МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ОБРАЗОВАНИЯ

«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

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

Кафедра защиты информации

*Изображение выглядит как текст, Шрифт, снимок экрана, логотип

Автоматически созданное описание*

**ЛАБОРАТОРНАЯ РАБОТА №4**

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

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

|  |  |
| --- | --- |
| Выполнил:  Студент гр. «АБс-324», «АВТФ»  *Петров Максим Игоревич*  «» октября 2025г  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) | Проверил:  *Ассистент кафедры ЗИ*  *Исаев Глеб Андреевич*  «\_\_\_» \_\_\_\_\_\_ 2025г  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) |

Новосибирск 2025

**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.

**Задание 1**

Вариант 8: *𝑥 𝑙𝑛( 𝑥 + 1) = 1*

**Задание 2**

Вариант 8: ranlux48

**Вариант 8**

1. Напишите программу, в которой определен массив из 𝑛 чисел (n𝑛 ≥ 10) и инициализирован целыми случайными числами из диапазона [0, 100].

2. Поменяйте местами минимальный и второй максимальный элементы местами в массиве из пункта 1.

3. Напишите программу, модифицирующую массив, как в примере: [35, 39, 89, 37, 96] → [96, 35, 39, 89, 37].

4. Вывести на экран массив длиной 𝑁 n × 10 таким образом, как указано на рисунках. Первый элемент массива равен 10. Необходимо выводить элементы строками по 10 элементов.

**Задание 3**

Вариант 8: ranlux48

**Задание 5**

5. RC4

**Задание 1**

**На С++:**

**#include <iostream>**

**#include <math.h>**

**#include <vector>**

**#include <iomanip>**

**#include <Windows.h>**

**using namespace std;**

**const double E = 0.0001;**

**//𝑥 𝑙𝑛( 𝑥 + 1) = 1**

**// Функция для расчета значения исходного уравнения**

**double f(double x) {**

**return (2 \* x) + cos(x) - 0;**

**}**

**// Функция для расчета производной исходной функции**

**double f1(double x) {**

**return 2 - sin(x) - 0;// ??**

**}**

**//методом половинного деления**

**int halfDivision(double a, double b, vector<double>& roots) {**

**if (f(a) \* f(b) >= 0.0) {**

**cout << "Неверный интервал" << endl;**

**return 0;**

**}**

**int k = 0;**

**double c = 0;**

**while (abs(b - a) >= E) {**

**cout << setw(7) << k << "|" << setw(8) << fixed << setprecision(4) << a << " | " << setw(8) << b << "|" << 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;**

**}**

**k++;**

**}**

**roots.push\_back(c);**

**cout << "Корень " << c << " с " << k << " итерациями" << endl;**

**return k;**

**}**

**//Метод Ньютона**

**int newtonMethod(double x0, vector<double>& roots) {**

**int k = 0;**

**double x1 = x0 - f(x0) / f1(x0);**

**while (abs(x1 - x0) >= E) {**

**cout << setw(7) << k << "|" << setw(8) << fixed << setprecision(4) << x0 << " | " << setw(8) << x1 << "|" << x1 - x0 << endl;**

**x0 = x1;**

**x1 = x0 - f(x0) / f1(x0);**

**k++;**

**}**

**roots.push\_back(x0);**

**cout << "Корень " << x0 << " с " << k << " итерациями" << endl;**

**return k;**

**}**

**//метода простых итераций**

**int simpleIterations(double x0, vector<double>& roots) {**

**int k = 0;**

**double x1 = 1 / log(x0 + 1);**

**while (abs(x1 - x0) > E) {**

**cout << setw(7) << k << "|" << setw(8) << fixed << setprecision(4) << x0 << " | " << setw(8) << x1 << "|" << x1 - x0 << endl;**

**x0 = x1;**

**x1 = 1 / log(x0 + 1);**

**k++;**

**}**

**roots.push\_back(x0);**

**cout << "Корень " << x0 << " с " << k << " итерациями" << endl;**

**return k;**

**}**

**int main() {**

**SetConsoleCP(1251);**

**SetConsoleOutputCP(1251);**

**cout << "программу уточнения корня методом половинного деления с точностью до E" << endl;**

**vector<double> roots;**

**int s1 = halfDivision(-1, 1, roots);**

**cout << endl << "программу уточнения корня Ньютоном с точностью до E" << endl;**

**int s2 = newtonMethod(10, roots);**

**cout << endl << "метода простых итераций" << endl;**

**int s3 = simpleIterations(2, roots);**

**cout << endl << "все корни уравнения" << endl;**

**for (double i : roots) {**

**cout << i << " ";**

**}**

**cout << endl << "Скорость сходимости: " << endl;**

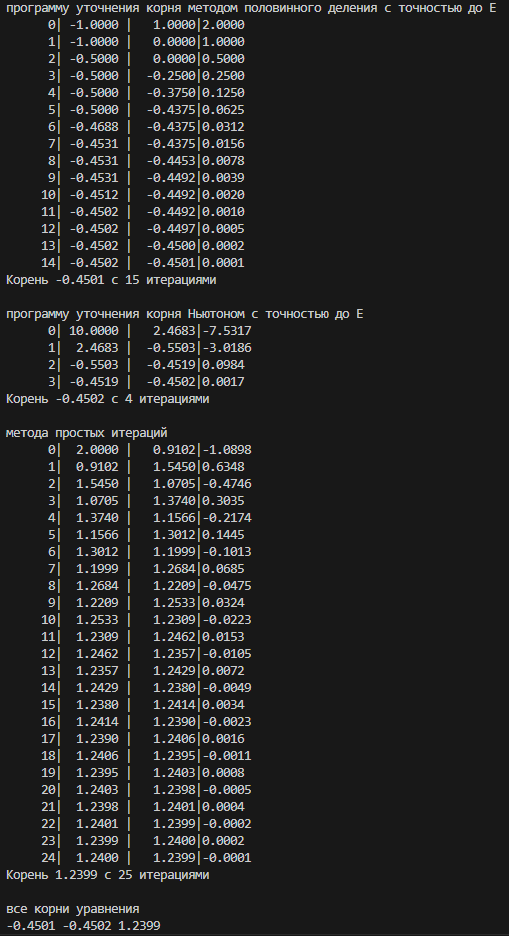
**cout << endl << "У Ньютона итераций " << s2 << endl;**

**cout << endl << "У метода простых итераций " << s3 << endl;**

**cout << endl << "У методом половинного деления " << s1 << endl;**

**}**

**Результат работы программы:**



**Задание 2**

**На С++:**

**#include <iostream>**

**#include <vector>**

**#include <algorithm>**

**#include <ctime>**

**#include <iomanip>**

**#include <Windows.h>**

**using namespace std;**

**vector<int> generateArray(int n, int minChis, int maxChis) { // Функция для генерации случайного массива >= 10**

**vector<int> arr(n);**

**for (int &num : arr) {**

**num = minChis + rand() % (maxChis - minChis + 1);**

**}**

**return arr;**

**}**

**int findMinIndex(vector<int> &arr) { // Функция для нахождения индекса минимального элемента**

**return min\_element(arr.begin(), arr.end()) - arr.begin();**

**}**

**int findMaxSecondIndex(vector<int> &arr) { // Функция для нахождения индекса второго максимального элемента**

**int maxVal = \*std::max\_element(arr.begin(), arr.end());**

**int secondMaxIndex = -1;**

**bool found = false;**

**for (size\_t i = 0; i < arr.size(); ++i) { // Условия для поиска второго максимального элемента**

**if (arr[i] < maxVal) {**

**if (!found || arr[i] > arr[secondMaxIndex]) {**

**secondMaxIndex = i;**

**found = true;**

**}**

**}**

**}**

**return secondMaxIndex;**

**}**

**void modArray(vector<int> &arr) { // Функция для модификации массива**

**if (!arr.empty()) { // Проверка если массив не пустой**

**int first = arr[0];**

**arr.erase(arr.begin());**

**arr.push\_back(first);**

**}**

**}**

**void printArray(vector<int> &arr ,int N) { // Функция для вывода массива**

**for (int i = 0; i < N; ++i) {**

**for (int j = 0; j < 10; ++j) { // Массив должен быть не меньше 10**

**cout << setw(3) << arr[i \* 10 + j] << " ";**

**}**

**cout << endl;**

**}**

**}**

**int main() {**

**SetConsoleCP(1251);**

**SetConsoleOutputCP(1251);**

**srand(time(0)); // Инициализируем генератор случайных чисел**

**int n; // Генерация массива**

**cout << "Введите размер массива (n >= 10): ";**

**cin >> n;**

**if (n < 10) { // Проверка размера массива**

**cerr << "Ошибка! Размер массива не удовлетворяет условию!" << endl;**

**return 1;**

**}**

**vector<int> arr = generateArray(n, 0, 100);**

**cout << "Исходный массив: " << endl;**

**for (int num : arr) cout << num << " "; // Вывод массива**

**cout << endl;**

**int minIndex = findMinIndex(arr);**

**int secMaxIndex = findMaxSecondIndex(arr);**

**if (secMaxIndex != -1) { // Обмен минимального и второго максимального элементов**

**swap(arr[minIndex], arr[secMaxIndex]);**

**}**

**cout << "Массив после изменения: " << endl;**

**for (int num : arr) cout << num << " "; // Вывод измененного массива**

**cout << endl;**

**modArray(arr); // Модификация массива**

**cout << "Массив после модификации: " << endl;**

**for (int num : arr) cout << num << " "; // Вывод модифицированного массива**

**cout << endl;**

**int N; // Количество строк**

**cout << "Введите количество строк для вывода массива формата N x 10: ";**

**cin >> N;**

**if (N <= 0) {**

**cerr << "Ошибка! Количество строк N должно быть положительным числом!";**

**return 1;**

**}**

**vector<int> outputArray(N \* 10);**

**for (int i = 0; i < N \* 10; ++i) {**

**outputArray[i] = 10 + i;**

**}**

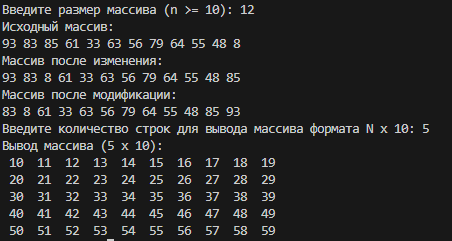
**cout << "Вывод массива (" << N << " x 10):\n";**

**printArray(outputArray, N);**

**return 0;**

**}**

**Результат работы программы:**



**На Python:**

**import random**

**# Функция для генерации случайного массива**

**def generate\_array(n, min\_val, max\_val):**

**return [random.randint(min\_val, max\_val) for \_ in range(n)]**

**# Функция для нахождения индекса минимального элемента**

**def find\_min\_index(arr):**

**return arr.index(min(arr))**

**# Функция для нахождения индекса второго максимального элемента**

**def find\_second\_max\_index(arr):**

**max\_val = max(arr)**

**second\_max = float('-inf')**

**second\_max\_index = -1**

**for i, num in enumerate(arr):**

**if num < max\_val and num > second\_max:**

**second\_max = num**

**second\_max\_index = i**

**return second\_max\_index**

**# Функция для модификации массива**

**def modify\_array(arr):**

**if arr:**

**first = arr.pop(0)**

**arr.append(first)**

**# Функция для вывода массива**

**def print\_array(arr, N):**

**for i in range(N):**

**print(" ".join(f"{arr[i \* 10 + j]:3}" for j in range(10)))**

**# Главная функция**

**def main():**

**n = int(input("Введите размер массива (n >= 10): "))**

**if n < 10:**

**print("Ошибка! Размер массива должен быть не меньше 10!")**

**return**

**arr = generate\_array(n, 0, 100)**

**print("Исходный массив:")**

**print(" ".join(map(str, arr)))**

**# Обмен минимального и второго максимального элементов**

**min\_index = find\_min\_index(arr)**

**second\_max\_index = find\_second\_max\_index(arr)**

**if second\_max\_index != -1:**

**arr[min\_index], arr[second\_max\_index] = arr[second\_max\_index], arr[min\_index]**

**print("\nМассив после изменения:")**

**print(" ".join(map(str, arr)))**

**# Модификация массива**

**modify\_array(arr)**

**print("\nМассив после модификации:")**

**print(" ".join(map(str, arr)))**

**# Формат вывода N x 10**

**N = int(input("\nВведите количество строк для вывода массива формата N x 10: "))**

**if N <= 0:**

**print("Ошибка! Количество строк N должно быть положительным числом!")**

**return**

**output\_array = [10 + i for i in range(N \* 10)]**

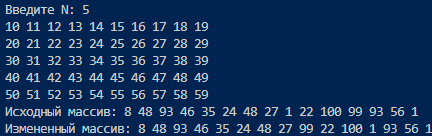
**print("\nВывод массива ({} x 10):".format(N))**

**print\_array(output\_array, N)**

**if \_\_name\_\_ == "\_\_main\_\_":**

**main()**

**Результат работы программы:**



**Задание 3**

**На С++**

**#include <iostream>**

**#include <Windows.h>**

**#include <random>**

**#include <vector>**

**using namespace std;**

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

**void generateRandomNumber(int\* arr, int n) {**

**random\_device rd;**

**ranlux48 engine(rd());**

**uniform\_int\_distribution<int> dist(1, 100); // Задавать диапазон для генерации чисел**

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

**arr[i] = dist(engine);**

**}**

**}**

**int X2(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 колличество 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 << " ";**

**// i кол элем в интервалов**

**// n1 / iter - ожидаемо**

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

**}**

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

**return x;**

**}**

**int main() {**

**system ("chcp 65001");**

**int arr1[n1];**

**generateRandomNumber(arr1, n1);**

**int arr2[n2];**

**generateRandomNumber(arr2, n2);**

**int arr3[n3];**

**generateRandomNumber(arr3, n3);**

**float krit = 44.314;**

**cout << " Для массива на 50 элементов " << endl;**

**//Результат проврки гипотезы**

**int x = X2(arr1, n1);**

**cout << " x^2 : " << x<< endl;**

**if (x < krit) {**

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

**}**

**else {**

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

**}**

**cout << " Для массива на 100 элементов " << endl;**

**x = X2(arr2, n2);**

**cout << " x^2 : " << x << endl;**

**if (x < krit) {**

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

**}**

**else {**

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

**}**

**cout << " Для массива на 1000 элементов " << endl;**

**x = X2(arr3, n3);**

**cout << " x^2 : " << x << endl;**

**if (x < krit) {**

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

**}**

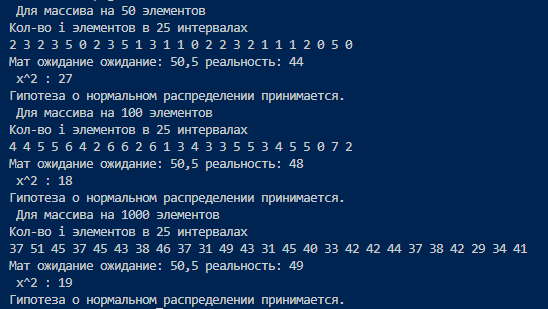
**else {**

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

**}**

**}**

**Результат работы программы:**



**Задание 4**

**На С++**

**#include <iostream>**

**#include <vector>**

**#include <Windows.h>**

**#include <ctime>**

**#include <cstdlib>**

**using namespace std;**

**// Типы выбора**

**const bool COOPERATE = true;**

**const bool BETRAY = false;**

**// Подсчет очков**

**pair<int, int> calculateScores(bool choiceA, bool choiceB) {**

**if (choiceA == COOPERATE && choiceB == COOPERATE) {**

**return {24, 24};**

**} else if (choiceA == COOPERATE && choiceB == BETRAY) {**

**return {0, 20};**

**} else if (choiceA == BETRAY && choiceB == COOPERATE) {**

**return {20, 0};**

**} else { // BETRAY && BETRAY**

**return {4, 4};**

**}**

**}**

**// Алгоритм 1: Всегда сотрудничать**

**bool alwaysCooperate(int round\_number, const vector<bool>& self\_choices, const vector<bool>& enemy\_choices) {**

**return COOPERATE;**

**}**

**// Алгоритм 2: Всегда предавать**

**bool alwaysBetray(int round\_number, const vector<bool>& self\_choices, const vector<bool>& enemy\_choices) {**

**return BETRAY;**

**}**

**// Алгоритм 3: Повторяет выбор противника из прошлого раунда)**

**bool titForTat(int round\_number, const vector<bool>& self\_choices, const vector<bool>& enemy\_choices) {**

**if (round\_number == 0) {**

**return COOPERATE; // Первый ход всегда сотрудничество**

**}**

**return enemy\_choices.back();**

**}**

**// Запуск игры между двумя алгоритмами**

**void playGame(bool (\*algorithmA)(int, const vector<bool>&, const vector<bool>&),**

**bool (\*algorithmB)(int, const vector<bool>&, const vector<bool>&)) {**

**int rounds = 100 + rand() % 101; // Количество раундов от 100 до 200**

**vector<bool> choicesA, choicesB;**

**int scoreA = 0, scoreB = 0;**

**cout << "Игра началась! Количество раундов: " << rounds << endl;**

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

**// Получение выбора обоих алгоритмов**

**bool choiceA = algorithmA(i, choicesA, choicesB);**

**bool choiceB = algorithmB(i, choicesB, choicesA);**

**// Сохранение выборов**

**choicesA.push\_back(choiceA);**

**choicesB.push\_back(choiceB);**

**// Подсчет очков**

**auto scores = calculateScores(choiceA, choiceB);**

**scoreA += scores.first;**

**scoreB += scores.second;**

**cout << "Раунд " << i + 1 << ": A выбрал " << (choiceA ? "Сотрудничество" : "Предательство")**

**<< ", B выбрал " << (choiceB ? "Сотрудничество" : "Предательство") << endl;**

**}**

**cout << "Игра окончена!" << endl;**

**cout << "Итоговый счет: A = " << scoreA << ", B = " << scoreB << endl;**

**}**

**// Тестирование алгоритмов**

**int main() {**

**SetConsoleCP(1251);**

**SetConsoleOutputCP(1251);**

**srand(time(0)); // Инициализация генератора случайных чисел**

**// Запуск игр**

**cout << "Игра 1: Всегда сотрудничает vs Всегда предает" << endl;**

**playGame(alwaysCooperate, alwaysBetray);**

**cout << "\nИгра 2: Всегда предает vs Повторяет выбор противника" << endl;**

**playGame(alwaysBetray, titForTat);**

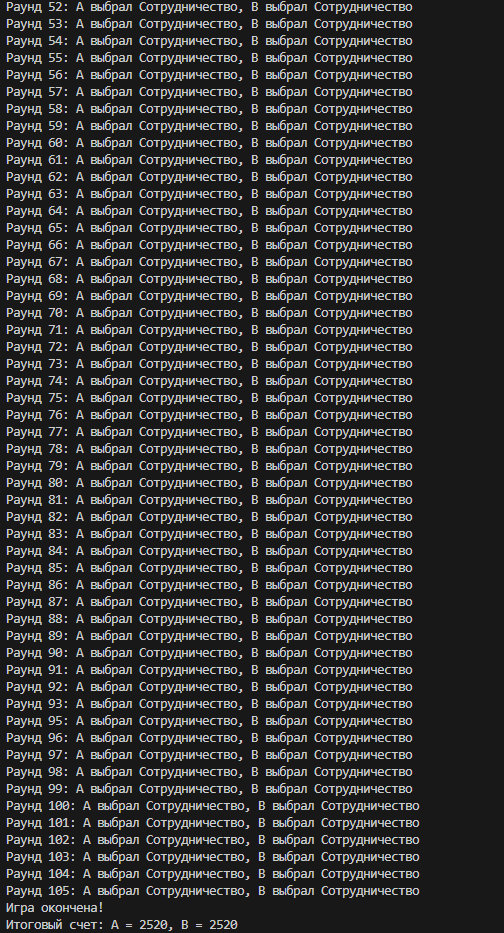
**cout << "\nИгра 3: Повторяет выбор противника vs Всегда сотрудничает" << endl;**

**playGame(titForTat, alwaysCooperate);**

**return 0;**

**}**

**Результат работы программы:**

****

**На Python**

**import random**

**# Типы выбора**

**COOPERATE = True**

**BETRAY = False**

**# Подсчет очков**

**def calculate\_scores(choiceA, choiceB):**

**if choiceA == COOPERATE and choiceB == COOPERATE:**

**return 24, 24**

**elif choiceA == COOPERATE and choiceB == BETRAY:**

**return 0, 20**

**elif choiceA == BETRAY and choiceB == COOPERATE:**

**return 20, 0**

**else: # BETRAY and BETRAY**

**return 4, 4**

**# Алгоритм 1: Всегда сотрудничать**

**def always\_cooperate(round\_number, self\_choices, enemy\_choices):**

**return COOPERATE**

**# Алгоритм 2: Всегда предавать**

**def always\_betray(round\_number, self\_choices, enemy\_choices):**

**return BETRAY**

**# Алгоритм 3: Повторяет выбор противника из прошлого раунда**

**def tit\_for\_tat(round\_number, self\_choices, enemy\_choices):**

**if round\_number == 0:**

**return COOPERATE # Первый ход всегда сотрудничество**

**return enemy\_choices[-1]**

**# Запуск игры между двумя алгоритмами**

**def play\_game(algorithmA, algorithmB):**

**rounds = random.randint(100, 200) # Количество раундов от 100 до 200**

**choicesA, choicesB = [], []**

**scoreA, scoreB = 0, 0**

**print(f"Игра началась! Количество раундов: {rounds}")**

**for i in range(rounds):**

**# Получение выбора обоих алгоритмов**

**choiceA = algorithmA(i, choicesA, choicesB)**

**choiceB = algorithmB(i, choicesB, choicesA)**

**# Сохранение выборов**

**choicesA.append(choiceA)**

**choicesB.append(choiceB)**

**# Подсчет очков**

**scores = calculate\_scores(choiceA, choiceB)**

**scoreA += scores[0]**

**scoreB += scores[1]**

**print(f"Раунд {i + 1}: A выбрал {'Сотрудничество' if choiceA else 'Предательство'}, "**

**f"B выбрал {'Сотрудничество' if choiceB else 'Предательство'}")**

**print("Игра окончена!")**

**print(f"Итоговый счет: A = {scoreA}, B = {scoreB}")**

**# Тестирование алгоритмов**

**if \_\_name\_\_ == "\_\_main\_\_":**

**random.seed() # Инициализация генератора случайных чисел**

**# Запуск игр**

**print("Игра 1: Всегда сотрудничает vs Всегда предает")**

**play\_game(always\_cooperate, always\_betray)**

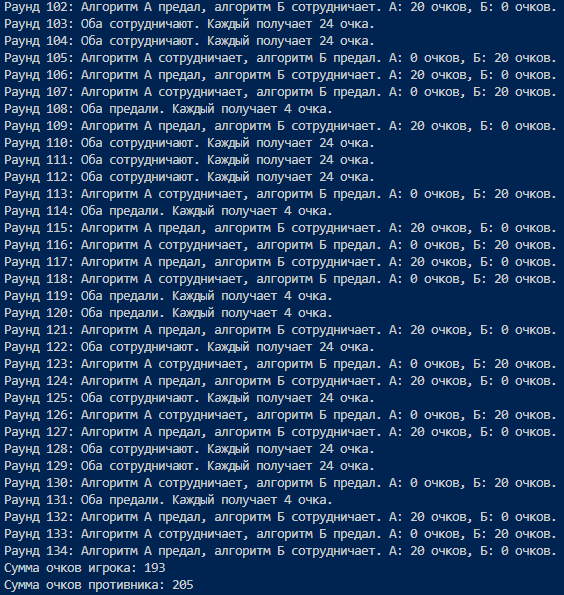
**print("\nИгра 2: Всегда предает vs Повторяет выбор противника")**

**play\_game(always\_betray, tit\_for\_tat)**

**print("\nИгра 3: Повторяет выбор противника vs Всегда сотрудничает")**

**play\_game(tit\_for\_tat, always\_cooperate)**

**Результат работы программы:**



**Задание 5**

**#include <iostream>**

**#include <Windows.h>**

**#include <random>**

**#include <vector>**

**using namespace std;**

**//Инициализация таблиц**

**void initialize(int\* S, int\* K, int s) {**

**for (int i = 0; i < s; i++) {**

**S[i] = i;**

**}**

**int j = 0;**

**for (int i = 0; i < s; i++) {**

**j = (j + S[i] + K[i]) % s;**

**swap(S[i], S[j]);**

**}**

**}**

**// Алгоритм псевдослучайной генерации**

**void generationAlgorithm(int\* S, vector<int>& output, int n) {**

**int i = 0;**

**int j = 0;**

**for (int k = 0; k < n; k++) {**

**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);**

**}**

**}**

**int main() {**

**system ("chcp 65001");**

**//2^i**

**const int n=32;**

**int S[n];**

**// Значения ключа K**

**int K[n];**

**// Заполнение ключа K**

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

**K[i] = i;**

**}**

**//Инициализация таблиц**

**initialize(S, K, n);**

**// Алгоритм псевдослучайной генерации**

**vector<int> output;**

**generationAlgorithm(S, output, n);**

**// Вывод сгенерированных чисел**

**cout << "Сгенерированные числа: "<<endl;**

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

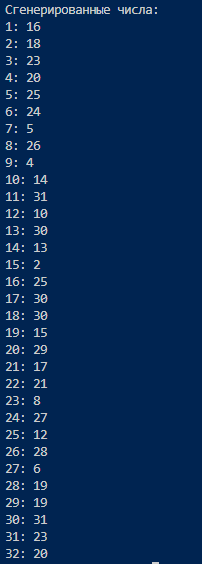
**cout << i+1 << ": " << output[i] << endl;**

**}**

**return 0;**

**}**

**Результат работы программы:**



**Вывод:**

В результате проделанной работы был изучен алгоритмы формирования и обработки одномерных массивов и последовательных контейнеров, программирование и отладка программ формирования и обработки массивов. А также написаны программы решения задач в соответствии с индивидуальным вариантом.

**Ссылка на гитхаб: https://github.com/Mixassss/laba4.git**