UNIVERZITET U TUZLI
FAKULTET ELEKTROTEHNIKE

Objektno-orijentirano programiranje

Zadaća 2

Tuzla, novembar/studeni 2018.

Sadržaj

Zadatak 1

Prosti brojevi su svi prirodni brojevi djeljivi samo sa 1 i sami sa sobom.

a) Kreirati funkciju koja će testirati da li je proslijeđeni broj prost. Ukoliko se proslijedi broj manji od 1, funkcija treba da generiše iznimku tipa `std::invalid_argument`.
   `bool isPrime(int);`
b) Napisati funkciju koja vraća prvih n prostih brojeva.
   `std::vector<int> primes(int n);`
c) Napisati program koji ispisuje sumu prvih 300'000 prostih brojeva.

Napomena:
Prilikom nalaženja sume prostih brojeva, kalkulacija velikih brojeva će usložiti. Obratiti pažnju na algoritam provjere da li je broj prost. Da li je neophodno provjeravati sve brojeve `[1 - n]`? Koji opseg je dovoljan?

Zadatak 2

Napisati program koji će odraditi animaciju na ekranu kao što je to prikazano na sljedećem linku. Ukoliko se veličina linije terminala u međuvremenu promjeni, sama animacija treba da se prilagodi novim uslovima. Pomoćne funkcije su date u prilogu zadaće:

- `size_t broj_kolona()` - vraća nazad broj karaktera koji stane u jednu liniju terminala. Ukoliko se veličina terminal prozora promjeni sljedeći poziv funkcije će vratiti novu veličinu terminal linije.
- `void pauziraj(unsigned int msec)` - Funkcija koja blokira program (nit pozivatelja funkcije) `msec` milisekundi.

Napomena:
Sljedeća linija koda: `std::cout << '\r';` će vratiti kursor na početak trenutne linije. Više informacija o `'\r'` (carriage return character) na sljedećem linku.

Zadatak 3

Napisati program koji će prikazati histogram za unesenu listu pozitivnih i negativnih cijelih brojeva koji predstavljaju težinske faktore. Iscrtani histogram treba da podijeli terminal prozor na dva jednaka dijela. Na lijevom dijelu prozora treba da bude negativna težinska vrijednost, dok na desnom pozitivna težinska vrijednost. Obratiti pažnju da podaci trebaju biti skalirani prema veličini linije terminala (iskoristiti `broj_kolona()` funkciju iz prethodnog zadatka) i to tako da najveća vrijednost ide do kraja svoje polovine linije samo u slučaju da je veća od svoje polovine (`broj_kolona() / 2`) ostale skalirati prema najvećoj težinskoj vrijednosti.

Primjer skaliranja, ukoliko funkcija `broj_kolona()` vrati vrijednost *90*, a korisnik unese brojeve *200, 300, 15, 40,* nakon skaliranja dobiti ćemo naredne vrijednosti: *30, 45, 2, 6.*

Primjer izvršenja programa na sljedećem linku.

Zadatak 4

Napisati funkciju `parni_neparni` koja uzima listu cijelih brojeva kao argument te nazad vraća modifikovanu verziju ulazne kolekcije takvu da su svi parni članovi na lijevoj strani, a svi neparni na desnoj strani kolekcije. Funkcija ne treba modifikovati proslijeđeni argument.

Kako treba izgledati potpis funkcije? Obrazložiti odgovor.

```
A. std::vector<int> parni_neparni(std::vector<int>&);
B. std::vector<int> parni_neparni(std::vector<int>);
C. std::vector<int> parni_neparni(const std::vector<int>&);
```

Zadatak 5

Napisati poboljšanu verziju prethodne funkcije takvu da osim ulazne liste cijelih brojeva, uzima i `predicate` funkciju kao argument (`std::function<bool(int)>` iz `functional` zaglavlja). Funkcija treba da poziva `predicate` nad svakim članom ulaznog argumenta, te ukoliko je rezultat `predicate` poziva tačan, član treba da se nalazi na lijevoj strani, odnosno ukoliko je netačan na desnoj strani rezultujuće kolekcije.

Primjer poziva partition funkcije:

```
#include <iostream>
#include <functional>
#include <vector>

/* partition implementation */

int main() {
      std::vector<int> nums{15,20,25,-10,-75,100,-255,430,200};
      auto result = partition(nums, [](int num) {
                                      // predicate
                                       return !(num % 2);// funkcija
                                    });
     for (auto num : result) {
           std::cout << num << ' ';
      }
      std::cout << std::endl;
      return 0;
}
```

Mogući ispis:

```
200 20 430 -10 100 -255 -75 25 15
```

Testirati implementaciju sa sljedećim predicate funkcijama:

- `[](int num) { return num > 0; }`
  Mogući ispis: `15 20 25 200 430 100 -255 -75 -10`
- `[](int num) { return !(num % 10); }`
  Mogući ispis: `200 20 430 -10 100 -255 -75 25 15`
- `[](int num) { return std::abs(num) > 99; }`
  Mogući ispis: `200 430 -255 100 -75 -10 25 20 15`

Zadatak 6

Modificirati zadatak4.cpp sa vježbi broj 7 tako da funkcija *transformiraj* tretira iznimke koje može generisati predikat funkcija. U slučaju da predikat funkcija baci iznimku tipa `std::invalid_argument` na trenutnu lokaciju upisati vrijednost 0. U slučaju da predikat funkcija baci iznimku tipa `std::string` na trenutnu lokaciju upisati vrijednost 1. U slučaju bilo koje druge iznimke, potrebno je zaustaviti izvršavanje funkcije.
Testirati poziv funkcije *transformiraj* sa različitim predikat funkcijama (funkcije koje bacaju iznimke različitih tipova).