Tekstualne datoteke

Ciljevi:

- upoznati se s pojmom tekstualne datoteke
- upoznati se s načinima rada s tekstualnim datotekama
- proučiti funkcije i metode koje se koriste pri radu s tekstualnim datotekama

Pregled lekcije

Definicija sloga

Tekstualna datoteka je sekvencijalna datoteka u kojoj je sadržaj zapisan sljedno tj. znak to znaka. Unutar tekstualne datoteke postoje još i dvije specijalne oznake a to su oznaka za kraj reda i oznaka za kraj same datoteke.

Postoje 3 biblioteke zaglavlja za rad s tekstualnim datotekama. To su:

- **ofstream –** za upis u datoteku
- **ifstream –** za ispis iz datoteke
- **fstream –** za upis i ispis iz datoteke

Možemo odmah primijetiti sličnost s bibliotekama zaglavlja za standardni upis/ispis (tokovi cin i cout) na ekran:

- **ostream –** za ispis
- **istream –** za upis
- **iostream –** za upis i ispis

Dakle, ovdje se radi o sličnim tokovima, samo što su oni usmjereni u tekstualnu datoteku a ne na ekran.

```
#include <iostream>
#include <fstream>
using namespace std;

int main () {
  ofstream datoteka;
  datoteka.open ("primjer.txt");
  datoteka << "Upis prvog reda.\n";
  datoteka << "Upis drugog reda.\n";
  datoteka.close();
  system("PAUSE");
  return 0;
}
```

Navedeni primjer stvara obekt klase ofstream koji predstavlja datotečni tok. Stvoreni objekt nam omogućava rad s tekstualnom datotekom tj. upis u datoteku. U slučaju našeg primjera koristeći metodu open stvara se datoteka pod nazivom primjer.txt. Metoda open pridružuje fizičku datoteku na disku stvorenom toku datoteka. Zatim se u datoteku upisuju 2 reda teksta (oznaka za kraj reda je \n) na način da se tekst usmjerava u datotečni tok. Metoda close zatvara datoteku tj. izlazni tok.

Da bismo otvorili fizičku datoteku koristeći objekt toka koristimo funkciju (metodu) open().

Sintaksa:

```
open (naziv datoteke, mod otvaranja)
```

Parametar 'naziv datoteke' je obavezan i on predstavlja fizički naziv datoteke koja će biti otvorena/stvorena. Parametar 'mod' nije obvezan a sastoji se od sljedećih modova (načina):

ios::in – otvara datoteku u modu za upis
ios::out – otvara datoteku u modu za ispis
ios::binary – otvara datoteku u binarnom modu (tekstualne datoteke nisu binarne)
ios::ate – postavlja inicijalnu poziciju pokazivača na kraj datoteke umjesto na početak
ios::app – sav upisani sadržaj u datoteku će se dodati na kraj već postojećeg sadržaja. Ovaj **mod** je moguć samo ukoliko je datoteka otvorena striktno za upis.
ios::trunc – sav sadržaj datoteke se briše prije upisa novog

Navedeni modovi otvaranja se mogu kombinirati koristeći logičko i (|). Dakle, ukoliko želimo otvoriti datoteku za upis, ispis i pri tome postaviti pokazivač na kraj datoteke napisat ćemo sljedeću liniju koda:

```
datoteka.open ("primjer.txt", ios::out | ios::in | ios::app);
```

Svaka klasa ima predefinirane (default) modove otvaranja u slučaju da se drugi parametar funkcije open() izostavi.

ofstream – ios::out
ifstream – ios::in
fstream – ios::in | ios::out

S obzirom da je funkcija open() obično prvo što se radi s objektom tipa datotečnog toka, svaka od spomenutih klasa (ofstream, ifstream i fstream) uključuje konstruktor koji automatski poziva open() funkciju. Tako da bi otvaranje datoteke iz prethodnog primjera mogli napisati i kao:

```
fstream datoteka ("primjer.txt", ios::out | ios::in | ios::app);
```


Da bismo provjerili da li je navedena linija koda prošla korektno tj. da li je datotečnom objektu zaista pridružena fizička datoteka, možemo koristiti funkciju is_open() koja vraća true/false ovisno o tome da li je rezultat pozitivan ili negativan.

Na kraju, po završetku rada pozivom funkcije close() zatvaramo fizičku datoteku pa ona ponovo postaje raspoloživa drugim procesima. Datotečni objekt se oslobađa i može se ponovo iskoristiti za povezivanje s nekom drugom datotekom.

Ako primijenimo sve spomenuto da bi unaprijedili naš programski primjer dobivamo sljedeći rezultat:

```
#include <iostream>
#include <fstream>
using namespace std;

int main () {
  ofstream datoteka ("primjer.txt");
  if (datoteka.is_open())
  {
      datoteka << "Upis prvog reda.\n";
      datoteka << "Upis drugog reda.\n";
      datoteka.close();
  }
  else cout << "Nije moguće otvoriti datoteku.";
  system("PAUSE");
  return 0;
}
```

Modificirajmo sada prethodni kod tako da umjesto upisa u datoteku, čitamo sav sadržaje niz nje. Dobivamo sljedeći kod:

```
#include <iostream>
#include <fstream>
#include <string>
using namespace std;

int main () {
  string linija;
  ifstream datoteka ("primjer.txt");
  if (datoteka.is_open())
  {
      while (!datoteka.eof() )
      {
         getline (datoteka, linija);
         cout << linija << endl;
      }
      datoteka.close();
  }
  else cout << "Nije moguće otvoriti datoteku.";
  system("PAUSE");
  return 0;
}
```


U navedenom primjeru otvaramo datoteku za čitanje te ukoliko je otvaranje prošlo u redu čitamo datoteku red po red sve dok ne dođemo do kraja datoteke. Iteracija while će izvršavati čitanje linije teksta iz datoteke, koju će pohraniti u varijablu linija koja je tipa string (niz znakova), sve dok je

ispunjen uvjet da nije kraj datoteke. Kraj datoteke ispitujemo funkcijom eof() koja vraća true ukoliko je dosegnut kraj datoteke. Čitanje linije teksta iz datoteke vršimo pomoću funkcije getline. Getline prima 2 parametra, ulazni tok podataka i varijablu u koju će podaci preuzeti iz toka biti pohranjeni.

U sljedećem primjeru ćemo riješiti sljedeći zadatak: Potrebno je napisati program koji će od korisnika tražiti unos naziva datoteke a potom će iz datoteke pročitati i ispisati sve linije koda a nakon toga će ispisati broj pojavljivanja svakog pojedinog slova abecede u cjelokupnom sadržaju datoteke.

```
#include <iostream>
#include <fstream>
#include <cstring>
using namespace std;

struct sl {
       char slovo;
       int broj;
};

void ISort(sl A[]) {
     for (int i = 1; i<26; i++) {
         int j = i-1;
         sl pom = A[i];
         while (j >= 0 &&
                A[j].broj < pom.broj) {
               A[j--+1] = A[j];
         }
         A[++j] = pom;
     }
}

int main () {
    char datoteka[50], linija[20000];
    sl slova[26];
    for (int i = 0; i<26; i++) {
        slova[i].slovo = i+'A';
        slova[i].broj = 0;
    }
    cout << "Upisite ime datoteke s tekstom: ";
    cin.getline(datoteka,50);
    ifstream tekst(datoteka);
    if (!tekst) {
       cout << "Nepostojeca datoteka. "
            << endl;
       system("PAUSE");
       return 0;
    }

    while (tekst.getline(linija,20000)) {
          cout << "Linija:    " << linija << endl;
          int duljina = strlen(linija);
```

```
            for (int i = 0; i < duljina; i++) {
                if (linija[i] >= 'A' &&
                    linija[i] <= 'Z')
                   slova[linija[i]-'A'].broj++;
                if (linija[i] >= 'a' &&
                    linija[i] <= 'z')
                   slova[linija[i]-'a'].broj++;
            }
    }

    ISort(slova);
    for (int i = 0; i<26; i++)
        cout << slova[i].slovo << "\t"
             << slova[i].broj << endl;
    tekst.close();
    system("PAUSE");
    return 0;
}
```

U ovom primjeru definiramo strukturu sl u koju ćemo pohraniti broj pojavljivanja pojedinih slova. Nakon toga definiramo funkciju ISort koja će napraviti sortiranje umetanjem. Ovo će biti silazno sortiranje, pa znakove nejednakosti treba izokrenuti s obzirom na standardni algoritam sortiranja umetanjem koji smo imali ranije. Osim toga, ovdje treba prepisivati cijele slogove, no to nema veze jer se sa slogom može raditi kao s bilo kojom varijablom elementarnog tipa, tj. nad slogovima se može koristiti operator pridruživanja vrijednosti =.

U main funkciji korisnik upisuje ime datoteke i zatim se datoteka s tim nazivom otvara za čitanje. Da bismo pročitali sadržaj datoteke koristit ćemo getline naredbu kao uvjet while iteracije. Naredba getline ima povratnu vrijednost! Ako čitanje uspije vraća true, a inače vraća false.

Oprez: Naredba getline će vratiti false u slučaju neuspješnog čitanja, bez obzira što se dogodilo, a ne samo u slučaju da smo došli do kraja datoteke! (Npr. vratit će false i u slučaju da je redak datoteke dulji od polja u kojeg ga želimo pohraniti).

Unutar polja slova pohranit ćemo uz svaki znak broj pojavljivanja tog znaka, s time da velika i mala slova promatramo kao isti znak. Prebrojavanje vršimo u istoj while iteraciji koja je spomenuta ranije.

Recimo da želimo brojati slova na većem uzorku. Tada prethodne rezultate treba upisati u datoteku, kako bi se mogli koristiti u sljedećim prebrojavanjima Za to nam treba mogućnost zapisivanja u datoteku, odnosno izlazni tok podataka.

Program koji smo već uradili, treba doraditi tako da čita prijašnje rezultate i da nove rezultate dodaje na prije izračunate. Dakle, broj pojava nećemo stavljati na početku na 0, već u prijašnjim brojanjima dobijen rezultat. Na kraju ćemo u tu istu datoteku zapisati novi rezultat. U datoteci nećemo imati zapisane vrijednosti sortirane, već redom po slovima. To će nam omogućavati da u datoteku zapisujemo samo brojeve.

Problem je također ako se isti tekst broji više puta. To bi narušilo točnost podataka. Stoga želimo zapisati koje smo sve tekstove već pobrojali. Za to nam treba još jedna datoteka. Problem je u tome što u ovu datoteku moramo dodavati retke, a ne je svaki put pobrisati i kreirati iznova. Također, neka potrebe da se datoteka otvara za čitanje, zatvara i ponovo otvara za pisanje s drugim handlerom.

Na kraju želimo pojedine operacije podijeliti po funkcijama tako da program bude što čitljiviji.

```
#include <iostream>
#include <fstream>
#include <cstring>
using namespace std;

struct sl {
       char slovo;
       int broj;
};

void ISort(sl A[]) {
     for (int i = 1; i<26; i++) {
         int j = i-1;
         sl pom = A[i];
         while (j >= 0 &&
                A[j].broj < pom.broj) {
               A[j--+1] = A[j];
         }
         A[++j] = pom;
     }
}

void InitUc (sl uc[]) {
     ifstream ucest("ucestalost.txt");
     for (int i = 0; i<26; i++) {
         uc[i].slovo = i+'A';
         if (ucest) ucest >> uc[i].broj;
         else uc[i].broj = 0;
     }
     if (ucest) ucest.close();
}

void Ulaz(char dat[]) {
     char linija[20000];
     cout << "Upisite ime datoteke s tekstom: ";
     cin.getline(dat,50);
     ifstream tekst(dat);
     if (!tekst) {
        cout << "Nepostojeca datoteka. "
             << endl;
          system("PAUSE");
```

```
            exit(0);
      }
      ifstream dato("datoteke.txt");
      while (dato && dato.getline(linija,20000)) {
            if (!strcmp(dat, linija)) {
               cout << "Datoteka je vec obradjena."
                    << endl;
               system("PAUSE");
               exit(0);
            }
      }
      dato.close();
      ofstream dat1("datoteke.txt", ios_base::app);
      dat1 << dat << endl;
      dat1.close();
}

void Izracun(char dat[], sl uc[]) {
      char linija[20000];
      ifstream tekst(dat);
      while (tekst.getline(linija,20000)) {
            int duljina = strlen(linija);
            for (int i = 0; i < duljina; i++) {
               if (linija[i] >= 'A' &&
                    linija[i] <= 'Z')
                   uc[linija[i]-'A'].broj++;
               if (linija[i] >= 'a' &&
                    linija[i] <= 'z')
                  uc[linija[i]-'a'].broj++;
            }
      }
      tekst.close();
}

void Azuriranje(char dat[],  sl uc[]) {
      ofstream ucest("ucestalost.txt");
      Izracun(dat,uc);
      for (int i = 0; i < 26; i++)
          ucest << uc[i].broj << endl;
      ucest.close();
}

int main () {
      char datoteka[50], linija[20000];
      sl slova[26];
      InitUc(slova);
      Ulaz(datoteka);
      Azuriranje(datoteka, slova);
      ISort(slova);
      for (int i = 0; i<26; i++)
```

```
        cout << slova[i].slovo << "\t" << slova[i].broj <<
endl;
    system ("PAUSE");
    return 0;
}
```

Dakle, navedeni primjer traži naziv datoteke i potom broji pojave svih znakova u datoteci. Prilikom ponovnog pokretanja programa, ukoliko se unese ista datoteka kao pri prethodnom pokretanju, javit će se poruka da je datoteka već obrađena. Ukoliko se unese nova datoteka, tada će se izbrojati broj pojavljivanja pojedinih znakova, zbrojiti s već do sada izbrojenim znakovima tijekom prethodnog pokretanja programa te će se kao takvi ispisati korisniku na ekran. Obrađene datoteke se pohranjuju u datoteku datoteke.txt a broj pojavljivanja pojedinih slova u datoteku ucestalost.txt. Ove dvije datoteke je potrebno ručno obrisati ukoliko želimo krenuti iz samog početka.

Zadaci za vježbu

- Objasnite što je to datoteka.
- Nabrojite biblioteke zaglavlja za rad s datotekama?
- Nabrojite modove (načine) otvaranja datoteke.
- Izradite sljedeći zadatak uz pomoć datoteka: Korisnik unosi 5 brojeva koji se pohranjuju u datoteku. Nakon toga se vrijednosti čitaju iz datoteke i ispisuju na ekran zajedno sa ukupnom sumom svih vrijednosti.

Programski primjeri za laboratorijske vježbe

Zadatak 1.

Izradite program u kojem:

1. Korisnik unosi 5 cijelih brojeva koji se spremaju u datoteku. Nakon toga se brojevi čitaju iz datoteke i rastavljaju na proste faktore što se potom ispisuje na ekran.

2. Se u datoteku pohranjuje 10 nasumičnih brojeva. Nakon toga se sadržaj datoteke čita i ispisuje na ekran, sortira metodom mjehuričastog sortiranja te se sortirani niz brojeva prepisuje preko starog sadržaja datoteke.

3. Korisnik unos 5 riječi koje se pohranjuju u datoteku. Nakon toga se ispisuju svi samoglasnici, njihovi ASCII kodovi i broj pojavljivanja pojedinog samoglasnika. Nakon toga je potrebno sadržaj datoteke prebaciti u polje te sortirati polje tako da svi samoglasnici budu u prvom a svi suglasnici u drugom dijelu polje. Rješenje ovog dijela realizirati pomoću funkcije.

4. Koristeći tekstualne datoteke unijeti 5 studenata sa sljedećim podacima: ime, prezime, prosjek ocjena. Ispisati studente i njihove podatke na ekran. Nakon toga se ti studenti sortiraju s obzirom na ime od manjeg znakovnog niza k većem. Sada se broj znakova u svakom drugom zapisu imena zbraja a u ostalim oduzima. Prethodna suma daje pomak s obzirom na koji je potrebno kriptirati znakove 'a', 'C', 'd', 'H', 'p'. Originalne i kriptirane znakove je potrebno zapisati u datoteku te potom ispisati sadržaj datoteke obrnutim redoslijedom.

5. Korisnik određuje broj željenih unos (svaki unos predstavlja jedan znak). Korisnik može unijeti i brojeve ali i znakove. Uneseni sadržaj je potrebno pohraniti u datoteku. Nakon unosa potrebno je parne brojeve sortirati uzlazno i postaviti ih na početak datoteke, neparne silazno i postaviti ih na kraj datoteke a znakove je potrebno sortirati abecedno (a - z) i smjestiti ih na sredinu datoteke (mogu se koristiti i pomoćne strukture u rješavanju).