**Содержание**

[**Введение 2**](#_Toc107826785)

[**1 Описание алгоритма 4**](#_Toc107826786)

[**1.1 Достоинства алгоритма быстрой сортировки: 5**](#_Toc107826787)

[**1.2 Недостатки алгоритма: 5**](#_Toc107826788)

[**2 Описание программы 5**](#_Toc107826789)

[**3 Блок-схемы работы программы 12**](#_Toc107826790)

[**4 Заключение 18**](#_Toc107826791)

[**Список литературы 19**](#_Toc107826792)

[**Приложение 20**](#_Toc107826793)

# Введение

Microsoft Visual Studio — это программная средапоразработке 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](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D1%84%D0%B0%D0%BA%D1%82%D0%BE%D1%80%D0%B8%D0%BD%D0%B3);

2)отладчик кода;

3)редактор форм, предназначенный для упрощённого конструирования графических интерфейсов;

4)веб-редактор;

5)дизайнер классов;

6)дизайнерсхем баз данных.

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. VisualBasic.NET – для разработки приложений на VisualBasic;
2. VisualC++ – на традиционном языке C++;
3. VisualC# – наязыкеC# (Microsoft);
4. VisualF# – наF# (Microsoft Developer Division).

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

Ниже перечислены основные преимущества IDE-среды Visual Studio:

1. Встроенный Web-сервер;
2. Поддержка множества языков при разработке;
3. Интуитивный стиль кодирования;
4. Более высокая скорость разработки;
5. Возможности отладки.

Язык программирования С— компилируемый статически типизированный язык программирования общего назначения, разработанный в 1969—1973 годах сотрудником Bell Labs Деннисом Ритчи как развитие языка Би.

Первоначально был разработан для реализации операционной системы UNIX, но впоследствии был перенесён на множество других платформ. Согласно дизайну языка, его конструкции близко сопоставляются типичным машинным инструкциям, благодаря чему он нашёл применение в проектах, для которых был свойственен язык ассемблера, в том числе как в операционных системах, так и в различном прикладном программном обеспечении для множества устройств — от суперкомпьютеров до встраиваемых систем. Язык программирования Си оказал существенное влияние на развитие индустрии программного обеспечения, а его синтаксис стал основой для таких языков программирования, как C++, C#, Java и Objective-C.

Язык Си разрабатывался как язык системного программирования, для которого можно создать однопроходный компилятор. Стандартная библиотека также невелика. Как следствие данных факторов — компиляторы разрабатываются сравнительно легко. Поэтому данный язык доступен на самых различных платформах. К тому же, несмотря на свою низкоуровневую природу, язык ориентирован на переносимость. Программы, соответствующие стандарту языка, могут компилироваться под различные архитектуры компьютеров.

Целью языка было облегчение написания больших программ с минимизацией ошибок по сравнению с ассемблером, следуя принципам процедурного программирования, но избегая всего, что может привести к дополнительным накладным расходам, специфичным для языков высокого уровня.

# 1 Описание алгоритма

Сортировка слиянием (Mergesort) — алгоритм сортировки, который упорядочивает списки (или другие структуры данных, доступ к элементам которых можно получать только последовательно, например — потоки) в определённом порядке. Эта сортировка — хороший пример использования принципа «разделяй и властвуй». Сначала задача разбивается на несколько подзадач меньшего размера. Затем эти задачи решаются с помощью рекурсивного вызова или непосредственно, если их размер достаточно мал. Наконец, их решения комбинируются, и получается решение исходной задачи.

Эффективно заранее создать временный массив и передать его в качестве аргумента функции. Эта сортировка рекурсивна, а потому возможен переход на квадратичную при небольшом числе элементов.

Общая идея алгоритма состоит в следующем:

1. Сортируемый массив разбивается на две части примерно одинакового размера;
2. Каждая из получившихся частей сортируется отдельно, например — тем же самым алгоритмом;
   1. . Рекурсивное разбиение задачи на меньшие происходит до тех пор, пока размер массива не достигнет единицы (любой массив длины 1 можно считать упорядоченным).
3. Два упорядоченных массива половинного размера соединяются в один.
   1. . Слияние двух подмассивов в третий результирующий массив.
   2. . «Прицепление» остатка. (Когда один из подмассивов закончился, добавляются все оставшиеся элементы второго подмассива в результирующий массив.)

## 1.1 Достоинства алгоритма быстрой сортировки:

* Работает даже на структурах данных последовательного доступа.
* Неплохо работает в параллельном варианте: легко разбить задачи между процессорами поровну, но трудно сделать так, чтобы другие процессоры взяли на себя работу, в случае если один процессор задержится.
* Не имеет «трудных» входных данных.
* Устойчивая - сохраняет порядок равных элементов (принадлежащих одному классу эквивалентности по сравнению).

### 1.2 Недостатки алгоритма:

* На «почти отсортированных» массивах работает столь же долго, как на хаотичных. Существует вариант сортировки слиянием, который работает быстрее на частично отсортированных данных, но он требует дополнительной памяти, в дополнении ко временному буферу, который используется непосредственно для сортировки.
* Требует дополнительной памяти по размеру исходного массива.

# 2 Описание программы

В программе для сортировки слияниемподключены следующие заголовочные файлы:*stdio.h*– заголовочный файл с классами, функциями и переменными для организации ввода-вывода; *Windows.h*– специфичный заголовочный файл, необходимый для использования в программе функционала, предоставляемого операционной системой Windows;*time.h*– заголовочный файл стандартной библиотеки C++, содержащий типы и функции для работы с датой и временем;.*conio.h*– заголовочный файл, используемый в старых компиляторах, работающих в операционных системах MS-DOS, для создания текстового интерфейса пользователя.

#pragmawarning(disable : 4996)

#include"menu.h"

#include<time.h>

#include<conio.h>

#include<stdio.h>

#include<Windows.h>

Далееидеттекстфункциисортировки слияниями Merge и MergeSort.

Функция Merge включаетв себя разделение массива на два подмассива. Послесортировкидвухподмассивов соединяет снова в один массив.

voidMerge(int\* Ap, intp, intq, intr)

{

int n1 = q - p + 1;

int n2 = r - q;

int\* L; int\* R;

L = newint[n1 + 1];

R = newint[n2 + 1];

for (inti = 1; i<= n1; i++)

{

L[i-1] = Ap[p + i - 1];

}

for (int j = 1; j <= n2; j++)

{

R[j-1] = Ap[q + j];

}

for (inti = 0, j = 0, k = p; k <= r; k++) {

if (i< n1 && j < n2) {

if (L[i] <= R[j]) {

Ap[k] = L[i];

i++;

}

else {

Ap[k] = R[j];

j++;

}

}

elseif (i< n1) {

Ap[k] = L[i];

i++;

}

else {

Ap[k] = R[j];

j++;

}

}

}

MergeSort включает в себя рекурсивную функцию, которая выполняет

основную функцию сортировки слияниями.

Рекурсивное разбиение задачи наменьшие происходит до тех пор, покаразмер массива не достигнет единицы (любой массив длины 1 можно считать упорядоченным).

voidMerge\_Sort(int\* Ap, intp, intr)

{

int q;

if (p<r)

{

q = (p + r) / 2;

Merge\_Sort(Ap, p, q);

Merge\_Sort(Ap, q + 1, r);

Merge(Ap, p, q, r);

}

}

Функция *save 1*и *save2* представляет из себя сохранение результата сортировки случайного набора чисел в произвольно названные файлы.

*save1* – исходныйнаборслучайныхчисел.

void save1(FILE\* F1, char\* F1name, int\* A, intsize)

{

F1 = fopen(F1name, "w");

if (F1 != NULL)

{

for (inti = 0; i<size; i++)

{

fprintf(F1, "%d\n", A[i]);

}

}

else

{

printf("Ошибка записи в файл");

}

fclose(F1);

}

*save2* – предназначена для сохранения отсортированного массива случайного набора чисел.

void save2(FILE\* F2, char\* F2name, int\* A, intsize)

{

F2 = fopen(F2name, "w");

if (F2 != NULL)

{

for (inti = 0; i<size; i++)

{

fprintf(F2, "%d\n", A[i]);

}

}

else

{

printf("Ошибка записи в файл");

}

fclose(F2);

}

Функцией *random* создается массив из случайного набора чисел, который сохраняется в файл с произвольным названием, после сохранения данный массив сортируется и массив отсортированных чисел сохраняется в другой файл,также с произвольным именем.

voidrandom(void)

{

int size;

FILE\* F1, \* F2;

char F1name[20];

char F2name[20];

printf("Введите файл, в котором будут находиться исходные числа: ");

gets\_s(F1name);

F1 = fopen(F1name, "w");

printf("Введите файл, в котором будут находиться отсортированные числа: ");

gets\_s(F2name);

F2 = fopen(F2name, "w");

printf("Введите размер массива (больше 1): ");

do

{

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

if (size< 2)

{

printf("Неверно ввёден размер массива\n");

printf("Введите размер массива (больше 1): ");

}

} while (size < 2);

int\* A = newint[size];

for (int i = 0; i <size; i++) //Заполнение массива А случайными числами

{

A[i] = (rand() % 20001) - 10000;

}

printf("Исходныечиславфайле \"%s\" \n", F1name);

save1(F1, F1name, A, size);

time\_t start = clock();

Merge\_Sort(A, 0, (size - 1));

time\_t stop = clock();

double time = (stop - start) / 1000.0;

printf("Время сортировки: %f секунд\n", time);

save2(F2, F2name, A, size);

printf("Отсортированные числа в файле \"%s\" \n", F2name);

}

В функция *sort* мы выбираем файл, в котором содержится сортированные числа в порядке возрастания, либо сортированные в порядке убывания.

Послевыбора файла и считывания чисел производится сортировка. Вконце работы функции готовый ответ сохраняется в файл с произвольным названием.

voidsort(void)

{

FILE \*F3,\* F4, \*F5, \*F6;

char F3name[20], F4name[20];

printf("Введитеназваниефайла: ");

gets\_s(F3name);

F3 = fopen(F3name, "r");

if (F3 == NULL)

{

printf("Невозможно открыть файл \"otsort.txt\"\n");

}

else

{

int size = 0;

intcurDigit = 0;

while (fscanf(F3, "%d", &curDigit) != EOF) {

size++;

}

fseek(F3, 0, SEEK\_SET);

int\* A = newint [size];

for (inti = 0; i< size; i++)

{

fscanf(F3, "%d", &A[i]);

}

fclose(F3);

time\_t start = clock();

Merge\_Sort(A, 0, size-1); //(sizeof(size) / sizeof(int) - 1)

time\_t stop = clock();

double time = (stop - start) / 1000.0;

printf("Время сортировки: %f секунд\n", time);

printf("Напишите, в какой файл сохранить отсортированные файлы (текстовый файл!): ");

gets\_s(F4name);

F4 = fopen(F4name, "w");

if (F4 != NULL)

{

for (inti = 0; i< size; i++)

{

fprintf(F4, "%d\n", A[i]);

}

}

else

{

printf("Ошибка записи в файл");

}

fclose(F4);

}

}

Функция *main* служит для вывода меню и выбора нужной сортировки, а также подключение русификации.

int main(void)

{

SetConsoleCP(1251); //русификация

SetConsoleOutputCP(1251);

srand(time(NULL));

int s = 1;

while (s != 0)

{

menu();

s = \_getch();

switch (s)

{

case'1': system("cls"); sort(); break;

case'2': system("cls"); random(); break;

case'3': system("cls"); exit(0); break;

default: system("cls");

}

}

return 0;

}

# 3 Блок-схемы работы программы

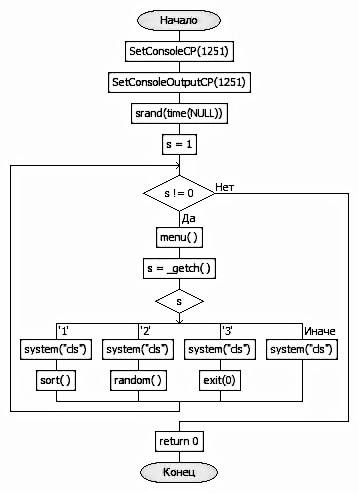
****

Рисунок 1 – Блок-схема функции main

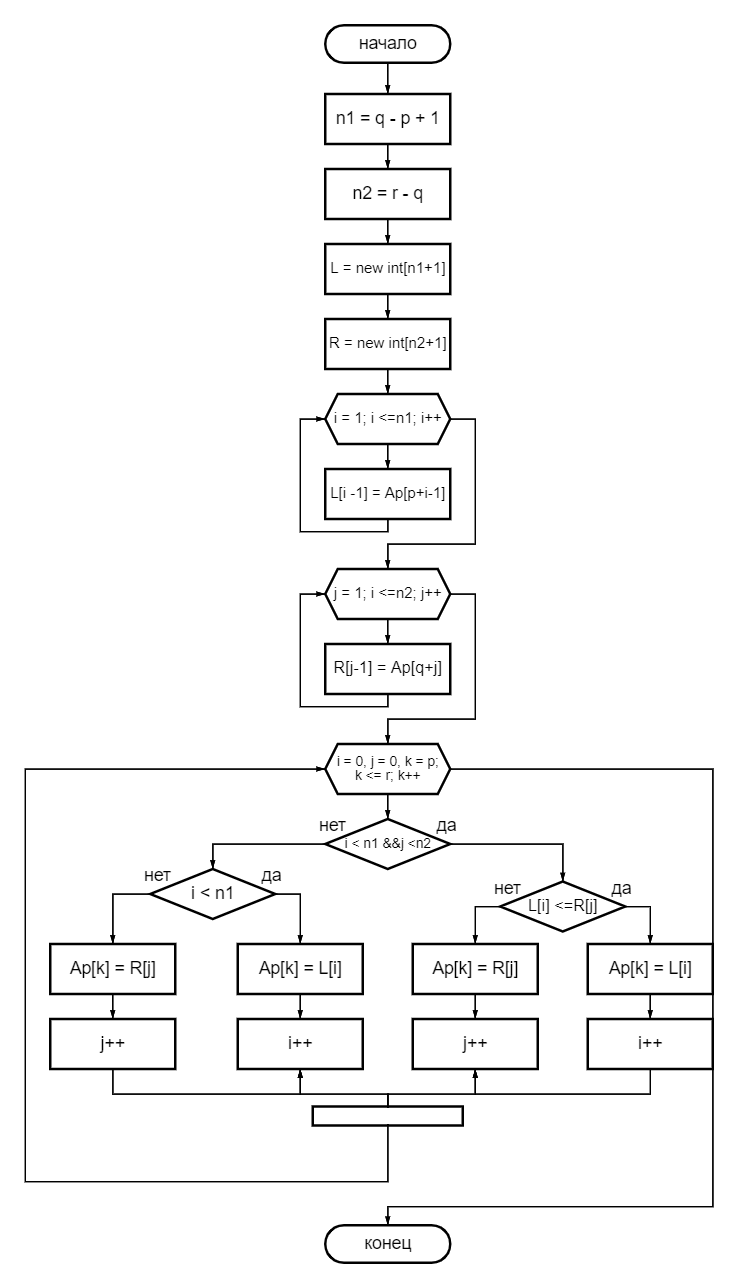
****

Рисунок 2 – Блок-схема функции Merge

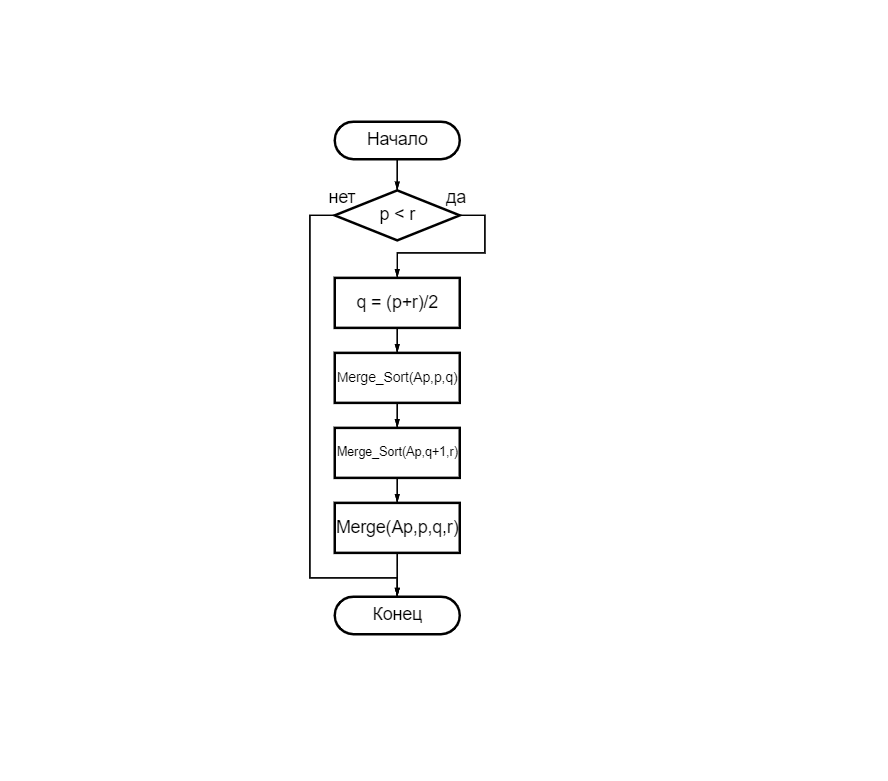


Рисунок 3 – Блок-схема функции MergeSort

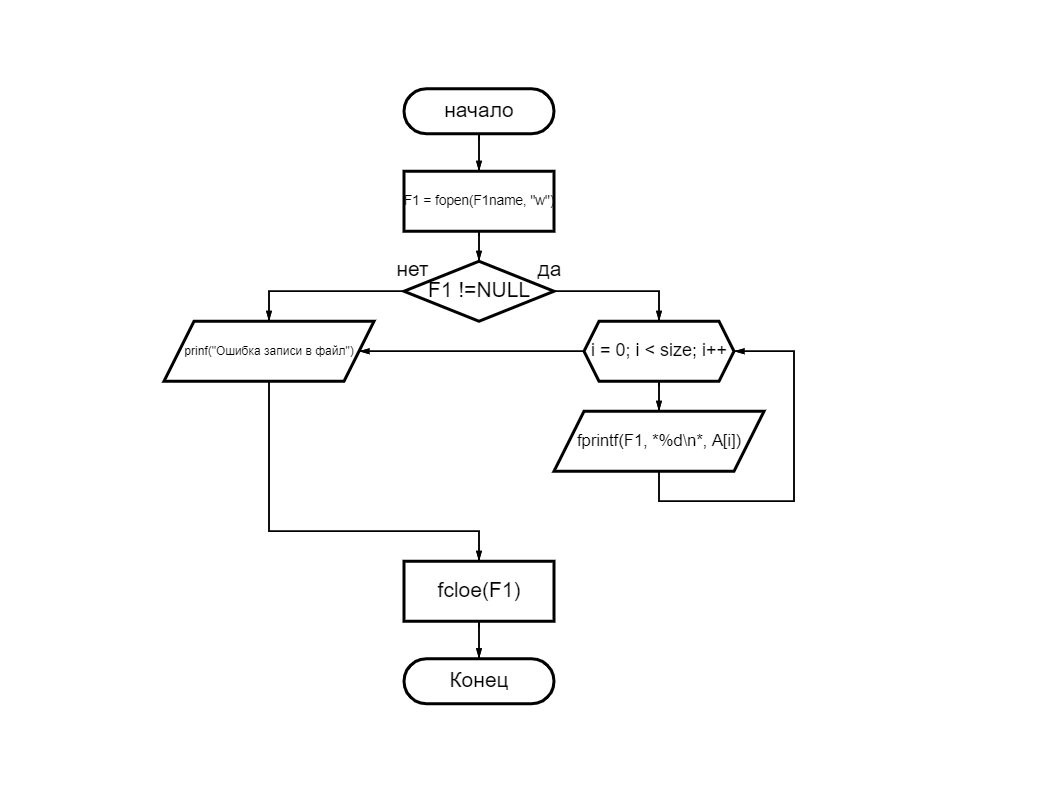
****

Рисунок 4 – Блок-схема функции save1

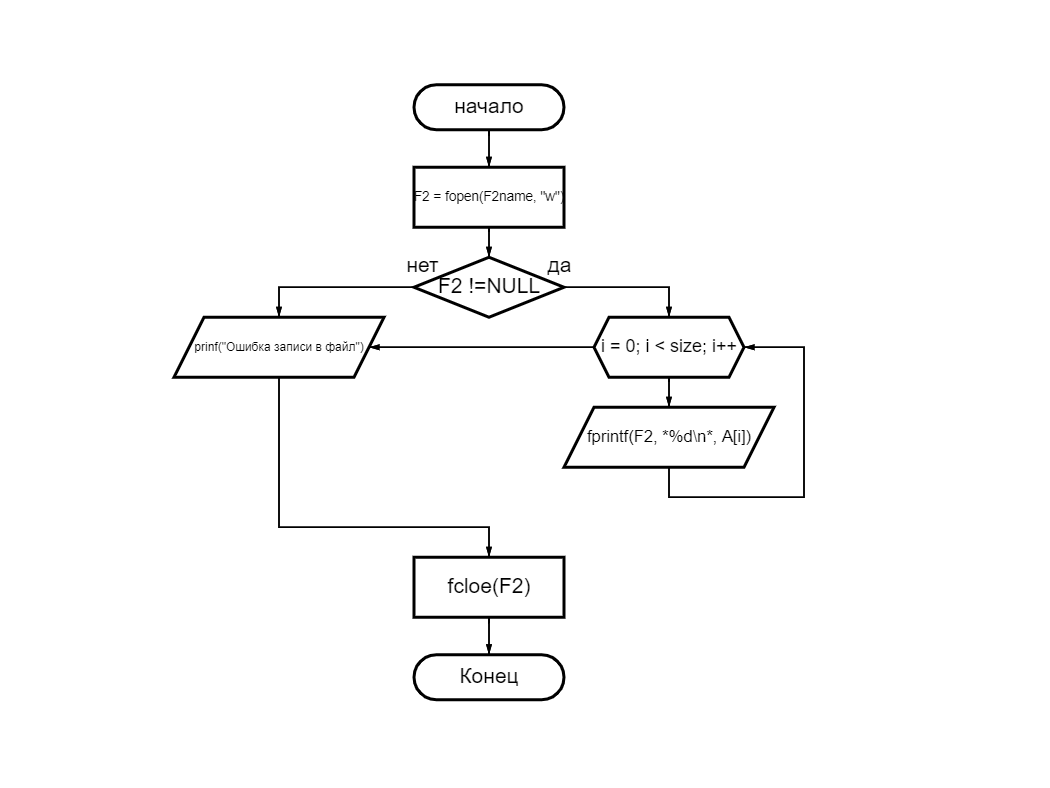
****

Рисунок 5 – Блок-схема функции save2

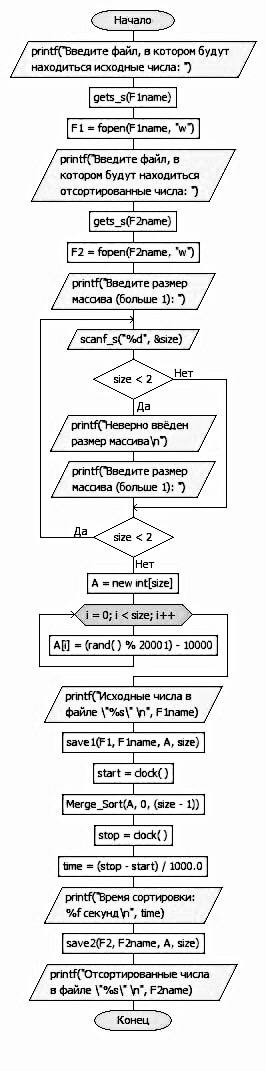
****

Рисунок 6 – Блок-схема функции random

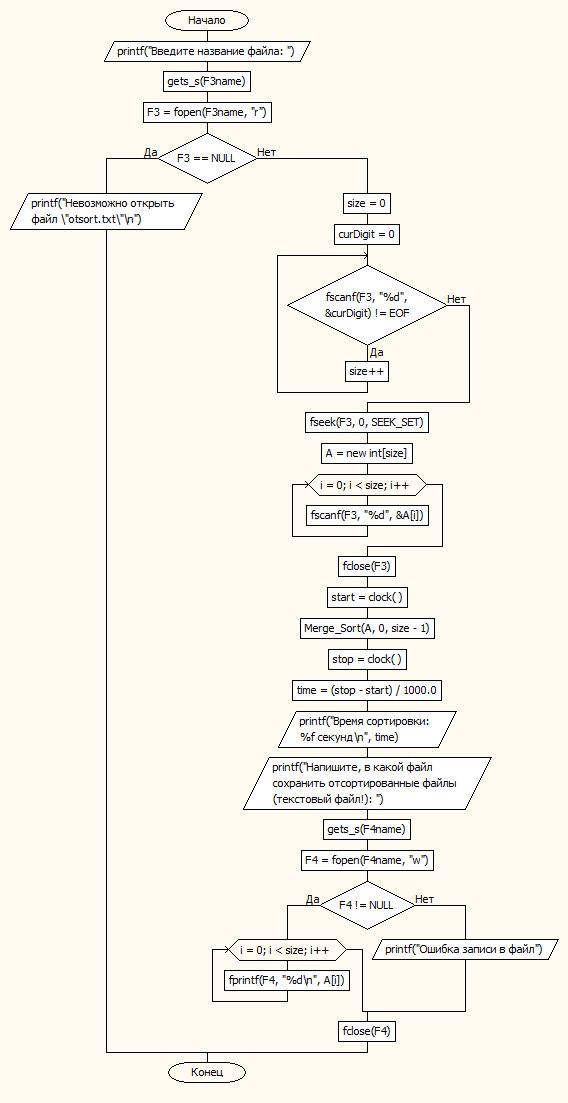


Рисунок 7 – Блок-схема функции sort

# 4 Заключение

В результате проделанной работы была разработана программа, с помощью которой, можно отсортировать числа в порядке возрастания используя метод сортировки слиянием.

# Список литературы

1. Красиков И.В. – Алгоритмы. Просто как дважды два. 2007 г.

2. Дейтел и Пол Харви. Как программировать на С.

3.<https://en.wikipedia.org/wiki/Merge_sort> - Википедия. Сортировка слиянием.

4. <https://www.cyberforum.ru/cpp-beginners/thread2411190.html> - Киберфорум. Сортировка MergeSort.

5. <https://habr.com/ru/company/edison/blog/431964/> - Хабр.

# Приложение

**Листинг программы**

#pragmawarning(disable : 4996)

#include"menu.h"

#include<time.h>

#include<conio.h>

#include<stdio.h>

#include<Windows.h>

void Merge(int\* Ap, intp, intq, intr)

{

int n1 = q - p + 1;

int n2 = r - q;

int\* L; int\* R;

L = newint[n1 + 1];

R = newint[n2 + 1];

for (int i = 1; i <= n1; i++)

{

L[i-1] = Ap[p + i - 1];

}

for (int j = 1; j <= n2; j++)

{

R[j-1] = Ap[q + j];

}

for (int i = 0, j = 0, k = p; k <= r; k++) {

if (i < n1 && j < n2) {

if (L[i] <= R[j]) {

Ap[k] = L[i];

i++;

}

else {

Ap[k] = R[j];

j++;

}

}

elseif (i < n1) {

Ap[k] = L[i];

i++;

}

else {

Ap[k] = R[j];

j++;

}

}

}

void Merge\_Sort(int\* Ap, intp, intr)

{

int q;

if (p<r)

{

q = (p + r) / 2;

Merge\_Sort(Ap, p, q);

Merge\_Sort(Ap, q + 1, r);

Merge(Ap, p, q, r);

}

}

void save1(FILE\* F1, char\* F1name, int\* A, intsize)

{

F1 = fopen(F1name, "w");

if (F1 != NULL)

{

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

{

fprintf(F1, "%d\n", A[i]);

}

}

else

{

printf("Ошибка записи в файл");

}

fclose(F1);

}

void save2(FILE\* F2, char\* F2name, int\* A, intsize)

{

F2 = fopen(F2name, "w");

if (F2 != NULL)

{

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

{

fprintf(F2, "%d\n", A[i]);

}

}

else

{

printf("Ошибка записи в файл");

}

fclose(F2);

}

void random(void)

{

int size;

FILE\* F1, \* F2;

char F1name[20];

char F2name[20];

printf("Введите файл, в котором будут находиться исходные числа: ");

gets\_s(F1name);

F1 = fopen(F1name, "w");

printf("Введите файл, в котором будут находиться отсортированные числа: ");

gets\_s(F2name);

F2 = fopen(F2name, "w");

printf("Введите размер массива (больше 1): ");

do

{

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

if (size < 2)

{

printf("Неверно ввёден размер массива\n");

printf("Введите размер массива (больше 1): ");

}

} while (size < 2);

int\* A = newint[size];

for (int i = 0; i < size; i++) //Заполнение массива А случайными числами

{

A[i] = (rand() % 20001) - 10000;

}

printf("Исходные числа в файле \"%s\" \n", F1name);

save1(F1, F1name, A, size);

time\_t start = clock();

Merge\_Sort(A, 0, (size - 1));

time\_t stop = clock();

int time = (stop - start) / 1000;

printf("Время сортировки: %i мс\n", time);

save2(F2, F2name, A, size);

printf("Отсортированные числа в файле \"%s\" \n", F2name);

}

void sort(void)

{

FILE \*F3,\* F4, \*F5, \*F6;

char F3name[20], F4name[20];

printf("Введите название файла: ");

gets\_s(F3name);

F3 = fopen(F3name, "r");

if (F3 == NULL)

{

printf("Невозможно открыть файл \"%s\"\n", &F3name);

}

else

{

int size = 0;

int curDigit = 0;

while (fscanf(F3, "%d", &curDigit) != EOF) {

size++;

}

fseek(F3, 0, SEEK\_SET);

int\* A = newint [size];

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

{

fscanf(F3, "%d", &A[i]);

}

fclose(F3);

time\_t start = clock();

Merge\_Sort(A, 0, size - 1);

time\_t stop = clock();

int time = (stop - start) / 1000;

printf("Время сортировки: %i мс\n", time);

printf("Напишите, в какой файл сохранить отсортированные файлы (текстовый файл!): ");

gets\_s(F4name);

F4 = fopen(F4name, "w");

if (F4 != NULL)

{

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

{

fprintf(F4, "%d\n", A[i]);

}

}

else

{

printf("Ошибка записи в файл");

}

fclose(F4);

}

}

int main(void)

{

SetConsoleCP(1251); //русификация

SetConsoleOutputCP(1251);

srand(time(NULL));

int s = 1;

while (s != 0)

{

menu();

s = \_getch();

switch (s)

{

case'1': system("cls"); sort(); break;

case'2': system("cls"); random(); break;

case'3': system("cls"); exit(0); break;

default: system("cls");

}

}

return 0;

}