МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования "**Национальный исследовательский** **Нижегородский** **государственный** **университет** **им**. **Н.И. Лобачевского**" (**ННГУ**)

**Институт информационных технологий, математики и механики**

Отчет по Лабораторной работе №1

**Работа с файлами, массивами, применение сортировок**

Выполнил:

студент группы 3821Б1ПМ3

Логинов В.А.

Проверил:

заведующий лабораторией суперкомпьютерных технологий и высокопроизводительных вычислений

Лебедев И.Г

Нижний Новгород  
2021

Оглавление

[1. Введение 2](#_Toc91062741)

[2. Постановка задачи 3](#_Toc91062742)

[3. Руководства пользователя 4](#_Toc91062743)

[3.1. Первая программа 4](#_Toc91062744)

[3.2. Вторая программа 5](#_Toc91062745)

[4. Руководство программиста 7](#_Toc91062746)

[4.1. Описание структуры программы 7](#_Toc91062747)

[4.2. Описание структур данных 8](#_Toc91062748)

[4.3. Описание алгоритмов 8](#_Toc91062749)

[4.3.1. Алгоритм заполнения файла случайными числами 8](#_Toc91062750)

[4.3.2. Алгоритм заполнения файла с клавиатуры 8](#_Toc91062751)

[4.3.3. Алгоритм заполнения файла из другого файла 9](#_Toc91062752)

[4.3.4. Алгоритм сортировки пузырек 9](#_Toc91062753)

[4.3.5. Алгоритм сортировки вставкой 10](#_Toc91062754)

[4.3.6. Алгоритм быстрой сортировки 11](#_Toc91062755)

[5. Эксперименты 14](#_Toc91062756)

[6. Заключение 15](#_Toc91062757)

[7. Литература 16](#_Toc91062758)

[Приложение 1 17](#_Toc91062759)

[Приложение 2 19](#_Toc91062760)

# Введение

Любому начинающему программисту необходимо освоить базовые принципы работы языков программирования. Язык С нередко называют языком программирования "среднего уровня" или даже "низкого уровня", так как он сочетает элементы языков высокого уровня с функциональностью и производительностью ассемблера и работает близко к аппаратной части компьютера. В итоге мы можем манипулировать данными на низком уровне и при этом использовать высокоуровневые конструкции для управления работы программы. Поэтому он прекрасно подходить для изучения работы аппаратной части компьютера.

Неотъемлемой частью работы компьютера является использование RAM. Понимание того, на что и чем выделяется память необходимо для написания эффективной в работе программы.

Также, в большинстве программ по анализу данных, происходить работа с числами, поэтому необходимо разбираться в типах данных и их применении, для симуляции работы с настоящими данными была использована случайная генерация чисел

Необходимость сохранения результатов работы для дальнейшего использования приводит к неизбежной работе с файлами, извлечения информации из них для программы и сохранения в файлы после работы кода. Также немаловажно представлять данные не в хаотичном порядке, трудно усвояемом человеком при большом объеме, а в упорядоченном, отсортированном. Для этих целей используются различные сортировки, каждая из которых подходит под конкретные задачи.

# Постановка задачи

Написать программу, которая состоит из двух подпрограмм. Первая программа создает текстовый файл с записанными в него числами. Программа принимает количество чисел n, максимальное и минимальное значение, также программа может создавать массив не только из случайных чисел, но и получать их с клавиатуры и из файла. Вторая программа читает текстовый файл с набором чисел, выводит консольный интерфейс (печать, сортировка, сброс, выход), выполняет выбранные действия. Реализовать три сортировки: пузырьком, вставкой, быстрая.

# Руководства пользователя

## Первая программа

После запуска программы откроется окно командной строки, где будет предложено выбрать один из трех источников данных (случайные числа, вводимые с клавиатуры, копируемые из файла):

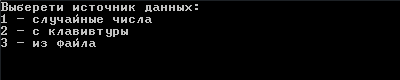


Рис 1. Выбор источника данных

Для выбора одного из источников необходимо нажать соответствующую ему цифру и клавишу «Enter».

При выборе источника «случайные числа» последовательно будет предложено ввести количеству необходимых чисел, минимальное из них и максимальное.

1.png

Рис.2 консоль для генерации случайных чисел

После ввода каждого значения необходимо нажимать клавишу «Enter».

При выборе источника данных «с клавиатуры» программа попросить ввести количество чисел, которое будет введено далее:

1.png

Рис.3 Консоль после выбора пункта «с клавиатуры»

После ввода количества и нажатия клавиши «Enter» программа начнет принимать вводимые с клавиатуры числа, после ввода каждого нажимайте «Enter», чтобы ввести следующее.

При выборе источника данных «из файла» программа предложит ввести полный путь к файлу, являющемуся источником данных.

1.png

Рис.4 Консоль после выбора файла, как источника данных

1.png

Рис.5 Пример ввода пути до файла

После выполнения одного из сценариев, программа создаст файл с числами, завершив свою работу.

## Вторая программа

После запуска программы будет предложено выбрать одно из четырех возможных действий: печать, сортировка, сброс, выход.

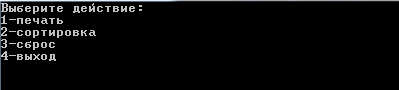


Рис. 6 Консоль после запуска второй программы

Для выбора одного из действий необходимо нажать соответствующую этому пункту цифру и клавишу «Enter». После каждого выполнения одного из трех действий, начальная консоль будет появляться снова.

При выборе пункта «выход» программа сохранит массив данных в файл, в том виде, в котором он сейчас находится и завершит работу.

При выборе действия «печать», программа осуществит распечатывание массива данных, с которым ведется работа на экран:

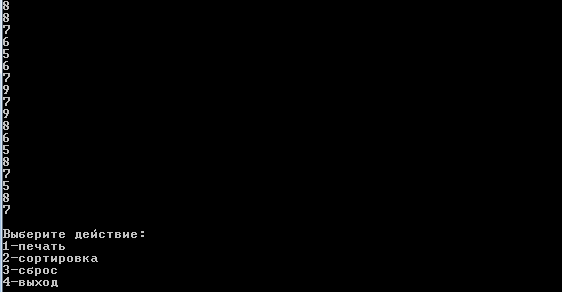


Рис.7 Консоль после действия «печать»

Выбор пункта «сортировка» выведет предложение выбрать одну из трех сортировок:

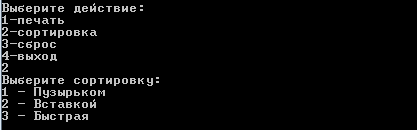


Рис.8 Консоль предлагает выбрать, какую сортировку применить

После выбора одной из сортировок, программа отсортирует массив по данному алгоритму.

Выбор пункта «сброс» вернет данные в массиве к формату, который он имел при запуске программы, то есть все примененные сортировки будут сброшены.

# Руководство программиста

## Описание структуры программы

Обе программы состоят из трех файлов каждая, один из них заголовочной файл, для объявления функций, другой файл отвечает за наполнение функций, а третий содержит исполняющую функцию main, в котором содержится основной алгоритм работы программы.

В первой программе в исполняющем файле «main\_labs\_1f.c» всего один блок кода, отвечающий за наполнения данными файла, для дальнейшей работы с ним второй программой. В файле с функциями «def\_labs\_1f.c» имеется три блока, каждый из которых содержит в себе функцию для одного из вариантов заполнения файла с данными:

1. «rand\_in\_f» помещает в файл случайные числа в количестве и диапазоне задающимися пользователем
2. «polzovat\_in\_f» помещает в файл числа вводимые с клавиатуры
3. «f\_in\_f» копирует данные из указанного пользователем файла в рабочий файл

Во второй программе в исполняемом файле «main\_labs\_2f.c» также один блок кода, отвечающий за работу с массивом данных (печать, сортировка, сброс, сохранение результатов) посредством использования функций, описанных в файле «def\_labs\_2f.c».

В файле «def\_labs\_2f.c» имеются следующие функции:

1. «Puzirek» функция содержащая алгоритм сортировки пузырьком
2. «Vstavka» функция содержащая алгоритм сортировки вставкой
3. «qike\_sort» функция содержащая алгоритм быстрой сортировки
4. «f\_in\_mas» функция копирующая данные из файла в рабочий массив

## Описание структур данных

В программе используются следующие классы переменных:

1. «\*file» типа данных «FILE» соответствующих файлу
2. «int\*» одномерный массив целых чисел
3. «int» для переменных, длины массива, счетчиков, временных значений целых чисел

## Описание алгоритмов

## Алгоритм заполнения файла случайными числами

Алгоритм реализован отдельной функцией, хранящейся в файле программы «def\_labs\_1f.c». Она принимает на вход количество необходимых чисел для генерации, минимальной и максимальное их них, а также файл для записи. При помощи цикла «for» программа, указанное пользователем количество раз, получает случайное число из встроенной функции «rand» и записывает его в новую строку в файле. После этого файл закрывается и сохраняется.

## Алгоритм заполнения файла с клавиатуры

Алгоритм реализован отдельной функцией, хранящейся в файле программы «def\_labs\_1f.c». Она принимает на вход количество чисел которые будут введены далее и файл для их сохранения. Сохраняя это количество в локальную переменную i, функция запускает цикл «for» i раз, на каждой итерации которого считывает вводимое пользователем число и записывает его в новую строку в файле. После этого файл закрывается и сохраняется.

## Алгоритм заполнения файла из другого файла

Алгоритм реализован отдельной функцией, хранящейся в файле программы «def\_labs\_1f.c». Она принимает на вход файл, в который будут сохранятся данные. Далее функция просит пользователя ввести путь к файлу, являющемуся источником данных, открывает его, и пробегаясь по всем строкам в нем узнает их количество, которое сохраняет в локальную переменную i. Далее функция запускает цикл «for» i раз, на каждой итерации которого считывает новую строку из файла-источника и записывает ее в новую строку в файле для сохранения. После этого оба файла закрываются и сохраняются.

## Алгоритм сортировки пузырек

Функция принимает на вход массив данных и его длину. Далее функция проходит по всему массиву циклом «for» столько раз, сколько в нем есть элементов. За каждый такое проход функция берет первый элемент и последовательно сравнивает его с каждым последующим, если находится элемент больше взятого сейчас, то он перезаписывается, и продолжается сравниваться с последующими. Таким образом, функция за каждый свой проход вытягивает наибольший из не отсортированных элементов в конец не отсортированной части массива и после завершения работы цикла массив будет отсортирован.

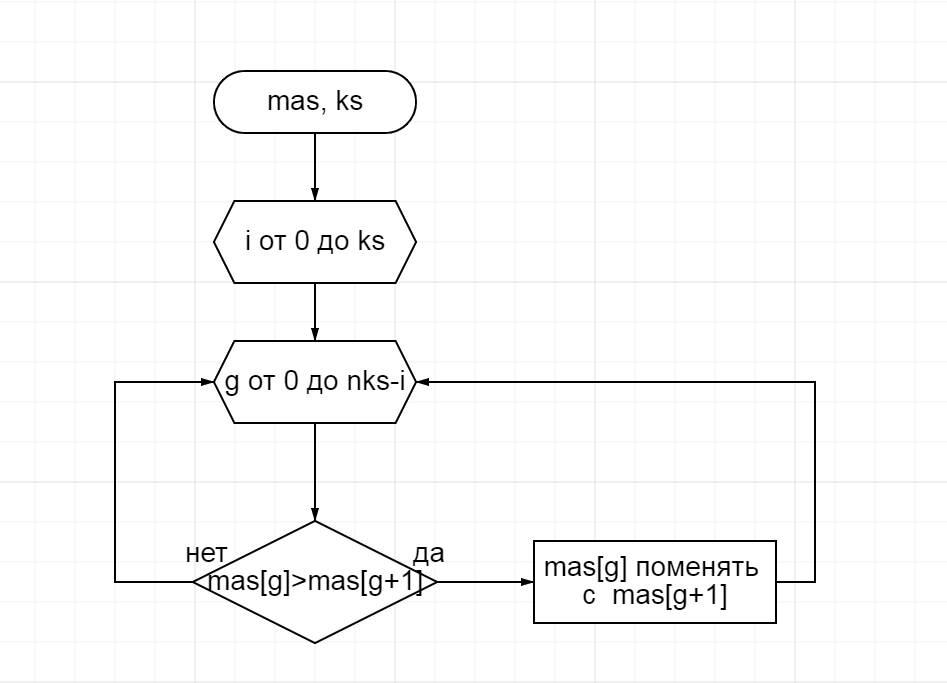


Рис.9 Блок-схема алгоритма сортировки пузырьком

## Алгоритм сортировки вставкой

Функция принимает на вход массив данных и его длину. Функция запускает цикл «for», идущий от начала и до конца массива. На каждой его итерации функции берет очередной элемент, и если он оказывается меньше находящегося слева элемента, но они меняются места. Так происходить до тех пор, пока елемент не окажется на первом месте или на месте, когда он меньше элемента справа, но больше элемента слева, то есть занимает логическое место в возрастающем массиве. Такие действия повторяются для каждого элемента массива, в результате чего он становится упорядоченным по возрастанию.

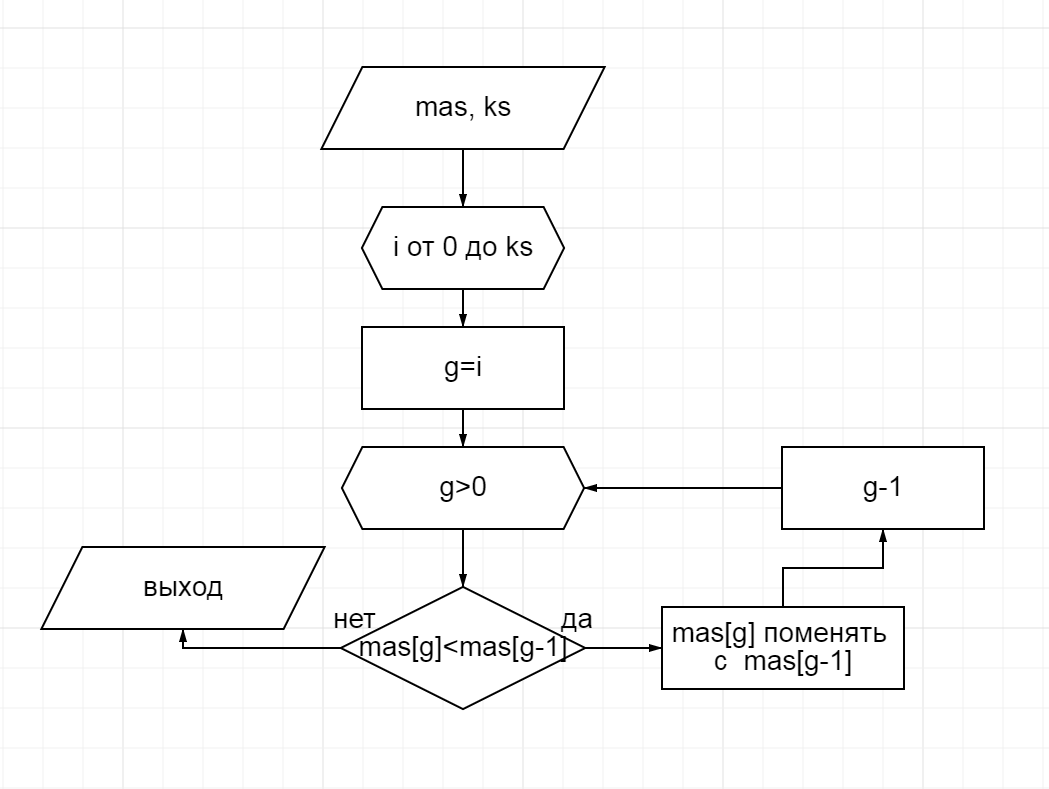


Рис.10 Блок-схема сортировки вставкой

## Алгоритм быстрой сортировки

Функция принимает на вход массив данных для сортировки, а также два числа, соответствующие началу и концу промежутка массива, который необходимо отсортировать. Далее функция записывает в переменную l значение индекса элемента массива, соответствующего началу фрагмента подлежащего сортировке, а в переменную r значение индекса элемента массива, соответствующего концу фрагмента подлежащего сортировке. После функция берет в качестве элемента, относительно которого будет происходить сортирование, элемент находящийся по середине сортируемой части. После чего циклом с предусловием, функция, одновременно идя с начала и конца сортируемой части( посредством увеличения индекса для левого элемента и уменьшение индекса для правого элемента), ищет в каждой из них элемент, который, для левой части является большим относительно среднего, а для в правой части является меньшим относительно среднего. Как только такие элементы были найдены, они меняются местами и функция продолжает идти по элементам, пока индексы правого и левого элемента не поровняются с индексом серединного элемента. Таким образом, в части, от начала до середины массива, у нас окажутся числа меньше среднего элемента, а в правой, числа, больше среднего элемента, но все они будут идти в хаотичном порядке. Для сортировки каждой из двух частей функция вызовет саму себя и произведет действия описанные выше для каждого вызова. Данные вызовы будут происходить до тех пор, пока массив не будет поделен на части по 3 элемента, где невозможно иметь как итог правую и левую часть из отсортированных по среднему элементу чисел, но в хаотичном порядке.

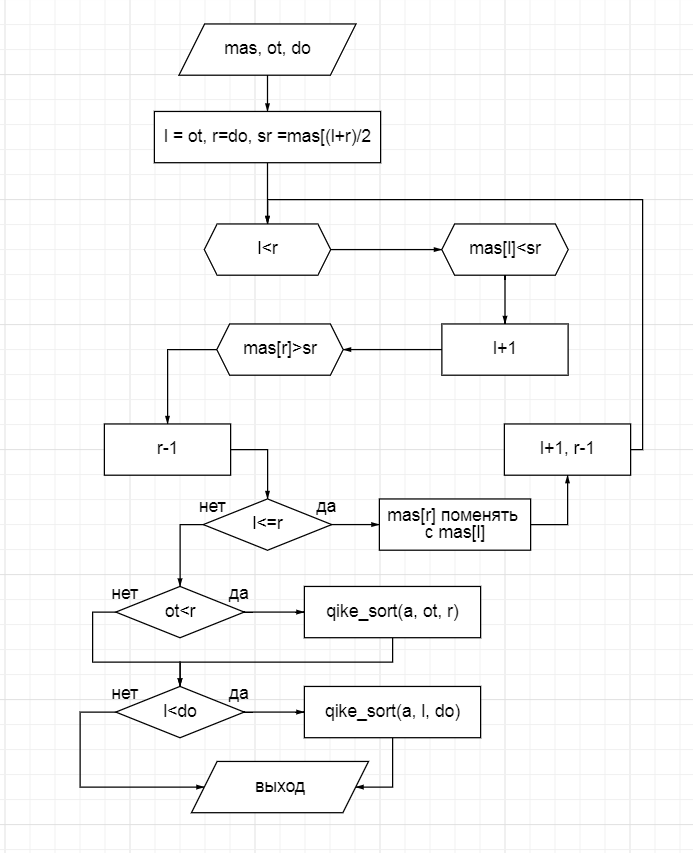


Рис. 11 Блок-схема алгоритма быстрой сортировки

# Эксперименты

Главным критерием выбора любой сортировки является время ее работы на больших по объему данных. Для сравнения времени работы реализованных в данной программе сортировок создадим 3 файла, с увеличение элементов в каждом в 10 раз по сравнению с предыдущим, а именно, первый файл на 1000 чисел, второй на 10000 чисел и третий на 100000 чисел. После чего запустим каждый алгоритм сортирования и измерим его время работы.

|  |  |  |  |
| --- | --- | --- | --- |
| Количество элементов в файле | Пузырек | Вставка | Быстрая |
| 1000 | 0.004 с | 0.002 с | 0.000000с |
| 10000 | 0.246 с | 0.111 с | 0.002 с |
| 100000 | 25.184 | 9.601 с | 0.024 с |

Таблица 1. Результаты экспериментов

Как видно из таблицы, при увеличении количества сортируемых чисел в 10 раз, Пузырек и Вставка увеличивают свое время работы почти в 100 раз, что соответствует их сложности O(N^2), быстрая же сортировка увеличивает время работы в промежутке между 10 раз и 20 раз. Таким образом, для маленьких объемов возможно использование сортировок вставкой и быстрая, для больших же объемов необходимо применять быструю сортировку, но ее алгоритм является более сложным в реализации.

# Заключение

В результате проведенной работы были написаны две программы, которые полностью выполнили поставленные перед ними задачи, а именно написать программу, которая состоит из двух подпрограмм. Первая программа создает текстовый файл с записанными в него числами. Программа принимает количество чисел n, максимальное и минимальное значение, также программа может создавать массив не только из случайных чисел, но и получать их с клавиатуры и из файла. Вторая программа читает текстовый файл с набором чисел, выводит консольный интерфейс (печать, сортировка, сброс, выход), выполняет выбранные действия. Реализовать три сортировки: пузырьком, вставкой, быстрая.

После проведенных экспериментов был сделан вывод о необходимости и возможности использования той или иной сортировки для разных исходных данных.

Умение представить данные в упорядоченном виде, путем применения сортировки значительно облегчает дальнейшую работу с данными, поэтому умения, полученные во время выполнения работы окажутся полезными.

# Литература

1. Динамическое выделение памяти в Си // Програмирование С и С++ – Режим доступа: <https://prog-cpp.ru/c-alloc/>
2. [OldNileCrocodile](https://habr.com/ru/users/OldNileCrocodile/) Кратко об указателях в Си: присваивание, разыменование и перемещение по массивам // Хабр – Режим доступа: <https://habr.com/ru/post/545674/>
3. Пользовательские функции в Си // Young Coder – Режим доступа: https://youngcoder.ru/lessons/10/polzovatelskie\_funkcii.php

# Приложение 1

Заголовочный файл «Header\_labs\_1f.h»

#ifndef \_\_Header\_labs\_1\_h\_\_

#define \_\_Header\_labs\_1\_h\_\_

void rand\_in\_f(FILE\* f);

void polzovat\_in\_f(FILE\* f);

void f\_in\_f(FILE\* f);

#endif

Исходный файл «def\_labs\_1f.c»

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include "Header\_labs\_1f.h"

#include <locale.h>

void rand\_in\_f(FILE \*f)

{

int N = 0, min = 0, max = 0;

int i = 0;

fopen\_s(&f, "..\\text1\_fr.txt", "w");

printf("Введите количество чисел: ");

scanf\_s("%d", &N);

printf("min = ");

scanf\_s("%d", &min);

printf("max = ");

scanf\_s("%d", &max);

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

{

int a = 0;

a = (int)(((double)rand() / RAND\_MAX) \* (max - min) + min);

fprintf(f, "%d\n", a);

}

fclose(f);

}

void polzovat\_in\_f(FILE\* f)

{

int n = 0;

int i = 0;

fopen\_s(&f, "..\\text1\_fr.txt", "w");

printf("Количество чисел :");

scanf\_s("%d", &n);

for (i = 0; i < n; i++)

{

int a = 0;

scanf\_s("%d", &a);

fprintf(f, "%d\n", a);

}

fclose(f);

}

void f\_in\_f(FILE\* f)

{

FILE\* frc = 0;

printf("Введите путь к файлу для чтения: ");

scanf\_s("\n");

char\* pkfr;

int i = 1;

pkfr = (char\*)malloc(256);

fgets(pkfr, 255, stdin);

pkfr[strlen(pkfr) - 1] = 0;

fopen\_s(&frc, pkfr, "r");

fopen\_s(&f, "..\\text1\_fr.txt", "w");

while (!feof(frc))

{

int a = 0;

fscanf\_s(frc, "%d\n", &a);

fprintf(f, "%d\n", a);

}

fclose(f);

fclose(frc);

free(pkfr);

}

Исходный файл «main\_labs\_1f.c»

#include<stdio.h>

#include<stdlib.h>

#include <string.h>

#include "Header\_labs\_1f.h"

#include <locale.h>

int main()

{

setlocale(LC\_ALL, "Russian");

FILE\* fr = 0;

int tmp = 0;

printf("Выберети источник данных:\n1 - случайные числа\n2 - с клавивтуры\n3 - из файла\n");

scanf\_s("%d", &tmp);

if (tmp == 1)

{

rand\_in\_f(fr);

}

else if (tmp == 2)

{

polzovat\_in\_f(fr);

}

else if (tmp == 3)

{

f\_in\_f(fr);

}

return 0;

}

# Приложение 2

Заголовочный файл «Header\_labs\_2f.h»

#ifndef \_\_Header\_labs\_2f\_h\_\_

#define \_\_Header\_labs\_2f\_h\_\_

void Puzirek(int\* a, int len\_a);

void Vstavka(int\* a, int len\_a);

void qike\_sort(int\* a, int ot\_e, int do\_e);

void f\_in\_mas(FILE\* f, int\* d, int\*\* a);

#endif

Исходный файл «def\_labs\_2f.c»

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include "Header\_labs\_2f.h"

#include <locale.h>

void Puzirek(int\* a, int len\_a)

{

int i = 0, g = 0, tmp = 0;

for (i = 0; i < len\_a; i++)

{

for (g = 0; g < (len\_a - i) - 1; g++)

{

if (a[g] > a[g + 1])

{

tmp = a[g];

a[g] = a[g + 1];

a[g + 1] = tmp;

}

}

}

}

void Vstavka(int\* a, int len\_a)

{

int i = 0, g = 1, tmp = 0;

for (i = 0; i < len\_a; i++)

{

g = i;

while (g > 0 && (a[g] < a[g - 1]))

{

tmp = a[g - 1];

a[g - 1] = a[g];

a[g] = tmp;

g--;

}

}

}

void qike\_sort(int\* a, int ot\_e, int do\_e)

{

int l = ot\_e;

int r = do\_e;

int sr = a[(int)((l + r) / 2)];

do

{

while (a[l] < sr)

l++;

while (a[r] > sr)

r--;

if (l <= r)

{

int tmp = 0;

tmp = a[r];

a[r] = a[l];

a[l] = tmp;

l++;

r--;

}

} while (l < r);

if (ot\_e<r)

qike\_sort(a, ot\_e, r);

if (l<do\_e)

qike\_sort(a, l, do\_e);

}

void f\_in\_mas(FILE \* f, int \*d, int\*\* a)

{

int i = 0;

fopen\_s(&f, "..\\text1\_fr.txt", "r");

while (!feof(f))

{

int v = 0;

fscanf\_s(f, "%d\n", &v);

(\*d)++;

}

freopen\_s(&f , "..\\text1\_fr.txt", "r", f);

\*a = (int\*)malloc(\*d \* sizeof(int));

for (i = 0; i < \*d; i++)

{

int c = 0;

fscanf\_s(f, "%d\n", &c);

(\*a)[i] = c;

}

fclose(f);

}

Исходный файл «main\_labs\_2f.c»

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include "Header\_labs\_2f.h"

#include <locale.h>

#include < time.h>

int main()

{

setlocale(LC\_ALL, "Russian");

int\* mas;

int ks = 0;

int ks\_n = 0;

int i = 0;

clock\_t t1, t2;

FILE\* fr = 0;

f\_in\_mas(fr, &ks, &mas);

ks\_n = ks;

while(1)

{

int tmp = 0;

printf("Выберите действие:\n1-печать\n2-сортировка\n3-сброс\n4-выход\n");

scanf\_s("%d", &tmp);

if (tmp == 1)

{

printf("\n");

for (i = 0; i < ks; i++)

{

printf("%d\n", mas[i]);

}

printf("\n");

}

else if (tmp == 2)

{

int tmp\_2 = 0;

printf("Выберите сортировку:\n1 - Пузырьком\n2 - Вставкой\n3 - Быстрая\n");

scanf\_s("%d", &tmp\_2);

if (tmp\_2 == 1)

{

t1 = clock();

Puzirek(mas, ks);

t2 = clock();

}

if (tmp\_2 == 2)

{

t1 = clock();

Vstavka(mas, ks);

t2 = clock();

}

if (tmp\_2 == 3)

{

t1 = clock();

qike\_sort(mas, 0, ks-1);

t2 = clock();

}

printf("\nВремя работы сортировки:%lf\n", (double)(t2 - t1) / CLOCKS\_PER\_SEC);

}

else if (tmp == 3)

{

f\_in\_mas(fr, &ks\_n, &mas);

ks\_n = ks;

}

else if (tmp == 4)

{

int i = 0;

fopen\_s(&fr, "..\\text1\_fr.txt", "w");

for (i = 0; i < ks; i++)

{

fprintf(fr, "%d\n", mas[i]);

}

fclose(fr);

free(mas);

break;

}

}

return 0;

}