UNIVERZITET U TUZLI
FAKULTET ELEKTROTEHNIKE

Objektno-orijentirano programiranje

Zadaća 1

Tuzla, novembar/studeni 2019.

Sadržaj

Zadatak 1

Napisati program koji predstavlja konverter mjernih jedinica. Program treba da sadrži:

1. Konverter mjernih jedinica temperature. Omogućiti unos temperature u Kelvinima, stepenima Celsius-a ili stepenima Fahrenheit-a.
2. Konverter mjernih jedinica brzine. Dati izbor unosa brzine u miljama ili kilometrima po satu. Ispisati brzinu u drugoj mjernoj jedinici.
3. Konverter dužine između svjetlosnih godina i kilometara, stopa i metara te inča i centimetara.
4. Konverter mase između kilograma i funte (kg ↔ lb)
5. Konverter potrošnje goriva između litara na 100 kilometara u milje po galonu.

Program kada se starta treba da ispiše meni za korisnika. Primjer korištenja programa je dat ispod.

```
Welcome to Unit converter. Please enter one of the following
options:
      1. Temperature
      2. Speed
      3. Length
      4. Weight
      5. Fuel economy
Your choice: 1
Please choose converter:
      1. Celsius to Fahrenheit
      2. Fahrenheit to Celsius
      3. Celsius to Kelvin
      4. Kelvin to Celsius
      5. Fahrenheit to Kelvin
      6. Kelvin to Fahrenheit
Your choice: 2
Enter temperature in degrees Fahrenheits: 105
105 degrees Fahrenheit is 40.5556 degrees Celsius.
```

Obratiti pažnju na organizaciju programa. Obavezno konstante za različite kategorije smjestiti u različite namespace-e. Na primjer:

```
namespace Temperature {
      const double celsiusKelvinCoeff = 273.15;
}
namespace Length {
      const double inchCmCoeff = 2.54;
}
```

Note: Obratiti pažnju na nevalidne unose. Negativna vrijednost Kelvina ne postoji, temperatura ispod -273.15 Celzija također. Svjetlosna godina je preko deset magnituda veća od kilometra, obratiti pažnju da se ne desi overflow pri računanju.

Zadatak 2

Napisati program koji će simulirati 16-bitni registar koristeći primitivne tipove iz C++-a. Potrebno je implementirati operacije setovanja i resetovanja bita na registru. Dio koda je već ispisan (u prilogu zadaće nalazi se file: `zadatak2.cpp`), na Vama je da dopunite dijelove koji nedostaju mijenjajući priložen file po želji. Jedan use-case navedenog programa je dat ispod.

```
1. Print register
2. Set bit in register
3. Reset bit in register
4. Exit
Enter option: 1
0000000000000000

1. Print register
2. Set bit in register
3. Reset bit in register
4. Exit
Enter option: 2
Enter bit number: 0
New register value: 1
0000000000000001

1. Print register
2. Set bit in register
3. Reset bit in register
4. Exit
Enter option: 2
Enter bit number: 3
New register value: 9
0000000000001001

1. Print register
2. Set bit in register
3. Reset bit in register
4. Exit
Enter option: 3
Enter bit number: 0
New register value: 8
0000000000001000
```

Nakon ovoga, proširiti zadatak za još jedan registar te prije biranja operacije nad registrom odabrati koji registar korisnik želi modifikovati (ispisati). Osim toga, uvesti još jednu opciju gdje se vrijednosti dva registra zamjene (registar a dobije vrijednost registra b, te registar b dobije vrijednost registra a). Zamjenu vrijednosti dva registra moguće je odraditi i bez privremene varijable?

Zadatak 3

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 (file: `zadatak3.cpp`):

- `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.

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

Zadatak 4

Napisati program koji će odraditi animaciju koristeći funkcije iz prethodnog zadatka. Ovaj puta korisnik zadaje granice prema kojima će se animacija kretati. Granice se zadaju u procentima od 0 do 100. Zbir 4 granice koje korisnik unese mora biti jednak 100 (niti jedan više, niti jedan manje). Unešeni procenti su ustvari postotak broja kolona u trenutnoj liniji terminala te se animacija mora tome i prilagoditi. Obratiti pažnju da se na početku i kraju animacije nalaze uglaste zagrade. Primjer izvršavanja možete pronaći na sljedećem linku.

Note:
Ukoliko Vam ostane jedno polje na kraju (prije zatvorene uglaste zagrade) koje ne može stati zbog upoređivanja cijelih i decimalnih brojeva, ili zbog cjelobrojnog dijeljenja, ostatak možete dodati na kraj četvrtog dijela animacije.