**Московский авиационный институт (национальный исследовательский университет)**

Институт №3. «Системы управления, информатика и электроэнергетика». Кафедра №304

Практическое задание

Группа М30-107Б-18

Выполнили:

Кривонос Александр

Принял:

Чечиков Юрий Борисович

Москва 2018

**Содержание**

1. Задание………………………………………………………………………………….2

2. Блок-схема алгоритма…………………………………………………………………3-9

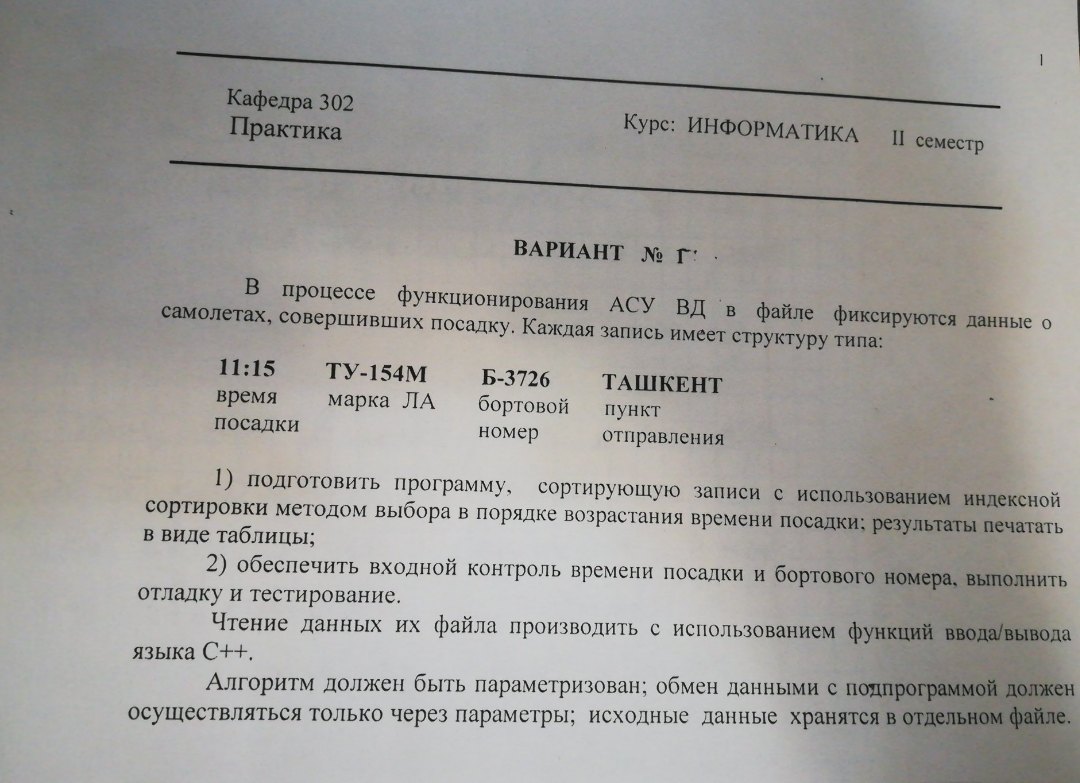
3. Псевдокод……………………………………………………………………………....10-14

4. Текст программы………………………………………………………………………15-18

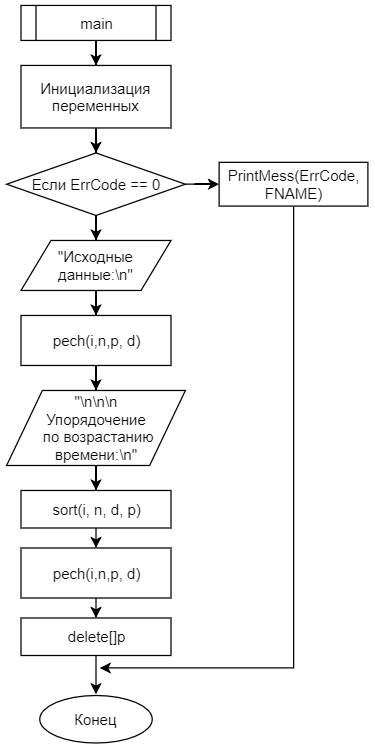
5. Тест (некорректный и корректный)………………………………………………….19-22

6. Вывод…………………………………………………………………………………..23

**Задание**



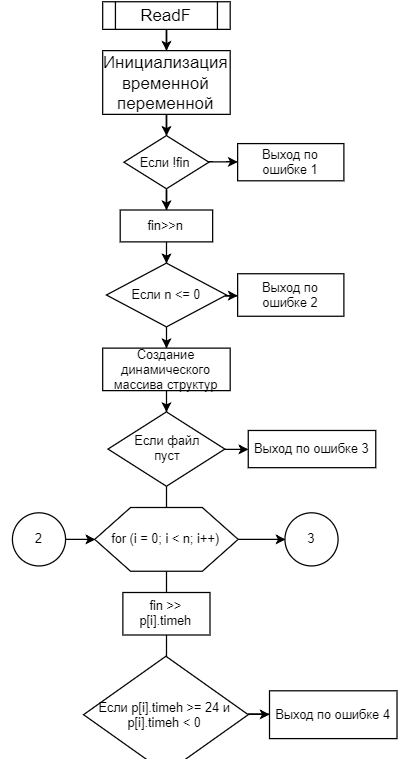
**Блок-схема алгоритма**

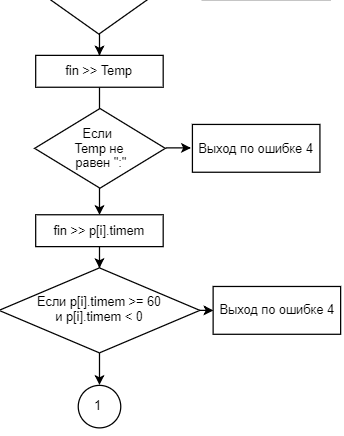


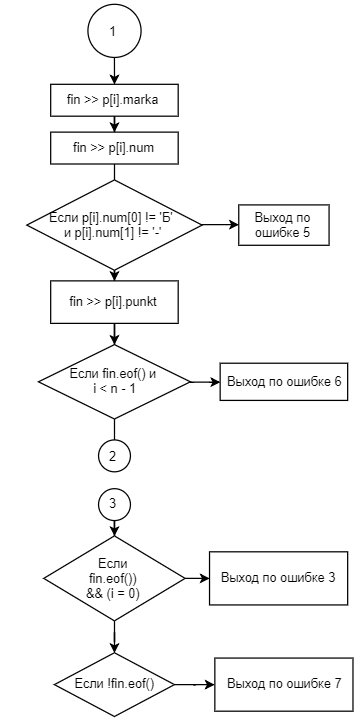
**Описание функций:**

1. ReadF
   1. Назначение: считывает данные и проверяет их корректность
   2. Прототип: int ReadF (int &n, int &i, info \*&p, dop \*&d, const char\* FNAME)
   3. Обращение: ReadF (n, i, p, d, FNAME)
   4. Описание параметров: возвращает код ошибки

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Параметр | &n | &i | \*&p | \*&d | \* FNAME |
| Тип | int | int | info | dop | const char |
| Назначение | количество структур | индекс для массива структур | массив структур | Дополнительный массив структур | файловая переменная |
| Входной/выходной | входной | входной | входной | входной | входной |

* 1. Блок-схема



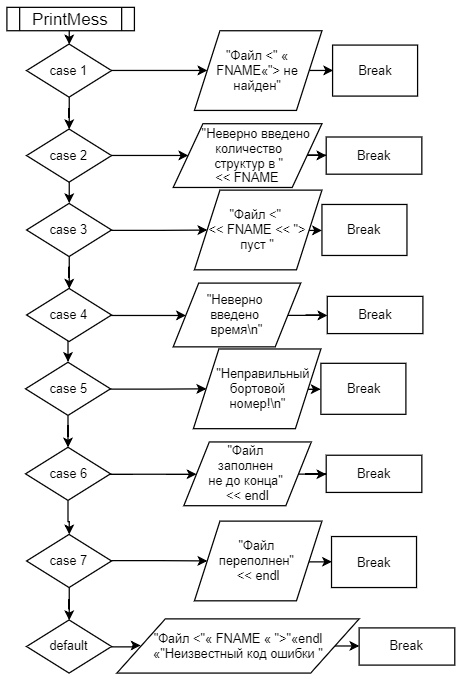


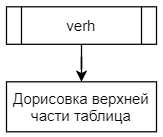
1. PrintMess
   1. Назначение: вывод сообщения об ошибке
   2. Прототип: void PrintMess(int ErrCode, const char\* FNAME)
   3. Обращение: PrintMess (ErrCode, FNAME)
   4. Описание параметров:

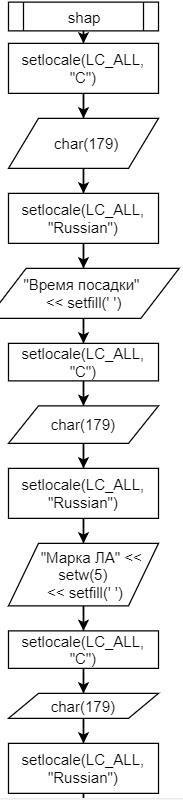
int ErrCode – код ошибки

const char \*FNAME – файловая переменная

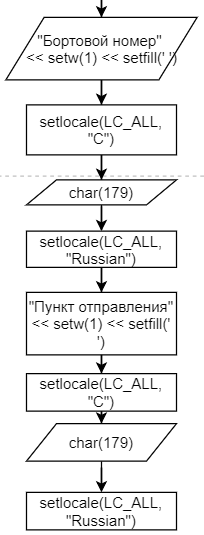
* 1. Блок-схема

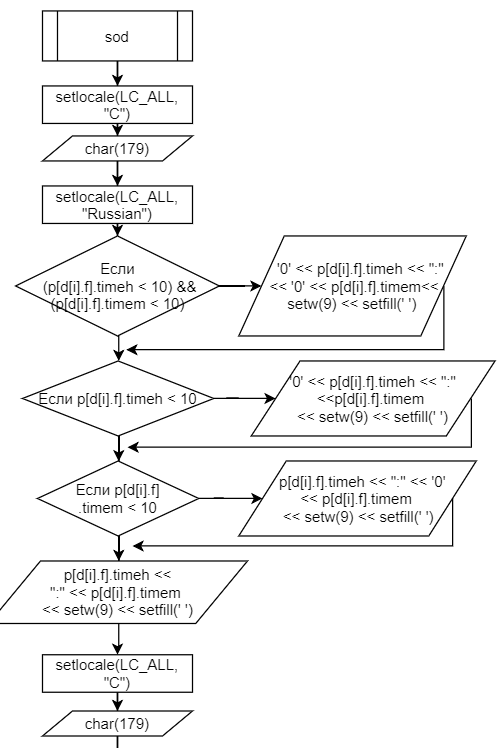


1. verh
   1. Назначение: дорисовка верхней части таблицы
   2. Прототип: void verh( )
   3. Обращение: verh ( )
   4. Блок-схема



1. shap
   1. Назначение: выводит шапку таблицы
   2. Прототип: void shap( )
   3. Обращение: shap ( )
   4. Блок-схема

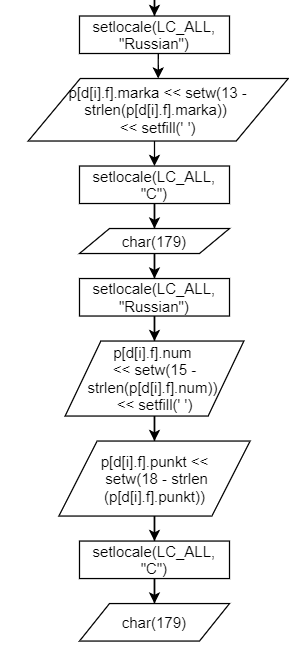


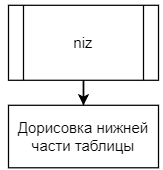
1. sod
   1. Назначение: выводит содержание таблицы
   2. Прототип: void sod(int &i, dop \*&d info \*p)
   3. Обращение: void sod(I, d, p)
   4. Описание параметров: не возвращает параметров,

так как тип void

|  |  |  |  |
| --- | --- | --- | --- |
| Параметр | &i | \*&p | \*&d |
| Тип | int | info | dop |
| Назначение | индекс для массива структур | массив структур | Дополнительный массив структур |
|  | входной | входной | входной |

* 1. Блок-схема

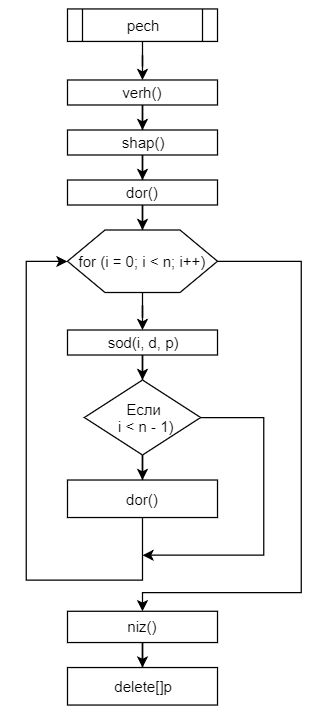


1. niz
   1. Назначение: дорисовка нижней части таблицы
   2. Прототип: void niz( )
   3. Обращение: niz ( )
   4. Блок-схема



1. dor
   1. Назначение: дорисовка ячеек таблицы
   2. Прототип: void dor( )
   3. Обращение: dor ( )
   4. Блок-схема
2. pech
   1. Назначение: печатает таблицу с содержимым
   2. Прототип: void pech(int &i, int &n, dop \*&d, info \*&p)
   3. Обращение: void pech(i, n, d, p)
   4. Описание параметров: не возвращает параметров, так как тип void

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Параметр | &n | &i | \*&p | \*&d |
| Тип | int | int | info | dop |
| Назначение | количество структур | индекс для массива структур | массив структур | Индексный массив структур |
|  | входной | входной | входной | входной |

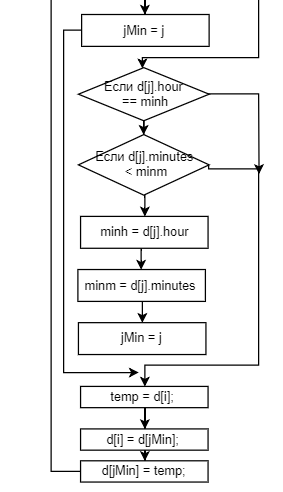
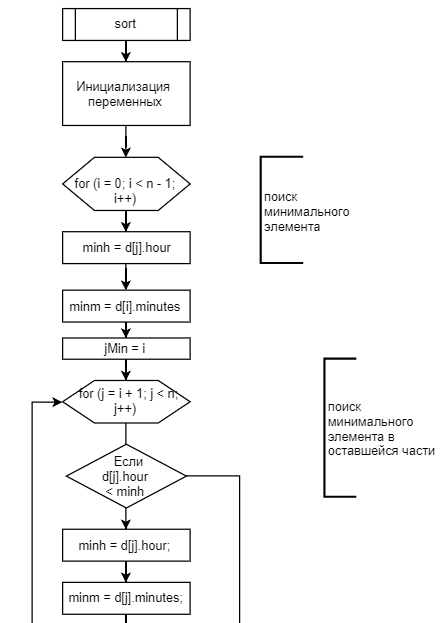


* 1. Блок-схема

1. sort
   1. Назначение: индексная сортировка выбором структур
   2. Прототип: void sort(int &i, int &n, info \*&p, dop \*&d)
   3. Обращение: void sort(i, n, p, d)
   4. Описание параметров: не возвращает параметров, так как тип void

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Параметр | &n | &i | \*&p | \*&d |
| Тип | int | int | info | dop |
| Назначение | количество структур | индекс для массива структур | массив структур | Индексный массив структур |
|  | входной | входной | входной | входной |

* 1. Блок-схема

****

**Псевдокод**

*Основная функция*

НЧ

Инициализация переменных

Если ErrCode = 0

Вывод на экран “Исходные данные:”

Функция pech()

Вывод на экран : “\n\n\nУпорядочение по возрастанию времени:\n”

Функция sort()

Функция pech()

Иначе

Функция PrintMess(ErrCode, FNAME)

КЦ

*Функция ReadF*

НЧ

Инициализация временной переменной под “:”

Если файл не найден

То

Выход по ошибке 1

Иначе

Чтение n из файла

Если n<=2

То

Выход по ошибке 2

Иначе

Создание динамического массива структур

Если файл пуст

То

Выход по ошибке 3

Иначе

От i=0 до i<N; i++

Чтение p[i].timeh из файла

Если p[i].timeh >= 24 и p[i].timeh < 0

То

Выход по ошибке 4

Иначе

Чтение Temp из файла

Если Temp != ':'

То

Выход по ошибке 4

Чтение p[i].timem из файла

Если p[i].timem >= 60 и p[i].timem < 0

То

Выход по ошибке 4

Чтение p[i].marka из файла

Чтение fin >> p[i].num из файла

Если p[i].num[0] != 'Б' и p[i].num[1] != '-'

То

Выход по ошибке 5

Чтение p[i].punkt из файла

Если fin.eof() и i < n – 1

То

Выход по ошибке 6

Если fin.eof() и i = 0

То

Выход по ошибке 3

Если конец файла не найден

То

Выход по ошибке 7

КЦ

*Функция PrintMess*

Если case 1

То

Вывод на экран : "Файл <" « FNAME«"> не найден"

КЦ

Иначе

Если case 2

То

Вывод на экран : "Неверно введено количество структур в " << FNAME

КЦ

Иначе

Если case 3

То

Вывод на экран : "Файл <" << FNAME << "> пуст "

КЦ

Иначе

Если case 4

То

Вывод на экран : "Неверно введено время\n"

КЦ

Иначе

Если case 5

То

Вывод на экран : "Неправильный бортовой номер!\n"

КЦ

Иначе

Если case 6

То

Вывод на экран : "Файл заполнен не до конца"

КЦ

Иначе

Если case 7

То

Вывод на экран : "Файл переполнен"

КЦ

Иначе

Если default

То

Вывод на экран : "Файл <" « FNAME « ">"«endl«"Неизвестный код ошибки "

КЦ

*Функция verh()*

НЧ

Дорисовка верхней части таблицы

КЦ

*Функция shap()*

НЧ

Дорисовка верхней части таблицы

Вывод на экран "Время посадки"

Дорисовка ячеек таблицы

Вывод на экран "Марка ЛА"

Дорисовка ячеек таблицы

Вывод на экран "Бортовой номер"

Дорисовка ячеек таблицы

Вывод на экран "Пункт отправления"

Дорисовка ячеек таблицы

КЦ

*Функция sod(int &i, dop \*&d, info \*p)*

НЧ

Вывод на экран содержания таблицы

КЦ

*Функция niz()*

НЧ

Дорисовка нижней части таблицы

КЦ

*Функция dor()*

НЧ

Дорисовка ячеек таблицы

КЦ

*Функция sort(int &i, int &n, dop \*&d, info \*&p)*

НЧ

Инициализация переменных

От j=0 до j<n-1;j++

minh = d[i].hour;

minm = d[i].minutes;

jMin = i;

От j=i+1 до j<n; j++

Поиск минимального элемента в оставшейся части

Сортировка индексного массива

КЦ

*Функция pech(int &i, int &n, dop \*&d, info \*&p)*

НЧ

Функция verh()

Функция shap()

Функция dor()

От i=0 до i<N; i++

Функция sod(i, p)

Если (i<n-1)

Функция dor();

Функция niz()

КЦ

**Текст программы**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Информатика и Вычислительная техника \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Language: cpp, MSVS 2010 and above \*

\* Programmers: M3O-107Б-18 \*

\* Кривонос Александр Александрович \*

\* Никитин Степан Витальевич \*

\* Created: 02.05.2019 Last revision: 12.05.2019 \*

\* Comment: Практическое задание 2 \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <iostream>

#include <fstream>

#include <iomanip>

#include <stdio.h>

#include <string>

using namespace std;

const char \*FNAME = "Info.txt"; //файловая переменная

const int MMAX = 40; //максимальное количество символов в символьный массивах

ifstream fin(FNAME);

struct info {

int timeh; //переменная для часов

int timem; //переменная для минут

char marka[MMAX]; //символьный массив для марки ЛА

char num[MMAX]; //символьный массив для бортового номера

char punkt[MMAX]; //символьный массив для пункта отправления

};

struct dop {

int hour; //переменная для запоминания часов

int minutes; //переменная для запоминания минут

int f; // переменная для индекса

};

int ReadF(int &n, //количество структур

int &i, //индекс для сортировки

info \*&p, //динамический массив структур

dop \*&d, //динамический индексный массив

const char\* FNAME) { //файловая переменная

char Temp; //временная переменная под ":"

/\*Если файл не найден\*/

if (!fin)

return 1; //выход по ошибке

fin >> n;

if (n <= 0)

return 2;

p = new info[n]; //динамический массив структур

d = new dop[n]; //динамический индексный масив

/\*Если файл пуст\*/

if (fin.eof())

return 3; //выход по ошибке

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

d[i].f = i; //заполняем индексное поле

fin >> p[i].timeh; //считываем часы

if ((p[i].timeh >= 24) || (p[i].timeh < 0))

return 4;

d[i].hour = p[i].timeh;

fin >> Temp;

if (Temp != ':')

return 4;

fin >> p[i].timem; //считываем минуты

if ((p[i].timem >= 60) || (p[i].timem < 0))

return 4;

d[i].minutes = p[i].timem;

fin >> p[i].marka; //считываем марку ЛА

fin >> p[i].num; //считываем бортовой номер

if ((p[i].num[0] != 'Б') || (p[i].num[1] != '-'))

return 5;

fin >> p[i].punkt; //считываем пункт отправления

if ((fin.eof()) && (i < n - 1))

return 6;

//pich(j, n, d);

}

if ((fin.eof()) && (i = 0))

return 3; //выход по ошибке

if (!fin.eof())

return 7; //выход по ошибке

return 0;

}

void PrintMess(int ErrCode, //код ошибки

const char\* FNAME) //файловая переменная

{

switch (ErrCode)

{

case 1: cout << "Файл <" << FNAME << "> не найден " << endl; break;

case 2: cout << "Неверно введено количество структур в " << FNAME << endl; break;

case 3: cout << "Файл <" << FNAME << "> пуст " << endl; break;

case 4: cout << "Неверно введено время\n"; break;

case 5: cout << "Неправильный бортовой номер!\n"; break;

case 6: cout << "Файл заполнен не до конца" << endl; break;

case 7: cout << "Файл переполнен" << endl; break;

default: cout << "Файл <" << FNAME << ">" << endl << "Неизвестный код ошибки "

<< ErrCode << endl; break;

}

}

/\*Дорисовка верхней части таблицы\*/

void verh() {

setlocale(LC\_ALL, "C");

cout << char(218) << setfill(char(196)) << setw(14) << char(194)

<< setfill(char(196)) << setw(13) << char(194)

<< setfill(char(196)) << setw(15) << char(194)

<< setfill(char(196)) << setw(18) << char(191) << endl;

setlocale(LC\_ALL, "Russian");

}

/\*Шапка таблицы\*/

void shap() {

setlocale(LC\_ALL, "C");

cout << char(179);

setlocale(LC\_ALL, "Russian");

cout << "Время посадки" << setfill(' ');

setlocale(LC\_ALL, "C");

cout << char(179);

setlocale(LC\_ALL, "Russian");

cout << "Марка ЛА" << setw(5) << setfill(' ');

setlocale(LC\_ALL, "C");

cout << char(179);

setlocale(LC\_ALL, "Russian");

cout << "Бортовой номер" << setw(1) << setfill(' ');

setlocale(LC\_ALL, "C");

cout << char(179);

setlocale(LC\_ALL, "Russian");

cout << "Пункт отправления" << setw(1) << setfill(' ');

setlocale(LC\_ALL, "C");

cout << char(179) << endl;

setlocale(LC\_ALL, "Russian");

}

/\*Содержание таблицы\*/

void sod(int &i, //индекс для сортировки

dop \*d, //динамический индексный массив структур

info \*p) { //динамический массив структур

setlocale(LC\_ALL, "C");

cout << char(179);

setlocale(LC\_ALL, "Russian");

if ((p[d[i].f].timeh < 10) && (p[d[i].f].timem < 10))

cout << '0' << p[d[i].f].timeh << ":" << '0' << p[d[i].f].timem << setw(9)

<< setfill(' ');

else if (p[d[i].f].timeh < 10)

cout << '0' << p[d[i].f].timeh << ":" << p[d[i].f].timem << setw(9) << setfill(' ');

else if (p[d[i].f].timem < 10)

cout << p[d[i].f].timeh << ":" << '0' << p[d[i].f].timem << setw(9) << setfill(' ');

else

cout << p[d[i].f].timeh << ":" << p[d[i].f].timem << setw(9) << setfill(' ');

setlocale(LC\_ALL, "C");

cout << char(179);

setlocale(LC\_ALL, "Russian");

cout << p[d[i].f].marka << setw(13 - strlen(p[d[i].f].marka)) << setfill(' ');

setlocale(LC\_ALL, "C");

cout << char(179);

setlocale(LC\_ALL, "Russian");

cout << p[d[i].f].num << setw(15 - strlen(p[d[i].f].num)) << setfill(' ');

setlocale(LC\_ALL, "C");

cout << char(179);

setlocale(LC\_ALL, "Russian");

cout << p[d[i].f].punkt << setw(18 - strlen(p[d[i].f].punkt));

setlocale(LC\_ALL, "C");

cout << char(179) << endl;

}

/\*Дорисовка нижней части таблицы\*/

void niz() {

setlocale(LC\_ALL, "C");

cout << char(192) << setfill(char(196)) << setw(14) << char(193)

<< setfill(char(196)) << setw(13) << char(193)

<< setfill(char(196)) << setw(15) << char(193)

<< setfill(char(196)) << setw(18) << char(217) << endl;

setlocale(LC\_ALL, "Russian");

}

/\*Дорисовка ячеек таблицы\*/

void dor() {

setlocale(LC\_ALL, "C");

cout << char(195) << setfill(char(196)) << setw(14) << char(197)

<< setfill(char(196)) << setw(13) << char(197)

<< setfill(char(196)) << setw(15) << char(197)

<< setfill(char(196)) << setw(18) << char(180) << endl;

setlocale(LC\_ALL, "Russian");

}

void pech(int &i, //индекс для сортировки

int &n, //количество структур

dop \*&d, //динамический индексный массив структур

info \*&p) { //динамический массив структур

verh(); //печать верха таблицы

shap(); //печать шапки таблицы

dor(); //дорисовка ячеек

for (i = 0; i < n; i++) { //цикл по индексам структур

sod(i, d, p); //печать содержания таблицы

if (i < n - 1)

dor(); //дорисовка ячеек

}

niz(); //печать низа таблицы

}

void sort(int &i, //индекс для сортировки

int &n, //количество структур

info \*&p, //динамический массив структур

dop \*&d) { //динамический индексный массив структур

int minh; //переменная для запоминания часов

int minm; //переменная для запоминания минут

int minf; //минимальный индекс

int jMin; //индекс минимального элемента

int j = 0; //индексы для сортировки

dop temp; //временная переменная

for (i = 0; i < n - 1; i++) {

/\*поиск минимального элемента\*/

minh = d[i].hour;

minm = d[i].minutes;

minf = d[i].f;

jMin = i;

/\*поиск минимального элемента в оставшейся части\*/

for (j = i + 1; j < n; j++) {

/\*возможный минимальный элемент\*/

if (d[j].hour < minh) {

minh = d[j].hour; //минимальному присвоить j-ый эл-т часов

minm = d[j].minutes; //минимальному присвоить j-ый эл-т минут

minf = d[j].f; //минимальному присвоить j-ый эл-т индексов

jMin = j; //минимальному индексу присвоить j

}//if p[j].timeh

if (d[j].hour == minh) {

if (d[j].minutes < minm) {

minh = d[j].hour; //минимальному присвоить j-ый эл-т часов

minm = d[j].minutes; //минимальному присвоить j-ый эл-т минут

minf = d[j].f; //минимальному присвоить j-ый эл-т индексов

jMin = j; //минимальному индексу присвоить j

}//if p[j].timem

}//if p[j].timeh

}//for j

temp = d[i]; //временной переменной присвоить i поля

d[i] = d[jMin]; //i-ым полям присвоить jMin

d[jMin] = temp; //jMin полям присвоить поля временной переменной

}

}

int main()

{

setlocale(LC\_ALL, "Russian");

int n = 0; //количество структур

int i = 0; //индекс для сортировки

info \*p; //динамический массив структур

dop \*d; //дополнительый массив структур

int ErrCode = ReadF(n, i, p, d, FNAME);

if (ErrCode == 0) {

cout << "Исходные данные:\n";

pech(i, n, d, p);

cout << "\n\n\nУпорядочение по возрастанию времени:\n";

sort(i, n, p, d);

pech(i, n, d, p);

delete[]p;

}

else

PrintMess(ErrCode, FNAME);

system("pause");

return 0;

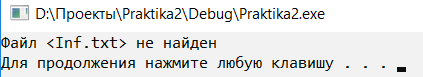
}

**Тесты (корректные и некорректные)**

**1.Некорректные**

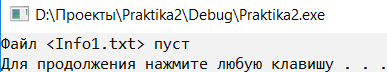
1. Цель: проверить работу программы без файлов с матрицей

Ожидаемый результат: Файл Inf.txt не найден.



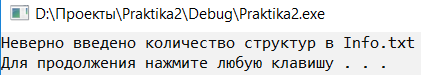
1. Цель: проверить работу программы с пустыми файлами.

Ожидаемый результат: Файл Info1.txt пустой.



1. Цель: проверить работу программы с некорректно введенным количеством структур.

Ожидаемый результат: Неверно введено количество структур в Info.txt



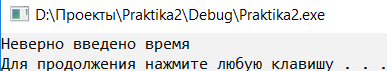
1. Цель: проверить работу программы с некорректно введенным временем.

Исходные данные: 2

-11:05 ТУ-154М Б-3726 Ташкент

12):09 МиГ-31 Б-3 Грозный

Ожидаемый результат: Неверно введено время



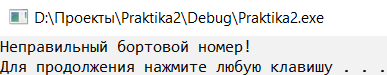
1. Цель: проверить работу программы с некорректно введенным бортовым номером.

Исходные данные: 2

11:05 ТУ-154М f-3726 Ташкент

12:09 МиГ-31 Б3 Грозный

Ожидаемый результат: Неправильный бортовой номер!



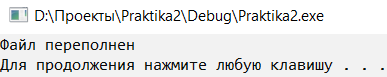
1. Цель: проверить работу программы с большим количеством структур, чем указывалось.

Исходные данные: 1

11:05 ТУ-154М Б-3726 Ташкент

12:09 МиГ-31 Б-3 Грозный

Ожидаемый результат: Файл переполнен



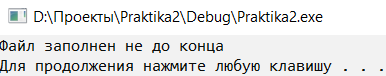
1. Цель: проверить работу программы с меньшим количеством структур, чем указывалось.

Исходные данные: 3

11:05 ТУ-154М Б-3726 Ташкент

12:09 МиГ-31 Б-3 Грозный

Ожидаемый результат: Файл заполнен не до конца



**2. Корректные**

Цель: проверить работу программы с корректно введенными данными

Исходные данные: 5

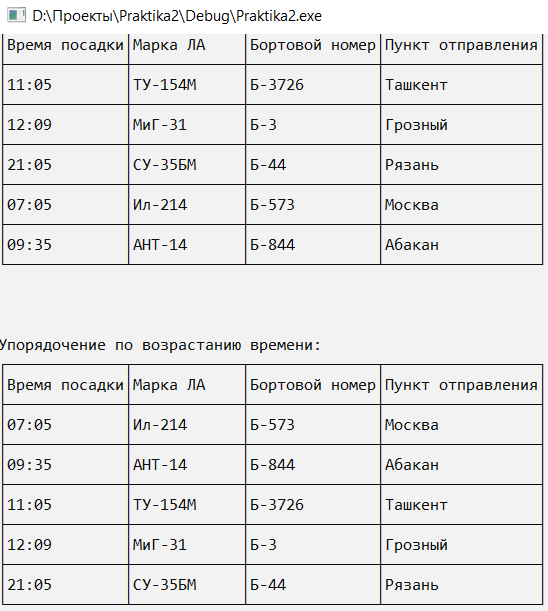
11:05 ТУ-154М Б-3726 Ташкент

12:09 МиГ-31 Б-3 Грозный

21:05 СУ-35БМ Б-44 Рязань

07:05 Ил-214 Б-573 Москва

09:35 АНТ-14 Б-844 Абакан

Ожидаемый результат: 07:05 Ил-214 Б-573 Москва 09:35 АНТ-14 Б-844 Абакан 11:05 ТУ-154М Б-3726 Ташкент 12:09 МиГ-31 Б-3 Грозный 21:05 СУ-35БМ Б-44 Рязань

Цель: проверить работу программы с корректно введенными данными

Исходные данные: 4

11:50 ТУ-154М Б-3726 Ташкент

00:00 МиГ-31 Б-3 Грозный

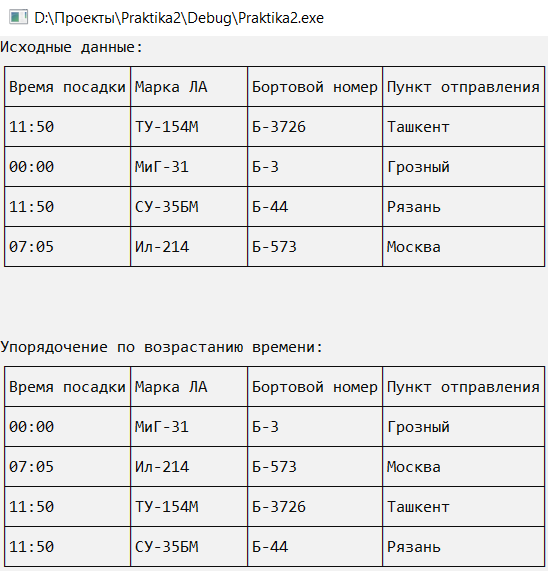
11:50 СУ-35БМ Б-44 Рязань

07:05 Ил-214 Б-573 Москва

Ожидаемый результат: 00:00 МиГ-31 Б-3 Грозный

07:05 Ил-214 Б-573 Москва

11:50 ТУ-154М Б-3726 Ташкент 11:50 СУ-35БМ Б-44 Рязань



**Вывод**

Разработка программы завершена на том основании, что

1) Полученные результаты совпали с ожидаемыми.

2) Считаем, что набор тестов полный.