## Лабораторная работа №4

## Статические одномерные массивы

***Цель лабораторной работы:*** *изучение структурной организации массивов и способов доступа к их элементам; совершенствование навыков процедурного программирования на языке C/C++ при решении задач обработки статических одномерных массивов.*

***Задание на программирование:*** *используя технологию процедурного программирования, разработать программу обработки одномерных статических массивов в соответствии с индивидуальным заданием.*

***Порядок выполнения работы:***

1. Получить у преподавателя индивидуальное задание и выполнить постановку задачи: сформулировать условие, определить входные и выходные данные, их ограничения.

2. Разработать математическую модель: описать с помощью формул и рисунков структуру массивов и процесс их преобразования.

3. Провести структуризацию задачи. С этой целью выделить в ней подзадачи, которые будут реализованы в виде отдельных функций (ввод исходных данных, вывод содержимого массива, обработка массива в соответствии с заданием и т.д.). При этом запрещается совмещать в одной функции решение нескольких подзадач.

4. Обязательное требование – введение собственных типов данных, логически точно разделяющих и группирующих информацию, используемую при решении конкретной задачи (тип значения элемента массива, тип массива и т.п.).

5. Построить схемы алгоритмов основных функций.

6. Составить программу на языке *C/C++*.

7. Входные данные на этапах тестирования и демонстрации работы преподавателю должны задаваться либо с использованием специально подобранных арифметических формул, либо вводиться с клавиатуры по запросу. **Датчики псевдослучайных чисел использовать запрещается**.

Если это явно не оговорено в конкретном варианте задания, значения элементов в каждом исходном массиве **должны быть разными**.

8. Выходные данные должны выводиться на экран с пояснениями. Операторы вывода любых результатов работы должны находиться либо в функции *main*(), либо в специальной функции вывода (например, преобразованного массива), вызов которой осуществляется из функции *main*().

9. Проверить работу программы на полном наборе тестов. Входные и выходные массивы должны выводиться в одном и том же формате.

10. Использовать стандартные потоковые объекты ввода/вывода ***cin*** и ***cout***.

11. Оформить отчет о лабораторной работе в составе: постановка задачи, математическая модель, схемы алгоритмов основных функций, текст программы, контрольные примеры (скриншоты).

12. Скриншоты тестов должны легко читаться. Все их неинформативные части должны быть удалены.

Номер зачетной книжки 2022/ 5356. Вариант - 7

***Задание на программирование:*** *используя технологию процедурного программирования, разработать программу обработки одномерных статических массивов в соответствии с индивидуальным заданием.*

Дан массив *z*0, *z*1,…, *zn-*1. Определить произведение суммы абсолютных значений элементов массива с чётными номерами, предшествующих последнему элементу массива с отрицательным значением, на общее количество элементов массива с отрицательными значениями.

Провести структуризацию задачи. С этой целью выделить в ней подзадачи, которые будут реализованы в виде отдельных функций (ввод исходных данных, вывод содержимого массива, обработка массива в соответствии с заданием и т.д.). При этом запрещается совмещать в одной функции решение нескольких подзадач.

Обязательное требование – введение собственных типов данных, логически точно разделяющих и группирующих информацию, используемую при решении конкретной задачи (тип значения элемента массива, тип массива и т.п.).

Выходные данные должны выводиться на экран с пояснениями. Операторы вывода любых результатов работы должны находиться либо в функции *main*(), либо в специальной функции вывода (например, преобразованного массива), вызов которой осуществляется из функции *main*().

Проверить работу программы на полном наборе тестов. Входные и выходные массивы должны выводиться в одном и том же формате.

Использовать стандартные потоковые объекты ввода/вывода ***cin*** и ***cout***.

***Математическая модель***

Решение задачи начинается с ввода исходных данных. Из формулировки задачи понятно, что исходный массив имеет размер и n количество элементов.

Для удобства хранения информации введем структуру ArrayData, которая содержит массив elements и размер size. Это позволяет хранить и передавать информацию о массиве вместе.

Функция ввода массива (void inputArray(ArrayData& arrayData)).

Пусть максимальный размер массива N = 100.

Вводится размер массива n.

Затем вводятся значения элементов массива в строку.

Для вывода значений элементов массива используется функция вывода массива (void printArray(const ArrayData& arrayData)).

При этом идет проверка на определение n-1 элемента с отрицательным значением и подсчитывается количество элементов с отрицательными значениями.

При этом могут быть два варианта: отрицательных значений вообще нет или хотя бы один такой элемент есть. Вначале предполагаем, что отрицательных значений нет, и присваиваем переменной *lastIndex== -1* значение -1 (такого индекса у элементов массива не может быть). После этого ищем первый элемент массива с отрицательным значением. Если такой элемент в массиве есть, фиксируем индекс этого элемента *lastIndex*. Величину этого элемента массива принимаем за исходное отрицательное значение элемента.

Далее, перебирая индексы остальных элементов массива с *i*+1 до n-1, для отрицательных элементов сравниваем их значения со значением элемента с индексом *lastIndex*. Если значение элемента массива с текущим индексом меньше значения элемента массива с индексом *lastIndex*, запоминаем индекс этого элемента, меняя значение *lastIndex.*

Если ни один элемент массива с отрицательным значением не найден, выполнение решения всей задачи прекращается и выводится результат 0.

Далее идет подсчет количества отрицательных элементов массива сount++.

Если элемент массива найден, то результат в нашем случае выглядит следующим образом:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | **1** | **2** | **3** | ***4*** |  |  |  | **n-2** | **n-1** |
| **0** | **1** | **2** | **3** | ***4*** | ***5*** | **6** | **7** | **8** | **9** |
| **1** | **2** | **0** | **-3** | **4** | **5** | **6** | **-7** | **-8** | **9** |
|  |  |  |  |  |  |  |  | **lastNegIndex** |  |

Далее идет перебор, среди них выбрать элементы с четными индексами. При этом при нахождении элемента с четным индексом, его значение приплюсовывается к сумме абсолютных значений элементов с четными номерами (sum+).

В итоге сумма абсолютных значений элементов с четными номерами (sum+) умножается на количество отрицательных элементов в массиве (сount) и получаем ответ.

Задача решена.

***Пример:***

n=10

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

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | **1** | **2** | **3** | ***4*** | ***5*** | **6** | **7** | **8** | **9** |
| **1** | **2** | **0** | **-3** | **4** | **5** | **6** | **-7** | **-8** | **9** |
|  |  |  |  |  |  |  |  | **lastIndex** |  |

Количество отрицательных элементов (negCount) - 3

Сумма значений четных элементов =1+0+4+6=11

Произведение суммы на количество отрицательных элементов =33

Реализация программы в двух вариантах.

1. Сначала заданы псевдотипы данных, затем создан свой тип данных ArrayData, который содержит массив elements и размер size. Это позволяет хранить и передавать информацию о массиве вместе.
2. Создан свой тип данных ArrayData, который содержит указатель на массив double\* RealArray с вещественным типом данных, и размер массива int size с целочисленным типом данных и константу – максимальный размер массива целочисленным типом данных. Это позволяет хранить и передавать информацию о массиве вместе. Оператор new позволяет выделять память.

В программе предусмотрена проверка на некорректный ввод размера массива, то есть ввод элементов массива до тех пор, пока не будет введено корректное значение.

Также программа предлагает пользователю повторить или завершить выполнение задачи после вывода результата.

Добавлена директива **#include <locale.h>** для работы с русскими символами.

Добавлена директива **#include <cmath>**для работы с математическими формулами.

Добавлена директива **#include <limits>** для определения минимальных и максимальных значений величин.

Результаты двух вариантов программы одинаковые.

***Схема алгоритма для определения индекса последнего отрицательного элемента***

*Начало* ***функция findLastNegativeIndex(const ArrayData& arrayData)***

*lastIndex= -1*

*ArraySize i*=*0*;

*i*< *arrayData.size*; *i*++

*arrayData.elements[i]*

*<0*

да

*lastIndex=i*

*arrayData.elements[i]*

нет

*return lastIndex*

***Схема алгоритма для определения количества отрицательных элементов***

*Начало* ***функция countNegativeElements(const ArrayData& arrayData)***

*count=0*

*ArraySize i*=*0*;

*i*< *arrayData.size*; *i*++

*arrayData.elements[i]*

*<0*

да

*count++*

*arrayData.elements[i]*

нет

*return count*

***Схема алгоритма для определения суммы абсолютных значений элементов массива с четными номерами, предшествующих последнему элементу массива с отрицательным значением***

*Начало sumEvenElements (const ArrayData& arrayData)*

*sum*=0

*lastIndex = findLastNegativeIndex(arrayData)*

ArraySize i = 0;

i <= lastIndex-1; i += 2

нет

arrayData.elements[i]

*i % 2 =0*

нет да

*sum += abs(arrayData.elements[i])*

*return sum*

***Текст программы***

***Вариант 1***

#include <iostream>

#include<stdlib.h>

#include <locale.h>

#include <limits>

using namespace std;

const int MAX\_SIZE = 100; // определение константы

typedef double RealArray[MAX\_SIZE]; // тип значений элементов массива

typedef int ArraySize; // тип массива

struct ArrayData {

ArraySize size; // размер массива

RealArray elements; // элементы массива

};

void inputArray(ArrayData& arrayData); // Функция ввода массива

void printArray(const ArrayData& arrayData); // Функция вывода массива

int findLastNegativeIndex(const ArrayData& arrayData); // Функция для определения индекса последнего отрицательного элемента

int countNegativeElements(const ArrayData& arrayData); // Функция для определения количества отрицательных элементов

double sumEvenElements(const ArrayData& arrayData); // Функция для определения суммы абсолютных значений элементов массива с четными номерами, предшествующих последнему элементу массива с отрицательным значением

double calculateProduct(double sum, int count); // Функция для вычисления произведения суммы абсолютных значений элементов массива с четными номерами, предшествующих последнему элементу массива с отрицательным значением

// Функция ввода массива

void inputArray(ArrayData& arrayData) {

cout << "Введите размер массива: ";

while (!(cin >> arrayData.size) || arrayData.size <= 0 || arrayData.size > MAX\_SIZE) {

cout << "Неверные данные! Введите размер массива (не более " << MAX\_SIZE << "): ";

cin.clear();

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

}

cout << "Введите в одной строке элементы массива, состоящего из " << arrayData.size;

cout << " вещественных чисел, и нажмите <Enter>" << endl;

for (ArraySize i = 0; i < arrayData.size; i++) {

while (!(cin >> arrayData.elements[i])) {

cout << "Некорректный ввод! Попробуйте снова: ";

cin.clear();

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

}

}

}

// Функция вывода массива

void printArray(const ArrayData& arrayData) {

cout << "Исходный массив:" << endl;

for (ArraySize i = 0; i < arrayData.size; i++) {

cout.width(4);

cout << arrayData.elements[i] << " ";

}

cout << endl;

}

// Функция для определения индекса последнего отрицательного элемента

int findLastNegativeIndex(const ArrayData& arrayData) {

int lastIndex = -1;

for (ArraySize i = 0; i < arrayData.size; i++) {

if (arrayData.elements[i] < 0) {

lastIndex = i;

}

}

return lastIndex;

}

// Функция для определения количества отрицательных элементов

int countNegativeElements(const ArrayData& arrayData) {

int count = 0;

for (ArraySize i = 0; i < arrayData.size; i++) {

if (arrayData.elements[i] < 0) {

count++;

}

}

return count;

}

// Функция для определения суммы абсолютных значений элементов массива с четными номерами, предшествующих последнему элементу массива с отрицательным значением

double sumEvenElements(const ArrayData& arrayData) {

double sum = 0;

int lastIndex = findLastNegativeIndex(arrayData);

for (ArraySize i = 0; i <= lastIndex-1; i += 2) {

sum += abs(arrayData.elements[i]);

}

return sum;

}

// Функция для вычисления произведения суммы абсолютных значений элементов массива с четными номерами, предшествующих последнему элементу массива с отрицательным значением

double calculateProduct(double sum, int count) {

return sum \* count;

}

int main() {

setlocale(LC\_ALL, "Russian");

ArrayData arrayData;

int choice;

do {

inputArray(arrayData);

printArray(arrayData);

int lastIndex = findLastNegativeIndex(arrayData);

if (lastIndex == -1) {

cout << "Отрицательных элементов в массиве нет." << endl;

cout << " Результат 0" << endl;

} else {

int count = countNegativeElements(arrayData);

double sum = sumEvenElements(arrayData);

double result = calculateProduct(sum, count);

cout << "Количество отрицательных элементов: " << count << endl;

cout << "Индекс последнего отрицательного элемента: " << lastIndex << endl;

cout << "Сумма: " << sum << endl;

cout << "Результат: " << result << endl;

}

cout << "Повторить - 1, Выход - 2: ";

cin >> choice;

} while (choice == 1);

return 0;

}

***Текст программы***

***Вариант 2***

#include <iostream>

#include <cmath>

#include <limits>

using namespace std;

struct ArrayData {

const int MAX\_SIZE = 100;

double\* RealArray; // указатель на массив

int size; // размер массива

};

void inputArray(ArrayData& arrayData); // Функция ввода массива

void printArray(const ArrayData& arrayData); // Функция вывода массива

int findLastNegativeIndex(const ArrayData& arrayData); // Функция для определения индекса последнего отрицательного элемента

int countNegativeElements(const ArrayData& arrayData); // Функция для определения количества отрицательных элементов

double sumEvenElements(const ArrayData& arrayData); // Функция для определения суммы абсолютных значений элементов массива с четными номерами, предшествующих последнему элементу массива с отрицательным значением

double calculateProduct(double sum, int count); // Функция для вычисления произведения суммы абсолютных значений элементов массива с четными номерами, предшествующих последнему элементу массива с отрицательным значением

// Функция ввода массива

void inputArray(ArrayData& arrayData) {

cout << "Введите размер массива: ";

while (!(cin >> arrayData.size) || arrayData.size <= 0 || arrayData.size > arrayData.MAX\_SIZE) {

cout << "Неверные данные! Введите размер массива (не более " << arrayData.MAX\_SIZE << "): ";

cin.clear();

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

}

arrayData.RealArray = new double[arrayData.size]; // выделение памяти для массива

cout << "Введите в одной строке элементы массива, состоящего из " << arrayData.size;

cout << " вещественных чисел, и нажмите <Enter>" << endl;

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

while (!(cin >> arrayData.RealArray[i])) {

cout << "Некорректный ввод! Попробуйте снова: ";

cin.clear();

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

}

}

}

// Функция вывода массива

void printArray(const ArrayData& arrayData) {

cout << "Исходный массив:" << endl;

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

cout.width(4);

cout << arrayData.RealArray[i] << " ";

}

cout << endl;

}

// Функция для определения индекса последнего отрицательного элемента

int findLastNegativeIndex(const ArrayData& arrayData) {

int lastIndex = -1;

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

if (arrayData.RealArray[i] < 0) {

lastIndex = i;

}

}

return lastIndex;

}

// Функция для определения количества отрицательных элементов

int countNegativeElements(const ArrayData& arrayData) {

int count = 0;

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

if (arrayData.RealArray[i] < 0) {

count++;

}

}

return count;

}

// Функция для определения суммы абсолютных значений элементов массива с четными номерами, предшествующих последнему элементу массива с отрицательным значением

double sumEvenElements(const ArrayData& arrayData) {

double sum = 0;

int lastIndex = findLastNegativeIndex(arrayData);

for (int i = 0; i <= lastIndex - 1; i += 2) {

sum += abs(arrayData.RealArray[i]);

}

return sum;

}

// Функция для вычисления произведения суммы абсолютных значений элементов массива с четными номерами, предшествующих последнему элементу массива с отрицательным значением

double calculateProduct(double sum, int count) {

return sum \* count;

}

int main() {

ArrayData arrayData;

int choice;

do {

inputArray(arrayData);

printArray(arrayData);

int lastIndex = findLastNegativeIndex(arrayData);

if (lastIndex == -1) {

cout << "Отрицательных элементов в массиве нет." << endl;

cout << "Результат 0" << endl;

} else {

int count = countNegativeElements(arrayData);

double sum = sumEvenElements(arrayData);

double result = calculateProduct(sum, count);

cout << "Количество отрицательных элементов: " << count << endl;

cout << "Индекс последнего отрицательного элемента: " << lastIndex << endl;

cout << "Сумма: " << sum << endl;

cout << "Результат: " << result << endl;

}

cout << "Повторить - 1, Выход - 2: ";

cin >> choice;

} while (choice == 1);

delete[] arrayData.RealArray; // освобождение памяти, выделенной для массива

return 0;

}

***Скриншоты результатов выполнения программы***

