|  |  |
| --- | --- |
| https://lh7-us.googleusercontent.com/QuBaagabzZYLr7U3ZbYOtMusd5FbWWxIvMUTN8jrLHzSLMg534z9gXRTIG1Us4i_lOwmWlaBxKedNt-SQ26dm4WmyqwjGDmEO6z8GE3QrZosqvHM88J2EFeVf1u0GzyCZQlhWmp1Zeo85tKo4LJVXQ | МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ  ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ (НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)» |

Институт № 3 «Системы управления, информатика и электроэнергетика»

Кафедра 304 «Вычислительные машины, системы и сети»

Отчёт по практике. Вариант №12  
по учебной дисциплине «Программирование»

на тему «Работа со структурами данных»

Выполнил  
студент группы М3О-119БВ-24

Нарзиев А.Т.

Принял

Ст. преподаватель каф. 304

Давыдкина Елена Александровна

Москва  
 2024

# Содержание

[Содержание 2](#_Toc196439019)

[Задание 3](#_Toc196439020)

[Решение 4](#_Toc196439021)

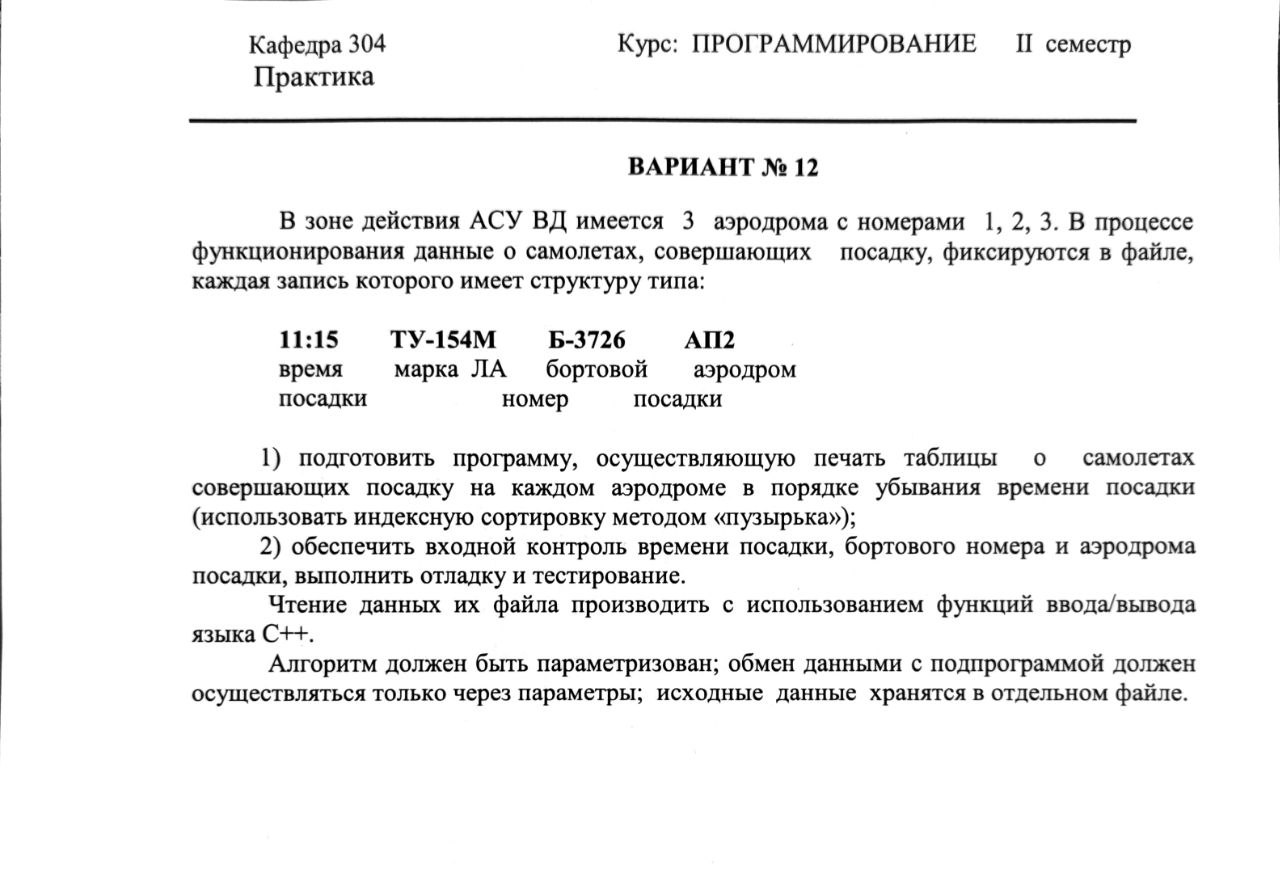
[1. Блок-схема: 4](#_Toc196439022)

[2. Код программы: 5](#_Toc196439023)

[Тесты 10](#_Toc196439024)

[Вывод 11](#_Toc196439025)

# Задание



# Решение

## Блок-схема:

## Код программы:

### Главный файл – main.cpp:

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

\*                               Курс Информатика                              \*

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

\* Project type  : Windows Console Application                                 \*

\* Project name  : Practics                                                    \*

\* File name     : main.cpp                                                    \*

\* Language      : CPP                                                         \*

\* Programmers   : Нарзиев Артемий Тимурович                                   \*

\* Modified By   :                                                             \*

\* Created       : 15.04.2025                                                  \*

\* Last Revision : 25.04.2025                                                  \*

\* Comment       : Работа со структурами данных. Вариант: 12                   \*

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

#include <iostream>

#include <fstream>

#include <string>

#include <cctype>

#include <iomanip>

// Для локали на русском

#include <locale>

#include <codecvt>

#include <windows.h>

using namespace std;

// Структура для хранения информации о самолете

struct Plane {

    string time;         // Время посадки в формате HH:MM

    int minutes;        // Время в минутах для сортировки

    string model;       // Название модели воздушного судна

    string bortNumber;  // Бортовой номер формата X-XXXX

    string airport;     // Код аэродрома (АП1, АП2, АП3)

};

const int MAX\_PLANES = 100; // Максимальное количество самолетов

Plane planes[MAX\_PLANES];    // Массив для хранения самолетов

int planeCount = 0;          // Счетчик самолетов

// Проверка формата времени

/\*\*

 \* Проверка корректности формата времени

 \* @param time Строка времени в формате HH:MM

 \* @return true если время валидно, false в противном случае

 \*/

bool isTimeValid(const string& time) {

    if (time.length() != 5 || time[2] != ':') return false;

    for (int i = 0; i < 5; i++) {

        if (i == 2) continue;

        if (!isdigit(time[i])) return false;

    }

    int hours = (time[0]-'0')\*10 + (time[1]-'0');

    int mins = (time[3]-'0')\*10 + (time[4]-'0');

    return (hours >= 0 && hours <= 23) && (mins >= 0 && mins <= 59);

}

// Проверка бортового номера

/\*\*

 \* Проверка формата бортового номера

 \* @param bort Строка бортового номера

 \* @return true если формат X-XXXX соблюден, false в противном случае

 \*/

bool isBortValid(const string& bort) {

    if (bort.length() != 6 || bort[1] != '-') return false;

    if (!isalpha(bort[0])) return false;

    for (int i = 2; i < 6; i++) {

        if (!isdigit(bort[i])) return false;

    }

    return true;

}

// Проверка аэродрома

/\*\*

 \* Проверка принадлежности аэродрому

 \* @param airport Код аэродрома

 \* @return true если код соответствует допустимым значениям

 \*/

bool isAirportValid(const string& airport) {

    return airport == "АП1" || airport == "АП2" || airport == "АП3";

}

/\*\*

 \* Обрезка пробелов в начале и конце строки

 \* @param s Исходная строка

 \* @return Очищенная от пробелов строка

 \*/

string trim(const string& s) {

    int start = 0, end = s.length() - 1;

    while (start <= end && (s[start] == ' ' || s[start] == '\t')) start++;

    while (end >= start && (s[end] == ' ' || s[end] == '\t')) end--;

    return s.substr(start, end - start + 1);

}

// Функция сортировки пузырьком для сортировки индексов самолетов по времени посадки

/\*\*

 \* Сортировка индексов массива по времени посадки (пузырьковая)

 \* @param indices Массив индексов для сортировки

 \* @param size Количество элементов в массиве

 \*/

void bubbleSort(int indices[], int size) {

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

        for (int j = 0; j < size - i - 1; j++) {

            if (planes[indices[j]].minutes < planes[indices[j + 1]].minutes) {

                int temp = indices[j];

                indices[j] = indices[j + 1];

                indices[j + 1] = temp;

            }

        }

    }

}

// Вывод таблицы

/\*\*

 \* Форматированный вывод таблицы с данными

 \* @param indices Отсортированные индексы записей

 \* @param count Количество выводимых записей

 \* @param airport Код аэродрома для заголовка

 \*/

void printTable(int indices[], int count, const string& airport) {

    const int TIME\_WIDTH = 12;

    const int MODEL\_WIDTH = 15;

    const int BORT\_WIDTH = 16;

    const int AIRPORT\_WIDTH = 18;

    cout << "\nАэродром " << airport << ":\n";

    cout << "┌────────────┬───────────────┬────────────────┬────────────────┐\n";

    cout << "│ Время      │ Марка ЛА      │ Бортовой номер │ Аэродром       │\n";

    cout << "├────────────┼───────────────┼────────────────┼────────────────┤\n";

    for (int i = 0; i < count; i++) {

        const Plane& p = planes[indices[i]];

        cout << "│ " << left << setw(TIME\_WIDTH-2) << p.time

             << " │ " << setw(MODEL\_WIDTH-2) << p.model.substr(0, MODEL\_WIDTH-2)

             << " │ " << setw(BORT\_WIDTH-2) << p.bortNumber

             << " │ " << setw(AIRPORT\_WIDTH-2) << p.airport << " │\n";

    }

    cout << "└────────────┴───────────────┴────────────────┴────────────────┘\n\n";

}

int main() {

    // Установка кодировки консоли для корректного отображения символов

    SetConsoleOutputCP(CP\_UTF8);

    setlocale(LC\_ALL, "en\_US.UTF-8");

    // Открытие файла с данными о посадках и проверка на ошибки

    ifstream file("landings.txt");

    if (!file.is\_open()) {

        cerr << "Ошибка открытия файла!";

        return 1; // Завершение программы при ошибке открытия файла

    }

    // Чтение данных из файла в массив

    string line;

    while (getline(file, line) && planeCount < MAX\_PLANES) {

        // Обработка строки (без использования stringstream)

        // Реализация разбиения строки на части через поиск запятых

        size\_t pos1 = line.find(','); // Найти первую запятую

        size\_t pos2 = line.find(',', pos1 + 1); // Найти вторую запятую

        size\_t pos3 = line.find(',', pos2 + 1); // Найти третью запятую

        // Проверка на корректность формата строки

        if (pos1 == string::npos || pos2 == string::npos || pos3 == string::npos) {

            cerr << "Некорректная строка: " << line << endl; // Вывод ошибки

            continue; // Переход к следующей строке

        }

        planes[planeCount].time = trim(line.substr(0, pos1));

        planes[planeCount].model = trim(line.substr(pos1 + 1, pos2 - pos1 - 1));

        planes[planeCount].bortNumber = trim(line.substr(pos2 + 1, pos3 - pos2 - 1));

        planes[planeCount].airport = trim(line.substr(pos3 + 1));

        // Приведение бортового номера в нормальный вид

        planes[planeCount].bortNumber[0] = toupper(planes[planeCount].bortNumber[0]);

        // Проверки

        if (!isTimeValid(planes[planeCount].time)) {

            cerr << "Пропуск: неверное время " << planes[planeCount].time << endl;

            continue;

        }

        if (!isBortValid(planes[planeCount].bortNumber)) {

            cerr << "Пропуск: неверный борт " << planes[planeCount].bortNumber << endl;

            continue;

        }

        if (!isAirportValid(planes[planeCount].airport)) {

            cerr << "Пропуск: неверный аэродром " << planes[planeCount].airport << endl;

            continue;

        }

        // Конвертация времени в минуты

        planes[planeCount].minutes =

            ((planes[planeCount].time[0] - '0') \* 10 + (planes[planeCount].time[1] - '0')) \* 60 +

            ((planes[planeCount].time[3] - '0') \* 10 + (planes[planeCount].time[4] - '0'));

        planeCount++; // Увеличение счетчика самолетов

    }

    file.close(); // Закрытие файла после чтения

    // Обработка аэродромов

    const string airports[] = {"АП1", "АП2", "АП3"}; // Список аэродромов

    const size\_t airportsCount = sizeof(airports) / sizeof(airports[0]); // Вычисляем количество аэропортов

    // Цикл по каждому аэродрому

    for (size\_t j = 0; j < airportsCount; j++) {

        const string& ap = airports[j]; // Получаем текущий аэропорт

        int indices[MAX\_PLANES];

        int count = 0;

        // Сбор индексов

        for (int i = 0; i < planeCount; i++) {

            if (planes[i].airport == ap) {

                indices[count++] = i;

            }

        }

        // Сортировка и вывод

        if (count == 0) {

            cout << "Аэродром " << ap << ": посадок нет\n";

            continue;

        }

        bubbleSort(indices, count);

        printTable(indices, count, ap);

    }

    return 0;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* КОНЕЦ main.cpp \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

### Файл записей – landings.txt:

25:70,InvalidTime,B-0000,АП2

09:15,Airbus,1-ABCD,АП3

18:45,SSJ-100,C-9876,АП4

24:00,Airbus-A380,A-9999,АП1

12-30,Cessna,X-1234,АП2

10:15,Boeing-737,ABCDEF,АП3

14:20, ,F-5678,АП2

16:40,Concorde,G-H 123,АП1

19:05,Ту-204,H-7B8C,АП3

12 : 45,SuperJet,I - 9012,АП2

08:00,AN-24,d-1122,АП1

14:45,Airbus-A320,B-5678,АП1

12:30,Boing-777,A-1234,АП1

10:30,LickIt-172,C-1122,АП2

16:20,Boeing-747,D-3344,АП2

07:00,Gymbro-190,E-5566,АП3

13:15,Bombardiro,F-7788,АП3

20:05,Crocodilo,G-9900,АП3

# Тесты

# Вывод