МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №4
по дисциплине «Алгоритмы и структуры данных»
Тема: Сортировки

Студент(ка) гр. 9382 _____ Русинов Д.А.

Преподаватель _____ Фирсов М.А.

Санкт-Петербург
2020

Цель работы.

Изучить алгоритмы сортировки массива.

Задание.

2. Сортировка простыми вставками; сортировка простыми вставками в список.

Основные теоретические положения.

Сортировка – последовательное расположение или разбиение на группы чего-либо в зависимости от выбранного критерия.

Функции и структуры данных.

В качестве структуры данных был использован двусвязный линейный список.

struct List;

Поля:

1) List* next – указатель на следующий элемент списка
2) List* prev – указатель на предыдущий элемент списка
3) int value – значение узла

Методы:

1) void append(int element) – добавляет элемент в конец списка
2) explicit List(int value) – конструктор
3) ~List() – деструктор

void showList(List* list) – функция для визуализации списка. Принимает сипсок.

List* insertionSort(List* list) – функция для сортировки списка простыми вставками. Возвращает начало на отсортированный старый список.

List* insert(int file = 0) – функция для считывания списка, аргумент – ввод из файла или из консоли. Если 0, то из консоли.

Описание алгоритма.

Общая суть сортировок вставками такова:

1) Перебираются элементы в неотсортированной части массива.
2) Каждый элемент вставляется в отсортированную часть массива на то место, где он должен находиться.

Но рассмотрим алгоритм более подробно на основе списка:

Проходим по списку, начиная с первого элемента. Для каждого рассматриваемого элемента необходимо найти такой элемент, который находится до и при этом меньше либо равный по значению. Сначала рассматриваемый элемент исключается из списка следующим образом:

1) Указатель на следующий элемент предыдущего элемента меняется на указатель на следующий элемент рассматриваемого элемента.
2) Если у рассматриваемого элемента есть следующий элемент, то указатель на предыдущий элемент следующего элемента изменяется на указатель на предыдущий элемент рассматриваемого элемента.

Затем идет сравнение с предыдущими элементами этого элемента. Если удалось найти такой элемент, который меньше либо равен рассматриваемому, то необходимо рассматриваемый вставить после него. Вставка происходит следующим образом:

1) Если у найденного элемента есть следующий элемент, то указатель на предыдущий элемент следующего элемента изменяется на рассматриваемый элемент.
2) Следующим элементом рассматриваемого элемента становится следующий элемент найденного элемента
3) Предыдущим элементом рассматриваемого элемента становится найденный элемент
4) Следующим элементом найденного элемента становится рассматриваемый элемент

Если не удастся найти элемент, удовлетворяющий требованиям, тогда мы выйдем за границу списка. В таком случае необходимо рассматриваемый элемент сделать началом списка. Данное действие выполняется следующим образом:

1) Предыдущим элементом начала списка становится рассматриваемый элемент
2) Предыдущим элементом предыдущего элемента списка становится ничего
3) Следующим элементом предыдущего элемента становится начало списка
4) Началом списка становится рассматриваемый элемент

На примере простых вставок смотрится главное преимущество большинства сортировок вставками, а именно — очень быстрая обработка почти упорядоченных массивов.

Тестирование.

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1.	5 5 3 9 2 6	Ввод из файла? 0 - если нет: 1 [Список до сортировки] 5 3 9 2 6 Начал рассматривать элемент 3 Сравнение 5 > 3 Вышел за границу списка, элемент 3 стал началом списка! [Промежуточный результат] 3 5 9 2 6 Начал рассматривать элемент 9 5 <= 9. Элемент 9 выставляется правее 5. [Промежуточный результат] 3 5 9 2 6 Начал рассматривать элемент 2	

		Сравнение 9 > 2 Сравнение 5 > 2 Сравнение 3 > 2 Вышел за границу списка, элемент 2 стал началом списка! [Промежуточный результат] 2 3 5 9 6 Начал рассматривать элемент 6 Сравнение 9 > 6 5 <= 6. Элемент 6 выставляется правее 5. [Промежуточный результат] 2 3 5 6 9 Прошелся по всему списку, сортировка окончена! [Список после сортировки] 2 3 5 6 9	
2.	10 5 3 9 2 6 0 -1 3 4 1	Ввод из файла? 0 - если нет: 1 [Список до сортировки] 5 3 9 2 6 0 -1 3 4 1 Начал рассматривать элемент 3 Сравнение 5 > 3 Вышел за границу списка, элемент 3 стал началом списка! [Промежуточный результат] 3 5 9 2 6 0 -1 3 4 1 Начал рассматривать элемент 9 5 <= 9. Элемент 9 выставляется правее 5. [Промежуточный результат] 3 5 9 2 6 0 -1 3 4 1 Начал рассматривать элемент 2	

		Сравнение 9 > 2 Сравнение 5 > 2 Сравнение 3 > 2 Вышел за границу списка, элемент 2 стал началом списка! [Промежуточный результат] 2 3 5 9 6 0 -1 3 4 1 Начал рассматривать элемент 6 Сравнение 9 > 6 5 <= 6. Элемент 6 выставляется правее 5. [Промежуточный результат] 2 3 5 6 9 0 -1 3 4 1 Начал рассматривать элемент 0 Сравнение 9 > 0 Сравнение 6 > 0 Сравнение 5 > 0 Сравнение 3 > 0 Сравнение 2 > 0 Вышел за границу списка, элемент 0 стал началом списка! [Промежуточный результат] 0 2 3 5 6 9 -1 3 4 1 Начал рассматривать элемент -1 Сравнение 9 > -1 Сравнение 6 > -1 Сравнение 5 > -1 Сравнение 3 > -1 Сравнение 2 > -1	

		Сравнение 0 > -1 Вышел за границу списка, элемент -1 стал началом списка! [Промежуточный результат] -1 0 2 3 5 6 9 3 4 1 Начал рассматривать элемент 3 Сравнение 9 > 3 Сравнение 6 > 3 Сравнение 5 > 3 3 <= 3. Элемент 3 выставляется правее 3. [Промежуточный результат] -1 0 2 3 3 5 6 9 4 1 Начал рассматривать элемент 4 Сравнение 9 > 4 Сравнение 6 > 4 Сравнение 5 > 4 3 <= 4. Элемент 4 выставляется правее 3. [Промежуточный результат] -1 0 2 3 3 4 5 6 9 1 Начал рассматривать элемент 1 Сравнение 9 > 1 Сравнение 6 > 1 Сравнение 5 > 1 Сравнение 4 > 1 Сравнение 3 > 1 Сравнение 3 > 1 Сравнение 2 > 1	

		0 <= 1. Элемент 1 выставляется правее 0. [Промежуточный результат] -1 0 1 2 3 3 4 5 6 9 Прошелся по всему списку, сортировка окончена! [Список после сортировки] -1 0 1 2 3 3 4 5 6 9 [Промежуточный результат] 0 2 3 5 6 9 -1 3 4 1 Начал рассматривать элемент -1 Сравнил с 9. 9 > -1. Сдвигаю 9 вправо Сравнил с 6. 6 > -1. Сдвигаю 6 вправо Сравнил с 5. 5 > -1. Сдвигаю 5 вправо Сравнил с 3. 3 > -1. Сдвигаю 3 вправо Сравнил с 2. 2 > -1. Сдвигаю 2 вправо Сравнил с 0. 0 > -1. Сдвигаю 0 вправо Вышел за границу списка, поэтому устанавливаю значение -1 в начало списка [Промежуточный результат] -1 0 2 3 5 6 9 3 4 1 Начал рассматривать элемент 3 Сравнил с 9. 9 > 3. Сдвигаю 9 вправо Сравнил с 6. 6 > 3. Сдвигаю 6 вправо Сравнил с 5. 5 > 3. Сдвигаю 5 вправо Дошел до значения 3. 3 <= 3. Поэтому устанавливаю 3 правее. [Промежуточный результат] -1 0 2 3 3 5 6 9 4 1	

		Начал рассматривать элемент 4 Сравнил с 9. 9 > 4. Сдвигаю 9 вправо Сравнил с 6. 6 > 4. Сдвигаю 6 вправо Сравнил с 5. 5 > 4. Сдвигаю 5 вправо Дошел до значения 3. 3 <= 4. Поэтому устанавливаю 4 правее. [Промежуточный результат] -1 0 2 3 3 4 5 6 9 1 Начал рассматривать элемент 1 Сравнил с 9. 9 > 1. Сдвигаю 9 вправо Сравнил с 6. 6 > 1. Сдвигаю 6 вправо Сравнил с 5. 5 > 1. Сдвигаю 5 вправо Сравнил с 4. 4 > 1. Сдвигаю 4 вправо Сравнил с 3. 3 > 1. Сдвигаю 3 вправо Сравнил с 3. 3 > 1. Сдвигаю 3 вправо Сравнил с 2. 2 > 1. Сдвигаю 2 вправо Дошел до значения 0. 0 <= 1. Поэтому устанавливаю 1 правее. [Промежуточный результат] -1 0 1 2 3 3 4 5 6 9 Прошелся по всему списку, сортировка окончена! [Список после сортировки] -1 0 1 2 3 3 4 5 6 9	
3.	1 1	Ввод из файла? 0 - если нет: 1 [Список до сортировки] 1 Прошелся по всему списку, сортировка окончена! [Список после сортировки] 1	

4.	-1	Ввод из файла? 0 - если нет: 0 Введите кол-во чисел: -1 Кол-во чисел должно быть больше 0	Проверка на некорректных данных
5.	5 1 1 1 1 1	Ввод из файла? 0 - если нет: 1 [Список до сортировки] 1 1 1 1 1 Начал рассматривать элемент 1 1 <= 1. Элемент 1 выставляется правее 1. [Промежуточный результат] 1 1 1 1 1 Начал рассматривать элемент 1 1 <= 1. Элемент 1 выставляется правее 1. [Промежуточный результат] 1 1 1 1 1 Начал рассматривать элемент 1 1 <= 1. Элемент 1 выставляется правее 1. [Промежуточный результат] 1 1 1 1 1 Начал рассматривать элемент 1 1 <= 1. Элемент 1 выставляется правее 1. [Промежуточный результат] 1 1 1 1 1 Прошелся по всему списку, сортировка окончена! [Список после сортировки] 1 1 1 1 1	
6.	5 1 -1 1 -1 1	Ввод из файла? 0 - если нет: 1 [Список до сортировки] 1 -1 1 -1 1 Начал рассматривать элемент -1 Сравнение 1 > -1	

		Вышел за границу списка, элемент -1 стал началом списка! [Промежуточный результат] -1 1 1 -1 1 Начал рассматривать элемент 1 1 <= 1. Элемент 1 выставляется правее 1. [Промежуточный результат] -1 1 1 -1 1 Начал рассматривать элемент -1 Сравнение 1 > -1 Сравнение 1 > -1 -1 <= -1. Элемент -1 выставляется правее -1. [Промежуточный результат] -1 -1 1 1 1 Начал рассматривать элемент 1 1 <= 1. Элемент 1 выставляется правее 1. [Промежуточный результат] -1 -1 1 1 1 Прошелся по всему списку, сортировка окончена! [Список после сортировки] -1 -1 1 1 1	

Выводы.

Был изучен алгоритм сортировки простыми вставками в списке, была создана программа, которая сортирует список данным алгоритмом.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.cpp

```
#include <iostream>
#include "fstream"


struct List { // односвязный линейный список
    List* next = nullptr;
    List* prev = nullptr;
    int value;

    void append(int element) { // добавить элемент в конец
        List* current = this;
        while(current->next) current = current->next;
        current->next = new List(element);
        current->next->prev = current;
    }

    explicit List(int value) : value(value) {} // конструктор

    ~List() { // деструктор
        delete next;
    }
};

// функция для вывода списка
void showList(List* list) {
    while (list) {
        std::cout << list->value << " ";
        list = list->next;
    }
}

List* insertionSort(List* list) {
    // начинать сортировку можно с первого элемента списка
    for (List* i = list->next; i;) {
        // начинаем сравнивать предыдущий элемент за рассматриваемым эле-
ментом
        List* j = i->prev;
        // запоминаем значение рассматриваемого элемента
        std::cout << "Начал рассматривать элемент " << i->value <<
std::endl;

        // доходим до элемента, который будет <= рассматриваемого
        while (j && j->value > i->value) {
            std::cout << "Сравнение " << j->value << " > " << i->value <<
std::endl;
            j = j->prev;
        }

        // запоминаем следующий рассматриваемый элемент
        List* next = i->next;

        // выдергиваем из списка текущий рассматриваемый элемент
```

```
        i->prev->next = i->next;
        if (i->next) i->next->prev = i->prev;

        if (j) {
            // если попали сюда, то рассматриваемый элемент устанавливаем
правее найденного
            // найденный <= рассматриваемого
            std::cout << j->value << " <= " << i->value << ". Элемент "
<< i->value << " выставляется правее " << j->value << "." << std::endl;
            if (j->next) j->next->prev = i;
            i->next = j->next;
            i->prev = j;
            j->next = i;
        } else {
            // если попали сюда, то рассматриваемый элемент станет нача-
лом списка
            // мы не нашли элементов, которые <= рассматриваемого
            std::cout << "Вышел за границу списка, элемент " << i->value
<< " стал началом списка!" << std::endl;
            list->prev = i;
            list->prev->prev = nullptr;
            list->prev->next = list;
            list = i;
        }

        // устанавливаем следующий рассматриваемый элемент
        i = next;

        std::cout << "[Промежуточный результат] ";
        showList(list);
        std::cout << std::endl;

    }
    std::cout << "Прошелся по всему списку, сортировка окончена!" <<
std::endl;
    return list;
}

// ввод из консоли или из файла
List* insert(int file = 0) {
    int N;
    List* source = nullptr;

    if (!file) {
        std::cout << "Введите кол-во чисел: ";
        std::cin >> N; // получаем кол-во элементов
        if (N <= 0) {
            std::cout << "Кол-во чисел должно быть больше 0";
            return source;
        }

        for (int i = 0; i < N; ++i) { // считываем N элементов
            int k;
            std::cout << "[" << i+1 << "] " << "Введите число: ";
            std::cin >> k;

            if (!source) source = new List(k);
            else source->append(k); // Добавляем в список
```

```
        }
    } else {
        std::ifstream input("file.txt"); // открываем файл
        if (!input.is_open()) { // проверяем на доступность
            input.close();
            std::cout << "Не удалось открыть файл file.txt" << std::endl;
            return source;
        }
        else {
            if(!(input >> N)) { // считываем N
                std::cout << "Не удалось считать кол-во элементов" <<
std::endl;
                return nullptr;
            }

            if (N <= 0) {
                std::cout << "Кол-во чисел должно быть больше 0";
                return source;
            }

            for (int i = 0; i < N; ++i) { // Считываем N элементов
                int k;
                if(!(input >> k)) { // проверяем, получилось ли считать
                    std::cout << "Задано N чисел, но было введено меньше
N чисел" << std::endl;
                    delete source;
                    return nullptr; // если не удалось, вызываем деструк-
тор списка
                }
                if (!source) source = new List(k); // инициализация
списка
                else source->append(k); // добавляем элемент
            }
        }
    }
    return source;
}

int main() {
    int file;
    std::cout << "Ввод из файла? 0 - если нет: ";
    std::cin >> file;
    List* source = insert(file); // считывание списка
    if (!source) return 0;

    std::cout << "[Список до сортировки] ";
    showList(source);
    std::cout << std::endl;
    List* sorted = insertionSort(source);
    std::cout << "[Список после сортировки] ";
    showList(sorted);
    delete sorted;
    return 0;
}
```