Лабораторная работа № 8

**Фильтрация данных**

**Цель:** изучить прием фильтрации, применить фильтрацию к созданной структуре данных.

**ТЕОРЕТИЧЕСКАЯ ЧАСТЬ**

#### Фильтр - это быстрый и легкий способ поиска подмножества данных и работы с ними в списке. В отфильтрованном списке отображаются только строки, отвечающие условиям. В отличие от сортировки, фильтр не меняет порядок записей в списке.

Виды фильтрации:

* По значению
* По типу
* По диапазону значений
* И т.д.

## Объявление структуры

Прежде чем будет создан объект структуры, должен быть определен ее формат. Это делается посредством объявления структуры –установка структурного шаблона ***struct***. Объявление структуры позволяет понять, переменные какого типа она содержит. Переменные, составляющие структуру, называются ее членами или элементами, или полями. В общем случае все члены структуры должны быть логически связаны друг с другом.

***Член структуры*** *–* это переменная, которая является частью структуры.

Для объявления структуры используется ***структурный шаблон***, который является основной схемой, описывающей содержание структуры.

Общий формат объявления:

*struct имя\_ типа\_ структуры {*

*тип имя\_элемента 1;*

*тип имя\_элемента 2;*

*тип имя элемента 3;*

*…*

*тип имя\_элемента М;*

*}* ;

Например, объявим структуру студент, содержащую элементы – фамилия студента и его средний балл, название группы, например, uir1. Для этот создадим шаблон:

*struct stud*

*{*

*char name[20];*

*char group\_name[20];*

*float bal;*

*}****;***

Первым в шаблоне стоит ключевое слово ***struct***. Оно определяет, что все, что стоит за ним, является структурой. Далее следует необязательный «тег» (*имя типа структуры*) – слово *stud*, являющееся сокращенной меткой, которую можно использовать для ссылки на эту структуру. *Имя структуры* – это ее ***спецификатор типа***.

*Компоненты структуры (список элементов структуры)* заключены в парные фигурные скобки. Каждый элемент определяется своим собственным описанием. Например, элемент *name* является символьным массивом, состоящим из 20 элементов. *Элементы могут быть данными любого типа, включая другие структуры*.

Например, объявим структуру студенческой группы ***group***, содержащую элементы – уникальный идентификатор группы *id*, название группы *group\_name* и массив объектов типа ***stud***, который должен содержать данные о студентах группы. Создадим шаблон:

*struct stud*

*{*

*int id\_group;*

*char name[20];*

*float bal;*

*};*

***struct group***

*{*

*int id\_group;*

*char group\_name[20];*

***stud st[30];***

*}****;***

Все данные в структуре (*struct*) пишутся в фигурных скобках, и в конце ставится запятая с точкой (**;**), завершающая определение шаблона.

***Обобщая:*** Любая структура должна начинаться с ключевого слова – ***struct***, которое сообщает компилятору, что будет создана структура. Объявление структуры представляет собой ***инструкцию***, поэтому оно заканчивается точкой с запятой (;).

Как видно из примера в *структурах* ***stud*** и ***group*** находятся данные различных типов, но они объединены в логическую связь. Данные в структуре должны иметь уникальные имена, но в различных структурах можно использовать одинаковые названия.

Можно разместить шаблон за пределами любой функции (вне), или внутри определения функции. Если установили шаблон внутри функции, то он может использоваться только внутри этой функции. Если вне, то шаблон доступен всем функциям программы, следующим за его определением. Установка шаблона не вызывает никаких действий в программе. Память для него не выделяется.

## Создание объекта

Слово «структура» используется двояко. Во-первых, в смысле «структурного шаблона». Шаблон является схемой без содержания; он сообщает компилятору, как делать что-либо, но не вызывает никаких действий в программе. Так при объявлении структуры *stud* в действительности не было создано ни одной переменной. Был определен лишь формат данных. Таким образом, структура, которая создана выше, не занимает в памяти компьютера места, так как, на самом деле, просто создали ***пользовательский тип данных***, аналогичный int, float.

Следующий шаг заключается в создании «структурной переменной»; это и есть второй смысл слова структура. Объявление структурных переменных приводит к выделению памяти для компонент структуры, куда можно записать данные или откуда можно прочитать их. Чтобы с помощью созданного структурного шаблона объявить реальную переменную (т.е. физический объект), можно воспользоваться несколькими способами.

1. Нужно записать инструкцию, объявляющую простую переменную, массив структур, указатель на структуру:

*void main(){*

*struct stud my, mas[5], \*pmy;*

*}*

Объявление ***структуры*** ничем не отличается от объявления любого типа данных в языке С (С++). На основании этого оператора компилятор создает структурную переменную типа *struct stud* с именем ***my***, массив ***mas***, состоящий из 5 элементов, типа *struct stud* и указатель *\*pmy* на переменную *struct stud*.

***Обобщая:*** при определении структуры определяется новый тип данных, но он не будет реализован до тех пор, пока не будет объявлена переменная того типа, который уже реально существует.

При объявлении структурной переменной С (С++) автоматически выделит объем памяти, достаточный для хранения всех членов структуры. В примере, переменная my будет размещена в памяти компьютера (в предположении, что name – символьный массив 20 элементов – 20 байт, char-значение *group\_name* – символьный массив 20 элементов – 20 байт, а float-значение *bal* – 4).

Одновременно с определением структуры можно сразу после объявления структуры объявить одну или несколько переменных, как показано в следующем примере.

Общий формат объявления структуры в этом случае будет выглядеть так.

*struct имя\_ типа\_ структуры {*

*тип имя\_элемента1;*

*тип имя\_элемента2;*

*тип имя элемента 3;*

*тип имя\_элементаМ;*

*}* ***структурные\_переменные****;*

Объявление переменных данного типа:

*struct stud {*

*char name[20];*

*char group\_name[20];*

*float bal;*

*}* ***my, mas[5], \*pmy;***

Этот фрагмент кода определяет структурный тип *stud* и объявляет переменные ***my, mas[5]*** и указатель ***\*pmy*** этого типа. Важно понимать, что каждая структурная переменная содержит собственные копии членов структуры. Например, поле *name* структуры *my* изолировано от поля *name* структуры *mas[5]*. Следовательно, изменения, вносимые в одно поле, никак не влияют на содержимое другого поля.

***Пример 1:*** Определение размера памяти, которая выделяется для структурной переменной.

#include <iostream>

struct stud {

char name[20];

char group\_name[20];

float bal;

}**;**

void main()

{

struct stud my;

int size;

size = sizeof(my); // size = 28

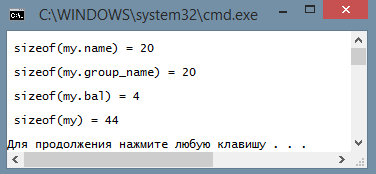
std::cout << "\n sizeof(my.name) = " << sizeof(my.name) << std::endl;

std::cout << "\n sizeof(my.group\_name) = " << sizeof(my.group\_name) << std::endl;

std::cout << "\n sizeof(my.bal) = " << sizeof(my.bal) << std::endl;

std::cout << "\n sizeof(my) = " << size << std::endl << std::endl;

}



1. Установить структурный шаблон с помощью макроопределения:

#include <iostream>

#include <Windows.h>

#define STUD struct stud

STUD{

char name[20];

char group\_name[20];

float bal;

}**;**

void main()

{

**//Объявить переменные:**

STUD my, mas[3], \*pmy;

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

setlocale(LC\_ALL, "Russian");

std::cout << "Введите имя, группу, средний балл" << std::endl << std::endl;

for (int i = 0; i < 3; i++)

{

std::cout << i+1 << std::endl;

std::cin >> mas[i].name;

std::cin >> mas[i].group\_name;

std::cin >> mas[i].bal;

}

std::cout << std::endl;

for (int i = 0; i < 3; i++)

{

std::cout << mas[i].name << '\t';

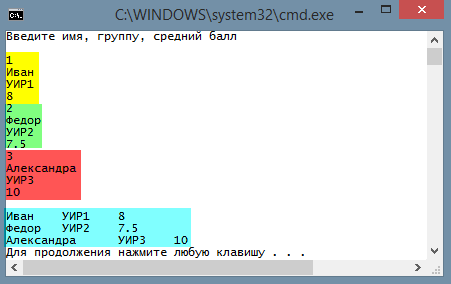
std::cout << mas[i].group\_name << '\t';

std::cout << mas[i].bal << std::endl;

}

}

Результат выполнения кода представлен на рисунке.



1. Одновременно с установкой шаблона (если на данную структуру больше не будет ссылок) можно сразу после объявления структуры объявить одну или несколько переменных.

Процесс определения структурного шаблона и процесс определения структурной переменной объединяется в один этап. Объединение шаблона и определений переменных является именно тем случаем, когда не нужно использовать *имя типа структуры*:

struct {**/\* без имени типа структуры \*/**

char name[20];

char group\_name[20];

float bal;

} **my;**

Форма с именем типа структуры удобнее, если используется структурный шаблон более одного раза.

1. Ввести новый тип данных (***stud***)-структура определенного вида с помощью инструкции ***typedef.***

typedef struct stud {

char name[20];

char group\_name[20];

float bal;

}**;**

Объявить переменные нового типа:

***stud*** my, mas[5], \*pmy;

Если программа достаточно объемна, представляется более удобным четвертый способ.

## Инструкция typedef

Язык С (C++) позволяет определять имена новых типов данных с помощью ключевого слова ***typedef***. На самом деле здесь не создается новый тип данных, а определяется новое имя существующему типу. Объявления ***typedef*** можно использовать для создания более коротких и содержательных имен стандартным типам данных, определенных в языке, или для типов, которые объявили пользователи. Пример:

typedef int integer;

где int – это тип, которому присваиваем новое имя, а integer – это то обозначение, которое будет использоваться вместо int.

Имена *typedef* разделяют пространство имен с обычными идентификаторами. Поэтому программа может иметь имя typedef и идентификатор локальной области с одним и тем же именем.

***Пример:*** Имена структур и другие идентификаторы находятся в различных пространствах имен. Используем одинаковые имена для объявления структуры и обычной переменной.

struct stud

{

char name[20];

char group\_name[20];

float bal;

};

int stud;

В этом фрагменте кода объявляется структура с именем stud и переменная типа int с тем же самым именем stud. Эти объявления не конфликтует друг с другом, так как перед именем структуры обязательно должно следовать ключевое слово ***struct***.

В **C**-**программах** надо указывать ключевые слова *struct* или *enum* перед именем структуры или перечисления. Часто программисты забывают указать эти слова, что приводит к появлению ошибки компиляции. Поэтому инструкция *typedef* упрощает написание кода, позволяя не писать эти ключевые слова перед именем структуры или перечисления.

struct stud {

char name[20];

char group\_name[20];

float bal;

};

typedef struct stud student;

void main()

{

//Объявить переменные:

student my, mas[3], \*pmy;

}

Большинство программистов используют ***typedef*** сразу при объявлении структуры, что заметно уменьшает код. Например:

*typedef struct {*

*char name[20];*

*char group\_name[20];*

*float bal;*

*} stud;*

Т.е. при объявлении вместо двух ключевых слов – ***typedef struct*** как в первом примере, сразу создаем тип *stud*. Теперь для объявления переменной созданного типа достаточно использовать одно слово.

Приведем пример использования ключевого слова *typedef* для базовый типов:

|  |  |
| --- | --- |
| typedef char Literal;  const Literal x = 'c';  //означает: const char x='c'; | typedef **int** Goda;  Goda youGoda;  //означает: int youGoda; |

Пример использования ключевого слова *typedef* для структур:

*struct stud {*

*char name[20];*

*char group\_name[20];*

*float bal;*

*};*

*void main()*

*{*

*typedef struct stud uir;*

*uir a, \*p, b[5];*

*}*

Здесь заменили *struct stud* синонимом *uir*.

***Обобщение:*** *typedef* позволяет объявить синоним любой переменной (для встроенного либо пользовательского типа данных). В отличие от объявлений ***class***,***struct***,***union***и***enum****,* объявления ***typedef*** не вводит новый тип – они вводят новые имена для уже существующих.

## Операции со структурами

Операции со структурами:

1. присваивание полю структуры значение того же типа;
2. можно получить адрес структуры. Операция взятия адреса (&);
3. можно обращаться к любому полю структуры;
4. для того, что бы определить размер структуры можно использовать операцию sizeof().

# ПЕРЕДАЧА СТРУКТУР В ФУНКЦИЮ

**Структура** является **типом данных** созданным пользователем, к ней можно применять такие же операции, как и к встроенным типам. Рассмотрим работу структур (**struct**) с функциями.

1. Структуру можно передавать в функцию целиком:

#include <iostream>

#include <stdio.h>

using namespace std;

struct stud {

char name[20];

char group\_name[20];

float bal;

} **my[]** ={ "Липницкая И.А.", "УИР 2", 9.85,

"Скоморощенко К.С.", "УИР 1", 9.25,

"Щурок Е.А.", "УИР 3", 8.25,

}**;**

void print\_stud(**struct stud str**); //Прототип функции вывода

void main()

{

setlocale(LC\_ALL, "Russian");

for (int i = 0; i < 3; i++)

{

print\_stud(my[i]);

}

}

void print\_stud(struct stud str)

{

printf("ФИО : %15s\t", str.name);

printf("Группа : %s \t", str.group\_name);

printf(" Ср. балл : %.2f\n", str.bal);

}

1. Можно передать указатель на структуру.

void print\_stud(struct stud\* str); //Прототип функции вывода

void vvod\_stud(struct stud\* str, int n); //Прототип функции ввода

Код программы:

#include <iostream>

#include <stdio.h>

using namespace std;

struct stud {

char name[20];

char group\_name[20];

float bal;

};

void vvod\_stud(struct stud\* str, int n); // Размерность массива int n

void print\_stud(struct stud\* str);

void main()

{

setlocale(LC\_ALL, "Russian");

stud \*pmy = new stud [3] ;

vvod\_stud(pmy, 3);

/\*}\*/

print\_stud(pmy);

cout << endl;

}

void print\_stud(struct stud\* str)

{

for (int i = 0; i < 3; i++)

{

printf("ФИО : %15s\t", str->name);

printf("Группа : %s \t", str->group\_name);

printf(" Ср. балл : %.2f\n", str->bal);

str++;

}

}

void vvod\_stud(struct stud\* str, int n)

{

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

{

printf("\nВведите \n");

printf("ФИО : ");

gets(str[i].name);

printf("Группа : ");

gets(str[i].group\_name);

printf("Ср. балл : ");

cin >> str[i].bal;

cin.ignore();

}

}

1. Можно передать элементы структуры по отдельности:

void stud\_out(char name[20], float f)

{

printf("имя: %s; балл: %.1f\n", name, f);

}

void main()

{

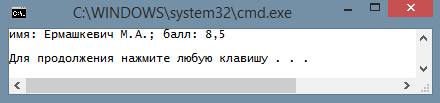
setlocale(LC\_ALL, "Russian");

struct stud a = { "Ермашкевич М.А.", "УИР 2", 8.5 };

stud\_out(a.name, a.bal);

cout << endl;

}



**Повторение:** При передаче самой структуры или ее элементов передаются значения. Т.е. при изменении элементов структуры в функции, настоящие элементы останутся нетронутыми. Для того, чтобы передавались настоящие элементы, в функцию надо передавать структуру или с помощью указателей, или с помощью ссылок.

***Пример 7:*** Создать структуру *Студент*: курс, группа, ФИО, какие предметы проходят, оценки за экзамен. Объявить динамический массив структур и заполнить его. Вывести на экран ведомость успеваемости. Передать структуру в функцию (ввод данных в массив структур).

Когда пользователь вводит данные в ответ на операцию извлечения, то эти данные помещаются в буфер ***std::cin***. Буфер данных – это просто часть памяти, зарезервированная для временного хранения данных, когда они перемещаются из одного места в другое. В этом случае буфер используется для хранения пользовательского ввода, пока он находится в режиме ожидании выделения для него переменных. В некоторых случаях возникает необходимость пропустить часть символов строки от начала до достижения конца строки (EOL) или конца файла (EOF). Именно этому и отвечает функция ***ignore()***. Она принимает два параметра: число пропускаемых символов и символ разделения. Например, вызов функции ignore(80, '\n') приведет к пропуску 80 символов, если ранее не будет найден символ начала новой строки. Последний затем будет удален из буфера, после чего функция ignore() завершит свою работу.

#include <stdlib.h>

#include <iostream>

#include <Windows.h>

using namespace std;

struct Stud1 //Объявление шаблона структуры Stud1

{

int Kurs;

int Gruppa;

char FIO[20];

char Predmet[30];

int Ocenka;

};

void GetData(Stud1 \*M, int N); //Прототип функции

void main() //Главная функция

{

setlocale(LC\_ALL, "Russian"); //Работа с кириллицей

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

system("CLS"); //Очистка экрана

int i = 0;

int N;

cout << "Введите N: ";

cin >> N;

Stud1 \*M1 = new Stud1[N]; //Объявление динамического массива

// из N структур типа Stud1

GetData(M1, N); //Ввод данных в массив структур

cout << "\n";

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

{

if (M1[i].Ocenka <= 3)

{

cout << "Студент " << M1[i].FIO << " не сдал экзамен по предмету " << M1[i].Predmet << " : " << M1[i].Ocenka << endl << endl;

}

}

delete[]M1;

system("PAUSE");

}

void GetData(Stud1 \*M, int N) //Функция заполняет данными структуру Stud1

{

cin.ignore(); // Удаляем лишние значения

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

{

setlocale(LC\_ALL, "Russian");

cout << "\n";

cout << "Курс: ";

cin >> M[i].Kurs;

cout << "Группа: ";

cin >> M[i].Gruppa;

cin.ignore();

cout << "ФИО: ";

cin.getline(M[i].FIO, 20);

//gets(M[i].FIO); //Второй вариант ввода данных

cout << "Предмет: ";

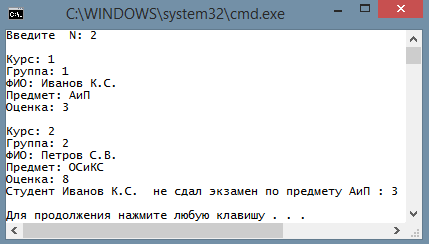
cin.getline(M[i].Predmet, 30);

cout << "Оценка: ";

cin >> M[i].Ocenka;

}

}



***Пример 7:*** Разработать программу, которая осуществляет работу с таблицей данных, содержащей информацию о студентах: №,Фамилия, Год рождения, Факультет. В программе использована структура. Проанализировать работу главной функции и функций, входящих в программу.

*# include <iostream>*

*# define str\_len 30*

*# define size 30*

*using namespace std;*

*int choice;*

*struct Student*

*{*

*char name[str\_len];*

*int year\_of\_birth;*

*char department[5];*

*};*

*struct Student list\_of\_student[size];*

*struct Student bad;*

*int current\_size = 0;*

*void enter\_new()*

*{*

*cout << "Ввод информации" << endl;*

*if (current\_size < size)*

*{*

*cout << "Строка номер ";*

*cout << current\_size + 1;*

*cout << endl << " \n Фамилия: " ;*

*cin >> list\_of\_student[current\_size].name;*

*cout << "\n Год рождения: " ;*

*cin >> list\_of\_student[current\_size].year\_of\_birth;*

*cout << "\n Факультет: " ;*

*cin >> list\_of\_student[current\_size].department;*

*current\_size++;*

*cout << endl;*

*}*

*else*

*cout << "Введено максимальное кол–во строк";*

*exit;*

*}*

*void del()*

*{*

*int d;*

*cout << "\nНомер строки, которую надо удалить (для удаления всех строк нажать 99)" << endl;*

*cin >> d;*

*if (d != 99)*

*{*

*for (int de1 = (d – 1); de1 < current\_size; de1++)*

*list\_of\_student[de1] = list\_of\_student[de1 + 1];*

*current\_size = current\_size – 1;*

*}*

*if (d == 99)*

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

*list\_of\_student[i] = bad;*

*exit;*

*}*

*void change()*

*{*

*int n, per;*

*cout << "\nВведите номер строки" << endl; cin >> n;*

*do*

*{*

*cout << "Введите: " << endl;*

*cout << "1–для изменения фамилии" << endl;*

*cout << "2–для изменения года рождения" << endl;*

*cout << "3–для изменения факультета" << endl;*

*cout << "4–конец\n";*

*cin >> per;*

*switch (per)*

*{*

*case 1: cout << "Новая фамилия";*

*cin >> list\_of\_student[n – 1].name; break;*

*case 2: cout << "Новый год рождения";*

*cin >> list\_of\_student[n – 1].year\_of\_birth; break;*

*case 3: cout << "Новый факультет ";*

*cin >> list\_of\_student[n – 1].department; break;*

*}*

*} while (per != 4);*

*exit;*

*}*

*void out()*

*{*

*int sw, n;*

*cout << "1–вывод 1 строки" << endl;*

*cout << "2–вывод всех строк" << endl;*

*cin >> sw;*

*if (sw == 1)*

*{*

*cout << "Номер выводимой строки " << endl; cin >> n; cout << endl;*

*cout << "Фамилия ";*

*cout << list\_of\_student[n – 1].name << endl;*

*cout << "Год рождения ";*

*cout << list\_of\_student[n – 1].year\_of\_birth << endl;*

*cout << "Факультет ";*

*cout << list\_of\_student[n – 1].department << endl;*

*}*

*if (sw == 2)*

*{*

*for (int i = 0; i < current\_size; i++)*

*{*

*cout << "Фамилия ";*

*cout << list\_of\_student[i].name << '\t';*

*cout << "Год рождения ";*

*cout << list\_of\_student[i].year\_of\_birth << '\t';*

*cout << "Факультет ";*

*cout << list\_of\_student[i].department << endl;*

*}*

*}*

*exit;*

*}*

*int main()*

*{*

*setlocale(LC\_CTYPE, "Russian");*

*cout << "Данных нет" << endl;*

*cout << "Введите:" << endl;*

*do*

*{*

*cout << "Введите:" << endl;*

*cout << "1–для удаления записи" << endl;*

*cout << "2–для ввода новой записи" << endl;*

*cout << "3–для изменения записи" << endl;*

*cout << "4–для вывода записи(ей)" << endl;*

*cout << "5–для выхода" << endl;*

*cin >> choice;*

*switch (choice)*

*{*

*case 1: del(); break;*

*case 2: enter\_new(); break;*

*case 3: change(); break;*

*case 4: out(); break;*

*}*

*} while (choice != 5);*

*}*

**ПРИМЕР**

void search() {

int i, j, found = 0;

student s1;

FILE\* fp;

int rno;

printf("Введит код студента для поиска : ");

scanf("%d", &rno);

fp = fopen("mystudents1.txt", "r");

while (fread(&s1, sizeof(student), 1, fp)) {

if (s1.rno == rno) {

found = 1;

printf("\n%-5d%-20s%-20s%-20s", s1.rno, s1.name, s1.surname, s1.second\_name);

for (j = 0; j < 3; j++) {

printf("%4d", s1.sub[j].mark);

}

printf("%5d%7.2f", s1.total, s1.per);

}

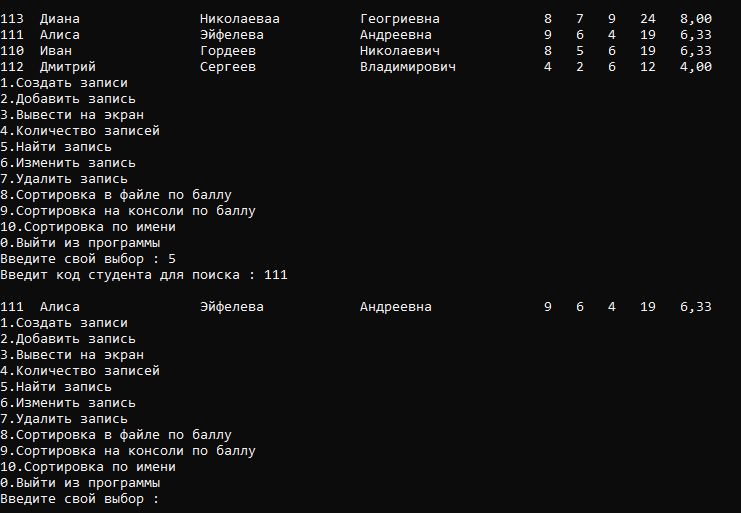
}

if (!found)

printf("\nНе найдено\n");

fclose(fp);

}



**ЗАДАНИЕ**

Создайте структуру Аэропорт с полями Номер рейса, Город прибытия, Город отправления, Время вылета, Время прилета, Авиакомпания (BELAVIA, Аэрофлот и др.), Цена билета, Класс (эконом/бизнес).

Фильтрация по диапазону: вывести рейсы, вылетающие с 16.00 до 18.30.

Фильтрация по значению: вывести рейсы для авиакомпании, которую ввел пользователь.

Реализовать функцию вывода на консоль билета по номеру рейса (ввести с клавиатуры). В билете подсчитать время в пути.