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 ЗИ*

*Петров М.И. Исаев Г.А.*

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

**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 <cmath>**

**#include <iomanip>**

**using namespace std;**

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

**double func(double x) {**

**return (x \* log(x + 1) - 1);**

**}**

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

**double derivative(double x) {**

**return x / (x + 1) + log(x + 1);**

**}**

**// Метод Ньютона для решения уравнения**

**void newtonMethod(double x0, double epsilon) {**

**int N = 0;**

**double xn = x0, xn1, diff;**

**cout << fixed << setprecision(6);**

**cout << "| N | xn | xn+1 | xn+1 – xn |\n";**

**do {**

**xn1 = xn - func(xn) / derivative(xn);**

**diff = abs(xn1 - xn);**

**cout << "| " << N << " | " << xn << " | " << xn1 << " | " << diff << " |\n";**

**xn = xn1;**

**N++;**

**} while (diff >= epsilon);**

**cout << "Корень уравнения методом Ньютона: " << xn << endl;**

**}**

**// Метод простых итераций для решения уравнения**

**void simpleIterationMethod(double x0, double epsilon) {**

**int N = 0;**

**double xn = x0, xn1, diff;**

**// Вывод таблицы с результатами иллюстрирующей шаги метода простых итераций**

**cout << fixed << setprecision(6);**

**cout << "| N | xn | xn+1 | xn+1 – xn |\n";**

**do {**

**// Применение формулы для метода простых итераций**

**xn1 = log(1/xn + 1);**

**diff = abs(xn1 - xn);**

**cout << "| " << N << " | " << xn << " | " << xn1 << " | " << diff << " |\n";**

**xn = xn1;**

**N++;**

**} while (diff >= epsilon);**

**// Вывод найденного корня уравнения методом простых итераций**

**cout << "Корень уравнения методом простых итераций: " << xn << endl;**

**}**

**//Метод половинного деления**

**void bisectionMethod(double a, double b, double epsilon) {**

**if (func(a) \* func(b) >= 0) {**

**cout << "Невозможно найти корень на указанном интервале. Пожалуйста, выберите другие a и b." << endl;**

**return;**

**}**

**int N = 1;**

**double an, bn;**

**cout << setw(5) << "N" << setw(10) << "an" << setw(10) << "bn" << setw(15) << "bn - an" << endl;**

**while ((b - a) >= epsilon) {**

**double c = (a + b) / 2;**

**if (func(c) == 0.0) {**

**break;**

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

**b = c;**

**} else {**

**a = c;**

**}**

**an = a;**

**bn = b;**

**cout << setw(5) << N << setw(10) << an << setw(10) << bn << setw(15) << bn - an << endl;**

**N++;**

**}**

**cout << "Корень уравнения на заданной точности равен: " << (a + b) / 2 << endl;**

**}**

**int main() {**

**double a = 0.1; // начальное значение интервала [a, b]**

**double b = 2.0; // начальное значение интервала [a, b]**

**double x0 = 0.1; // начальное приближение**

**double epsilon = 0.0001; // требуемая точность**

**bisectionMethod (a, b, epsilon);**

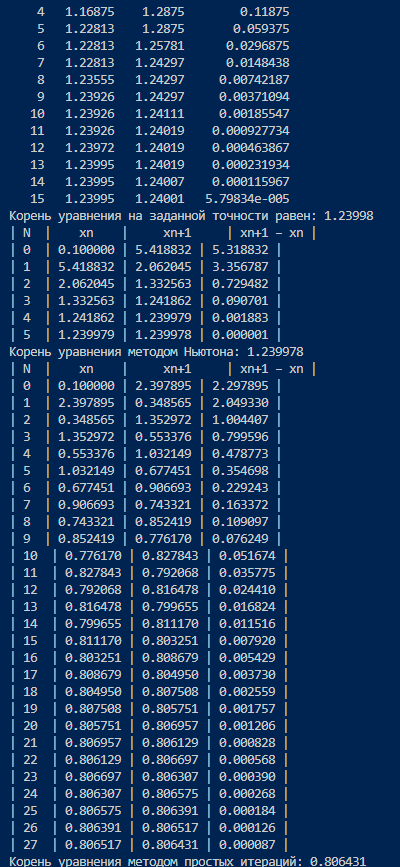
**newtonMethod (x0, epsilon); // Вызов функции для метода Ньютона**

**simpleIterationMethod (x0, epsilon); // Вызов функции для метода простых итераций**

**return 0;**

**}**

**Результат работы программы:**



**Задание 2**

**На С++:**

**#include <iostream>**

**#include <vector>**

**#include <algorithm>**

**#include <random>**

**#include <ctime>**

**using namespace std;**

**int main() {**

**int N;**

**cout << "Введите N: ";**

**cin >> N;**

**vector<int> array(N \* 10); // Создаем массив длиной N \* 10**

**// Заполняем массив последовательными числами, начиная с 10**

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

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

**}**

**// Вывод массива по 10 элементов в строке**

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

**cout << array[i] << " ";**

**if ((i + 1) % 10 == 0) {**

**cout << endl;**

**}**

**}**

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

**const int n = 15; // Размер массива (не менее 10)**

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

**// Генерация случайных чисел и заполнение массива**

**random\_device rd;**

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

**uniform\_int\_distribution<int> dist(1, 100);**

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

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

**}**

**// Вывод исходного массива**

**cout << "Исходный массив: ";**

**for (int num : numbers) {**

**cout << num << " " << endl;**

**}**

**// Поиск минимального и второго максимального элемента**

**int min\_index = min\_element(numbers.begin(), numbers.end()) - numbers.begin();**

**int max\_index = max\_element(numbers.begin(), numbers.end()) - numbers.begin();**

**// Находим второй максимальный элемент, исключая первый**

**int second\_max\_index = min\_index; // Инициализируем индексом минимального (на случай, если все элементы равны)**

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

**if (numbers[i] > numbers[second\_max\_index] && i != max\_index) {**

**second\_max\_index = i;**

**}**

**}**

**// Меняем местами минимальный и второй максимальный элементы**

**swap(numbers[min\_index], numbers[second\_max\_index]);**

**// Вывод измененного массива**

**cout << "Измененный массив: ";**

**for (int num : numbers) {**

**cout << num << " ";**

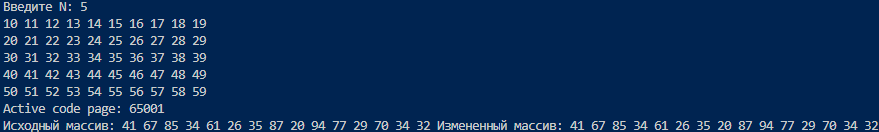
**}**

**cout << endl;**

**return 0;**

**}**

**Результат работы программы:**



**На Python:**

**import random**

**N = int(input("Введите N: "))**

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

**for i in range(N \* 10):**

**print(array[i], end=" ")**

**if (i + 1) % 10 == 0:**

**print()**

**n = 15**

**numbers = [random.randint(1, 100) for \_ in range(n)]**

**print("Исходный массив:", end=" ")**

**for num in numbers:**

**print(num, end=" ")**

**min\_index = numbers.index(min(numbers))**

**max\_index = numbers.index(max(numbers))**

**second\_max\_index = min\_index**

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

**if numbers[i] > numbers[second\_max\_index] and i != max\_index:**

**second\_max\_index = i**

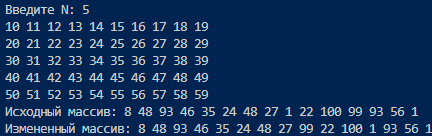
**numbers[min\_index], numbers[second\_max\_index] = numbers[second\_max\_index], numbers[min\_index]**

**print("\nИзмененный массив:", end=" ")**

**for num in numbers:**

**print(num, end=" ")**

**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;**

**}**

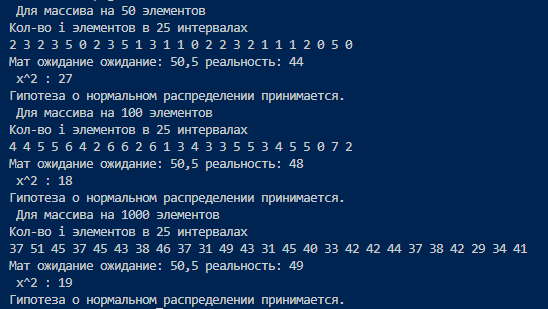
**else {**

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

**}**

**}**

**Результат работы программы:**



**Задание 4**

**На С++**

**#include <iostream>**

**#include <vector>**

**#include <cstdlib>**

**#include <ctime>**

**using namespace std;**

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

**// код для алгоритма**

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

**return true; // В начале игры всегда сотрудничаем**

**} else {**

**// Расчет стратегии алгоритма на основе предыдущих раундов**

**int total\_rounds = self\_choices.size();**

**int self\_cooperate\_count = 0, enemy\_cooperate\_count = 0;**

**for (int i = 0; i < total\_rounds; ++i) {**

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

**self\_cooperate\_count++;**

**}**

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

**enemy\_cooperate\_count++;**

**}**

**}**

**// Пример простой стратегии: если противник предал на предыдущем раунде, тогда предаем, иначе сотрудничаем**

**if (enemy\_choices[round\_number - 1] == false) {**

**return false;**

**} else {**

**return true;**

**}**

**}**

**}**

**int main() {**

**srand(time(0));**

**int num\_rounds = rand() % 101 + 100; // Генерация случайного количества раундов от 100 до 200**

**vector<bool> self\_choices;**

**vector<bool> enemy\_choices;**

**for (int i = 0; i < num\_rounds; ++i) {**

**self\_choices.push\_back(func(i, self\_choices, enemy\_choices));**

**enemy\_choices.push\_back(rand() % 2); // Генерация случайного выбора противника: 0 - предательство, 1 - сотрудничество**

**}**

**// Вывод результатов**

**for (int i = 0; i < num\_rounds; ++i) {**

**cout << "Раунд " << i << ": ";**

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

**cout << "Оба сотрудничают. Каждый получает 24 очка.\n";**

**} else if (!self\_choices[i] && !enemy\_choices[i]) {**

**cout << "Оба предали. Каждый получает 4 очка.\n";**

**} else if (self\_choices[i] && !enemy\_choices[i]) {**

**cout << "Алгоритм А сотрудничает, алгоритм Б предал. А: 0 очков, Б: 20 очков.\n";**

**} else {**

**cout << "Алгоритм А предал, алгоритм Б сотрудничает. А: 20 очков, Б: 0 очков.\n";**

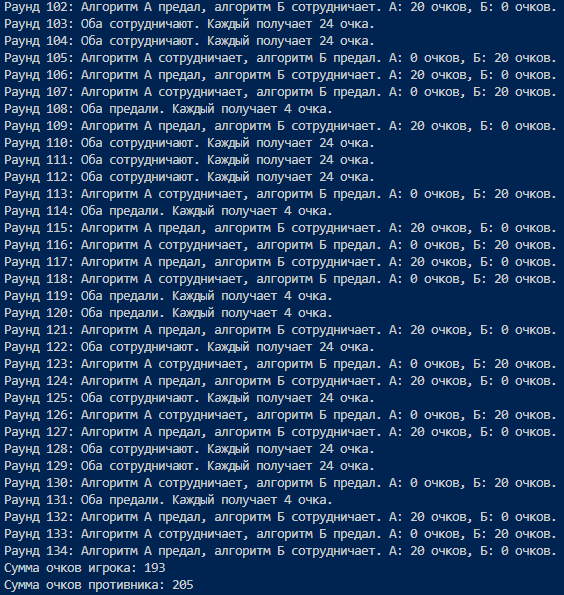
**}**

**}**

**return 0;**

**}**

**Результат работы программы:**



**На Python**

**import random**

**class Player:**

**def \_\_init\_\_(self):**

**self.score = 0**

**self.moves = []**

**def make\_move(self, previous\_moves):**

**pass**

**class Betrayer(Player):**

**def make\_move(self, previous\_moves):**

**return 'betray'**

**class Cooperator(Player):**

**def make\_move(self, previous\_moves):**

**return 'cooperate'**

**class RandomPlayer(Player):**

**def make\_move(self, previous\_moves):**

**return random.choice(['betray', 'cooperate'])**

**class Game:**

**def \_\_init\_\_(self, player1, player2):**

**self.player1 = player1**

**self.player2 = player2**

**self.rounds = random.randint(100, 200)**

**def play\_round(self, moves):**

**if moves[0] == 'betray' and moves[1] == 'betray':**

**return 4, 4**

**elif moves[0] == 'cooperate' and moves[1] == 'betray':**

**return 0, 20**

**elif moves[0] == 'cooperate' and moves[1] == 'cooperate':**

**return 24, 24**

**else:**

**return 20, 0**

**def play\_game(self):**

**for \_ in range(self.rounds):**

**moves = [self.player1.make\_move(self.player2.moves), self.player2.make\_move(self.player1.moves)]**

**self.player1.moves.append(moves[1])**

**self.player2.moves.append(moves[0])**

**p1\_score, p2\_score = self.play\_round(moves)**

**self.player1.score += p1\_score**

**self.player2.score += p2\_score**

**return self.player1.score, self.player2.score**

**# Создаем игроков**

**player1 = Betrayer()**

**player2 = Cooperator()**

**player3 = RandomPlayer()**

**# Создаем игру**

**game = Game(player1, player2)**

**# Играем**

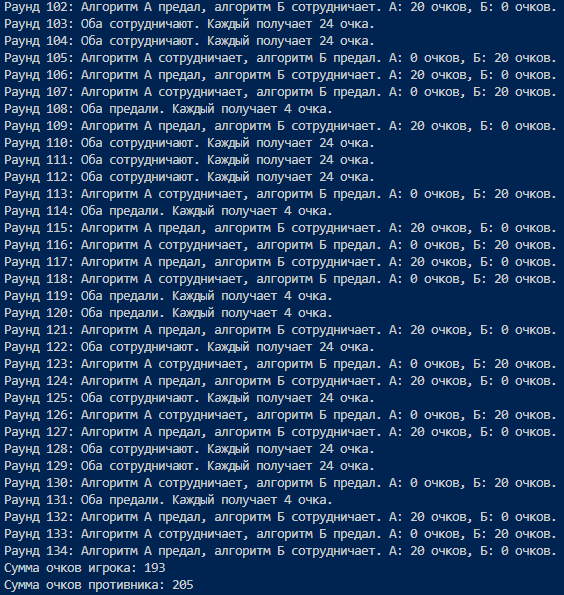
**score1, score2 = game.play\_game()**

**print("Результаты игры:")**

**print(f"Игрок 1: {score1} очков")**

**print(f"Игрок 2: {score2} очков")**

**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++) {**

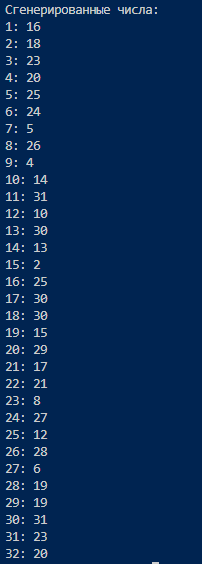
**cout << i+1 << ": " << output[i] << endl;**

**}**

**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**