# Рад са датотекама

## 5.1. Основни задаци за писање и читање из датотеке

### Задатак: Учешљавање студената два тока

Сви студенти прве године факултета распоређени су у два тока, а њихови подаци смештени су у две текстуалне датотеке. У свакој датотеци налази се листа студената сортирана лексикографски по презимену, а затим по имену. Написати програм који спаја ове две листе учешљавањем у једну јединствену сортирану листу и резултат уписује у нову датотеку.

#### Опис улаза

Са стандардног улаза задаје се прво назив датотеке која садржи листу студената првог тока и број студената на том току, затим назив датотеке и број студената на другом току. Подаци о сваком студенту дати су у новом реду, где сваки ред садржи редом презиме, име и број индекса, раздвојене размаком. Листе у оба фајла су већ сортиране лексикографски по презимену, а затим по имену.

#### Опис излаза

Као резултат извршавања програма датотека са називом `studentiLista.txt` садржи јединствену, лексикографски сортирану листу свих студената из оба фајла. У случају грешке током извршавања програма на стандардни излаз исписати `-1` и прекинути програм.

#### Пример

**Улаз**

```
prviTok.txt 3
drugiTok.txt 4
```

**Садржај датотеке *prviTok.txt***

```
Ilic Stefan 12/2025
Petrovic Nikola 20/2025
Zoric Milica 4/2025
```

**Садржај датотеке *drugiTok.txt***

```
Jovanovic Marko 7/2025
Petrovic Ana 13/2025
Stankovic Ivana 9/2025
Tomic Milica 17/2025
```

**Излаз, садржај датотеке *studentiLista.txt***

```
Ilic Stefan 12/2025
Jovanovic Marko 7/2025
Petrovic Ana 13/2025
Petrovic Nikola 20/2025
Stankovic Ivana 9/2025
Tomic Milica 17/2025
Zoric Milica 4/2025
```

**Решење**

Како су обе листе већ сортиране у траженом поретку, да бисмо одредили првог студента нове листе довољно је да упоредимо прва два студента са оба тока. Водећи се истим принципом, даље можемо да упоређујемо редом по једног студента са оба тока, при чему прелазимо на наредног само у оној датотеци из које смо преписали податак. Када дођемо до краја било које од две листе, потребно је само преписати редом преостале податке из друге листе.

Податке о једном студенту можемо представити коришћењем структуре, а поређење два студента можемо имплементирати дефинисањем помоћне функције. Учитавање комплетног садржаја две датотеке у колекције типа `vector<Student>` можемо прескочити ради уштеде меморије.

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

using namespace std;

struct Student {
    string prezime;
    string ime;
    string indeks;
};

bool manji_ili_jednak(Student s1, Student s2) {
    if (s1.prezime < s2.prezime) return true;
```

```
    if (s1.prezime > s2.prezime) return false;

    if (s1.ime < s2.ime) return true;
    if (s1.ime > s2.ime) return false;

    return s1.indeks <= s2.indeks;
}

int main() {

    string fajl1, fajl2;
    int n, m;

    cin >> fajl1 >> n;
    cin >> fajl2 >> m;

    ifstream tok1(fajl1);
    ifstream tok2(fajl2);
    ofstream tok3("studentiLista.txt");

    if (!tok1.is_open() || !tok2.is_open()) {
        cout << "-1'" << endl;
        return 0;
    }

    Student s1, s2;
    int i = 0, j = 0;

    tok1 >> s1.prezime >> s1.ime >> s1.indeks;
    tok2 >> s2.prezime >> s2.ime >> s2.indeks;

    while (i < n && j < m) {
        if (manji_ili_jednak(s1, s2)) {
            tok3 << s1.prezime << "␣" << s1.ime << "␣" << s1.indeks << endl;
            i++;
            if (i < n)
                tok1 >> s1.prezime >> s1.ime >> s1.indeks;
        } else {
            tok3 << s2.prezime << "␣" << s2.ime << "␣" << s2.indeks << endl;
            j++;
            if (j < m)
                tok2 >> s2.prezime >> s2.ime >> s2.indeks;
        }
```

```
    }

    while (i < n) {
        tok3 << s1.prezime << "␣" << s1.ime << "␣" << s1.indeks << endl;
        i++;
        if (i < n)
            tok1 >> s1.prezime >> s1.ime >> s1.indeks;
    }

    while (j < m) {
        tok3 << s2.prezime << "␣" << s2.ime << "␣" << s2.indeks << endl;
        j++;
        if (j < m)
            tok2 >> s2.prezime >> s2.ime >> s2.indeks;
    }

    return 0;
}
```

## Задатак: Подела студената на токове

Познат је списак студената који су уписали један предмет. Потребно је разврстати их у два тока, тако да студенти са парним бројем индекса буду распоређени у први ток, а са непарним у други. Студенти у оквиру једног тока треба да буду сортирани лексикографски по презимену, а затим по имену.

### Опис улаза

Са стандардног улаза учитава се назив датотеке која садржи листу студената који су уписали предмет. Подаци о сваком студенту дати су у новом реду, где сваки ред садржи редом презиме, име и број индекса, раздвојене размаком. Студенти у улазном фајлу су већ сортирани лексикографски по презимену, а затим по имену.

### Опис излаза

Као резултат извршавања програма датотека са називом `prviTok.txt` садржи лексикографски сортирану листу свих студената који припадају првом току, а датотека `drugiTok.txt` сортирану листу студената другог тока. У случају грешке током извршавања програма исписати `-1` и прекинути програм.

### Пример

**Улаз**

`studentiLista.txt`

**Садржај датотеке *studentiLista.txt***

```
Ilic Stefan 12/2025
Jovanovic Marko 7/2025
Petrovic Ana 13/2025
Petrovic Nikola 20/2025
Stankovic Ivana 9/2025
Tomic Milica 17/2025
Zoric Milica 4/2025
```

**Излаз, садржај датотеке *prviTok.txt***

```
Ilic Stefan 12/2025
Petrovic Nikola 20/2025
Zoric Milica 4/2025
```

**Излаз, садржај датотеке *drugiTok.txt***

```
Jovanovic Marko 7/2025
Petrovic Ana 13/2025
Stankovic Ivana 9/2025
Tomic Milica 17/2025
```

**Решење**

Како је листа свих студената већ сортирана, нема потребе додатно вршити сортирање студената по токовима, већ је довољно једном проћи кроз листу и, уколико је број индекса тренутног студента дељив са два, додати га у датотеку која садржи студенте првог тока. Иначе, потребно је додати га у датотеку у којој се налазе студенти другог тока.

Пошто број студената није унапред познат, морамо обрађивати датотеку линију по линију све док у њој има неког садржаја. Учитавање једне линије из датотеке може се остварити већ поменутом функцијом `getline`.

Податке о једном студенту можемо поново представити коришћењем структуре, при чему уместо поља `indeks` можемо имати два нова поља која ће редом чувати број индекса и годину уписа (ради лакше провере да ли је број индекса дељив са два).

```
#include<iostream>
#include<fstream>
#include<string>
#include<sstream>
```

```
using namespace std;

struct Student {
        string prezime;
        string ime;
        int brojIndeksa;
        int godinaUpisa;
};

int main(){

        string fajl;
        cin >> fajl;

        ifstream studenti(fajl);

        string prviTok = "prviTok.txt", drugiTok = "drugiTok.txt";
        ofstream tok1(prviTok);
        ofstream tok2(drugiTok);

        if(!studenti.is_open()){
                cout << -1 << endl;
                return 0;
        }

        Student s;
        string linija;

        while(getline(studenti, linija)){
                char crtica;

                stringstream student(linija);
                student >> s.prezime >> s.ime >> s.brojIndeksa >>
                   crtica >> s.godinaUpisa;
                if(s.brojIndeksa % 2){
                        tok2 << s.prezime << "␣" << s.ime << "␣"
                           << s.brojIndeksa << "/" <<
                           s.godinaUpisa << "\n";
                }else{
                        tok1 << s.prezime << "␣" << s.ime << "␣"
                           << s.brojIndeksa << "/" <<
                           s.godinaUpisa << "\n";
                }
```

```
        }

        return 0;
}


}
```

## Задатак: Распоређивање студената у групе

Познат је списак студената који су уписали један предмет. Потребно је разврстати их у $n$ група, тако да се студенти чији бројеви индекса дају исти остатак при дељењу са $n$ нађу у истој групи. Студенти у оквиру једне групе треба да буду сортирани лексикографски по презимену, а затим по имену.

### Опис улаза

Са стандардног улаза учитава се назив датотеке која садржи листу студената који су уписали предмет и број група $n$. Подаци о сваком студенту дати су у новом реду, где сваки ред садржи редом презиме, име и број индекса, раздвојене размаком. Листа студената у улазном фајлу је већ сортирана лексикографски по презимену, а затим по имену.

### Опис излаза

Као резултат извршавања програма датотека са називом `studentiGrupa_i.txt` садржи лексикографски сортирану листу свих студената који припадају групи `i` (где је `i` остатак пре дељењу њиховог броја индекса са $n$). У случају грешке током извршавања програма на стандардни излаз исписати `-1` и прекинути програм.

### Пример

#### Улаз

```
UP.txt 3
```

#### Садржај датотеке *UP.txt*

```
Ilic Stefan 12/2025
Jovanovic Marko 7/2025
Petrovic Ana 13/2025
Petrovic Nikola 20/2025
Stankovic Ivana 9/2025
```

```
Tomic Milica 17/2025
Zoric Milica 4/2025
```

**Излаз, садржај датотеке *studentiGrupa_ 0.txt***

```
Ilic Stefan 12/2025
Stankovic Ivana 9/2025
```

**Излаз, садржај датотеке *studentiGrupa_ 1.txt***

```
Jovanovic Marko 7/2025
Petrovic Ana 13/2025
Zoric Milica 4/2025
```

**Излаз, садржај датотеке *studentiGrupa_ 2.txt***

```
Petrovic Nikola 20/2025
Tomic Milica 17/2025
```

**Решење**

Једно могуће решење подразумевало би да за сваки од могућих остатака $i$ при дељењу бројем $n$ прођемо кроз целу листу студената и издвојимо у фајл `studentiGrupa_i.txt` оне за које је остатак при дељењу њиховог броја индекса са $n$ баш $i$.

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

using namespace std;

struct Student {
        string prezime;
        string ime;
        int brojIndeksa;
        int godinaUpisa;
};

int main(){

        string fajl;
        int n;
        cin >> fajl >> n;

        Student s;
        string linija;
```

```
        for(int i = 0; i < n; i++){

                ifstream studenti(fajl);

                if(!studenti.is_open()){
                        cout << -1 << endl;
                        return 0;
                }

                ofstream grupa("studentiGrupa_" + to_string(i) +
                   ".txt");

                while(getline(studenti, linija)){
                        char crtica;

                        stringstream student(linija);
                        student >> s.prezime >> s.ime >>
                           s.brojIndeksa >> crtica >>
                           s.godinaUpisa;
                        if(s.brojIndeksa % n == i){
                                grupa << s.prezime << "␣" << s.ime
                                   << "␣" << s.brojIndeksa << "/"
                                   << s.godinaUpisa << "\n";
                        }
                }
        }

        return 0;
}
```

Друго могуће решење је да најпре направимо низ у коме ћемо чувати излазне токове (`ofstream`) за сваки од фајлова `studentiGrupa_i.txt`. На тај начин, не морамо више пута да пролазимо кроз листу студената, већ у једном пролазу можемо да разврстамо студенте уписујући их у одговарајући фајл.

```
#include<iostream>
#include<fstream>
#include<sstream>
#include<vector>

using namespace std;

struct Student {
```

```
        string prezime;
        string ime;
        int brojIndeksa;
        int godinaUpisa;
};

int main(){

        string fajl;
        int n;
        cin >> fajl >> n;

        vector<ofstream> grupe(n);
        for(int i = 0; i < n; i++)
                grupe[i] = ofstream("studentiGrupa_" +
                   to_string(i) + ".txt");

        Student s;
        string linija;

        ifstream studenti(fajl);

        if(!studenti.is_open()){
                cout << -1 << endl;
                return 0;
        }

        while(getline(studenti, linija)){
                char crtica;

                stringstream student(linija);
                student >> s.prezime >> s.ime >> s.brojIndeksa >>
                   crtica >> s.godinaUpisa;

                int ostatak = s.brojIndeksa % n;

                grupe[ostatak] << s.prezime << "␣" << s.ime << "␣"
                   << s.brojIndeksa << "/" << s.godinaUpisa <<
                   "\n";

        }
```

```
        return 0;
}
```

## Задатак: Аутоматски прегледач испита

Сваки студент на испиту креира директоријум са својим именом, у ком се налазе датотеке са задацима. За сваки од 5 задатака постоји тачно по једна датотека, именована бројем задатка, са екстензијом `.cpp`. Радови свих студената се у овом формату прослеђују аутоматском прегледачу на оцењивање.

Резултат прегледача је једна `CSV` датотека која садржи по један ред за сваки задатак свих студената и број тестова које такво решење пролази, а вредности су раздвојене размаком.

Написати програм који форматира улазну `CSV` датотеку, односно групише резултате по студентима, тако да нова `CSV` датотека садржи по један ред за сваког студента. Поени за један задатак једнаки су броју тестова који пролазе, али само уколико то решење пролази све елеиминационе тестове. У супротном, број поена је 0, без обзира на број тестова.

### Опис улаза

Улазна датотека под називом `rezultati.csv` садржи редове у следећем формату:

```
/home/januar1/radovi/UP_Ime_Prezime_indeks 1.cpp 1 8
```

Прва колона представља име директоријума студента, затим следи име датотеке са решењем задатка, а након тога ознака (`0` или `1`) да ли задатак пролази све елиминационе тестове и на крају укупан број тестова који пролазе. За сваког студента улазна датотека садржи тачно 5 редова, по један за сваки задатак. Редови за различите студенте се налазе један за другим и листа не мора бити сортирана.

### Опис излаза

Излазна `CSV` датотека садржи форматиране податке, по један ред за сваког студента, прво име директоријума, након чега следе његови поени, редом по задацима.

### Пример

**Улаз, садржај датотеке *rezultati.csv***

```
/home/januar1/radovi/UP_Nikola_Petrovic_119 1.cpp 1 9
/home/januar1/radovi/UP_Ana_Pavlovic_110 1.cpp 1 10
/home/januar1/radovi/UP_Nikola_Petrovic_119 2.cpp 0 5
/home/januar1/radovi/UP_Ana_Pavlovic_110 2.cpp 1 9
/home/januar1/radovi/UP_Nikola_Petrovic_119 3.cpp 1 8
```

```
/home/januar1/radovi/UP_Ana_Pavlovic_110 3.cpp 0 5
/home/januar1/radovi/UP_Nikola_Petrovic_119 4.cpp 1 10
/home/januar1/radovi/UP_Ana_Pavlovic_110 4.cpp 1 8
/home/januar1/radovi/UP_Nikola_Petrovic_119 5.cpp 0 3
/home/januar1/radovi/UP_Ana_Pavlovic_110 5.cpp 1 7
```

**Излаз, садржај датотеке *formatirano.csv***

```
/home/januar1/radovi/UP_Ana_Pavlovic_110 10 9 0 8 7
/home/januar1/radovi/UP_Nikola_Petrovic_119 9 0 8 10 0
```

**Решење**

Овај проблем можемо решити сортирањем редова улазне датотеке, прво по имену директоријума, а затим по имену датотеке решења. На овај начин, свих 5 редова који одговарају једном студенту биће узастопни и то баш у одговарајућем редоследу по задацима. У те сврхе можемо учитати све податке у једну колекцију, а за представљање једног реда можемо дефинисати структуру. Резултате уређене на овај начин можемо једноставно преписати у излазну датотеку, при чему за број поена треба да водимо рачуна и о елиминационим тестовима.

```
#include <iostream>
#include <fstream>
#include <string>
#include <sstream>
#include <vector>
#include <algorithm>

using namespace std;

struct Rezultat {
    string direktorijum;
    string zadatak;
    int eliminacioni;
    int testovi;

    bool operator<(const Rezultat& drugi) const {
        if (direktorijum != drugi.direktorijum)
            return direktorijum < drugi.direktorijum;
        return zadatak < drugi.zadatak;
    }
};

void ucitajRezultate(ifstream& ulaz, vector<Rezultat>& lista) {
```

```
    string linija;
    while (getline(ulaz, linija)) {
        stringstream ss(linija);
        Rezultat r;
        ss >> r.direktorijum >> r.zadatak >> r.eliminacioni >> r.testovi;
        lista.push_back(r);
    }
}

void ispisiFormatiraneRezultate(ofstream& izlaz, vector<Rezultat>& lista) {
    sort(lista.begin(), lista.end());

    for (int i = 0; i < lista.size(); i += 5) {
        izlaz << lista[i].direktorijum << "␣";
        for (int j = 0; j < 5; ++j) {
            int poeni = lista[i + j].eliminacioni ? lista[i + j].testovi : 0;
            izlaz << poeni << "␣";
        }
        izlaz << endl;
    }
}

int main() {
    const string ulaznaDatoteka = "rezultati.csv";
    const string izlaznaDatoteka = "formatirano.csv";

    ifstream ulaz(ulaznaDatoteka);
    if (!ulaz.is_open()) {
        cout << "-1" << endl;
        return 0;
    }

    ofstream izlaz(izlaznaDatoteka);
    vector<Rezultat> lista;

    ucitajRezultate(ulaz, lista);
    ispisiFormatiraneRezultate(izlaz, lista);

    return 0;
}
```

## Задатак: Кандидати за стипендије

Написати програм који помаже радницима студентске службе да идентификују студенте који испуњавају услове за добитак стипендије. Студент стиче право да се пријави за стипендију ако:

- се налази на другој, трећој, четвртој или петој години студија
- има просек преко 9.0 и
- има ефикасност од 60 еспб годишње

### Опис улаза

Са стандардног улаза уносе се редом: име датотеке која садржи податке о студентима, име датотеке у коју је потребно уписати резултате претраге и година студија по којој се врши претрага. Улазна датотека садржи редове у следећем формату:

```
godinaStudija ime prezime prosek efikasnost
```

### Опис излаза

Излазна датотека треба да садржи сортирану листу студената са тражене године студија који испуњавају услове за добитак стипендије, и то информације о имену, презимену и просеку. Студенти су у листи сортирани најпре по просеку нерастуће, а затим по презимену и имену неопадајуће. У случају грешке током извршавања програма исписати поруку о грешци која се догодила и прекинути програм.

### Пример

#### Улаз

```
studenti.txt
trecaGodina.txt
3
```

#### Садржај датотеке *studenti.txt*

```
2 Pera Peric 8.5 60
3 Mika Mikic 9.67 60
4 Laza Lazic 6.34 34.5
5 Zika Zikic 9.07 60
5 Petar Petrovic 8.76 60
7 Marija Jankovic 10.0 60
3 Ana Anic 9.43 60
```

```
2 Milica Ilic 7.34 42.5
4 Pavle Pavlovic 7.86 54
6 Jova Jovic 9.80 40
3 Aca Acic 9.67 60
4 Jovan Jovanovic 10.0 60
```

**Излаз, садржај датотеке *trecaGodina.txt***

```
Aca Acic 9.67
Mika Mikic 9.67
Ana Anic 9.43
```

**Решење**

Овај проблем може се ефикасно решити једним проласком кроз листу студената у улазној датотеци. Учитавамо информације о студентима, линију по линију и, када наиђемо на студента који се налази на траженој години студија, проверимо да ли испуњава услове за стицање стипендије. Уколико су сви услови испуњени, информације о тренутном студенту се могу додати у низ. Након проласка кроз датотеку, низ ће садржати информације о свим студентима који испуњавају услове за добитак стипендије. Након тога, потребно је сортирати низ. Резултате уређене на овај начин можемо једноставно преписати у излазну датотеку.

Приликом обраде грешака можемо користити `EXIT_FAILURE`. То је макро дефинисан у заглављу `<cstdlib>` и представља код за неуспешан завршетак програма. Користи се да оперативном систему сигнализира да је програм прекинут због грешке (нпр. погрешан улаз, неуспешно отварање датотеке и сл.).

```
#include<iostream>
#include<vector>
#include<fstream>
#include<string>
#include<sstream>
#include<algorithm>
#include<cstdlib>

using namespace std;

struct Student {
        string prezime;
        string ime;
        int godinaStudija;
        double prosek;
        double efikasnost;
};
```

```
void greska(string poruka){
        cout << poruka << endl;
        exit(EXIT_FAILURE);
}

void obradiFajl(ifstream& studenti, const int& trazenaGodina,
   vector<Student>& rezultat){
    string linija;
    Student s;

    while(getline(studenti, linija)){
        stringstream student(linija);
        int godina;

        student >> godina;
        if(godina != trazenaGodina)
            continue;
        else{
            s.godinaStudija = trazenaGodina;
            student >> s.ime >> s.prezime >> s.prosek >>
               s.efikasnost;
            if(s.prosek >= 9.0 && s.efikasnost == 60)
                rezultat.push_back(s);
        }

    }

}

void ispisiRezultate(ofstream& stipendisti, vector<Student>&
   rezultat){
    sort(begin(rezultat), end(rezultat),
    [](const Student &student1, const Student &student2){
        if(student1.prosek == student2.prosek)
            if(student1.prezime == student2.prezime)
                return student1.ime < student2.ime;
            else
                return student1.prezime < student2.prezime;
        return student1.prosek > student2.prosek;

    });
    for(Student s : rezultat)
```

```
        stipendisti << s.ime << "␣" << s.prezime << "␣" <<
           s.prosek << "\n";

}


int main(){
        string ulazniFajl, izlazniFajl;
        cin >> ulazniFajl >> izlazniFajl;

        int trazenaGodina;
        cin >> trazenaGodina;

    if(trazenaGodina < 2 || trazenaGodina > 5){
        greska("Greska!␣Niste␣uneli␣validnu␣vrednost␣za␣godinu!");
    }

        vector<Student> rezultat;

        ifstream studenti(ulazniFajl);
        ofstream stipendisti(izlazniFajl);

        if(!studenti.is_open()){
                greska("Greska!␣Datoteka␣nije␣otvorena!");
        }

    obradiFajl(studenti, trazenaGodina, rezultat);
    ispisiRezultate(stipendisti, rezultat);


        return 0;
}


}
```

## Задатак: Пријава испита

У току је пријава испита за јануарски испитни рок. Познат је списак студената који су већ пријавили да ће изаћи на испит из Увода у програмиње. Потребно је написати програм који ће за нове испитне пријаве на списак додати оне студенте који нису већ пријавили испит.

### Опис улаза

Са стандардног улаза се учитава назив фајла у коме се налази списак пријављених студената, а затим у сваком реду по једна испитна пријава до краја улаза. Свака испитна пријава садржи презиме, име и број индекса студента раздвојене размаком.

### Опис излаза

На крај фајла који садржи списак пријављених студената дописати оне студенте који нису већ пријавили испит.

### Пример

**Улаз, садржај датотеке *prijava.txt***

```
Stankovic Ivana 9/2025
Petrovic Ana 13/2025
Ilic Stefan 12/2025
Tomic Milica 17/2025
```

**Стандардни улаз**

```
prijava.txt
Jovanovic Marko 7/2025
Petrovic Ana 13/2025
Zoric Milica 4/2025
```

**Излаз, садржај датотеке *prijava.txt***

```
Stankovic Ivana 9/2025
Petrovic Ana 13/2025
Ilic Stefan 12/2025
Tomic Milica 17/2025
Jovanovic Marko 7/2025
Zoric Milica 4/2025
```

### Решење

Овај задатак можемо решити тако што за сваког унетог студента најпре проверимо проласком кроз фајл да ли већ постоји на списку. Уколико је одговор одричан, уписујемо га на крај фајла. Како не бисмо обрисали садржај фајла пре уписивања, потребно је да га отворимо у моду за надовезивање (`ios::app`). Пошто су студенти представљени помоћу структуре `Student`, потребно је дефинисати унутар ње оператор ==, како би било могуће вршити проверу да ли су студенти једнаки.

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

using namespace std;

struct Student {
        string prezime;
        string ime;
        int brojIndeksa;
        int godinaUpisa;

        bool operator==(const Student& other){
                return  prezime == other.prezime &&
                                ime == other.ime &&
                                brojIndeksa == other.brojIndeksa &&
                                godinaUpisa == other.godinaUpisa;
        }
};

int main(){

        string ime_fajla;
        cin >> ime_fajla;

        cin.ignore();

        ofstream dopisivanje(ime_fajla, ios::app);

        Student s;
        string linija;

        while(getline(cin, linija)){
                char crtica;

                stringstream student(linija);
                student >> s.prezime >> s.ime >> s.brojIndeksa >>
                   crtica >> s.godinaUpisa;

                ifstream spisak(ime_fajla);

                if(!spisak.is_open()){
                        cout << -1 << endl;
```

```
                return 0;
        }

        bool nadjen = false;
        while(getline(spisak, linija)){
                Student s1;

                stringstream student1(linija);

                student1 >> s1.prezime >> s1.ime >>
                   s1.brojIndeksa >> crtica >>
                   s1.godinaUpisa;

                if(s == s1){
                        nadjen = true;
                        break;
                }
        }

        if(!nadjen)
                dopisivanje << s.prezime << "␣" << s.ime
                   << "␣" << s.brojIndeksa << "/" <<
                   s.godinaUpisa << endl;
    }


    return 0;
}
```