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*

*Аврамчук Илья Вячеславович Исаев Г. А.*

«21» мая 2024г«31» мая 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** «Методы решения нелинейный уравнений»   
**Задание №2** «Вычисления в одномерных массивах»   
**Задание №3** «Работа с генераторами рандомных чисел»

**Задание №4** «Дилемма заключенного»   
**Задание №5** «Генерация псевдослучайных чисел»

**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 «Методы решения нелинейный уравнений»**

Отделите корни заданного уравнения, согласно варианту из табл.1, и уточните их одним из методов с точностью до ε=10-4 . Решить уравнения методом половинного деления, методом Ньютона и методом простых итераций.

**Вариант 1**

**Уравнение: cos(x + 0.5) - x = 2  
Язык программирования: С++  
Код программы:**  
#include <iostream> // Подключение стандартной библиотеки для ввода-вывода

#include <cmath> // Подключение библиотеки математических функций

#include <iomanip> // Подключение библиотеки для управления выводом формата чисел

using namespace std; // Использование пространства имен std (стандартной библиотеки)

// Функция, для которой ищем корни: cos(x + 0.5) - x - 2

double f(double x) {

return cos(x + 0.5) - x - 2; // Определение функции и её возвращаемое значение

}

// Производная функции: -sin(x + 0.5) - 1

double df(double x) {

return -sin(x + 0.5) - 1; // Вычисление производной функции

}

// Метод половинного деления для нахождения корня на отрезке [a, b] с точностью ε

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

// Проверка наличия корней на концах отрезка

if (f(a) \* f(b) > 0) {

cout << "На заданном интервале нет корней или их количество нечетное." << endl;

return; // Возвращаемся, если корни отсутствуют

}

// Переменные для хранения текущего приближения корня и числа итераций

double c, fc;

int iter = 0;

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

cout << setw(5) << "N" << setw(15) << "a" << setw(15) << "b" << setw(15) << "b - a" << endl;

// Начало цикла метода половинного деления

do {

iter++; // Увеличение числа итераций

c = (a + b) / 2; // Нахождение середины интервала

fc = f(c); // Вычисление значения функции в точке c

// Вывод текущей итерации: номер, значения a, b и длина интервала b - a

cout << setw(5) << iter << setw(15) << a << setw(15) << b << setw(15) << b - a << endl;

// Проверка условия сходимости: если знаки функции на концах интервала разные, сдвигаем границы

if (f(a) \* fc < 0)

b = c; // Перемещаем правую границу

else

a = c; // Перемещаем левую границу

} while (fabs(b - a) > epsilon); // Повторяем, пока не достигнем заданной точности

// Вывод приближенного значения корня

cout << "Приближенный корень методом половинного деления: " << setprecision(10) << c << endl;

}

// Метод Ньютона для нахождения корня с начальным приближением x0 и точностью ε

void newtonMethod(double x0, double epsilon) {

double x = x0; // Начальное приближение

int iter = 0; // Счетчик итераций

cout << "Метод Ньютона:" << endl;

cout << setw(5) << "N" << setw(15) << "x\_n" << setw(15) << "x\_n+1" << setw(15) << "x\_n+1 - x\_n" << endl;

do {

iter++; // Увеличение числа итераций

double fx = f(x); // Вычисление значения функции в точке x

double dfx = df(x); // Вычисление значения производной функции в точке x

double x1 = x - fx / dfx; // Вычисление следующего приближения методом Ньютона

// Вывод текущей итерации: номер, текущее и следующее приближение, их разность

cout << setw(5) << iter << setw(15) << x << setw(15) << x1 << setw(15) << x1 - x << endl;

x = x1; // Переходим к следующему приближению

} while (fabs(f(x)) > epsilon); // Повторяем, пока не достигнем заданной точности

// Вывод приближенного значения корня

cout << "Приближенный корень методом Ньютона: " << setprecision(10) << x << endl;

}

// Метод простых итераций для нахождения корня с начальным приближением x0 и точностью ε

void fixedPointIteration(double x0, double epsilon) {

double x = x0; // Начальное приближение

int iter = 0; // Счетчик итераций

cout << "Метод простых итераций:" << endl;

cout << setw(5) << "N" << setw(15) << "x\_n" << setw(15) << "x\_n+1" << setw(15) << "x\_n+1 - x\_n" << endl;

do {

iter++; // Увеличение числа итераций

double x1 = cos(x + 0.5) - 2; // Функция g(x) = cos(x + 0.5) - 2 для метода простых итераций

// Вывод текущей итерации: номер, текущее и следующее приближение, их разность

cout << setw(5) << iter << setw(15) << x << setw(15) << x1 << setw(15) << x1 - x << endl;

x = x1; // Переходим к следующему приближению

} while (fabs(f(x)) > epsilon); // Повторяем, пока не достигнем заданной точности

// Вывод приближенного значения корня

cout << "Приближенный корень методом простых итераций: " << setprecision(10) << x << endl;

}

// Основная функция программы

int main() {

double a = -5.0; // Начальная левая граница интервала для метода половинного деления

double b = 5.0; // Начальная правая граница интервала для метода половинного деления

double x0 = 0.0; // Начальное приближение для метода Ньютона и метода простых итераций

double epsilon = 1e-4; // Заданная точность

// Вызываем метод половинного деления

bisectionMethod(a, b, epsilon);

cout << endl;

// Вызываем метод Ньютона

newtonMethod(x0, epsilon);

cout << endl;

// Вызываем метод простых итераций

fixedPointIteration(x0, epsilon);

return 0; // Возвращаем успешный код завершения программы

}

**Язык программирования: Java  
Код программы:**import java.util.Formatter;

public class Main {

// Function: cos(x + 0.5) - x - 2

public static double f(double x) {

return Math.cos(x + 0.5) - x - 2;

}

// Derivative of the function: -sin(x + 0.5) - 1

public static double df(double x) {

return -Math.sin(x + 0.5) - 1;

}

// Bisection method for finding the root on the interval [a, b] with precision ε

public static void bisectionMethod(double a, double b, double epsilon) {

if (f(a) \* f(b) > 0) {

System.out.println("No roots or an even number of roots exist on the given interval.");

return;

}

double c;

int iter = 0;

System.out.println("Bisection Method:");

System.out.printf("%5s %15s %15s %15s\n", "N", "a", "b", "b - a");

do {

iter++;

c = (a + b) / 2;

double fc = f(c);

System.out.printf("%5d %15.10f %15.10f %15.10f\n", iter, a, b, b - a);

if (f(a) \* fc < 0) {

b = c;

} else {

a = c;

}

} while (Math.abs(b - a) > epsilon);

System.out.printf("Approximate root by bisection method: %.10f\n", c);

}

// Newton's method for finding the root with initial approximation x0 and precision ε

public static void newtonMethod(double x0, double epsilon) {

double x = x0;

int iter = 0;

System.out.println("Newton's Method:");

System.out.printf("%5s %15s %15s %15s\n", "N", "x\_n", "x\_n+1", "x\_n+1 - x\_n");

do {

iter++;

double fx = f(x);

double dfx = df(x);

double x1 = x - fx / dfx;

System.out.printf("%5d %15.10f %15.10f %15.10f\n", iter, x, x1, x1 - x);

x = x1;

} while (Math.abs(f(x)) > epsilon);

System.out.printf("Approximate root by Newton's method: %.10f\n", x);

}

// Fixed-point iteration method for finding the root with initial approximation x0 and precision ε

public static void fixedPointIteration(double x0, double epsilon) {

double x = x0;

int iter = 0;

System.out.println("Fixed-Point Iteration Method:");

System.out.printf("%5s %15s %15s %15s\n", "N", "x\_n", "x\_n+1", "x\_n+1 - x\_n");

do {

iter++;

double x1 = Math.cos(x + 0.5) - 2;

System.out.printf("%5d %15.10f %15.10f %15.10f\n", iter, x, x1, x1 - x);

x = x1;

} while (Math.abs(f(x)) > epsilon);

System.out.printf("Approximate root by fixed-point iteration method: %.10f\n", x);

}

public static void main(String[] args) {

double a = -5.0;

double b = 5.0;

double x0 = 0.0;

double epsilon = 1e-4;

bisectionMethod(a, b, epsilon);

System.out.println();

newtonMethod(x0, epsilon);

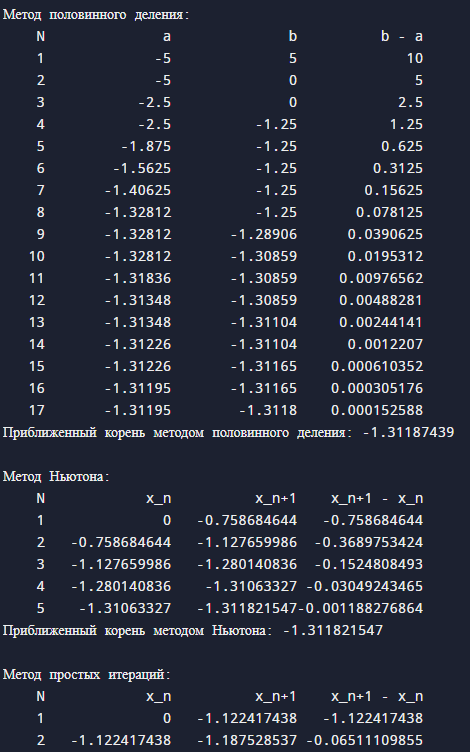
System.out.println();

fixedPointIteration(x0, epsilon);

}

}

**Результат работы программы:**



**Задание №2 «Вычисления в одномерных массивах»**   
**Вариант 1**

Алгоритм - minstd\_rand

1. Напишите программу, в которой определен массив из n целых чисел (n ≥ 10). Инициализируйте массив случайными числами из диапазона [150, 300].
2. Найдите и выведите самую длинную последовательность чисел массива из пункта 1, упорядоченную по убыванию.
3. Определите и инициализируйте новый массив, состоящий из чисел, меньших среднеарифметического значения массива из пункта 1.
4. Напишите программу, модифицирующую массив случайных символов размера n (n ≥ 5), как в примере: [0,I,D,Q,I,A] → [D,Q,I,A,0,I].

5. Определите и инициализируйте массив, состоящий из случайных целых чисел, входящих в диапазон [100, 900]. Отсортируйте новый массив по убыванию. Определите, сколько раз каждая цифра встречается в элементах массива.

**Язык программирования: С++  
Код программы:**

#include <iostream>

#include <vector>

#include <random>

#include <numeric>

#include <algorithm>

#include <array>

using namespace std;

// Функция для генерации случайного числа в диапазоне [min, max]

int random\_int(minstd\_rand& rng, int min, int max) {

uniform\_int\_distribution<int> dist(min, max);

return dist(rng);

}

// Функция для генерации случайного символа ('A'-'Z', '0'-'9')

char random\_char(minstd\_rand& rng) {

uniform\_int\_distribution<int> dist(0, 35);

int num = dist(rng);

if (num < 26) {

return static\_cast<char>('A' + num);

} else {

return static\_cast<char>('0' + (num - 26));

}

}

// Функция для генерации случайного целого числа в диапазоне [100, 900]

int random\_number(minstd\_rand& rng) {

uniform\_int\_distribution<int> dist(100, 900);

return dist(rng);

}

// Функция для нахождения самой длинной убывающей последовательности

vector<int> longest\_decreasing\_sequence(const vector<int>& arr) {

vector<int> longest\_seq;

vector<int> current\_seq;

for (size\_t i = 1; i < arr.size(); ++i) {

if (arr[i] < arr[i - 1]) {

current\_seq.push\_back(arr[i - 1]);

} else {

current\_seq.push\_back(arr[i - 1]);

if (current\_seq.size() > longest\_seq.size()) {

longest\_seq = current\_seq;

}

current\_seq.clear();

}

}

current\_seq.push\_back(arr.back());

if (current\_seq.size() > longest\_seq.size()) {

longest\_seq = current\_seq;

}

return longest\_seq;

}

// Функция для вывода массива символов

void print\_chars(const vector<char>& chars) {

cout << "[";

for (size\_t i = 0; i < chars.size(); ++i) {

cout << chars[i];

if (i < chars.size() - 1) {

cout << ",";

}

}

cout << "]" << endl;

}

// Функция для вывода массива целых чисел

void print\_numbers(const vector<int>& numbers) {

cout << "[";

for (size\_t i = 0; i < numbers.size(); ++i) {

cout << numbers[i];

if (i < numbers.size() - 1) {

cout << ",";

}

}

cout << "]" << endl;

}

int main() {

minstd\_rand rng(random\_device{}());

// Часть 1: Работа с массивом целых чисел

int n;

cout << "Введите количество чисел в массиве (n >= 10): ";

cin >> n;

if (n < 10) {

cerr << "Ошибка: количество чисел должно быть не менее 10\n";

return 1;

}

vector<int> arr(n);

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

arr[i] = random\_int(rng, 150, 300);

}

cout << "Исходный массив:\n";

for (int num : arr) {

cout << num << " ";

}

cout << "\n";

vector<int> longest\_seq = longest\_decreasing\_sequence(arr);

cout << "Самая длинная убывающая последовательность:\n";

for (int num : longest\_seq) {

cout << num << " ";

}

cout << "\n";

double average = static\_cast<double>(accumulate(arr.begin(), arr.end(), 0)) / n;

vector<int> less\_than\_average;

for (int num : arr) {

if (num < average) {

less\_than\_average.push\_back(num);

}

}

cout << "Массив чисел, меньших среднеарифметического (" << average << "):\n";

for (int num : less\_than\_average) {

cout << num << " ";

}

cout << "\n";

// Часть 2: Работа с массивом символов

int m;

cout << "Введите размер массива символов (m >= 5): ";

cin >> m;

if (m < 5) {

cerr << "Ошибка: размер массива должен быть не менее 5\n";

return 1;

}

vector<char> chars(m);

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

chars[i] = random\_char(rng);

}

cout << "Исходный массив символов: ";

print\_chars(chars);

vector<char> moved\_chars;

moved\_chars.push\_back(chars[0]);

moved\_chars.push\_back(chars[1]);

chars.erase(chars.begin(), chars.begin() + 2);

chars.insert(chars.end(), moved\_chars.begin(), moved\_chars.end());

cout << "Модифицированный массив символов: ";

print\_chars(chars);

// Часть 3: Работа с массивом целых чисел в диапазоне [100, 900]

int p;

cout << "Введите размер массива целых чисел: ";

cin >> p;

vector<int> numbers(p);

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

numbers[i] = random\_number(rng);

}

cout << "Исходный массив чисел: ";

print\_numbers(numbers);

sort(numbers.begin(), numbers.end(), greater<int>());

cout << "Отсортированный массив чисел (по убыванию): ";

print\_numbers(numbers);

array<int, 10> digit\_counts = {0};

for (int number : numbers) {

while (number > 0) {

int digit = number % 10;

digit\_counts[digit]++;

number /= 10;

}

}

cout << "Количество вхождений каждой цифры в массиве:\n";

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

cout << "Цифра " << i << ": " << digit\_counts[i] << " раз(а)\n";

}

return 0;

}

**Язык программирования: Java  
Код программы:**import java.util.\*;

public class Main {

private static final Random rng = new Random();

public static int randomInt(int min, int max) {

return rng.nextInt(max - min + 1) + min;

}

public static char randomChar() {

int num = rng.nextInt(36);

if (num < 26) {

return (char) ('A' + num);

} else {

return (char) ('0' + (num - 26));

}

}

public static int randomNumber() {

return rng.nextInt(801) + 100;

}

public static List<Integer> longestDecreasingSequence(List<Integer> arr) {

List<Integer> longestSeq = new ArrayList<>();

List<Integer> currentSeq = new ArrayList<>();

for (int i = 1; i < arr.size(); ++i) {

if (arr.get(i) < arr.get(i - 1)) {

currentSeq.add(arr.get(i - 1));

} else {

currentSeq.add(arr.get(i - 1));

if (currentSeq.size() > longestSeq.size()) {

longestSeq = new ArrayList<>(currentSeq);

}

currentSeq.clear();

}

}

currentSeq.add(arr.get(arr.size() - 1));

if (currentSeq.size() > longestSeq.size()) {

longestSeq = new ArrayList<>(currentSeq);

}

return longestSeq;

}

public static void printChars(List<Character> chars) {

System.out.print("[");

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

System.out.print(chars.get(i));

if (i < chars.size() - 1) {

System.out.print(",");

}

}

System.out.println("]");

}

public static void printNumbers(List<Integer> numbers) {

System.out.print("[");

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

System.out.print(numbers.get(i));

if (i < numbers.size() - 1) {

System.out.print(",");

}

}

System.out.println("]");

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter the number of integers in the array (n >= 10): ");

int n = scanner.nextInt();

if (n < 10) {

System.err.println("Error: The number of integers must be at least 10");

return;

}

List<Integer> arr = new ArrayList<>();

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

arr.add(randomInt(150, 300));

}

System.out.println("Original array:");

printNumbers(arr);

List<Integer> longestSeq = longestDecreasingSequence(arr);

System.out.println("Longest decreasing sequence:");

printNumbers(longestSeq);

double average = arr.stream().mapToDouble(Integer::doubleValue).average().orElse(0.0);

List<Integer> lessThanAverage = new ArrayList<>();

for (int num : arr) {

if (num < average) {

lessThanAverage.add(num);

}

}

System.out.println("Array of numbers less than the arithmetic mean (" + average + "):");

printNumbers(lessThanAverage);

System.out.print("Enter the size of the character array (m >= 5): ");

int m = scanner.nextInt();

if (m < 5) {

System.err.println("Error: The size of the array must be at least 5");

return;

}

List<Character> chars = new ArrayList<>();

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

chars.add(randomChar());

}

System.out.print("Original character array: ");

printChars(chars);

List<Character> movedChars = new ArrayList<>();

movedChars.add(chars.get(0));

movedChars.add(chars.get(1));

chars.remove(0);

chars.remove(0);

chars.addAll(movedChars);

System.out.print("Modified character array: ");

printChars(chars);

System.out.print("Enter the size of the integer array: ");

int p = scanner.nextInt();

List<Integer> numbers = new ArrayList<>();

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

numbers.add(randomNumber());

}

System.out.print("Original number array: ");

printNumbers(numbers);

numbers.sort(Collections.reverseOrder());

System.out.print("Sorted number array (descending): ");

printNumbers(numbers);

int[] digitCounts = new int[10];

for (int number : numbers) {

while (number > 0) {

int digit = number % 10;

digitCounts[digit]++;

number /= 10;

}

}

System.out.println("Number of occurrences of each digit in the array:");

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

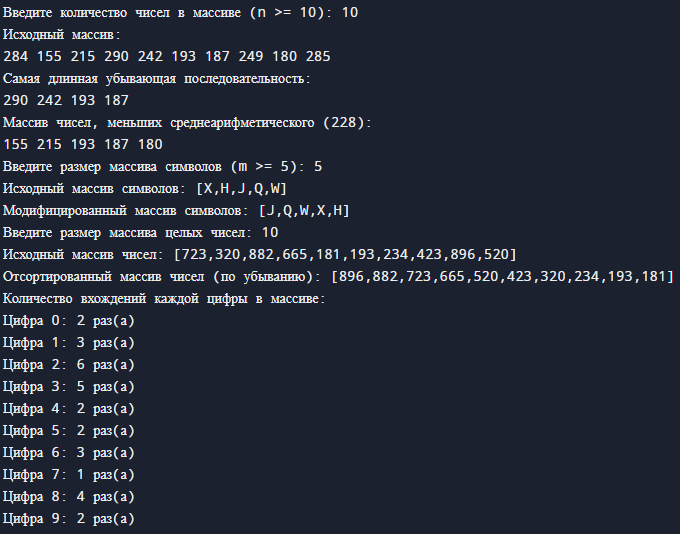
System.out.println("Digit " + i + ": " + digitCounts[i] + " time(s)");

}

}

}

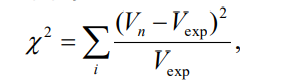
**Результат работы программы:**



**Задание №3 «Работа с генераторами рандомных чисел»**

Алгоритм - minstd\_rand

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. Значение критерия хи-квадрат для каждой из выборок рассчитывается по формуле:



где Vn – количество элементов, попавших в i-ый интервал;

Vexp – ожидаемое число попаданий в i-ый интервал

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

**Язык программирования: С++  
Код программы:**#include <iostream>

#include <vector>

#include <random> // Для minstd\_rand и uniform\_int\_distribution

#include <cmath> // Для вычислений

using namespace std;

// Функция для генерации случайного числа в диапазоне [1, 100]

int random\_number(minstd\_rand& rng) {

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

return dist(rng);

}

// Функция для вывода массива чисел (первые 10 элементов или все в зависимости от параметра show\_all)

void print\_array(const vector<int>& arr, bool show\_all = false) {

cout << "[";

int count = show\_all ? arr.size() : min(10, static\_cast<int>(arr.size())); // определяем количество элементов для вывода

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

cout << arr[i];

if (i < count - 1) {

cout << ",";

}

}

if (!show\_all && arr.size() > 10) {

cout << ",...";

}

cout << "]" << endl;

}

// Функция для вычисления критерия хи-квадрат

double chi\_square\_test(const vector<int>& arr, int expected\_value) {

int n = arr.size();

int num\_intervals = 10; // количество интервалов

int interval\_size = 100 / num\_intervals; // размер каждого интервала

vector<int> observed\_counts(num\_intervals, 0); // фактическое количество элементов в каждом интервале

// Заполнение фактического количества элементов в интервалах

for (int number : arr) {

int interval\_index = (number - 1) / interval\_size; // определение индекса интервала

observed\_counts[interval\_index]++;

}

// Вычисление критерия хи-квадрат

double chi\_square = 0.0;

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

double expected = static\_cast<double>(n) / num\_intervals; // ожидаемое количество элементов в каждом интервале

chi\_square += pow(observed\_counts[i] - expected, 2) / expected;

}

return chi\_square;

}

int main() {

minstd\_rand rng(random\_device{}()); // Инициализация генератора случайных чисел

// Создание и заполнение массивов размерами 50, 100 и 1000 случайными числами от 1 до 100

vector<int> arr50(50);

vector<int> arr100(100);

vector<int> arr1000(1000);

for (int i = 0; i < 50; ++i) {

arr50[i] = random\_number(rng);

}

for (int i = 0; i < 100; ++i) {

arr100[i] = random\_number(rng);

}

for (int i = 0; i < 1000; ++i) {

arr1000[i] = random\_number(rng);

}

// Вывод массивов (первые 10 элементов или все)

bool show\_all = false; // измените на true, чтобы вывести все элементы массивов

cout << "Массив из 50 элементов: ";

print\_array(arr50, show\_all);

cout << "Массив из 100 элементов: ";

print\_array(arr100, show\_all);

cout << "Массив из 1000 элементов: ";

print\_array(arr1000, show\_all);

// Вычисление критерия хи-квадрат для каждого массива

double chi\_square\_50 = chi\_square\_test(arr50, 50 / 10);

double chi\_square\_100 = chi\_square\_test(arr100, 100 / 10);

double chi\_square\_1000 = chi\_square\_test(arr1000, 1000 / 10);

// Вывод результатов теста

cout << "Результаты критерия хи-квадрат:\n";

cout << "Для массива из 50 элементов: " << chi\_square\_50 << endl;

cout << "Ожидаемое математическое ожидание: " << 50 / 10 << endl; // для массива из 50 элементов

cout << "Реальное математическое ожидание: " << static\_cast<double>(50) / 10 << endl; // для массива из 50 элементов

double chi\_critical\_value = 16.919; // значение хи-критического для степеней свободы 9 и уровня значимости 0.05

cout << "Хи-критическое значение для уровня значимости 0.05: " << chi\_critical\_value << endl;

cout << "Результат проверки гипотезы (превышение хи-критического): " << (chi\_square\_50 > chi\_critical\_value ? "Гипотеза отвергается" : "Гипотеза принимается") << endl;

cout << endl;

cout << "Для массива из 100 элементов: " << chi\_square\_100 << endl;

cout << "Ожидаемое математическое ожидание: " << 100 / 10 << endl; // для массива из 100 элементов

cout << "Реальное математическое ожидание: " << static\_cast<double>(100) / 10 << endl; // для массива из 100 элементов

cout << "Хи-критическое значение для уровня значимости 0.05: " << chi\_critical\_value << endl;

cout << "Результат проверки гипотезы (превышение хи-критического): " << (chi\_square\_100 > chi\_critical\_value ? "Гипотеза отвергается" : "Гипотеза принимается") << endl;

cout << endl;

cout << "Для массива из 1000 элементов: " << chi\_square\_1000 << endl;

cout << "Ожидаемое математическое ожидание: " << 1000 / 10 << endl; // для массива из 1000 элементов

cout << "Реальное математическое ожидание: " << static\_cast<double>(1000) / 10 << endl; // для массива из 1000 элементов

cout << "Хи-критическое значение для уровня значимости 0.05: " << chi\_critical\_value << endl;

cout << "Результат проверки гипотезы (превышение хи-критического): " << (chi\_square\_1000 > chi\_critical\_value ? "Гипотеза отвергается" : "Гипотеза принимается") << endl;

return 0;

}

**Язык программирования: Java  
Код программы:**

import java.util.Random;

import java.util.Arrays;

public class ChiSquareTest {

public static int randomNumber(Random rng) {

return rng.nextInt(100) + 1;

}

public static void printArray(int[] arr, boolean showAll) {

System.out.print("[");

int count = showAll ? arr.length : Math.min(10, arr.length);

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

System.out.print(arr[i]);

if (i < count - 1) {

System.out.print(",");

}

}

if (!showAll && arr.length > 10) {

System.out.print(",...");

}

System.out.println("]");

}

public static double chiSquareTest(int[] arr, int expectedValue) {

int n = arr.length;

int numIntervals = 10;

int intervalSize = 100 / numIntervals;

int[] observedCounts = new int[numIntervals];

for (int number : arr) {

int intervalIndex = (number - 1) / intervalSize;

observedCounts[intervalIndex]++;

}

double chiSquare = 0.0;

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

double expected = (double) n / numIntervals;

chiSquare += Math.pow(observedCounts[i] - expected, 2) / expected;

}

return chiSquare;

}

public static void main(String[] args) {

Random rng = new Random();

int[] arr50 = new int[50];

int[] arr100 = new int[100];

int[] arr1000 = new int[1000];

for (int i = 0; i < 50; ++i) {

arr50[i] = randomNumber(rng);

}

for (int i = 0; i < 100; ++i) {

arr100[i] = randomNumber(rng);

}

for (int i = 0; i < 1000; ++i) {

arr1000[i] = randomNumber(rng);

}

boolean showAll = false;

System.out.print("Массив из 50 элементов: ");

printArray(arr50, showAll);

System.out.print("Массив из 100 элементов: ");

printArray(arr100, showAll);

System.out.print("Массив из 1000 элементов: ");

printArray(arr1000, showAll);

double chiSquare50 = chiSquareTest(arr50, 50 / 10);

double chiSquare100 = chiSquareTest(arr100, 100 / 10);

double chiSquare1000 = chiSquareTest(arr1000, 1000 / 10);

System.out.println("Результаты критерия хи-квадрат:");

System.out.println("Для массива из 50 элементов: " + chiSquare50);

System.out.println("Ожидаемое математическое ожидание: " + (50 / 10));

System.out.println("Реальное математическое ожидание: " + ((double) 50 / 10));

double chiCriticalValue = 16.919;

System.out.println("Хи-критическое значение для уровня значимости 0.05: " + chiCriticalValue);

System.out.println("Результат проверки гипотезы (превышение хи-критического): " + (chiSquare50 > chiCriticalValue ? "Гипотеза отвергается" : "Гипотеза принимается"));

System.out.println();

System.out.println("Для массива из 100 элементов: " + chiSquare100);

System.out.println("Ожидаемое математическое ожидание: " + (100 / 10));

System.out.println("Реальное математическое ожидание: " + ((double) 100 / 10));

System.out.println("Хи-критическое значение для уровня значимости 0.05: " + chiCriticalValue);

System.out.println("Результат проверки гипотезы (превышение хи-критического): " + (chiSquare100 > chiCriticalValue ? "Гипотеза отвергается" : "Гипотеза принимается"));

System.out.println();

System.out.println("Для массива из 1000 элементов: " + chiSquare1000);

System.out.println("Ожидаемое математическое ожидание: " + (1000 / 10));

System.out.println("Реальное математическое ожидание: " + ((double) 1000 / 10));

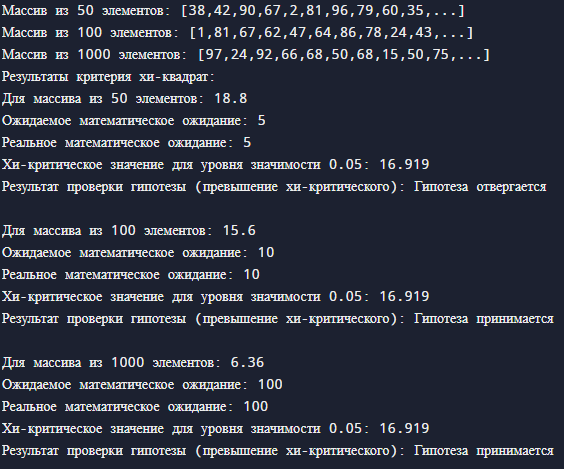
System.out.println("Хи-критическое значение для уровня значимости 0.05: " + chiCriticalValue);

System.out.println("Результат проверки гипотезы (превышение хи-критического): " + (chiSquare1000 > chiCriticalValue ? "Гипотеза отвергается" : "Гипотеза принимается"));

}

}

**Результат работы программы:**



**Задание №4 «Дилемма заключенного»**

Требуется реализовать игру «Предать или сотрудничать» и реализовать 3 алгоритма поведения в игре. Игра состоит из случайного кол-ва раундов от 100 до 200 (итоговое кол-во раундов при каждом запуске игры генерируется случайно). На протяжении игровой сессии сражаются 2 алгоритма. В каждом раунде каждый алгоритм выбирает, либо сотрудничество, либо предательство. Если алгоритм А выбирает предательство и алгоритм Б выбирает предательство они получают по 4 очка. Если алгоритм А выбирает сотрудничество, а алгоритм Б выбирает предательство - алгоритм А получает 0 очков, а алгоритм Б получает 20 очков. Если оба алгоритма выбирают сотрудничество оба получают 24 очка. Каждому алгоритму в каждом раунде известны результаты всех предыдуще раундов текущей игровой сессии, на основе этих данных алгоритм может выбирать будет он сотрудничать или предаст. Каждый алгоритм должен представлять из себя функцию с сигнатурой:

* boolean func(int32 round\_number, array[boolean] self\_choices, array[boolean] enemy\_choices)
* round\_number – номер текущего
* self\_choices – массив булевых значений, содержит информацию о собственных выборов (предать или сотрудничать) за все предыдущие раунды
* enemy\_choices - массив булевых значений, содержит информацию о выборов (предать или сотрудничать) противника за все предыдущие раунды
* true – сотруднничество
* false – предательство.

**Язык программирования: С++  
Код программы:**#include <iostream>

#include <random>

#include <vector>

using namespace std;

random\_device rd;

ranlux48\_base random\_(rd());

bool algorithm1(int32\_t round\_number, const vector<bool>& self\_choices, const vector<bool>& enemy\_choices) { // четные раунды предаем,в нечетных сотрудничество

if (round\_number % 2 == 0) {

return false;

} else {

return true;

}

}

bool algorithm2(int32\_t round\_number, const vector<bool>& self\_choices, const vector<bool>& enemy\_choices) { // алгоритм с отступом

if (round\_number == 0) {

return true; // Начинает с сотрудничества

}

if (round\_number > 0 && round\_number % 3 == 0) {

return true; // Сотрудничает каждый третий раунд

}

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

return true; // Если противник сотрудничал в прошлом раунде, сотрудничаем и мы

}

return false; // Иначе предаем

}

bool algorithm3() { // рандом

return random\_() % 2;

}

void game1\_2(int number) { // 1 и 2 алгоритм

int scoreA = 0, scoreB = 0;

vector<bool> choicesA;

vector<bool> choicesB;

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

bool choiceA = algorithm1(i, choicesA, choicesB);

bool choiceB = algorithm2(i, choicesB, choicesA);

choicesA.push\_back(choiceA);

choicesB.push\_back(choiceB);

if (choiceA && choiceB) {

scoreA += 24;

scoreB += 24;

}

else if (choiceA == true && choiceB == false) {

scoreB += 20;

}

else if (choiceA == false && choiceB == true) {

scoreA += 20;

}

else {

scoreA += 4;

scoreB += 4;

}

}

cout << "Результат 1 алгоритма: " << scoreA << endl << "Результат 2 алгоритма: " << scoreB << endl << endl;

}

void game2\_3(int number) { // 2 и 3 алгоритм

int scoreA = 0, scoreB = 0;

vector<bool> choicesA;

vector<bool> choicesB;

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

bool choiceA = algorithm2(i, choicesA, choicesB);

bool choiceB = algorithm3();

choicesA.push\_back(choiceA);

choicesB.push\_back(choiceB);

if (choiceA && choiceB) {

scoreA += 24;

scoreB += 24;

}

else if (choiceA == true && choiceB == false) {

scoreB += 20;

}

else if (choiceA == false && choiceB == true) {

scoreA += 20;

}

else {

scoreA += 4;

scoreB += 4;

}

}

cout << "Результат 2 алгоритма: " << scoreA << endl << "Результат 3 алгоритма: " << scoreB << endl << endl;

}

void game1\_3(int number) { // 1 и 3 алгоритм

int scoreA = 0, scoreB = 0;

vector<bool> choicesA;

vector<bool> choicesB;

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

bool choiceA = algorithm1(i, choicesA, choicesB);

bool choiceB = algorithm3();

choicesA.push\_back(choiceA);

choicesB.push\_back(choiceB);

if (choiceA && choiceB) {

scoreA += 24;

scoreB += 24;

}

else if (choiceA == true && choiceB == false) {

scoreB += 20;

}

else if (choiceA == false && choiceB == true) {

scoreA += 20;

}

else {

scoreA += 4;

scoreB += 4;

}

}

cout << "Результат 1 алгоритма: " << scoreA << endl << "Результат 3 алгоритма: " << scoreB << endl << endl;

}

int main() {

setlocale(LC\_ALL, "Russian");

int number = random\_() % (101) + 100;

cout << "Количество раундов: " << number << endl;

game1\_2(number);

game2\_3(number);

game1\_3(number);

return 0;

} **Язык программирования: Java  
Код программы:**

import java.util.Random;

import java.util.Vector;

public class PrisonersDilemma {

private static Random random = new Random();

public static boolean algorithm1(int roundNumber, Vector<Boolean> selfChoices, Vector<Boolean> enemyChoices) {

if (roundNumber % 2 == 0) {

return false;

} else {

return true;

}

}

public static boolean algorithm2(int roundNumber, Vector<Boolean> selfChoices, Vector<Boolean> enemyChoices) {

if (roundNumber == 0) {

return true;

}

if (roundNumber > 0 && roundNumber % 3 == 0) {

return true;

}

if (enemyChoices.get(roundNumber - 1)) {

return true;

}

return false;

}

public static boolean algorithm3() {

return random.nextBoolean();

}

public static void game1\_2(int number) {

int scoreA = 0, scoreB = 0;

Vector<Boolean> choicesA = new Vector<>();

Vector<Boolean> choicesB = new Vector<>();

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

boolean choiceA = algorithm1(i, choicesA, choicesB);

boolean choiceB = algorithm2(i, choicesB, choicesA);

choicesA.add(choiceA);

choicesB.add(choiceB);

if (choiceA && choiceB) {

scoreA += 24;

scoreB += 24;

} else if (choiceA && !choiceB) {

scoreB += 20;

} else if (!choiceA && choiceB) {

scoreA += 20;

} else {

scoreA += 4;

scoreB += 4;

}

}

System.out.println("Результат 1 алгоритма: " + scoreA);

System.out.println("Результат 2 алгоритма: " + scoreB);

System.out.println();

}

public static void game2\_3(int number) {

int scoreA = 0, scoreB = 0;

Vector<Boolean> choicesA = new Vector<>();

Vector<Boolean> choicesB = new Vector<>();

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

boolean choiceA = algorithm2(i, choicesA, choicesB);

boolean choiceB = algorithm3();

choicesA.add(choiceA);

choicesB.add(choiceB);

if (choiceA && choiceB) {

scoreA += 24;

scoreB += 24;

} else if (choiceA && !choiceB) {

scoreB += 20;

} else if (!choiceA && choiceB) {

scoreA += 20;

} else {

scoreA += 4;

scoreB += 4;

}

}

System.out.println("Результат 2 алгоритма: " + scoreA);

System.out.println("Результат 3 алгоритма: " + scoreB);

System.out.println();

}

public static void game1\_3(int number) {

int scoreA = 0, scoreB = 0;

Vector<Boolean> choicesA = new Vector<>();

Vector<Boolean> choicesB = new Vector<>();

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

boolean choiceA = algorithm1(i, choicesA, choicesB);

boolean choiceB = algorithm3();

choicesA.add(choiceA);

choicesB.add(choiceB);

if (choiceA && choiceB) {

scoreA += 24;

scoreB += 24;

} else if (choiceA && !choiceB) {

scoreB += 20;

} else if (!choiceA && choiceB) {

scoreA += 20;

} else {

scoreA += 4;

scoreB += 4;

}

}

System.out.println("Результат 1 алгоритма: " + scoreA);

System.out.println("Результат 3 алгоритма: " + scoreB);

System.out.println();

}

public static void main(String[] args) {

int number = random.nextInt(101) + 100;

System.out.println("Количество раундов: " + number);

game1\_2(number);

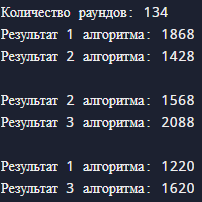
game2\_3(number);

game1\_3(number);

}

}

**Результат работы программы:**



**Задание № 5  
Вариант — 1**

Линейный конгруэнтный генератор. Последовательность чисел X1, X2, …, определена как Xi = (A\*Xi-1+B) mod C,

где С - диапазон значений, при этом C > 0; А - множитель (0 <= А <= С); В - инкрементирующее значение (0 <= B <= C). Также задается значение seed: X0 - начальное значение (0 <= X0 < C).

Так как числа последовательности образуются с использованием деления по модулю, в какой-то момент последовательность начнет повторяться. Необходимо вывести сгенерированную последовательность и найти индекс, с которого начинается повторение последовательности.

**Язык программирования: С++  
Код программы:**#include <iostream>

#include <vector>

#include <unordered\_set>

using namespace std;

struct Generator { //Определяет структуру `Generator`, которая представляет собой линейный конгруэнтный генератор.

int X0, A, B, C;

unordered\_set<int> sequence; // Множества для хранения сгенерированных чисел и повторяющихся чисел соответственно.

unordered\_set<int> repeating;

Generator(int X0, int A, int B, int C) : X0(X0), A(A), B(B), C(C) {} // Конструктор структуры `Generator`, инициализирующий поля значениями, переданными в качестве аргументов.

int next() {

int current = (A \* X0 + B) % C; // генерируем число по формуле

if (sequence.count(current) > 0) { // проверяет было ли сгенерированно это число

repeating.insert(current);//добавляем число во множество повторяющихся чисел

return -1; //возращаем -1

}

sequence.insert(current); //добавляем числов последовательность

X0 = current; //сгенерированное число становится текущим значением

return current;

}

};

int main() {

int X0, A, B, C; //x0- начальное значение , а- множитель , в- - инкрементирующее значение, С - диапазон значений

cout << "Enter X0, A, B, C: ";

cin >> X0 >> A >> B >> C;

Generator gen(X0, A, B, C);

vector<int> sequence; // Объявляет вектор для хранения сгенерированной последовательности чисел.

sequence.push\_back(X0); // вносим в вектор начальное значение

int index = 0;

while (true) {

int num = gen.next(); //генерация след числа

if (num == -1) // если -1,то началось повторение

break;

sequence.push\_back(num); //добавление сгенерированного числа в вектор

index++; //увеличиваем индекс

}

cout << "Generated sequence: ";

for (int num : sequence) {

cout << num << " ";

}

cout << "\nIndex of repeating sequence: " << index + 1 << endl;

return 0;

}

**Язык программирования: Java  
Код программы:**

import java.util.\*;

class Generator {

private int X0, A, B, C;

private Set<Integer> sequence;

private Set<Integer> repeating;

public Generator(int X0, int A, int B, int C) {

this.X0 = X0;

this.A = A;

this.B = B;

this.C = C;

this.sequence = new HashSet<>();

this.repeating = new HashSet<>();

}

public int next() {

int current = (A \* X0 + B) % C;

if (sequence.contains(current)) {

repeating.add(current);

return -1;

}

sequence.add(current);

X0 = current;

return current;

}

}

public class Main {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter X0, A, B, C: ");

int X0 = scanner.nextInt();

int A = scanner.nextInt();

int B = scanner.nextInt();

int C = scanner.nextInt();

Generator gen = new Generator(X0, A, B, C);

List<Integer> sequence = new ArrayList<>();

sequence.add(X0);

int index = 0;

while (true) {

int num = gen.next();

if (num == -1)

break;

sequence.add(num);

index++;

}

System.out.print("Generated sequence: ");

for (int num : sequence) {

System.out.print(num + " ");

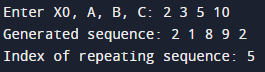
}

System.out.println("\nIndex of repeating sequence: " + (index + 1));

}

}

**Результат работы программы:**



**Вывод**

В ходе лабораторной работы были выполнены все поставленные задачи, которые были направлены на изучение алгоритмов формирования и обработки одномерных массивов и последовательных контейнеров, программирование и отладка программ формирования и обработки массивов., на двух языках программирования: C++ и Java.