**Описание**

В зоне действия АСУ ВД имеется 3 аэродрома с номерами 1,2,3. В процессе функционирования данные о самолетах, совершающих посадку, фиксируются в файле, каждая запись которого имеет структуру типа:

**ТУ-154М Б-3726 11:15 АП2**

марка ЛА бортовой время аэродром

номер посадки посадки

1. Подготовить программу, осуществляющую печать таблицы о самолетах, совершающих посадку на каждом аэродроме, в порядке возрастания времени посадки (использовать сортировку методом «пузырька»);
2. Обеспечить входной контроль бортового номера, времени посадки аэродрома посадки, выполнить отладку и тестирование.

Чтение данных из файла производить с использованием функций ввода/вывода языка С++.

Алгоритм должен быть параметризован; обмен дынными с подпрограммой должен осуществляться через параметры; исходные данные хранятся в отдельном файле.

**Текст программы**

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

\* Project name : - \*

\* Project type : win32 console app \*

\* File\_name : - \*

\* Language : cpp. MSVS 2013 and above \*

\* Programmers : Usupov Tengiz T. \*

\* Modified : - \*

\* Created : 07.04.18 11:54 \*

\* Last Revision : 11.05.18 19:45 \*

\* Comment : \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*

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

#include<iostream>

#include<iomanip>

#include<fstream>

#include<string>

#include<cmath>

using namespace std;

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

\* Г Л О Б А Л Ь Н Ы Е К О Н С Т А Н Т Ы \*

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

const string SourceFile = "SourceFile.txt";

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

\* С Т Р У К Т У Р Ы \*

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

struct SAMOLET

{

string brand; // марка ЛА

string bort\_number; // бортовой номер ЛА

string time; // время посадки ЛА

string AP; // аэродром посадки

};

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

/\* П Р О Т О Т И П Ы Ф У Н К Ц И Й \*/

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

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

/\* Подсчет строк в исходном файле \*/

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

int Count\_Lines();

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

/\* Проверка полученной информации \*/

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

void Check\_Data(SAMOLET \*plane, int lines);

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

/\* Чтение из файла \*/

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

int Read\_File(SAMOLET \*LA, int lines);

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

/\* Сортировка методом "пузырька" \*/

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

int SortingBubble(SAMOLET \*LA, int lines);

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

/\* Печать результатов \*/

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

int Print\_Res(SAMOLET \*LA, int lines);

bool Error = false;//Флаг наличия ошибок

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

/\* О С Н О В Н А Я П Р О Г Р А М М А \*/

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

int main()

{

int Number\_Lines = Count\_Lines(); //подсчет количества строк

SAMOLET \*LA = new SAMOLET[Number\_Lines]; //создание динамической матрицы структуры

Read\_File(LA, Number\_Lines); //чтение из файла

if (Number\_Lines != 2)

{

if ((Number\_Lines == 1) && (LA[0].brand == ""))

{

cout << "File is empty!" << endl;

system("pause");

return 0;

}//if

Check\_Data(LA, Number\_Lines); //проверка на корректность данных

SortingBubble(LA, Number\_Lines); //сортировка методом "пузырька"

}

return 0;

}//main

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

/\* РЕАЛИЗАЦИЯ ФУНКЦИЙ \*/

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

int Count\_Lines()

{

string str; //Воспомогательная строка

int lines = 0; //Счетчик строк

ifstream file(SourceFile);

if (!file)//Если файл отсутствует

{

cout << "Error! File " << SourceFile << " not found." << endl;

file.close();//закрытие файла для чтения

system("pause");

return 2;

}//if

while (!file.eof()) //Пока файл не закончился

{

getline(file, str);//Чтение строки

lines++;

}//while

file.close();

return lines;

}//Count\_Lines

int Read\_File(SAMOLET \*LA, int lines)

{

ifstream fin(SourceFile);

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

{

fin >> LA[i].brand;

fin >> LA[i].bort\_number;

fin >> LA[i].time;

fin >> LA[i].AP;

}//for

return 0;

}//Read\_File

void Check\_Data(SAMOLET \*LA, int lines)

{

bool time\_Error; //флаг ошибки во времени посадки

bool brand\_Error; //флаг ошибки в марке ЛА

bool bort\_number\_Error; //флаг ошибки в бортовом номере

bool AP\_Error; //флаг ошибки в аэродроме посадкие

string temp; //воспомогательная строка

string hours = ""; //строка для записи часов

string minutes = ""; //строка для записи минут

int k;

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

{

AP\_Error = false; //Инициализация флагов

time\_Error = false;

brand\_Error = false;

bort\_number\_Error = false;

hours = ""; //Инициализация часов

minutes = ""; //Инициализация минут

//проверка марки ЛА на корректность

if (LA[i].brand == "")//Если строка пуста

{

brand\_Error = true;

}

//проверка бортового номера на корректность

if ((LA[i].bort\_number == "") || (LA[i].bort\_number.size() != 6))

{

bort\_number\_Error = true;

}

else

{

if ((LA[i].bort\_number[0] != 'B') || (LA[i].bort\_number[1] != '-'))

{

bort\_number\_Error = true;

}

for (k = 2; k < LA[i].bort\_number.size(); k++)

{

if (((int)LA[i].bort\_number[k] < 48) ||

(((int)LA[i].bort\_number[k] > 57)))//если последние 4 эл-та не цифры

{

bort\_number\_Error = true;

}

}

}

//проверка на уникальность бортового номера

for (k = i - 1; k >= 0; k--)

{

if (LA[i].bort\_number == LA[k].bort\_number)

{

bort\_number\_Error = true;

}

}

//проверка времени на корректность

if ((LA[i].time.size() != 5) || (LA[i].time[2] != ':'))

{

time\_Error = true;

}

else

{

k = 0;//инициализация индекса элементов в строке

while ((LA[i].time[k] != LA[i].time[2]))

{

hours += LA[i].time[k];

k++;

}

k++;

while (LA[i].time[k] != '\0')

{

minutes += LA[i].time[k];//запись минут

k++;

}

for (k = 0; k < 2; k++)

if (((int)hours[k] < 48) || ((int)hours[k] > 57))//если элементы в строке часов не цифры, 48 и 57 коды аски

{

time\_Error = true;

}

else

{

if ((hours > "23") || (hours < "0"))//если часы не в интервале от 0 до 23

time\_Error = true;

}

for (k = 0; k < 2; k++)

if (((int)minutes[k] < 48) || ((int)minutes[k] > 57))//если элементы в строке минут не цифры

{

time\_Error = true;

}

else

{

if ((minutes > "59") || (minutes < "0"))//если минуты не в интервале от 0 до 59

time\_Error = true;

}

}

//проверка аэродрома посадки на корректность

if ((LA[i].AP == "") || (LA[i].AP.size() != 3))

{

AP\_Error = true;

}

else

{

if ((LA[i].AP[0] != 'A') || (LA[i].AP[1] != 'L'))

{

AP\_Error = true;

}

for (k = 2; k < LA[i].AP.size(); k++)

{

if (((int)LA[i].AP[k] < 49) || ((int)LA[i].AP[k] > 51))

{

AP\_Error = true;

}

}

}

if (time\_Error)

{

cout << " -Time is set incorrect." << endl;

LA[i].time = "Error";

Error = true;

}

if (brand\_Error)

{

cout << " -Brand is set incorrect." << endl;

LA[i].brand = "Error";

Error = true;

}

if (bort\_number\_Error)

{

cout << " -Onboard is set incorrect or not unigue in the list. " << endl;

LA[i].bort\_number = "Error";

Error = true;

}

if (AP\_Error)

{

cout << " -Aerodrome landing is set incorrect." << endl;

LA[i].AP = "Error";

Error = true;

}

}//for

Print\_Res(LA, lines);

}//Check\_Data

int SortingBubble(SAMOLET \*LA, int lines)

{

for (int i = 0; i < lines - 1; i++)

{

for (int j = lines - 2; j >= i; j--)

{

if (LA[j].time >= LA[j + 1].time)

{

SAMOLET Temp = LA[j];

LA[j] = LA[j + 1];

LA[j + 1] = Temp;

}//if

}//for(j)

}//for(i)

cout << "\n\t\t\tSort data" << endl;

Print\_Res(LA, lines);

return 0;

}//SortingBubble

int Print\_Res(SAMOLET \*LA, int lines)

{

setlocale(LC\_ALL, "C"); // подключение английского языка

unsigned char a, b, c, d, e, f; // таблица ASCII

a = 196;

b = 179;

c = 191;

d = 195;

e = 218;

f = 180;

cout << e;

for (int j = 0; j < 59; j++)

{

cout << a;

}//for

cout << c << "\n" << b << setw(15) << "Airplane brand" << b << setw(14) << "Board number" << setw(3) << b << setw(5);

cout << "Time" << setw(3) << b << setw(14) << "Aerodrome" << setw(5) << b << "\n" << d;

for (int j = 0; j < 59; j++)

{

cout << a;

}//for

cout << f;

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

{

cout << "\n" << b << setw(11) << LA[i].brand << setw(5) << b << setw(11) << LA[i].bort\_number << setw(6) << b << setw(6) << LA[i].time;

cout << setw(2) << b << setw(11) << LA[i].AP << setw(8) << b << "\n" << d;

for (int j = 0; j < 59; j++)

{

cout << a;

}//for

cout << f;

}//for

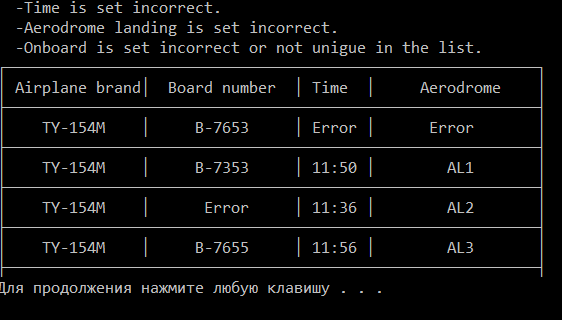
cout << endl;

system("PAUSE");

return 0;

}//Print\_Res

**Результат работы программы**

**

Исходные данные:

TY-154M B-7653 11:S6 AL4

TY-154M B-7353 11:50 AL1

TY-154M B-7653 11:36 AL2

TY-154M B-7655 11:56 AL3

Результат программы:

Исходные данные:

TY-154M B-7653 11:16 AL2

TY-154M B-7353 11:50 AL1

TY-154M B-7633 11:36 AL2

TY-154M B-7655 11:56 AL3

Результат программы: