Univerzitet u Beogradu
Elektrotehnički fakultet

Odabrana poglavlja iz numeričke analize

Najbolje racionalne aproksimacije realnih brojeva

Prvi projektni zadatak

Student:
Jelena Pančevski 2023/3231

Beograd, školska 2023/2024

Projektni zadatak

Najbolje racionalne aproksimacije I i II vrste. *Neka je dat realan broj α. Racionalni broj p/q je najbolja racionalna aproksimacija realnog broja I vrste ako važi nejednakost*

$$\left|\alpha - \frac{p}{q}\right| < \left|\alpha - \frac{r}{s}\right|$$

za sve razlomke r/s ≠ p/q takve da 0 < s ≤ q. Racionalni broj p/q je najbolja racionalna aproksimacija II vrste ako važi nejednakost

$$|q\alpha - p| < |s\alpha - r|$$

za sve razlomke r/s ≠ p/q takve da 0 < s ≤ q.

Neka je dat pozitivan realan broj α sa konačnim decimalskim zapisom i neka su dati prirodni brojevi n i m, tako da n < m. Formirati niz razlomaka p/q takvih da za imenilac q važi n ≤ q ≤ m (tj. q = n, n + 1, . . . , m) i pri tom imeniocu q pridružujemo brojilac p koji određujemo zaokruživanjem na najbliži priodan broj proizvoda α · q. Predstaviti svaki razlomak p/q u obliku verižnog razlomka. U nizu razlomaka p/q izdvojiti:

- *najbolje racionalne aproksimacije I vrste,*
- *najbolje racionalne aproksimacije II vrste,*
- *sortirati sve razlomke p/q po uslovu minimalnosti apsolutne greške* $\left|\alpha - \frac{p}{q}\right|$.

Rešenje problema

U cilju rešavanja ovog problema korišćen je programski jezik C++ kao i biblioteke *iostream*, *vector*, *cmath*, *iomanip*, *algorithm*, *string* i *numeric*. Deklarisana promenljiva *numberofspaces* služi za poravnanje prilikom ispisa, a kasnije će biti više reči o njoj.

```
#include <iostream>
#include <vector>
#include <cmath>
#include <iomanip>
#include <algorithm>
#include <string>
#include <numeric>

using namespace std;
int numberofspaces = 0;
```

Biblioteke korišćene u projektu

Definisana je klasa *Fraction* sa privatnim poljima:

- *int p* – predstavlja brojilac razlomka
- *int q* – predstavlja imenilac razlomka
- *vector<int> decimals* – predstavlja niz verižnih decimala razlomka *p/q*.
- *int charactersnum* – predstavlja ukupan broj karaktera potreban za ispis verižnih decimala, služi prilikom poravnanja ispisa
- *double errorValue* – predstavlja vrednost apsolutne greške $\left|\alpha - \frac{p}{q}\right|$.
- *bool isFirst* – predstavlja informaciju da li je razlomak *p/q* najbolja racionalna aproksimacija I vrste.
- *bool isSecond* – predstavlja informaciju da li je razlomak *p/q* najbolja racionalna aproksimacija II vrste.
- *int index* – predstavlja redni broj razlomka *p/q*.

Konstruktoru klase *Fraction* prosleđuju se vrednosti *p (brojilac)*, *q (imenilac)* i α (pozitivan realan broj čiju aproksimaciju određujemo). Nakon inicijalizovanja svih prethodno navedenih polja, poziva se funkcija *calculateDecimals* koja računa verižne decimale razlomka *p/q*, a potom funkcija *calculateErrorValue* koja prima argument α i računa vrednost apsolutne greške $\left|\alpha - \frac{p}{q}\right|$.

```
class Fraction {
    int p;
    int q;
   vector <int> decimals;
   int charactersnum;
   double errorValue;
   bool isFirst;
   bool isSecond;
   int index;

public:
   Fraction( int p,  int q, double alfa) {
      this->p = p;
      this->q = q;
      this->isFirst = false;
      this->isSecond = false;
      this->errorValue = 0;
      this->charactersnum = 0;
      this->index = -1;
      this->calculateDecimals();
      this->calculateErrorValue(alfa);
   }
   void calculateErrorValue(double alfa) {
      errorValue = abs(alfa - (double)p / q);
   }
```

Klasa Fraction sa privatnim poljima, konstruktorom i funckijom calculateErrorValue

U funkciji *calculateDecimals* implementiran je algoritam za računanje verižnih decimala. Izračunate verižne decimale smeštaju se u niz *decimals*, a informacija o broju karaktera za ispis verižnih decimala se čuva u polju *charactersnum*. U globalnoj promenljivoj *numberofspaces* čuva se maksimalan broj karaktera potrebnih za ispis verižnih decimala.

```
void calculateDecimals() {
    double alfa = ((double)p / q);
    vector <double> x;
    vector <double> ds;
    double d;
    int i = 0;
    x.push_back(alfa);
    this->decimals.push_back(floor(x.at(i)));
    d = x.at(i) - this->decimals.at(i);

    this->charactersnum += to_string(this->decimals.at(i)).length() + 1; //poravnanje za ispis

    while (d > 1e-12){
        ds.push_back(d);
        i++;
        x.push_back(((double)1 / (ds.at(i - 1))));
        this->decimals.push_back(floor(x.at(i)));
        d = x.at(i) - this->decimals.at(i);

        this->charactersnum += to_string(this->decimals.at(i)).length() + 1; //poravnanje za ispis
    };

    while (this->decimals.at(this->decimals.size() - 1) == 1) {
        this->decimals.pop_back();
        this->decimals.at(this->decimals.size() - 1)++;

        if (this->decimals.at(this->decimals.size() - 1) >= 10)this->charactersnum++; //poravnanje za ispis
        this->charactersnum -= 2;
    }
    if (numberofspaces < this->charactersnum) {
        numberofspaces = this->charactersnum;
    }

}
```

Funkcija calculateDecimals

Nakon funkcije *calculateDecimals* definisane su *get* i *set* funkcije za privatna polja klase *Fraction* koje su prikazane na narednoj slici.

```
void setFirst(bool value) {
    isFirst = value;
}
void setSecond(bool value) {
    isSecond = value;
}
void setIndex(int index) {
    this->index = index;
}

bool getIsFirst() {
    return isFirst;
}
bool getIsSecond() {
    return isSecond;
}
int getP() {
    return p;
}
int getQ() {
    return q;
}
int getIndex() {
    return this->index;
}
double getErrorValue() {
    return errorValue;
}
```

get i set funkcije za privatna polja klase Fraction

Potom slede funkcije koje olakšavaju ispis vrednosti polja klase *Fraction*. Funkcije *getFraction* i *printFraction* služe za ispis samog razlomka *(p/q)*, funkcija *printErrorVal* služi za ispis apsolutne greške sa tačnošću od 20 decimala, *printDecimals* ispisuje verižne decimale razlomka, dok *printAprox* ispisuje da li je racionalni broj *p/q* najbolja racionalna aproksimacija prve vrste broja α (*I*) , najbolja racionalna aproksimacija druge vrste broja α (*II*) ili ne predstavlja najbolju racionalnu aproksimaciju ni prve ni druge vrste (*N*).

Za klasu *Fraction* je takođe izvršeno preklapanje operatora < čime je omogućeno poređenje dva razlomka na osnovu apsolutne greške koje će se koristiti prilikom sortiranja razlomaka po uslovu minimalnosti apsolutne greške.

```
string getFraction() {
    return to_string(this->p) + "/" + to_string(this->q);
}
void printFraction() {
    printf("%d/%d", p, q);
}
void printErrorVal() {
    printf("%0.20lf", errorValue);
}
void printDecimals() {

    printf("\t[");
    for (int i = 0; i < this->decimals.size(); i++) {
        cout << this->decimals.at(i);
        if (i == 0)  cout << ";";
        else if (i != this->decimals.size() - 1)cout << ",";

    }
    printf("]");

    for (int i = this->charactersnum; i < numberofspaces; i++) {
        printf(" ");
    }

}
void printAprox() {
    if (isSecond) printf("II\n");
    else if (isFirst) printf("I\n");
    else printf("N\n");
}

bool operator<(const Fraction &fraction ) {
    return this->errorValue < fraction.errorValue;
}
```

Funkcije za ispis polja klase Fraction i preklopljen operator<.

Nakon klase *Fraction* definisane su funkcije *fristErr* i *secondErr* koje za dati realan broj α, brojilac *p* i imenilac *q* računaju apsolutnu grešku potrebnu za proveru da li je racionalni broj *p/q* najbolja racionalna aproksimacija prve odnosno druge vrste respektivno.

Funkcija *calculateNumerator* računa za dati realan broj α i dati imenilac *q* brojilac p koji se određuje zaokruživanjem na najbliži prirodan broj proizvoda $\alpha \cdot q$ korišćenjem funkcije *round.*

```
double firstErr(double alfa, int p, int q) {
    return  abs(alfa - ((double)p / q));
};
double secondErr(double alfa, int p, int q) {
    return  abs((alfa*q - p));
};
int calculateNumerator(double alfa, int q) {
    int p = round(alfa * q);  // najblizi prirodan broj
    return p;
};
```

Pomoćne funckije fristErr, secondErr i calculateNumerator

Funkcija *FirstorSecond* za dati realan broj *α,* imenilac *q*, vektor odnosno dinamički niz razlomaka *fractions*, proverava za svaki od razlomaka u nizu *fractions* da li predstavlja najbolju racionalnu aproksimaciju prve i druge vrste broja *α*. U funkciji se najpre određuje minimalna greška za prvu i drugu vrstu počevši od imenilaca *s=1* dok se ne stigne do *q*. Minimalna greška prve vrste smešta se u promenljivu *firstMin*, dok se minimalna greška druge vrste smešta u promenljivu *secondMin*.

```
void FirstorSecond(double alfa, int q, vector <Fraction>& fractions) {
    double firstMin = INT16_MAX;
    double secondMin = INT16_MAX;
    int s = 1;
    for (s = 1; s < q; s++) {
        //first
        double r = calculateNumerator(alfa, s);
        if (s == 1) {
            firstMin = firstErr(alfa, r, s);
        }
        double currentErr = firstErr(alfa, r, s);
        if (currentErr < firstMin) {
            firstMin = currentErr;
        }
        //second
        if (s == 1) {
            secondMin = secondErr(alfa, r, s);
        }
        currentErr = secondErr(alfa, r, s);
        if (currentErr < secondMin) {
            secondMin = currentErr;
        }
    }
```

FirstorSecond funkcija – određivanje minimalne greške za I i II vrstu

Nakon toga se za svaki razlomak unutar niza *fractions* najpre računa greška za drugu vrstu pozivom funkcije *secondErr*, greška se potom smešta u promenljivu *currentErr* i upoređuje sa trenutnim minimumom druge vrste.

Ako je uslov *currentErr* < *secondMin* ispunjen, razlomak predstavlja najbolju racionalnu aproksimaciju II vrste, a samim tim i prve vrste iz tih razloga se pozivaju set funkcije za polja *isSecond* i *isFirst* i vrednosti tih polja se setuju na *true.* Takođe se ažurira minimalna greška za II vrstu na trenutno izračunatu grešku. Potom se računa greška za I vrstu, pozivom funkcije *firstErr* i ukoliko važi uslov da je *currentErr* < *firstMin* onda se ažurira minimum za I vrstu.

Ako uslov *currentErr* < *secondMin* nije ispunjen, poziva se metoda *setSecond* i prosleđuje joj se vrednost *false,* razlomak ne predstavlja najbolju aproksimaciju II vrste. Nakon toga se proverava da li razlomak predstavlja najbolju aproksimaciju I vrste računanjem greške za I vrstu pozivom funkcije *fristErr* i proverom uslova *currentErr* < *firstMin.* Ukoliko je uslov ispunjen, razlomak predstavlja najbolju racionalnu aproksimaciju I vrste i potom se setuje vrednost polja *isFirst* na *true* i ažurira vrednost *firstMin* na *currentErr*, a u suportnom se *isFirst* setuje na *false*.

```
for (int i = 0; i < fractions.size(); i++) {
    double currentErr = secondErr(alfa, fractions.at(i).getP(), fractions.at(i).getQ());
    if (currentErr < secondMin) {
        secondMin = currentErr;
        fractions.at(i).setSecond(true);
        fractions.at(i).setFirst(true);
        currentErr = firstErr(alfa, fractions.at(i).getP(), fractions.at(i).getQ());
        if (currentErr < firstMin) {
            firstMin = currentErr;
        }
    }
    else {
        fractions.at(i).setSecond(false);
        currentErr = firstErr(alfa, fractions.at(i).getP(), fractions.at(i).getQ());
        if (currentErr < firstMin) {
            firstMin = currentErr;
            fractions.at(i).setFirst(true);
        }
        else {
            fractions.at(i).setFirst(false);
        }
    }


}
```

FirstorSecond funkcija – određivanje za svaki razlomak da li predstavlja najbolju racionalnu aproksimaciju I i II vrste

Nakon funkcije *FirstorSecond* sledi funkcija *printSortedFractions* koja prima kao prvi argument vektor objekata klase *Fraction fractions* i vrši tabelarni ispis razlomaka tako što se u prvoj koloni nalazi sam razlomak *p/q*, potom sledi verižni zapis razlomka, odstupanje tj. vrednost apsolutne greške i informacija da li razlomak predstavlja najbolju racionalnu aproksimaciju *I/II* vrste ili ni jedne *(N).*

```
void printSortedFractions(vector <Fraction>& fractions) {
    printf("Razlomak \t Verizni zapis \t\t Odstupanje \t\t Vrsta \n");
    for (int i = 0; i < fractions.size(); i++) {
        Fraction f = fractions.at(i);
        f.printFraction();
        printf("\t");
        f.printDecimals();
        printf("\t\t");
        f.printErrorVal();
        printf("\t\t");
        f.printAprox();
    }
}
```

printSortedFractions funkcija za tabelarni prikaz razlomaka

Unutar *main* funkcije deklarišu se promenljive α, *n* i *m*. Potom se traži od korisnika da se unesu vrednosti za deklarisane promenljive, nakon čega se proverava da li su unete korektne vrednosti, odnosno da li je α pozitivan realan broj kao i da li su *n* i *m* prirodni brojevi tako da važi da je *n*<*m*. Ukoliko neki od uslova nije ispunjen program se završava sa ispisom greške.

```
int main() {
    double alfa;
    int n;
    int m;
    cout << "Unesite pozitivan realan broj alfa\n";
    cin >> alfa;
    cout << "Unesite prirodan broj n\n";
    cin >> n;
    cout << "Unesite prirodan broj m tako da vazi n<m\n";
    cin >> m;

    if (alfa < 0) {
        printf("Nekorektan unos za alfa - alfa nije pozitivan realan broj\n");
        return -1;
    }
    if (n <= 0 || m <= 0 ) {
        printf("Nekorektan unos za n i m - brojevi ne pripadaju skupu prirodnih brojeva\n");
        return -1;
    }
    if (n >= m) {
        printf("Nekoretkan unos za n i m - nije ispunjen uslov n<m \n");
        return -1;
    }
```

Main funkcija – unos vrednosti i provera unetih vrednosti za promenljive α, n i m

Ukoliko su svi uslovi ispunjeni, kreira se vektor, tj. dinamički niz objekta klase *Fractions*. Zatim se za sve imenioce *q* čije vrednosti idu od *n* do *m* računa brojilac *p* pozivom metode *calculateNumerator*. Zatim se proverava da li je tako dobijen razlomak redukovan pozivom funkcije *gcd* kojoj prosleđujemo imenilac *q* i brojilac *p* i koja vraća najveći zajednički delilac. Ako je povratna vrednost veća od jedinice, takav razlomak može da se redukuje i samim tim već

postoji u nizu *fractions* te ga iz tih razloga preskačemo. Ukoliko je razlomak redukovan kreiramo objekat klase *Fraction* sa dobijenim vrednostima p,q i α i dodajemo ga u vektor *fractions*.

Nakon završetka kreiranja svih razlomaka, poziva se funkcija *FirstorSecond* za određivanje da li su razlomci najbolje racionalne aproksimacije prve/druge vrste. Potom se redom ispisuju svi razlomci koji su najbolje racionalne aproksimacije prve vrste, a nakon toga svi razlomci koji su najbolje racionalne aproksimacije druge vrste.

Nakon toga se vrši sortiranje svih dobijenih razlomaka na osnovu vrednosti apsolutne greške pozivom funkcije *sort* na vektor *fractions*. Kada se završi sortiranje poziva se funkcija *printSortedFractions* koja vrši tabelarni ispis razlomaka.

```
vector <Fraction> fractions;
for (int q = n; q <= m; q++) {
    int p = calculateNumerator(alfa, q);
    if (gcd(p, q) != 1) {
        //Preskacemo neredukovane razlomke
        continue;
    }
    Fraction f(p, q, alfa);
    fractions.push_back(f);

}

FirstorSecond(alfa, n, fractions);

printf("Za konstantu alfa = %0.20lf pri izboru n = %d i m = %d \n", alfa,n,m);

string first = "Najbolje racionalne aproksimacije I vrste su ";
string second = "Najbolje racionalne aproksimacije II vrste su ";
for (int i = 0; i < fractions.size(); i++) {
    if (fractions.at(i).getIsFirst()) {
        first.append(fractions.at(i).getFraction() + " ");
    }
    if (fractions.at(i).getIsSecond()) {
        second.append(fractions.at(i).getFraction()+" ");
    }
}
printf((first +"\n").c_str());
printf((second +"\n").c_str());

sort(fractions.begin(), fractions.end());
printSortedFractions(fractions);
```

Main funkcija – kreiranje razlomaka, određivanje najboljih racionalnih aproksimacija I i II vrste, sortiranje svih dobijenih razlomaka na osnovu vrednosti apsolutne greške i tabelarni ispis

Test primeri

Primer 1

Za ulazne vrednosti *α= 0.5849625007211561815, n=7, m=53 p*rogram izvršava sledeći ispis.

```
Unesite pozitivan realan broj alfa
0.5849625007211561815
Unesite prirodan broj n
7
Unesite prirodan broj m tako da vazi n<m
53
Za konstantu alfa = 0.58496250072115618668 pri izboru n = 7 i m = 53
Najbolje racionalne aproksimacije I vrste su 4/7 7/12 17/29 24/41 31/53
Najbolje racionalne aproksimacije II vrste su 7/12 24/41 31/53
Razlomak         Verizni zapis           Odstupanje              Vrsta
31/53           [0;1,1,2,2,4]                  0.00005684034379771497          II
24/41           [0;1,1,2,2,3]                  0.00040335293738036349          II
17/29           [0;1,1,2,2,2]                  0.0012443958305679014           I
7/12            [0;1,1,2,2]                    0.0016291673878228163           II
27/46           [0;1,1,2,2,1,2]                0.0019940210179743012           N
10/17           [0;1,1,2,3]                    0.0032727933964908917           N
25/43           [0;1,1,2,1,1,3]                0.0035671518839468507           N
18/31           [0;1,1,2,1,1,2]                0.0043173394308335664           N
23/39           [0;1,1,2,3,2]                  0.0047810890224335711           N
29/50           [0;1,1,2,1,1,1,2]              0.0049625007211562266           N
13/22           [0;1,1,2,4]                    0.0059465901879347526           N
11/19           [0;1,1,2,1,2]                  0.0060151323001035317           N
29/49           [0;1,1,2,4,2]                  0.0068742339727213552           N
26/45           [0;1,1,2,1,2,2]                0.0071847229433784631           N
16/27           [0;1,1,2,5]                    0.0076300918714363730           N
15/26           [0;1,1,2,1,3]                  0.0080394237980793148           N
19/32           [0;1,1,2,6]                    0.0087874992788438133           N
19/33           [0;1,1,2,1,4]                  0.0092049249635803854           N
22/37           [0;1,1,2,7]                    0.0096320938734384409           N
23/40           [0;1,1,2,1,5]                  0.0099625007211562311           N
25/42           [0;1,1,2,8]                    0.0102755945169390461           N
27/47           [0;1,1,2,1,6]                  0.0104944156147731559           N
4/7             [0;1,1,3]                      0.0135339292925847898           I
13/23           [0;1,1,3,3]                    0.0197451094168084040           N
9/16            [0;1,1,3,2]                    0.0224625007211561867           N
11/18           [0;1,1,1,1,3]                  0.0261486103899549737           N
5/9             [0;1,1,4]                      0.0294069451656006064           N
8/13            [0;1,1,1,1,2]                  0.0304221146634592321           N
6/11            [0;1,1,5]                      0.0395079552666107725           N
5/8             [0;1,1,1,2]                    0.0400374992788438133           N
```

Ispis programa za vrednosti vrednosti *α= 0.5849625007211561815, n=7, m=53*

Primer 2

Za ulazne vrednosti α= 3.141256, *n=1, m=10* program izvršava sledeći ispis.

```
Unesite pozitivan realan broj alfa
 3.141256
Unesite prirodan broj n
1
Unesite prirodan broj m tako da vazi n<m
10
Za konstantu alfa = 3.14125599999999982614 pri izboru n = 1 i m = 10
Najbolje racionalne aproksimacije I vrste su 3/1 13/4 16/5 19/6 22/7
Najbolje racionalne aproksimacije II vrste su 3/1 22/7
Razlomak         Verizni zapis          Odstupanje              Vrsta
22/7             [3;7]          0.00160114285714296756          II
25/8             [3;8]          0.01625599999999982614          N
19/6             [3;6]          0.02541066666666669249          I
28/9             [3;9]          0.03014488888888866569          N
31/10            [3;10]         0.04125599999999973733          N
16/5             [3;5]          0.05874400000000035149          I
13/4             [3;4]          0.10874400000000017386          I
3/1              [3;]           0.14125599999999982614          II
```

Ispis programa za vrednosti vrednosti α= 3.141256, n=1, m=10

Kod

U nastavku je dat kompletan kod rešenja. Kod je napisan u Visual Studio 2019 razvojnom okruženju, korišćenjem C++ 17 standarda.

```
#include <iostream>
#include <vector>
#include <cmath>
#include <iomanip>
#include <algorithm>
#include <string>
#include <numeric>

using namespace std;
int numberofspaces = 0;

class Fraction {
        int p;
        int q;
       vector <int> decimals;
       int charactersnum;
       double errorValue;
       bool isFirst;
       bool isSecond;
       int index;

public:
       Fraction( int p,  int q, double alfa) {
              this->p = p;
              this->q = q;
              this->isFirst = false;
              this->isSecond = false;
              this->errorValue = 0;
              this->charactersnum = 0;
              this->index = -1;
              this->calculateDecimals();
              this->calculateErrorValue(alfa);
       }
       void calculateErrorValue(double alfa) {
              errorValue = abs(alfa - (double)p / q);
       }

       void calculateDecimals() {
              double alfa = ((double)p / q);
              vector <double> x;
              vector <double> ds;
              double d;
              int i = 0;
              x.push_back(alfa);
              this->decimals.push_back(floor(x.at(i)));
              d = x.at(i) - this->decimals.at(i);

              this->charactersnum += to_string(this->decimals.at(i)).length() + 1;
//poravnanje za ispis

              while (d > 1e-12){
                     ds.push_back(d);
```

```
                        i++;
                        x.push_back(((double)1 / (ds.at(i - 1))));
                        this->decimals.push_back(floor(x.at(i)));
                        d = x.at(i) - this->decimals.at(i);

                        this->charactersnum += to_string(this->decimals.at(i)).length() + 1;
//poravnanje za ispis
                };

                while (this->decimals.at(this->decimals.size() - 1) == 1) {
                        this->decimals.pop_back();
                        this->decimals.at(this->decimals.size() - 1)++;

                        if (this->decimals.at(this->decimals.size() - 1) >= 10)this-
>charactersnum++; //poravnanje za ispis
                        this->charactersnum -= 2;
                }
                if (numberofspaces < this->charactersnum) {
                        numberofspaces = this->charactersnum;
                }

        }
        void setFirst(bool value) {
                isFirst = value;
        }
        void setSecond(bool value) {
                isSecond = value;
        }
        void setIndex(int index) {
                this->index = index;
        }

        bool getIsFirst() {
                return isFirst;
        }
        bool getIsSecond() {
                return isSecond;
        }
        int getP() {
                return p;
        }
        int getQ() {
                return q;
        }
        int getIndex() {
                return this->index;
        }
        double getErrorValue() {
                return errorValue;
        }
        string getFraction() {
                return to_string(this->p) + "/" + to_string(this->q);
        }
        void printFraction() {
                printf("%d/%d", p, q);
        }
        void printErrorVal() {
                printf("%0.20lf", errorValue);
```

```
	}
	void printDecimals() {

		printf("\t[");
		for (int i = 0; i < this->decimals.size(); i++) {
			cout << this->decimals.at(i);
			if (i == 0)  cout << ";";
			else if (i != this->decimals.size() - 1)cout << ",";

		}
		printf("]");

		for (int i = this->charactersnum; i < numberofspaces; i++) {
			printf(" ");
		}

	}
	void printAprox() {
		if (isSecond) printf("II\n");
		else if (isFirst) printf("I\n");
		else printf("N\n");
	}

	bool operator<(const Fraction &fraction ) {
		return this->errorValue < fraction.errorValue;
	}
};
double firstErr(double alfa, int p, int q) {
	return abs(alfa - ((double)p / q));
};
double secondErr(double alfa, int p, int q) {
	return abs((alfa*q - p));
};
int calculateNumerator(double alfa, int q) {
	int p = round(alfa * q);  // najblizi prirodan broj
	return p;
};

void FirstorSecond(double alfa, int q, vector <Fraction>& fractions) {
	double firstMin = INT16_MAX;
	double secondMin = INT16_MAX;
	int s = 1;
	for (s = 1; s < q; s++) {
		//first
		double r = calculateNumerator(alfa, s);
		if (s == 1) {
			firstMin = firstErr(alfa, r, s);
		}
		double currentErr = firstErr(alfa, r, s);
		if (currentErr < firstMin) {
			firstMin = currentErr;
		}
		//second
		if (s == 1) {
			secondMin = secondErr(alfa, r, s);
		}
		currentErr = secondErr(alfa, r, s);
		if (currentErr < secondMin) {
```

```
                        secondMin = currentErr;
                }
        }

        for (int i = 0; i < fractions.size(); i++) {
                double currentErr = secondErr(alfa, fractions.at(i).getP(),
fractions.at(i).getQ());
                if (currentErr < secondMin) {
                        secondMin = currentErr;
                        fractions.at(i).setSecond(true);
                        fractions.at(i).setFirst(true);
                        currentErr = firstErr(alfa, fractions.at(i).getP(),
fractions.at(i).getQ());
                        if (currentErr < firstMin) {
                                firstMin = currentErr;
                        }
                }
                else {
                        fractions.at(i).setSecond(false);
                        currentErr = firstErr(alfa, fractions.at(i).getP(),
fractions.at(i).getQ());
                        if (currentErr < firstMin) {
                                firstMin = currentErr;
                                fractions.at(i).setFirst(true);
                        }
                        else {
                                fractions.at(i).setFirst(false);
                        }
                }


        }



};

void printSortedFractions(vector <Fraction>& fractions) {
        printf("Razlomak \t Verizni zapis \t\t Odstupanje \t\t Vrsta \n");
        for (int i = 0; i < fractions.size(); i++) {
                Fraction f = fractions.at(i);
                f.printFraction();
                printf("\t");
                f.printDecimals();
                printf("\t\t");
                f.printErrorVal();
                printf("\t\t");
                f.printAprox();
        }
}

int main() {
        double alfa;
        int n;
        int m;
        cout << "Unesite pozitivan realan broj alfa\n";
        cin >> alfa;
        cout << "Unesite prirodan broj n\n";
```

```
	cin >> n;
	cout << "Unesite prirodan broj m tako da vazi n<m\n";
	cin >> m;

	if (alfa < 0) {
		printf("Nekorektan unos za alfa - alfa nije pozitivan realan broj\n");
		return -1;
	}
	if (n <= 0 || m <= 0 ) {
		printf("Nekorektan unos za n i m - brojevi ne pripadaju skupu prirodnih
brojeva\n");
		return -1;
	}
	if (n >= m) {
		printf("Nekoretkan unos za n i m - nije ispunjen uslov n<m \n");
		return -1;
	}

	vector <Fraction> fractions;
	for (int q = n; q <= m; q++) {
		int p = calculateNumerator(alfa, q);
		if (gcd(p, q) != 1) {
			//Preskacemo neredukovane razlomke
			continue;
		}
		Fraction f(p, q, alfa);
		fractions.push_back(f);

	}

	FirstorSecond(alfa, n, fractions);

	printf("Za konstantu alfa = %0.20lf pri izboru n = %d i m = %d \n", alfa,n,m);

	string first = "Najbolje racionalne aproksimacije I vrste su ";
	string second = "Najbolje racionalne aproksimacije II vrste su ";
	for (int i = 0; i < fractions.size(); i++) {
		if (fractions.at(i).getIsFirst()) {
			first.append(fractions.at(i).getFraction() + " ");
		}
		if (fractions.at(i).getIsSecond()) {
			second.append(fractions.at(i).getFraction()+" ");
		}
	}
	printf((first +"\n").c_str());
	printf((second +"\n").c_str());

	sort(fractions.begin(), fractions.end());
	printSortedFractions(fractions);


}
```