МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

ФАКУЛЬТЕТ ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

Утвержден на заседании кафедры

«Вычислительная техника»

"\_\_\_" \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 20 г.

Заведующий кафедрой

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ М.А. Митрохин

**ОТЧЕТ ПО УЧЕБНОЙ (ОЗНАКОМИТЕЛЬНОЙ) ПРАКТИКЕ**

(2022/2023 учебный год)

Андреянов Ярослав Ильич

Направление подготовки 09.03.01 «Информатика и вычислительная техника»

Наименование профиля подготовки «Программное обеспечение средств вычислительной техники и автоматизированных систем»

Форма обучения – очная Срок обучения в соответствии с ФГОС – 4 года

Год обучения 1 семестр 2

Период прохождения практики с 29.06.2023 по 12.07.2023

Кафедра «Вычислительная техника»

Заведующий кафедрой д.т.н., профессор, Митрохин М.А.

*(должность, ученая степень, ученое звание, Ф.И.О.)*

Руководитель практики д.т.н., профессор, Зинкин С.А.

*(должность, ученая степень, ученое звание, Ф.И.О.)*

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

ФАКУЛЬТЕТ ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

Утвержден на заседании кафедры

«Вычислительная техника»

"\_\_\_" \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 20 г.

Заведующий кафедрой

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ М.А. Митрохин

**ИНДИВИДУАЛЬНЫЙ ПЛАН ПРОХОЖДЕНИЯ УЧЕБНОЙ (ОЗНАКОМИТЕЛЬНОЙ) ПРАКТИКИ**

(2022/2023 учебный год)

Андреянов Ярослав Ильич

Направление подготовки 09.03.01 «Информатика и вычислительная техника»

Наименование профиля подготовки «Программное обеспечение средств вычислительной техники и автоматизированных систем»

Форма обучения – очная Срок обучения в соответствии с ФГОС – 4 года

Год обучения 1 семестр 2

Период прохождения практики с 29.06.2023 по 12.07.2023

Кафедра «Вычислительная техника»

Заведующий кафедрой д.т.н., профессор, Митрохин М.А.

*(должность, ученая степень, ученое звание, Ф.И.О.)*

Руководитель практики д.т.н., профессор, Зинкин С.А.

*(должность, ученая степень, ученое звание, Ф.И.О.)*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № п/п | Планируемая форма работы во время практики | Количество часов | Календарные сроки проведения работы | Подпись  руководителя  практики от вуза |
| 1 | Выбор темы и разработка индивидуального плана проведения работ | 2 | 29.06.2023 –  29.06.2023 |  |
| 2 | Подбор и изучение материала по теме работы | 15 | 30.06.2023 –  02.07.23 |  |
| 3 | Разработка алгоритма | 43 | 02.07.23 –  06.07.23 |  |
| 4 | Описание алгоритма и программы | 18 | 6.07.23 –  08.07.23 |  |
| 5 | Тестирование | 5 | 08.07.23 –  08.07.23 |  |
| 6 | Получение и анализ результатов | 10 | 08.07.23 –  10.07.23 |  |
| 7 | Оформление отчёта | 15 | 10.07.23 –  12.07.2023 |  |
|  | **Общий объём часов** | 108 |  |  |

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

ФАКУЛЬТЕТ ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

**ОТЧЁТ**

**О ПРОХОЖДЕНИИ УЧЕБНОЙ (ОЗНАКОМИТЕЛЬНОЙ) ПРАКТИКИ**

(2022/2023 учебный год)

Андреянов Ярослав Ильич

Направление подготовки 09.03.01 «Информатика и вычислительная техника»

Наименование профиля подготовки «Программное обеспечение средств вычислительной техники и автоматизированных систем»

Форма обучения – очная Срок обучения в соответствии с ФГОС – 4 года

Год обучения 1 семестр 2

Период прохождения практики с 29.06.2023 по 12.07.2023

Кафедра «Вычислительная техника»

Андреянов Я. И. выполнил практическое задание «Сортировка выбором». В том числе: осуществил работу с файлами, протестировал и отладил программу, оформил отчёт. На первоначальном этапе был изучен и проанализирован алгоритм, выбран метод решения и язык программирования C++, на котором была написана программа.

Бакалавр Андреянов Я. И. "\_\_\_" \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2023г.

*(должность, Ф.И.О., подпись, дата)*

Руководитель практики Зинкин С. А. "\_\_\_" \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2023г.

*(должность, Ф.И.О., подпись, дата)*

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

ФАКУЛЬТЕТ ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

**ОТЗЫВ**

**О ПРОХОЖДЕНИИ УЧЕБНОЙ (ОЗНАКОМИТЕЛЬНОЙ) ПРАКТИКИ**

(2022/2023 учебный год)

Андреянов Ярослав Ильич

Направление подготовки 09.03.01 «Информатика и вычислительная техника»

Наименование профиля подготовки «Программное обеспечение средств вычислительной техники и автоматизированных систем»

Форма обучения – очная Срок обучения в соответствии с ФГОС – 4 года

Год обучения 1 семестр 2

Период прохождения практики с 29.06.2023 по 12.07.2023

Кафедра «Вычислительная техника»

Во время выполнения работы Андреянов Я. И. показал себя ответственным, добросовестным студентом, знающим свой предмет, имеющим представление о современном состоянии науки, владеющим современными общенаучными знаниями по информатике и вычислительной технике, программированию и сортировке.

За выполнение работы Андреянов Я. И. заслуживает оценки «\_\_\_\_\_\_».

Руководитель практики д.т.н., профессор, Зинкин С.А.

*(должность, ученая степень, ученое звание, Ф.И.О., подпись)*

"\_\_\_" \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2023г.

Содержание:

Введение...............................................................................................................7

1. Постановка задачи...........................................................................................8

1.1. Достоинства алгоритма......................................................................8

1.2. Недостатки алгоритма........................................................................9

1.3. Типичные сценарии применения......................................................9

2. Выбор решения................................................................................................9

3. Описание программы....................................................................................12

4. Схемы программы.........................................................................................18

4.1. Блок-схема программы....................................................................19

4.2. Блок-схема алгоритма......................................................................23

5. Тестирование программы.............................................................................26

5.1. Тестирование на разных наборах данных......................................26

5.2. Анализ полученных результатов тестирования (анализ работы алгоритма)................................................................................................28

6. Отладка...........................................................................................................28

7. Совместная разработка.................................................................................30

Заключение.........................................................................................................31

Список используемой литературы...................................................................32

Приложение А....................................................................................................33

Приложение Б Листинг.....................................................................................37

Введение

Данные являются важной частью современной жизни. И очень часто возникает необходимость упорядочить или сгруппировать их по определенному критерию.

Сортировка способна помочь работе бухгалтера в оформлении сводных таблиц, подведению итогов учебного года, созданию рейтингов, формированию предпочтений пользователя на определенных сервисах и т.д.

Первые прототипы современных методов сортировки появились еще в XIX веке, это значит, что уже тогда возникала проблема упорядочения данных. Сегодня же при непрерывном росте вычислительных мощностей и увеличении хранимых массивов данных еще острее стоит проблема сортировки информации пользователя и оператора ЭВМ. Поэтому появилось множество алгоритмов сортировки. Одни из них являются более эффективными по времени, другие – проще для понимания и реализации. Достаточно простой является сортировка выбором.

Сортировка выбором — это алгоритм сортировки массивов, в котором на каждой итерации во всей последовательности неотсортированных данных выбирается минимальный элемент (при сортировке по возрастанию) и помещается в первую позицию неотсортированной последовательности. Тем самым готовая (отсортированная) последовательность увеличивается на один элемент, а исходная (неотсортированная) последовательность на один элемент уменьшается.

Целью прохождения практики является осуществление алгоритма сортировки методом выбора с использованием псевдографического интерфейса.

**1. Постановка задачи**

Традиционно стоит начать изложение решений задачи с ее постановки. Обычно задача сортировки предполагает упорядочивание некоторого массива целых чисел по возрастанию. Но на самом деле, это является некоторым упрощением. Излагаемые в этом разделе алгоритмы можно применять для упорядочивания массива любых объектов, между которыми установлено отношение порядка (то есть про любые два элемента можно сказать: первый больше второго, второй больше первого или они равны). Упорядочивать можно как по возрастанию, так и по убыванию. Мы же воспользуемся двухсторонней сортировкой выбора.

Двухсторонняя сортировка выбором осуществляется следующим образом: массив делится на две части. Одна из частей называется отсортированной, а другая неотсортированной. Алгоритм предполагает проход по всему массиву с тем, чтобы длина отсортированной части стала равна длине всего массива. В рамках каждой итерации мы находим минимум и максимум в неотсортированной части массива и меняем местами этот минимум с первым элементом неотсортированной части массива, а максимум с последним. После чего мы увеличиваем длину отсортированной части массива на две элемента.

1.1. Достоинства алгоритма

Алгоритм прост в реализации и не требует большого количества памяти.

Алгоритм может быть использован для сортировки массивов любого размера и типа данных, включая строки и числа с плавающей точкой.

Алгоритм легко адаптируется для сортировки по возрастанию или убыванию, в зависимости от того, какие данные нужно отсортировать.

Алгоритм не требует знания порядка данных в массиве, что позволяет ему работать с массивами, которые были отсортированы в другом порядке.

Алгоритм является одним из самых простых алгоритмов сортировки, что делает его идеальным выбором для начинающих программистов.

1.2. Недостатки алгоритма

Сортировка выбором может быть неэффективной для массивов с большим количеством элементов, так как каждый элемент должен быть выбран и сравнен с другими элементами.

Сортировка выбором требует больше операций сравнения, чем другие алгоритмы сортировки, такие как пузырьковая сортировка или сортировка вставками.

Сортировка выбором требует больше операций сравнения, чем другие алгоритмы сортировки, такие как сортировка вставками.

Сортировка выбором является менее эффективной, чем сортировка слиянием или быстрая сортировка, когда речь идет о сортировке массивов, содержащих повторяющиеся элементы.

1.3. Типичные сценарии применения

Сортировка выбором может быть использована для быстрой сортировки маленьких массивов.

Сортировка выбором может быть использована для сортировки массивов чисел в порядке возрастания или убывания.

Сортировка выбора может быть использована в задачах, связанных с сортировкой строк, например, для сортировки строк по длине или по алфавиту.

**2. Выбор решения**

Microsoft Visual Studio — это программная среда по разработке приложений для ОС Windows, как консольных, так и с графическим интерфейсом.

Функциональная структура среды включает в себя:

1. Редактор исходного кода с поддержкой технологии IntelliSense и возможностью простейшего рефакторинга кода;

2. Отладчик кода;

3. Редактор форм, предназначенный для упрощённого конструирования графических интерфейсов;

4. Веб-редактор;

5. Дизайнер классов;

Visual Studio также позволяет создавать и подключать сторонние дополнения (плагины) для расширения функциональности практически на каждом уровне, включая добавление поддержки систем контроля версий исходного кода, добавление новых наборов инструментов (для редактирования и визуального проектирования кода на предметно-ориентированных языках программирования или инструментов для прочих аспектов процесса разработки программного обеспечения).

В комплект входят следующие основные компоненты:

1. Visual Basic.NET – для разработки приложений на VisualBasic;

2. Visual C++ – на традиционном языке C++;

3. Visual C# – на языке C# (Microsoft);

4. Visual F# – на F# (Microsoft Developer Division).

Интегрированная среда разработки (IntegratedDevelopmentEnvironment – IDE) Visual Studio предлагает ряд высокоуровневых функциональных возможностей, которые выходят за рамки базового управления кодом.

Ниже перечислены основные преимущества IDE-среды Visual Studio:

1. Встроенный Git;

2. Поддержка множества языков при разработке;

3. Интуитивный стиль кодирования;

4. Более высокая скорость разработки;

5. Возможности отладки.

Язык программирования С++ представляет высокоуровневый компилируемый язык программирования общего назначения со статической типизацией, который подходит для создания самых различных приложений. На сегодняшний день С++ является одним из самых популярных и распространенных языков.

Своими корнями он уходит в язык Си, который был разработан в 1969—1973 годах в компании Bell Labs программистом Деннисом Ритчи. В начале 1980-х годов датский программист Бьерн Страуструп, который в то время работал в компании Bell Labs, разработал С++ как расширение к языку Си. Фактически вначале C++ просто дополнял язык Си некоторыми возможностями объектно-ориентированного программирования. И поэтому сам Страуструп вначале называл его как "C with classes" ("Си с классами").

С++ является мощным языком, унаследовав от Си богатые возможности по работе с памятью. Поэтому нередко С++ находит свое применение в системном программировании, в частности, при создании операционных систем, драйверов, различных утилит, антивирусов и т.д. К слову сказать, ОС Windows большей частью написана на С++. Но только системным программированием применение данного языка не ограничивается. С++ можно использовать в программах любого уровня, где важны скорость работы и производительность. Нередко он применяется для создания графических приложений, различных прикладных программ. Также особенно часто его используют для создания игр с богатой насыщенной визуализацией. Кроме того, в последнее время набирает ход мобильное направление, где С++ тоже нашел свое применение. И даже в веб-разработке также можно использовать С++ для создания веб-приложений или каких-то вспомогательных сервисов, которые обслуживают веб-приложения. В общем С++ – язык широкого пользования, на котором можно создавать практически любые виды программ.

**3. Описание программы**

В программе для быстрой сортировки подключены следующие заголовочные файлы: *iostream* – заголовочный файл с классами, функциями и переменными для организации ввода-вывода; *fstream* – заголовочный файл для работы с файлами; *vector* - заголовочный файл с классами, функциями и переменными для организации массива с неизвестном количеством элементов; *stdlib.h* – для организации получения случайных чисел; *time.h*, *ctime* – заголовочный файл стандартной библиотеки C++, содержащий типы и функции для работы с датой и временем; *Windows.h* – специфичный заголовочный файл, необходимый для использования в программе функционала, предоставляемого операционной системой Windows; *using namespace std* – использование стандартного пространства имен.

#include <iostream>

#include <fstream>

#include <vector>

#include <stdlib.h>

#include <time.h>

#include <ctime>

#include <windows.h>

using namespace std;

В функции menu() представлены варианты взаимодействия пользователя с программой: выход из программы, заполнение массива, его сортировка с помощью методов однонаправленного или двунаправленного выбора, вывод массива на экран и запись массива в файл.

void menu(){

cout << "0. выйти из программы" << endl;

cout << "1. заполнить массив" << endl;

cout << "2. считать массив" << endl;

cout << "3. отсортировать массив с помощью однонаправленного алгоритма выбора (selection sort)" << endl;

cout << "4. отсортировать массив с помощью двунаправленного алгоритма выбора (selection sort)" << endl;

cout << "5. вывести массив на экран" << endl;

cout << "6. записать массив в файл" << endl;

cout << "выберите действие: ";}

Функция main() реализует алгоритм выбора действия пользователя. При выборе выхода из программы выводится сообщение "До свидания!" и прекращает свою работу. Если выбрано действие заполнения массива вызывается функция fill\_out() после её выполнения выводится сообщение "массив заполнен". Если выбрать действие считать массив, то он загрузится из файла и будет выведено сообщение "массив заполнен". Если было получено значение три, будет выполнено действие сортировки массива с помощью одностороннего алгоритма выборам путём использования функции unidirectional\_sort(). Если было получено значение три, выполняется сортировка массива с помощью двухсторонней сортировки выбором путём выбора функции bidirectional\_sort(). При выборе действия вывода массива на экран используется функция print() или её второй прототип. В первом случае будет выведен отсортированный массив, во втором не отсортированный. Если было введено иное значение, то в консоль будет выведена ошибка "Введено неверное число" и повторно будет вызвана функция меню. При выборе функции записать массив в файл, массив будет сохранён.

int main()

{

//------------------блок переменных-----------------

unsigned int num, search\_time = NULL;

vector <int> mas;

//-------------------блок настроек------------------

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

srand(time(NULL));

//---------------------блок кода--------------------

do {

menu();

cin >> num;

switch (num)

{

case 0:

cout << "До свидания!";

break;

case 1:

fill\_out(N);

cout << "массив заполнен" << endl;

system("pause");

break;

case 2:

read(mas, N);

cout << "массив считан" << endl;

system("pause");

break;

case 3:

search\_time = unidirectional\_sort(mas);

cout << "массив отсортирован" << endl;

system("pause");

break;

case 4:

search\_time = bidirectional\_sort(mas);

cout << "массив отсортирован" << endl;

system("pause");

break;

case 5:

if (search\_time == NULL)

{

print(mas);

}

else print(mas, search\_time);

system("pause");

break;

case 6:

write(mas);

cout << "массив записан" << endl;

system("pause");

break;

default:

cout << " Введено неверное число" << endl;

}

system("CLS");

} while (num);

return 0;

}

Функция fill\_out() служит для заполнения массива случайными целыми числами в диапазоне от -100 до 101.

void fill\_out(int N) // заполнение массива

{

ofstream fout("out.txt");

for (int i = 0; i < N; i++)

{

fout << (-100 + rand() % (101 - (-100) + 1)) << " ";

}

fout.close();

}

Функция unidirectional\_sort() реализует алгоритм однонаправленной сортировки массива. В данной функции производится обход неотсортированной части массива в поисках наименьшего значения и его перемещение в начало массива. Данная итерация повторяется до тех пор, пока весь массив не будет отсортирован.

unsigned int unidirectional\_sort(vector<int>& mas)

{

unsigned int start\_time = clock();

int N = size(mas), tmp;

for (int i = 0; i < N; i++)

{

int index = i;

for (int j = i + 1; j < N; j++)

{

if (mas[index] > mas[j])

{

index = j;

}

}

tmp = mas[i];

mas[i] = mas[index];

mas[index] = tmp;

}

unsigned int end\_time = clock();

return end\_time - start\_time;

}

В функция bidirectional\_sort() выполняется двунаправленная сортировка массива. Алгоритм данной функции заключается в том, что в случайно сгенерированном массиве целых чисел происходит поиск максимального и минимального элементов и перемещение их в конец и начало отсортированной части массива соответственно. Алгоритм продолжает свою работу пока в не отсортированной части массива ничего не останется.

unsigned int bidirectional\_sort(vector<int>& mas)

{

unsigned int start\_time = clock();

int N = size(mas);

for (int i = 0, j = N - 1; i < j; i++, j--)

{

int min = mas[i], max = mas[i];

int min\_i = i, max\_i = i;

for (int k = i; k <= j; k++)

{

if (mas[k] > max)

{

max = mas[k];

max\_i = k;

}

else if (mas[k] < min)

{

min = mas[k];

min\_i = k;

}

}

swap(mas[i], mas[min\_i]);

if (mas[min\_i] == max)

swap(mas[j], mas[min\_i]);

else

swap(mas[j], mas[max\_i]);

}

unsigned int end\_time = clock();

return end\_time - start\_time;

}

Для функции print() реализована перегрузка. В первом случае функция выводит отсортированный массив и время работы выбранного алгоритма. Во втором случае выводится только упорядоченный массив.

void print(vector<int>& mas, unsigned int search\_time)

{

int N = size(mas);

for (int i = 0; i < N; i++)

cout << mas[i] << ' ';

cout << endl << "затраченое время: " << search\_time << " сек.";

}

void print(vector<int>& mas)

{

int N = size(mas);

for (int i = 0; i < N; i++)

cout << mas[i] << ' ';

cout << endl;

}

В функциях void read и void write была реализована работа с файлами. В первом случае мы записываем данные в файл, во втором считываем их из файла. Данная функция позволяет не потерять массив данных при выключении программы.

void read(vector<int>& mas, int N)

{

ifstream fin("out.txt");

int tmp;

for (int i = 0; i < N; i++)

{

fin >> tmp;

mas.push\_back(tmp);

}

fin.close();

}

void write(vector<int>& mas)

{

int N = size(mas);

ofstream fout("rez.txt");

for (int i = 0; i < N; i++)

{

fout << mas[i] << " ";

}

fout.close();

}

**4. Схемы программы**

Понятие схемы программы принадлежит советскому математику Ляпунову А. А., которое он ввел в 1953г., исходя из общей концепции необходимости и возможности формализации процесса программирования.

В настоящее время теория схем программ это - широко разветвленная область исследования, которая имеет многочисленные выходы в практику программирования и содержит фундаментальные результаты не только по операторным методам программирования, определившим современное состояние автоматизации программирования, но и по признаваемым перспективными методам рекурсивного и параллельного программирования.

Схема программ как математическая модель удовлетворяет следующим требованиям:

1. Модель позволяет изучать свойства достаточно широких классов программ, а не отдельных конкретных программ;

2. Сохраняет все интересующие исследователя свойства и особенности рассматриваемого класса программ;

3. Позволяет игнорировать несущественные для данной проблемы свойства, например, синтаксические детали;

4. Модель модифицируема и допускает нововведения, отслеживающие развитие языков программирования;

5. Удобно, когда структура модели изобразительно подобна структуре программ, что дает возможность привлекать на некоторых этапах исследований программистскую интуицию.

Подтверждением последнего требования является то, что схемы программ также, как и программы, представляют собой тексты в некоторых формальных языках, которые являются упрощенными репликами языков программирования.

4.1. Блок-схема программы

На рисунке 1 изображена блок-схема алгоритма работы пользовательского интерфейса. Данный алгоритм описывает возможности выбора в меню. Каждый пункт меню открывает определённую функциональность программы. Пользовательский интерфейс выполнен в консольном варианте.

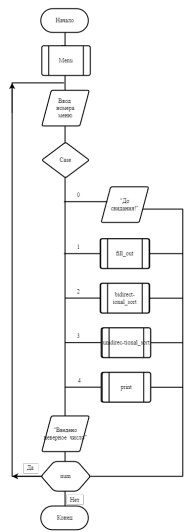


Рисунок 1 – Блок-схема главной функции

На рисунке 2 изображена блок-схема алгоритма заполнения целевого массива случайными целыми числами в диапазоне от -100 до 101.

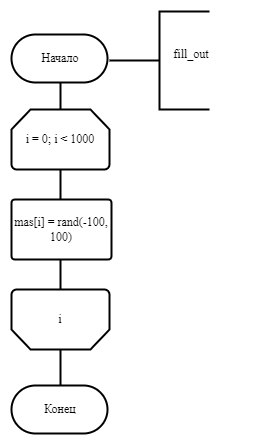


Рисунок 2 - Блок-схема функции заполнения

На рисунке 3 изображена блок-схема алгоритма однонаправленной сортировки выбором. Данный алгоритм условно делит массив на две части отсортированный и не отсортированный, ищет минимальный элемент массива. После того как он находит его, алгоритм меняет значения местами с первым элементом массива и делает шаг на второй элемент. Первый помечается как отсортированный, а всё что после остаются неизменными. После алгоритм проходит по неотсортированной находя следующий наименьший элемент. И так до конца массива постепенно увеличивая отсортированную область пока все элементы массива не будут отсортированы.

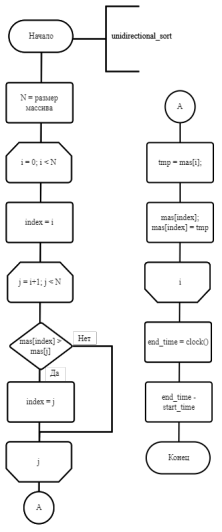


Рисунок 3 – Блок-схема однонаправленный функции сортировки выбором

На рисунке 4 изображена блок-схема алгоритма двунаправленной сортировки выбором. Этот алгоритм также условно делит массив на отсортированную и не отсортированную. Их отличие состоит в том, что при проходе по массиву алгоритм ищет и максимальный и минимальный элементы массива. Отсортировывая массив как с начала, так и с конца, оставляет неотсортированную часть в центре постепенно уменьшаться.

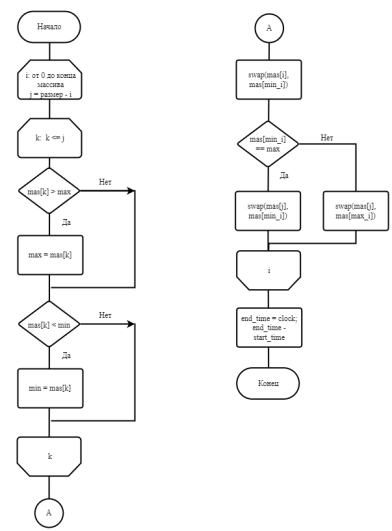


Рисунок 4 – Блок-схема двунаправленный функции сортировки выбором

На рисунке 4 изображена блок-схема алгоритма вывода. Если пользователь выбрал действие заполнения массива, но не использовал ни одного из алгоритмов сортировки выбором в консоль будет выведен неотсортированный массив. В противным случае в консоли будет выведен упорядоченный массив и время работы алгоритма.

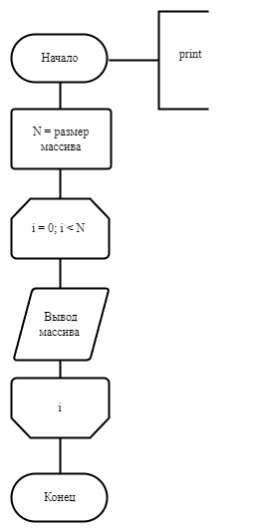


Рисунок 5 – Блок-схема функции вывода

4.2. Блок-схема алгоритма

Метод сортировки выбором. Исходный массив длиной N разбивается на две части: итог и остаток. Участок массива, называемый итогом, располагается с начала массива и должен быть упорядоченным, а участок массива, называемый остатком, располагается вплотную за итогом и содержит исходные числа не отсортированной части исходного массива.

Однонаправленный алгоритм сортировки выбором:

Шаг 1. Полагается i=0, т.е. считается, что итоговый участок - пуст.

Шаг 2. В остатке массива ищется минимальный элемент, он меняется местом с первым элементом остатка (i-ым элементом массива). После чего значение i увеличивается на единицу, тем самым расширяя итоговый участок массива (отсортированную часть исходного массива).

Шаг 3. Если i < N, то повторяется Шаг 2. В противном случае - конец алгоритма, т.к. итог становится равным всему массиву.

Конец алгоритма.

Схема алгоритма методом сортировки выбора представлена на рис. 6.

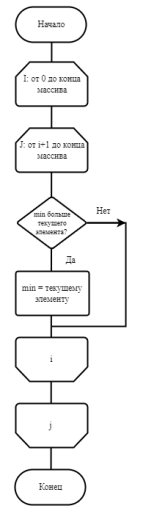


Рисунок 6 – Блок-схема однонаправленного алгоритма сортировки выбором

Двунаправленный метод сортировки выбором. Исходный массив длиной N разбивается на две части: итог и остаток. Участок массива, называемый итогом, располагается с начала и конца массива, он должен быть упорядоченным. А участок массива, называемый остатком, располагается вплотную между итогами и содержит исходные числа не отсортированной части исходного массива.

Двунаправленного алгоритма сортировки выбором:

Шаг 1. Полагается i=0, т.е. считается, что итоговый участок - пуст.

Шаг 2. В остатке массива ищется максимальный или минимальный элемент, они меняются местами с первым и последним элементом остатка соответственно. После чего значение i увеличивается на две единицу, тем самым расширяя итоговый участок массива (отсортированную часть исходного массива).

Шаг 3. Если i < N, то повторяется Шаг 2. В противном случае - конец алгоритма, т.к. итог становится равным всему массиву.

Конец алгоритма.

Схема алгоритма методом сортировки выбора представлена на рис. 7.

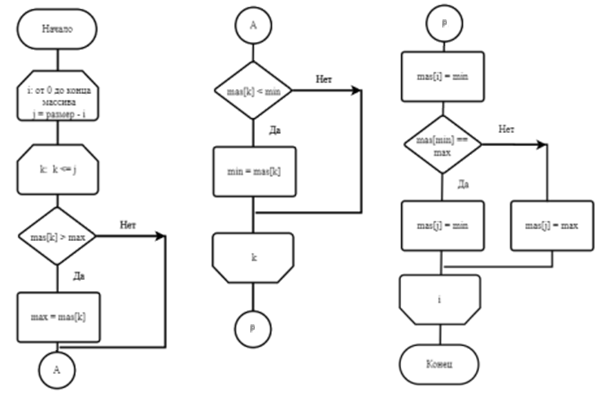


Рисунок 7 – Блок-схема двунаправленного алгоритма сортировки выбора

**5. Тестирование программы**

Перед тем как человек начнет пользоваться новой версией компьютерной программы, сайта или мобильного приложения, продукт должен быть протестирован. Тестирование программы это - проверка соответствия между реальным и ожидаемым поведением программы, осуществляемая на конечном наборе тестов, выбранном определенным образом. В более широком смысле, тестирование - одна из техник контроля качества, включающая в себя активности по планированию работ, проектированию тестов, выполнению тестирования и анализу полученных результатов.

5.1. Тестирование на разных наборах данных

Тестовый набор данных, а также результаты тестирования представлены в таблице 1. Результаты тестирования приведены в приложении А.

Таблица 1 - Таблица тестов

|  |  |  |  |
| --- | --- | --- | --- |
| № теста | Размер массива, элементы | Время выполнения однонаправленной сортировки, сек | Время выполнения двунаправленной сортировки, сек |
| 1 | 10 000 | 2,255 | 1,081 |
| 2 | 20 000 | 8,867 | 4,317 |
| 3 | 30 000 | 19,962 | 9,707 |
| 4 | 40 000 | 35,463 | 17,252 |
| 5 | 50 000 | 55,446 | 26,950 |
| 6 | 60 000 | 79,816 | 38,816 |
| 7 | 70 000 | 108,724 | 52,824 |
| 8 | 80 000 | 143,251 | 69,004 |
| 9 | 90 000 | 181,569 | 87,322 |
| 10 | 100 000 | 224,496 | 107,875 |
| 11 | 110 000 | 270,493 | 130,477 |

На рисунках 8, 9 изображен пример работы программы. На первом рисунке массив заполнен случайными данными. На втором произведена сортировка и выведен на экран упорядоченный массив.

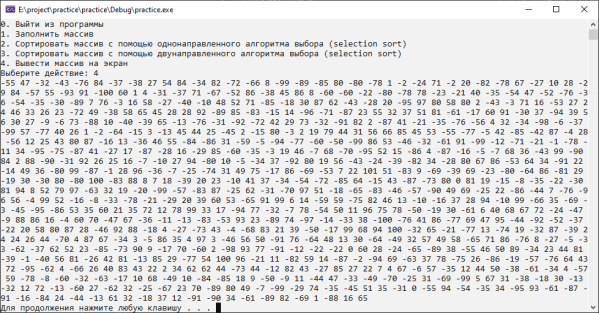


Рисунок 8 – Сгенерированный массив

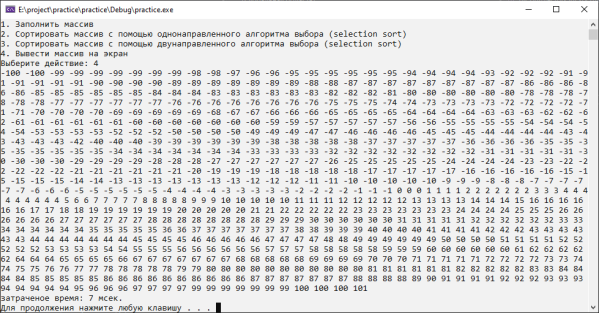


Рисунок 9 – Отсортированный массив

5.2. Анализ полученных результатов тестирования (анализ работы алгоритма)

На основании анализа данных, полученных в результате тестирования алгоритма сортировки выбором, можно сделать вывод, что время, затраченное на работу программы относительно количества элементов, увеличивается линейно, то есть с увеличением количества элементов пропорционально увеличивается время работы программы, а также видна повышенная эффективность двухсторонней сортировки по затраченному времени в 2 раза.

Рисунок 10 – Результаты тестирования

**6. Отладка**

В ходе отладки был выявлен незначительный графический дефект, не влияющий на работоспособность программы. Была перепутана нумерация пунктов меню, что могло привести к затруднению использования данный программы конечным пользователем. Данная ошибка зафиксирована на рисунке 11 и исправлена на рисунке 12.

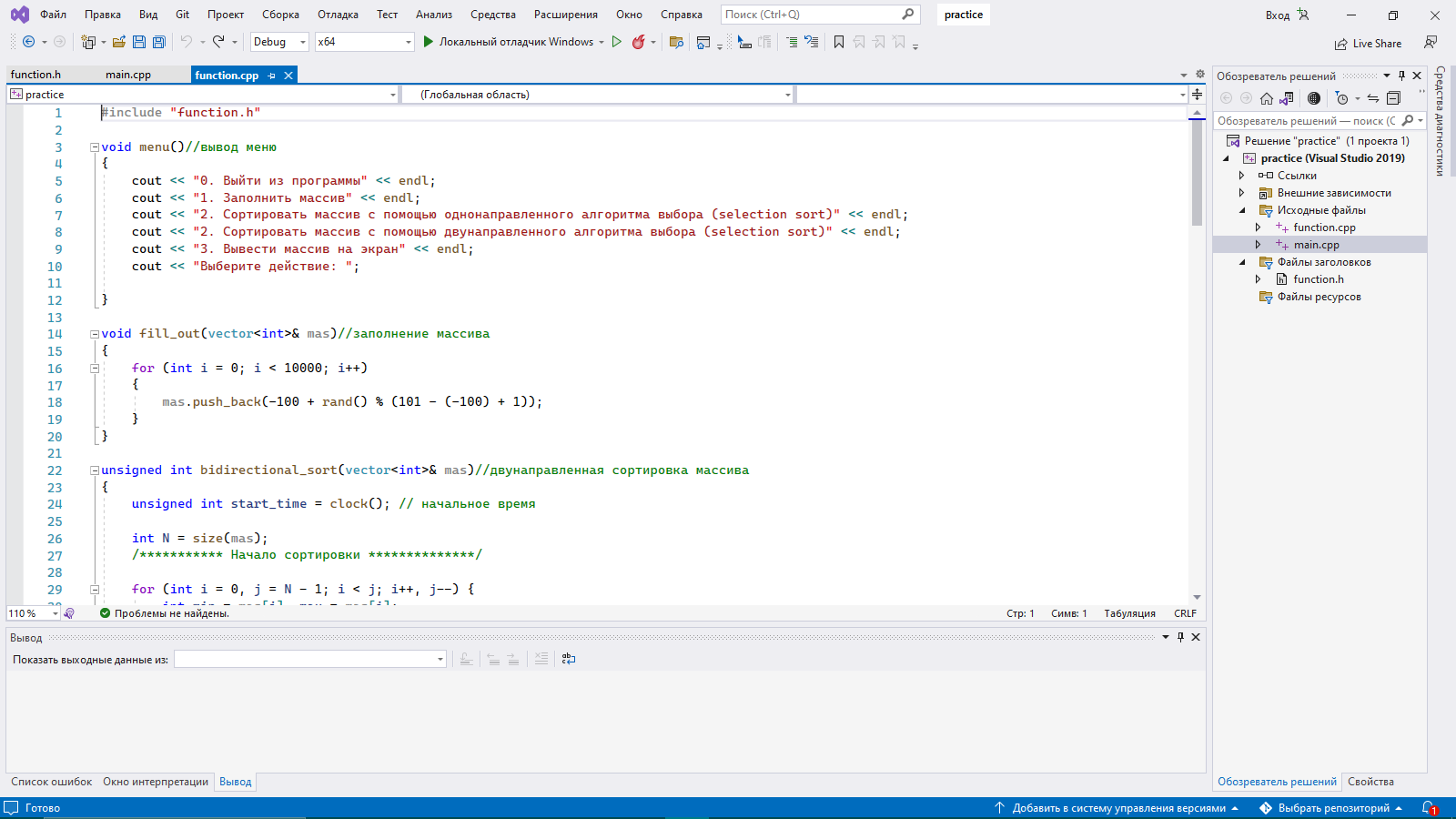


Рисунок 11 – Неправильная нумерация функции меню

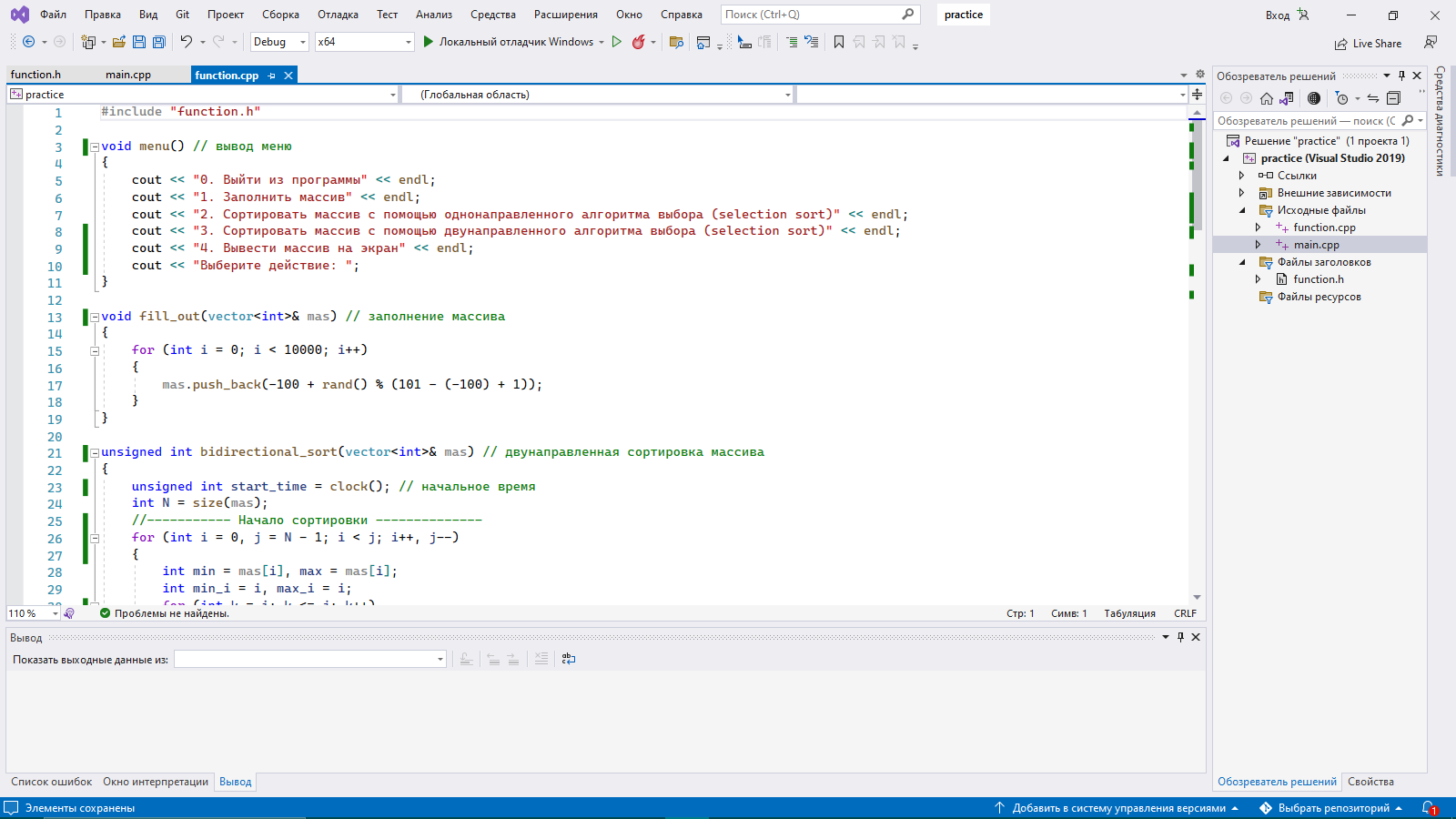


Рисунок 12 – Исправленная нумерация функций меню

За исключением вышеописанной ошибки других достойных упоминания дефектов не было зафиксировано. Программа работает исправно в соответствии с ожиданиями.

**7. Совместная разработка**

В ходе совместной разработки моя роль заключалась в проверке работоспособности программы, поиске орфографических и логических ошибок и тестировании программы на разных наборах данных. Так же я осуществлял проверку контроля версий через GitHab, исправил графический дефект и поддерживал связь через приложение Discord.

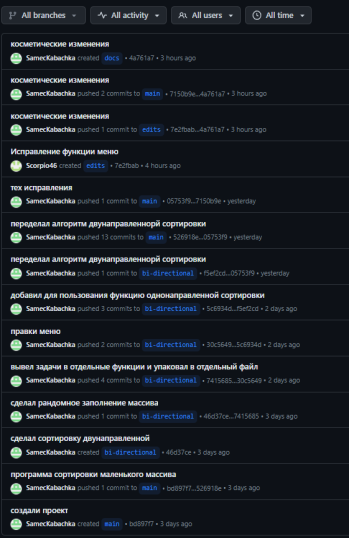


Рисунок 13 – История изменения репозитория

Заключение

При выполнении данной работы были получены навыки совместной работы с помощью сервиса GitHub, навыки использования программы Git Bash. Был изучен алгоритм сортировки вставками. Мною был написан алгоритм, считающий количество перестановок элементов массива во время сортировки и время работы программы в секундах. Было выполнено тестирование программы на разных наборах данных и отладка данной программы. При выполнении практической работы были улучшены базовые навыки программирования на языке С++. Улучшены навыки отладки, тестирования программ и работы с системами контроля версий.

В дальнейшем программу можно улучшить путем подключения упрощающих реализацию данной сортировки библиотек и улучшения псевдографического интерфейса.

Список используемой литературы

1. ГОСТ 2.105–95 «Общие требования к текстовым документам»

2. Сортировка выбором [Электронный ресурс]. – Режим доступа: https://habr.com/ru/companies/otus/articles/509674/

3. Введение в C++ | Язык программирования C++ [Электронный ресурс]. – Режим доступа: https://metanit.com/cpp/tutorial/1.1.php

4. РАЗДЕЛ 5. Введение в теорию схем программ [Электронный ресурс]. – Режим доступа: https://studopedia.org/3-119064.html

Приложение А

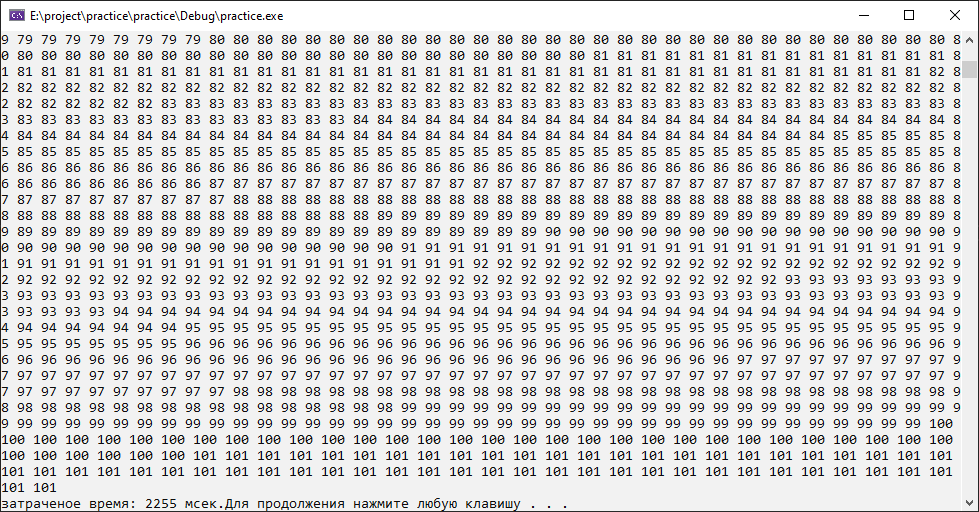


Рисунок А1 – 10 000 элементов

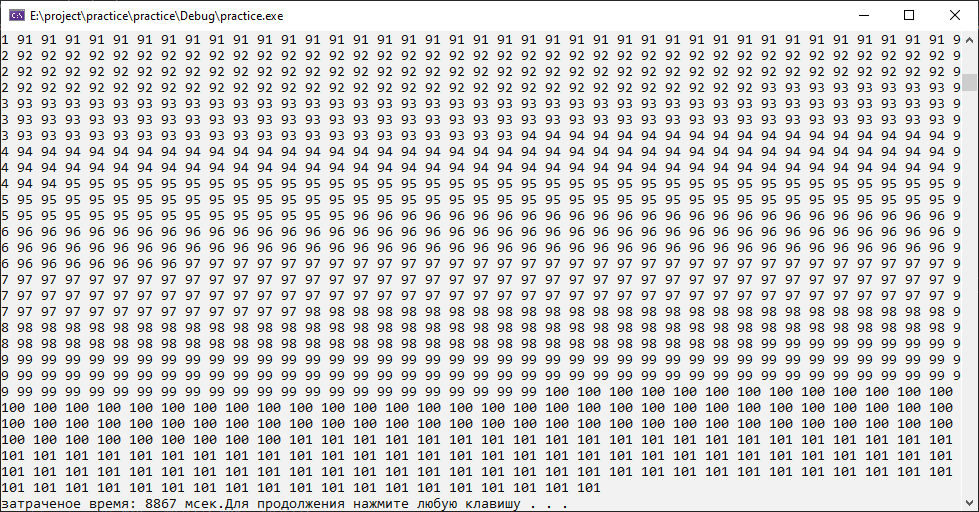


Рисунок А2 – 20 000 элементов

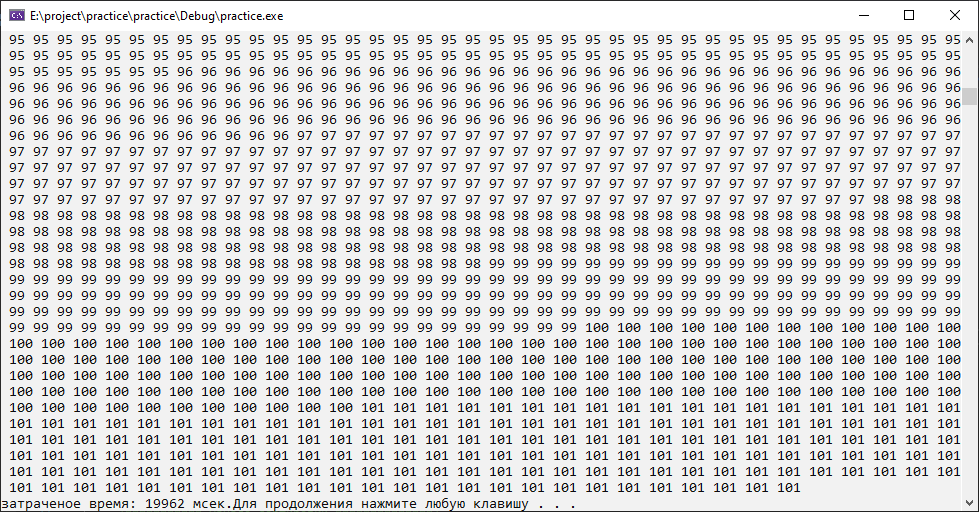


Рисунок А3 – 30 000 элементов

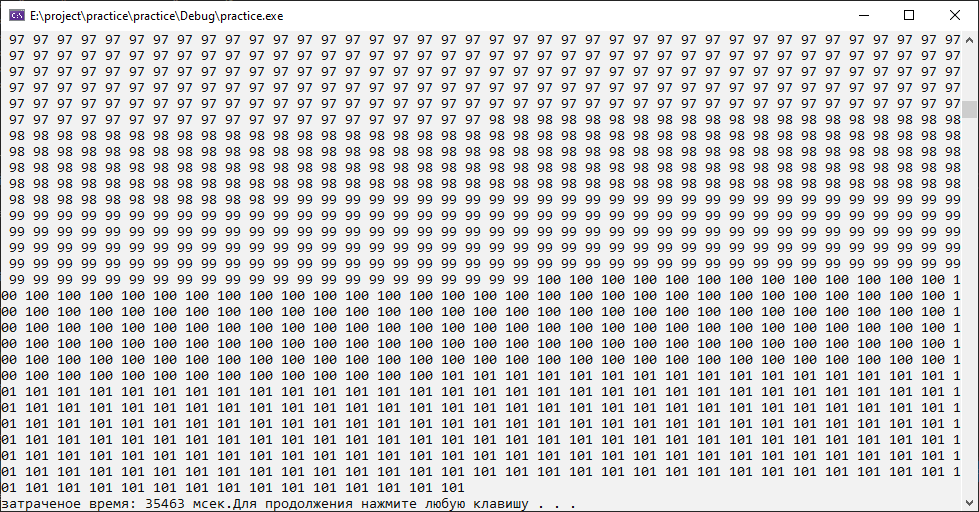


Рисунок А4 – 40 000 элементов

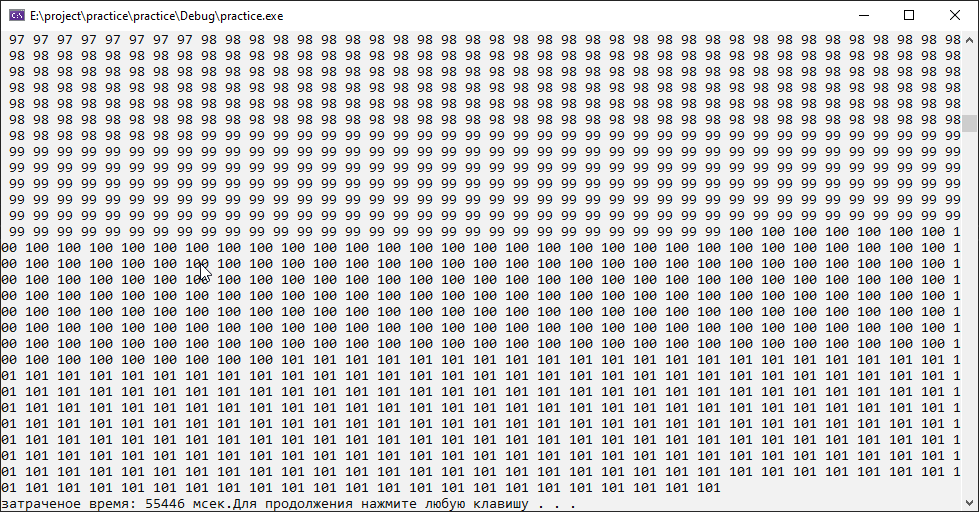


Рисунок А5 – 50 000 элементов

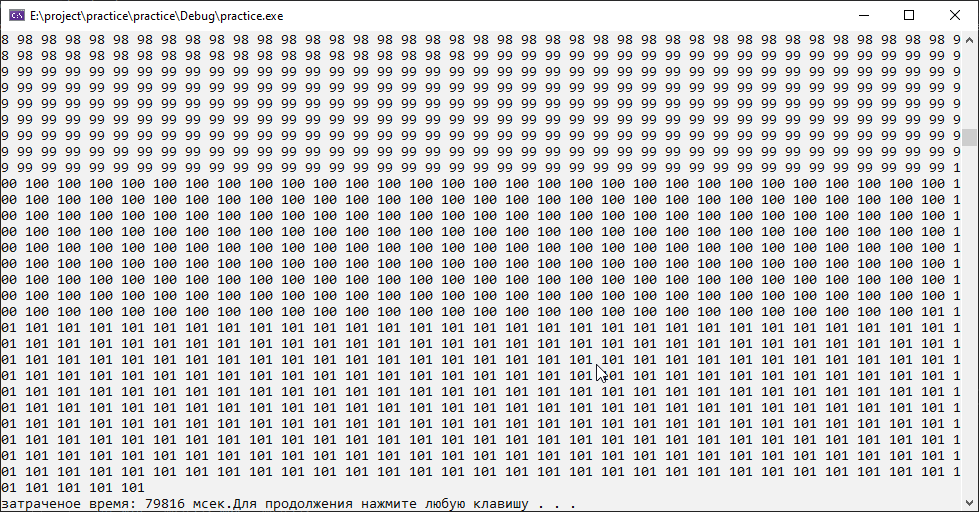


Рисунок А6 – 60 000 элементов

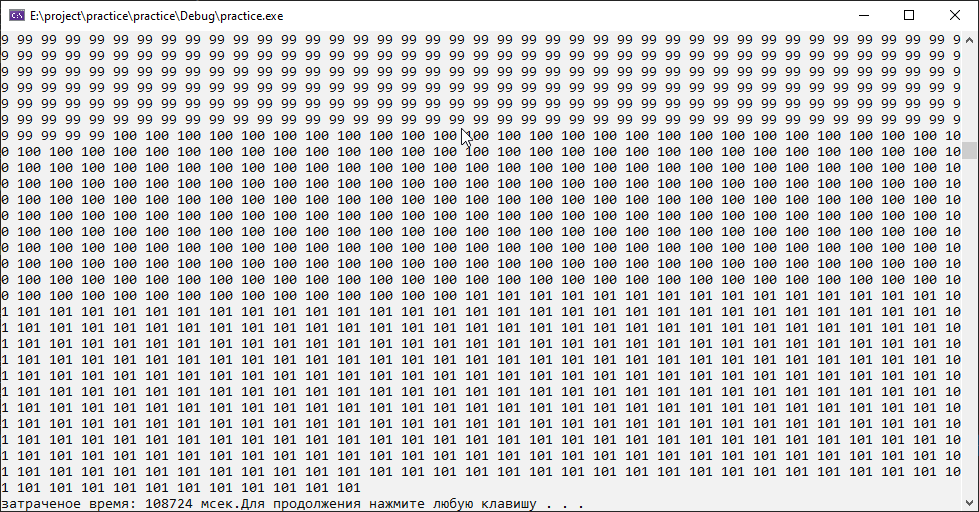


Рисунок А7 – 70 000 элементов

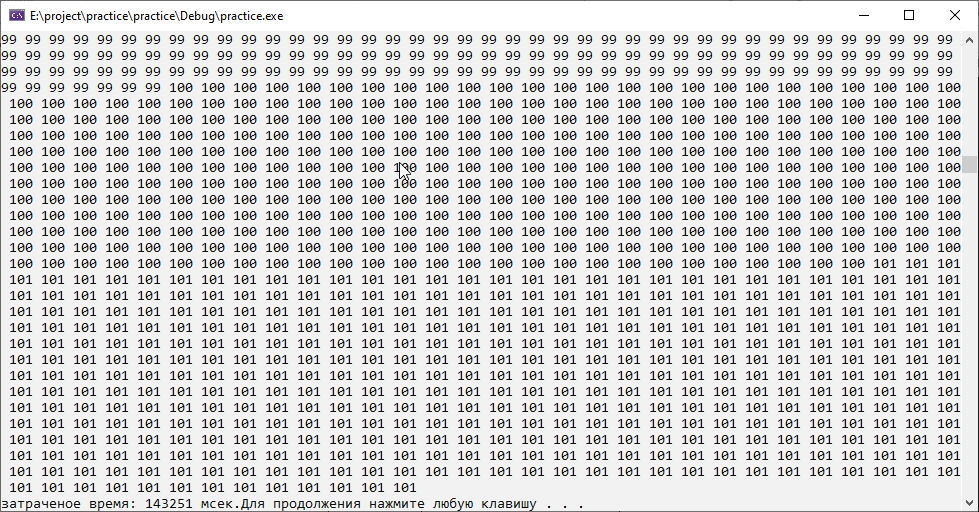


Рисунок А8 – 80 000 элементов

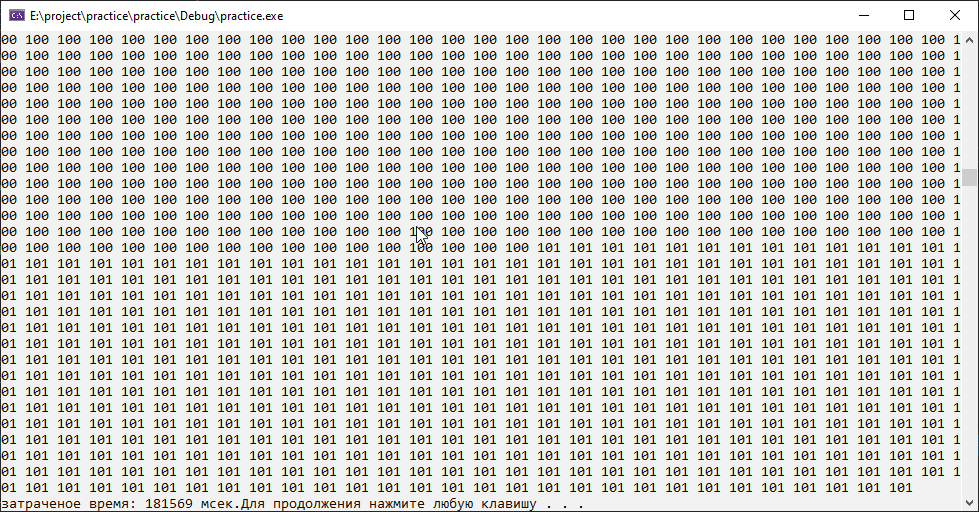


Рисунок А9 – 90 000 элементов

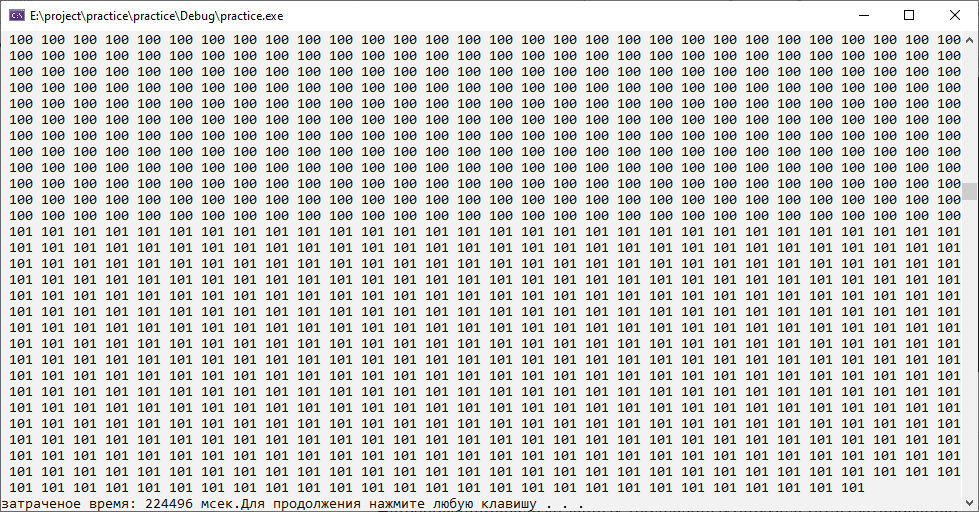


Рисунок А10 – 100 000 элементов

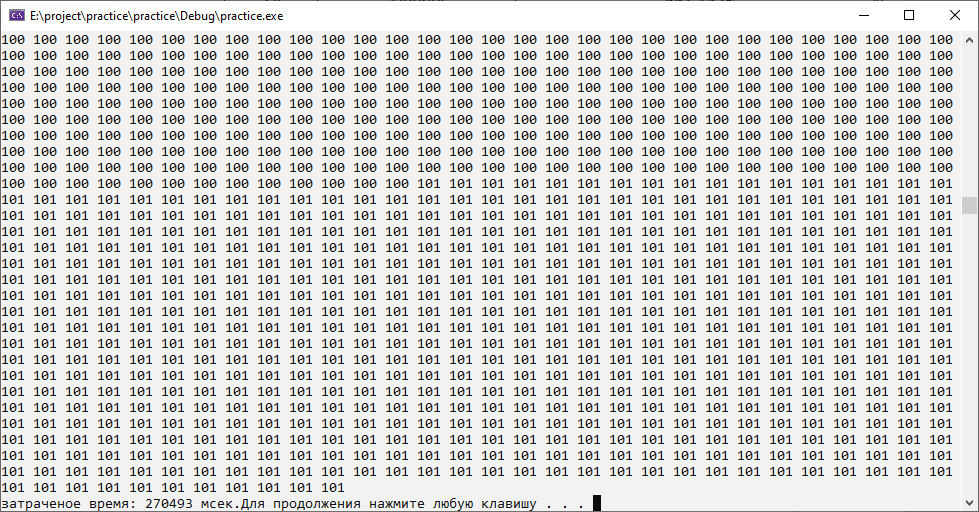


Рисунок А11 – 110 000 элементов

Приложение Б Листинг

Файл function.cpp

#include "function.h"

void menu() // вывод меню

{

cout << "0. выйти из программы" << endl;

cout << "1. заполнить массив" << endl;

cout << "2. считать массив" << endl;

cout << "3. отсортировать массив с помощью однонаправленного алгоритма выбора (selection sort)" << endl;

cout << "4. отсортировать массив с помощью двунаправленного алгоритма выбора (selection sort)" << endl;

cout << "5. вывести массив на экран" << endl;

cout << "6. записать массив в файл" << endl;

cout << "выберите действие: ";

}

void fill\_out(int N) // заполнение массива

{

ofstream fout("out.txt");

for (int i = 0; i < N; i++)

{

fout << (-100 + rand() % (101 - (-100) + 1)) << " ";

}

fout.close();

}

void read(vector<int>& mas, int N)

{

ifstream fin("out.txt");

int tmp;

for (int i = 0; i < N; i++)

{

fin >> tmp;

mas.push\_back(tmp);

}

fin.close();

}

void write(vector<int>& mas)

{

int N = size(mas);

ofstream fout("rez.txt");

for (int i = 0; i < N; i++)

{

fout << mas[i] << " ";

}

fout.close();

}

unsigned int bidirectional\_sort(vector<int>& mas)

{

unsigned int start\_time = clock(); // начальное время

int N = size(mas);

//------------начало сортировки---------------

for (int i = 0, j = N - 1; i < j; i++, j--)

{

int min = mas[i], max = mas[i];

int min\_i = i, max\_i = i;

for (int k = i; k <= j; k++)

{

if (mas[k] > max)

{

max = mas[k];

max\_i = k;

}

else if (mas[k] < min)

{

min = mas[k];

min\_i = k;

}

}

swap(mas[i], mas[min\_i]);

if (mas[min\_i] == max)

swap(mas[j], mas[min\_i]);

else

swap(mas[j], mas[max\_i]);

}

unsigned int end\_time = clock(); // конечное время

return end\_time - start\_time; // искомое время

}

unsigned int unidirectional\_sort(vector<int>& mas)

{

unsigned int start\_time = clock(); // начальное время

int N = size(mas), tmp;

for (int i = 0; i < N; i++)

{

int index = i;

for (int j = i + 1; j < N; j++)

{

if (mas[index] > mas[j])

{

index = j;

}

}

tmp = mas[i];

mas[i] = mas[index];

mas[index] = tmp;

}

unsigned int end\_time = clock(); // конечное время

return end\_time - start\_time; // искомое время

}

void print(vector<int>& mas, unsigned int search\_time)

{

int N = size(mas);

for (int i = 0; i < N; i++)

cout << mas[i] << ' ';

cout << endl << "затраченое время: " << search\_time << " сек.";

}

void print(vector<int>& mas) // выворд массива

{

int N = size(mas);

for (int i = 0; i < N; i++)

cout << mas[i] << ' ';

cout << endl;

}

Файл function.h

#pragma once

#include <iostream>

#include <fstream>

#include <vector>

#include <stdlib.h> // нужен для вызова функций rand(), srand()

#include <time.h> // нужен для вызова функции time()

#include <ctime>

#include <windows.h>

using namespace std;

void menu(); // вывод меню

void fill\_out(int N); // заполнение массива

void read(vector<int>& mas, int N);

void write(vector<int>& mas);

unsigned int bidirectional\_sort(vector<int>& mas);

unsigned int unidirectional\_sort(vector<int>& mas);

void print(vector<int>& mas, unsigned int search\_time);

void print(vector<int>& mas); // выворд массив

Файл main.cpp

#include "function.h"

const int N = 1000;

int main()

{

//------------------блок переменных-----------------

unsigned int num, search\_time = NULL;

vector <int> mas;

//-------------------блок настроек------------------

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

srand(time(NULL));

//---------------------блок кода--------------------

do {

menu();

cin >> num;

switch (num)

{

case 0:

cout << "До свидания!";

break;

case 1:

fill\_out(N);

cout << "массив заполнен" << endl;

system("pause");

break;

case 2:

read(mas, N);

cout << "массив считан" << endl;

system("pause");

break;

case 3:

search\_time = unidirectional\_sort(mas);

cout << "массив отсортирован" << endl;

system("pause");

break;

case 4:

search\_time = bidirectional\_sort(mas);

cout << "массив отсортирован" << endl;

system("pause");

break;

case 5:

if (search\_time == NULL)

{

print(mas);

}

else print(mas, search\_time);

system("pause");

break;

case 6:

write(mas);

cout << "массив записан" << endl;

system("pause");

break;

default:

cout << " Введено неверное число" << endl;

}

system("CLS");

} while (num);

return 0;

}