Министерство науки и высшего образования Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования «Тульский государственный университет»

КАФЕДРА ИНФОРМАЦИОННОЙ БЕЗОПАСНОСТИ

**РАБОТА С МАССИВАМИ В ЯЗЫКЕ ПРОГРАММИРОВАНИЯ C**

отчет о лабораторной работе № 4

по дисциплине

*ТЕХНОЛОГИИ И МЕТОДЫ ПРОГРАММИРОВАНИЯ*

***ВАРИАНТ 13***

|  |  |  |
| --- | --- | --- |
| Выполнила: | ст. гр. 230711 | Павлова В.С. |
| Проверил: | асс. каф. ИБ | Курбаков М.Ю. |

Тула, 2023 г.

**ЦЕЛЬ И ЗАДАЧА РАБОТЫ**

**Цель:** изучение структурированных типов данных – одномерный массив и двумерный массив.

**Задача:** написать программу, реализующую действия с одномерными и двумерными массивами соответственно варианту. В вариантах для работы с двумерными массивами задандвухмерный массив , где n>1, m>1.

**ЗАДАНИЕ НА РАБОТУ**

**№1. Задание по теме «Одномерные массивы»:** задан массив. Определить сколько раз меняется знак в данной последовательности чисел, запомнить номера позиций, в которых меняется знак.

**№2. Задание по теме «Многомерные массивы»**: вычислить сумму элементов матрицы вдоль ее диагоналей.

# **СХЕМА ПРОГРАММЫ ДЛЯ ЗАДАНИЯ №1**

Схема алгоритма программы, реализующей обработку одномерного массива по условию задания представлена на рисунке 1.

  
Рисунок 1 – Схема алгоритма программы, реализующей обработку одномерного массива по условию задания

# **ТЕКСТ ПРОГРАММЫ ДЛЯ ЗАДАНИЯ №1**

Текст программы на языке программирования С/С++, реализующей обработку одномерного массива по заданию представлен в листинге 1.

## **Листинг 1. Текст программы**

#include <iostream>

int main() {

setlocale(LC\_ALL, "RUSSIAN");

const int n = 10;

std::cout << "Установлен текущий размер массива: " << n << "\n";

int a[n], changes[n];

std::cout << "Введите элементы массива: ";

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

{

std::cin >> a[i];

changes[i] = 0;

}

int sign = (a[0] >= 0 ? 1 : -1);

int count = 0;

for (int i = 1; i < n; i++) {

if ((a[i] >= 0 && sign < 0) || (a[i] < 0 && sign >= 0)) {

changes[i] = 1;

sign \*= -1;

count++;

}

}

std::cout << "Число изменений знака: " << count << "\n";

std::cout << "Позиции изменения знака: ";

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

if (changes[i] == 1) std::cout << i << " ";

} std::cout << "\n";

return 0;

}

**ИНСТРУКЦИЯ ПРОГРАММИСТА ДЛЯ ЗАДАНИЯ №1**

Структуры данных, используемые в программе, приведены в таблице 1.

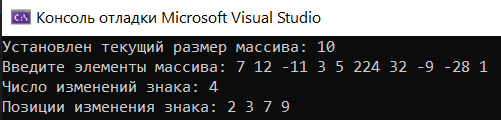
Таблица 1 – Структуры данных в программе

|  |  |  |
| --- | --- | --- |
| **Имя** | **Тип** | **Предназначение** |
| sign | int | Знак числа (+1 или -1) |
| n | int | Размер массива |
| a | int[n] | Массив чисел |
| changed | int[n] | Массив отслеживания изменения знаков чисел |

# **ДЕМОНСТРАЦИОННЫЙ ПРИМЕР ДЛЯ ЗАДАНИЯ №1**

Рассмотрим следующий пример. Для количества чисел n = 10 и их набора: *7 12 -11 3 5 224 32 -9 -28 12* число изменений знака равно **4**, поскольку знак меняется с «+» на «-» и наоборот у чисел {12; -11; 3}, а также с «+» на «-» и с «-» на «+» на участке {32; -9; -28; 12}.

Результат работы программы с соответствующим набором данных представлен на рисунке 2.

  
Рисунок 2 – Пример работы программы

# **СХЕМА ПРОГРАММЫ ДЛЯ ЗАДАНИЯ №2**

Схема алгоритма программы, реализующей обработку двумерного массива согласно заданию представлена на рисунке 3.



Рисунок 3 – Схема алгоритма программы, реализующей обработку двумерного массива согласно заданию

# **ТЕКСТ ПРОГРАММЫ ДЛЯ ЗАДАНИЯ №2**

Текст программы на языке программирования С/С++, реализующей обработку двумерного массива согласно заданию представлен в листинге 2.

## **Листинг 2. Текст программы**

#include <iostream>

#include <vector>

int main()

{

setlocale(LC\_ALL, "russian");

const int n = 5, m = 5;

std::cout << "Установлены текущие размеры матрицы: " << m << "x" << n << "\n";

int a[m][n];

std::cout << "Введите элементы матрицы:\n";

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

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

std::cin >> a[i][j];

}

}

int sum1 = 0, sum2 = 0;

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

{

sum1 += a[i][i]; // сумма элементов на главной диагонали

sum2 += a[i][m - i - 1]; // сумма элементов на побочной диагонали

}

std::cout << "Сумма элементов на главной диагонали: " << sum1 << "\n";

std::cout << "Сумма элементов на побочной диагонали: " << sum2 << "\n";

return 0;

}

**ИНСТРУКЦИЯ ПРОГРАММИСТА ДЛЯ ЗАДАНИЯ №2**

Структуры данных, используемые в программе, приведены в таблице 2.

Таблица 2 – Структуры данных в программе

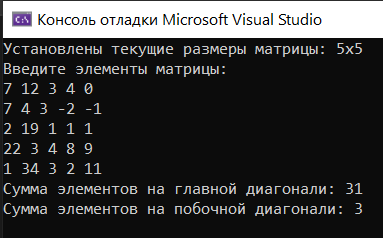
|  |  |  |
| --- | --- | --- |
| **Имя** | **Тип** | **Предназначение** |
| n,m | int | Размерность массива (MxN) |
| sum1 | int | Сумма элементов на главной диагонали |
| sum2 | int | Сумма элементов на побочной диагонали |
| a | int[m][n] | Матрица |

# **ДЕМОНСТРАЦИОННЫЙ ПРИМЕР ДЛЯ ЗАДАНИЯ №2**

Рассмотрим следующий пример с матрицей размером 5x5:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **7** | 12 | 3 | 4 | 0 | Для данного примера сумма элементов вдоль главной диагонали равна 7 + 4 + 1 + 8 + 11 = 31, вдоль побочной равна 0 + (-2) + 1 + 3 + 1 = 3. |
| 7 | **4** | 3 | -2 | -1 |
| 2 | 19 | **1** | 1 | 1 |
| 22 | 3 | 4 | **8** | 9 |
| 1 | 34 | 3 | 2 | **11** |

Результат работы программы изображен на рисунке 4.

  
Рисунок 4 – Пример работы программы

# **ВЫВОДЫ**

В ходе выполнения работы я ознакомилась с такой структурой данной, как массив – совокупностью переменных одного типа, к которым обращаются с помощью общего имени. Массивы могут быть одномерными и многомерными (массивы массивов), они представляют собой удобное средство группирования связанных переменных. В памяти массивы хранятся в подряд идущих ячейках.