Tecniche di Programmazione

Esercitazione 3

Allocazione dinamica

Esercizio 3.1

Scrivere un programma che inserita una sequenza di interi positivi in input ne restituisca il minimo usando solamente variabili di tipo puntatore ad int, anziché variabili di tipo int. La sequenza di input termina quando viene inserito il numero 0 (escluso).
Tutta la memoria utilizzata deve essere allocata dinamicamente (`malloc`), e quando non più utilizzata deve essere rilasciata.

Nota bene: il programma deve controllare che la sequenza inserita sia costituita da numeri interi positivi.

Esercizio 3.2

Ripetere il precedente esercizio usando un'unica chiamata della funzione `malloc`.
Suggerimento: allocare la memoria per due interi adiacenti.

Esercizio 3.3

Scrivere un programma che prenda in input un reale ed un carattere rappresentanti rispettivamente un valore di temperatura e la scala di temperatura scelta ed effettui la conversione nelle altre scale, utilizzando puntatori e allocazione dinamica della memoria.
Le scale di Temperatura da considerare sono:

- Celsius (carattere ‘C’)
- Kelvin (carattere ‘K’)
- Fahrenheit (carattere ‘F’).

Esempio:

```
t=400.0
c='C'
```
deve stampare i valori
```
K = 673.15
F = 752.0
```

Tutta la memoria allocata dinamicamente deve essere rilasciata quando non più utilizzata.

Formule di conversione:
`K = C + 273.15`
`F = C * 9/5 + 32`

Esercizio 3.4

Ripetere il precedente esercizio usando un'unica chiamata della funzione `malloc`.

Esercizio 3.5

Scrivere un programma che prenda in input da tastiera un intero `N`, successivamente allochi dinamicamente in memoria lo spazio di `N` interi, e ne stampi indirizzo e contenuto di ognuno di essi.

Funzioni

Definire un ***singolo*** file in cui sono presenti le dichiarazioni di tutte le funzioni dei seguenti esercizi.
Scrivere ***una*** funzione main per effettuare i test delle funzioni in maniera incrementale.

Esercizio 3.6

Scrivere la funzione

```
int * allocaInt();
```

che alloca dinamicamente in memoria un intero e ne restituisce in uscita il puntatore.

Esercizio 3.7

Scrivere la funzione

```
void printInt(int *i1, int *i2);
```

che, dati in input due puntatori ad intero ne stampa il valore.

Esercizio 3.8

Scrivere la funzione

```
void soluzioneEquazione(int a, int b, int c);
```

che, dati in input due interi, risolva l’equazione di secondo grado :

`ax^2 + bx + c = 0`

E stampi i risultati.

Esercizio 3.9

Scrivere la funzione

```
int MCD(int i1, int i2);
```

che, dati in input due interi (`i1,i2`) calcoli il massimo comun divisore e ritorni il risultato.

Esercizio 3.10

Scrivere la funzione

```
int mcm(int i1, int i2);
```

che, dati in input due interi (`i1,i2`) calcoli il minimo comune multiplo e ritorni il risultato.

Esercizio 3.11

Scrivere la funzione

```
void conversioneTemperatura(float t, char c);
```

Che prende in input una temperatura `t` e una scala di riferimento data dal char `c`. La funzione deve effettuare la conversione nelle altre scale, esattamente come nell'esercizio 3.3.

Esercizio 3.12

Scrivere la funzione

```
void* conversioneTemperaturaP(float *t, char c);
```

che, dati in input un puntatore a float `t` e un char `c`, svolga lo stesso compito della funzione precedente.
Inoltre, deve ritornare il puntatore alla memoria allocata contenente la soluzione (il risultato della conversione in *entrambe* le scale).
Le due conversioni risultanti devono essere salvate in una zona di memoria allocata dinamicamente con un'unica chiamata alla funzione `malloc()`.

NB: leggere il prossimo esercizio per un po' di contesto.

Esercizio 3.13

Scrivere la funzione

```
void printConversione(void *temperatura);
```

che, dato in ingresso un puntatore a `void` stampi il risultato della funzione precedente.