Università di Catania
Dipartimento di Matematica e Informatica
Corso di Studio in Informatica, A.A. 2024-2025
Prova di Laboratorio
19 Febbraio 2025

Descrizione del programma

Scrivere un programma in C che:

A. Prenda un input da tastiera (argomenti della funzione main) costituito da due numeri in virgola mobile **x** e **y**, ed un numero intero **n**. Tali parametri devono rispettare le seguenti specifiche:
   - I numeri x e y devono appartenere entrmabi al range [100.0,1000.0]
   - Deve essere y>x
   - Deve essere y-x>300.0;
   - il numero n deve appartenere al range [15,30].

B. Produca n numeri in virgola mobile (double) che appartengano al range [x, y]; inserisca ogni numero in una struttura dati dinamica **PILA**, e tale numero sia stampato sullo standard output.

C. Rimuova dalla pila tutti gli elementi precedentemente inseriti e stampi sullo standard output tali elementi (i numeri double).

D. Con riferimento al punto C, gli elementi rimossi dalla pila vanno contestuamente salvati all'interno di un array di n elementi double.

E. Si stampi infine la media degli n valori contenuti all'interno dell'array ed il numero di elementi che risultano maggiori della media stessa.

Specifiche

Il programma potra' essere strutturato in un unico file sorgente, ma dovra' contenere almeno le seguenti funzioni:

1. **readInput():** funzione che prende in input l'array di puntatori a carattere argv ed il numero di argomenti argc della funzione main, controlla che gli argomenti richiesti siano nel numero e nei limiti specificati, e restituisca i parametri {x,y,n} in una struct da restituire al chiamante.

2. **genDouble()**: funzione che produca un valore double pseudo-casuale che appartenga ad un range specificato mediante parametri formali. NB: impiegare **opportunamente la funzione di generazione di numeri pseudo-casuali riportata in seguito nel testo (get_random()) e prensente nel file random.c, unitamente alla costante** UINT_MAX (<limits.h>).

3. **Push() e Pop()**: funzioni per la gestione della pila;

4. **buildStack()**: funzione che crei una pila di numeri double sulla base delle specifiche presenti al punto B, utilizzando opportunamente la funzione genDouble() e la funzione push(). La funzione dovra' stampare sullo standard output tutti i numeri generati ed inseriti nella pila, come specificato nel punto B.

5. **buildArray()**: funzione che rimuove tutti gli elementi dalla pila mediante la funzione pop(), stampa i valori sullo standard output come specificato nel punto C, ed infine che li inserisce in array di n double.

6. **elabValues()**: funzione che prende in input un array di n double, ne calcoli la i) media aritmetica ed ii) il numero di valori dell'array che risultano maggiori a tale media. Infine stampa i) e ii) sullo standard output.

È VIETATO usare variabili globali.

Durata della prova: 120 minuti. NB: Inserire nome, cognome e numero di matricola all'interno del file sorgente.

Generazione di numeri pseudocasuali:

- Si consideri la seguente funzione get_random() per la generazione di numeri pseudo-casuali interi positivi (la funzione e' presente nel file **random.c situato nella home directory della macchina virtuale**):

```
unsigned int get_random() {
      static unsigned int m_w = 123456;
      static unsigned int m_z = 789123;
      m_z = 36969 * (m_z & 65535) + (m_z >> 16);
      m_w = 18000 * (m_w & 65535) + (m_w >> 16);
      return (m_z << 16) + m_w;
```

```
    }
```

NB: Ai fini della eventuale generazione di numeri in virgola mobile, si faccia uso della costante UINT_MAX (<limits.h>) unitamente alla funzione get_random().

Input e output di test (file output.txt situato nella home directory della macchina virtuale):

(./a.out <x> <y> <n>)

```
$./a.out 200 800 20


** build_stack() **
Dato inserito (0): 345.122099
Dato inserito (1): 755.072944
Dato inserito (2): 682.017512
Dato inserito (3): 204.803674
Dato inserito (4): 468.580755
Dato inserito (5): 366.906065
Dato inserito (6): 797.045489
Dato inserito (7): 589.037990
Dato inserito (8): 464.557631
Dato inserito (9): 433.587222
Dato inserito (10): 581.251165
Dato inserito (11): 710.368997
Dato inserito (12): 727.231486
Dato inserito (13): 310.602995
Dato inserito (14): 591.033284
Dato inserito (15): 214.968371
Dato inserito (16): 349.200717
Dato inserito (17): 283.710068
Dato inserito (18): 212.181921
Dato inserito (19): 265.625206

** build_array() **
Dato estratto (0): 265.625206
Dato estratto (1): 212.181921
Dato estratto (2): 283.710068
Dato estratto (3): 349.200717
Dato estratto (4): 214.968371
Dato estratto (5): 591.033284
```

```
Dato estratto (6): 310.602995
Dato estratto (7): 727.231486
Dato estratto (8): 710.368997
Dato estratto (9): 581.251165
Dato estratto (10): 433.587222
Dato estratto (11): 464.557631
Dato estratto (12): 589.037990
Dato estratto (13): 797.045489
Dato estratto (14): 366.906065
Dato estratto (15): 468.580755
Dato estratto (16): 204.803674
Dato estratto (17): 682.017512
Dato estratto (18): 755.072944
Dato estratto (19): 345.122099

** elabValues(), avg=467.645280, no. of values > 467.645280 = 9
```