UNIVERZITET U TUZLI
FAKULTET ELEKTROTEHNIKE

Objektno-orijentirano programiranje

Zadaća 2

Tuzla, novembar/studeni 2020.

Sadržaj

Zadatak 1

Napisati program koji će učitavati engleski rječnik iz file-a `words.txt` koji se nalazi u prilogu zadaće. Riječi učitati u vektor te napisati funkciju `over25chars` koja će uzeti ovaj vektor kao argument te vratiti nazad sve riječi koje imaju više od 25 karaktera. Funkciju pozvati iz main-a te ove riječi ispisati na ekran. Mjeriti razliku u izvršenju programa kada vektor uzimamo po referenci i po kopiji kao argument. Za mjerenje koristiti komandu `time`. Ukoliko nam se izvršni file zove `zad3` mjerenje je moguće odraditi na sljedeći način:

```
time ./zad3
```

Za učitavanje rječnika potrebno je koristiti objekat tipa `ifstream` iz zaglavlja `fstream`. Objekti ovog tipa imaju vrlo slično ponašanje `istream` objektu `cin`. Detalje `ifstream`-a možete pročitati na sljedećem linku, a primjer učitavanja `words.txt` file-a u vektor je dat ispod. Osim ovoga na početku programa je potrebno učitati `fstream` zaglavlje te je file koji čitamo (u našem slučaju `words.txt`) potrebno postaviti u isti folder kao izvršni file.

```
#include <fstream>
//...
  std::ifstream dictionaryFile("words.txt");
  std::vector<std::string> dictionary;
  std::string word;
  while (dictionaryFile >> word) {
    dictionary.push_back(std::move(word));
  }
```

Zadatak 2

Napisati program koji će učitati rječnik iz zadatka 3 te zatražiti od korisnika unos jedne riječi sa tastature. Nakon unesene riječi potrebno je pronaći sve riječi iz učitanog rječnika čija je Hamming-ova udaljenost strogo manja od 2 (uključujući i riječ čija je udaljenost 0). Za ovu funkcionalnost potrebno je iskoristiti funkciju iz zadatka 2. Na kraju, program treba da ispiše pronađene riječi sortirane po udaljenosti (od najmanje do najveće udaljenosti). Primjer izvršenja programa na sljedećem linku.

Obavezno koristiti vektor kao kolekciju za implementaciju ovog zadatka. Kod sortiranja ključan detalj je funkcija kriterija sortiranja. Šta je potrebno proslijediti u klauzulu lambda izraza?

Zadatak 3

Potrebno je implementirati jednostavan kalkulator koji operira nad jednim double-om. Kalkulator treba da podržava operacije sabiranja, oduzimanja, množenja, dijeljenja, kvadriranja, korjenovanja, undo i redo. Primjer izvršenja programa se nalazi na sljedećem linku. Dio implementacije ovog zadatka se nalazi u prilogu zadaće.

Zadatak 4

Primjer sa predavanja broj 3 je razbijen u funkcije i nalazi se u prilogu zadaće. Potrebno je modifikovati priloženi kod tako da se od korisnika tražiti unos sve dok ne unese barem jedan element u listu. Primjer izvršenja programa na sljedećem linku.

Zadatak 5

Napisati program koji će tražiti od korisnika unos rečenice nakon čega program treba da ispiše koliko se puta svako slovo u rečenici ponavlja. Program treba da obraća pažnju na mala i velika slova unutar rečenice te ne treba ispisivati slova koja se nikada nisu pojavila u rečenici.

Zadatak 6

Napisati program koji od korisnika sa standardnog ulaza uzima sekvencu karaktera. Dozvoljene sekvence su “END” i sekvence koje uključuju kombinacije karaktera ‘A’, ‘B’, ‘C’ i ‘D’. Ukoliko korisnik unese nepravilnu sekvencu ispisati adekvatnu poruku. Nakon što korisnik unese ispravnu sekvencu, potrebno je da se ispiše vrijednost sekvence. Vrijednost sekvence se računa kao suma vrijednosti pojedinačnih karaktera unutar sekvence(gdje je ‘A’ = 135, ‘B’ = 151, ‘C’ = 111 i ‘D’ = 126) + dužina sekvence * 180 . Primjerice za sekvencu “AABD” vrijednost je 1267(135 + 135 + 151 + 126 + 4 * 180). Ova petlja treba da se nastavi dok korisnik ne unese sekvencu END. Nakon unosa sekvence END korisniku treba ispisati sve dotad unesene sekvence sortirane silazno po vrijednosti. Ukoliko dvije sekvence imaju istu vrijednost, onda o redoslijedu treba odlučiti leksikografski (u ovu svrhu koristiti metod compare iz string klase).