МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ государственное БЮДЖЕТНОЕ

образовательное учреждение

высшего образования

«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кафедра защиты информации

**

**ОТЧЁТ**

**по лабораторной работе № 4**

**«**Последовательные одномерные контейнеры**»**

**по дисциплине: «***Программирование***»**

Выполнил:Проверил:

Студент гр. «АБ-121», «АВТФ» *доцент кафедры ЗИ*

*Втюрин Александр Романович Архипова А. Б.*

«\_\_\_» \_\_\_\_\_\_ 20\_\_г«\_\_\_» \_\_\_\_\_\_ 2022 г.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись) (подпись)

Новосибирск 2022

**Цели и задачи работы:** изучение алгоритмов формирования и обработки одномерных массивов и последовательных контейнеров, программирование и отладка программ формирования и обработки массивов.

**Задание к работе:** Написать программу решения задачи в соответствии с индивидуальным вариантом.

**Методика выполнения работы:**

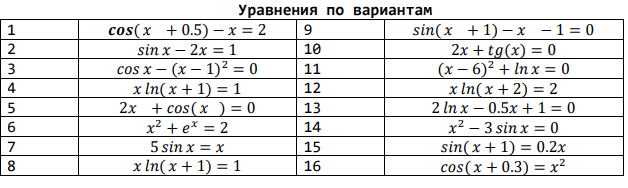
1. Разработать алгоритм решения задачи по индивидуальному заданию.

2. Написать и отладить программу решения задачи.

3. Протестировать работу программы на различных исходных данных

Задания к работе:

Задание 1: Отделите корни заданного уравнения, согласно варианту из табл.1, и уточните их одним из методов с точностью до ε=10-4 . Решить уравнения методом половинного деления, методом Ньютона и методом простых итераций. Либо другими методами, перечисленными в теоретической части к заданию 1.



Порядок выполнения работы

Отделите графически все корни уравнения.

1. Составьте методику решения нелинейного уравнения для метода половинного деления, метода Ньютона, метода простых итераций.

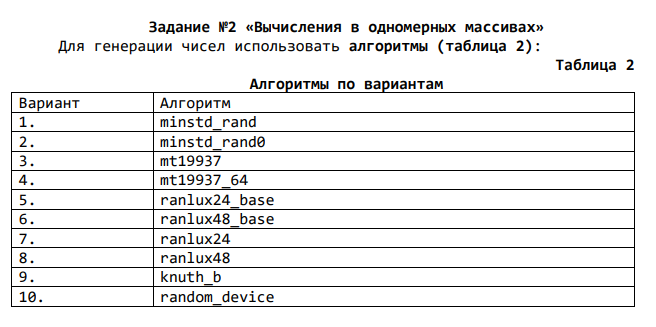
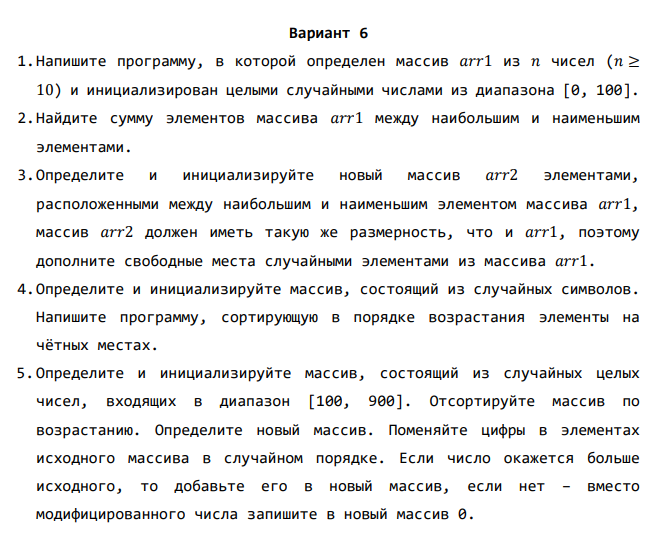
2. Составьте программу уточнения корня методом половинного деления с точностью до ε, выводящую результаты в таблицу

3. Составьте программу уточнения корня методом Ньютона и методом простых итераций с точностью до ε, выводящую результаты в таблицу:

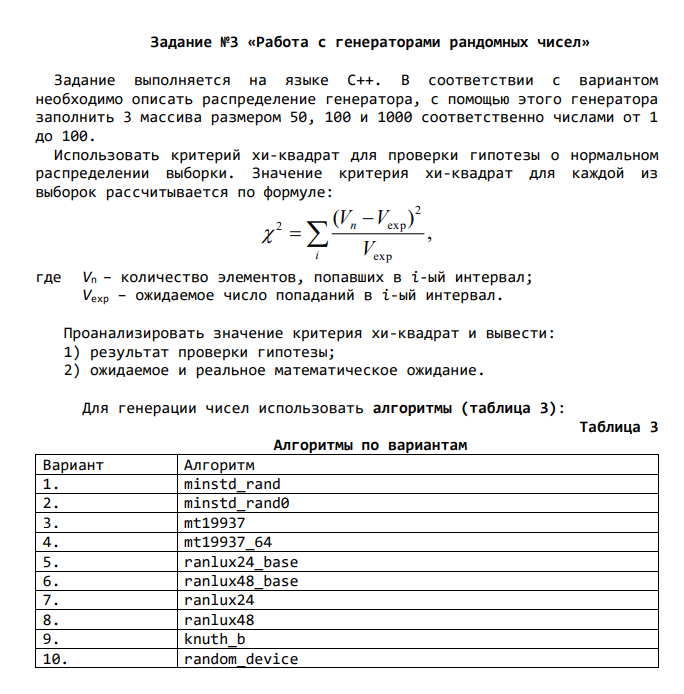
4. Найдите все корни уравнения и выпишите их с верными знаками.

5. Сравнить методы решения нелинейных уравнений по скорости сходимости

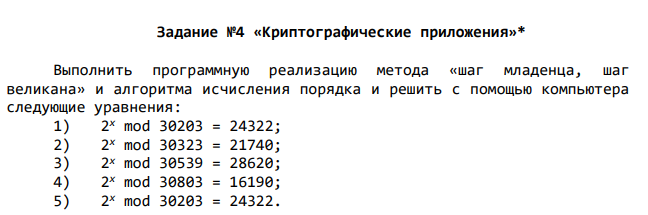
Задание 2:

Задание 3:



Задание 4:



**Теоретическая часть:**

Задание 1:

3. Метод половинного деления

Пусть дано нелинейное уравнение 𝑓(𝑥) = 0 и отделен простой корень х\*, т.е. найден такой отрезок [a0,b0], что x\*Є[a0,b0],и на концах отрезка функция имеет значения, противоположные по знаку (f(a0)\*f(b0)

Отрезок [a0,b0] называется начальным интервалом неопределенности, требуется уточнить местоположение корня уравнения с заданной точностью ε.

Процедура уточнения положения корня заключается в построении последовательности вложенных друг в друга отрезков, каждый из которых содержит корень уравнения. Для этого находится середина текущего интервала неопределенности ск=(ak+bk)/2, k=0,1,2,… и в качестве следующего интервала неопределенности из двух возможных выбирается тот, на концах которого функция f(х) имеет различные знаки.

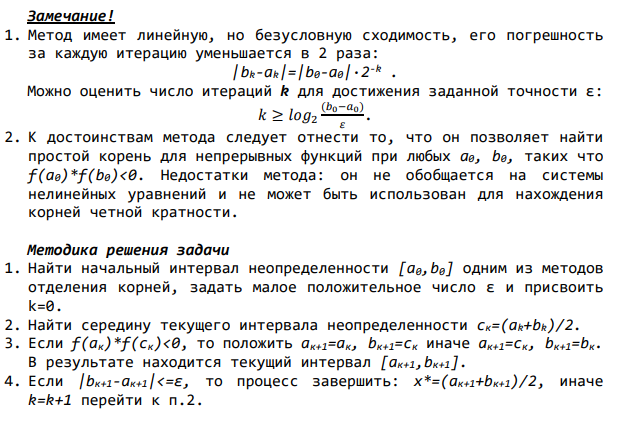
Процесс завершается, когда длина текущего интервала неопределенности становится меньше заданной величины ε, задающей точность нахождения корня. В качестве приближенного значения корня берется середина последнего интервала неопределенности.

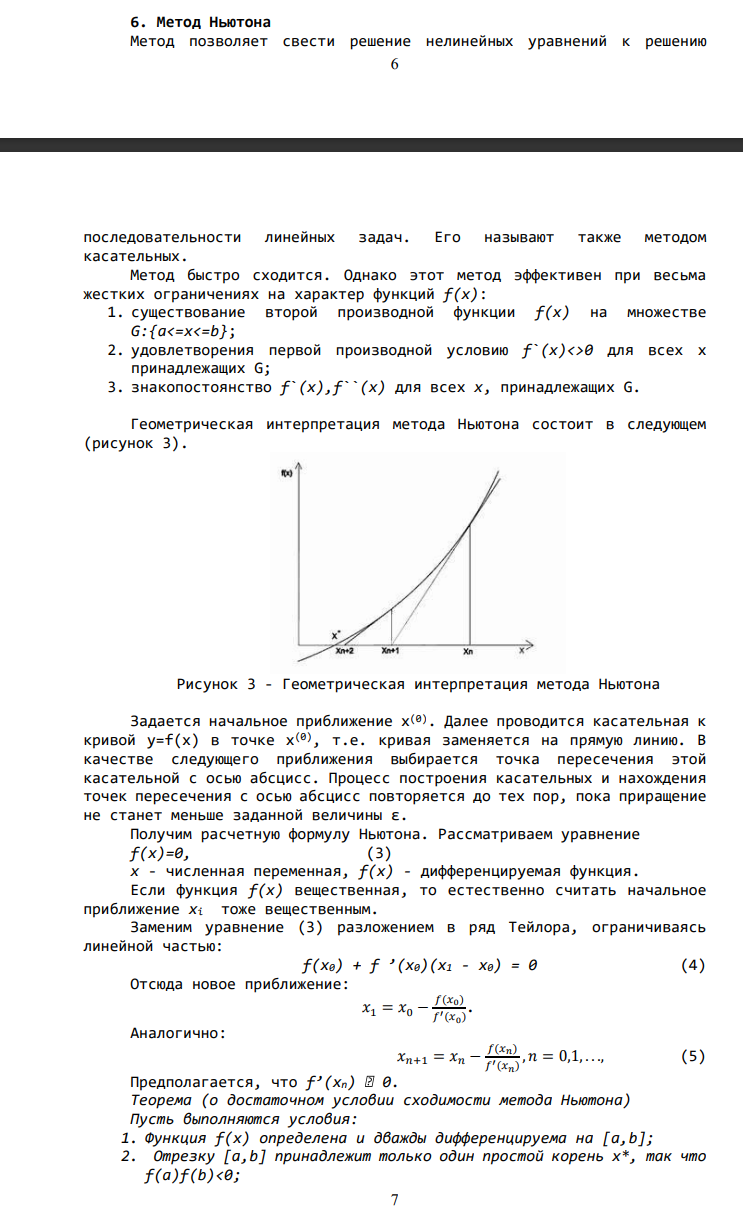
В основе метода половинного деления лежит теорема о вложенных отрезках. Последовательность отрезков

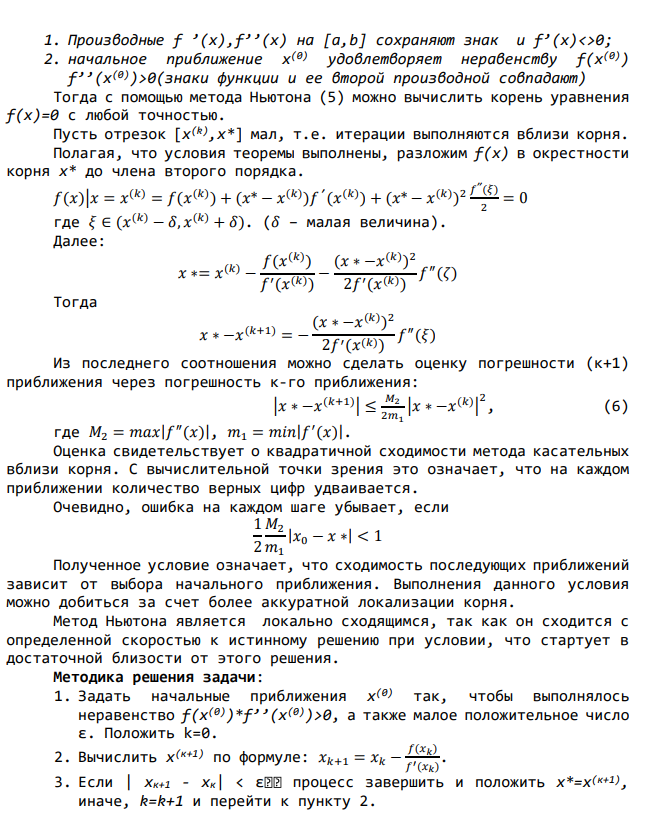
[𝑎1, 𝑏1] ⊃ [𝑎2, 𝑏2] ⊃. . . ⊃ [𝑎𝑛, 𝑏𝑛] ⊃. ..

называется вложенной. При условии, что длины отрезков |𝑏𝑛 − 𝑎𝑛| → 0 𝑛→∞ , эта последовательность называется стягивающейся.

Теорема Кантора. Для всякой стягивающейся последовательности вложенных отрезков существует единственная точка х\*, принадлежащая всем отрезкам этой последовательности.







Практическая часть:

**С++**

#define \_USE\_MATH\_DEFINES

#include <iostream>

#include <random>

#include <map>

#include <vector>

#include <cmath>

#include <math.h>

#include <algorithm>

#include <locale>

using namespace std;

void Task1pol(double a, double b, int q)

{

double c;

double d;

d = b - a;

cout.precision(4);

cout << q << " " << a << " " << b << " ";

cout << round(d) / 10000 << endl;

c = (a + b) / 2;

if (abs(a - b) < 0.0001) cout << "метод половинного деления " << c << endl;

else {

q++;

if ((pow(a, 2) + pow(M\_E, a) - 2) \* (pow(c, 2) + pow(M\_E, c) - 2) < 0) Task1pol(a, c, q);

if ((pow(c, 2) + pow(M\_E, c) - 2) \* (pow(c, b) + pow(M\_E, b) - 2) < 0) Task1pol(c, b, q);

}

}

double f(double x) {

return pow(x, 2) + pow(M\_E, x) - 2;

}

double f1(double x) {

return 2 \* x + pow(M\_E, x);

}

double f2(double x) {

return 2 + pow(M\_E, x);

}

void Task1nut(double a1, double b1) {

double a = a1;

double b = b1;

double x;

double e = 0.0001;

int k = 0;

if (f(a) \* f2(a) > 0) {

x = a;

}

else {

if (f(b) \* f2(b) > 0) {

x = b;

}

else {

x = -0.0001;

}

}

k = 0;

while (1) {

cout << k << " " << round(x) / 10000 << " " << round(x - f(x) / f1(x)) / 10000 << " " << round((x - x - f(x) / f1(x)) \* 10000) / 10000 << endl;

x = x - f(x) / f1(x);

k += 1;

if (fabs(f(x)) < e) {

break;

}

}

cout.precision(4);

cout << "Метод Ньютона " << x << endl;

}

void Task2()

{

ranlux48\_base gen(time(0));

cout << "1.исходный массив:" << endl;

vector<double> mas;

cout << "Введите кол-во элементов массива: ";

int n = 0;

cin >> n;

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

{

mas.push\_back(gen() % 101);

}

for (double i : mas)

{

cout << i << ' ';

}

cout << endl << endl;

cout << "2.среднее значение э-в массива:" << endl;

double sum = 0;

for (double x : mas)

{

sum += x;

}

double sred = sum / n;

cout << sred << endl << endl;

cout << "3.сумма квадратов разности:" << endl;

double sumKvadr = 0;

for (double x : mas)

{

sumKvadr += pow((x - sred), 2);

}

cout << sumKvadr << endl << endl;

cout << "4.числа с сменёнными разрядами:" << endl;

int poz = 0;

for (double x : mas)

{

if ((int)x % 2 == 0 && poz % 2 == 0)

{

int intX = x;

int a = intX % 10;

intX = intX / 10;

intX = intX + (a \* 10);

cout << intX << ' ';

}

poz++;

}

cout << endl << endl;

cout << "5.перенос первого числа массива в конец:" << endl;

double m = mas[0];

mas.erase(mas.cbegin());

mas.push\_back(m);

for (double i : mas)

{

cout << i << ' ';

}

cout << endl << endl;

cout << "6.подсчёт кол-ва конкретных элементов массива и удаление повторяющихся:" << endl;

vector<int> mas2;

cout << "Введите кол-во элементов массива: ";

cin >> n;

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

{

mas2.push\_back((gen() % 21) - 10);

}

cout << "Исходный массив: " << endl;

for (int i : mas2)

{

cout << i << ' ';

}

cout << endl << "Кол - во конкретных чисел в массиве: " << endl;

map<int, int> countNum;

for (int i : mas2)

{

if (countNum.find(i) != countNum.end())

{

countNum[i]++;

}

else

{

pair<int, int> m = { i,1 };

countNum.insert(m);

}

}

for (auto i : countNum)

{

cout << i.first << " = " << i.second << endl;

}

cout << "Массив после обработки: " << endl;

vector <int> mas2\_copy;

for (int i : mas2)

{

if (countNum[i] > 1)

{

countNum[i]--;

continue;

}

else

{

mas2\_copy.push\_back(i);

}

}

for (int i : mas2\_copy)

{

cout << i << ' ';

}

}

double XiKvadratCalk(map<double, double> mas)

{

double xn = 0, n = 0, xSred = 0;

for (auto x : mas)

{

xn = xn + (x.first \* x.second);

n++;

}

xSred = xn / n;

double xPromeg = 0, S = 0;

for (auto x : mas)

{

xPromeg = xPromeg + pow((xSred - x.first), 2) \* x.second;

}

S = sqrt(xPromeg / (n - 1));

double U = 0, funkU = 0, n0 = 0, XiKvadrat = 0;

for (auto x : mas)

{

U = (x.first - xSred) / S;

funkU = pow(M\_E, (pow(U, 2) / -2)) / sqrt(2 \* M\_PI);

n0 = (n \* funkU) / S;

XiKvadrat = XiKvadrat + pow((x.second - n0), 2) / n0;

}

return XiKvadrat;

}

void Task3()

{

ranlux48\_base gen(time(0));

int n1 = 50, n2 = 100, n3 = 1000;

map<double, double> mas50;

for (int i = 0; i < n1; i++)

{

mas50[(1 + gen() % 100)]++;

}

double XiKvadrat = XiKvadratCalk(mas50);

cout << "XiKvadrat первого массива: " << XiKvadrat << endl << "Критический XiKvadrat для заданных условий: " << 67.5 << endl;

if (XiKvadrat > 67.5)

{

cout << "Гипотеза ложна" << endl << endl;

}

else

{

cout << "Гипотеза верна" << endl << endl;

}

map<double, double> mas100;

for (int i = 0; i < n2; i++)

{

mas100[(1 + gen() % 100)]++;

}

XiKvadrat = XiKvadratCalk(mas100);

cout << "XiKvadrat второго массива: " << XiKvadrat << endl << "Критический XiKvadrat для заданных условий: " << 124.3 << endl;

if (XiKvadrat > 124.3)

{

cout << "Гипотеза ложна" << endl << endl;

}

else

{

cout << "Гипотеза верна" << endl << endl;

}

map<double, double> mas1000;

for (int i = 0; i < n3; i++)

{

mas1000[(1 + gen() % 100)]++;

}

XiKvadrat = XiKvadratCalk(mas1000);

cout << "XiKvadrat второго массива: " << XiKvadrat << endl << "Критический XiKvadrat для заданных условий: " << 1074.7 << endl;

if (XiKvadrat > 1074.7)

{

cout << "Гипотеза ложна" << endl << endl;

}

else

{

cout << "Гипотеза верна" << endl << endl;

}

}

int ryd1Gen(int a, int range, int p)

{

int genNum = 1;

for (int i = 1; i <= range; i++)

{

genNum = (a \* genNum) % p;

}

return genNum;

}

int ryd2Gen(int a, int y, int range, int p)

{

int genNum = y;

for (int i = 1; i <= range; i++)

{

genNum = (a \* genNum) % p;

}

return genNum;

}

void Task4()

{

//Шаг младенца шаг великана

// 2^x mod 30203 = 24322 a^x mod p = y

setlocale(LC\_ALL, "Rus");

int p = 30203, y = 24322, a = 2;

map <int, int> ryd1;

map <int, int> ryd2;

int m = sqrt(p) + 1;

for (int i = 1; i <= m; i++)

{

ryd1[i] = ryd1Gen(a, i \* m, p);

}

for (int i = 1; i <= m; i++)

{

ryd2[i] = ryd2Gen(a, y, i, p);

}

int i0 = 0, j0 = 0;

for (auto& i : ryd1)

{

for (auto& j : ryd2)

{

if (i.second == j.second)

{

i0 = i.first;

j0 = j.first;

}

}

}

cout << "Уравнения вида (2^x mod 30203 = 24322) имеет решение при x = " << i0 \* m - j0 << endl;

}

void DelitelCheck(int x, int p, int& BazMnog, int& StepenBazMnog) {

int l = 0;

for (int i = 1; pow(x, i) < p; i++)

{ //факторная база из первых 3 простых чисел

if (pow(2, i) == x)

{

BazMnog = 2;

StepenBazMnog = i;

}

}

for (int i = 1; pow(x, i) < p; i++)

{

if (pow(3, i) == x) {

BazMnog = 3;

StepenBazMnog = i;

}

}

for (int i = 1; pow(x, i) < p; i++)

{

if (pow(5, i) == x) {

BazMnog = 5;

StepenBazMnog = i;

}

}

}

int numGen1(int a, int x, int p)

{

int genNum = 1;

for (int i = 0; i < x; i++)

{

genNum = (a \* genNum) % p;

}

return genNum;

}

int numGen2(int a, int x, int p, int y)

{

int genNum = y;

for (int i = 0; i < x; i++)

{

genNum = (a \* genNum) % p;

}

return genNum;

}

void Task5()

{

//Алгоритм исчисления порядка

setlocale(LC\_ALL, "Rus");

int a = 2;

int p = 30203;

int y = 24322;

int x = 0, i = 1, BazMnog = 0, StepenBazMnog = 0;

while (BazMnog == 0) //по результатам цикл должен найти некотрый базовый множитель и степень такое что BazMnog^StepenBazMnog= y a^x mod p

{

x = numGen2(a, i++, p, y); //находится гладкое число вида ya^r

DelitelCheck(x, p, BazMnog, StepenBazMnog); //проверка на гладкость

}

int stepen = i;

i = 0;

int DelBazMnog = 0;

while (DelBazMnog == 0)

{

x = numGen1(a, ++i, p);

if (x == BazMnog)

{

DelBazMnog = i % p;

}

}

cout << "Уравнения вида (2^x mod 30203 = 24322) имеет решение при x = " << (DelBazMnog \* StepenBazMnog - stepen + 1) % (p - 1) << endl;

}

int main()

{

setlocale(LC\_ALL, "Rus");

/\* cout << "Task1:" << endl;

Task1pol(-10, 10, 0);

Task1nut(0, 1);

cout << "Task2:" << endl;

Task2();\*/

//cout << "\nTask3:" << endl;

//Task3();

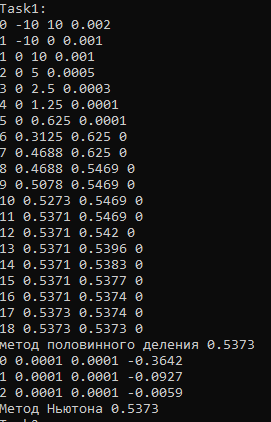
cout << "Task4:" << endl;

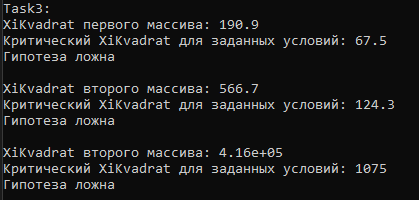
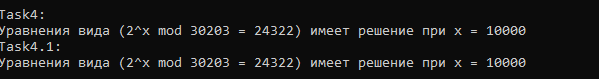
Task4();

cout << "Task4.1:" << endl;

Task5();

}

**** ****

**** ****

**С#**

using System;

using System.Collections.Generic;

public static class GlobalMembers

{

public static void Task1pol(double a, double b, int q)

{

double c;

double d;

d = b - a;

Console.Write("{0:4}", q);

Console.Write("{0:4}", " ");

Console.Write("{0:4}", a);

Console.Write("{0:4}", " ");

Console.Write("{0:4}", b);

Console.Write("{0:4}", " ");

Console.Write("{0:4}", Math.Round(d) / 10000);

Console.Write("{0:4}", "\n");

c = (a + b) / 2;

if (Math.Abs(a - b) < 0.0001)

{

Console.Write("{0:4}", "метод половинного деления ");

Console.Write("{0:4}", c);

Console.Write("{0:4}", "\n");

}

else

{

q++;

if ((Math.Pow(a, 2) + Math.Pow(Math.E, a) - 2) \* (Math.Pow(c, 2) + Math.Pow(Math.E, c) - 2) < 0)

{

Task1pol(a, c, q);

}

if ((Math.Pow(c, 2) + Math.Pow(Math.E, c) - 2) \* (Math.Pow(c, b) + Math.Pow(Math.E, b) - 2) < 0)

{

Task1pol(c, b, q);

}

}

}

public static double f(double x)

{

return Math.Pow(x, 2) + Math.Pow(Math.E, x) - 2;

}

public static double f1(double x)

{

return 2 \* x + Math.Pow(Math.E, x);

}

public static double f2(double x)

{

return 2 + Math.Pow(Math.E, x);

}

public static void Task1nut(double a1, double b1)

{

double a = a1;

double b = b1;

double x;

double e = 0.0001;

int k = 0;

if (f(a) \* f2(a) > 0)

{

x = a;

}

else

{

if (f(b) \* f2(b) > 0)

{

x = b;

}

else

{

x = -0.0001;

}

}

k = 0;

while (true)

{

Console.Write(k);

Console.Write(" ");

Console.Write(Math.Round(x) / 10000);

Console.Write(" ");

Console.Write(Math.Round(x - f(x) / f1(x)) / 10000);

Console.Write(" ");

Console.Write(Math.Round((x - x - f(x) / f1(x)) \* 10000) / 10000);

Console.Write("\n");

x = x - f(x) / f1(x);

k += 1;

if (Math.Abs(f(x)) < e)

{

break;

}

}

Console.Write("{0:4}", "Метод Ньютона ");

Console.Write("{0:4}", x);

Console.Write("{0:4}", "\n");

}

public static int ryd1Gen(int a, int range, int p)

{

int genNum = 1;

for (int i = 1; i <= range; i++)

{

genNum = (a \* genNum) % p;

}

return genNum;

}

public static int ryd2Gen(int a, int y, int range, int p)

{

int genNum = y;

for (int i = 1; i <= range; i++)

{

genNum = (a \* genNum) % p;

}

return genNum;

}

public static void Task4()

{

//Шаг младенца шаг великана

// 2^x mod 30203 = 24322 a^x mod p = y

int p = 30203;

int y = 24322;

int a = 2;

SortedDictionary<int, int> ryd1 = new SortedDictionary<int, int>();

SortedDictionary<int, int> ryd2 = new SortedDictionary<int, int>();

int m = Math.Sqrt(p) + 1;

for (int i = 1; i <= m; i++)

{

ryd1[i] = ryd1Gen(a, i \* m, p);

}

for (int i = 1; i <= m; i++)

{

ryd2[i] = ryd2Gen(a, y, i, p);

}

int i0 = 0;

int j0 = 0;

foreach (var i in ryd1)

{

foreach (var j in ryd2)

{

if (i.second == j.second)

{

i0 = i.first;

j0 = j.first;

}

}

}

Console.Write("Уравнения вида (2^x mod 30203 = 24322) имеет решение при x = ");

Console.Write(i0 \* m - j0);

Console.Write("\n");

}

public static void DelitelCheck(int x, int p, ref int BazMnog, ref int StepenBazMnog)

{

int l = 0;

for (int i = 1; Math.Pow(x, i) < p; i++)

{ //факторная база из первых 3 простых чисел

if (Math.Pow(2, i) == x)

{

BazMnog = 2;

StepenBazMnog = i;

}

}

for (int i = 1; Math.Pow(x, i) < p; i++)

{

if (Math.Pow(3, i) == x)

{

BazMnog = 3;

StepenBazMnog = i;

}

}

for (int i = 1; Math.Pow(x, i) < p; i++)

{

if (Math.Pow(5, i) == x)

{

BazMnog = 5;

StepenBazMnog = i;

}

}

}

public static int numGen1(int a, int x, int p)

{

int genNum = 1;

for (int i = 0; i < x; i++)

{

genNum = (a \* genNum) % p;

}

return genNum;

}

public static int numGen2(int a, int x, int p, int y)

{

int genNum = y;

for (int i = 0; i < x; i++)

{

genNum = (a \* genNum) % p;

}

return genNum;

}

public static void Task5()

{

//Алгоритм исчисления порядка

int a = 2;

int p = 30203;

int y = 24322;

int x = 0;

int i = 1;

int BazMnog = 0;

int StepenBazMnog = 0;

while (BazMnog == 0) //по результатам цикл должен найти некотрый базовый множитель и степень такое что BazMnog^StepenBazMnog= y a^x mod p

{

x = numGen2(a, i++, p, y); //находится гладкое число вида ya^r

DelitelCheck(x, p, ref BazMnog, ref StepenBazMnog); //проверка на гладкость

}

int stepen = i;

i = 0;

int DelBazMnog = 0;

while (DelBazMnog == 0)

{

x = numGen1(a, ++i, p);

if (x == BazMnog)

{

DelBazMnog = i % p;

}

}

Console.Write("Уравнения вида (2^x mod 30203 = 24322) имеет решение при x = ");

Console.Write((DelBazMnog \* StepenBazMnog - stepen + 1) % (p - 1));

Console.Write("\n");

}

static int Main()

{

Console.Write("Task1:");

Console.Write("\n");

Task1pol(-10, 10, 0);

Task1nut(0, 1);

Console.Write("Task4:");

Console.Write("\n");

Task4();

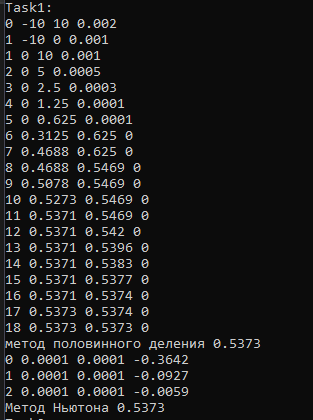
Console.Write("Task4.1:");

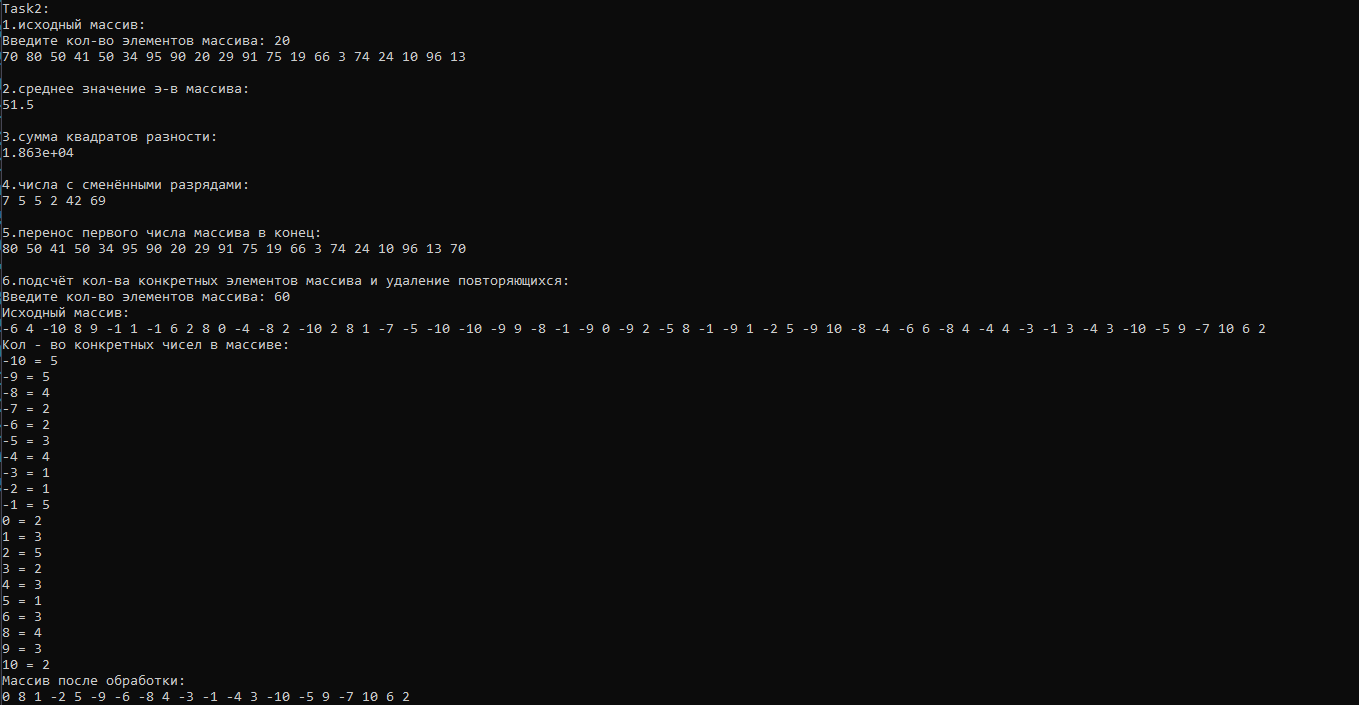
Console.Write("\n");

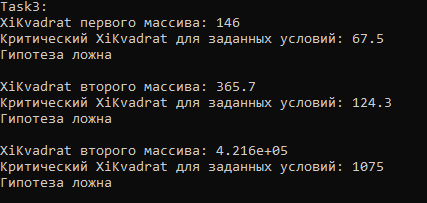
Task5();

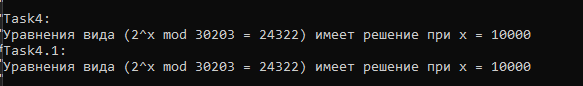
}

}

****

****

****

****

**Вывод:**

В ходе проделанной работы изучены Последовательные одномерные контейнеры. Реализовала циклический вычислительный процесс. Задачи решены в соответствии с индивидуальным вариантом. Поставленные задачи реализованы на двух языках программирования: C++ и C#.