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. «АБс-324», «АВТФ» *Ассистент кафедры ЗИ*

*Клаус В. А. Исаев Г. А.*

«\_\_» \_\_\_\_\_\_ 2024 г.«\_\_\_» \_\_\_\_\_\_ 2024 г.

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

(подпись) (подпись)

Новосибирск 2024

**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.

**Задания:**

1. Отделите корни заданного уравнения, согласно варианту из таблицы, и уточните их одним из методов с точностью до ε=10-4 . Решить уравнения методом половинного деления, методом Ньютона и методом простых итераций. **Вариант 10 (2𝑥 + 𝑡𝑔(𝑥) = 0).**
2. 2.1. Напишите программу, в которой определен массив из 𝑛 чисел (𝑛 ≥ 10) и инициализирован случайными числами из диапазона [10, 100].

2.2. Найдите два наименьших элемента массива и вычислите сумму остатков получаемых от деления элементов массива из пункта 1 на минимальный элемент.

2.3. Определите массив. Найдите самую длинную возрастающую последовательность чисел в массиве. Определите и инициализируйте новый массив найденной последовательностью.

2.4. Вывести на экран массив длиной 𝑁 × 8. Первый элемент массива равен 2000. Необходимо выводить элементы строками по 8 элементов.

**Вариант 10. Алгоритм вариант 10 (random\_device).**

1. 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.

**Вариант 10. Алгоритм вариант 10 (random\_device).**

1. Реализовать иру «Предать или сотрудничать» и реализовать 3 алгоритма поведения в игре.
2. Генерация псевдослучайных чисел. **Вариант 5 (RC4).**

**Задание 1**

**C++:**

// 2𝑥 + 𝑡𝑔(𝑥) = 0

#include <iostream>

#include <math.h>

#include <iomanip>

using namespace std;

// Метод Ньютона

double f(double x) { // значение функции

    return 2\*x + tan(x);

}

double fsh(double x) { // значение производной функции

    return 2 + 1/(cos(x)\*cos(x));

}

double newton(double x0, double eps) {

    double x1 = x0 - f(x0) / fsh(x0);

    int iternum = 1;

    while (fabs(x1 - x0) > eps) {

        cout << iternum << setw(15) << x0 << setw(15) << x1 << setw(15) << x1 - x0 << endl;

        x0 = x1;

        x1 = x0 - f(x0) / fsh(x0);

        iternum++;

    }

    return x0;

}

// Метод простых итераций

double phi(double x) { // итерационная функция

    return x - f(x) / fsh(x);

}

double SimpleIterations(double x0, double eps) {

    double x1 = phi(x0);

    int iternum = 1;

    while (fabs(x1 - x0) > eps) {

        cout << iternum << setw(15) << x0 << setw(15) << x1 << setw(15) << x1 - x0 << endl;

        x0 = x1;

        x1 = phi(x0);

        iternum++;

    }

    return x0;

}

// Метод половинного деления

double bisection(double a, double b, double eps) {

    double c = 0.0;

    int iternum = 1;

    while (fabs(b - a) > eps) {

        cout << iternum << setw(15) << a << setw(15) << b << setw(15) << b - a << endl;

        c = (a + b) / 2;

        if (f(c) == 0.0) {

            break;

        } else if (f(c) \* f(a) < 0){

            b = c;

        } else {

            a = c;

        }

        iternum++;

    }

    return c;

}

int main() {

    setlocale(LC\_ALL, "Rus");

    srand(time(0));

    // Метод Ньютона

    cout << "Метод Ньютона:" << endl << "N" << setw(15) << "xn" << setw(15) << "xn+1" << setw(15) << "xn+1 - xn" << endl;

    double x0 = 1.5; // начальное приближение

    double eps = 0.0001; // точность решения

    double x = newton(x0, eps);

    cout << "Корень: " << x << endl << endl;

    // Метод простых итераций

    cout << "Метод простых итераций:" << endl << "N" << setw(15) << "xn" << setw(15) << "xn+1" << setw(15) << "xn+1 - xn" << endl;

    x0 = 1.5; // начальное приближение

    eps = 0.0001; // точность решение

    x = SimpleIterations(x0, eps);

    cout << "Корень: " << x << endl << endl;

    // Метод половинного деления

    cout << "Метод половинного деления:" << endl << "N" << setw(15) << "an" << setw(15) << "bn" << setw(15) << "bn - an" << endl;

    double a = -10.0; // начальный интервал

    double b = 5.0;

    eps = 0.0001; // точность решения

    x = bisection(a, b, eps);

    cout << "Корень: " << x << endl << endl;

    return 0;

}

**Метод Ньютона:**

**N xn xn+1 xn+1 - xn**

**1 1.5 1.41528 -0.0847234**

**2 1.41528 1.20446 -0.210819**

**3 1.20446 0.692358 -0.5121**

**4 0.692358 0.0919857 -0.600372**

**5 0.0919857 0.000173645 -0.0918121**

**6 0.000173645 1.16353e-12 -0.000173645**

**Корень: 1.16353e-12**

**Метод простых итераций:**

**N xn xn+1 xn+1 - xn**

**1 1.5 1.41528 -0.0847234**

**2 1.41528 1.20446 -0.210819**

**3 1.20446 0.692358 -0.5121**

**4 0.692358 0.0919857 -0.600372**

**5 0.0919857 0.000173645 -0.0918121**

**6 0.000173645 1.16353e-12 -0.000173645**

**Корень: 1.16353e-12**

**Метод половинного деления:**

**N an bn bn - an**

**1 -10 5 15**

**2 -2.5 5 7.5**

**3 -2.5 1.25 3.75**

**4 -0.625 1.25 1.875**

**5 -0.625 0.3125 0.9375**

**6 -0.15625 0.3125 0.46875**

**7 -0.15625 0.078125 0.234375**

**8 -0.0390625 0.078125 0.117188**

**9 -0.0390625 0.0195312 0.0585938**

**10 -0.00976562 0.0195312 0.0292969**

**11 -0.00976562 0.00488281 0.0146484**

**12 -0.00244141 0.00488281 0.00732422**

**13 -0.00244141 0.0012207 0.00366211**

**14 -0.000610352 0.0012207 0.00183105**

**15 -0.000610352 0.000305176 0.000915527**

**16 -0.000152588 0.000305176 0.000457764**

**17 -0.000152588 7.62939e-05 0.000228882**

**18 -3.8147e-05 7.62939e-05 0.000114441**

**Корень: 1.90735e-05**

**Python:**

import math

# Метод Ньютона

def f(x):  # значение функции

    return 2\*x + math.tan(x)

def fsh(x):  # значение производной функции

    return 2 + 1/(math.cos(x)\*\*2)

def newton(x0, eps):

    x1 = x0 - f(x0) / fsh(x0)

    iternum = 1

    while abs(x1 - x0) > eps:

        print(f"{iternum:2d} {x0:15.10f} {x1:15.10f} {x1 - x0:15.10f}")

        x0 = x1

        x1 = x0 - f(x0) / fsh(x0)

        iternum += 1

    return x0

# Метод простых итераций

def phi(x):  # итерационная функция

    return x - f(x) / fsh(x)

def simple\_iterations(x0, eps):

    x1 = phi(x0)

    iternum = 1

    while abs(x1 - x0) > eps:

        print(f"{iternum:2d} {x0:15.10f} {x1:15.10f} {x1 - x0:15.10f}")

        x0 = x1

        x1 = phi(x0)

        iternum += 1

    return x0

# Метод половинного деления

def bisection(a, b, eps):

    c = 0.0

    iternum = 1

    while abs(b - a) > eps:

        print(f"{iternum:2d} {a:15.10f} {b:15.10f} {b - a:15.10f}")

        c = (a + b) / 2

        if f(c) == 0.0:

            break

        elif f(c) \* f(a) < 0:

            b = c

        else:

            a = c

        iternum += 1

    return c

# Метод Ньютона

print("Метод Ньютона:\nN\txn\txn+1\txn+1 - xn")

x0 = 1.5  # начальное приближение

eps = 0.0001  # точность решения

x = newton(x0, eps)

print(f"Корень: {x:.10f}\n")

# Метод простых итераций

print("Метод простых итераций:\nN\txn\txn+1\txn+1 - xn")

x0 = 1.5  # начальное приближение

eps = 0.0001  # точность решения

x = simple\_iterations(x0, eps)

print(f"Корень: {x:.10f}\n")

# Метод половинного деления

print("Метод половинного деления:\nN\tan\tbn\tbn - an")

a = -10.0  # начальный интервал

b = 5.0

eps = 0.0001  # точность решения

x = bisection(a, b, eps)

print(f"Корень: {x:.10f}\n")

**Метод Ньютона:**

**N xn xn+1 xn+1 - xn**

**1 1.5 1.41528 -0.0847234**

**2 1.41528 1.20446 -0.210819**

**3 1.20446 0.692358 -0.5121**

**4 0.692358 0.0919857 -0.600372**

**5 0.0919857 0.000173645 -0.0918121**

**6 0.000173645 1.16353e-12 -0.000173645**

**Корень: 1.16353e-12**

**Метод простых итераций:**

**N xn xn+1 xn+1 - xn**

**1 1.5 1.41528 -0.0847234**

**2 1.41528 1.20446 -0.210819**

**3 1.20446 0.692358 -0.5121**

**4 0.692358 0.0919857 -0.600372**

**5 0.0919857 0.000173645 -0.0918121**

**6 0.000173645 1.16353e-12 -0.000173645**

**Корень: 1.16353e-12**

**Метод половинного деления:**

**N an bn bn - an**

**1 -10 5 15**

**2 -2.5 5 7.5**

**3 -2.5 1.25 3.75**

**4 -0.625 1.25 1.875**

**5 -0.625 0.3125 0.9375**

**6 -0.15625 0.3125 0.46875**

**7 -0.15625 0.078125 0.234375**

**8 -0.0390625 0.078125 0.117188**

**9 -0.0390625 0.0195312 0.0585938**

**10 -0.00976562 0.0195312 0.0292969**

**11 -0.00976562 0.00488281 0.0146484**

**12 -0.00244141 0.00488281 0.00732422**

**13 -0.00244141 0.0012207 0.00366211**

**14 -0.000610352 0.0012207 0.00183105**

**15 -0.000610352 0.000305176 0.000915527**

**16 -0.000152588 0.000305176 0.000457764**

**17 -0.000152588 7.62939e-05 0.000228882**

**18 -3.8147e-05 7.62939e-05 0.000114441**

**Корень: 1.90735e-05**

**Задание 2**

**С++:**

// Задание вариант 10 (Вариант алгоритма 10 (random\_device))

#include <iostream>

#include <vector>

#include <random>

using namespace std;

vector<int> Generate\_Vector(int n) { // ф-ия заполнения массива рандомными числами

    random\_device generator;

    vector<int> nums;

    for (int i = 0; i < n; ++i) {

        int randnums = 10 + generator() % 90;

        nums.push\_back(randnums);

    }

    return nums;

}

int find\_mins(vector<int> nums, int& min1, int& min2) { // ф-ия нахождения 2 мин.чисел и суммы отсатков при / эл.массива на мин.число

    for (auto randnums : nums) {

        if (randnums < min1) {

            min2 = min1;

            min1 = randnums;

        } else if (randnums < min2 && randnums != min1) {

            min2 = randnums;

        }

    }

    int sumost = 0;

    for (auto elem : nums) {

        sumost += (elem % min1);

    }

    return sumost;

}

vector<int> FindLongSeq(vector<int> nums) { // ф-ия нахождения самой длинной последовательность массива

    vector<int> LongSeq;

    vector<int> CurrentSeq;

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

        if (CurrentSeq.empty() || nums[i] > CurrentSeq.back()) {

            CurrentSeq.push\_back(nums[i]);

        } else {

            if (CurrentSeq.size() > LongSeq.size()) {

                LongSeq = CurrentSeq;

            }

            CurrentSeq.clear();

            CurrentSeq.push\_back(nums[i]);

        }

    }

    if (CurrentSeq.size() > LongSeq.size()) {

        LongSeq = CurrentSeq;

    }

    return LongSeq;

}

void matrix(int n) { // Ф-ия заполнения массива и его вывод по n кол-ву столбцов

    cout << "Массив: " << endl;

    n \*= 8;

    int count = 2;

    vector<int> nums3(n);

    nums3[0] = 2000;

    cout << nums3[0] << " ";

    for (int i = 1; i < n; ++i) {

        nums3[i] = nums3[i-1] + 10;

        cout << nums3[i] << " ";

        if (count % 8 == 0) {

            cout << endl;

        }

        count++;

    }

}

template <typename T>

ostream& operator<<(ostream& os, const vector<T>& vec) { // перегрузка вывода вектора

    for (const auto& element : vec) {

        os << element << " ";

    }

    return os;

}

int main(){

    srand(time(0));

    setlocale(LC\_ALL, "Rus");

    random\_device generator;

    // Блок 1

    int n = 0;

    while (n > 10 || n <= 0) {

        cout << "1. Введите размерность массива рандомных чисел(не больше 10): ";

        cin >> n;

    }

    vector<int> nums = Generate\_Vector(n);

    cout << "Вектор рандомных чисел: " << nums << endl << endl;

    // Блок 2

    int min1 = 1000, min2 = 1000;

    int sumost = find\_mins(nums, min1, min2);

    cout << "2. Минимальные числа: " << min1 << " " << min2 << endl << "Сумма остатков: " << sumost << endl << endl;

    // Блок 3

    n = 0;

    while (n > 10 || n <= 0) {

        cout << "3. Введите размерность массива чисел(не больше 10): ";

        cin >> n;

    }

    vector<int> nums2(n);

    cout << "Введите массив чисел: ";

    for (int i = 0; i < n; ++i) {

        cin >> nums2[i];

    }

    vector<int> LongSeq = FindLongSeq(nums2); // ф-ия нахождения самой длинной последовательности массива

    cout << "Самая длинная последовательность массива: ";

    for (int i : LongSeq) {

        cout << i << " ";

    }

    cout << endl << endl;;

    // Блок 4

    n = 0;

    while(n <= 0 || n > 10) {

        cout << "4. Введите кол-во столбцов в массиве(Не больше 10): ";

        cin >> n;

    }

    matrix(n);

    return 0;

}

**Input: 1. Введите размерность массива рандомных чисел(не больше 10): 5**

**Рандомные числа: 83 12 60 19 97**

**2. Минимальные числа: 12 19**

**Сумма остатков: 19**

**3. Введите размерность массива чисел(не больше 10): 5**

**Введите массив чисел: 1 2 3 1 2**

**Самая длинная последовательность массива: 1 2 3**

**4. Введите кол-во столбцов в массиве(Не больше 10): 5**

**Массив:**

**2000 2010 2020 2030 2040 2050 2060 2070**

**2080 2090 2100 2110 2120 2130 2140 2150**

**2160 2170 2180 2190 2200 2210 2220 2230**

**2240 2250 2260 2270 2280 2290 2300 2310**

**2320 2330 2340 2350 2360 2370 2380 2390**

**Python:**

import random

def generate\_vector(n):  # функция заполнения массива случайными числами

    nums = [random.randint(10, 99) for \_ in range(n)]

    return nums

def find\_mins(nums):  # функция нахождения 2 минимальных чисел и суммы остатков при делении элементов массива на минимальное число

    min1 = min(nums)

    min2 = min(x for x in nums if x != min1)

    sumost = sum(elem % min1 for elem in nums)

    return min1, min2, sumost

def find\_long\_seq(nums):  # функция нахождения самой длинной последовательности массива

    long\_seq = []

    current\_seq = []

    for num in nums:

        if not current\_seq or num > current\_seq[-1]:

            current\_seq.append(num)

        else:

            if len(current\_seq) > len(long\_seq):

                long\_seq = current\_seq

            current\_seq = [num]

    if len(current\_seq) > len(long\_seq):

        long\_seq = current\_seq

    return long\_seq

def matrix(n):  # функция заполнения массива и его вывод по n количеству столбцов

    print("Массив:")

    n \*= 8

    nums3 = [2000 + i \* 10 for i in range(n)]

    for i, num in enumerate(nums3):

        print(f"{num} ", end='')

        if (i + 1) % 8 == 0:

            print()

    print()

# Блок 1

n = int(input("1. Введите размерность массива случайных чисел (не больше 10): "))

while n > 10 or n <= 0:

    n = int(input("Введите корректное значение: "))

nums = generate\_vector(n)

print("Вектор случайных чисел:", nums, "\n")

# Блок 2

min1, min2, sumost = find\_mins(nums)

print(f"2. Минимальные числа: {min1} {min2}\nСумма остатков: {sumost}\n")

# Блок 3

n = int(input("3. Введите размерность массива чисел (не больше 10): "))

while n > 10 or n <= 0:

    n = int(input("Введите корректное значение: "))

nums2 = [int(input()) for \_ in range(n)]

long\_seq = find\_long\_seq(nums2)

print("Самая длинная последовательность массива:", long\_seq, "\n")

# Блок 4

n = int(input("4. Введите количество столбцов в массиве (Не больше 10): "))

while n <= 0 or n > 10:

    n = int(input("Введите корректное значение: "))

matrix(n)

**Input: 1. Введите размерность массива рандомных чисел(не больше 10): 5**

**Рандомные числа: 13 10 90 78 100**

**2. Минимальные числа: 10 13**

**Сумма остатков: 11**

**3. Введите размерность массива чисел(не больше 10): 5**

**Введите массив чисел:**

**1**

**2**

**3**

**1**

**2**

**Самая длинная последовательность массива: 1 2 3**

**4. Введите кол-во столбцов в массиве(Не больше 10): 5**

**Массив:**

**2000 2010 2020 2030 2040 2050 2060 2070**

**2080 2090 2100 2110 2120 2130 2140 2150**

**2160 2170 2180 2190 2200 2210 2220 2230**

**2240 2250 2260 2270 2280 2290 2300 2310**

**2320 2330 2340 2350 2360 2370 2380 2390**

**Задание 3**

**С++:**

#include <iostream>

#include <random>

#include <vector>

#include <math.h>

using namespace std;

vector<int> GenerateVector(int n) { // ф-ия заполнения массива рандомными числами

    random\_device generator;

    vector<int> nums;

    for (int i = 0; i < n; ++i) {

        int randnums = 10 + generator() % 90;

        nums.push\_back(randnums);

    }

    return nums;

}

float X2(vector<int> arr, int n) {

    const int iter = 25;

    int sum[iter] = { 0 };

    int mat = 0;

    for (int i = 0; i < n; ++i) { // делим интервалы и подсчитываем значения в них

        sum[arr[i] \* iter / 101]++;

        mat += arr[i];

    }

    float x = 0;

    cout << "Кол-во i элементов в 25 интервалах" << endl;

    for (int i : sum) {

        cout << i << " ";

        x += float((i - (n/iter)) \* (i-(n/iter))) / (n / iter); // xi^2

    }

    cout << endl << "Мат. ожидание 50,5, реальность: " << mat/n << endl;

    return x;

}

void Hypothesis(vector<int> nums) { // Проверка гипотезы

    float krit = 44.314;

    cout << "Для массива на " << nums.size() << " элементов: " << endl;

    float x = X2(nums, nums.size());

    cout << "x^2: " << x << endl;

    if (x < krit) {

        cout << "Гипотеза о нормальном распределении принимается." << endl << endl;

    } else {

        cout << "Гипотеза о нормальном распределении отклоняется." << endl << endl;

    }

}

int main(){

    setlocale(LC\_ALL, "Rus");

    srand(time(0));

    // Генерирую числа для каждого массива

    vector<int> arr50 = GenerateVector(50);

    vector<int> arr100 = GenerateVector(100);

    vector<int> arr1000 = GenerateVector(1000);

    // Результаты проверки гипотезы

    Hypothesis(arr50);

    Hypothesis(arr100);

    Hypothesis(arr1000);

    return 0;

}

**Для массива на 50 элементов:**

**Кол-во i элементов в 25 интервалах**

**0 0 2 3 3 4 1 1 1 3 3 2 1 6 2 0 0 2 3 4 0 2 3 4 0**

**Мат. ожидание 50,5, реальность: 52**

**x^2: 31**

**Гипотеза о нормальном распределении принимается.**

**Для массива на 100 элементов:**

**Кол-во i элементов в 25 интервалах**

**0 0 3 2 4 1 1 8 5 4 8 4 6 9 7 2 7 3 2 5 4 2 7 4 2**

**Мат. ожидание 50,5, реальность: 55**

**x^2: 40.5**

**Гипотеза о нормальном распределении принимается.**

**Для массива на 1000 элементов:**

**Кол-во i элементов в 25 интервалах**

**0 0 43 43 52 40 48 38 52 41 41 45 40 45 38 42 48 44 53 41 41 53 41 42 29**

**Мат. ожидание 50,5, реальность: 53**

**x^2: 104.5**

**Гипотеза о нормальном распределении отклоняется.**

**Задание 4**

**С++:**

#include <iostream>

#include <vector>

using namespace std;

bool sotr() { // Всегда сотрудничает

    return true;

}

bool pred() { // Всегда предает

    return false;

}

bool notrepeat(int round\_number, vector<bool> enemy\_choices) { // Сначала сотрудничает, а далее не повторяет выбор соперника за предыдущих ход

    if (round\_number == 0) return true; // первый раунд - сотрудничает

    else if (enemy\_choices[round\_number - 2] == true) return false; // не повторяет предыдущий ход соперника

    else return true;

}

void play(int rounds, string a\_choices, int& score\_a, int& score\_b) {

    vector<bool> self\_choices; // Свои решения

    vector<bool> enemy\_choices; // решения противника

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

        bool a\_choice, b\_choice;

        if (a\_choices == "Сотрудничает") {

            a\_choice = sotr();

        } else if (a\_choices == "Предаёт") {

            a\_choice = pred();

        } else if (a\_choices == "Не повторяет") {

            a\_choice = notrepeat(i, enemy\_choices);

        }

        b\_choice = notrepeat(i, self\_choices);

        if (a\_choice == true && b\_choice == true) { // оба сотрудничают

            self\_choices.push\_back(true);

            enemy\_choices.push\_back(true);

        } else if (a\_choice == false && b\_choice == false) { // оба предают

            self\_choices.push\_back(false);

            enemy\_choices.push\_back(false);

        } else if (a\_choice == true && b\_choice == false) { // a сотрудничает, b предает

            self\_choices.push\_back(true);

            enemy\_choices.push\_back(false);

        } else { // a предает, b сотрудничает

            self\_choices.push\_back(false);

            enemy\_choices.push\_back(true);

        }

    }

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

        if (self\_choices[i] == true && enemy\_choices[i] == true) {

            score\_a += 24;

            score\_b += 24;

        } else if (self\_choices[i] == false && enemy\_choices[i] == false) {

            score\_a += 4;

            score\_b += 4;

        } else if (self\_choices[i] == true && enemy\_choices[i] == false) {

            score\_a += 0;

            score\_b += 20;

        } else {

            score\_a += 20;

            score\_b += 0;

        }

    }

}

int main() {

    setlocale(LC\_ALL, "Rus");

    srand(time(0));

    // Игра 1(Мы всегда сотрудничаем, противник всегда не повторяет)

    int rounds = 100 + rand() % 101;

    int score\_a = 0, score\_b = 0;

    play(rounds, "Сотрудничает", score\_a, score\_b);

    cout << "Игра 1.\nКол-во раундов: " << rounds << "\nАлгоритм A: " << score\_a << "\nАлгоритм B: " << score\_b << endl << endl;

    // Игра 2(Мы всегда предаём, противник всегда не повторяет)

    rounds = 100 + rand() % 101;

    score\_a = 0, score\_b = 0;

    play(rounds, "Предаёт", score\_a, score\_b);

    cout << "Игра 2.\nКол-во раундов: " << rounds << "\nАлгоритм A: " << score\_a << "\nАлгоритм B: " << score\_b << endl << endl;

    // Игра 3(Мы всегда не повторяем, противник всегда не повторяет)

    rounds = 100 + rand() % 101;

    score\_a = 0, score\_b = 0;

    play(rounds, "Не повторяет", score\_a, score\_b);

    cout << "Игра 3.\nКол-во раундов: " << rounds << "\nАлгоритм A: " << score\_a << "\nАлгоритм B: " << score\_b << endl;

    return 0;

}

**Input: Игра 1.**

**Кол-во раундов: 124**

**Алгоритм A: 24**

**Алгоритм B: 2484**

**Игра 2.**

**Кол-во раундов: 152**

**Алгоритм A: 3040**

**Алгоритм B: 0**

**Игра 3.**

**Кол-во раундов: 176**

**Алгоритм A: 2464**

**Алгоритм B: 2464**

**Python:**

import random

def sotr():  # Всегда сотрудничает

    return True

def pred():  # Всегда предает

    return False

def notrepeat(round\_number, enemy\_choices):  # Сначала сотрудничает, затем не повторяет выбор соперника за предыдущий ход

    if round\_number == 0:

        return True  # первый раунд - сотрудничает

    else:

        return not enemy\_choices[round\_number - 1]  # не повторяет предыдущий ход соперника

def play(rounds, a\_choices):

    self\_choices = []  # Свои решения

    enemy\_choices = []  # Решения противника

    score\_a = 0

    score\_b = 0

    for i in range(rounds):

        if a\_choices == "Сотрудничает":

            a\_choice = sotr()

        elif a\_choices == "Предаёт":

            a\_choice = pred()

        elif a\_choices == "Не повторяет":

            a\_choice = notrepeat(i, enemy\_choices)

        b\_choice = notrepeat(i, self\_choices)

        self\_choices.append(a\_choice)

        enemy\_choices.append(b\_choice)

    for i in range(rounds):

        if self\_choices[i] and enemy\_choices[i]:

            score\_a += 24

            score\_b += 24

        elif not self\_choices[i] and not enemy\_choices[i]:

            score\_a += 4

            score\_b += 4

        elif self\_choices[i] and not enemy\_choices[i]:

            score\_a += 0

            score\_b += 20

        else:

            score\_a += 20

            score\_b += 0

    return score\_a, score\_b

# Игра 1 (Мы всегда сотрудничаем, противник всегда не повторяет)

rounds = 100 + random.randint(0, 100)

score\_a, score\_b = play(rounds, "Сотрудничает")

print(f"Игра 1.\nКол-во раундов: {rounds}\nАлгоритм A: {score\_a}\nАлгоритм B: {score\_b}\n")

# Игра 2 (Мы всегда предаём, противник всегда не повторяет)

rounds = 100 + random.randint(0, 100)

score\_a, score\_b = play(rounds, "Предаёт")

print(f"Игра 2.\nКол-во раундов: {rounds}\nАлгоритм A: {score\_a}\nАлгоритм B: {score\_b}\n")

# Игра 3 (Мы всегда не повторяем, противник всегда не повторяет)

rounds = 100 + random.randint(0, 100)

score\_a, score\_b = play(rounds, "Не повторяет")

print(f"Игра 3.\nКол-во раундов: {rounds}\nАлгоритм A: {score\_a}\nАлгоритм B: {score\_b}")

**Input: Игра 1.**

**Кол-во раундов: 168**

**Алгоритм A: 24**

**Алгоритм B: 3364**

**Игра 2.**

**Кол-во раундов: 163**

**Алгоритм A: 3260**

**Алгоритм B: 0**

**Игра 3.**

**Кол-во раундов: 104**

**Алгоритм A: 1456**

**Алгоритм B: 1456**

**Задание 5**

**С++:**

// вариант 5(RC4)

#include <iostream>

#include <vector>

#include <math.h>

using namespace std;

vector<int> RC4(int n, int GeneratingOutput) { // алгоритм псевдослучайной генерации чисел

    // Инициализация таблиц

    vector<int> S; // Таблица S

    for (int i = 0; i < pow(2, n); i++) { // таблица замен S

        S.push\_back(i);

    }

    vector<int> K; // Таблица K

    int key = 1; // Значение ключа

    for (int i = 0; i < pow(2, n); i++) { // Таблица K с ключом от 1 до 6

        if (key != 6) { // Записываем в таблицу K значения от 1 до 6

            K.push\_back(key);

            key++;

        } else {

            K.push\_back(key);

            key = 1;

        }

    }

    int j = 0;

    for (int i = 0; i < pow(2, n); i++) { // перестановка

        j = (j + S[i] + K[i]) % (int(pow(2, n)));

        swap(S[i], S[j]);

    }

    // Алгоритм псевдослучайной генерации

    vector<int> Z; // Рандомные числа

    int i = 0, t = 0, z = 0;

    j = 0;

    while (GeneratingOutput != 0) {

        i = (i + 1) % (int(pow(2, n)));

        j = (j + S[i]) % (int(pow(2, n)));

        swap(S[i], S[j]);

        t = (S[i] + S[j]) % (int(pow(2, n)));

        z = S[t];

        Z.push\_back(z);

        GeneratingOutput--;

    }

    return Z;

}

template <typename T>

ostream& operator<<(ostream& os, const vector<T>& vec) { // перегрузка вывода вектора

    for (const auto& element : vec) {

        os << element << " ";

    }

    return os;

}

int main(){

    setlocale(LC\_ALL, "Rus");

    int n;

    cout << "Введите число n(ограничение): ";

    cin >> n;

    int countRandNums;

    cout << "Введите необходимое кол-во рандомных чисел: ";

    cin >> countRandNums;

    vector<int> randomnums = RC4(n, countRandNums);

    cout << "Рандомные числа: " << randomnums << endl;

    return 0;

}

**Input: Введите число n: 4**

**Введите необходимое кол-во рандомных чисел: 5**

**Рандомные числа:**

**2 4 10 15 3**

**Вывод:**

В ходе выполнения лабораторной работы №4 я познакомился с новыми методами генерации рандомных чисел и самостоятельно реализовал один из них - RC4, который пригодится мне в дальнейшем изучении криптографии.

**Ссылка на репозиторий:** [нажмите](https://github.com/JamesSevil/laba4)