МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ государственное БЮДЖЕТНОЕ

образовательное учреждение

высшего образования

«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

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

Кафедра защиты информации

**

**ЛАБОРАТОРНАЯ РАБОТА №4**

**по дисциплине: «Программирование»**

**на тему: «**Последовательные одномерные контейнеры**»**

|  |  |
| --- | --- |
| Выполнил:  Студент гр. «АБ-420», «АВТФ»,  *Никифоров Максим Владимирович*  «29» апреля 2025г  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) | Проверил:  *Ассистент кафедры ЗИ*  *Исаев Глеб Андреевич*  «\_\_\_» \_\_\_\_\_\_ 2025г  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) |

Новосибирск 2025

Содержание

Оглавление

[ВВЕДЕНИЕ 3](#_Toc196863726)

[Ход выполнения 4](#_Toc196863727)

[Задание 1. 4](#_Toc196863728)

[Реализация на C++. 4](#_Toc196863729)

[Задание 2. 8](#_Toc196863730)

[Реализация на C++. 9](#_Toc196863731)

[Реализация на Go. 10](#_Toc196863732)

[Задание 3. 14](#_Toc196863733)

[Реализация на C++. 14](#_Toc196863734)

[Задание 4. 16](#_Toc196863735)

[Реализация на C++. 16](#_Toc196863736)

[Реализация на Go. 17](#_Toc196863737)

[Вывод 20](#_Toc196863738)

# ВВЕДЕНИЕ

**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.

# Ход выполнения

## Задание 1.

Вариант 5 – 2𝑙𝑛𝑥 – 0,5𝑥 + 1 = 0.

### Реализация на C++.

#include <iostream>  
#include <iomanip>  
#include <cmath>  
  
using namespace std;  
  
double f(double x) {  
 return 2\*log(x) - 0.5\*x + 1;  
}  
  
// Производная функции для метода Ньютона  
double df(double x) {  
 return (2 / x - 0.5);  
}  
  
// Функция для метода простых итераций  
double phi(double x) {  
 return x - 0.5 \* (2 \* log(x) - 0.5 \* x + 1);  
}  
  
// Метод половинного деления  
void bisection(double a, double b, double precision) {  
 cout << "Метод половинного деления:\n";  
 cout << "N\tan\t\t\tbn\t\t\tbn - an\n";  
 cout << "————————————————————————————————————\n";  
  
 int n = 0;  
 double c;  
 while (fabs(b - a) > precision) {  
 n++;  
 c = (a + b) / 2;  
 cout << n << "\t" << fixed << setprecision(6) << a << "\t" << b << "\t" << fabs(b - a) << endl;  
  
 if (f(c) == 0.0) {  
 break;  
 } else if (f(a) \* f(c) < 0) {  
 b = c;  
 } else {  
 a = c;  
 }  
 }  
  
 cout << "Найденный корень: " << ((a + b) / 2) << endl;  
 cout << "Значение в корне: " << f(((a + b) / 2)) << endl;  
 cout << "Количество итераций: " << n << "\n\n";  
}  
  
// Метод Ньютона  
void newton(double x0, double precision) {  
 cout << "Метод Ньютона:\n";  
 cout << "N\tXn\t\t\tXn+1\t\tXn+1 - Xn\n";  
 cout << "————————————————————————————————————\n";  
  
 int n = 0;  
 double x = x0;  
 double x\_prev;  
 double delta;  
  
 do {  
 n++;  
 x\_prev = x;  
 x = x\_prev - f(x\_prev) / df(x\_prev);  
 delta = fabs(x - x\_prev);  
 cout << n << "\t" << fixed << setprecision(6) << x\_prev << "\t" << x << "\t" << delta << endl;  
 } while (fabs(f(x)) > precision);  
  
 cout << "Найденный корень: " << x << endl;  
 cout << "Значение в корне: " << f(x) << endl;  
 cout << "Количество итераций: " << n << "\n\n";  
}  
  
// Метод простых итераций  
void simple\_iteration(double x0, double precision) {  
 cout << "Метод простых итераций:\n";  
 cout << "N\tXn\t\t\tXn+1\t\tXn+1 - Xn\n";  
 cout << "————————————————————————————————————\n";  
  
 int n = 0;  
 double x = x0;  
 double x\_prev;  
 double delta;  
  
 do {  
 n++;  
 x\_prev = x;  
 x = phi(x\_prev);  
 delta = fabs(x - x\_prev);  
 cout << n << "\t" << fixed << setprecision(6) << x\_prev << "\t" << x << "\t" << delta << endl;  
 } while (fabs(f(x)) > precision);  
  
 cout << "Найденный корень: " << x << endl;  
 cout << "Значение в корне: " << f(x) << endl;  
 cout << "Количество итераций: " << n << "\n";  
}  
  
int main() {  
 double precision = 0.0001; // требуемая точность  
 double a = 0.0, b = 1.0; // графически определили интервал  
 // Начальное приближение для Ньютона и простых итераций (середина интервала)  
 double x0 = (a+b)/2;  
  
 cout << "Решение уравнения 2𝑙𝑛𝑥 − 0.5𝑥 + 1 = 0 с точностью " << precision << endl;  
 bisection(a, b, precision);  
 newton(x0, precision);  
 simple\_iteration(x0, precision);  
  
 return 0;  
}

Реализация на Go.

package main

import (

"fmt"

"math"

)

// f(x) = 2ln(x) − 0.5x + 1

func f(x float64) float64 {

return 2\*math.Log(x) - 0.5\*x + 1

}

// Производная функции для метода Ньютона

func df(x float64) float64 {

return 2/x - 0.5

}

// Функция для метода простых итераций

func phi(x float64) float64 {

return x - 0.5\*(2\*math.Log(x)-0.5\*x+1)

}

// Метод половинного деления

func bisection(a, b, precision float64) {

fmt.Println("Метод половинного деления:")

fmt.Println("N\tan\t\tbn\t\tbn - an")

fmt.Println("————————————————————————————————————")

n := 0

var c float64

for math.Abs(b-a) > precision {

n++

c = (a + b) / 2

fmt.Printf("%d\t%.6f\t%.6f\t%.6f\n", n, a, b, math.Abs(b-a))

if f(c) == 0.0 {

break

} else if f(a)\*f(c) < 0 {

b = c

} else {

a = c

}

}

root := (a + b) / 2

fmt.Printf("Найденный корень: %.6f\n", root)

fmt.Printf("Значение в корне: %.6f\n", f(root))

fmt.Printf("Количество итераций: %d\n\n", n)

}

// Метод Ньютона

func newton(x0, precision float64) {

fmt.Println("Метод Ньютона:")

fmt.Println("N\tXn\t\tXn+1\t\tXn+1 - Xn")

fmt.Println("————————————————————————————————————")

n := 0

x := x0

var xPrev, delta float64

for {

n++

xPrev = x

x = xPrev - f(xPrev)/df(xPrev)

delta = math.Abs(x - xPrev)

fmt.Printf("%d\t%.6f\t%.6f\t%.6f\n", n, xPrev, x, delta)

if math.Abs(f(x)) <= precision {

break

}

}

fmt.Printf("Найденный корень: %.6f\n", x)

fmt.Printf("Значение в корне: %.6f\n", f(x))

fmt.Printf("Количество итераций: %d\n\n", n)

}

// Метод простых итераций

func simpleIteration(x0, precision float64) {

fmt.Println("Метод простых итераций:")

fmt.Println("N\tXn\t\tXn+1\t\tXn+1 - Xn")

fmt.Println("————————————————————————————————————")

n := 0

x := x0

var xPrev, delta float64

for {

n++

xPrev = x

x = phi(xPrev)

delta = math.Abs(x - xPrev)

fmt.Printf("%d\t%.6f\t%.6f\t%.6f\n", n, xPrev, x, delta)

if math.Abs(f(x)) <= precision {

break

}

}

fmt.Printf("Найденный корень: %.6f\n", x)

fmt.Printf("Значение в корне: %.6f\n", f(x))

fmt.Printf("Количество итераций: %d\n", n)

}

func main() {

precision := 0.0001 // требуемая точность

a, b := 0.0, 1.0 // графически определили интервал

// Начальное приближение для Ньютона и простых итераций (середина интервала)

x0 := (a + b) / 2

fmt.Printf("Решение 2ln(x) − 0.5x + 1 = 0 с точностью %.4f\n\n", precision)

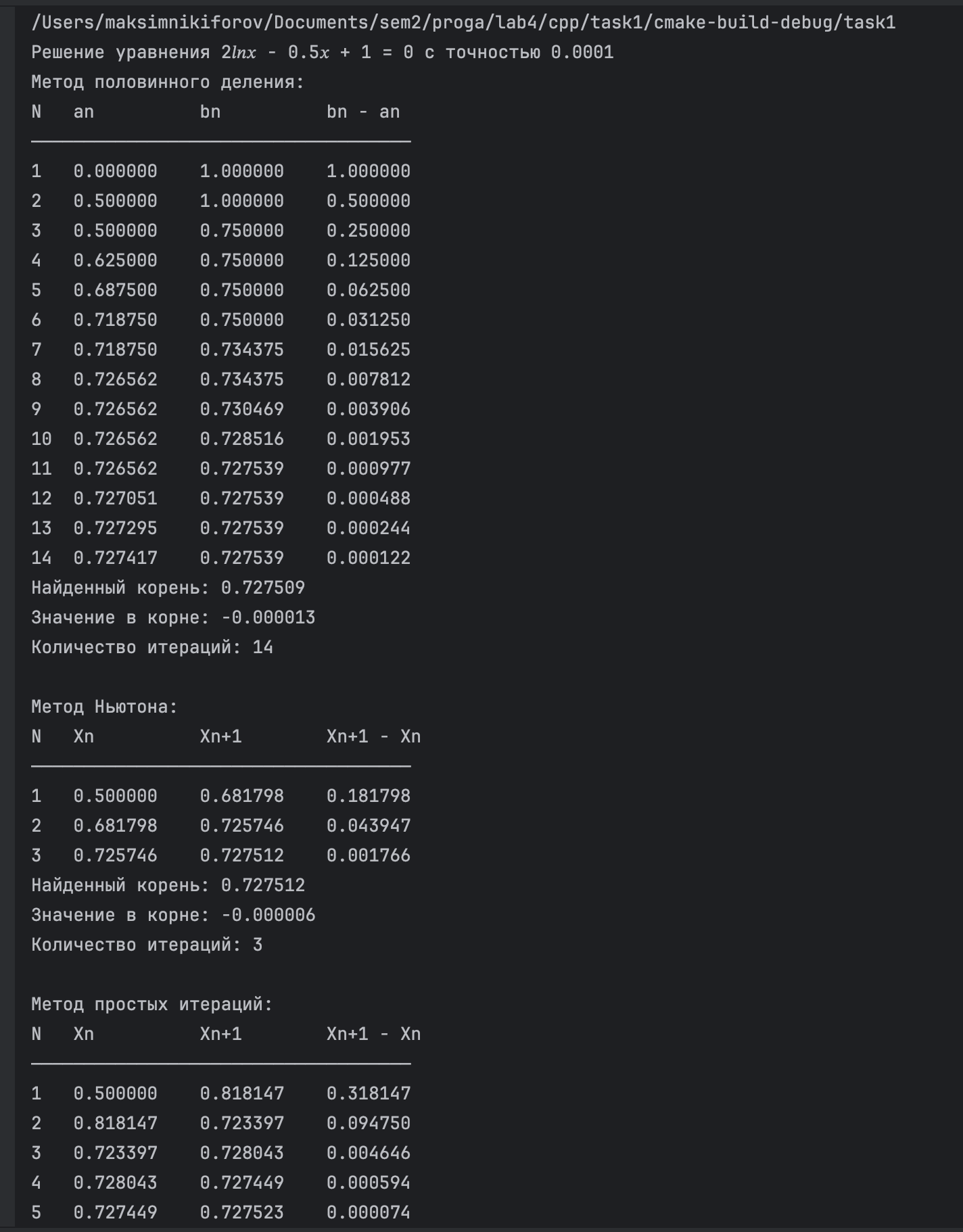
bisection(a, b, precision)

newton(x0, precision)

simpleIteration(x0, precision)

}

Тестирование работы программы. Рисунок 1.

Рисунок 1 – Результат работы

## Задание 2.

Алгоритм 1, вариант 6.

### Реализация на C++.

#include <iostream>  
#include <vector>  
#include <random>  
#include <algorithm>  
#include <iomanip>  
  
using namespace std;  
  
// генерация псевдослучайного числа в диапазоне min-max  
int random\_int(int min, int max) {  
 ranlux48 gen(random\_device{}());  
 uniform\_int\_distribution<int> dist(min, max);  
 return dist(gen);  
}  
  
void arrayPrint(int n, bool snake = false) {  
 cout << "\n";  
 int minVal = 100;  
  
 vector<vector<int>> arr(n, vector<int>(6));  
  
 // Заполнение массива  
 for (int i = 0; i < n; ++i) {  
 for (int j = 0; j < 6; ++j) {  
 arr[i][j] = minVal++;  
 }  
 }  
  
 // Вывод массива  
 for (int i = 0; i < n; ++i) {  
 if (snake && i % 2 == 0) {  
 for (int j = 6 - 1; j >= 0; --j)  
 cout << setw(4) << arr[i][j];  
 } else {  
 for (int j = 0; j < 6; ++j)  
 cout << setw(4) << arr[i][j];  
 }  
 cout << '\n';  
 }  
  
}  
  
int main() {  
 cout << "Введите количество чисел (≥10): ";  
 int n;  
 cin >> n;  
 vector<int> vec(n);  
 cout << "Исходный массив: ";  
  
 int posNumAmount = 0;  
 for (int& num : vec) {  
 num = random\_int(-100, 100);  
 cout << num << " ";  
 if (num > 0) { posNumAmount+=1; }  
 }  
 cout << "\nПоложительных элементов: " << posNumAmount << endl;  
  
 auto it = find(vec.rbegin(), vec.rend(), 0);  
 if (it != vec.rend()) {  
 int sumAftZero = accumulate(it.base(), vec.end(), 0);  
 cout << sumAftZero << "\n";  
 } else {  
 cout << "В массиве нет нуля!\n";  
 }  
  
 std::vector<int> current;  
 std::vector<int> longest;  
 int prev = -101;  
  
 for (int val : vec) {  
 if (current.empty() || val > prev) {  
 current.push\_back(val);  
 } else {  
 if (current.size() > longest.size()) {  
 longest = current;  
 }  
 current = {val};  
 }  
 prev = val;  
 }  
  
 if (current.size() > longest.size()) { longest = current; }  
  
 cout << "Самая длинная возрастающая последовательность: ";  
 for (int x : longest) {  
 cout << x << " ";  
 }  
  
 arrayPrint(n, false);  
 arrayPrint(n, true);  
  
 return 0;  
}

### Реализация на Go.

package main

import (

"fmt"

"math/rand"

)

// генерация псевдослучайного числа в диапазоне min-max

func randomInt(min, max int) int {

return rand.Intn(max-min+1) + min

}

func arrayPrint(n int, snake bool) {

fmt.Println()

minVal := 100

arr := make([][]int, n)

for i := range arr {

arr[i] = make([]int, 6)

}

// Заполнение массива

for i := 0; i < n; i++ {

for j := 0; j < 6; j++ {

arr[i][j] = minVal

minVal++

}

}

// Вывод массива

for i := 0; i < n; i++ {

if snake && i%2 == 0 {

for j := 5; j >= 0; j-- {

fmt.Printf("%4d", arr[i][j])

}

} else {

for j := 0; j < 6; j++ {

fmt.Printf("%4d", arr[i][j])

}

}

fmt.Println()

}

}

func main() {

var n int

fmt.Print("Введите количество чисел (≥10): ")

fmt.Scan(&n)

vec := make([]int, n)

fmt.Print("Исходный массив: ")

posNumAmount := 0

for i := range vec {

vec[i] = randomInt(-100, 100)

fmt.Print(vec[i], " ")

if vec[i] > 0 {

posNumAmount++

}

}

fmt.Printf("\nПоложительных элементов: %d\n", posNumAmount)

// Поиск последнего нуля (с конца) и суммирование после него

foundZero := false

sumAftZero := 0

for i := n - 1; i >= 0; i-- {

if vec[i] == 0 {

foundZero = true

for j := i + 1; j < n; j++ {

sumAftZero += vec[j]

}

break

}

}

if foundZero {

fmt.Printf("%d\n", sumAftZero)

} else {

fmt.Println("В массиве нет нуля!")

}

// Поиск самой длинной возрастающей подпоследовательности

current := []int{}

longest := []int{}

prev := -101

for \_, val := range vec {

if len(current) == 0 || val > prev {

current = append(current, val)

} else {

if len(current) > len(longest) {

longest = append([]int(nil), current...)

}

current = []int{val}

}

prev = val

}

if len(current) > len(longest) {

longest = current

}

fmt.Print("Самая длинная возрастающая последовательность: ")

for \_, x := range longest {

fmt.Print(x, " ")

}

fmt.Println()

arrayPrint(n, false)

arrayPrint(n, true)

}

Тестирование работы программы. Рисунок 2.

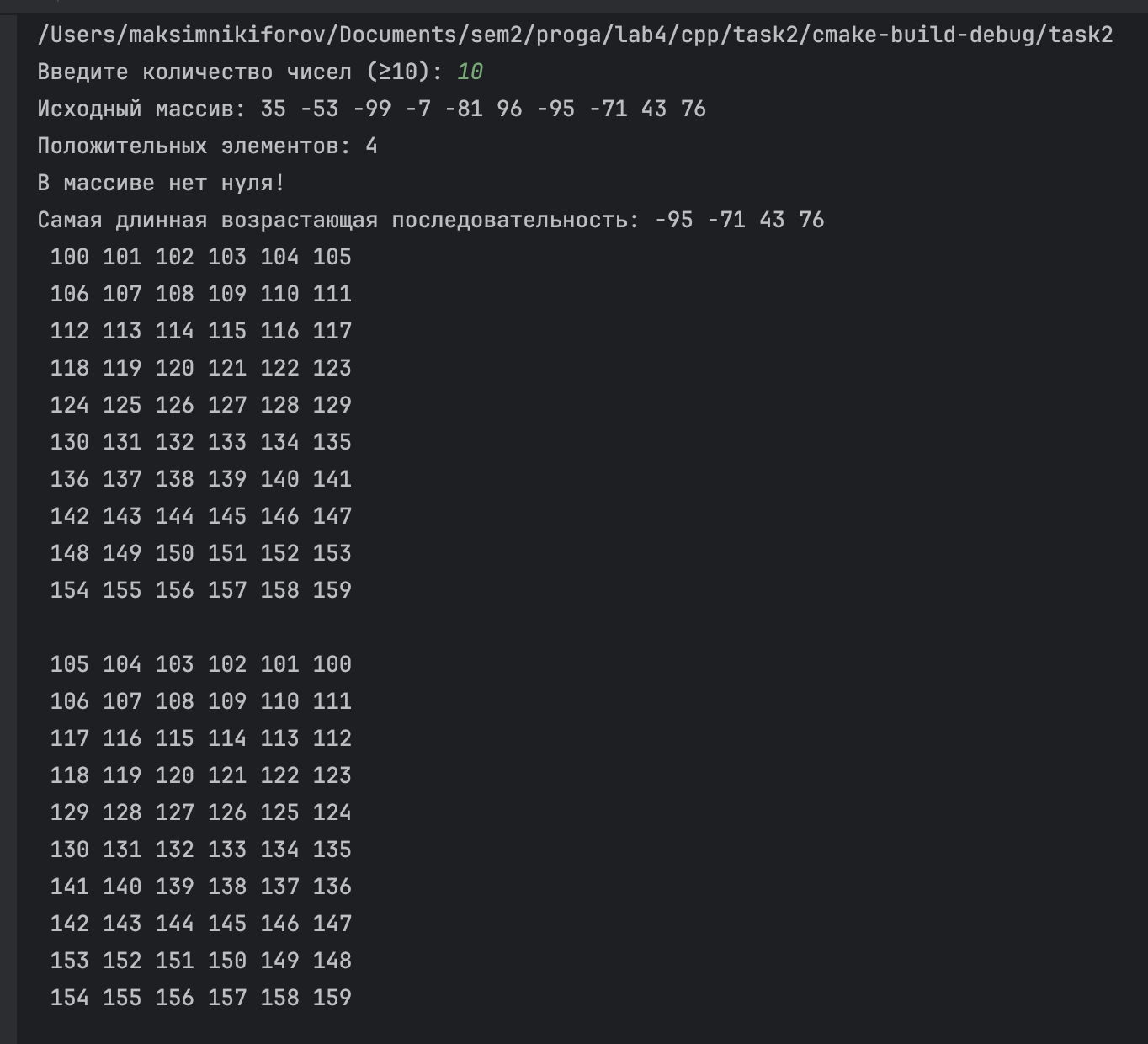


Рисунок 2 – Результат работы программы

## Задание 3.

Вариант 1.

### Реализация на C++.

#include <iostream>  
#include <vector>  
#include <cstdint>  
  
using namespace std;  
  
void initRc4(vector<int>& S, const vector<int>& K, int n) {  
 int N = pow(2,n); // 2^n  
 S.resize(N);  
  
 for (int i = 0; i < N; ++i) {  
 S[i] = i;  
 }  
  
 int j = 0;  
 for (int i = 0; i < N; ++i) {  
 j = (j + S[i] + K[i % K.size()]) % N;  
 swap(S[i], S[j]);  
 }  
}  
  
vector<int> genRc4(const vector<int>& S\_init, int n, int num\_bytes) {  
 int N = pow(2,n); // 2^n  
 vector<int> S = S\_init;  
 vector<int> output;  
 output.reserve(num\_bytes);  
  
 int i = 0, j = 0;  
  
 for (int count = 0; count < num\_bytes; ++count) {  
 i = (i + 1) % N;  
 j = (j + S[i]) % N;  
 swap(S[i], S[j]);  
 int t = (S[i] + S[j]) % N;  
 int K = S[t];  
 output.push\_back(K);  
 }  
  
 return output;  
}  
  
int main() {  
 int n = 8;  
 vector<int> K = {1, 2, 3, 4, 5}; // Пример ключа  
  
 vector<int> S;  
 initRc4(S, K, n);  
  
 int num\_bytes = 5;  
 vector<int> keystream = genRc4(S, n, num\_bytes);  
  
 for (int byte : keystream) {  
 cout << byte << " ";  
 }  
 cout << endl;  
  
 return 0;  
}

Тестирование работы программы. Рисунок 3.

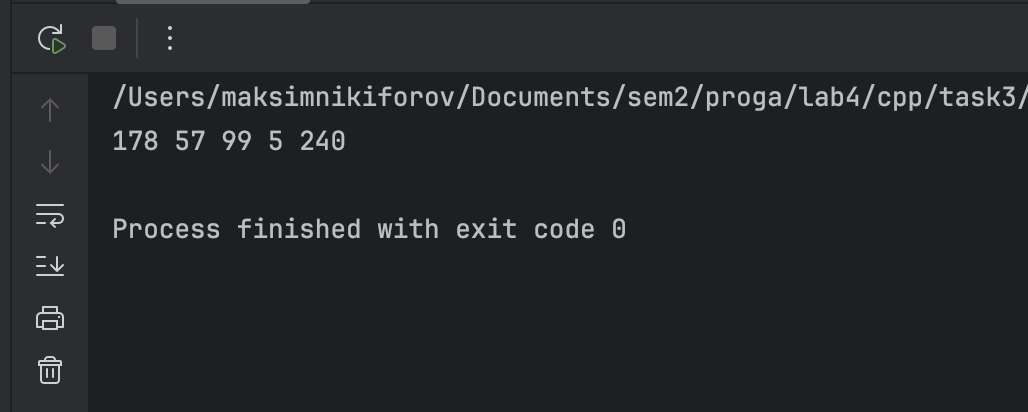


Рисунок 3 – Результат генерации чисел

## Задание 4.

### Реализация на C++.

#include <iostream>  
#include <vector>  
#include <map>  
#include <algorithm>  
#include <string>  
  
using namespace std;  
  
string findBorda(const vector<string>& candidates, const vector<vector<string>>& votes) {  
 map<string, int> scores;  
  
 for (const auto& vote : votes) {  
 for (int i = 0; i < vote.size(); ++i) {  
 scores[vote[i]] += (candidates.size() - 1 - i);  
 }  
 }  
  
 // Находим максимальный балл  
 int maxScore = max\_element(  
 scores.begin(), scores.end(),  
 [](const auto& a, const auto& b) {  
 return a.second < b.second;  
 }  
 )->second;  
  
 // Собираем всех кандидатов с этим баллом  
 vector<string> winners;  
 for (const auto& [candidate, score] : scores) {  
 if (score == maxScore) {  
 winners.push\_back(candidate);  
 }  
 }  
  
 return winners.size() == 1 ? winners[0] : "Ничья между несколькими кандидатами";  
}  
  
string findCondorce(const vector<string>& candidates, const vector<vector<string>>& votes) {  
 map<string, map<string, int>> pairwise;  
  
 // заполнение матрицы сравнений   
 for (const auto& vote : votes) {  
 for (int i = 0; i < vote.size(); ++i) {  
 for (int j = i + 1; j < vote.size(); ++j) {  
 pairwise[vote[i]][vote[j]]++;  
 }  
 }  
 }  
  
 // Поиск на победителя Кондорсе  
 for (const auto& candidate : candidates) {  
 bool isWinner = true;  
  
 for (const auto& opponent : candidates) {  
 if (candidate == opponent) continue;  
  
 if (pairwise[candidate][opponent] <= pairwise[opponent][candidate]) {  
 isWinner = false;  
 break;  
 }  
 }  
  
 if (isWinner) return candidate;  
 }  
  
 return "Победитель Кондорсе не найден";  
}  
  
int main() {  
 vector<string> candidates = {"a", "b", "c"};  
 vector<vector<string>> votes = {  
 {"a", "c", "b"},  
 {"a", "c", "b"},  
 {"c", "b", "a"},  
 };  
  
 string winBorda = findBorda(candidates, votes);  
 string winConrorce = findCondorce(candidates, votes);  
  
 cout << "Победитель по Борда: " << winBorda << "\n";  
 cout << "Победитель по Кондорсе: " << winConrorce;  
 if (winBorda != winConrorce)  
 cout << "\nРазные алгоритмы могут давать разные результаты в зависимости от специфики голосования";  
  
 return 0;  
}

### Реализация на Go.

package main

import (

"fmt"

)

func findBorda(candidates []string, votes [][]string) string {

scores := make(map[string]int)

for \_, vote := range votes {

for i, candidate := range vote {

scores[candidate] += len(candidates) - 1 - i

}

}

maxScore := -1

var winners []string

for candidate, score := range scores {

if score > maxScore {

maxScore = score

winners = []string{candidate}

} else if score == maxScore {

winners = append(winners, candidate)

}

}

if len(winners) == 1 {

return winners[0]

}

return "Ничья между несколькими кандидатами"

}

func findCondorcet(candidates []string, votes [][]string) string {

pairwise := make(map[string]map[string]int)

for \_, c1 := range candidates {

pairwise[c1] = make(map[string]int)

}

for \_, vote := range votes {

for i := 0; i < len(vote); i++ {

for j := i + 1; j < len(vote); j++ {

pairwise[vote[i]][vote[j]]++

}

}

}

for \_, candidate := range candidates {

isWinner := true

for \_, opponent := range candidates {

if candidate == opponent {

continue

}

if pairwise[candidate][opponent] <= pairwise[opponent][candidate] {

isWinner = false

break

}

}

if isWinner {

return candidate

}

}

return "Победитель Кондорсе не найден"

}

func main() {

candidates := []string{"a", "b", "c"}

votes := [][]string{

{"a", "c", "b"},

{"a", "c", "b"},

{"c", "b", "a"},

}

winBorda := findBorda(candidates, votes)

winCondorcet := findCondorcet(candidates, votes)

fmt.Println("Победитель по Борда:", winBorda)

fmt.Println("Победитель по Кондорсе:", winCondorcet)

if winBorda != winCondorcet {

fmt.Println("Разные алгоритмы могут давать разные результаты в зависимости от специфики голосования")

}

}

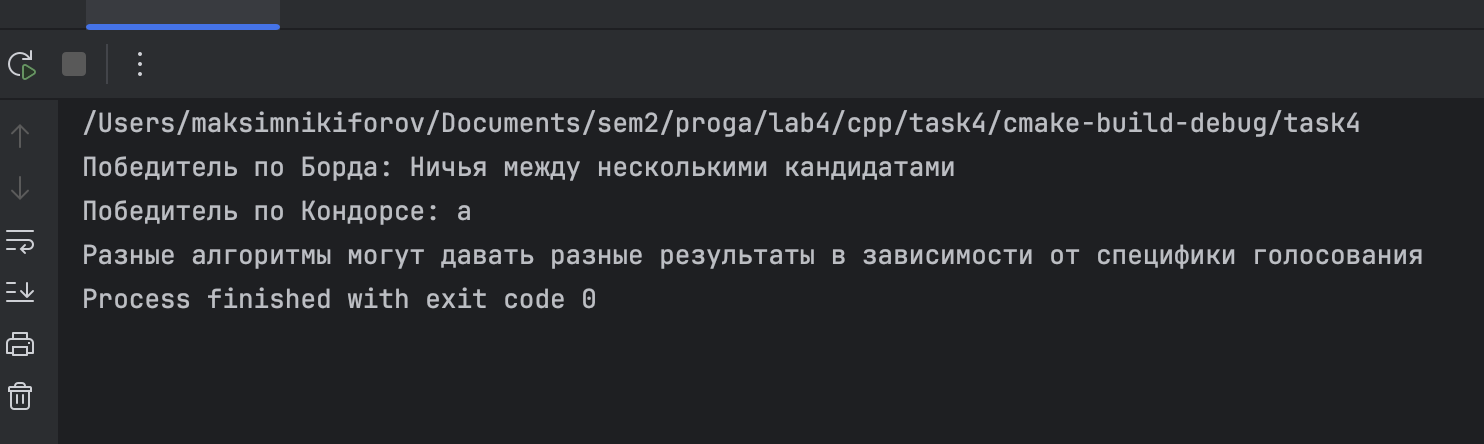
Тестирование работы программы. Рисунок 4.

Рисунок 4 – Результат работы программы

# Вывод

В ходе выполнения данной лабораторной работы по программированию были достигнуты все поставленные цели и задачи. Были изучены методы работы с последовательными одномерными контейнерами, рассмотрены алгоритмы генерации псевдослучайных чисел и основы теории голосований. Также были успешно решены уравнения с применением метода половинного деления, метода Ньютона и метода простых итераций.