**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**

Федеральное государственное автономное образовательное учреждение высшего профессионального образования

**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

Инженерная школа информационных технологий и робототехники

Направление 15.03.04 «Автоматизация технологических процессов и производств»

Отделение автоматизации и робототехники

Рассмотрение различных методов сортировки одномерных массивов

Наименование лабораторной работы

Отчет по лабораторной работе № 2

по дисциплине «Основы программирования и алгоритмизации»

Наименование учебной дисциплины

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Выполнил студент гр. 8Т32 |  | C:\Users\pavlu\OneDrive\Рабочий стол\Подпись.jpg |  |  |  | П. П. Гаврильев |
|  |  | Подпись |  | Дата |  | И.О. Фамилия |
|  |  |  |  |  |  |  |
| Проверил ассистент ОАР |  |  |  |  |  | А. В. Павловский |
| Должность |  | Подпись |  | Дата |  | И.О. Фамилия |

Томск – 2024 г.

**Цель работы:**

Ознакомиться и реализовать с различными методами сортировки на практике на языке программирования С++.

**Ход работы:**

Таблица 1 – Формулировка заданий

|  |  |
| --- | --- |
| **№** | Формулировка |
| **1** | Сортировка пузырьком |
| **2** | Сортировка вставкой |
| **3** | Сортировка слиянием |

Задание 1.

В данном задании поставлена задача реализовать сортировку массива методом пузырьковой сортировки. Для реализации этой задачи были спроектированы блок схемы, представленные на рисунках 1.1 и 1.2. По данным блок схемам были написаны программа, которая представлены в листингах 1.1 и 1.2. Результат выполнения программы представлен на рисунке 1.3.

Программа запрашивает размер массива, после которого вводятся элементы массива. Для сортировки данного массива, была реализована функция sortbubble. После вызова функции на консоль пользователя выводится отсортированный массив.

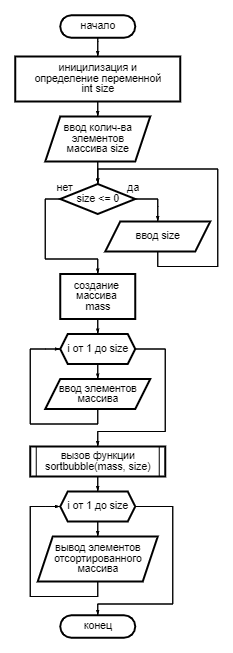
****

Рисунок 1.1 – Блок схема программы

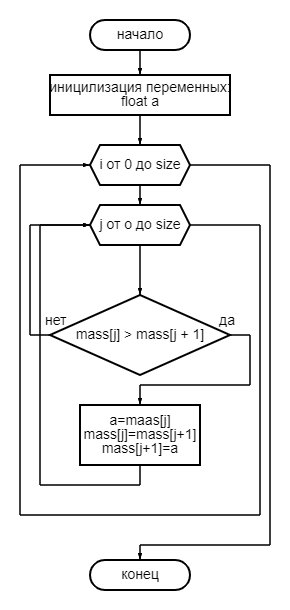


Рисунок 1.2 – Блок схема функции sortbubble(mass, size)

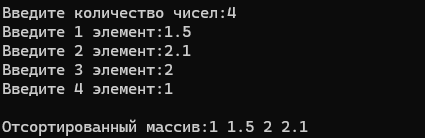


Рисунок 1.3 – Итог запуска программы

Листинг 1.1 – Код программы

|  |
| --- |
| #include <iostream>  #include "sortbubble.h"  int main()  {  setlocale(LC\_ALL, "ru");  int size;  std::cout << "Введите количество чисел:";  std::cin >> size;  while (size <= 0) {  std::cout << "Кол-во чисел не может быть отрицательным или равным нулю. Введите еще раз::";  std::cin >> size;  }  float\* mass = new float[size];  for (int i = 0; i < size; i++) {  std::cout << "Введите " << i + 1 << " элемент:";  std::cin>> mass[i];  }  mergesort(mass, size);  std::cout << "\nОтсортированный массив:";  for (int i = 0; i < size; i++) {  std::cout << mass[i] << " ";  }  delete[] mass;  } |

Листинг 1.2 - Библиотека sortbubble(mass, size)

|  |
| --- |
| #pragma once  void sortbubble(float\* mass, int size)  {  float a;  for (int i = 0; i < size; i++) {  for (int j = 0; j < size - 1; j++) {  if (mass[j] > mass[j + 1]) {  a = mass[j];  mass[j] = mass[j + 1];  mass[j + 1] = a;  }  }  }  } |

Задание 2.

В данном задании поставлена задача реализовать сортировку массива методом вставкой сортировки. Для реализации этой задачи были спроектированы блок схемы, представленные на рисунках 2.1 и 2.2. По данным блок схемам были написаны программа, которая представлены в листингах 2.1 и 2.2. Результат выполнения программы представлен на рисунке 2.3.

Программа запрашивает размер массива, после которого вводятся элементы массива. Для сортировки данного массива, была реализована функция insertionsort. После вызова функции на консоль пользователя выводится отсортированный массив.

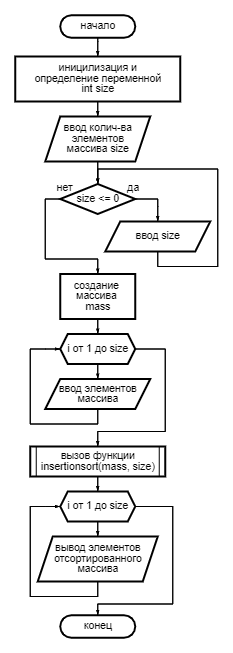


Рисунок 2.1 – Блок схема программы

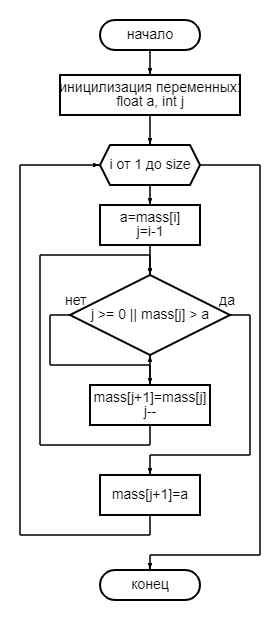


Рисунок 2.2-Блок схема функции insertionsort(mass, size)

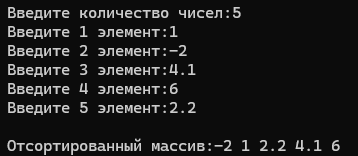


Рисунок 2.3 – Итог запуска программы

Листинг 2.1 – Код программы

|  |
| --- |
| #include <iostream>  #include "insertionsort.h"  int main()  {  setlocale(LC\_ALL, "ru");  int size;  std::cout << "Введите количество чисел:";  std::cin >> size;  while (size <= 0) {  std::cout << "Кол-во чисел не может быть отрицательным или равным нулю. Введите еще раз::";  std::cin >> size;  }  float\* mass = new float[size];  for (int i = 0; i < size; i++) {  std::cout << "Введите " << i + 1 << " элемент:";  std::cin>> mass[i];  }  insertionsort(mass, size);  std::cout << "\nОтсортированный массив:";  for (int i = size-1; i >=0; i--) {  std::cout << mass[i]<<" ";  }  delete[] mass;  } |

Листинг 2.2-библиотека insertionsort

|  |
| --- |
| #pragma once  #include "insertionsort.h"  void insertionsort(float\* mass, int size)  {  float a;  int j;  for (int i = 1; i < size; i++) {  a = mass[i];  j = i - 1;  while (j >= 0 || mass[j] > a) {  mass[j + 1] = mass[j];  j--;  }  mass[j + 1] = a;  }  } |

Задание 3.

В данном задании поставлена задача реализовать сортировку массива методом сортировки слиянием. Для реализации этой задачи были спроектированы блок схемы, представленные на рисунках: 3.1, 3.2, 3.3. По данным блок схемам были написаны программа, которая представлена в листингах 3.1 и 3.2. Результат выполнения программы представлен на рисунке 3.4.

Программа запрашивает размер массива, после которого вводятся элементы массива. Для сортировки данного массива, была реализована функция mergesort. После вызова функции на консоль пользователя выводится отсортированный массив.

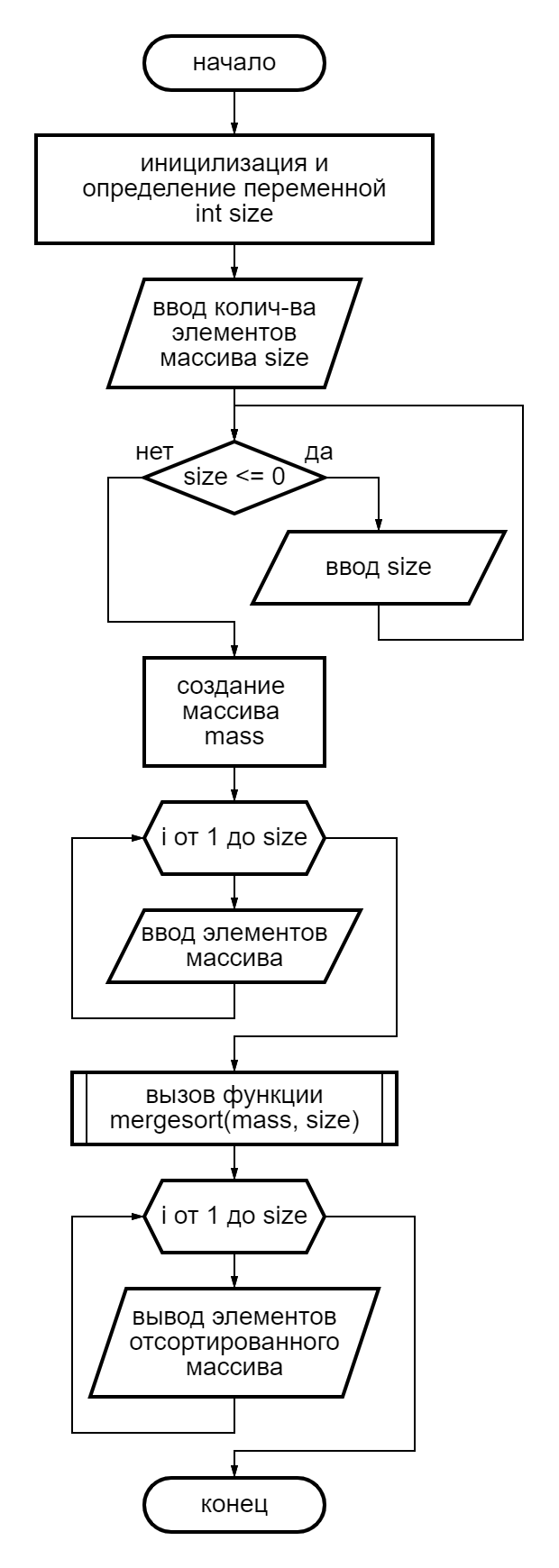


Рисунок 3.1 – Блок схема программы

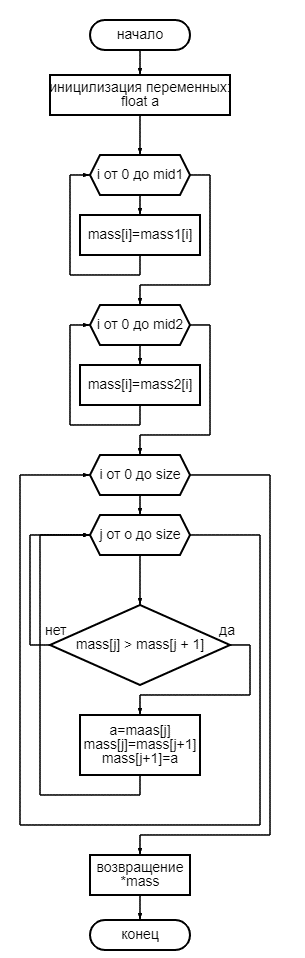


Рисунок 3.2 – Блок схема функции merge (mass, mass1, mass2, mid1, mid2, size)

|  |  |
| --- | --- |
|  |  |
| a) | б) |

Рисунок 3.3 – Блок схема функции mergesort(mass, size)

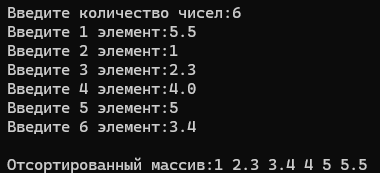


Рисунок 3.4 – Итог запуска программы

Листинг 3.1 – Код программы 3

|  |
| --- |
| #include <iostream>  #include "mergesort.h"  int main()  {  setlocale(LC\_ALL, "ru");  int size;  std::cout << "Введите количество чисел:";  std::cin >> size;  while (size <= 0) {  std::cout << "Кол-во чисел не может быть отрицательным или равным нулю. Введите еще раз::";  std::cin >> size;  }  float\* mass = new float[size];  for (int i = 0; i < size; i++) {  std::cout << "Введите " << i + 1 << " элемент:";  std::cin>> mass[i];  }    mergesort(mass, size);  std::cout << "\nОтсортированный массив:";  for (int i = 0; i < size; i++) {  std::cout << mass[i] << " ";  }  delete[] mass;  } |

Листинг 3.2 – библиотека mergesort

|  |
| --- |
| #pragma once  #include "mergesort.h"  void merge(float\* mass, float\* mass1, float\* mass2, int mid1, int mid2, int size) {  float a;  for (int i = 0; i < mid1; i++) {  mass[i] = mass1[i];  }  for (int i = 0; i < mid2; i++) {  mass[i+mid1] = mass2[i];  }  for (int i = 0; i < size; i++) {  for (int j = 0; j < size - 1; j++) {  if (mass[j] > mass[j + 1]) {  a = mass[j];  mass[j] = mass[j + 1];  mass[j + 1] = a;  }  } |

Продолжение листинга 3.2

|  |
| --- |
| }  }  void mergesort(float\* mass, int size){  int mid1 = size / 2;  int mid2 = size - mid1;  if (size > 2) {  float\* mass1 = new float[mid1];  float\* mass2 = new float[mid2];  for (int i = 0; i < mid1; i++) {  mass1[i] = mass[i];  }  for (int i = 0; i < mid2; i++) {  mass2[i] = mass[i + mid1];  }  mergesort(mass1, mid1);  mergesort(mass2, mid2);  merge(mass, mass1, mass2, mid1, mid2, size);  delete[] mass1;  delete[] mass2;  }  if (size == 2) {  if (mass[0] > mass[1]) {  float a = mass[0];  mass[0] = mass[1];  mass[1] = a;  }  }  } |

**Вывод**

В ходе лабораторной работы была изучены несколько видов сортировки. На основе этой информации была проделана работа по написанию кода, представленных на листингах (1.1, 1.2, 2.1, 2.2, 3.1, 3.2), для заданий. Также были сделаны блок схемы, которые показаны на рисунках (1.1, 1.2, 2.1, 2.2, 3.1, 3.2, 3.3). В итоге благодаря этой информации можно реализовать различные виды сортировок, и с помощью этих сортировок можно работать с большим количеством чисел.