МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МИРЭА – Российский технологический университет»

РТУ МИРЭА

ОТЧЕТ
ПО ПРАКТИЧЕСКОЙ РАБОТЕ №4
по дисциплине
«Структуры и алгоритмы обработки данных»

Выполнил студент группы ИКБО-13-21 Дамарад Д.В.

Принял
старший преподаватель Скворцова Л.А.

Практическая «__»___2021 г.
работа выполнена
«Зачтено» «__»___2021 г.

Москва 2022

СОДЕРЖАНИЕ

1 ПОСТАНОВКА ЗАДАЧИ

Разработать набор операций для управления таблицей, созданной на основе статического и динамического массива и вектора.

Аэропорт (управление вылетом пассажирских авиарейсов): пункт назначения, номер рейса, дата вылета, время вылета, время прибытия в пункт назначения, количество свободных мест, информация о задержке вылета в часах. Количество строк на табло отображения 10.

Операции

1) Вставить информацию по новому рейсу в таблицу перед рейсом с большим номером.

2) Удалить информацию о вылетевшем рейсе и сохранить ее в архивной таблице.

3) Вывести рейсы, готовые к вылету по расписанию (за 2 часа до времени вылета).

4) Сформировать список номеров рейсов, вылетающих в заданный пункт назначения.

2 ПРОЕКТИРОВАНИЕ И РЕАЛИЗАЦИЯ

Структура записи представлена на Рисунок 1.

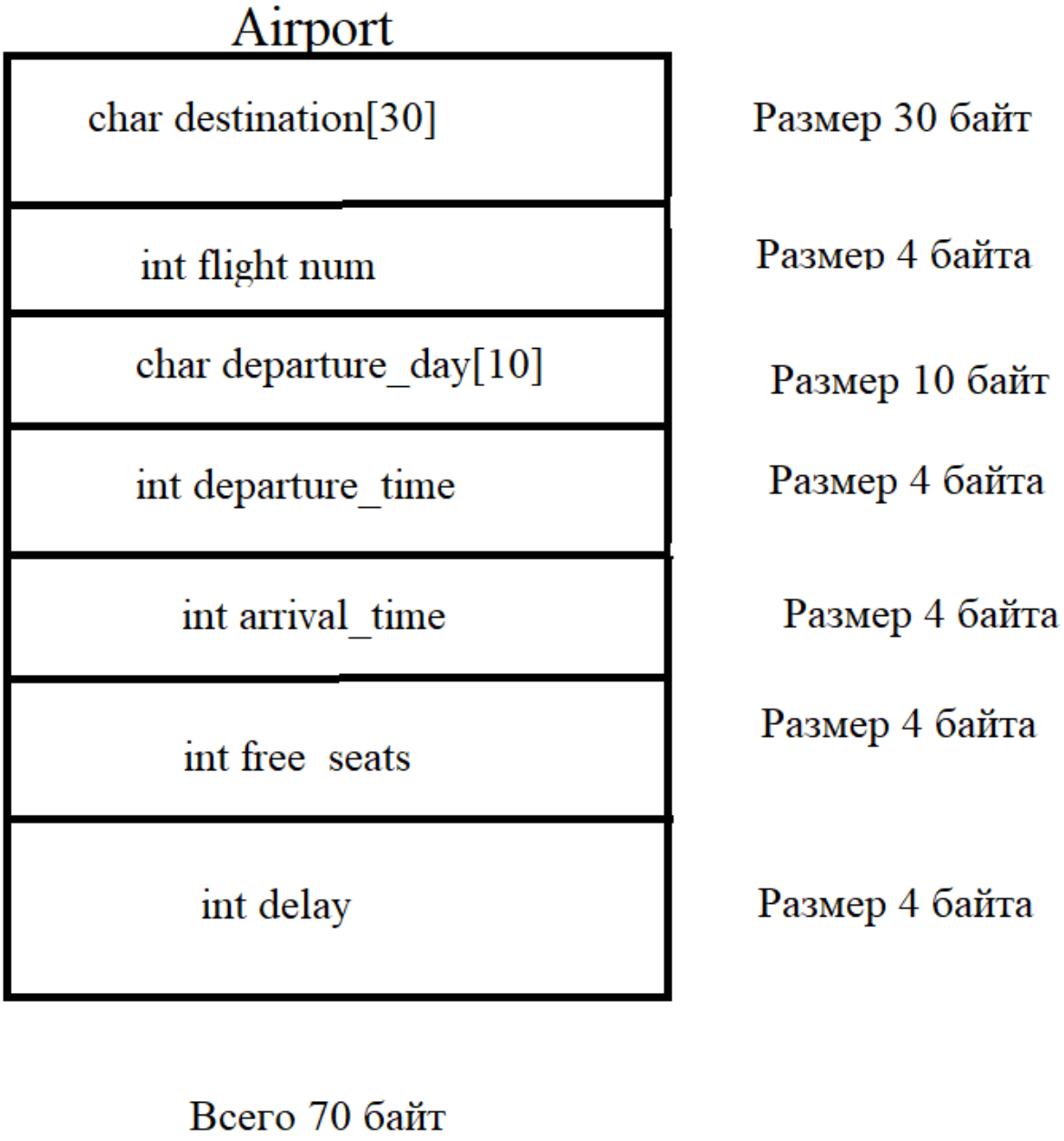

Рисунок 1 - структура записи

Определение структуры записи средствами языка программирования:

```
struct Airport {
    char destination[30]; //Пункт назначения
    int flight_num; //Номер рейса
    char departure_day[10]; //День вылета
    int departure_time; //Время вылета
    int arrival_time; //Время прибытия
    int free_seats; // Кол-во свободных мест
    int delay; // Информация о задержке вылета в часах
};
```

Декомпозиция:

Задачу следует разбить на следующие подзадачи:

- Заполнение одной записи.
- Вывод одной записи.
- Вставить запись.
- Вывести запись.
- Вставить информацию по новому рейсу в таблицу перед рейсом с большим номером.
- Удалить информацию о вылетевшем рейсе и сохранить ее в архивной таблице.
- Вывести рейсы, готовые к вылету по расписанию (за 2 часа до времени вылета).
- Сформировать список номеров рейсов, вылетающих в заданный пункт назначения.

Определение функций (на примере статического массива):

1. Заполнение одной записи.

Предусловие: Airport &r – ссылка на переменной, в которой будут заполняться поля.

Постусловие: инициализованная переменная типа Airport.

void inputOneRecord(Airport& r).

2. Вывод одной записи.

Предусловие: Airport &r – переменная, которая будет выведена экран.

Постусловие: выведенная переменная типа Airport.

void outputOneRecord(Airport r).

3. Вставить запись.

Предусловие: Airport *r – указатель на первый элемент массива типа Airport, int i–итератор.

Постусловие: Инициализированный элемент массива типа Airport.

void insertRecord(Airport* r, int i)

4. Вывести запись.

Предусловие: Airport *r – указатель на первый элемент массива типа Airport, int i–итератор.

Постусловие: выведенный элемент массива типа Airport.

void outBoard(Airport* r, int i)

5. Вставить информацию по новому рейсу в таблицу перед рейсом с большим номером.

Предусловие: Airport r[] –массив элементов типа Airport, int n–количество заполненных элементов в массиве, Airport newelement – новый элемент, который будет вставлен в таблицу перед рейсом с большим номером.

Постусловие: измененный массив типа Airport.

void insertRecordBeforeMax(Airport r[], int n, Airport newelement)

6. Удалить информацию о вылетевшем рейсе и сохранить ее в архивной таблице.

Предусловие: Airport r[] –массив элементов типа Airport, int n–количество заполненных элементов в массиве, Airport ar[] – архивное табло, int time – текущее время.

Постусловие: измененный массив элементов типа Airport и новый массив элементов типа Airport (архивное табло).

void deleteLeft(Airport r[], Airport ar[], int n, int time)

7. Вывести рейсы, готовые к вылету по расписанию (за 2 часа до времени вылета).

Предусловие: Airport r[] –массив элементов типа Airport, int n–количество

заполненных элементов в массиве, int time – текущее время.

Постусловие: измененный массив элементов типа Airport и новый массив элементов типа Airport (архивное табло).

void outputReady(int time, Airport r[], int n)

8. Сформировать список номеров рейсов, вылетающих в заданный пункт назначения.

Предусловие: Airport r[] –массив элементов типа Airport, int n–количество заполненных элементов в массиве, char s[]-массив символов, заданный пункт назначения.

Постусловие: измененный массив элементов типа Airport и новый массив элементов типа Airport (архивное табло).

void outputGivenDestination(char s[], int n, Airport r[])

Разработка алгоритмов функций и представление их на псевдокоде:

1. Заполнение одной записи.

```
void inputOneRecord(Airport& r) {
   cout << "----Вставка инфорармации по рейсу----" << endl;
   cout << "Пункт назначения: ";  r.destination←input;
   cout << "Номер рейса: ";  r.flight_num←input;
   cout << "День вылета: "; r.departure_day←input;
   cout << "Время вылета: "; r.departure_time←input;
   cout << "Время прибытия: "; r.arrival_time←input;
   cout << "Кол-во свободных мест: "; r.free_seats←input;
   cout << "Информация о задержке вылета в часах: "; r.delay←input;
   cout << "-----------------------------------" << endl;
}
```

2. Вывод одной записи.

```
void outputOneRecord(Airport r) {
   cout << "Пункт назначения: "; cout << r.destination << endl;
   cout << "Номер рейса: "; cout << r.flight_num << endl;
   cout << "День вылета: "; cout << r.departure_day << endl;
   cout << "Время вылета: "; cout << r.departure_time << endl;
   cout << "Время прибытия: "; cout << r.arrival_time << endl;
```

```
    cout << "Кол-во свободных мест: "; cout << r.free_seats << endl;
    cout << "Информация о задержке вылета в часах: "; cout << r.delay << endl;
}
```

3. Вставить запись.

```
void insertRecord(Airport* r, int i) {
    inputOneRecord(r[i]);
}
```

4. Вывести запись

```
void outBoard(Airport* r, int i) {
    outputOneRecord(r[i]);
}
```

5. Вставить информацию по новому рейсу в таблицу перед рейсом с большим номером.

```
void insertRecordBeforeMax(Airport r[], int n, Airport newelement) {
    int max ← 0;
    Airport temp, sdvig;
    for для заполненных элементов таблицы {
        if (r[i].flight_num > max) {
            max = r[i].flight_num;
        }
    }
    for для заполненных элементов таблицы {
        if r[i].flight_num равно max {
            temp ← r[i];
            r[i] ← newelement;
            for (int j ← i+1; j < n+1; j++){
                sdvig ← r[j];
                r[j] ← temp;
                temp ← sdvig;
                i=i+1;
            }
        }
    }
}
```

6. Удалить информацию о вылетевшем рейсе и сохранить ее в архивной таблице.

```
void deleteLeft(Airport r[], Airport ar[], int n, int time) {
    int j ← 0, k ← 0, x;
    for для заполненных элементов таблицы {
        if time - r[i].departure_time больше или равно 0 {
```

```
            ar[j] ← r[i];
            j←j+1;
        }
        else {
            r[k] ← r[i];
            k←k+1;
        }
    }
    cout << "Желаете вывести таблицу?" << endl
        << "1-Архивная таблица" << endl
        << "2-Изменное табло" << endl
        << "Другое значение - выход" << endl;
    x←input;
    switch (x)
    {
    case 1: {
        cout << endl << "----Архивная таблица------" << endl;
        for (int i←0; i < j; i++) {
            cout << "----Информация по рейсу №" << i + 1 << "----" << endl;
            outBoard(ar, i);
            cout << "--------------------------------" << endl;
        }
        break;
    }
    case 2: {
        for (int i ← 0; i < k; i++) {
            cout << "----Информация по рейсу №" << i + 1 << "----" << endl;
            outBoard(r, i);
            cout << "--------------------------------" << endl;
        }
        break;
    }
    default: {
        break;
    }
    }
}
```

7. Вывести рейсы, готовые к вылету по расписанию (за 2 часа до времени вылета).

```
void outputReady(int time, Airport r[], int n) {
   for для заполненных элементов таблицы {
      if r[i].departure_time - time равно 2 {
```

```
            cout << "----Рейс №" << i + 1 << " готов к вылету по расписанию----" <<
endl;
            outputOneRecord(r[i]);
            cout << "-----------------------------------------------" << endl;
        }
        else {
            cout << "Нет рейсов, которые готовы вылететь." << endl;
        }
    }
}
```

8. Сформировать список номеров рейсов, вылетающих в заданный пункт назначения.

```
void outputGivenDestination(char s[], int n, Airport r[]) {
    for для заполненных элементов таблицы {
        if r[i].destination равно s {
            cout << "----Рейс №" << i + 1 << " вылетает в заданный пункт
назначения----" << endl;
            outputOneRecord(r[i]);
            cout << "-----------------------------------------------" << endl;
        }
    }
}
```

Набор тестовых данных по наполнению таблицы:

Количество рейсов (n): 3
----Вставка информации по рейсу----
Пункт назначения: Novorossiysk
Номер рейса: 3
День вылета: Monday
Время вылета: 7
Время прибытия: 10
Кол-во свободных мест: 0
Информация о задержке вылета в часах: 0

----Вставка информации по рейсу----
Пункт назначения: Anapa
Номер рейса: 7
День вылета: Wednsday
Время вылета: 3
Время прибытия: 8
Кол-во свободных мест: 8
Информация о задержке вылета в часах: 0

----Вставка инфомации по рейсу----
Пункт назначения: Moscow
Номер рейса: 4
День вылета: wednsday
Время вылета: 5
Время прибытия: 9
Кол-во свободных мест: 9
Информация о задержке вылета в часах: 0

Тесты операций:

1) insertRecordBeforeMax

Информация, которая будет вставлена:

Пункт назначения: Omsk
Номер рейса: 77
День вылета: Sunday
Время вылета: 12
Время прибытия: 18
Кол-во свободных мест: 5
Информация о задержке вылета в часах: 0

Результат:

-----Новое табло-----
----Информармация по рейсу №1----
Пункт назначения: Novorossiysk
Номер рейса: 3
День вылета: Monday
Время вылета: 7
Время прибытия: 10
Кол-во свободных мест: 0
Информация о задержке вылета в часах: 0

----Информармация по рейсу №2----
Пункт назначения: Omsk
Номер рейса: 77
День вылета: Sunday
Время вылета: 12
Время прибытия: 18
Кол-во свободных мест: 5
Информация о задержке вылета в часах: 0

----Информармация по рейсу №3----
Пункт назначения: Anapa
Номер рейса: 7
День вылета: Wednsday
Время вылета: 3
Время прибытия: 8
Кол-во свободных мест: 8
Информация о задержке вылета в часах: 0

----Информармация по рейсу №4----
Пункт назначения: Moscow
Номер рейса: 4
День вылета: wednsday
Время вылета: 5
Время прибытия: 9
Кол-во свободных мест: 9
Информация о задержке вылета в часах: 0
---
2) deleteLeft

time = 6

Результат:

----Архивная таблица------
----Информация по рейсу №1----

Пункт назначения: Anapa
Номер рейса: 7
День вылета: Wednsday
Время вылета: 3
Время прибытия: 8
Кол-во свободных мест: 8
Информация о задержке вылета в часах: 0

----Информация по рейсу №2----
Пункт назначения: Moscow
Номер рейса: 4
День вылета: wednsday
Время вылета: 5
Время прибытия: 9
Кол-во свободных мест: 9
Информация о задержке вылета в часах: 0

Измененная таблица:

----Информация по рейсу №1----
Пункт назначения: Novorossiysk
Номер рейса: 3
День вылета: Monday
Время вылета: 7
Время прибытия: 10
Кол-во свободных мест: 0
Информация о задержке вылета в часах: 0
---

3) outputReady

time=1

Результат:

----Рейс №2 готов к вылету по расписанию----
Пункт назначения: Anapa
Номер рейса: 7
День вылета: Wednsday
Время вылета: 3
Время прибытия: 8
Кол-во свободных мест: 8
Информация о задержке вылета в часах: 0
---

4) outputGivenDestination

given_destination =”Anapa”

Результат:

----Рейс №2 вылетает в заданный пункт назначения----
Пункт назначения: Anapa
Номер рейса: 7
День вылета: Wednsday
Время вылета: 3
Время прибытия: 8
Кол-во свободных мест: 8
Информация о задержке вылета в часах: 0

Стоит отметить, что все функции для реализации задачи на динамическом массиве и векторе схожи, изменены только параметры функций.

Полный код программы

static.cpp

```
#include <iostream>
using namespace std;
const int max_size = 100;

struct Airport {
    char destination[30]; //Пункт назначения
    int flight_num; //Номер рейса
    char departure_day[10]; //День вылета
    int departure_time; //Время вылета
    int arrival_time; //Время прибытия
    int free_seats; // Кол-во свободных мест
    int delay; // Информация о задержке вылета в часах
};

void inputOneRecord(Airport& r) {
    cout << "----Вставка инфорармации по рейсу----" << endl;
    cout << "Пункт назначения: "; cin >> r.destination;
    cout << "Номер рейса: "; cin >> r.flight_num;
    cout << "День вылета: "; cin >> r.departure_day;
    cout << "Время вылета: "; cin >> r.departure_time;
    cout << "Время прибытия: "; cin >> r.arrival_time;
    cout << "Кол-во свободных мест: "; cin >> r.free_seats;
    cout << "Информация о задержке вылета в часах: "; cin >> r.delay;
    cout << "-------------------------------------" << endl;
}

void outputOneRecord(Airport r) {
    cout << "Пункт назначения: "; cout << r.destination << endl;
    cout << "Номер рейса: "; cout << r.flight_num << endl;
    cout << "День вылета: "; cout << r.departure_day << endl;
    cout << "Время вылета: "; cout << r.departure_time << endl;
    cout << "Время прибытия: "; cout << r.arrival_time << endl;
    cout << "Кол-во свободных мест: "; cout << r.free_seats << endl;
    cout << "Информация о задержке вылета в часах: "; cout << r.delay << endl;
}

void insertRecord(Airport* r, int i) {
    inputOneRecord(r[i]);
}

void outBoard(Airport* r, int i) {
    outputOneRecord(r[i]);
}

void insertRecordBeforeMax(Airport r[], int n, Airport newelement) {
    int max = 0;
    Airport temp, sdvig;
    for (int i = 0; i < n; i++) {
        if (r[i].flight_num > max) {
            max = r[i].flight_num;
        }
    }
    for (int i = 0; i < n; i++) {
        if (r[i].flight_num==max) {
            temp = r[i];
```

```
            r[i] = newelement;
            for (int j = i+1; j < n+1; j++){
                sdvig = r[j];
                r[j] = temp;
                temp = sdvig;
                i++;
            }
        }
    }
}

void deleteLeft(Airport r[], Airport ar[], int n, int time) {
    int j = 0, k = 0, x;
    for (int i = 0; i < n; i++) {
        if (time - r[i].departure_time >= 0) {
            ar[j] = r[i];
            j++;
        }
        else {
            r[k] = r[i];
            k++;
        }
    }
    cout << "Желаете вывести таблицу?" << endl
        << "1-Архивная таблица" << endl
        << "2-Измененное табло" << endl
        << "Другое значение - выход" << endl;
    cin >> x;
    switch (x)
    {
    case 1: {
        cout << endl << "----Архивная таблица------" << endl;
        for (int i = 0; i < j; i++) {
            cout << "----Информация по рейсу №" << i + 1 << "----" << endl;
            outBoard(ar, i);
            cout << "--------------------------------" << endl;
        }
        break;
    }
    case 2: {
        for (int i = 0; i < k; i++) {
            cout << "----Информация по рейсу №" << i + 1 << "----" << endl;
            outBoard(r, i);
            cout << "--------------------------------" << endl;
        }
        break;
    }
    default: {
        break;
    }
    }
}

void outputReady(int time, Airport r[], int n) {
    for (int i = 0; i < n; i++) {
        if (r[i].departure_time - time == 2) {
            cout << "----Рейс №" << i + 1 << " готов к вылету по расписанию----" << endl;
            outputOneRecord(r[i]);
            cout << "-----------------------------------------------" << endl;
        }
        else {
            cout << "Нет рейсов, которые готовы вылететь." << endl;
        }
    }
```

```
}

void outputGivenDestination(char s[], int n, Airport r[]) {
    for (int i = 0; i < n; i++) {
        if (strcmp(r[i].destination,s)==0) {
            cout << "----Рейс №" << i + 1 << " вылетает в заданный пункт назначения----" <<
endl;
            outputOneRecord(r[i]);
            cout << "-----------------------------------------------" << endl;
        }
    }
}

int main()
{
    setlocale(LC_ALL, "rus");
    int n, y;
    Airport info[max_size];
    cout << "Введите количество рейсов: "; cin >> n;
    if (n > 0 && n <= max_size) {
        for (int i = 0; i < n; i++) {
            insertRecord(info, i);
        }
    }
    else {
        cout << "n должно быть больше нуля";
        exit(0);
    }
    for (int i = 0; i < n; i++) {
        cout << "----Информация по рейсу №" << i + 1 << "----" << endl;
        outBoard(info, i);
        cout << "--------------------------------" << endl;
    }
    povtor: // метка для goto
    cout << "Введите номер задачи:" << endl
        << "1)Вставить информацию по новому рейсу в таблицу перед рейсом с большим
номером." << endl
        << "2)Удалить информацию о вылетевшем рейсе и сохранить ее в архивной таблице." <<
endl
        << "3)Вывести рейсы, готовые к вылету по расписанию (за 2 часа до времени вылета)."
<< endl
        << "4)Сформировать список номеров рейсов, вылетающих в заданный пункт назначения."
<< endl;
    cout << "Номер: "; cin >> y;
    switch (y) {
    case 1: {
        Airport newflight;
        cout << "Введите информацию по новому рейсу" << endl;
        inputOneRecord(newflight);
        insertRecordBeforeMax(info, n, newflight);
        cout << endl << "-----Новое табло-----" << endl;
        for (int i = 0; i < n + 1; i++) {
            cout << "----Информация по рейсу №" << i + 1 << "----" << endl;
            outBoard(info, i);
            cout << "--------------------------------" << endl;
        }
        int q;
        cout << "Продолжить? 1 - да 0 - нет: ";
        cin >> q;
        if (q == 1) {
            goto povtor; //151 строка в main
        }
        else {
            exit(0);
```

```
        }
        break;
    }
    case 2: {
        int current_time;
        cout << "Введите текущее время (в часах): "; cin >> current_time;
        Airport archive[max_size];
        deleteLeft(info, archive, n, current_time);
        int q;
        cout << "Продолжить? 1 - да 0 - нет: ";
        cin >> q;
        if (q == 1) {
            goto povtor; //151 строка в main
        }
        else {
            exit(0);
        }
        break;
    }
    case 3: {
        int current_time;
        cout << "Введите текущее время (в часах): "; cin >> current_time;
        outputReady(current_time, info, n);
        int q;
        cout << "Продолжить? 1 - да 0 - нет: ";
        cin >> q;
        if (q == 1) {
            goto povtor; //151 строка в main
        }
        else {
            exit(0);
        }
        break;
    }
    case 4: {
        char given_destination[30];
        cout << "Введите пункт назначения: "; cin >> given_destination;
        outputGivenDestination(given_destination, n, info);
        int q;
        cout << "Продолжить? 1 - да 0 - нет: ";
        cin >> q;
        if (q == 1) {
            goto povtor; //151 строка в main
        }
        else {
            exit(0);
        }
        break;
    }
    default: {
        cout << "Нет такой задачи, соответсвующей введенному вами номером" << endl;
        int q;
        cout << "Продолжить? 1 - да 0 - нет: ";
        cin >> q;
        if (q == 1) {
            goto povtor; //151 строка в main
        }
        else {
            exit(0);
        }
        break;
    }
    }
    return 0;
```

}

dinamic.cpp

```
#include <iostream>
using namespace std;
int max_size = 100;

struct Airport {
    char destination[30]; //Пункт назначения
    int flight_num; //Номер рейса
    char departure_day[10]; //День вылета
    int departure_time; //Время вылета
    int arrival_time; //Время прибытия
    int free_seats; // Кол-во свободных мест
    int delay; // Информация о задержке вылета в часах
};

void inputOneRecord(Airport& r) {
    cout << "----Вставка инфорармации по рейсу----" << endl;
    cout << "Пункт назначения: "; cin >> r.destination;
    cout << "Номер рейса: "; cin >> r.flight_num;
    cout << "День вылета: "; cin >> r.departure_day;
    cout << "Время вылета: "; cin >> r.departure_time;
    cout << "Время прибытия: "; cin >> r.arrival_time;
    cout << "Кол-во свободных мест: "; cin >> r.free_seats;
    cout << "Информация о задержке вылета в часах: "; cin >> r.delay;
    cout << "-------------------------------------" << endl;
}

void outputOneRecord(Airport r) {
    cout << "Пункт назначения: "; cout << r.destination << endl;
    cout << "Номер рейса: "; cout << r.flight_num << endl;
    cout << "День вылета: "; cout << r.departure_day << endl;
    cout << "Время вылета: "; cout << r.departure_time << endl;
    cout << "Время прибытия: "; cout << r.arrival_time << endl;
    cout << "Кол-во свободных мест: "; cout << r.free_seats << endl;
    cout << "Информация о задержке вылета в часах: "; cout << r.delay << endl;
}

void insertRecord(Airport* r, int i) {
    inputOneRecord(r[i]);
}

void outBoard(Airport* r, int i) {
    outputOneRecord(r[i]);
}

void insertRecordBeforeMax(Airport r[], int n, Airport newelement) {
    int max = 0;
    Airport temp, sdvig;
    for (int i = 0; i < n; i++) {
        if (r[i].flight_num > max) {
            max = r[i].flight_num;
        }
    }
    for (int i = 0; i < n; i++) {
        if (r[i].flight_num == max) {
            temp = r[i];
            r[i] = newelement;
            for (int j = i + 1; j < n + 1; j++) {
                sdvig = r[j];
                r[j] = temp;
                temp = sdvig;
```

```
                i++;
            }
        }
    }
}

void deleteLeft(Airport r[], Airport ar[], int n, int time) {
    int j = 0, k = 0, x;
    for (int i = 0; i < n; i++) {
        if (time - r[i].departure_time >= 0) {
            ar[j] = r[i];
            j++;
        }
        else {
            r[k] = r[i];
            k++;
        }
    }
    cout << "Желаете вывести таблицу?" << endl
        << "1-Архивная таблица" << endl
        << "2-Изменное табло" << endl
        << "Другое значение - выход" << endl;
    cin >> x;
    switch (x)
    {
    case 1: {
        cout << endl << "----Архивная таблица------" << endl;
        for (int i = 0; i < j; i++) {
            cout << "----Информация по рейсу №" << i + 1 << "----" << endl;
            outBoard(ar, i);
            cout << "--------------------------------" << endl;
        }
        break;
    }
    case 2: {
        for (int i = 0; i < k; i++) {
            cout << "----Информация по рейсу №" << i + 1 << "----" << endl;
            outBoard(r, i);
            cout << "--------------------------------" << endl;
        }
        break;
    }
    default: {
        break;
    }
    }
}

void outputReady(int time, Airport r[], int n) {
    for (int i = 0; i < n; i++) {
        if (r[i].departure_time - time == 2) {
            cout << "----Рейс №" << i + 1 << " готов к вылету по расписанию----" << endl;
            outputOneRecord(r[i]);
            cout << "------------------------------------------------" << endl;
        }
    }
}

void outputGivenDestination(char s[], int n, Airport r[]) {
    for (int i = 0; i < n; i++) {
        if (strcmp(r[i].destination, s) == 0) {
            cout << "----Рейс №" << i + 1 << " вылетает в заданный пункт назначения----" <<
endl;
            outputOneRecord(r[i]);
```

```
            cout << "-----------------------------------------------" << endl;
        }
    }
}

int main()
{
    setlocale(LC_ALL, "rus");
    int n, y;
    Airport* info=new Airport[max_size];
    cout << "Введите количество рейсов: "; cin >> n;
    if (n > 0 && n <= max_size) {
        for (int i = 0; i < n; i++) {
            insertRecord(info, i);
        }
    }
    else {
        cout << "n должно быть больше нуля";
        exit(0);
    }
    for (int i = 0; i < n; i++) {
        cout << "----Информармация по рейсу №" << i + 1 << "----" << endl;
        outBoard(info, i);
        cout << "--------------------------------" << endl;
    }
    povtor:
    cout << "Введите номер задачи:" << endl
        << "1)Вставить информацию по новому рейсу в таблицу перед рейсом с большим
номером." << endl
        << "2)Удалить информацию о вылетевшем рейсе и сохранить ее в архивной таблице." <<
endl
        << "3)Вывести рейсы, готовые к вылету по расписанию (за 2 часа до времени вылета)."
<< endl
        << "4)Сформировать список номеров рейсов, вылетающих в заданный пункт назначения."
<< endl;
    cout << "Номер: "; cin >> y;
    switch (y) {
    case 1: {
        Airport newflight;
        cout << "Введите информацию по новому рейсу" << endl;
        inputOneRecord(newflight);
        insertRecordBeforeMax(info, n, newflight);
        cout << endl << "-----Новое табло-----" << endl;
        for (int i = 0; i < n + 1; i++) {
            cout << "----Информация по рейсу №" << i + 1 << "----" << endl;
            outBoard(info, i);
            cout << "--------------------------------" << endl;
        }
        int q;
        cout << "Продолжить? 1 - да 0 - нет: ";
        cin >> q;
        if (q == 1) {
            goto povtor; //148 строка в main
        }
        else {
            exit(0);
        }
        break;
    }
    case 2: {
        int current_time;
        cout << "Введите текущее время (в часах): "; cin >> current_time;
        Airport* archive= new Airport[max_size];
        deleteLeft(info, archive, n, current_time);
```

```
            delete[] archive;
            int q;
            cout << "Продолжить? 1 - да 0 - нет: ";
            cin >> q;
            if (q == 1) {
                goto povtor; //148 строка в main
            }
            else {
                exit(0);
            }
            break;
        }
        case 3: {
            int current_time;
            cout << "Введите текущее время (в часах): "; cin >> current_time;
            outputReady(current_time, info, n);
            int q;
            cout << "Продолжить? 1 - да 0 - нет: ";
            cin >> q;
            if (q == 1) {
                goto povtor; //148 строка в main
            }
            else {
                exit(0);
            }
            break;
        }
        case 4: {
            char given_destination[30];
            cout << "Введите пункт назначения: "; cin >> given_destination;
            outputGivenDestination(given_destination, n, info);
            int q;
            cout << "Продолжить? 1 - да 0 - нет: ";
            cin >> q;
            if (q == 1) {
                goto povtor; //148 строка в main
            }
            else {
                exit(0);
            }
            break;
        }
        default: {
            cout << "Нет такой задачи, соответсвующей введенному вами номером" << endl;
            int q;
            cout << "Продолжить? 1 - да 0 - нет: ";
            cin >> q;
            if (q == 1) {
                goto povtor; //148 строка в main
            }
            else {
                exit(0);
            }
            break;
        }
        }
        delete[] info;
        return 0;
}
```

vector.cpp

```
#include <iostream>
#include <vector>
using namespace std;
```

```
int max_size = 100;

struct Airport {
    char destination[30]; //Пункт назначения
    int flight_num; //Номер рейса
    char departure_day[10]; //День вылета
    int departure_time; //Время вылета
    int arrival_time; //Время прибытия
    int free_seats; // Кол-во свободных мест
    int delay; // Информация о задержке вылета в часах
};

void inputOneRecord(Airport& r) {
    cout << "----Вставка инфорармации по рейсу----" << endl;
    cout << "Пункт назначения: "; cin >> r.destination;
    cout << "Номер рейса: "; cin >> r.flight_num;
    cout << "День вылета: "; cin >> r.departure_day;
    cout << "Время вылета: "; cin >> r.departure_time;
    cout << "Время прибытия: "; cin >> r.arrival_time;
    cout << "Кол-во свободных мест: "; cin >> r.free_seats;
    cout << "Информация о задержке вылета в часах: "; cin >> r.delay;
    cout << "-------------------------------------" << endl;
}

void outputOneRecord(Airport r) {
    cout << "Пункт назначения: "; cout << r.destination << endl;
    cout << "Номер рейса: "; cout << r.flight_num << endl;
    cout << "День вылета: "; cout << r.departure_day << endl;
    cout << "Время вылета: "; cout << r.departure_time << endl;
    cout << "Время прибытия: "; cout << r.arrival_time << endl;
    cout << "Кол-во свободных мест: "; cout << r.free_seats << endl;
    cout << "Информация о задержке вылета в часах: "; cout << r.delay << endl;
}

void insertRecord(vector<Airport>&r, int i) {
    inputOneRecord(r[i]);
}

void outBoard(vector<Airport>& r, int i) {
    outputOneRecord(r[i]);
}

void insertRecordBeforeMax(vector<Airport>& r, int n, Airport newelement) {
    int max = 0;
    Airport temp, sdvig;
    for (int i = 0; i < n; i++) {
        if (r[i].flight_num > max) {
            max = r[i].flight_num;
        }
    }
    for (int i = 0; i < n; i++) {
        if (r[i].flight_num == max) {
            temp = r[i];
            r[i] = newelement;
            for (int j = i + 1; j < n + 1; j++) {
                sdvig = r[j];
                r[j] = temp;
                temp = sdvig;
                i++;
            }
        }
    }
}
```

```
void deleteLeft(vector<Airport>& r, vector<Airport>& ar, int n, int time) {
    int j = 0, k = 0, x;
    for (int i = 0; i < n; i++) {
        if (time - r[i].departure_time >= 0) {
            ar[j] = r[i];
            j++;
        }
        else {
            r[k] = r[i];
            k++;
        }
    }
    cout << "Желаете вывести таблицу?" << endl
        << "1-Архивная таблица" << endl
        << "2-Изменное табло" << endl
        << "Другое значение - выход" << endl;
    cin >> x;
    switch (x)
    {
    case 1: {
        cout << endl << "----Архивная таблица------" << endl;
        for (int i = 0; i < j; i++) {
            cout << "----Информация по рейсу №" << i + 1 << "----" << endl;
            outBoard(ar, i);
            cout << "--------------------------------" << endl;
        }
        break;
    }
    case 2: {
        for (int i = 0; i < k; i++) {
            cout << "----Информация по рейсу №" << i + 1 << "----" << endl;
            outBoard(r, i);
            cout << "--------------------------------" << endl;
        }
        break;
    }
    default: {
        break;
    }
    }
}

void outputReady(int time, vector<Airport>& r, int n) {
    for (int i = 0; i < n; i++) {
        if (r[i].departure_time - time == 2) {
            cout << "----Рейс №" << i + 1 << " готов к вылету по расписанию----" << endl;
            outputOneRecord(r[i]);
            cout << "-----------------------------------------------" << endl;
        }
        else {
            cout << "Нет рейсов, которые готовы вылететь." << endl;
        }
    }
}

void outputGivenDestination(char s[], int n, vector<Airport>& r) {
    for (int i = 0; i < n; i++) {
        if (strcmp(r[i].destination, s) == 0) {
            cout << "----Рейс №" << i + 1 << " вылетает в заданный пункт назначения----" <<
endl;
            outputOneRecord(r[i]);
            cout << "-----------------------------------------------" << endl;
        }
    }
```

```
}

int main()
{
    setlocale(LC_ALL, "rus");
    int n, y;
    cout << "Введите количество рейсов: "; cin >> n;
    vector<Airport> info(max_size);
    if (n > 0) {
        for (int i = 0; i < n; i++) {
            insertRecord(info, i);
        }
    }
    else {
        cout << "n должно быть больше нуля";
        exit(0);
    }
    for (int i = 0; i < n; i++) {
        cout << "----Информация по рейсу №" << i + 1 << "----" << endl;
        outBoard(info, i);
        cout << "--------------------------------" << endl;
    }
    povtor: // метка для goto
    cout << "Введите номер задачи:" << endl
        << "1)Вставить информацию по новому рейсу в таблицу перед рейсом с большим
номером." << endl
        << "2)Удалить информацию о вылетевшем рейсе и сохранить ее в архивной таблице." <<
endl
        << "3)Вывести рейсы, готовые к вылету по расписанию (за 2 часа до времени вылета)."
<< endl
        << "4)Сформировать список номеров рейсов, вылетающих в заданный пункт назначения."
<< endl;
    cout << "Номер: "; cin >> y;
    switch (y) {
    case 1: {
        Airport newflight;
        cout << "Введите информацию по новому рейсу" << endl;
        inputOneRecord(newflight);
        insertRecordBeforeMax(info, n, newflight);
        cout << endl << "-----Новое табло-----" << endl;
        for (int i = 0; i < n + 1; i++) {
            cout << "----Информация по рейсу №" << i + 1 << "----" << endl;
            outBoard(info, i);
            cout << "--------------------------------" << endl;
        }
        int q;
        cout << "Продолжить? 1 - да 0 - нет: ";
        cin >> q;
        if (q == 1) {
            goto povtor; //152 строка в main
        }
        else {
            exit(0);
        }
        break;
    }
    case 2: {
        int current_time;
        cout << "Введите текущее время (в часах): "; cin >> current_time;
        vector<Airport> archive(max_size);
        deleteLeft(info, archive, n, current_time);
        int q;
        cout << "Продолжить? 1 - да 0 - нет: ";
        cin >> q;
```

```
            if (q == 1) {
                goto povtor; //152 строка в main
            }
            else {
                exit(0);
            }
            break;
        }
        case 3: {
            int current_time;
            cout << "Введите текущее время (в часах): "; cin >> current_time;
            outputReady(current_time, info, n);
            int q;
            cout << "Продолжить? 1 - да 0 - нет: ";
            cin >> q;
            if (q == 1) {
                goto povtor; //152 строка в main
            }
            else {
                exit(0);
            }
            break;
        }
        case 4: {
            char given_destination[30];
            cout << "Введите пункт назначения: "; cin >> given_destination;
            outputGivenDestination(given_destination, n, info);
            int q;
            cout << "Продолжить? 1 - да 0 - нет: ";
            cin >> q;
            if (q == 1) {
                goto povtor; //152 строка в main
            }
            else {
                exit(0);
            }
            break;
        }
        default: {
            cout << "Нет такой задачи, соответсвующей введенному вами номером" << endl;
            int q;
            cout << "Продолжить? 1 - да 0 - нет: ";
            cin >> q;
            if (q == 1) {
                goto povtor; //152 строка в main
            }
            else {
                exit(0);
            }
            break;
        }
        }
        return 0;
}
```

3 ВЫВОДЫ

В ходе работы на языке C++ было разработано приложение, реализующее задачу в соответствии с персональным вариантом в 3 вариациях: на статическом массиве, динамическом массиве и векторе.