Ulazno-izlazni tijekovi u programskom jeziku C++

Kučina, Sven

Sveučilište u Rijeci – Odjel za informatiku

Preddiplomski sveučilišni studij informatike

Sven Kučina

Ulazno-izlazni tijekovi u programskom jeziku C++

Završni rad

Mentor: prof. dr. sc. Maja Matetić

Rijeka, 2021.

Sadržaj

1. Uvod

Ulazno-izlazni tijekovi u programskom jeziku C++ omogućuju C++ programima da čitaju bajtove sa određenih ulaza te da ih ispisuju na određene izlaze. Ulazno-izlazni tijekovi su neophodni za čitanje podataka sa na primjer tipkovnice ili iz datoteke te pri ispisu istih na standardni izlaz ili u datoteku.

Iako možemo koristiti pojmove „ulaz“ i „izlaz“ kolokvijalno te time sasvim intuitivno opisati način rada C++ programa, navedeni pojmovi zapravo imaju tehničku definiciju. Ulaz i i izlaz su zapravo objekti što znači da imaju vlastite članske funkcije i članske varijable. Budući da klase i objekti nisu temelj ovog rada, nećemo ih značajno proučavati.

Kada C++ program čita podatke sa tipkovnice korsiti ulazni tijek *cin*, a kada ispisuje podatke na ekran koristi izlazni tijek *cout*. Prilikom tog procesa, C++ program koristi međuspremnik kao privremenu memoriju za tijek podataka između programa i uređaja. Međutim, pomoću takvih tijekova možemo samo privremeno pohraniti podatke. Jedom kada program završi izvođenje svi podaci će biti izgubljeni. Ako to ne želimo odnosno ako želimo spremiti podatke za buduće korištenje moramo koristiti datoteke.

Kako koristimo *cin* i *cout* tijekove na sličan način C++ program može čitati podatke iz datoteke pomoću objekta *ifstream* što je skraćeno od *input stream* te ispisivati podatke u neku drugu datoteku pomoću objekta *ofstream* što je skraćeno od *output stream*. Na taj način možemo automatizirati unos podataka u program tako da se ne mora uvijek ponovno pisati ulaz za program, već se ulaz jednostavno uzme iz datoteke te ga može koristiti više programa. Također možemo i spremiti izlaz u datoteku što nam omogućuje trajnu pohranu podataka te pristup tim podacima kada nam zatreba. Datoteke su odličan način za čitanje velikih količina podataka. Na taj način smanjujemo opterećenje korisniku u smislu da ne mora pisati čitav ulaz. Sadržaj datoteke ostaje nepromijenjen sve dok ga program ili korisnik ne promijeni.

Kada program uzima ulaz iz datoteke kažemo da čita iz datoteke. Kada program šalje izlaz u datoteku kažemo da program piše u datoteku.

Zadatak završnog rada je teorijski opisati rad za datotečnim tijekovima tipa *fstream,* datotekama, te ilustrirati praktičnu primjenu na primjerima izvedenim u programskom jeziku C++.

2. Osnove korištenja datoteka

Kada C++ program čita ulaz sa tipkovnice on ga čita u potpunosti od početka pa do kraja. Ako želimo ponovno pročitati određeni podatak program mora ponovno pročitati cijelu datoteku ispočetka. Jednako se događa i sa datotekama. Kada imamo ulaz iz datoteke program ga čita od početka pa sve do kraja pri čemu se ne može vratiti i čitati neki dio u datoteci više puta.

Također vrijedi i za ispis u datoteku. Program piše izlaz u izlaznu datoteku od početka datoteke. Nije mu dozvoljeno da se vrati i radi izmjene na izlaznoj datoteci osim ako će dodavati izmjene od početka.

Kako bismo omogućili čitanje ulaza iz datoteke i pisanja izlaza u datoteku moramo program i datoteku povezati sa ulaznim ili izlaznim tijekom. Tijekovi su zapravo objekti odnosno posebna vrsta varijable. Tijekovi *cin* i *cout* su za nas već deklarirani. Ako želimo tijek povezati za datotekom moramo ga prvo deklarirati kao i bilo koju drugu varijablu.

Kako bi deklarirali varijablu preko koje ćemo ulaznu datoteku povezati sa programom deklariramo sljedeće:

```
ifstream ime_varijable;
```

Kako bi deklarirali varijablu preko koje ćemo izlaznu datoteku povezati sa programom deklariramo sljedeće:

```
ofstream ime_varijable;
```

Ifstream i *ofstream* su definirani u biblioteci *fstream* tako da bilo koji program koji se koristi datotečnim tijekovima mora sadržavati sljedeću direktivu:

```
#include <fstream>
```

Kada koristimo *ifstream* i *ofstream* naš program također mora sadržavati sljedeće, uglavnom na početku datoteke:

```
using namespace std;
```

Nakon što smo deklarirali varijable tipa ifstream i ofstream, potrebno je povezati datoteku sa varijablom kako bismo povezali program za datotekom. To se naziva „otvaranje datoteke“. Funkcija *open* nam to omogućava. Na primjer, želimo povezati izlaznu datoteku sa varijablom tijeka. Prvo je deklariramo:

```
ofstream izlazna_dat;
```

Te je zatim otvorimo i navedemo ime datoteke u koju će se spremati izlaz:

```
izlazna_dat.open(„izlaz.txt“)
```

Ako datoteka „izlaz.txt“ ne postoji bit će stvorena automatski. Ako postoji, sve što se prethodno nalazilo u datoteci će se izbrisati i izmijeniti sa novim izlaznim podacima osim ako ne navedemo konstantu *ios:app* definiranu u biblioteci *iostream*. *ios:app* nam omogućuje da na kraj postojeće datoteke dodamo izlaz. Kako bi se podaci dodali na kraj datoteke pišemo sljedeće:

```
izlazna_dat.open(„izlaz.txt“, ios:app);
```

Na isti način bismo otvorili i ulaznu datoteku:

```
ulazna_dat.open(„ulaz.txt“)
```

Kod navođenja imena datoteke bitno je znati razliku između apsolutne i relativne putanje datoteke te je potrebno znati gdje se datoteka nalazi na disku. U navedenom gornjem primjeru, koristi se relativna putanja odnosno podrazumijeva se da se datoteka nalazi u istom direktoriju kao i naš C++ projekt. Ako želimo čitati datoteku koja se ne nalazi u istom direktoriju kao i naš C++ program moramo pisati apsolutnu putanja oblika:

```
ulazna_dat.open(„C:/Users/pero/Documents/ulaz.txt“)
```

Također je bitno napomenuti da je funkcija *open* različita funkcija ovisno o tome otvaramo li ulaznu ili izlaznu datoteku. Kao što smo napomenuli, *ifstream* i *ofstream* su dvije različite klase. Iako su slične, nisu iste. Dakle kada instanciramo objekt tipa *ifstream* ili *ofstream*, taj objekt možem putem točkastog operatora pozivati članske funkcije koje se razlikuju ovisno o tome je li navedeni objekt tipa klase *ifstream* ili *ofstream*. Bit je u tome da je funkcija *open* unutar klase *ifstream* različita od funkcije *open* unutar klase *ofstream* odnosno datoteke se otvaraju na različite načine u ovisnosti o tome radi li se o ulaznoj ili izlaznoj datoteci.

Na kraju svakog programa prilikom kojega smo baratali s datotekama poželjno je i preporuča se da programer manualno „zatvori“ datoteku. Kao što smo datoteku otvorili tako bismo je trebali i zatvoriti. Naime, ako program nepravilno završi svoje izvođenje zbog pogreške, a nismo zatvorili datoteku, može doći do oštećenja datoteke. Također je važno zatvoriti izlaznu datoteku ako će naš program kasnije trebati čitati ulaz iz te datoteke. To sprječavamo sljedećom linijom koda te se preporuča dodavanje te linije uvijek kada završimo rad sa željenom datotekom.

```
izlazna_dat.close();
```

2.1. Korištenje datoteka kao ulaz

Kada smo deklarirali i povezali varijablu tipa ifstream sa željenom datotekom htjet ćemo čitati i iste datoteke te zapisati određene podatke u varijable kako bismo mogli baratati tim podacima unutar programa.

Kod *cin* se koristi ulaz sa tipkovnice te kada bismo htjeli omogoćiti korisniku da unese nešto bismo pisali:

```
cin>>ime_varijable;
```

Time bismo spremili što god je korisnik unio u varijablu. Kada koristimo ulaz iz datoteke korisnik ne utječe na vrijednost varijable, već se podatak čita iz datoteke. Pri tome se neće prikazati nikakvo sučelje korisniku koje bi se inače prikazalo. Operator „>>“ se naziva operator ekstrakcije. Sintaksa za čitanje podataka iz datoteke (koju smo prethodno otvorili) i spremanje u varijablu:

```
ime_datoteke>>ime_varijable;
```

Program može čitati i spremiti više varijabli odjednom na sljedeći način ako su varijable tipa integer:

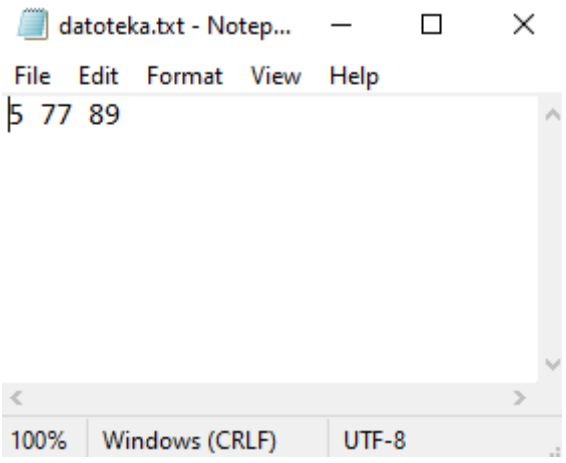
```
ime_datoteke>>ime_varijable_1>>ime_varijable_2>>ime_varijable_3
```

Predstavimo prvi primjer programa koji koristi ulaz iz datoteke. Kod programa u C++ jeziku je sljedeći:

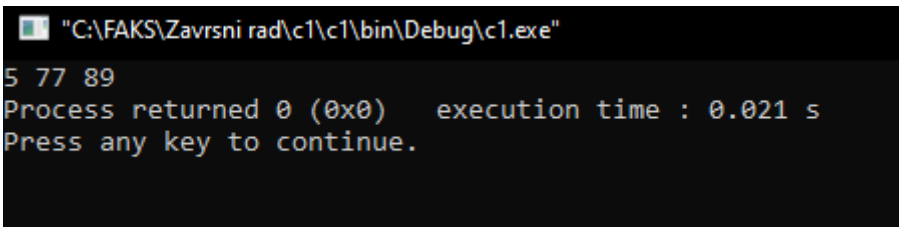
main.cpp

```cpp
#include <iostream>
#include <fstream>

using namespace std;

int main()
{
    int x,y,z;

    ifstream ulazna_dat;
    ulazna_dat.open("datoteka.txt");
    ulazna_dat>>x>>y>>z;

    cout<<x<<" "<<y<<" "<<z;
    ulazna_dat.close();
    return 0;
}
```

Slika 1 - Korištenje datoteke za ulaz



Slika 2 - Sadržaj datoteke "datoteka.txt"



Slika 3 - Izlaz programa

Pojasnimo ukratko što se događa u kodu. Za početak je bitno uključiti biblioteke *iostream (*za osnovne ulazno-izlazne tijekove kako bismo omogućili ispis na ekran) i *fstream* (za baratanje sa datotekama):

```
#include <iostream>
#include <fstream>
```

Unutar funkcije *main* deklariramo tri varijable tipa *integer* i deklariramo objekt *ulazna_dat* klase *fstream* te zatim povezujemo objekt (posebna varijabla) odnosno program sa datotekom na disku, specificirajući njenu putanju.

```
int x,y,z;
ifstream ulazna_dat;
ulazna_dat.open("datoteka.txt");
```

Zatim iz datoteke čitamo tri prirodna broja putem operatora ekstrakcije. Program sprema ta tri broja u tri varijable tipa *integer*. Unutar datoteke brojevi su odvojeni razmacima što program shvaća da su tri različite varijable. Ispisujemo rezultat na ekran i zatvaramo datoteku:

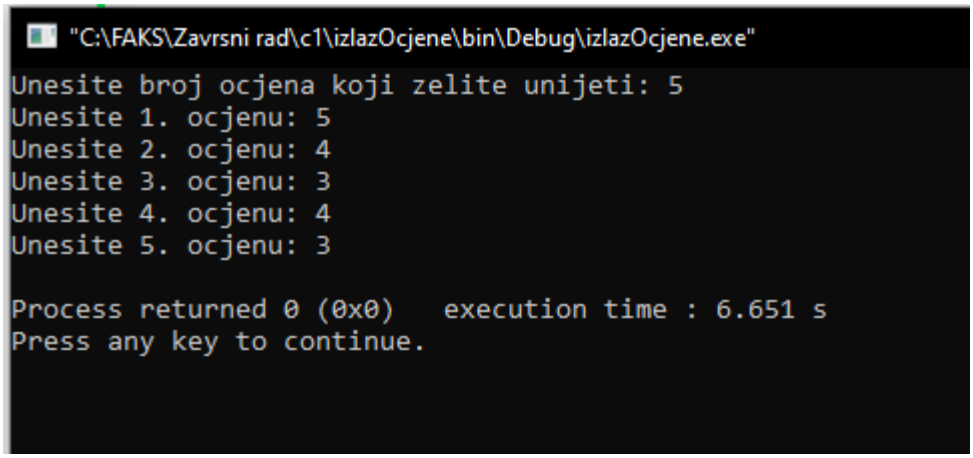
```
ulazna_dat>>x>>y>>z;
cout<<x<<" "<<y<<" "<<z;
ulazna_dat.close();
```

2.2. Korištenje datoteka za izlaz

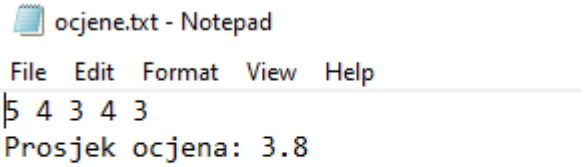
Želimo na primjer spremiti ocjene studenta u izlaznu datoteku i izračunati njihov prosjek. Korisniku ćemo dati mogućnost da unese koliko god ocjena želi i zatim ćemo zapisati te ocjene i njihov prosjek u vanjsku datoteku.

```cpp
#include <iostream>
#include <fstream>

using namespace std;

int main()
{
    ofstream izlaznaDat;
    izlaznaDat.open("ocjene.txt");
    int n, ocjena;
    double suma=0;
    double prosjek;
    cout<<"Unesite broj ocjena koji zelite unijeti: ";
    cin>>n;

    for (int i=0; i<n; i++){
        cout<<"Unesite "<<i+1<<". ocjenu: ";
        cin>>ocjena;
        izlaznaDat<<ocjena<<" ";
        suma=suma+ocjena;
    }
    prosjek=suma/n;
    izlaznaDat<<endl;
    izlaznaDat<<"Prosjek ocjena: "<<prosjek;

    izlaznaDat.close();
    return 0;
}
```

Slika 4 - Primjer korištenja datoteka za izlaz



Slika 5 - Omogućavanje unosa korisniku



Slika 6 - Izlazna datoteka "ocjene.txt"

Pojasnimo kod. Kao i inače, uključujemo biblioteke *iostream* i *fstream*, deklariramo objekt odnosno varijablu tipa *ofstream,* te zatim povezujemo program sa datotekom „ocjene.txt“ putem članske funkcije *open.* Deklariramo varijable potrebne za omogućavanje unosa korisniku, ispis ocjena u datoteku i izračunavanje prosjeka.

Omogućujemo korisniku unos broja ocjena kojima želi izračunati prosjek te ih ispisati u izlaznu datoteku:

```
cout<<"Unesite broj ocjena koji zelite unijeti: ";
cin>>n;
```

Koristimo *for* petlju koja omogućuje korisniku unos ocjena u količini koju je korisnik definirao. Kada korisnik unese jednu ocjenu, ona se odmah zapisuje u izlaznu datoteku i računa se suma ocjena koja nam treba za izračunati prosjek:

```
for (int i=0; i<n; i++){
        cout<<"Unesite "<<i+1<<". ocjenu: ";
        cin>>ocjena;
        izlaznaDat<<ocjena<<" ";
        suma=suma+ocjena;
}
```

Nakon što je korisnik unio željene ocjene, računamo prosjek ocjena te ga zapisujemo u izlaznu datoteku. Naposlijetku, kada smo gotovi sa radom nad datotekom, zatvaramo datoteku:

```
prosjek=suma/n;
izlaznaDat<<endl;
izlaznaDat<<"Prosjek ocjena: "<<prosjek;
izlaznaDat.close();
return 0;
```

3. Dodatne tehnike upravljanja datotekama

3.1. Provjera povezanosti programa i datoteke

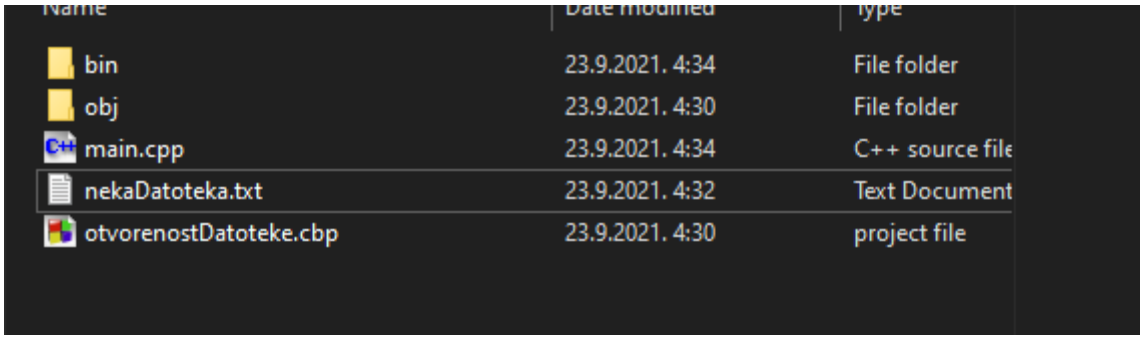
Kod otvaranja datoteke odnosno povezivanja programa s datotekom može doći do određenih greški. Najčešće greške su nepostojanje datoteke na disku i pogrešno naveden naziv datoteke. Ako ne definiramo provjeru greški program će svejedno nastaviti raditi čak iako postoje greške te može napraviti nešto neočekivano. Zato radimo provjeru greški.

Možemo koristiti člansku funkciju *fail* za provjeru otvorenosti datoteke. Funkcija *fail* je tipa *bool* te vraća *true* ako je zaista došlo do greške ili *false* ako nema greške odnosno datoteka je uspješno otvorena.

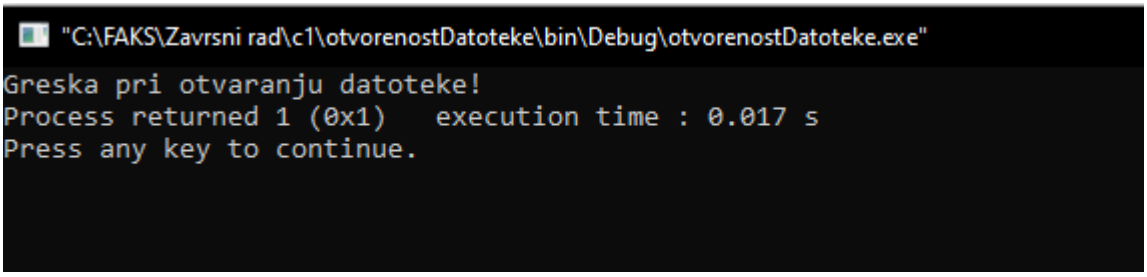
Pogledajmo primjer.

```
1  #include <iostream>
2  #include <fstream>
3  #include <cstdlib>
4
5  using namespace std;
6
7  int main()
8  {
9      ifstream dat;
10     dat.open("nekaDatotek.txt");
11
12     if (dat.fail()){
13         cout<<"Greska pri otvaranju datoteke!";
14         exit(1);
15     }
16     else
17         cout<<"Datoteka je uspjesno povezana sa programom!";
18     return 0;
19 }
20
```

Slika 7 - Primjer provjere za greške prilikom otvaranja datoteke



Slika 8 - Naziv datoteke koju pokušavamo povezati s programom



Slika 9 - Poruka o greški

Pojasnimo kod. Deklariramo varijablu tipa *ifstream* i pokušavamo je povezati s datotekom naziva „nekaDatotek.txt“. To je krivi naziv datoteke jer se datoteka koju smo pokušali povezati s programom naziva „nekaDatoteka.txt“.

Putem *if* izjave i funkcije *fail* zatim provjeravamo da li smo uspješno otvorili datoteku. Ako nismo odnosno ako je *boolean* vrijednost koju vraća funkcija *fail* istinita, ispisuje se poruka o greški, inače, ispisuje se poruka o uspjehu. Funkcija *exit(1)* odmah završava izvođenje programa jer nema smisla raditi na datoteci koju nismo uspješno povezali s našim programom. exit() je definirana u biblioteci *cstdlib* te smo je morali uključiti na početku koda.

```
if (dat.fail()){
        cout<<"Greska pri otvaranju datoteke!";
        exit(1);
}
else
        cout<<"Datoteka je uspjesno povezana sa programom!";
return 0;
```

Također možemo koristiti funkciju *is_open()* za provjeru povezanosti programa i datoteke. Funkcija *is_open()* je gotovo jednaka funkciji *fail()*. Razlika je u tome što funkcija *is_open()* vraća vrijednost *true* kada je datoteka otvorena.

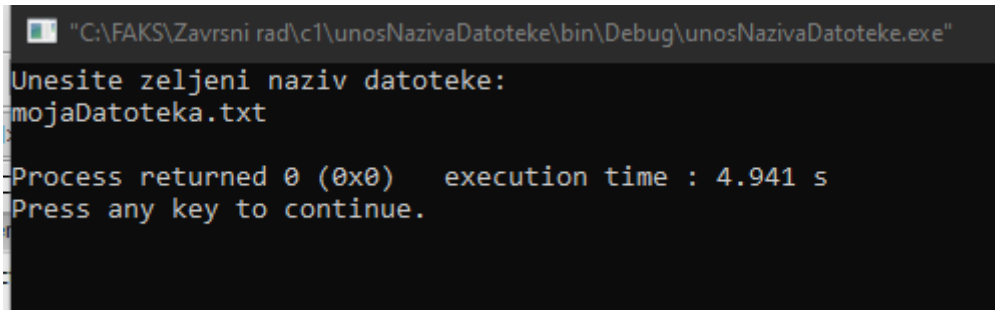
```
if (dat.is_open())
        cout<<"Datoteka je uspjesno povezana sa programom!";
else{
        cout<<"Greska pri otvaranju datoteke!";
        exit(1);
}

return 0;
```

3.2. Omogućavanje korisniku unosa imena datoteke

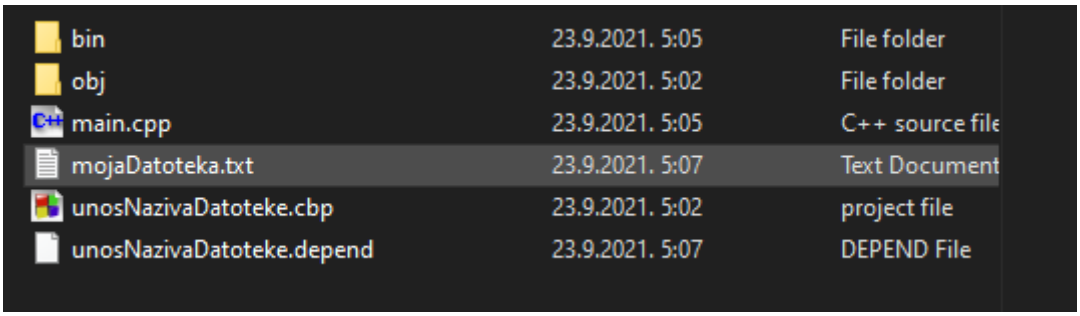
Korisniku ćemo omogućiti unos naziva datoteke na sljedeći način:

```
1   #include <iostream>
2   #include <fstream>
3
4   using namespace std;
5
6   int main()
7   {
8       char naziv[20];
9
10      cout << "Unesite zeljeni naziv datoteke: " << endl;
11      cin>>naziv;
12
13      ofstream dat;
14      dat.open(naziv);
15      dat.close();
16      return 0;
17  }
18
```

Slika 10 - Omogućavanje korisniku unos naziva datoteke



Slika 11 - Korisnik unosi željeni naziv



Slika 12 - Sadržaj projektnog direktorija

Deklariramo prvo varijablu tipa *char* u koju stane 20 znakova. Zatim preko *cin* tijeka omogućujemo korisniku unos naziva datoteke i stvaramo je na disku.

3.3. Provjera kraja datoteke

Kada pišemo program koji čita ulaz iz datoteke vrlo često ćemo htjeti da naš program čita sve podatke unutar datoteke. Ako želimo (kao u prijašnjem primjeru) odrediti prosjek određenih brojeva, ali u ovom slučaju da program čita brojeve iz datoteke, a ne preko unosa korisnika, moramo omogućiti da program pročita sve brojeve u datoteci dok ih sve ne pročita.

Općenito se koriste dvije metode za testiranje kraja datoteke. Metoda sa operatorom ekstrakcije i metoda sa funkcijom *eof.* Funkcija *eof* provjerava kraj datoteke i skraćeno je od *end of file* na engleskom.

Metodu sa operatorom ekstrakcije koristimo kada želimo čitati numeričke podatke:

```
while (ulaz>>sljedeci)
```

Metodu sa funkcijom *eof* koristimo kada čitamo tekstualne podatke.

```
while (!ulaz.eof())
```

3.4. Prosljeđivanje varijabli datotečnih tijekova funkcijama

Objekte tipa *ifstream* i *ofstream* možemo prosljeđivati funkcijama. Na taj način možemo rastaviti i modularizirati naš rad sa datotekama tako da za određenu radnju nad datotekom postoji odgovarajuća funkcija. Umjesto da sve pišemo unutar *main* funkcije (što se smatra neurednim) možemo imati različite dijelove odgovorne za različite procese. Time se smanjuje neurednost koda i olakšava čitanje ostalim programerima koji proučavaju kod.

Tijekovi koje definiramo biti će parametri funkcija s time da parametar za tijek mora biti referentan što znači da mu prosljeđujemo adresu objekta te svaka operacija koja se vrši nad varijablom tijeka unutar funkcije bit će izvršena nad samom datotekom.

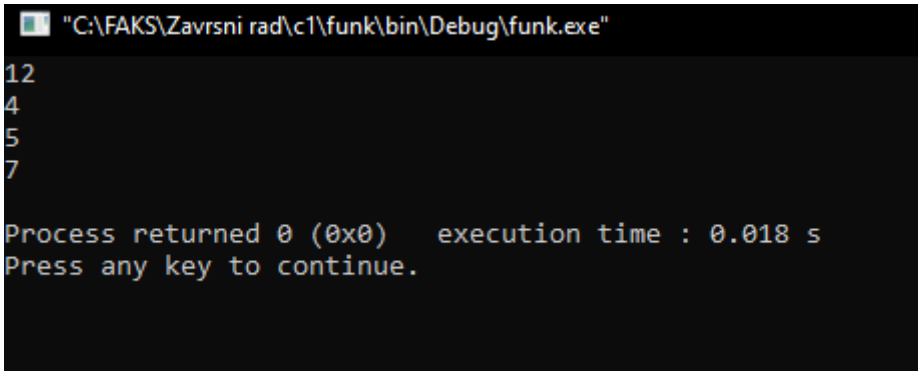
Objasnimo sljedeći primjer. U kodu stvaramo tri funkcije kojima ćemo proslijediti varijablu tijeka. Funkcija *otvoriDatoteku(ifstream& f, string s)* je tipa *bool* i ona vraća vrijednost *true* ako je datoteka uspješno otvorena. Unutar funkcije otvaramo datoteku na uobičajen način s uključenom provjerom za greške. Budući da je parametar tijeka referentan sve operacije izvršene nad varijablom *f* unutar funkcije *otvoriDatoteku()* izvršit će se nad varijablom tijeka *ulaz* unutar *main* funkcije i naposlijetku nad samom datotekom „brojevi.txt“.

Funkcija *ispisi(ifstream& f)* je tipa *void* te ne vraća nikakvu vrijednost, već služi samo za ispis sadržaja datoteke „brojevi.txt“ na ekran. Primijetimo pritom da se koristi metoda sa operatorom ekstrakcije za čitanje podataka iz datoteke.

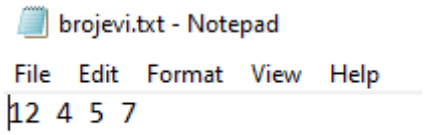
Funkcija *zatvoriDatoteku(ifstream& f)* zatvara datoteku.

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

bool otvoriDatoteku(ifstream&, string);
void zatvoriDatoteku(ifstream&);
void ispisi(ifstream&);

int main()
{

    bool jeOtvorena;
    ifstream ulaz;

    jeOtvorena = otvoriDatoteku(ulaz, "brojevi.txt");

    if(jeOtvorena){
        ispisi(ulaz);
        zatvoriDatoteku(ulaz);
    }
    else
        cout<<"Greska"<<endl;
    return 0;
}

bool otvoriDatoteku(ifstream& f, string s){
    f.open(s);
    if(f.is_open())
        return true;
    else
        return false;
}

void ispisi(ifstream& f){
    int a;
    while(f>>a)
        cout<<a<<endl;
}

void zatvoriDatoteku(ifstream& f){
    f.close();
}
```

Slika 13 - Korištenje funkcija za rad sa datotekom



Slika 14 - Ispis sadržaja datoteke "brojevi.txt" na ekran



Slika 15 - Sadržaj datoteke "brojevi.txt"

4. Oblikovanje izlaza datoteke - zastavice

Izlaz u datoteku možemo formatirati na razne načine. Želimo li odrediti razmak između određenih dijelova ispisa, postaviti broj decimalnih mjesta ili prikazati pozitivne brojeve sa znakom „+“ ispred broja.

Kao što možemo formatirati izlaz na ekran tako možemo formatirati i izlaz u datoteku. To radimo pomoću takozvanih zastavica. Zastavice postavljamo funkcijom *setf* što je na engleskom skraćeno od *set flags* što znači „postavi zastavicu“. Zastavica, ovisno o argumentu koji joj postavimo, govori računalu kako da ispiše izlaz na ekran ili u datoteku na određeni način.

Na primjer sljedeća zastavica govori računalu da ispisuje brojeve tipa *double* kako ih uobičajeno pišemo. To znači da ako postavimo tu zastavico te pokušamo ispisati broj 3 tipa *double* ispisat će se kao 3.000000. Zastavicu postavljamo ovako:

```
ime_datoteke.setf(ios::fixed);
```

Ako želimo postaviti broj decimalnih mjesta definiramo sljedeću zastavicu:

```
ime_datoteke.precision(broj_decimalnih_mjesta);
```

Želimo li prikazati plus znak ispred svakog pozitivnog broja pišemo sljedeće:

```
ime_datoteke.setf(ios::showpos);
```

Minus znakovi će biti postavljeni ispred svakog negativnog broja bez postavljanja bilo kakvih zastavica.

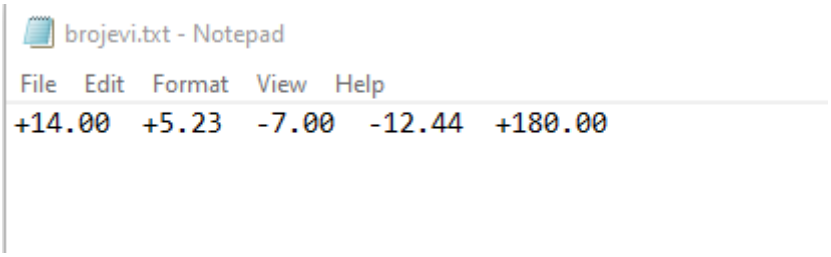
Pogledajmo sljedeći primjer.

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

int main()
{
    double a[5];
    ofstream datoteka;
    datoteka.open("brojevi.txt");
    datoteka.setf(ios::fixed);
    datoteka.precision(2);
    datoteka.setf(ios::showpos);

    cout<<"Unesite 5 brojeva: "<<endl;
    for (int i=0; i<5; i++){
        cin>>a[i];
        datoteka<<a[i]<<"  ";
    }

    datoteka.close();
    return 0;
}
```

Slika 16 - Primjer formatiranja izlaza

```
"C:\FAKS\Zavrsni rad\c1\formatiranje\bin\Debug\formatiranje.exe"
Unesite 5 brojeva:
14
5.23234
-7
-12.4423
180

Process returned 0 (0x0)   execution time : 20.354 s
Press any key to continue.
```

Slika 17 - Unos brojeva koji će biti ispisanu u izlaznu datoteku



Slika 18 - Formatirani zapis u datoteci

Koristimo niz kako bi korisniku omogućili unos od pet brojeva. Prije samog unosa brojeva i njihovog ispisa u datoteku, određujemo oblikovanje ispisa odnosno kako će se brojevi zapisati u datoteku. Određujemo da će se svi brojevi tipa *double* prikazivati sa dvije decimalne točke te ako je broj pozitivan dodat ćemo znak plus ispred broja. To smo omogućili sa sljedeće tri linije:

```
datoteka.setf(ios::fixed);
datoteka.precision(2);
datoteka.setf(ios::showpos);
```

5. Zaključak

Datoteke koristimo kada želimo spremiti izlaz, ponovno ga koristiti ili automatizirati ulaz kako korisnik ne bi morao nepotrebno unositi unose velikih količina.

Tijekove tipa *ifstream* i *ofstream* možemo povezati sa datotekom pozivajući člansku funkciju *open.* Radi izbjegavanja nepredvidivih događaja i grešaka u programu bitno je provjeriti ako je datoteka pravilno povezana sa programom. Kada god završimo rad sa datotekama potrebno ih je zatvoriti.

Klasa je tip čije su varijable objekti. Objekt je varijabla koja ima vlastite članske funkcije i članske varijable. *ifstream* i *ofstream* su klase.

Korištenjem datoteka u programiranju imamo način trajne pohrane podataka. Pohrana podataka je bitna u slučaju da želimo te iste podatke ponovno koristiti u budućnosti. Datoteke su također neophodne za čitanje velikih količina podataka te kako ih korisnik ne bi morao sam unositi. Kada program završi svoje izvođenje, svi podaci utipkani putem tipkovnice bit će izbrisani. Korištenjem datoteka možemo to izbjeći.

Sadržaj datoteke ostaje nepromijenjen sve dok ga korisnik ili program ne promijeni. Ulazna datoteka se može koristiti neograničen broj puta za različite programe bez potrebe da se podaci zasebno unose.

Danas gotovo svaki program / aplikacija / alat koristi datoteke za pohranu podataka. Web stranice također rade slično. Jedina razlika je u tome što program (preglednik koji koristimo) zahtjeva Internet vezu i putanju do željene datoteke na poslužitelju.

Sažetak

Za rad sa datotekama najprije je potrebno uključiti biblioteku *fstream* u program:

```
#include <fstream>
```

Datoteke s kojima radimo moramo najprije deklarirati i povezati sa programom. To radimo na sljedeći način:

```
ifstream ulaz;
ulaz.open(„naziv_datoteke“);
ofstream izlaz;
izlaz.open(„naziv_datoteke“);
```

Želimo li provjeriti ako je datoteka uspješno otvorena pišemo sljedeće:

```
if (ulaz.is_open()){
        //rad nad datotekom
}
else
        cout<<“Greska“;
```

ili

```
if (ulaz.fail())
        cout<<“Greska“;
else{
        //rad nad datotekom
}
```

Možemo koristiti funkcije pri radu s datotekama kako bi poboljšali čitljivost i povećali modularnost koda. Kod definiranja i prototipiranja funkcija bitno je proslijediti varijablu tijeka kao referentni parametar tako da se operacije vršene unutar funkcije također izvrše na željenoj varijabli tijeka odnosno željenoj datoteci.

```
nekaFunkcija(ifstream& f);
```

Pomoću zastavica možemo oblikovati izlaz u datoteku. Sintaksa je sljedeća:

```
datoteka.setf(željena_zastavica);
```

Ključne riječi: programski jezik C++, ulazno-izlazni tijekovi, datoteka, čitanje datoteke, pisanje u datoteku, zastavice

Popis slika

Sve slike su osobno napravljene koristeći alat *Snipping Tool.*

Literatura

Knjige

1. Walter Savitch: Problem Solving in C++, Pearson Publishing, 2006
2. Maja Matetić: Skripta uz predmet Programiranje 1 (digitalna skripta), Odjel za informatiku, Sveučilište u Rijeci, Rijeka 2008.

Izvori sa interneta

1. https://www.geeksforgeeks.org/
2. https://www.w3schools.com/