare tramite il pro-

---

<sup>1</sup> Questo esercizio sarà svolto in modo multimediale e inserito sul Portale, tra il materiale comune, nelle settimane successive.

gramma che anche questa via non è percorribile: il comportamento della *scanf* in caso di errore nei dati non è dominabile da chi scrive il programma.

c) Realizzare a questo punto un algoritmo che, tenendo conto delle rappresentazioni binarie dei numeri senza segno e in complemento a 2, permetta di rilevare il valore max: per i numeri con segno, si può attribuire a *value* il valore iniziale di 0, poi si incrementa ripetutamente *value*. È noto che se si incrementa di 1 il valore massimo positivo, si ottiene overflow e il valore diventa negativo. Il valore cercato è dunque il precedente al primo valore negativo trovato. Tradurre l'algoritmo in programma e collaudarlo. Come si può modificare l'algoritmo (e il programma) perché operi con i numeri senza segno?

# 12BHD INFORMATICA, A.A. 2019/2020

## Esercitazione di Laboratorio 5

---

### Obiettivi dell'esercitazione

- Scrivere programmi in grado di memorizzare e elaborare molti valori

### Contenuti tecnici

- Consolidamento uso dei cicli
- Introduzione all'uso dei vettori

---

### Da risolvere preferibilmente in laboratorio

Esercizio 1. Scrivere un programma C che, acquisiti 2 numeri interi positivi ne calcoli il massimo comune divisore utilizzando la formula di Eulero.

Formula di Eulero o metodo dei resti: si procede per divisioni successive del numero maggiore per quello minore, sostituendo ad ogni passo il valore maggiore con il minore ed il minore col resto della divisione. Il processo termina quando il resto è 0.

Esempio: A = 34 , B = 18

passo 1:  $34 \% 18 = 16$

passo 2:  $18 \% 16 = 2$

passo 3:  $16 \% 2 = 0 \leftarrow$  stop!

Risultato: MCD = 2

Suggerimento: disegnare innanzitutto il flowchart del metodo tenendo conto che ogni passo corrisponde ad una iterazione ed in secondo luogo procedere alla stesura del codice.

Esercizio 2. Si scriva un programma C che definisca e manipoli un vettore composto di 10 elementi interi; il programma deve

- a. Acquisire valori da tastiera e memorizzarli all'interno del vettore
- b. Stampare il contenuto del vettore al termine dell'acquisizione
- c. Calcolare e stampare la media dei valori nel vettore utilizzando una variabile di tipo float
- d. Individuare e stampare a video il valore massimo e la sua posizione ordinale nel vettore.

Approfondimento: considerare il caso in cui il valore massimo occorre più di una volta, e stampare tutte le relative posizioni.

- Esercizio 3. Scrivere un programma C che definisca due vettori  $v1$  e  $v2$  di  $N$  elementi di tipo intero e memorizzi nei vettori valori “accettabili” acquisiti da tastiera secondo quanto segue:
- In  $v1$  siano memorizzati solo i valori positivi ed i valori negativi multipli di 3
  - In  $v2$  siano memorizzati solo i valori negativi non multipli di 3 e dispari
  - Tutti gli altri valori acquisiti siano ignorati
  - L'inserimento si conclude quando uno dei due vettori è pieno; a questo punto si stampi a video il contenuto dei vettori acquisiti.

---

Da risolvere a casa

- Esercizio 4. <sup>1</sup>Scrivere un programma C che acquisisca un massimo di  $N$  valori interi, con  $N$  costante definita a piacimento. L'acquisizione deve procedere finché la serie di numeri è monotona, ovvero costituita da numeri in ordine crescente o decrescente. Stampare il contenuto del vettore al termine dell'acquisizione

Esempi:

( $N=10$ )

1 4 6 10 4	← l'inserimento del valore 4 termina le iterazioni
9 7 6 7	← l'inserimento del valore 7 termina le iterazioni
1 2 3 4 5 6 7 8 9 10	← ho acquisito 10 numeri quindi mi fermo

Suggerimento: scrivere innanzitutto una versione semplificata scegliendo una singola direzione di monotonia (o crescente o decrescente), quindi passare alla soluzione completa.

- Esercizio 5. Scrivere un programma C che scandisca un vettore di  $N$  valori interi, e determini se esiste una serie crescente di tre numeri consecutivi. In caso positivo, il programma deve stampare la serie di numeri e la posizione del primo valore.

- Esercizio 6. Si scriva un programma C che analizzi il contenuto di un vettore di dati tutti positivi alla ricerca di valori duplicati. Il programma dovrà in particolare:

- Acquisire i valori del vettore da tastiera. L'acquisizione termini con l'introduzione un valore negativo, che non deve essere memorizzato.
- Entrare in un ciclo in cui chiede l'introduzione di un valore, e fornire in un secondo vettore le posizioni in cui viene trovato quel valore nel primo vettore, se presente. Il ciclo termini con l'introduzione di un valore negativo.

---

<sup>1</sup> Questo esercizio sarà svolto in modo multimediale e inserito sul Portale, tra il materiale comune, nelle settimane successive.

Nota: una tecnica simile si può adottare se si cerca la posizione dei massimi (o dei minimi) di un vettore, quando di massimi (o di minimi) ci può essere più di uno.

# 12BHD INFORMATICA, A.A. 2019/2020

## Esercitazione di Laboratorio 6

---

### Obiettivi dell'esercitazione

- Elaborare e manipolare il contenuto di un vettore precedentemente acquisito
- Scrivere programmi che includano semplici funzioni

### Contenuti tecnici

- Uso avanzato dei vettori
- Uso dei cicli annidati per l'analisi dei vettori
- Uso preliminare di funzioni di calcolo con parametri passati *by value*

---

### Da risolvere preferibilmente in laboratorio

Esercizio 1. Si scriva un programma C che:

- a. legga un vettore di N elementi interi (con N costante predefinita)
- b. determini se gli elementi di tale vettore costituiscono una successione palindroma.

Suggerimento: una successione si dice palindroma se e' identica letta da sinistra verso destra o da destra verso sinistra.

Esempio: le seguenti successioni di valori sono palindrome:

12 3 12  
1 4 5 4 1  
10 10 10

mentre la seguente non è palindroma:

1 3 4 3 2

Esercizio 2. Si scriva un programma C che:

- a. legga 2 vettori di N elementi interi (con N costante predefinita)
- b. stabilisca se i due vettori contengono gli stessi elementi, anche disposti in ordine differente

Esempio: siano dati i due vettori seguenti:

v1 → 15 3 12 13 29  
v2 → 15 29 13 3 12

questi contengono gli stessi valori, anche se in posizioni differenti.

Invece, i due vettori seguenti:

v1 → 11 3 12 18 29  
v2 → 12 29 13 4 12

non contengono gli stessi valori.

Approfondimento: considerare la possibilità che ci siano valori ripetuti tra quelli memorizzati nei vettori. Ad esempio

v1 → 12 3 12 13 29  
v2 → 12 29 13 3 12

contengono gli stessi valori ed il 12 compare 2 volte per vettore.

Invece, i due vettori seguenti:

v1 → 12 3 13 13 29  
v2 → 12 29 13 3 12

non contengono gli stessi valori.

Da risolvere a casa

---

Esercizio 3. Si scriva un programma C che analizzi il contenuto di un vettore alla ricerca di valori replicati. Il programma dovrà in particolare:

- a. Acquisire i valori del vettore da tastiera
- b. Scandire il vettore stabilendo se al suo interno esistono valori ripetuti 2 o più volte.
- c. Stampi l'elenco dei numeri ripetuti e il numero di occorrenze relative, verificando che ciascun numero compaia una volta sola in tale elenco.

Esercizio 4. Si scriva un programma C che legga da tastiera in un vettore di lunghezza N una sequenza di N numeri, li ordini in senso crescente man mano che vengono introdotti (ordinamento per inserimento, insertion sort) e alla fine stampi il contenuto del vettore.

Suggerimento: a prima vista, sembra che l'algoritmo da seguire sia:

- leggo un valore
  - cerco la sua posizione nel vettore, tenendo conto di quanti ho inserito
  - sposto tutti i successivi in avanti di una posizione (partendo dal basso)
  - inserisco il valore nella sua posizione ordinata
- Ad una più attenta analisi, ci si accorge che la parte che cerca la posizione in cui inserire il nuovo valore è ridondante. Si può procedere così;
- leggo un valore
  - analizzo il vettore a partire dal basso: se il valore che ho introdotto è minore del dato del vettore che sto considerando, sposto quest'ultimo nella cella successiva
  - itero il procedimento tornando all'indietro, finché non trovo un dato minore del valore introdotto (o sono arrivato in cima al vettore): il valore è da inserire nella cella successiva.

Esempio: ho già inserito nel vettore i dati 2, 5, 7, 9. Il valore letto sia 3. Confronto 3 con 9. 3 è minore, sposto in avanti di una posizione 9. Il vettore diventa 2, 5, 7, , 9. Confronto 3 con 7, è ancora minore, sposto il 7 in avanti di un posto. Il vettore diventa 2, 5, , 7, 9. Quando arrivo a confrontare 3 con 2, verifico che è maggiore, lo inserisco nella cella successiva (che era stata liberata al passo precedente). Formalizzare l'algoritmo e tradurlo in C.

Esercizio 5. Si scriva un programma C che legga da tastiera due numeri interi corrispondenti a base ed esponente, ed esegua il calcolo della potenza  $\text{base}^{\text{esponente}}$ . Il programma deve invocare una funzione chiamata *power* dal programma main, con il seguente prototipo:

*int power(int base, int exponent);*

Esempio: siano dati i seguenti valori

*base=3*

*exponent=2*

Il risultato di  $\text{base}^{\text{exponent}}$  sarà 9. In un altro caso con

*base=2*

*exponent=3*

Il risultato di base<sup>esponente</sup> sarà 8.

Suggerimento: all'interno della funzione, calcolare la potenza moltiplicando iterativamente la base per se stessa un numero di volte pari all'esponente.

Esercizio 6. (per i più curiosi, per i Talenti e per quelli che aspirano a diventarlo)

Un numero è palindromo se può essere letto da sinistra verso destra o da destra verso sinistra. Realizzare un programma che, per ogni numero intero da 2 fino ad un massimo stabilito come parametro, ad esempio 30, verifichi se esiste una base in cui quel numero è palindromo. Ad esempio, 5 è palindromo in base 2 ( $101_2$ ), 16 è palindromo in base 3 ( $121_3$ ), ecc. Per questo esercizio non ci si preoccupi di “rappresentare” le cifre, è sufficiente indicarne il valore (per intenderci, in base 16 la cifra di valore 10 non è richiesto che venga rappresentata come A, è sufficiente il valore 10). Inoltre si escludano i casi di 1 sola cifra (la risposta sarebbe banale): questo implica che per un generico numero N, si deve provare con base che va da 2 a N-1.

Suggerimento: dato il valore di un numero N e data una base b, per verificare se è palindromo (in quella base) bisogna convertire il numero in quella base, memorizzando le cifre che lo compongono, e poi verificare se è indifferente leggere le cifre da sinistra a destra o da destra a sinistra. Ma c'è un metodo più furbo, che non richiede la memorizzazione dell'intero numero.

Infatti: Dato il valore di un numero, l'algoritmo delle divisioni per una base permette di ricavare la sequenza di cifre che lo rappresentano (in quella base): le cifre sono calcolate a partire dalla meno significativa. Esempio: dato  $5_{10}$ , si ricavano per la base 3 le cifre 2, 1 (infatti  $5_{10} \rightarrow 12_3$ ) Data la sequenza di cifre, per ricavare il valore del numero si parte dalla cifra più significativa e si applica l'algoritmo del prodotto (per la base) e somma (della cifra corrente). Esempio:  $12_3 \rightarrow 1*3+2 = 5_{10}$ .

Pertanto per verificare se un numero è palindromo, si possono utilizzare i due algoritmi precedenti in cascata: se il valore del numero di partenza è uguale al valore del numero ricavato dalla combinazione dei due algoritmi precedenti, si è in presenza di un palindromo. Si osservi che, sebbene l'algoritmo non richieda la memorizzazione delle cifre in un vettore, può essere conveniente memorizzare se si vuole poi visualizzare.

Il calcolo del valore delle cifre invertite sia realizzato da una funzione.

Esercizio 7. Si realizzi un algoritmo in grado di tabulare il valore della funzione  $\arcsin(x)$  per x compreso nell'intervallo  $[a,b]$  con passo c. I valori di a, b, c siano forniti in input da tastiera. Non si utilizzi a tale scopo la funzione  $\arcsin()$  di libreria, ma si realizzi una funzione  $\text{invsin}(z1,z2,k,e)$  in grado di calcolare numericamente col metodo di bisezione la radice dell'equazione  $\sin(z)=k$  con z (restituito come valore di ritorno della funzione) compreso in  $[z1,z2]$  e con precisione pari a e, supponendo che la funzione  $\sin()$  sia monotona nell'intervallo  $[z1,z2]$ . Si descriva l'algoritmo realizzato

utilizzando il linguaggio di programmazione C. L'algoritmo realizzato deve essere testato sul calcolatore in modo da verificarne la correttezza sintattica e semantica.

- Esercizio 8. <sup>1</sup>Realizzare un programma che generi e stampi tutte le terne pitagoriche nell'intervallo degli interi (A, B e C formano una terna pitagorica se  $A^2 + B^2 = C^2$ ). E' richiesto che il test venga effettuato da una funzione che restituisca il valore TRUE se la terna passata come parametro e' pitagorica, FALSE altrimenti. Suggerimento: attenzione all'overflow della somma!

---

<sup>1</sup> Questo esercizio sarà svolto in modo multimediale e inserito sul Portale, tra il materiale comune, nelle settimane successive.

# 12BHD INFORMATICA, A.A. 2019/2020

## Esercitazione di Laboratorio 7

---

### Obiettivi dell'esercitazione

- Scrivere programmi che utilizzino caratteri e stringhe

### Contenuti tecnici

- Uso avanzato delle funzioni e dei vettori
- Uso del tipo *char*
- Uso delle funzionalità contenute in *ctype.h* e *math.h*

---

### Da risolvere preferibilmente in laboratorio

Esercizio 1. Si scriva un programma C che, dati due vettori di uguale dimensione N (*vbase* e *vexponent*), elevi ciascun elemento del vettore *vbase* alla potenza indicata nell'elemento di *vexponent* avente lo stesso indice (ossia *vbase[i]* elevato a *vexponent[i]*). I risultati dovranno essere memorizzati nella corrispondente posizione di un terzo vettore denominato *vres*. Si utilizzi la funzione *power* definita nel corso del precedente laboratorio e avente il seguente prototipo:

```
int power(int base, int exponent);
```

Vengano inseriti prima i valori delle N basi e poi quelli degli N esponenti; vengano alla fine visualizzati i valori di *vres*.

#### Esempio

Siano inseriti dall'utente i valori seguenti (per N pari a 5):

vbase	→	5 2 7 4 9
vexponent	→	2 6 1 8 3

Il vettore risultato sarà il seguente:

vres	→	25 64 7 65536 729
------	---	-------------------

Suggerimento: richiamare tante volte la funzione *power* quanti sono gli elementi dei vettori e ogni volta salvare il contenuto in una posizione opportuna di *vres*.

Esercizio 2. Si scriva un programma C che

- a. nel **main** chieda all'utente di inserire N valori e li metta in un vettore *vett*, quindi chieda un ulteriore valore *x*
- b. **passi** sia il vettore sia *x* ad **una funzione** che moltiplicherà ciascuno degli elementi del vettore per *x* e il cui prototipo sia  
*void mult(int v[], int n, int x);*

Il **main** poi visualizzi il vettore dopo la moltiplicazione.

Nota bene: la funzione riceve il vettore per riferimento e quindi può modificare i valori stessi del vettore.

Esercizio 3. Si scriva un programma C che acquisisca caratteri da tastiera fino alla ricezione di un “a capo”. Dopo tale evento il programma deve fornire all'utente le seguenti statistiche:

- a. il numero di caratteri introdotti;
- b. il numero di caratteri alfabetici;
- c. il numero di caratteri maiuscoli;
- d. il numero di cifre decimali;

- e. il numero di caratteri di spaziatura;
- f. Il numero di parole digitate, dove per parola si intende una sequenza di caratteri alfabetici contigui (“ciao 123 mondo !” dà 2 parole).

Suggerimento: Si utilizzino le funzioni della libreria standard dichiarate nell’header file `<ctype.h>` e si utilizzi una singola variabile di tipo carattere per l’acquisizione.

---

#### Da risolvere a casa

- Esercizio 4. Si scriva un programma in grado di manipolare gli elementi di un vettore di interi. Tale programma, dopo aver acquisito il contenuto del vettore, invoca due funzioni:
- a. `avgVect`: calcola la media degli elementi del vettore, restituendo tale valore alla funzione chiamante;
  - b. `upperLimit`: conta il numero di elementi che hanno valore superiore ad un certo limite, restituendolo alla funzione chiamante.

Il programma deve infine visualizzare la media dei valori del vettore e il numero di elementi che superano la media.

Suggerimento: per la funzione `mediaVett` il prototipo sarà:

```
float avgVect (int v[], int n);
```

mentre per la funzione `superanoLimite` il prototipo sarà:

```
int upperLimit (int v[], int n, float limit);
```

Approfondimento: si condensino le due funzioni descritte in un’unica funzione che restituisca il valore medio e che memorizzi nella variabile corrispondente al parametro `superiori` del prototipo il numero di elementi di valore superiore alla media:

```
float over_Avg (int v[], int n, int *superiori);
```

- Esercizio 5. Si scriva un programma C che:

- a. definisca due variabili di tipo carattere;
- b. ne acquisisca il contenuto da tastiera;
- c. stabilisca se i caratteri sono entrambi alfabetici:
  - i. in caso positivo, controlli se sono uguali e, se non lo sono, stampi i due caratteri in ordine alfabetico;
  - ii. in caso negativo, specifichi tramite messaggio se almeno uno dei caratteri è una cifra.

- Esercizio 6. Si realizzi un programma che permetta di inserire da tastiera un testo e che lo stampi su video, cambiando in maiuscolo ogni carattere di inizio parola.

Ad esempio se in ingresso viene fornito il seguente testo:

```
fatti non foste  
per viver come bruti  
ma per seguir virtute e canoscenza
```

su video deve apparire così:

```
Fatti Non Foste  
Per Viver Come Brutti  
Ma Per Seguir Virtute E Canoscenza
```

Suggerimento: utilizzare il metodo dei flag per stabilire se si è dentro una parola o fuori dalla parola

- Esercizio 7. <sup>1</sup>Si realizzi un programma in linguaggio C che analizzi un documento di tipo testo costituito da un numero impreciso di righe. Il documento viene caricato in input tramite la tastiera. Il programma deve effettuare le seguenti operazioni:
- contare il numero totale di righe;
  - contare il numero totale di parole;
  - determinare la lunghezza media di una parola.

Ai fini di questo problema, si considera come parola una sequenza di caratteri appartenenti tutti alla medesima riga e privi di spazi.

Ad esempio se in ingresso viene fornito il seguente documento:

```
fatti non foste  
per viver come bruti  
ma per seguir virtute e canoscenza
```

Il programma deve generare un output simile al seguente:

```
Numero di righe: 3  
Numero di parole: 13  
Lunghezza media di una parola: 4.5
```

Suggerimento: modificare il programma precedente (se è stato ben fatto, occorreranno pochissimi interventi).

---

<sup>1</sup> Questo esercizio sarà svolto in modo multimediale e inserito sul Portale, tra il materiale comune, nelle settimane successive.

# **12BHD INFORMATICA, A.A. 2019/2020**

## Esercitazione di Laboratorio 8

---

### Obiettivi dell'esercitazione

- Scrivere programmi che leggano e manipolino caratteri, stringhe e matrici

### Contenuti tecnici

- Uso dello specificatore di formato `%s`
- Uso delle funzionalità contenute in `string.h` e `ctype.h`
- Uso delle funzioni per la formattazione delle stringhe
- Uso delle matrici

---

### Da risolvere preferibilmente in laboratorio

Esercizio 1. Si scriva un programma che:

- a. Definisca un vettore di caratteri e acquisisca una stringa al suo interno
- b. Analizzi tale stringa rispondendo alle seguenti domande
  - i. Quanto è lunga la stringa?
  - ii. Quanti caratteri sono alfabetici e quanti numerici?

Approfondimento: acquisita una seconda stringa, stabilire se quest'ultima è inclusa nella prima (ad esempio: “importante” include “porta”)

Esercizio 2. Si scriva un programma C che:

- a. Acquisisca una stringa di massimo N caratteri (con N valore costante)
- b. Ne manipoli il contenuto
  - i. Trasformando tutte le lettere minuscole in maiuscole
  - ii. Rimpiazzando tutti i caratteri non alfanumerici con il carattere ‘\_’
  - iii. Sostituendo i caratteri numerici con il carattere ‘\*’
- c. Scandisca la stringa manipolata per contare quante parole sono presenti al suo interno, considerando una o più occorrenze del carattere ‘\_’ come separatore tra parole.

Approfondimento: l'ordine in cui vengono eseguite le manipolazioni influenza il risultato? Verificare la risposta scrivendo due versioni del programma che manipolino la stringa in modi differenti.

Esercizio 3. Si scriva un programma che acquisisca 2 stringhe corrispondenti a 2 orari nel formato `hh:mm`. Il programma deve:

- a. Controllare le stringhe, segnalando i casi in cui il formato non sia rispettato (ad esempio 10,30 non è valido)
- b. Stabilire se l'orario contenuto nella prima stringa è precedente a quello contenuto nella seconda stringa
- c. In caso affermativo, tradurre i 2 orari in valori interi corrispondenti all'orario espresso come distanza in minuti da 00:00 e calcolarne la differenza
- d. Convertire il risultato (sarà un numero intero positivo) in una stringa così composta “<intervallo calcolato>\_minuti” e la stampi a video.

---

Da risolvere a casa

- Esercizio 4. Si scriva un programma che acquisisca utilizzando la funzione *gets* una stringa composta da un massimo di 5 parole separate da spazi, per un totale di massimo 60 caratteri. Il programma deve
- Stabilire quante sono le parole contenute effettivamente nella stringa
  - Calcolare la media della lunghezza delle parole
  - Produrre una statistica sulla lunghezza delle parole

Esempio:

Se la stringa inserita è “questa stringa contiene cinque parole”  
allora visualizzerà a video

- La stringa contiene 5 parole
- La lunghezza media delle parole è 6,6 caratteri
- La stringa contiene
  - o 3 parole da 6 caratteri
  - o 1 parola da 7 caratteri
  - o 1 parola da 8 caratteri

Invece, se la stringa inserita è “questa stringa è corta”  
allora visualizzerà a video

- La stringa contiene 4 parole
- La lunghezza media delle parole è 4,75 caratteri
- La stringa contiene
  - o 1 parola da 1 carattere
  - o 1 parola da 5 caratteri
  - o 1 parola da 6 caratteri
  - o 1 parola da 7 caratteri

- Esercizio 5. Si scriva un programma che acquisisca 3 stringhe, ciascuna contenente il nome di un prodotto e il relativo prezzo separati da spazio. Una volta memorizzate queste informazioni in opportune variabili, il programma riceve un’ulteriore stringa contente un nome di prodotto e un valore intero corrispondente a una quantità.

Il programma deve:

- Stabilire se il prodotto inserito per ultimo corrisponde ad uno dei prodotti memorizzati in precedenza
- In caso negativo, richiedere un altro inserimento di nome prodotto e quantità
- In caso positivo, calcolare e visualizzare il costo totale moltiplicando quantità e prezzo del prodotto in questione

- Esercizio 6. <sup>1</sup>Si scriva un programma C che acquisisca sequenze di caratteri da tastiera conclusa da un ritorno a capo. Il programma deve continuare ad acquisire

---

<sup>1</sup> Questo esercizio sarà svolto in modo multimediale e inserito sul Portale, tra il materiale comune, nelle settimane successive.

sequenze fino alla ricezione di un EOF (*Ctrl-z*). Il programma deve quindi stampare le sequenze caratteri acquisiti

- a. Sostituendo ad ogni sequenza "ch" il carattere 'k'
- b. Sostituendo le doppie con una sola ripetizione del carattere.

Esempio: rischio → riskio

cammello → camelio

Esercizio 7. Si scriva un programma C che:

- legga da tastiera (per righe o per colonne, a scelta) una matrice quadrata di dimensione uguale a 5 righe e 5 colonne
- rintracci se tale matrice contiene delle sequenze di elementi adiacenti uguali a zero di lunghezza uguale o maggiore di 3
- visualizzi l'indice di riga in cui tali sequenze si presentano.

Esempio.

Sia la matrice la seguente:

```
0 0 0 4 5  
1 2 0 4 5  
1 0 0 4 0  
1 2 3 4 5  
1 0 0 0 0
```

La sequenza di valori "0 0 0" compare nella prima e nell'ultima riga e quindi occorre riportare una indicazione del tipo:

La sequenza compare nella riga 1

La sequenza compare nella riga 5

Si osservi che la riga 3 non contiene la sequenza indicata in quanto i tre zeri non si trovano in posizioni contigue.

#### FACOLTATIVO

Si effettui lo stesso controllo anche lungo le colonne.

Nell'esempio precedente occorre visualizzare, oltre ai messaggi già indicati, anche il seguente:

La sequenza compare nella colonna 3

#### FACOLTATIVO

Si generalizzi l'esercizio precedente per gestire sequenze di lunghezza variabile e contenenti un valore variabile.

Ovvero si leggano da tastiera due valori che specificano la lunghezza e il valore contenuto nelle sequenze da ricercare.

Nel caso sia:

Lunghezza della sequenza = 3

Valore nella sequenza = 0

occorre risolvere l'esercizio originale.  
Se invece si introduce

Lunghezza della sequenza = 4  
Valore nella sequenza = 1

occorre ricercare delle sequenze di 4 valori adiacenti uguali a 1,  
ovvero "1 1 1 1".

# 12BHD INFORMATICA, A.A. 2019/2020

## Esercitazione di Laboratorio 9

---

### Obiettivi dell'esercitazione

- Realizzare programmi con dati complessi, uso dei file

### Contenuti tecnici

- Uso di matrici di interi e caratteri
- Uso di insiemi di vettori “paralleli”
- Lettura di file di tipo testo

---

### Da risolvere preferibilmente in laboratorio

Esercizio 1. Una matrice di caratteri rappresenta in forma schematizzata una palude. La palude è costituita da zone di fango, rappresentate dal carattere ‘.’, e da zone pietrose, indicate dal carattere ‘\*’. Le dimensioni della matrice possono essere fissate a piacere, mediante dei `#define`, comunque non superiori a 25 righe e 80 colonne.

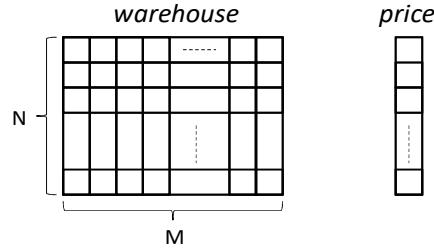
Esempio di palude:

```
**.*.*....*  
..*.*....**.  
*.....*....  
.**.*.*.*.  
..*.*....*.*
```

Realizzare un programma che cerchi nella palude un percorso da sinistra a destra, senza salti, costituito tutto da zone pietrose adiacenti. Si ipotizzi che, adiacente a destra di un punto pietroso, ci possa essere al più un altro punto pietroso (non ci sono diramazioni), sulla stessa riga, sulla riga in alto o sulla riga in basso. Il programma deve visualizzare la sequenza righe in cui ci sono le pietre del percorso trovato (le colonne sono implicite, ci deve essere una pietra per ogni colonna), oppure avvertire che non esiste un percorso. Per la prima versione del programma si utilizzi una matrice di stringhe predefinita. Come approfondimento, la palude viene introdotta da tastiera, e nella versione finale, si legge la palude da file.

Esercizio 2. Si scriva un programma in grado di gestire un listino prezzi, ovvero un programma con cui sia possibile gestire un elenco di prodotti e i loro relativi prezzi in €. Il programma utilizza una matrice di caratteri chiamata *warehouse* di dimensione NxM per memorizzare i nomi dei prodotti (massimo N prodotti) e un vettore di numeri decimali chiamato *price* di dimensione N usato per memorizzare i prezzi dei prodotti (il prezzo presente nell'*i*-esima cella di *price* corrisponde al prezzo del prodotto il cui

nome è memorizzato nell' $i$ -esima riga di *warehouse*). Di seguito la visualizzazione logica dei dati definiti.



Nel vettore *price*, il valore -2 indica linea libera (ossia nessun prodotto nella riga corrispondente in *warehouse*), mentre un valore positivo indica che la linea corrispondente in *warehouse* contiene un prodotto valido a cui è stato associato un prezzo. All'avvio del programma, il vettore *price* è totalmente inizializzato a -2 (ossia il listino è vuoto/non contiene prodotti).

Il programma propone all'utente un elenco di operazioni possibili sottoforma di menu. L'utente decide quale operazione eseguire selezionando il numero associato all'operazione di suo interesse.

Le operazioni possibili sono:

- 1) Inserimento di un nuovo prodotto e relativo prezzo
- 2) Stampa listino attuale (elenco dei prodotti con i relativi prezzi)
- 3) Uscita dal programma

Le prime due operazioni devono essere realizzate tramite l'invocazione delle seguenti due funzioni di cui si forniscono prototipo e funzionalità:

- *insert\_product*: è una funzione che permette di inserire un nuovo prodotto e il suo prezzo nel listino (il nome del prodotto deve essere di max M-1 caratteri). La funzione restituisce il valore 1 se il prodotto non era ancora presente nel listino e il suo inserimento nel listino è avvenuto con successo, 0 se già presente (ossia prodotto già inserito in precedenza), 2 se il listino è pieno (e quindi non risulta possibile inserire il nuovo prodotto). Se il prodotto non era ancora presente nel listino la funzione deve inserire il suo nome nella prima cella libera presente in *warehouse* e il prezzo a lui associato nella cella corrispondente di *price*.

```
int insert_product(char warehouse[][], float price[], int n, char new_product[],
                  float price_new_product);
```

Il parametro  $n$  corrisponde al numero di righe della matrice *warehouse*, che corrisponde anche al numero di celle del vettore *price* (in fase di invocazione nel nostro caso passeremo il valore  $N$ ).

- *print\_all*: è una funzione che permette di visualizzare a video il contenuto del listino (elenco prodotti e relativi prezzi). Inoltre, la funzione restituisce due valori (tramite due parametri passati per indirizzo): il prezzo medio ed il prezzo massimo dei prodotti presenti nel listino. Visualizzare a video i due valori restituiti.

```
void print_all(char warehouse[][M], float price[], int n, float *avg, float *max);
```

Avvertenza: l'esercizio suggerisce di utilizzare il vettore *price* contemporaneamente in due modi: per contenere il prezzo della merce, come flag di posizione vuota. Tenerne conto quando si cerca il nome di un prodotto, quando si effettua la stampa, etc.

#### Da risolvere a casa

- Esercizio 3. Si scriva un programma che legga da un file, il cui nome è introdotto da tastiera, alcune informazioni ferroviarie. Per ciascuna linea, il file contiene le seguenti informazioni (ciascuno dei campi non superi i 20 caratteri di lunghezza e sia privo di spazi)

```
<stazione_partenza> <ora_partenza> <stazione_arrivo> <ora_arrivo>
```

Il programma riceve poi da tastiera il nome di una città: il programma calcoli e stampi il numero di treni in arrivo ed il numero di treni in partenza da tale città (se inclusa nell'elenco).

- Esercizio 4. Estendere il programma realizzato come Esercizio 2 aggiungendo due ulteriori funzioni:
- 4) Aggiornamento prezzo prodotto (*update\_product*)
  - 5) Rimozione prodotto (*remove\_product*)

Le due operazioni sono realizzate tramite l'invocazione delle seguenti funzioni:

- a. *update\_product*: è una funzione che permette di aggiornare il prezzo di uno specifico prodotto. La funzione riceve il nome del prodotto da aggiornare e il suo nuovo prezzo. La funzione restituisce 1 se l'aggiornamento è avvenuto con successo, 0 se il prodotto non esiste nel listino.

```
int update_product(char warehouse[][M], float price[], int n, char product[], int new_price);
```

- b. *remove\_product*: è una funzione che permette di rimuovere un prodotto dal listino; restituisce 1 se la rimozione è avvenuta con

successo, 0 se il prodotto non esiste. La funzione deve impostare il valore -2 nel vettore *price* in corrispondenza dell'elemento rimosso.

```
int remove_product(char warehouse[]/[M], float price[], int n, char old_product[]);
```

Esercizio 5. Si realizzi un programma in grado di simulare il movimento di un topolino.

Si dispone di una matrice Q, di dimensione NxN, che descrive le caratteristiche di una certa area: in ciascuna locazione Q(x,y) della matrice vi e' la quota del quadratino di superficie posto alle coordinate x,y. A partire da una posizione iniziale del topolino, x0,y0, si stampino le coordinate di tutti i quadratini toccati dal topolino se esso segue le seguenti regole di movimento:

- Il topolino si sposta ad ogni passo di un solo quadratino, nelle 8 direzioni possibili
- Il topolino sceglie il quadratino su cui muoversi determinando il quadratino di quota massima tra gli 8 adiacenti
- Se tutti i quadratini adiacenti sono ad una quota inferiore o uguale rispetto al quadratino attuale, il topolino si ferma.

Esempio:

Supponendo di avere la seguente matrice Q di dimensione 10x10, e di aver scelto come punto iniziale il punto (3,7), allora il topolino effettuerà i seguenti spostamenti: (3,7) (4,8) (5,8) (6,8) (7,9). Il punto un cui il topolino si ferma è quindi il punto (7,9).

	<b>1</b>	<b>2</b>	<b>3</b>	<b>4</b>	<b>5</b>	<b>6</b>	<b>7</b>	<b>8</b>	<b>9</b>	<b>10</b>
<b>1</b>	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
<b>2</b>	0.1	0.2	0.2	0.2	0.2	0.3	0.3	0.3	0.3	0.1
<b>3</b>	0.1	0.2	0.3	0.3	0.4	0.5	0.5	0.5	0.5	0.1
<b>4</b>	0.1	0.2	0.3	0.3	0.5	0.5	0.5	0.7	0.7	0.1
<b>5</b>	0.1	0.2	0.4	0.4	0.5	0.7	0.7	0.8	0.7	0.1
<b>6</b>	0.1	0.2	0.4	0.4	0.5	0.7	0.8	0.9	0.8	0.1
<b>7</b>	0.1	0.2	0.4	0.4	0.5	0.7	0.8	0.9	1.4	0.1
<b>8</b>	0.1	0.2	0.4	0.4	0.5	0.7	0.8	0.9	1.2	0.1
<b>9</b>	0.1	0.2	0.4	0.4	0.5	0.7	0.8	0.9	1.1	0.1
<b>10</b>	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1

In fase di test dell'algoritmo risolutivo, i su indicati valori siano assegnati come valori iniziali della matrice mappa, nella definizione.

Successivamente i valori della matrice siano letti da un file, il cui nome sia introdotto da tastiera.

Nota: la prima riga e la prima colonna non fanno parte della matrice, sono riportati per comodità come numerazione delle colonne e delle righe.

Esercizio 6. Si realizzi un programma che permetta di introdurre da tastiera una matrice di interi. Le dimensioni della matrice,  $M \times N$ , possono essere fissate a priori con dei `#define`, ma sarebbe preferibile che fosse il programma a determinare automaticamente il numero di righe e il numero di colonne della matrice introdotta, entro i limiti massimi fissati dal programma (vedi nota). Il programma deve azzerare la sottomatrice di dimensioni  $m \times n$ , con  $m < M$ ,  $n < N$ , a partire dall'elemento  $r, c$  (riga, colonna). I valori di  $m$ ,  $n$ ,  $c$  ed  $r$  siano richiesti da tastiera. Il programma visualizzi la matrice prima e dopo l'azzeramento. Il programma deve controllare di non andare oltre i limiti della matrice.

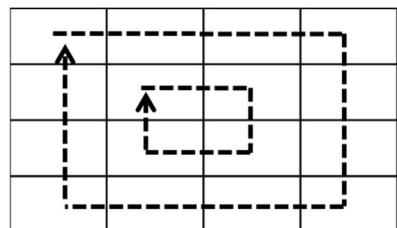
Approfondimento: generalizzare il programma realizzando l'operazione mediante una funzione che effettui la sostituzione di valore, il cui prototipo sia:

```
void Set (int m[][N], int m, int n, int i, int j, int val)
```

dove  $val$  è il valore da sostituire.

Nota: un modo semplice per rilevare le dimensioni di una matrice introdotta da tastiera è il seguente: si pone riga e colonna a zero. Si legge fino ad EOF (<CONTROL>+Z) un intero e il carattere successivo. Si memorizza il valore letto nella posizione (riga, colonna) e si incrementa colonna. Si testa poi il carattere letto: se è new-line (è finita la riga), si azzerà colonna e si incrementa riga. Alla fine del ciclo riga dice quante righe si sono lette, invece colonna deve essere memorizzata prima dell'azzeramento. Servono ovviamente i controlli per non superare il numero di righe e colonne effettive della matrice.

Esercizio 7. Si scriva un programma in grado di riempire una matrice quadrata di dimensioni  $N \times N$  di interi (con  $N$  pari e maggiore o uguale a 4 definito come costante tramite la direttiva `define`) secondo lo schema delle cornici concentriche; per ogni cornice si parta dalla cella in alto a sinistra e si riempia la cornice progressivamente con numeri crescenti a partire da 1.



1	2	3	4
12	1	2	5
11	4	3	6
10	9	8	7

Si proceda infine con la stampa della matrice assicurandosi che colonne e righe siano correttamente allineate.

# 12BHD INFORMATICA, A.A. 2019/2020

## Esercitazione di Laboratorio 10

---

### Obiettivi dell'esercitazione

- Scrivere programmi in grado di ricevere parametri da linea di comando
- Scrivere programmi in grado di gestire in modo efficiente grandi quantità di dati

### Contenuti tecnici

- I parametri da linea di comando *argc* e *argv*
- Consolidamento della lettura di dati da file

---

### *Da risolvere preferibilmente in laboratorio*

Esercizio 1. Realizzare un programma che riceva da linea di comando due numeri interi compresi nell'intervallo [-10000,10000], ne faccia la somma e la visualizzi. Il programma deve effettuare tutti i controlli necessari sul numero dei parametri e sulla loro correttezza.

Esercizio 2. Si scriva un programma che ricevuti tre parametri da linea di comando: *val1*, *val2* e *ch*, corrispondenti a 2 numeri interi (*val1*, *val2*) e una lettera (*ch*). Il programma esegua e visualizzi le seguenti operazioni fra i due valori a seconda del valore di *ch*:

- *ch* = *a*: *val1* + *val2*
- *ch* = *b*: *val1* - *val2*
- *ch* = *c*: *val1* \* *val2*
- *ch* = *d*: *val1* / *val2* se *val2* diverso da 0, altrimenti segnali un errore.

Esercizio 3. Scrivere un programma che permetta di memorizzare un insieme di coordinate del piano cartesiano.

Il programma deve acquisire da tastiera in due vettori paralleli le coordinate *x* e *y* di 4 punti, corrispondenti ai punti toccati da un percorso e manipolare i dati ricevuti in base ad un parametro ricevuto da linea di comando:

- Parametro “*-m*” : calcola e stampa a video la lunghezza del percorso composto dai 4 segmenti.
- Parametro “*-a*” : calcola e stampa a video la distanza minima tra le coordinate inserite.

---

### *Da risolvere a casa*

Esercizio 4. <sup>1</sup>Si modifichi l'esercizio 3 in modo che riceva da linea di comando, oltre al parametro “*-a*” o “*-m*”, le quattro coordinate nel seguente modo:

---

<sup>1</sup> Questo esercizio sarà svolto in modo multimediale e inserito sul Portale, tra il materiale comune, nelle settimane successive.

```
program.exe -a x1,y1 x2,y2 x3,y3 x4,y4  
oppure:  
program.exe -m x1,y1 x2,y2 x3,y3 x4,y4
```

Attenzione: ogni coppia di coordinate  $x_i, y_i$  deve essere scritta senza spazi in mezzo, mentre le coppie di coordinate devono essere separate da almeno uno spazio.

Esercizio 5. Si consideri il seguente esercizio già proposto e riportato qui per comodità.  
Si chiede di realizzarlo leggendo la matrice da un file (invece che dalla tastiera) il cui nome viene passato come primo parametro nella riga di comando, sviluppando anche la seconda parte facoltativa, in cui la lunghezza della sequenza e il valore da cercare sia passati rispettivamente come secondo e terzo parametro.

Testo dell'esercizio:

Si scriva un programma C che:

- legga da tastiera una matrice quadrata di dimensione uguale a 5 righe e 5 colonne
- rintracci se tale matrice contiene delle sequenze di elementi adiacenti uguali a zero di lunghezza uguale o maggiore di 3
- visualizzi l'indice di riga in cui tali sequenze si presentano.

Esempio.

Sia la matrice la seguente:

```
0 0 0 4 5  
1 2 0 4 5  
1 0 0 4 0  
1 2 3 4 5  
1 0 0 0 0
```

La sequenza di valori "0 0 0" compare nella prima e nell'ultima riga e quindi occorre riportare una indicazione del tipo:

La sequenza compare nella riga 1

La sequenza compare nella riga 5

Si osservi che la riga 3 non contiene la sequenza indicata in quanto i tre zeri non si trovano in posizioni contigue.

#### FACOLTATIVO

Si effettui lo stesso controllo anche lungo le colonne.

Nell'esempio precedente occorre visualizzare, oltre ai messaggi già indicati, anche il seguente:

La sequenza compare nella colonna 3

## FACOLTATIVO

Si generalizzi l'esercizio precedente per gestire sequenze di lunghezza variabile e contenenti un valore variabile.

Ovvero si leggano da tastiera due valori che specificano la lunghezza e il valore contenuto nelle sequenze da ricercare.

Nel caso sia:

Lunghezza della sequenza = 3

Valore nella sequenza = 0

occorre risolvere l'esercizio originale.

Se invece si introduce

Lunghezza della sequenza = 4

Valore nella sequenza = 1

occorre ricercare delle sequenze di 4 valori adiacenti uguali a 1,  
ovvero "1 1 1 1".

# 12BHD INFORMATICA, A.A. 2019/2020

## Esercitazione di Laboratorio 11

---

### Obiettivi dell'esercitazione

- Scrivere programmi complessi con lettura da file e uso delle strutture

### Contenuti tecnici

- Consolidamento della lettura di dati da file
- La definizione di strutture dati
- `typedef` e .

---

### Da risolvere preferibilmente in laboratorio

Esercizio 1. Scrivere un programma che permetta di memorizzare un insieme di coordinate del piano cartesiano. Nel programma si definisca una struttura dati contenente due campi come segue:

```
struct coordinate
{
    int x;
    int y;
};
```

Il programma deve acquisire da tastiera le coordinate  $x$  e  $y$  di 4 punti, corrispondenti ai punti toccati da un percorso e manipolare i dati ricevuti in base ad un parametro ricevuto da linea di comando:

- Parametro “ $-m$ ” : calcola e stampa a video la lunghezza del percorso composto dai 4 segmenti.
- Parametro “ $-a$ ” : calcola e stampa a video la distanza minima tra le coordinate inserite.

Esercizio 2. Si scriva un programma per la gestione di una rubrica di massimo 100 persone. Il programma deve permettere la memorizzazione dei seguenti dati in una struttura:

```
typedef struct nomi {
    - Nome                                char nome[20];
    - Cognome                             char cognome[20];
    - Numero di telefono fisso          char fisso[20];
    - Numero di telefono mobile         char mobile[20];
} Nomi;
```

Il programma deve permettere all'utente di effettuare, tramite scelta da menu, l'inserimento di un nuovo nominativo (tollerando eventuali duplicazioni di nome e cognome, ma segnalandole all'utente e chiedendo un'ulteriore conferma), e la stampa dell'elenco completo.

Esercizio 3. <sup>1</sup>Si scriva un programma in linguaggio C che legga il contenuto di un file dopo averne ricevuto il nome da linea di comando. Il numero di righe del file sia al massimo 80 e ciascuna riga del file contenga i seguenti campi, ciascuno composto al massimo da 20 caratteri e privo di spazi

*<materia> <nome prof> <cognome prof> <periodo> <crediti> <% superamento esame>*

Il programma dovrà stampare a video:

- a. il nome della materia che assegna più crediti in assoluto
- b. per ciascun periodo didattico (considerandone al massimo 4), la materia più difficile da superare

Il programma dovrà infine richiedere l'inserimento da tastiera di un cognome di professore (massimo 20 caratteri) e stampare a video:

- c. la somma dei crediti assegnati dalle materie che insegna
- d. la media di superamento degli esami da lui tenuti.

Approfondimento: nei punti a) e b), se si verificasse il caso in cui siano presenti due o più materie che assegnano il numero massimo di crediti o abbiano nello stesso periodo il minor tasso di superamento, il programma dovrà stampare l'elenco completo delle materie identificate

Esempio: di seguito un possibile contenuto di file:

*Equitazione Donato Cavallo 1 5 50*

*Canottaggio Remo Controcorrente 2 4 70*

*Velocità Tina Svelta 1 10 80*

#### Da risolvere a casa

Esercizio 4. Si scriva un programma che legga da un file (il cui nome è ricevuto come primo parametro sulla linea di comando) alcune informazioni ferroviarie. Per ciascuna linea, il file contiene le seguenti informazioni (ciascuno dei campi non superi i 20 caratteri di lunghezza e sia privo di spazi)

*<stazione\_partenza> <ora\_partenza> <stazione\_arrivo> <ora\_arrivo>*

Il programma può eventualmente ricevere come ulteriore parametro da linea di comando il nome di una città; in tale caso, il programma calcoli e stampi il numero di treni in arrivo ed il numero di treni in partenza da tale città (se inclusa nell'elenco).

---

<sup>1</sup> Questo esercizio sarà svolto in modo multimediale e inserito sul Portale, tra il materiale comune, nelle settimane successive.

- Esercizio 5. Scrivere un programma in linguaggio C per la gestione di una gara che:
- legga da file di tipo testo gara.txt una sequenza, in ordine qualsiasi, di nomi e relativo punteggio. I nomi s'intendono senza spazi in mezzo e il punteggio è un intero; si può ipotizzare che ci siano al più 100 atleti;
  - visualizzi sul video l'elenco ordinato in senso alfabetico, con relativo punteggio;
  - contemporaneamente effettui l'output su un file di tipo testo, garaord.txt degli stessi dati ordinati;
  - visualizzi sul video i nomi dei primi tre classificati, senza riordinare l'elenco.

Si rammenti che il primo classificato è quello che ha avuto il punteggio massimo, il secondo è quello che ha avuto il punteggio massimo inferiore al primo, e così via (occorre, in generale, cercare il massimo inferiore al massimo precedente).

Nota: si consiglia di utilizzare l'inserimento ordinato, visto nelle esercitazioni passate, in un vettore di struct.

- Esercizio 6. Realizzare una programma per la gestione degli esami di questo corso.  
I comandi che il programma deve gestire sono:

- I <Cognome\_Nome> <Matricola> <Voto>  
per inserire i dati di uno studente.
- C <Cognome\_Nome>  
per cancellare i dati dello studente con quel nome.
- C <Matricola>  
per cancellare i dati dello studente con quella matricola.
- V <Cognome\_Nome>  
per visualizzare i dati dello studente con quel nome.
- V <Matricola>  
per visualizzare i dati dello studente con quella matricola.
- P  
per stampare tutto il data-base.
- L <Nomefile>  
per leggere i dati (già ordinati) dal file con quel nome.
- S <Nomefile>  
per salvare i dati del data-base nel file con quel nome.

Utilizzare un vettore di struct ordinato in base al campo <Cognome\_Nome>.

NOTE: i comandi sono da fornire (e il programma deve accettarli) nella forma fissata dalle specifiche, senza inutili domande ridondanti. Ad esempio, per inserire i dati di uno studente:

*I Rossi Francesco Maria 123456 30*

Pertanto per i comandi C (cancellazione) e V (visualizzazione) il programma deve comprendere da solo (dal dato) se si è inserito un nome oppure una matricola.

Si ipotizzi che i Cognome\_Nome non contenga cifre, che la Matricola sia solo numerica e che l'input sia corretto.