МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ государственное БЮДЖЕТНОЕ

образовательное учреждение

высшего образования

«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

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

Кафедра защиты информации



**ЛАБОРАТОРНАЯ РАБОТА №4**

**«Последовательные одномерные контейнеры»**

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

Выполнил:Проверил:

Студент гр. «АБс-322», «АВТФ *Ассистент кафедры ЗИ*

*Батталов Игорь Павлович Исаев Г. А.*

«» 2024г«\_\_\_» \_\_\_\_\_\_ 2024 г.

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

(подпись) (подпись)

Новосибирск 2024

# Цель

Изучение алгоритмов формирования и обработки одномерных массивов и последовательных контейнеров, программирование и отладка программ формирования и обработки массивов.

# Задание

Написать программу решения задачи в соответствии с индивидуальным вариантом.

**Задание 1.** «Методы решения нелинейный уравнений»

**Задание 2.** «Вычисления в одномерных массивах».

1. Напишите программу, в которой определен массив 𝑎𝑟𝑟1 из 𝑛 чисел (15 ≤ 𝑛 ≤ 30) и инициализирован целыми случайными числами из диапазона [10, 30]. Определите массив 𝑎𝑟𝑟2 из 5 чисел и инициализируйте массив 𝑎𝑟𝑟2 целыми случайными числами из диапазона [10, 30].

2. Выведите количество чётных элементов массива 𝑎𝑟𝑟1 на нечётных местах, а также сами элементы. 15

3. Определите, какие числа из массива 𝑎𝑟𝑟2 встречаются в массиве 𝑎𝑟𝑟1. Выведите эти числа и сколько раз они встречаются в массиве 𝑎𝑟𝑟1. Если ни одно число не встречается, то выведите соответствующее сообщение.

4. Напишите программу, модифицирующую массив случайных символов размера 𝑛 , как в примере: [v,!,N,R,(,3] → [!,N,R,(,3,v].

5. Определите и инициализируйте массив, состоящий из случайных целых чисел, входящих в диапазон [1000, 9000]. Вычислите сумму элементов массива. Напишите программу, меняющую местами цифры в элементах массива в порядке возрастания (например, 8315 → 1358). Отсортируйте массив в порядке возрастания. Вычислите сумму элементов массива и сравните с суммой элементов исходного массива.

**Задание 3.** «Работа с генераторами рандомных чисел». Задание выполняется на языке C++. В соответствии с вариантом необходимо описать распределение генератора, с помощью этого генератора заполнить 3 массива размером 50, 100 и 1000 соответственно числами от 1 до 100.

Использовать критерий хи-квадрат для проверки гипотезы о нормальном распределении выборки.

Проанализировать значение критерия хи-квадрат и вывести: 1) результат проверки гипотезы; 2) ожидаемое и реальное математическое ожидание.

**Задание 4.** «Дилемма заключенного»\*. Требуется реализовать игру «Предать или сотрудничать» и реализовать 3 алгоритма поведения в игре.

**Задание 5.** «Генерация псевдослучайных чисел». Вариант 2 (Генератор псевдослучайных чисел на основе алгоритма BBS).

**Задание 1**

**С++**

**Листинг программы**

#include <iostream>

#include <vector>

#include <iomanip>

using namespace std;

double func(double x) {

return 5 \* sin(x) - x;

}

double kanonFunc(double x) {//Каноническое уравнение

return 5 \* sin(x);

}

double derivativeFunc(double x) {

return 5 \* cos(x) - 1;

}

void methodHalfDivision(vector<pair<double, double>> intervals, double E) {//Метод половинного деления

cout << "Метод половинного деления" << endl;

for (pair<double, double> interval : intervals) {

int N = 0;

cout << "N" << "\t" << "an" << "\t\t" << "bn" << "\t\t" << "bn - an" << endl;

while (true) {

double c = (interval.first + interval.second) / 2;

if ((func(interval.first) < 0 and func(c) > 0) or (func(interval.first) > 0 and func(c) < 0)) {

interval.second = c;

}

if ((func(interval.second) < 0 and func(c) > 0) or (func(interval.second) > 0 and func(c) < 0)) {

interval.first = c;

}

cout << ++N << "\t" << interval.first << "\t" << interval.second << "\t" << interval.second - interval.first << endl;

if (abs(interval.second - interval.first) < E) {

cout << "Корень: " << (interval.first + interval.second) / 2 << " " << endl;

break;

}

}

}

}

void newtonsMethod(vector<pair<double, double>> intervals, double E) {//Мето Ньютона

cout << "Метод Ньютона" << endl;

for (pair<double, double> interval : intervals) {

int N = 0;

cout << "N" << "\t" << "an" << "\t\t" << "bn" << "\t\t" << "bn - an" << endl;

while (true) {

interval.second = interval.first - func(interval.first) / derivativeFunc(interval.first);

if (abs(interval.second - interval.first) < E) {

cout << "Корень: " << interval.second << endl;

break;

}

cout << ++N << "\t" << interval.first << "\t" << interval.second << "\t" << interval.second - interval.first << endl;

interval.first = interval.second;

}

}

}

void newtonsModMethod(vector<pair<double, double>> intervals, double E) {//Модифицированный метод Ньютона

cout << "Модифицированный метод Ньютона" << endl;

for (pair<double, double> interval : intervals) {

int N = 0;

double x0 = interval.first;

cout << "N" << "\t" << "an" << "\t\t" << "bn" << "\t\t" << "bn - an" << endl;

while (true) {

interval.second = interval.first - func(interval.first) / derivativeFunc(x0);

if (abs(interval.second - interval.first) < E) {

cout << "Корень: " << interval.second << endl;

break;

}

cout << ++N << "\t" << interval.first << "\t" << interval.second << "\t" << interval.second - interval.first << endl;

interval.first = interval.second;

}

}

}

int main() {

setlocale(LC\_ALL, "RUSSIAN");

cout << fixed << setprecision(6);

vector<pair<double, double>> intervals{ { -3, -2}, {0.5, -1}, {2, 3} };

double E = 0.0001;

methodHalfDivision(intervals, E);

cout << "-----------------" << endl;

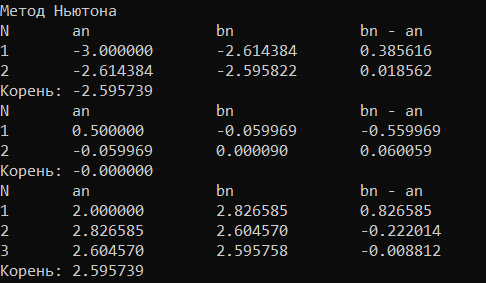
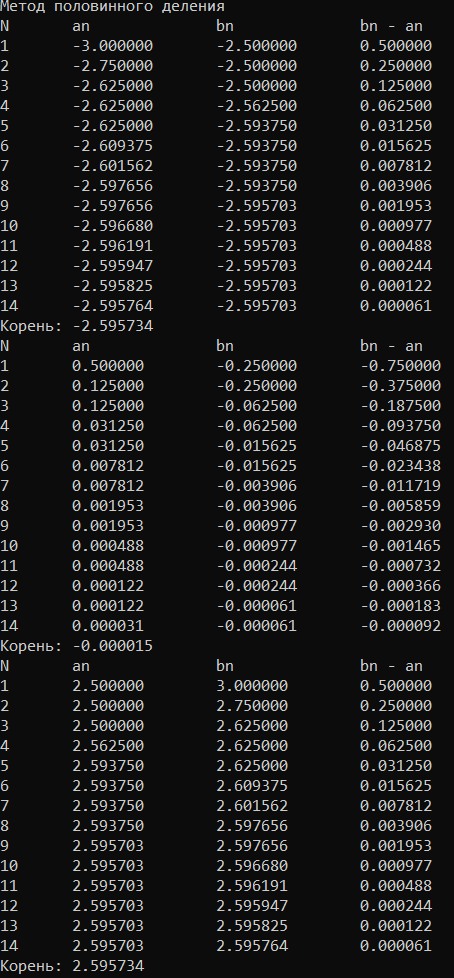
newtonsMethod(intervals, E);

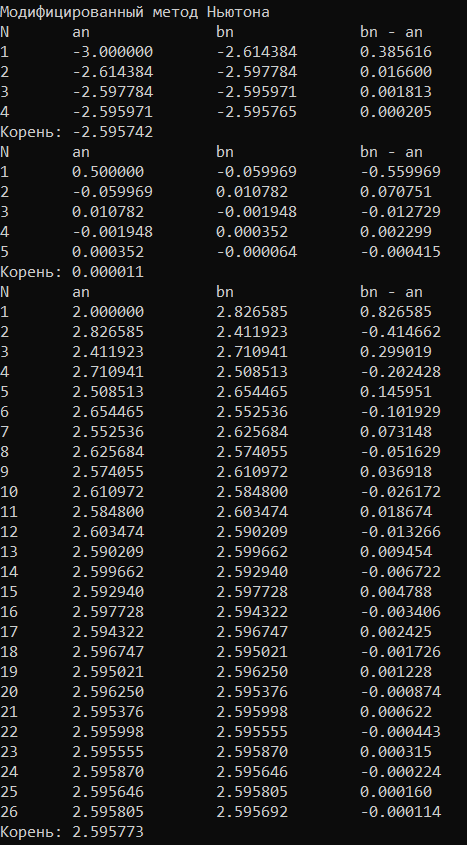
cout << "-----------------" << endl;

newtonsModMethod(intervals, E);

}

**Результат работы программы**

****

****

**С#**

**Листинг программы**

using System;

using System.Collections.Generic;

class Program

{

static double Func(double x)

{

return 5 \* Math.Sin(x) - x;

}

static double KanonFunc(double x)

{

return 5 \* Math.Sin(x);

}

static double DerivativeFunc(double x)

{

return 5 \* Math.Cos(x) - 1;

}

static void MethodHalfDivision(List<Tuple<double, double>> intervals, double E)

{

Console.WriteLine("Метод половинного деления");

foreach (Tuple<double, double> interval in intervals)

{

int N = 0;

Console.WriteLine("N" + "\t" + "an" + "\t" + "bn" + "\t" + "bn - an");

double It1 = interval.Item1;

double It2 = interval.Item2;

while (true)

{

double c = (It1 + It2) / 2;

if ((Func(It1) < 0 && Func(c) > 0) || (Func(It1) > 0 && Func(c) < 0))

{

It2 = c;

}

if ((Func(interval.Item2) < 0 && Func(c) > 0) || (Func(interval.Item2) > 0 && Func(c) < 0))

{

It1 = c;

}

Console.WriteLine(++N + "\t" + String.Format("{00:F4}", It1) + "\t" + String.Format("{00:F4}", It2) + "\t" + String.Format("{00:F4}", It2 - It1));

if (Math.Abs(It2 - It1) < E)

{

Console.WriteLine("Корень: " + String.Format("{00:F4}", (It1 + It2) / 2) + " ");

break;

}

}

}

}

static void NewtonsMethod(List<Tuple<double, double>> intervals, double E)

{

Console.WriteLine("Метод Ньютона");

foreach (Tuple<double, double> interval in intervals)

{

int N = 0;

Console.WriteLine("N" + "\t" + "an" + "\t" + "bn" + "\t" + "bn - an");

double It1 = interval.Item1;

double It2 = interval.Item2;

while (true)

{

It2 = It1 - Func(It1) / DerivativeFunc(It1);

if (Math.Abs(It2 - It1) < E)

{

Console.WriteLine("Корень: " + String.Format("{00:F4}", It2));

break;

}

Console.WriteLine(++N + "\t" + String.Format("{00:F4}", It1) + "\t" + String.Format("{00:F4}", It2) + "\t" + String.Format("{00:F4}", It2 - It1));

It1 = It2;

}

}

}

static void NewtonsModMethod(List<Tuple<double, double>> intervals, double E)

{

Console.WriteLine("Модифицированный метод Ньютона");

foreach (Tuple<double, double> interval in intervals)

{

int N = 0;

double x0 = interval.Item1;

Console.WriteLine("N" + "\t" + "an" + "\t" + "bn" + "\t" + "bn - an");

double It1 = interval.Item1;

double It2 = interval.Item2;

while (true)

{

It2 = It1 - Func(It1) / DerivativeFunc(x0);

if (Math.Abs(It2 - It1) < E)

{

Console.WriteLine("Корень: " + String.Format("{00:F4}", It2));

break;

}

Console.WriteLine(++N + "\t" + String.Format("{00:F4}", It1) + "\t" + String.Format("{00:F4}", It2) + "\t" + String.Format("{00:F4}", (It2 - It1)));

It1 = It2;

}

}

}

static void Main()

{

List<Tuple<double, double>> intervals = new List<Tuple<double, double>> { Tuple.Create(-3.0, -2.0), Tuple.Create(-0.5, 5.0), Tuple.Create(2.0, 3.0) };

double E = 0.0001;

MethodHalfDivision(intervals, E);

Console.WriteLine("-----------------");

NewtonsMethod(intervals, E);

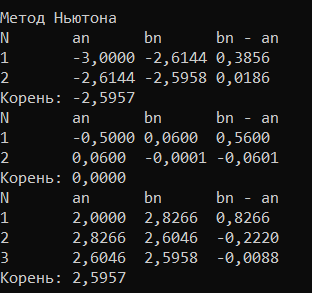
Console.WriteLine("-----------------");

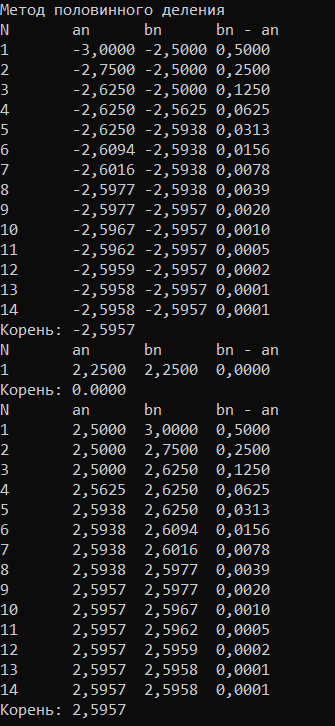
NewtonsModMethod(intervals, E);

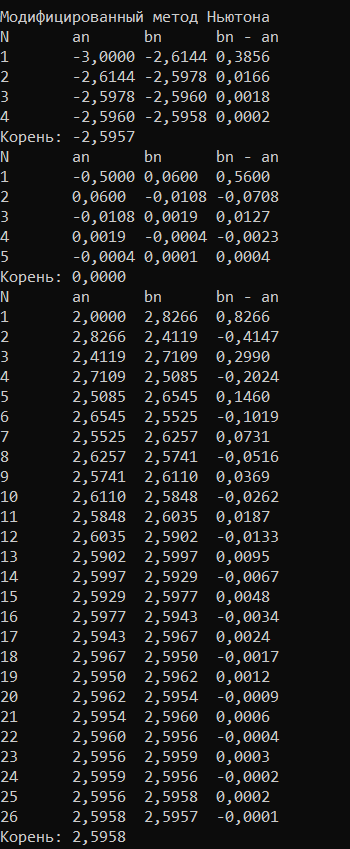
}

}

**Результат работы программы**

****

****

****

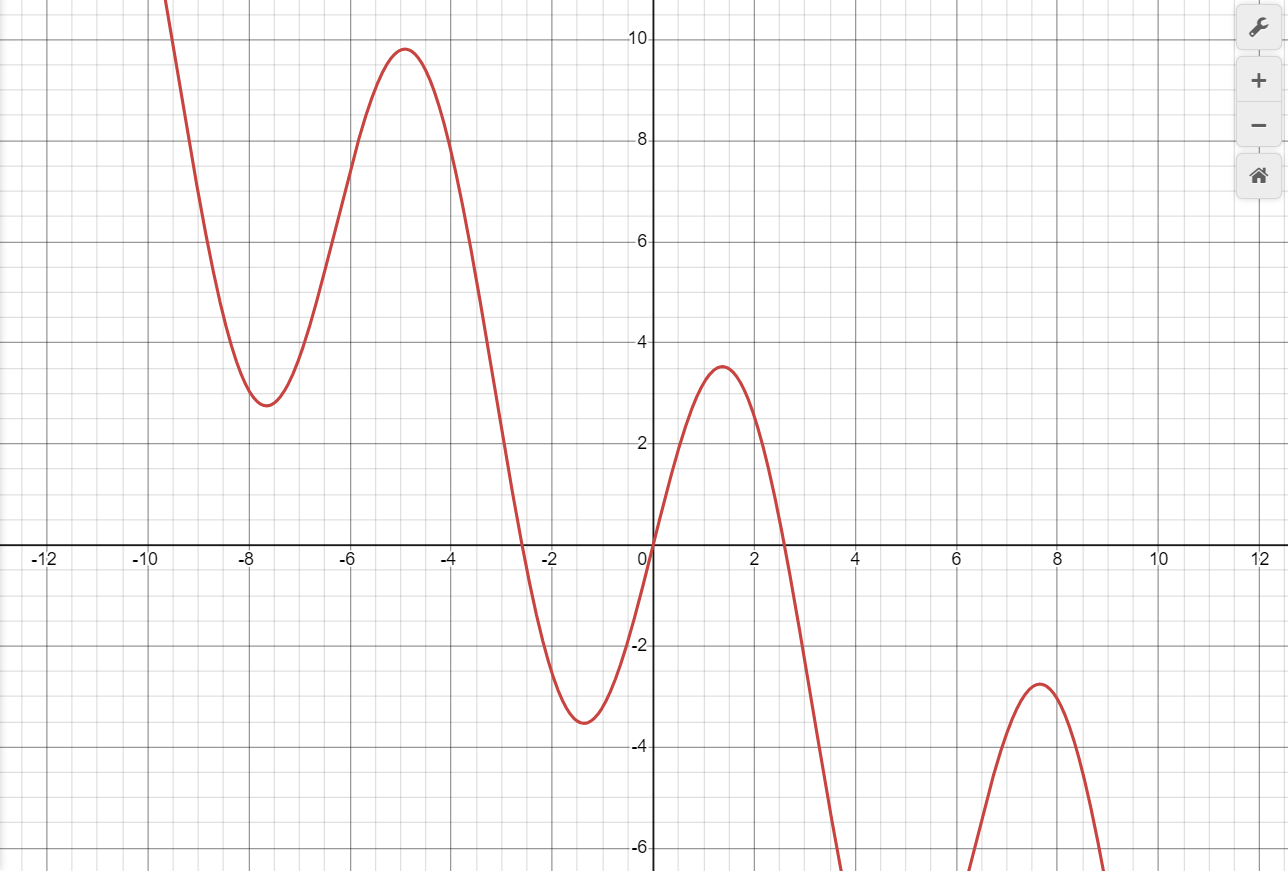
****

График функции 5 \* sin(x) – x = y

**Задание 2**

**С++**

**Листинг программы**

#include <iostream>

#include <random>

#include <vector>

using namespace std;

template<typename T>

ostream& operator<<(ostream& os, vector<T> arr) {

for (T element : arr) {

cout << element << " ";

}

return os;

}

void initializingArray(vector<int>& arr2, vector<int>& arr1, discard\_block\_engine<ranlux24\_base, 223, 23> generator, int n) {

for (; arr1.size() != n; ) {

int num = generator() % 30;

if (num >= 10) {

arr1.push\_back(num);

}

}

for (; arr2.size() != 5; ) {

int num = generator() % 30;

if (num >= 10) {

arr2.push\_back(num);

}

}

}

void initializingArray(vector<char>& arraySymbols, discard\_block\_engine<ranlux24\_base, 223, 23> generator) {

for (; arraySymbols.size() != 5; ) {

char num = generator() % 127;

if (num >= 33) {

arraySymbols.push\_back(num);

}

}

}

void initializingArray(vector<int>& arr4, discard\_block\_engine<ranlux24\_base, 223, 23> generator) {

for (; arr4.size() != 20; ) {

int num = generator() % 9000;

if (num >= 1000) {

arr4.push\_back(num);

}

}

}

void outputEvenNum(vector<int> arr1) {

int elementIndex = 0;

int countEvenElements = 0;

cout << "Четные числа на нечетных местах: ";

for (int element : arr1) {

if (elementIndex % 2 != 0 and element % 2 == 0) {

cout << element << " ";

countEvenElements++;

}

elementIndex++;

}

cout << "\n" << "Количество четных чисел на нечетных местах: " << countEvenElements << endl;

}

void comparingArrays(vector<int> arr1, vector<int> arr2) {

int countRepeatNum = 0;

for (int elementArr2 : arr2) {

int numRepetitions = 0;

for (int elementArr1 : arr1) {

if (elementArr1 == elementArr2) {

numRepetitions++;

}

}

if (numRepetitions) {

cout << "Число: " << elementArr2 << " повторяется " << numRepetitions << " раз" << endl;

countRepeatNum++;

}

}

if (countRepeatNum == 0) cout << "В массивах нет одинаковых элементов" << endl;

}

vector<char> modificationArray(vector<char> arraySymbols, int n) {

int firstElement = arraySymbols[0];

for (int i = 0; i < n - 1; i++) arraySymbols[i] = arraySymbols[i + 1];

arraySymbols[n - 1] = firstElement;

return arraySymbols;

}

int calculatingAmount(vector<int> arr4) {

int sum = 0;

for (int element : arr4) {

sum += element;

}

return sum;

}

void sort(vector<int>& figures, int n) {

for (int i = 0; i < n - 1; i++) {

int k = i;

for (int j = i + 1; j < n; j++) if (figures[j] < figures[k]) k = j;

int c = figures[k];

figures[k] = figures[i];

figures[i] = c;

}

}

void swapPlaces(vector<int>& arr4) {

for (int& element : arr4) {

vector<int> figures;

while (element > 0) {

figures.push\_back(element % 10);

element /= 10;

}

sort(figures, 4);

element = 0;

for (int figure : figures) {

element = element \* 10 + figure;

}

}

}

int main() {

setlocale(LC\_ALL, "RUSSIAN");

discard\_block\_engine<ranlux24\_base, 223, 23> generator (time(0));

int n;

cout << "Введите n" << endl;

cin >> n;

vector<int> arr1; vector<int> arr2; vector<char> arr3; vector<int> arr4;

initializingArray(arr2, arr1, generator, n);

initializingArray(arr3, generator);

initializingArray(arr4, generator);

cout << "Массив arr1: " << arr1 << "\nМассив arr2: " << arr2 << endl;

outputEvenNum(arr1);

comparingArrays(arr1, arr2);

cout << "Массив случайных символов: " << arr3 << endl;

cout << "Массив случайных символов после модификации: " << modificationArray(arr3, 5) << endl;

cout << "Массив arr4: " << arr4 << endl;

cout << "Сумма элементов массива arr4: " << calculatingAmount(arr4) << endl;

swapPlaces(arr4);

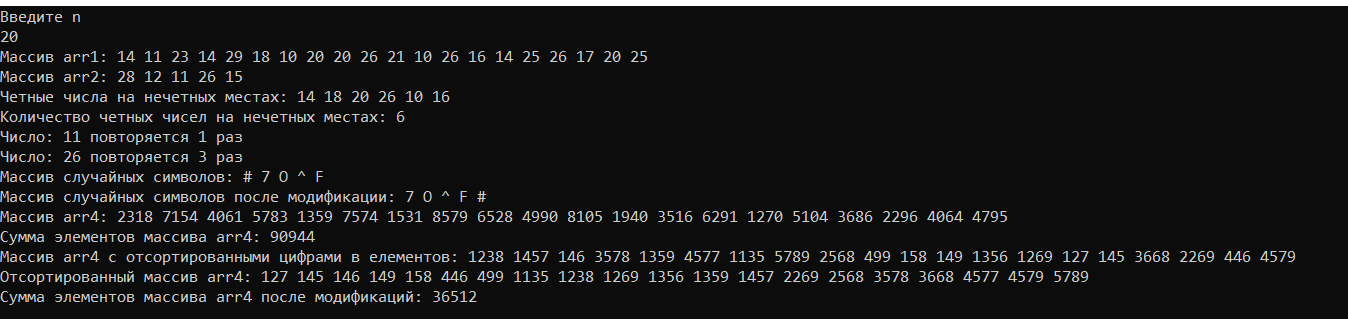
cout << "Массив arr4 с отсортированными цифрами в елементов: " << arr4 << endl;

sort(arr4, 20);

cout << "Отсортированный массив arr4: " << arr4 << endl;

cout << "Сумма элементов массива arr4 после модификаций: " << calculatingAmount(arr4) << endl;

}

**Результат работы программы**

**С#**

**Листинг программы**

using System;

using System.Collections.Generic;

namespace CodeConversion

{

class Program

{

static void Main(string[] args)

{

Random generator = new Random();

int n;

Console.WriteLine("Введите n");

n = Convert.ToInt32(Console.ReadLine());

List<int> arr1 = new List<int>();

List<int> arr2 = new List<int>();

List<char> arr3 = new List<char>();

List<int> arr4 = new List<int>();

InitializingArray(arr2, arr1, generator, n);

InitializingArray(arr3, generator);

InitializingArray(arr4, generator);

Console.WriteLine("Массив arr1: " + string.Join(" ", arr1));

Console.WriteLine("Массив arr2: " + string.Join(" ", arr2));

OutputEvenNum(arr1);

ComparingArrays(arr1, arr2);

Console.WriteLine("Массив случайных символов: " + string.Join(" ", arr3));

Console.WriteLine("Массив случайных символов после модификации: " + string.Join(" ", ModificationArray(arr3, 5)));

Console.WriteLine("Массив arr4: " + string.Join(" ", arr4));

Console.WriteLine("Сумма элементов массива arr4: " + CalculatingAmount(arr4));

SwapPlaces(arr4);

Console.WriteLine("Массив arr4 с отсортированными цифрами в элементах: " + string.Join(" ", arr4));

Sort(arr4, 20);

Console.WriteLine("Отсортированный массив arr4: " + string.Join(" ", arr4));

Console.WriteLine("Сумма элементов массива arr4 после модификаций: " + CalculatingAmount(arr4));

}

static void InitializingArray(List<int> arr2, List<int> arr1, Random generator, int n)

{

while (arr1.Count != n)

{

int num = generator.Next() % 30;

if (num >= 10)

{

arr1.Add(num);

}

}

while (arr2.Count != 5)

{

int num = generator.Next() % 30;

if (num >= 10)

{

arr2.Add(num);

}

}

}

static void InitializingArray(List<char> arraySymbols, Random generator)

{

while (arraySymbols.Count != 5)

{

char num = (char)(generator.Next() % 127);

if (num >= 33)

{

arraySymbols.Add(num);

}

}

}

static void InitializingArray(List<int> arr4, Random generator)

{

while (arr4.Count != 20)

{

int num = generator.Next() % 9000;

if (num >= 1000)

{

arr4.Add(num);

}

}

}

static void OutputEvenNum(List<int> arr1)

{

int elementIndex = 0;

int countEvenElements = 0;

Console.Write("Четные числа на нечетных местах: ");

foreach (int element in arr1)

{

if (elementIndex % 2 != 0 && element % 2 == 0)

{

Console.Write(element + " ");

countEvenElements++;

}

elementIndex++;

}

Console.WriteLine("\nКоличество четных чисел на нечетных местах: " + countEvenElements);

}

static void ComparingArrays(List<int> arr1, List<int> arr2)

{

int countRepeatNum = 0;

foreach (int elementArr2 in arr2)

{

int numRepetitions = 0;

foreach (int elementArr1 in arr1)

{

if (elementArr1 == elementArr2)

{

numRepetitions++;

}

}

if (numRepetitions > 0)

{

Console.WriteLine("Число: " + elementArr2 + " повторяется " + numRepetitions + " раз");

countRepeatNum++;

}

}

if (countRepeatNum == 0)

{

Console.WriteLine("В массивах нет одинаковых элементов");

}

}

static List<char> ModificationArray(List<char> arraySymbols, int n)

{

char firstElement = arraySymbols[0];

for (int i = 0; i < n - 1; i++)

{

arraySymbols[i] = arraySymbols[i + 1];

}

arraySymbols[n - 1] = firstElement;

return arraySymbols;

}

static int CalculatingAmount(List<int> arr4)

{

int sum = 0;

foreach (int element in arr4)

{

sum += element;

}

return sum;

}

static void Sort(List<int> figures, int n)

{

for (int i = 0; i < n - 1; i++)

{

int k = i;

for (int j = i + 1; j < n; j++)

{

if (figures[j] < figures[k])

{

k = j;

}

}

int c = figures[k];

figures[k] = figures[i];

figures[i] = c;

}

}

static void SwapPlaces(List<int> arr4)

{

for (int i = 0; i < arr4.Count; i++)

{

List<int> figures = new List<int>();

int element = arr4[i];

while (element > 0)

{

figures.Add(element % 10);

element /= 10;

}

Sort(figures, 4);

element = 0;

foreach (int figure in figures)

{

element = element \* 10 + figure;

}

arr4[i] = element;

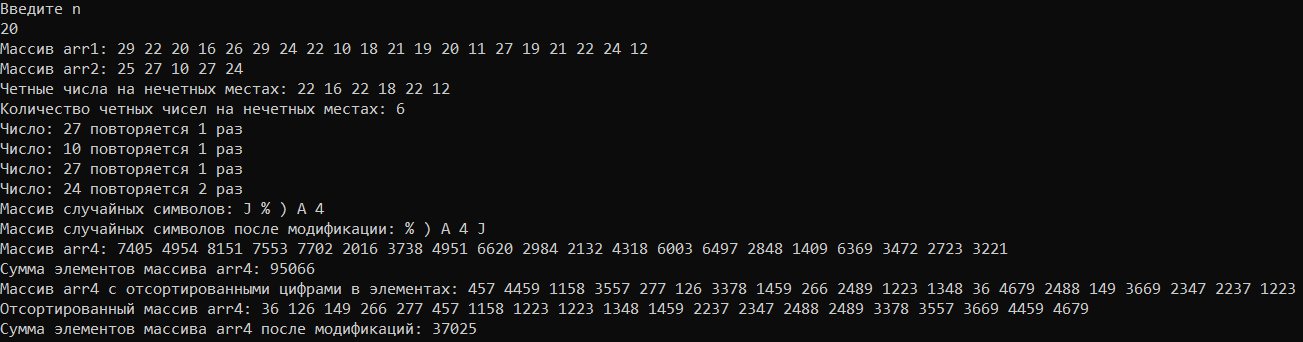
}

}

}

}

**Результат работы программы**

****

**Задание 3**

**С++**

**Листинг программы**

#include <iostream>

#include <vector>

#include <random>

#include <cmath>

using namespace std;

template<typename T>

ostream& operator<<(ostream& os, vector<T> vec) {

for (T element : vec) {

cout << element << " ";

}

return os;

}

void initializingArray(vector<int>& array, int size, discard\_block\_engine<ranlux24\_base, 223, 23> generator) {

for (int i = 0; i < size; i++) {

array.push\_back(generator() % 100);

}

}

int calculatingMin(vector<int> array) {

int minElement = 1000;

for (int element : array) {

if (element < minElement) minElement = element;

}

return minElement;

}

int calculatingMax(vector<int> array) {

int maxElement = 0;

for (int element : array) {

if (element > maxElement) maxElement = element;

}

return maxElement;

}

int arithmeticMean(vector<int> array, int size) {

int sum = 0;

for (int element: array) {

sum += element;

}

return sum / size;

}

int arithmeticMeanSquare(vector<int> array, int size) {

int sum = 0;

for (int element : array) {

sum += pow(element, 2);

}

return sum / size;

}

int standardDeviation(vector<int> array, int size) {

return sqrt(arithmeticMeanSquare(array, size) - arithmeticMean(array, size));

}

double normalCDF(double value)

{

return 0.5 \* erfc(-value \* sqrt(0.5));

}

vector<double> calculatingProbability(int widthIntervals, int amountIntervals, vector<int> array, int size,vector<pair<double, double>> Intervals) {

vector<double> probability;

Intervals[0].first = 1;

for (int i = 0; i <= amountIntervals; i++) {

double P;

if (i == 0) {

P = normalCDF((Intervals[i].second - arithmeticMean(array, size)) / standardDeviation(array, size));

}

else if (i == amountIntervals) {

P = normalCDF(1 - normalCDF((Intervals[i].first - arithmeticMean(array, size)) / standardDeviation(array, size)));

}

else {

P = normalCDF((Intervals[i].second - arithmeticMean(array, size)) / standardDeviation(array, size)) - normalCDF((Intervals[i].first - arithmeticMean(array, size)) / standardDeviation(array, size));

}

probability.push\_back(P \* size);

}

return probability;

}

double calculationX2(vector<double> Vn, vector<double> Vexp, int amountIntervals) {

double X2 = 0;

for (int i = 0; i <= amountIntervals; i++) {

X2 += pow(Vn[i] - Vexp[i], 2) / Vexp[i];

}

return X2;

}

double calculatingExpected(int degreeFreedom) {

if (degreeFreedom == 5) {

return 15.1;

}

if (degreeFreedom == 4) {

return 13.3;

}

if (degreeFreedom == 6) {

return 16.8;

}

if (degreeFreedom == 9) {

return 21.7;

}

}

void hypothesisTesting(double X2, double expect) {

if (X2 > expect) {

cout << "Гипотеза отвергается" << endl;

}

else {

cout << "Гипотеза не отвергается" << endl;

}

}

void distributionIntervals(int size, vector<int> array) {

int amountIntervals = 1 + log2(size);//Количество интервалов разбиения. Формула Стёрджесса

int widthIntervals = (calculatingMax(array) - calculatingMin(array)) / amountIntervals;//Ширина каждого интервала

vector<double> Vn;

vector<pair<double, double>> Intervals;

Intervals.push\_back({ 0, widthIntervals });

for (int i = 0; i <= amountIntervals; i++) {

double frequency = 0;

for (int element : array) {

if (element > Intervals[i].first and element <= Intervals[i].second) frequency++;

}

if (frequency < 5) {

Vn[i - 1] += frequency;

Intervals[i - 1].second += widthIntervals;

amountIntervals--;

}

else Vn.push\_back(frequency);

Intervals.push\_back({ Intervals[i].second, Intervals[i].second + widthIntervals });

}

cout << "Частота Vn: " << Vn << endl;

vector<double> probability = calculatingProbability(widthIntervals, amountIntervals, array, size, Intervals);

cout << "Частота Vexp: " << probability << endl;

cout << "Степень свободы: " << amountIntervals - 1 << endl;

double X2 = calculationX2(Vn, probability, amountIntervals);

double expect = calculatingExpected(amountIntervals - 1);

cout << "Ожидаемое: " << expect << endl;

cout << "Реальное: " << X2 << endl;

hypothesisTesting(X2, expect);

}

int main() {

setlocale(LC\_ALL, "RUSSIAN");

vector<int> arr50;

vector<int> arr100;

vector<int> arr1000;

discard\_block\_engine<ranlux24\_base, 223, 23> generator (time(0));

cout << "Массив 1" << "\nКоличество элементов: 50" << endl;

initializingArray(arr50, 50, generator);

distributionIntervals(50, arr50);

cout << "Массив 2" << "\nКоличество элементов: 100" << endl;

initializingArray(arr100, 100, generator);

distributionIntervals(100, arr100);

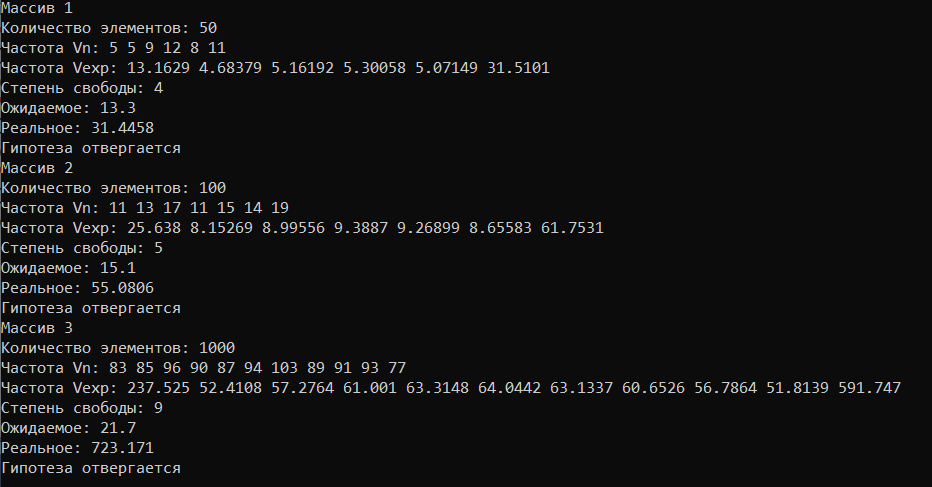
cout << "Массив 3" << "\nКоличество элементов: 1000" << endl;

initializingArray(arr1000, 1000, generator);

distributionIntervals(1000, arr1000);

}

**Результат работы программы**

****

Для проверки гипотезы о нормальном распределении сначала нужно найти количество интервалов разбиения по формуле Стёрджесса n = 1 + log2N, где N – размер массива. Чтобы найти ширину каждого интервала применим формулу Δ = Xmax – Xmin / n. Затем находим количество элементов попавших в i-ый интервал и ожидаемое число попаданий в i-ый интервал. Потом находим значение критерия хи-квадрат и сравниваем с таблицей распределения хи-квадрат.

**Задание 4**

**С++**

**Листинг программы**

#include <iostream>

#include <vector>

using namespace std;

void scoringPoints(int& pointFirstAlgorithm, int& pointSecondAlgorithm, vector<bool> firstAlgorithm, vector<bool> secondAlgorithm) {

for (size\_t i = 0; i < firstAlgorithm.size(); i++) {//Подсчет очков

if (firstAlgorithm[i] == true and secondAlgorithm[i] == true) {

pointFirstAlgorithm += 24;

pointSecondAlgorithm += 24;

}

if (firstAlgorithm[i] == false and secondAlgorithm[i] == false) {

pointFirstAlgorithm += 4;

pointSecondAlgorithm += 4;

}

if (firstAlgorithm[i] == false and secondAlgorithm[i] == true) {

pointFirstAlgorithm += 20;

}

if (firstAlgorithm[i] == true and secondAlgorithm[i] == false) {

pointSecondAlgorithm += 20;

}

}

}

bool algoritmOne(int round\_number, vector<bool> self\_choices, vector<bool> enemy\_choices) {

int count\_enemy\_false = 0;

int count\_enemy\_true = 0;

for (bool choices : enemy\_choices) {

if (choices) count\_enemy\_true += 1;

else count\_enemy\_false += 1;

}

if (count\_enemy\_false > count\_enemy\_true) return false;

if (count\_enemy\_false < count\_enemy\_true) return true;

if (count\_enemy\_false == count\_enemy\_true) return true;

}

bool algorithmTwo(int round\_number, vector<bool> self\_choices, vector<bool> enemy\_choices) {

int myPointAlgorithm = 0;

int enemyPointAlgorithm = 0;

scoringPoints(myPointAlgorithm, enemyPointAlgorithm, self\_choices, enemy\_choices);

if (round\_number < 75) {

if (myPointAlgorithm < enemyPointAlgorithm) return true;

if (myPointAlgorithm > enemyPointAlgorithm) return false;

if (myPointAlgorithm == enemyPointAlgorithm) return true;

}

else {

if (myPointAlgorithm < enemyPointAlgorithm) return false;

if (myPointAlgorithm > enemyPointAlgorithm) return true;

if (myPointAlgorithm == enemyPointAlgorithm) return false;

}

}

bool algorithmThree(int round\_number, vector<bool> self\_choices, vector<bool> enemy\_choices) {

if (round\_number % 2 == 0) return true;

if (round\_number % 3 == 0) return false;

if (round\_number % 25 == 0) return false;

if (round\_number % 100 == 0) return true;

return true;

}

int generateNumRounds() {

srand(time(0));

while (true) {

int numberRounds = rand() % 201;

if (numberRounds >= 100) {

cout << "Колво раундов: " << numberRounds << endl;

return numberRounds;

}

}

}

void battleOne(vector<bool>& firstAlgorithm, vector<bool>& secondAlgorithm) {

int numberRounds = generateNumRounds();

for (int i = 1; i < numberRounds; i++) {//Изменить колво раундов. Обозначить какой то переменной

firstAlgorithm.push\_back(algoritmOne(i, firstAlgorithm, secondAlgorithm));

secondAlgorithm.push\_back(algorithmTwo(i, secondAlgorithm, firstAlgorithm));

}

}

void battleTwo(vector<bool>& firstAlgorithm, vector<bool>& secondAlgorithm) {

int numberRounds = generateNumRounds();

for (int i = 1; i < numberRounds; i++) {//Изменить колво раундов. Обозначить какой то переменной

firstAlgorithm.push\_back(algorithmThree(i, firstAlgorithm, secondAlgorithm));

secondAlgorithm.push\_back(algorithmTwo(i, secondAlgorithm, firstAlgorithm));

}

}

void battleThree(vector<bool>& firstAlgorithm, vector<bool>& secondAlgorithm) {

int numberRounds = generateNumRounds();

for (int i = 1; i < numberRounds; i++) {//Изменить колво раундов. Обозначить какой то переменной

firstAlgorithm.push\_back(algorithmThree(i, firstAlgorithm, secondAlgorithm));

secondAlgorithm.push\_back(algoritmOne(i, secondAlgorithm, firstAlgorithm));

}

}

void battleAlgorithms(pair<int, int> algorithms) {

vector<bool> firstAlgorithm;

vector<bool> secondAlgorithm;

int pointFirstAlgorithm = 0;

int pointSecondAlgorithm = 0;

if (algorithms.first == 1 and algorithms.second == 2) battleOne(firstAlgorithm, secondAlgorithm);

if (algorithms.first == 2 and algorithms.second == 3) battleTwo(firstAlgorithm, secondAlgorithm);

if (algorithms.first == 1 and algorithms.second == 3) battleThree(firstAlgorithm, secondAlgorithm);

scoringPoints(pointFirstAlgorithm, pointSecondAlgorithm, firstAlgorithm, secondAlgorithm);

cout << "Очки первого алгоритма: " << pointFirstAlgorithm << "\nОчки второго алгоритма: " << pointSecondAlgorithm;

}

int main() {

setlocale(LC\_ALL, "RUSSIAN");

pair<int, int> algorithms;

cout << "Выберите 2 алгоритма" << endl;

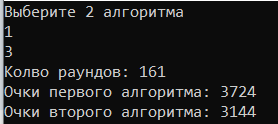
cin >> algorithms.first;

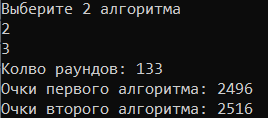
cin >> algorithms.second;

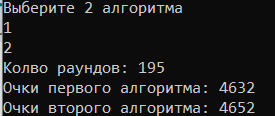
battleAlgorithms(algorithms);

}

**Результат работы программы**

****

****

****

**С#**

**Листинг программы**

using System;

using System.Collections.Generic;

class Program

{

static void ScoringPoints(ref int pointFirstAlgorithm, ref int pointSecondAlgorithm, List<bool> firstAlgorithm, List<bool> secondAlgorithm)

{

for (int i = 0; i < firstAlgorithm.Count; i++)

{

if (firstAlgorithm[i] && secondAlgorithm[i])

{

pointFirstAlgorithm += 24;

pointSecondAlgorithm += 24;

}

if (!firstAlgorithm[i] && !secondAlgorithm[i])

{

pointFirstAlgorithm += 4;

pointSecondAlgorithm += 4;

}

if (!firstAlgorithm[i] && secondAlgorithm[i])

{

pointFirstAlgorithm += 20;

}

if (firstAlgorithm[i] && !secondAlgorithm[i])

{

pointSecondAlgorithm += 20;

}

}

}

static bool AlgorithmOne(int roundNumber, List<bool> selfChoices, List<bool> enemyChoices)

{

int countEnemyFalse = 0;

int countEnemyTrue = 0;

foreach (bool choices in enemyChoices)

{

if (choices)

countEnemyTrue++;

else

countEnemyFalse++;

}

if (countEnemyFalse > countEnemyTrue)

return false;

if (countEnemyFalse < countEnemyTrue)

return true;

return true;

}

static bool AlgorithmTwo(int roundNumber, List<bool> selfChoices, List<bool> enemyChoices)

{

int myPointAlgorithm = 0;

int enemyPointAlgorithm = 0;

ScoringPoints(ref myPointAlgorithm, ref enemyPointAlgorithm, selfChoices, enemyChoices);

if (roundNumber < 75)

{

if (myPointAlgorithm < enemyPointAlgorithm)

return true;

if (myPointAlgorithm > enemyPointAlgorithm)

return false;

return true;

}

else

{

if (myPointAlgorithm < enemyPointAlgorithm)

return false;

if (myPointAlgorithm > enemyPointAlgorithm)

return true;

return false;

}

}

static bool AlgorithmThree(int roundNumber, List<bool> selfChoices, List<bool> enemyChoices)

{

if (roundNumber % 2 == 0)

return true;

if (roundNumber % 3 == 0 || roundNumber % 25 == 0)

return false;

if (roundNumber % 100 == 0)

return true;

return true;

}

static int GenerateNumRounds()

{

Random random = new Random();

while (true)

{

int numberRounds = random.Next(0, 201);

if (numberRounds >= 100)

{

Console.WriteLine("Количество раундов: " + numberRounds);

return numberRounds;

}

}

}

static void BattleOne(List<bool> firstAlgorithm, List<bool> secondAlgorithm)

{

int numberRounds = GenerateNumRounds();

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

{

firstAlgorithm.Add(AlgorithmOne(i, firstAlgorithm, secondAlgorithm));

secondAlgorithm.Add(AlgorithmTwo(i, secondAlgorithm, firstAlgorithm));

}

}

static void BattleTwo(List<bool> firstAlgorithm, List<bool> secondAlgorithm)

{

int numberRounds = GenerateNumRounds();

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

{

firstAlgorithm.Add(AlgorithmThree(i, firstAlgorithm, secondAlgorithm));

secondAlgorithm.Add(AlgorithmTwo(i, secondAlgorithm, firstAlgorithm));

}

}

static void BattleThree(List<bool> firstAlgorithm, List<bool> secondAlgorithm)

{

int numberRounds = GenerateNumRounds();

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

{

firstAlgorithm.Add(AlgorithmThree(i, firstAlgorithm, secondAlgorithm));

secondAlgorithm.Add(AlgorithmOne(i, secondAlgorithm, firstAlgorithm));

}

}

static void BattleAlgorithms((int, int) algorithms)

{

List<bool> firstAlgorithm = new List<bool>();

List<bool> secondAlgorithm = new List<bool>();

int pointFirstAlgorithm = 0;

int pointSecondAlgorithm = 0;

if (algorithms.Item1 == 1 && algorithms.Item2 == 2)

BattleOne(firstAlgorithm, secondAlgorithm);

if (algorithms.Item1 == 2 && algorithms.Item2 == 3)

BattleTwo(firstAlgorithm, secondAlgorithm);

if (algorithms.Item1 == 1 && algorithms.Item2 == 3)

BattleThree(firstAlgorithm, secondAlgorithm);

ScoringPoints(ref pointFirstAlgorithm, ref pointSecondAlgorithm, firstAlgorithm, secondAlgorithm);

Console.WriteLine("Очки первого алгоритма: " + pointFirstAlgorithm + "\nОчки второго алгоритма: " + pointSecondAlgorithm);

}

static void Main()

{

(int, int) algorithms;

Console.WriteLine("Выберите 2 алгоритма");

algorithms.Item1 = int.Parse(Console.ReadLine());

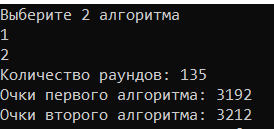
algorithms.Item2 = int.Parse(Console.ReadLine());

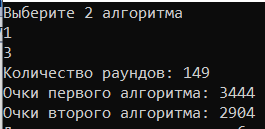
BattleAlgorithms(algorithms);

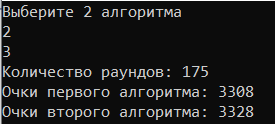
}

}

**Результат работы программы**

****

****

****

**Задание 5**

**С++**

**Листинг программы**

#include <iostream>

#include <vector>

using namespace std;

ostream& operator<<(ostream& os, vector<uint64\_t> array) {

int i = 0;

for (uint64\_t element : array) {

if (i % 10 == 0) cout << "\n";

cout << element << " ";

i++;

}

return os;

}

uint64\_t gcd(uint64\_t c, uint64\_t m) {//Нахождение НОД

if (m == 0) {

return c;

}

return gcd(m, c % m);

}

uint64\_t calculateRandValue(uint64\_t x, uint64\_t M) {//Нахождение случайного числа

return (x \* x) % M;

}

uint64\_t calculateX(uint64\_t M) {//Нахождение x

while (true) {

uint64\_t x = rand() % 10000;

if (gcd(M, x) == 1) return x;

}

}

void calculatePandQ(uint64\_t& p, uint64\_t& q) {//Нахождение p и q

while (100) {

p = rand() % 10000;

q = rand() % 10000;

if (p > 1000 and q > 1000) {

if (p % 4 == 3 and q % 4 == 3) break;

}

}

}

void generatingNumbers(uint64\_t M, vector<uint64\_t>& array) {//Генерация случайных чисел

uint64\_t previousX = calculateX(M);

for (int i = 0; i < 20; i++) {

uint64\_t xn = calculateRandValue(previousX, M);

previousX = xn;

array.push\_back(xn);

}

}

int main() {

setlocale(LC\_ALL, "RUSSIAN");

srand(time(0));

vector<uint64\_t> randNumbers;

uint64\_t p;

uint64\_t q;

calculatePandQ(p, q);

uint64\_t M = p \* q;

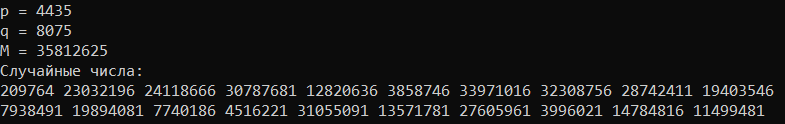
cout << "p = " << p << "\n" << "q = " << q << "\n"<< "M = "<< M << "\n" << "Случайные числа: ";

generatingNumbers(M, randNumbers);

cout << randNumbers;

}

**Результат работы программы**

****

**Вывод**

В ходе выполнения лабораторной работы были достигнуты поставленные цели и задачи. В процессе работы были изучены алгоритмы формирования и обработки одномерных массивов и последовательных контейнеров, а также осуществлено программирование и отладка соответствующих программ.