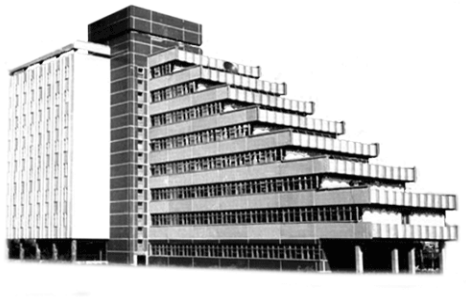
МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Учебный Центр Информационных Технологий «Информатика»



Лабораторная работа №5

по дисциплине «Информатика и программирование I часть»

Направление подготовки: 230105 - «Программное обеспечение вычислительной техники и автоматизированных систем»

Выполнил слушатель: Аветисян Арташес Робертович
Вариант: 6
Дата сдачи: 04.10.2020
Преподаватель: Прытков Д. В.

Новосибирск, 2020г.

1 Задание

Изучение алгоритмов сортировки

Сортировка выбором. Выбирается минимальный элемент в массиве и запоминается. Затем удаляется, а все последующие за ним элементы сдвигаются на один влево. Сам элемент заносится на освободившуюся последнюю позицию.

2 Теоретический материал

Алгоритм сортировки - это алгоритм для упорядочивания элементов в списке. В случае, когда элемент списка имеет несколько полей, поле, служащее критерием порядка, называется ключом сортировки. На практике в качестве ключа часто выступает число, а в остальных полях хранятся какие-либо данные, никак не влияющие на работу алгоритма.

Оценка алгоритма сортировки

Алгоритмы сортировки оцениваются по скорости выполнения и эффективности использования памяти

Время — основной параметр, характеризующий быстродействие алгоритма. Называется также вычислительной сложностью. Для упорядочения важны худшее, среднее и лучшее поведение алгоритма в терминах мощности входного множества A.

Если на вход алгоритму подаётся множество A, то обозначим n = |A|. Для типичного алгоритма хорошее поведение — это O(n log n) и плохое поведение — это O(n2). Идеальное поведение для упорядочения — O(n).

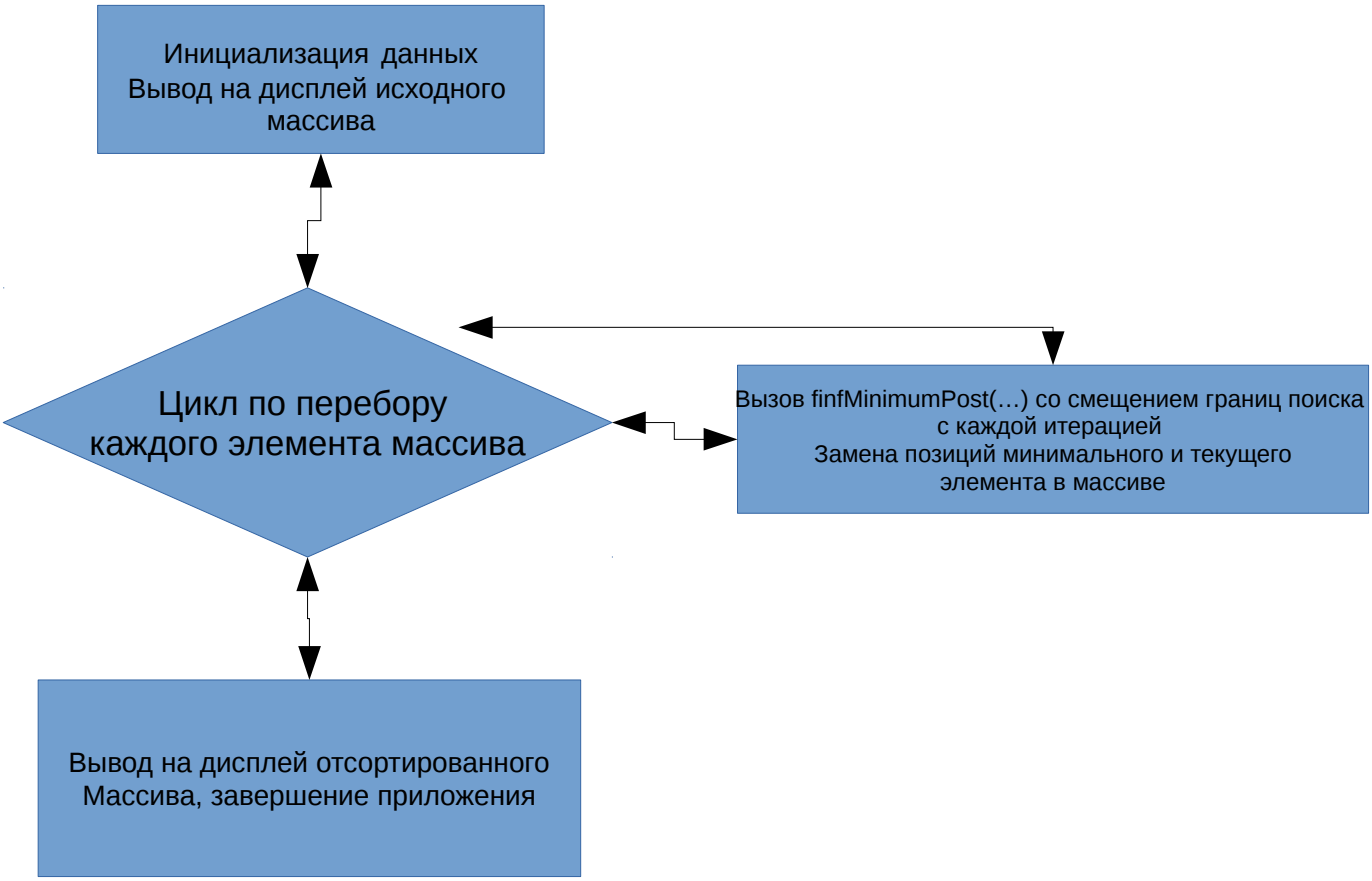
Память — ряд алгоритмов требует выделения дополнительной памяти под временное хранение данных. Как правило, эти алгоритмы требуют O(log n) памяти. При оценке не учитывается место, которое занимает исходный массив и независящие от входной последовательности затраты, например, на хранение кода программы (так как всё это потребляет O(1)). Алгоритмы сортировки, не потребляющие дополнительной памяти, относят к сортировкам на месте.

Сортировка выбором (*Selection sort*) — алгоритм сортировки. Может быть как устойчивый, так и неустойчивый. На массиве из n элементов имеет время выполнения в худшем, среднем и лучшем случае O (n^2), предполагая что сравнения делаются за постоянное время.

Шаги алгоритма:

1. находим номер минимального значения в текущем списке
2. производим обмен этого значения со значением первой неотсортированной позиции (обмен не нужен, если минимальный элемент уже находится на данной позиции)
3. теперь сортируем хвост списка, исключив из рассмотрения уже отсортированные элементы

3 Описание алгоритма приложения



4 Описание реализации

Препроцессорные директивы:

#include - используется для включения в исходный код заголовочных файлов библиотек

Подключаемые заголовочные файлы библиотеки:

- stdio.h
- stdlib.h

Используемые функции:

int main()

Указанная основная функция, являющаяся начальной точкой для выполнения программы. Она обычно управляет выполнением программы, вызывая другие ее функции. Как правило, выполнение программы завершается в конце функции **main**, но по разным причинам это может случиться и в других местах программы. Может быть ипользована для передачи аргументов в приложение.

int findMinimumPos(int* a, int start, int end)

Возвращает позицию (**int**) наименьшего элемента в массива **a**, с возможностью указания позиции начала итерации по массиву (**int start**) и ее завершения (**int end**)

Используемые конструкции и операторы:

if

Оператор выбора **if** позволяет выполнять или опустить выполнять определенных участков кода, в зависимости от того является ли истинным или ложным условие этого оператора. Одно из самых важных назначений оператора выбора **if** так это то, что он позволяет программе совершить действие на выбор, в зависимости от того, например, какие данные ввел пользователь.

for

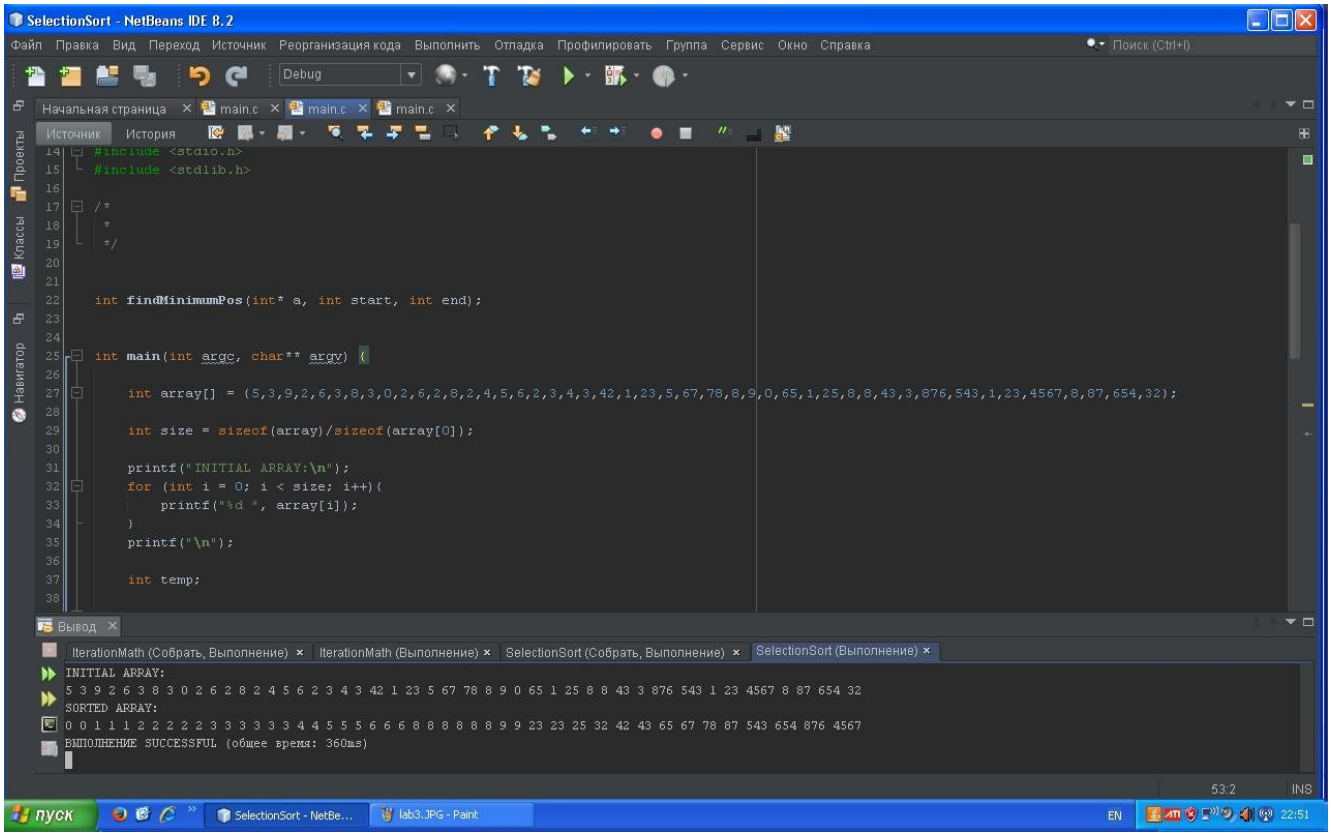
Параметрический цикл (цикл с заданным числом повторений), реализуемый при помощи операций инициализация (присваивание параметру цикла начального значения), проверки условия повторения цикла, модификации - изменения значения параметра для следующего прохождения тела цикла

5 Пример работы программы

На рисунке 1 показана фотография дисплея компьютера с развернутым окном среды разработки NetBeans IDE 8.2 в WindowsXP после выполнения разработанного приложения.

На второй строке выведен массив целых чисел без сортировки, на четвертой строке — отсортированный.

Рисунок 1. Фотография экрана во время выполнения приложения, разработанного согласно задачи лабораторной работы



Выводы

Изучен теоретическим материал по работе с алгоритмами сортировки который применен в практике для решения задачи в рамках данной лабораторной работы.

Успешно разработан и апробирован алгоритм сортировки целых чисел выбором.

Приложение. Текст программы с комментариями

```
/*
 * File:   main.c
 * Author: Artashes
 * lab 5, var - 6
 * Created on 11 сентября 2020 г., 22:50
 */

#include <stdio.h>
#include <stdlib.h>

/*
 *
 */
```

```
int findMinimumPos(int* a, int start, int end);


int main(int argc, char** argv) {

    int array[] =
{5,3,9,2,6,3,8,3,0,2,6,2,8,2,4,5,6,2,3,4,3,42,1,23,5,67,78,8,9,0,65,1,25,8,8,43,3,876,543,1,23,4567,8,87
,654,32};

    int size = sizeof(array)/sizeof(array[0]);

    printf("INITIAL ARRAY:\n");
    for (int i = 0; i < size; i++){
        printf("%d ", array[i]);
    }
    printf("\n");

    int temp;

    for (int i = 0; i < size; i++){
        int minimumPosition = findMinimumPos(array, i, size);
        //printf("minimumPosition: %d\n", minimumPosition);
        temp = array[i];
        array[i] = array[minimumPosition];
        array[minimumPosition] = temp;
    }

    printf("SORTED ARRAY:\n");
    for (int i = 0; i < size; i++){
        printf("%d ", array[i]);
    }

    return (EXIT_SUCCESS);
}

int findMinimumPos(int* a, int start, int end){

    //printf("first element of arrya %d, start %d end%d\n", a[0], start, end);

    int pos = start;
    int minimum = a[start];

    for (int i = start; i < end; i++){
        if ((i+1) != end ){
            if (minimum > a[i+1]){
                pos = i+1;
                minimum = a[i+1];
            }
        }
    }

    return pos;
```

}

Защита: