МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ФГАОУ ВО «Крымский федеральный университет имени В. И. Вернадского»
Физико-технический институт
Кафедра компьютерной инженерии и моделирования

Лабораторная работа №5
по курсу «Структуры и алгоритмы обработки данных»
на тему: «Действия над одномерными массивами в блок-схемах»

Выполнил:
студент 1 курса
группы ПИ-б-о-233(1)
Иващенко Денис Олегович

Зачтено (100, 100, 100).
09.12.2023

Проверила:
старший преподаватель
кафедры компьютерной
инженерии и моделирования
Горская И.Ю.

Симферополь, 2023

Лабораторная работа № 5

Тема: Действия над одномерными массивами в блок-схемах

Цель работы: научиться действовать над одномерными массивами в блок-схемах.

Перед выполнением лабораторной работы:

1. Были изучены теоретические сведения в методических указаниях к выполнению данной лабораторной работы; подробно рассмотрены приведенные практические примеры.

2. Прочитан соответствующий материал в электронном конспекте лектора по данному курсу.

В соответствии с индивидуальным заданием выполнены два задания.

13)Сложить два полинома заданных степеней(коэффициенты хранятся в массивах)

1) Для начала установим библиотеки с которыми нам нужно будет работать(iostream, vector,algorithm)

2) Сделаем функцию addPolynamials(которая принимает константные динамические массивы). Дальше определяется, какие размер будет у массива size(с встроенной функцией std::Max). Дальше создаем вектор result, а дальше идет сложение коэффициентов полиномов с помощью циклов for.Функция возвращает result. Также, вектора начинаются с концов полиномов.

3) Дальше создадим вектора(полиномы), вызываем функцию в векторе result. Дальше выведем результат сложения с помощью цикла for, в котором после выполнения задач в теле i - уменьшается. Все будет действовать до тех пор пока i>=0;

```cpp
#include <iostream>
#include <vector>

// Функция для сложения полиномов
std::vector<int> addPolynomials(const std::vector<int>& poly1, const std::vector<int>& poly2) {
    // Определение размера результирующего полинома
    int size = std::max(poly1.size(), poly2.size());

    // Создание результирующего полинома с начальными значениями 0
    std::vector<int> result(size, 0);

    // Сложение коэффициентов полиномов
    for (int i = 0; i < poly1.size(); i++) {
        result[i] += poly1[i];
    }

    for (int i = 0; i < poly2.size(); i++) {
        result[i] += poly2[i];
    }

    return result;
}

int main() {
    // Первый полином: 4x^3 + 2x^2 + x + 1

    std::vector<int> poly1 = {1, 1, 2, 4};

    // Второй полином: 2x^2 + 2x + 3
    std::vector<int> poly2 = {3, 2, 2};

    // Выполнение сложения полиномов
    std::vector<int> result = addPolynomials(poly1, poly2);

    // Вывод результата
    std::cout << "Результат сложения двух полиномов: ";
    for (int i = result.size() - 1; i >= 0; i--) {
```

```cpp
            if (result[i] != 0) {
                std::cout << result[i] << "x^" << i;
                if (i != 0) {
                    std::cout << " + ";
                }
            }
        }
        std::cout << std::endl;

        return 0;
    }
```

4)Вот ответ (пример, то есть входные данные написаны в комментариях над динамическими массивами <vector>):

```
Результат сложения двух полиномов: 4x^3 + 4x^2 + 3x^1 + 4x^0
```

5) Элемент называется локальным минимумом(максимумом), если у него нет соседа, меньшего (большего), чем он сам. Найти все локальные минимумы и максимумы в заданном массиве из n элементов.

1)Сначала пропишем библиотеки для дальнейшего решения данной задачи(iostream- для работы с вводом и выводом , vector- для работы с динамическим массивом, algorithm)

2) Далее я создаю массив, для нахождения локальных минимумов и максимумов(так будет наглядней и удобно видеть какой ответ будет верный)

3) Я делаю цикл , который будет действовать до тех пор пока размер массива sbrod будет равен i.Далее мы сравниваем каждый элемент со своими соседями при этом учитывая гранитные условия. В данной ситуации нам нужно найти локальный минимум поэтому i-ый элемент массива должен быть меньше своих соседей. После идет вывод элементов, подходящие нам.

4) Такая же ситуация и с локальным максимумом. Только тут должно быть наоборот, чтобы элемент массива должен быть больше своего соседа.

5) результат программы:

```cpp
#include <iostream>
#include <vector>
#include <algorithm>

int main() {
    // массив в котором есть набор чисел
    std::vector<int> sbrod = {3, 1, 4, 1, 5, 9, 2, 6, 5};
// вычисление локального минимума
//создаем цикл
     std::cout<<"Ваш локальный минимум данного массива равен : ";
    for (int i = 0; i < sbrod.size(); i++) {
        //проверяется, является ли текущий элемент sbrod[i] меньше предыдущего элемента sbrod[i - 1]
        //и следующего элемента sbrod[i + 1],
        //при этом учитываются граничные условия. Если условие выполняется, то текущий элемент считается
        //локальным минимумом и выводится на экран с помощью std::cout.
        if ((i == 0 ||   sbrod[i] < sbrod[i - 1]) && (i == sbrod.size() - 1 || sbrod[i] < sbrod[i + 1])) {
            std::cout << sbrod[i]<< " ";
        }
    }
    std::cout<<"Локальный максимум данного массива: ";
     for (int i = 0; i < sbrod.size(); i++) {
        //проверяется, является ли текущий элемент sbrod[i] больше предыдущего элемента sbrod[i - 1]
        //и следующего элемента sbrod[i + 1],
        //при этом учитываются граничные условия. Если условие выполняется, то текущий элемент считается
        //локальным максимумом и выводится на экран с помощью std::cout.
        if ((i == 0 || sbrod[i] > sbrod[i - 1]) && (i == sbrod.size() - 1 || sbrod[i] > sbrod[i + 1])) {
            std::cout << sbrod[i]<< " ";
        }
    }

    return 0;
}
```

17) Ввести одноместный массив из n элементов. Отсортировать массив

```
Ваш локальный минимум данного массива равен : 1 1 2 5 Локальный максимум данного массива: 3 4 9 6

...Program finished with exit code 0
```

по неубыванию(невозрастанию) методом прямого выбора.

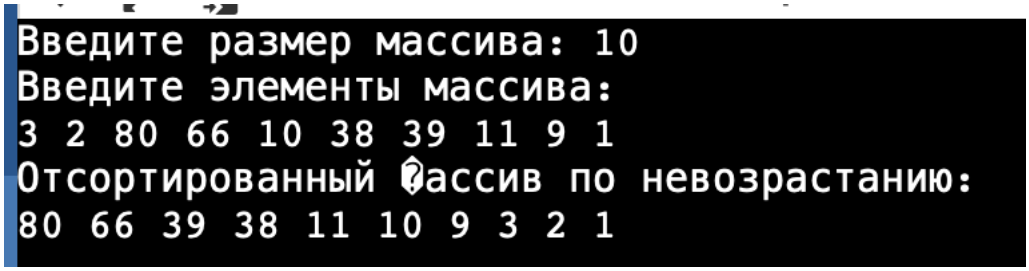
1)Сначала подключаем нужную библиотеку для работы с вводом и выводом.

2) Далее мы заполняем наш массив

3) начинается сортировка методом прямого выбора(находит наименьший ключ и становится первым элементом и так продолжается со всеми элементами)

4) После этого мы делаем цикл, который выводит результат на экран(вывод элементов массива на экран).

5) вот сам результат работы нашей программы :



```cpp
#include <iostream>

int main() {
    //переменная n
    int n;
    //сколько чисел будет вводить пользователь в массив
    std::cout << "Введите размер массива: ";
    std::cin >> n;

    int arr[n];
//цикл, который выполняется до тех пор пока i < n
    //тут мы вписываем числа в массив
    std::cout << "Введите элементы массива:\n";
    for (int i = 0; i < n; i++) {
        std::cin >> arr[i];
    }
 // происходит перебор элементов вектора
    for (int i = 1; i < n; i++) {
        // на какждой иттерации элемент сохраняется в переменной key
        int key = arr[i];
        //j инициализируется как i-1
        int j = i - 1;
//запускается цикл while, который выполняется, пока j больше
        //или равно 0 и элемент sortiv[j] больше key.
        while (j >= 0 && arr[j] < key) {
            // выполняется сдвиг элементов вправо
            arr[j + 1] = arr[j];
            j--;
        }
   // здесь уже встает на свое место
        //key присваивается sortiv[j+1]
        arr[j + 1] = key;
    }

    std::cout << "Отсортированный массив по невозрастанию:\n";
    for (int i = 0; i < n; i++) {
        std::cout << arr[i] << " ";
    }
    std::cout << std::endl;

    return 0;
}
```

Вывод: сегодня на лабораторной работе по предмету «Структуры и алгоритмы обработки данных» я научился действовать **н**ад одномерными массивами в блок-схемах.