Природно-математички факултет
Универзитет у Бањој Луци

Итеративне наредбе – Група 2 средње школе

Аутори: Драган Матић, Милана Грбић,
Милан Предојевић, Ненад Вилендечић

2025

Садржај

Увод

На другом предавању Пољетне школе програмирања за ученике средњих школа обрађена је тема итеративне наредбе.

У наставку се налазе задаци који су рађени током предавања, са циљем да ученици примјене и утврде стечено знање кроз практичне примјере. На крају документа налазе се задаци за домаћи рад, који служе за додатно вјежбање и самостално истраживање обрађених концепата.

Задаци

1. Задатак: Производња малина

Власник имања, Маринко, одлучио је да гаји малине. Направио је план за n година. Прве године планира да произведе t тона малина, а сваке сљедеће да повећа производњу за $p\%$. Написати програм којим се одређује колико тона малина Маринко планира да произведе n-те године узгајања малина.

Примјер извршавања

```
Unesi broj godina...
5
Unesi informaciju koliko tona malina Marinko planira da proizvede prve godine...
2
Unesi procenat...
25
5-te godine Marinko ce proizvesti 4.88 malina
```

Рјешење

```
#include<iostream>
#include<bits/stdc++.h>
using namespace std;

int main(){

    int  godine;
    double tone, procenat, toneN;

    cout<<"Unesi broj godina..."<<endl;
    cin>>godine;


    cout<<"Unesi informaciju koliko tona malina Marinko planira
   da proizvede prve godine..."<<endl;
    cin>>tone;

    cout<<"Unesi procenat..."<<endl;
    cin>>procenat;

    toneN = tone;

    for(int i=1; i<godine; i++)
        toneN = toneN+toneN*procenat/100;

    cout<<godine<<"-te godine Marinko ce proizvesti " << fixed <<
   setprecision(2)<<toneN<<" malina"<<endl;

    return 0;
}
```

2. Задатак: Генерисање чудних бројева

Чудни бројеви су они који се могу изразити као збир квадрата два броја. На пример, број 5 је чудан јер важи: $5 = 1^2 + 2^2$. Написати програм који генерише све чудне бројеве који су мањи од n, гдје је n број унесен са тастатуре.

Примјер извршавања

```
Unesite broj n: 10
Cudni brojevi manji od 10:
1 = 0^2 + 1^2
2 = 1^2 + 1^2
4 = 0^2 + 2^2
5 = 1^2 + 2^2
8 = 2^2 + 2^2
9 = 0^2 + 3^2
```

Рјешење

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

int main() {
    int n;
    cout << "Unesite broj n: ";
    cin >> n;

    if (n <= 0) {
        cout << "Greska: N mora biti pozitivan broj." << endl;
        return 1;
    }

    cout << "Cudni brojevi manji od " << n << ":" << endl;
    for (int i = 1; i < n; i++) {
        for (int a = 0; a * a <= i; a++) {
            for (int b = a; b * b + a * a <= i; b++) {
                if (a * a + b * b == i) {
                    cout << i << " = " << a << "^2 + " << b
<< "^2" << endl;
                    break;
                }
            }
        }
    }
    return 0;
}
```

3. Задатак: Hammingovo растојање

Написати програм који за два цијела броја, који се уносе са тастатуре, на екрану исписује њихово Hammingovo растојање. Hammingovo растојање између два броја дефинише се као број битова на којима се разликују њихови бинарни записи. Бројеви који се уносе са тастатуре су у декадном запису.

Примјер извршавања

```
Unesi prvi broj
8
Unesi drugi broj
9
Hammingovo rastojanje brojeva 8 i 9 iznosi: 1
```

Рјешење

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

int main(){

    int  broj1, broj2;

    cout<<"Unesi prvi broj"<<endl;
    cin>>broj1;

    cout<<"Unesi drugi broj"<<endl;
    cin>>broj2;

    int Hamming = 0;

    int broj11 = broj1, broj21 = broj2;

    do{
        int zadnja1 = broj1 % 2;
        broj1 /= 2;
        int zadnja2 = broj2 % 2;
        broj2 /= 2;

        if(zadnja1 != zadnja2)
        Hamming++;


    }while(broj1>0 || broj2>0);

    cout<<"Hammingovo rastojanje brojeva "<<broj11<<" i
   "<<broj21<<" iznosi: "<<Hamming;

    return 0;
}
```

4. Задатак: Парно непарни

Перо се игра картама. Све карте које има у руци је сложио тако да прво иду све карте са парним бројевима, а затим оне са непарним бројевима (могуће је и да је Перо имао само карте са парним бројевима или само карте са непарним бројевима). Написати програм који провјерава да ли је Перо исправно сложио карте. Карта ас и карте са сликама имају вриједности редом 11, 12, 13 и 14.

Примјер извршавања

```
Unesite Perine karte...
6
4
8
12
9
11
^Z
da
```

Рјешење

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

int main() {
    bool OK = true;         // da li je serija za sada ispravna
    bool neparni = false;  // da li smo presli u dio sa neparnim brojevima
    int x;                      // trenutni element

    cout<<"Unesite Perine karte..."<<endl;

    while (cin >> x && OK) { // ucitavamo brojeve do kraja ulaza i dok je procitani dio ispravno slozen
        cin >> x;
        if (x % 2 != 0)       // ako je broj neparan
        neparni = true;  // prelazimo na citanje neparnih
        else if (neparni)     // ako je paran broj u neparnom delu niza
        OK = false;       // konstatujemo da je doslo do greske
    }

    cout << (OK ? "da" : "ne") << endl; // ispisujemo rezultat
    return 0;
}
```

5. Задатак: Апроксимација броја π

Број π је један од најпознатијих и најважнијих бројева у математици, а његова присутност у различитим гранама науке, од геометрије до физике, чини га фасцинантним и неисцрпним извором истраживања. Иако је вриједност броја π била позната још у старом Египту и Вавилону, прва тачна апроксимација постигнута је тек много касније, уз развој све софистицираних математичких техника. У овом контексту, апроксимација броја π представља изазов који не само да подстиче развој математичких метода, већ и рачунарских алгоритама. Кроз ове задатке, имаћете прилику да истражите различите приступе за апроксимацију броја π.

(а) Leibnizова формула

Leibnizова формула за апроксимацију броја π дата је сљедећом бесконачном сумом:

$$\frac{\pi}{4} = \sum_{k=0}^{\infty} \frac{(-1)^k}{2k+1} \tag{1}$$

Имплементирати итеративни алгоритам који рачуна апроксимацију броја π, на дату тачност ϵ, користећи ову формулу.

(б) Nilakanthова серија

Nilakanthова серија користи суму:

$$\pi = 3 + 4 \sum_{k=1}^{\infty} \frac{(-1)^{k+1}}{(2k)(2k+1)(2k+2)} \tag{2}$$

Написати алгоритам који рачуна апроксимацију броја π, на дату тачност ϵ, користећи ову формулу.

(в) Monte Carlo метода

Користећи Monte Carlo технику, број π може се апроксимирати помоћу случајно генерисаних тачака:

- Генерисати N случајних тачака (x, y) унутар квадрата $[-1, 1] \times [-1, 1]$.
- Одредити колико тачака упада унутар јединичног круга $x^2 + y^2 \leq 1$.
- Апроксимација се добија као:

$$\pi \approx 4 \times \frac{\text{број тачака у кругу}}{\text{укпан број тачака}} \tag{3}$$

Написати програм који имплементира ову методу.

(г) Gauss-Legendre алгоритам

Gauss-Legendre алгоритам је итеративни метод за апроксимацију броја π који се заснива на аритметичко-геометријској средини (AGM) и елиптичким интегралним. Његова конвергенција је квадратна, што значи да се број тачних цифара удвостручује при свакој итерацији.

Дефинишемо почетне вриједности:

$$a_0 = 1,\ b_0 = \frac{1}{\sqrt{2}},\ t_0 = \frac{1}{4},\ p_0 = 1$$

Затим итеративно рачунамо:

1. Аритметичка средина:

$$a_{n+1} = \frac{a_n + b_n}{2}$$

2. Геометријска средина:

$$b_{n+1} = \sqrt{a_n b_n}$$

3. Ажурирање t параметра:

$$t_{n+1} = t_n - p_n(a_n - a_{n+1})^2$$

4. Ажурирање пондера:

$$p_{n+1} = 2p_n$$

Када добијемо довољно итерација, број π се апроксимира као:

$$\pi \approx \frac{(a_n + b_n)^2}{4t_n} \tag{4}$$

Написати алгоритам који рачуна апроксимацију броја π, на дату тачност ϵ.

(д) Wallisov производ

Wallisov производ је бесконачан производ који конвергира ка вриједности броја π. Дат је сљедећим изразом:

$$\pi = 2\prod_{n=1}^{\infty} \frac{(2n)(2n)}{(2n-1)(2n+1)}.$$

Написати алгоритам који рачуна апроксимацију броја π, на дату тачност ϵ.

(ђ) BBP формула

Bailey-Borwein-Plouffe (BBP) формула омогућава израчунавање цифара броја π без потребе за претходним цифрама:

$$\pi = \sum_{k=0}^{\infty} \frac{1}{16^k}\left(\frac{4}{8k+1} - \frac{2}{8k+4} - \frac{1}{8k+5} - \frac{1}{8k+6}\right) \tag{5}$$

Имплементирати алгоритам који користи ову формулу за апроксимацију броја π.

(e) Процијенити који од наведених алгоритама најбрже конвергира броју π.

Примјер извршавања

```
Unesi epsilon...
0.0000001
Leibniz aproksimacija broja pi je: 3.14159
Nilakantha aproksimacija broja pi je: 3.14159
Monte-Carlo aproksimacija broja pi je: 3.14093
Gauss-Legendre aproksimacija broja pi je: 3.14159
Wallis aproksimacija broja pi je: 3.14131
BBP aproksimacija broja pi je: 3.14159
```

Рјешење

```cpp
#include<iostream>
#include <cmath>
using namespace std;
double leibniz(double epsilon){
    double rezultat = 0.0;

    double term;
    int k = 0;
    do {
        term = (k % 2 == 0 ? 1.0 : -1.0) / (2 * k + 1);
        rezultat += term;
        k++;
    } while (fabs(term) >= epsilon);


    return 4*rezultat;
}
double nilakantha(double epsilon) {
    double rezultat = 3.0;
    double term;
    int k = 0;
    do {
        term = 4.0 / ((2 * k + 2) * (2 * k + 3) * (2 * k + 4));
        rezultat += (k % 2 == 0 ? term : -term);
        k++;
    } while (fabs(term) >= epsilon);
    return rezultat;
}
double monte_carlo(int iterations = 10000000) {
    int inside = 0;
    for (int i = 0; i < iterations; i++) {
        double x = (double)rand() / RAND_MAX;
        double y = (double)rand() / RAND_MAX;
        if (x * x + y * y <= 1) {
```

```
35              inside++;
36          }
37      }
38      return 4.0 * inside / iterations;
39  }
40
41  double gauss_legendre(double epsilon) {
42      double a = 1.0, b = 1.0 / sqrt(2), t = 1.0 / 4.0, p = 1.0;
43      double a_next, srezultat = 0, nrezultat=0;
44      do {
45          srezultat = nrezultat; // novi rezultat je sada stari
    rezultat
46          a_next = (a + b) / 2;
47          b = sqrt(a * b);
48          t -= p * (a - a_next) * (a - a_next);
49          a = a_next;
50          p *= 2;
51          nrezultat = (a + b) * (a + b) / (4 * t);
52      } while (fabs(nrezultat - srezultat) >= epsilon);
53      return nrezultat;
54  }
55
56  double wallis(double epsilon) {
57      double srezultat = 2.0, nrezultat = 2.0;
58      double term;
59      int n = 1;
60      do {
61          srezultat = nrezultat; // novi rezultat je sada stari
    rezultat
62          term = (2.0 * n) * (2.0 * n) / ((2.0 * n - 1) * (2.0 * n
    + 1));
63          nrezultat *= term;
64          n++;
65      }  while (fabs(nrezultat - srezultat) >= epsilon);
66      return nrezultat;
67  }
68
69  double bbp(double epsilon) {
70      double rezultat = 0.0;
71      int k = 0.0;
72      double term;
73      do{
74          term = (1.0 / pow(16, k)) *
75          (4.0 / (8 * k + 1) -
76          2.0 / (8 * k + 4) -
77          1.0 / (8 * k + 5) -
78          1.0 / (8 * k + 6));
79          rezultat += term;
80          k++;
81      }while(term>epsilon);
82      return rezultat;
```

```
}

int main(){

    double pi;
    // double epsilon = 0.0000001;

    double epsilon;
    cout<<"Unesi epsilon..."<<endl;
    cin>>epsilon;

    pi = leibniz(epsilon);
    cout<<"Leibniz aproksimacija broja pi je: "<<pi<<endl;
    pi = nilakantha(epsilon);
    cout<<"Nilakantha aproksimacija broja pi je: "<<pi<<endl;

    pi = monte_carlo();
    cout<<"Monte-Carlo aproksimacija broja pi je: "<<pi<<endl;
    pi = gauss_legendre(epsilon);
    cout<<"Gauss-Legendre aproksimacija broja pi je: "<<pi<<endl;

    pi = wallis(epsilon);
    cout<<"Wallis aproksimacija broja pi je: "<<pi<<endl;

    pi = bbp(epsilon);
    cout<<"BBP aproksimacija broja pi je: "<<pi<<endl;


    return 0;
}
```

Домаћи рад

1. Задатак: e^x

Написати програм који учитава реалне бројеве x и ϵ и са тачношћу ϵ израчунава и исписује суму

$$S = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \dots$$

За суму S се каже да је израчуната са тачношћу ϵ ако је апсолутна вредност последњег члана суме мања од ϵ.

Примјер извршавања

```
Unesite x: 4
Unesite epsilon: 0.001
S je: 54.5971
```

2. Задатак: Једнакост растојања

Становници једне дугачке улице желе да одреде положај на којем ће бити направљена антена за мобилну телефонију. Пошто желе да локацију одреде на најправеднији могући начин, договорили су се да антену саграде на мјесту на ком ће збир растојања свих оних који се налазе лијево од антене до ње, бити једнак збиру растојања свих оних који се налазе десно од антене до ње. Ако су познате координате свих кућа у улици (можемо замислити да су то координате тачака на једној правој), напиши програм који одређује положај антенте.