**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**

Федеральное государственное бюджетное образовательное учреждение   
высшего образования

**«Сибирский государственный университет науки и технологий   
имени академика М.Ф. Решетнева»**

Институт информатики и телекоммуникаций

Кафедра информатики и вычислительной техники

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ**

Языки программирования

|  |
| --- |
| Повторение основ работы с классами |

Руководитель А.В. Проскурин

подпись, дата инициалы, фамилия

Обучающийся БПИ22-02, 221219040 К.В. Трифонов

номер группы, зачетной книжки подпись, дата инициалы, фамилия

Красноярск 2023 г.

# ЦЕЛЬ РАБОТЫ

Проверка и получение навыков: использования языковых средств и способов динамического выделения памяти; организации форматного ввода и вывода; Получение практических навыков разработки и отладки программ, использующих объекты класса и файловый ввод/вывод. Изучение различных реализаций конструкторов и областей их применения, а также аспектов перегрузки операций и работы с дружественными функциями.

# порядок выполнения работы

Ознакомиться с общей постановкой задачи.

2. Ознакомится с вариантом задания – соответствует вашему номеру в списке группы (при нехватке заданий вариант задания вычисляется как номер\_в\_списке\_группы - количество\_заданий).

3. Разработать класс согласно варианту задания.

4. Написать и отладить программу на подготовленных наборах тестовых данных.

5. Подготовить отчет по лабораторной работе. Отчет должен включать в себя:

• титульный лист;

• цель лабораторной работы;

• постановку задачи;

• текст программы с комментариями;

• демонстрацию работы программы (Снимки экрана при выполнении действий программы с описанием).

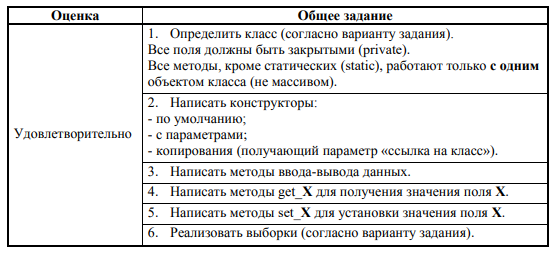
• краткие ответы на контрольные вопросы;

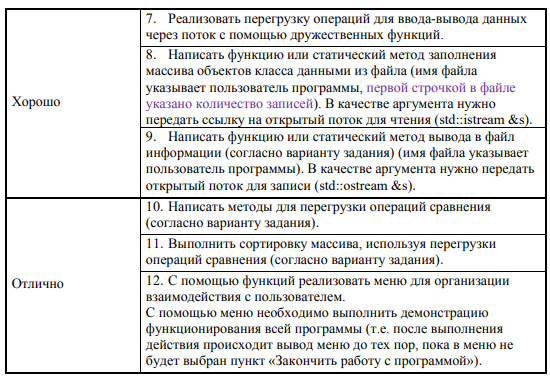
• выводы по лабораторной работе.

6. Защитить лабораторную работу перед преподавателем.

# постановка задачи

Необходимо разработать программу, формирующую и обрабатывающий динамический массив объектов класса (согласно варианту задания). В реализуемой программе необходимо предусмотреть возможность демонстрации работы конструкторов и методов созданного класса. В зависимости от оценки, на которую вы претендуете, необходимо выполнить следующие задания (Для каждой следующей оценки нужно выполнить ВСЕ предыдущие задания, если обратное не указано явно):





**Вариант №21.**

Класс *Aeroflot* (Рейс):

Поля: Пункт назначения, Номер рейса, Тип самолета, Время вылета, Дни недели.

Реализовать выборки: а) списка рейсов для заданного пункта назначения; б) списка рейсов самолетов указанного типа; в) списка рейсов для заданного дня недели, время вылета которых находится в заданном интервале.

Реализовать сортировку массива объектов класса на основе:

- времени вылета.

# ХОД РАБОТЫ

## Определение закрытых полей класса:

*class* aeroflot{

        string destination;

        string index;

        string type;

*int* time;

        string day;

## Конструкторы:

*//конструкторы*

        aeroflot() {}; *//по-молчанию*

        aeroflot(*int* *T*); *//с параметрами*

        aeroflot(*const* aeroflot *&B*); *//копий*

*//деструктор*

        ~aeroflot() {}

## Методы ввода-вывода:

*//методы ввода-вывода*

*void* set();

*void* print()*const*;

*void* set\_time(*const* *int* *T*);

*void* get\_time()*const*;

Метод set()

*void* aeroflot::set(){

    cout << "Укажите назначение рейса:\n";

    cin >> destination;

    cout << "Укажите номер:\n";

    cin >> index;

    cout << "Укажите тип:\n";

    cin >> type;

    cout << "Укажите время:\n";

    cin >> time;

    cout << "Укажите день недели:\n";

    cin >> day;

}

Метод print()

*void* aeroflot::print() *const*{

    cout << "-----------\n";

    cout << "Назначение: " << destination << "\n";

    cout << "Номер: " << index << "\n";

    cout << "Тип: " << type << "\n";

    cout << "Время: " << time / 100 << ":" << time % 100 << "\n";

    cout << "День недели: " << day << "\n";

}

Вывод данных вылета с форматом времени ЧЧ:ММ

## Методы get-set\_X:

*void* set\_time(*const* *int* *T*);

*void* get\_time()*const*;

Метод set\_time()

*void* aeroflot::set\_time(*const* *int* *T*){

    time = *T*;

}

Задаёт поле времени вылета

**Метод get\_time()**

*void* aeroflot::get\_time() *const*{

    cout << "Время рейса: " << time << "\n";

}

Возвращает время вылета у рейса

*//Функции ввода-вывода для массивов*

*//Заполнение массива*

*void* setMA(aeroflot *\*A*, *int* *n*){

    for (*int* i = 0; i < *n*; i++)

    {

        cout << "Заполните рейс № " << i + 1 << ":\n";

*A*[i].set();

    }

}

*//Вывод массива*

*void* getMA(aeroflot *\*A*, *int* *n*){

    for (*int* i = 0; i < *n*; i++)

    {

        cout << "Рейс № " << i + 1 << ":\n";

*A*[i].print();

    }

}

## Выборки:

*//Выборки*

*friend* *void* DEST(aeroflot *\*B*, *int* *n*, string *D*);

*friend* *void* TYPE(aeroflot *\*B*, *int* *n*, string *T*);

*friend* *void* TIME(aeroflot *\*B*, *int* *n*, string *D*, *int* *t1*, *int* *t2*);

*friend* *void* SORT(aeroflot *\*B*, *int* *n*);

*void* DEST(aeroflot *\*B*, *int* *n*, string *D*){

    cout << "Для заданного пункта назначения рейсы с номерами:\n";

*int* c = 0;

    for (*int* i = 0; i < *n*; i++)

    {

        if ((*B*[i].destination == *D*))

        {

            cout << *B*[i].index << "\n";

            c++;

        }

    }

    cout << "Всего: " << c << "\n";

}

Выборка по полю назначения

*void* TYPE(aeroflot *\*B*, *int* *n*, string *T*){

    cout << "Для заданного типа рейсы с номерами:\n";

*int* c = 0;

    for (*int* i = 0; i < *n*; i++)

    {

        if (*B*[i].type == *T*)

        {

            cout << *B*[i].index << "\n";

            c++;

        }

    }

    cout << "Всего: " << c << "\n";

}

Выборка по полю тип

*void* TIME(aeroflot *\*B*, *int* *n*, string *D*, *int* *t1*, *int* *t2*){

    cout << "Для заданного дня недели и диапазона времени назначения рейсы с номерами:\n";

*int* c = 0;

    for (*int* i = 0; i < *n*; i++)

    {

        if ((*B*[i].day == *D*) && (*t1* <= *B*[i].time) && (*B*[i].time <= *t2*))

        {

            cout << *B*[i].index << "\n";

            c++;

        }

    }

    cout << "Всего: " << c << "\n";

}

Выборка по полю дня вылета и диапазону времени с выводом на экран и подсчетом общего кол-ва

## Перегрузка операций ввода-вывода через поток:

*//потоковый ввод-вывод*

*friend* ostream*&* operator<<(ostream*&* *s\_out*, *const* aeroflot*&* *D*);

*friend* istream*&* operator>>(istream*&* *s\_in*, aeroflot*&* *D*);

ostream *&*operator<<(ostream *&s\_out*, *const* aeroflot *&D*){

*s\_out* << *D*.destination << " " << *D*.index << " " << *D*.type << " " << *D*.time << " " << *D*.day << "\n";

    return *s\_out*;

}

Вывод полей в поток

istream *&*operator>>(istream *&s\_in*, aeroflot *&D*){

*// cout<<"Укажите назначение, номер, тип, время вылета и день недели через пробел: \n";*

*s\_in* >> *D*.destination >> *D*.index >> *D*.type >> *D*.time >> *D*.day;

    return *s\_in*;

}

Заполнение полей из потока (подсказка ввода пользователю закомментирована, чтобы не спамить в консоль при чтении потока файла)

## Заполнение массива объектов из потока файла:

*void* importA(istream *&fin*, aeroflot *\*B*, *const* *int* *n*){

*int* c;

*fin* >> c;

    if (c != *n*)

    {

        cout << "Массив задан неверно!\n";

    }

    else

    {

        for (*int* i = 0; i < c; i++)

        {

*fin* >> *B*[i];

        }

    }

}

На вход функции подается поток, созданный вне функции по файлу пользлователя, указатель на массив и длинна массива. Сравнивается кол-во данных в файле (по числу на первой строчке) с длинной массива.

## Запись в поток файл массива объектов:

*void* exportA(ostream *&fout*, *const* aeroflot *\*B*, *const* *int* *n*){

*fout* << *n* << "\n";

    for (*int* i = 0; i < *n*; i++)

    {

*fout* << *B*[i];

    }

}

На вход функции подается поток, созданный вне функции по файлу пользлователя, указатель на массив и длинна массива.

*//Оператор присваивания*

*const* *void* aeroflot::operator=(*const* aeroflot *B*){

    time = *B*.time;

    destination = *B*.destination;

    index = *B*.index;

    type = *B*.type;

    day = *B*.day;

}

## Перегрузка операций сравнения:

*const* *bool* operator == (*const* aeroflot *&B*);

*const* *bool* operator > (*const* aeroflot *&B*)*const*;

*const* *bool* operator < (*const* aeroflot *&B*)*const*;

*//Оператор равенства*

*const* *bool* aeroflot::operator==(*const* aeroflot *&B*){

*bool* answ = ((*B*.destination == destination) and (*B*.index == index) and (*B*.type == type) and (*B*.time == time) and (*B*.day == day));

    return answ;

}

*const* *bool* aeroflot::operator>(*const* aeroflot *&B*)*const*{

*bool* answ = *B*.time > time;

    return answ;

}

*const* *bool* aeroflot::operator<(*const* aeroflot *&B*)*const*{

*bool* answ = *B*.time < time;

    return answ;

}

## Сортировка массива по времени вылета через перегрузку операторов сравнения:

*void* SORT(aeroflot *\*B*, *int* *n*){

    for (*int* i = 0; i < *n*; i++)

    {

*bool* flag = true;

        for (*int* j = 0; j < *n* - 1; j++)

        {

            if (*B*[j] < *B*[j + 1])

            {

                flag = false;

                aeroflot b; *// создание дополнительной переменной*

                b = *B*[j];

*B*[j] = *B*[j + 1]; *// меняем местами значения*

*B*[j + 1] = b;

            }

        }

        if (flag)

        {

            break;

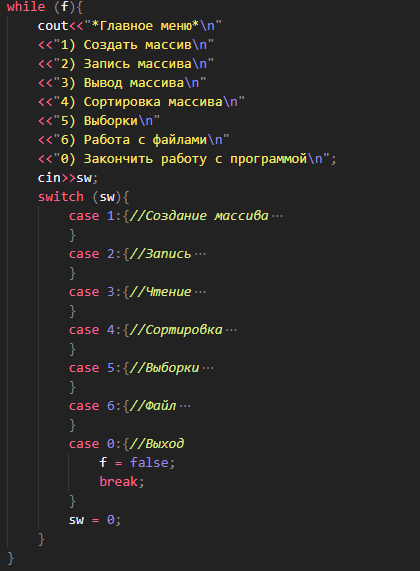
        }

    }

}

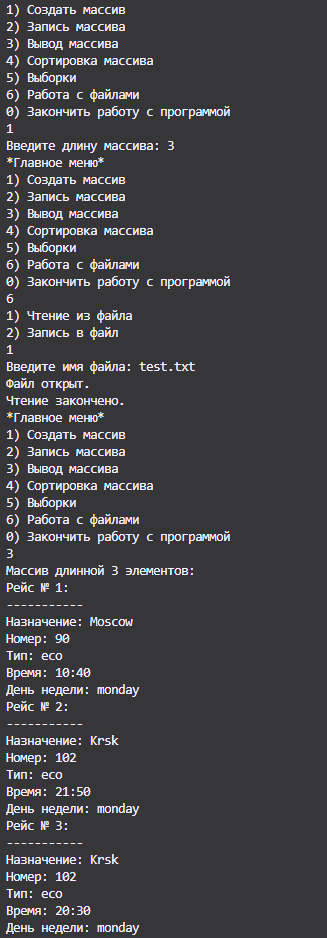
Используется метод пузырьковой сортировки с привязкой к полю времени

## Меню:

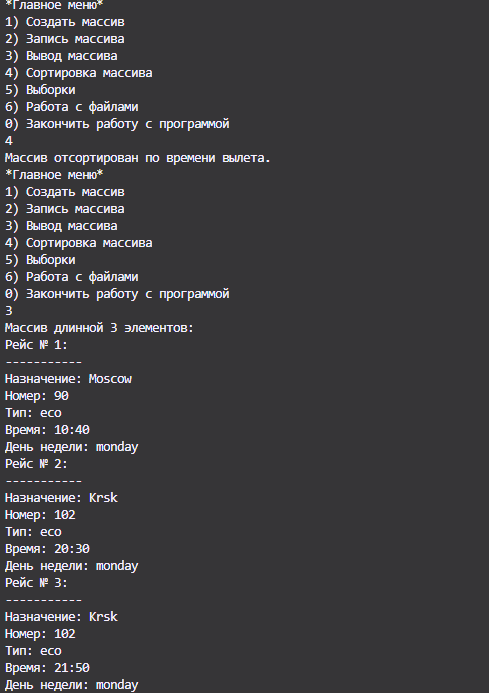


## Примеры работы

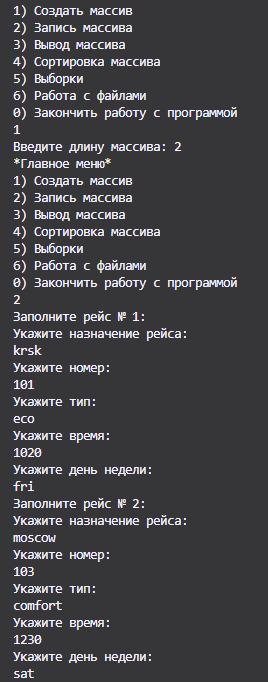
Пример создания массива длиной 3 объекта и последующее его заполнение из файла test.txt с выводом на экран



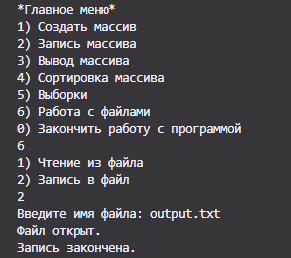
Пример сортировки массива по времени вылета (из файла test.txt)



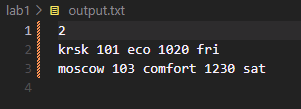
Пример создания и заполнения массива из 2 объектов



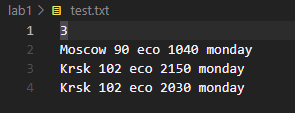
Вывод массива в файл output.txt



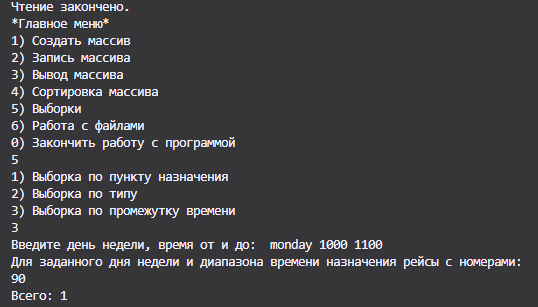
Cодержимое выходного файла:



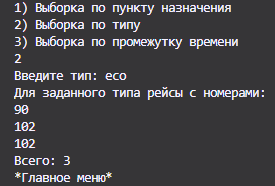
Для тестирования выборок данные брались из предварительно заполненного файла test.txt:



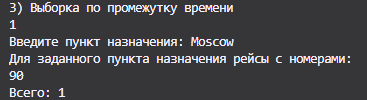
Пример работы выборки по дню недели и промежутку времени:



Пример работы выборки по типу:



Пример работы выборки по пункту назначения:



# ОТВЕТЫ НА КОНТРОЛЬНЫЕ ВОПРОСЫ

1. **Раскройте понятие указателя в языке С. Опишите синтаксис объявления указателя.**

*Указатель* - переменная, которая хранит адрес сущностей (т.е. других переменных любого типа, будь то структура, или массив), объявляются как обычные переменные с префиксом из минимум одной (\*).

1. **В чем преимущество использования операций new и delete в языке С++ по сравнению с аналогичными средствами языка С?**

Не нужно вручную рассчитывать размер объекта

1. **В чем заключаются преимущества использования динамических массивов?**

Размер массива задаётся самим пользователем

1. **Как производится обращение к элементам динамических массивов?**

По индексу, как и в статических

1. **Как производится определение пользовательских функций в языке С?**

*void* set\_time(*const* *int* *T*);

Пишется тип значения, которое возвращает функция (int, void…), название функции и в скобках данные на вход функции (или пустые скобы).

1. **Как происходит передача параметров в функцию по значению/ адресу?**

При передаче параметров в функцию по значению она работает с копиями фактических переменных, не имея возможности изменять исходные переменные, для вывода данных функция должна их вернуть через return. При передаче по адресу функция обращается к данным непосредственно в ячейку памяти по адресу переменной и может его изменять.

1. **Какого типа значения может возвращать функция? Как вернуть из функции несколько значений?**

Функция может вернуть любые виды значения, за исключением функции (если функция не должна возвращать значение, то её тип void), вернуть несколько значений можно через указатели, адреса, структуры.

1. **Как организуется доступ к файлам? Какие виды доступа вы знаете? Каким образом можно считать/записать данные из текстового файла?**

Доступ организуется через открытие потока ifstream – для чтения, ofstream – для записи, указывается имя файла, относительно main и данные записываются в файл последовательно через оператор потока <<, последовательно записываются в переменную через оператор >>, далее поток необходимо закрыть методом close().

1. **Что такое класс? Как выполнить объявление класса?**

Класс — это пользовательский тип данных, который имеет заданное количество полей и методов. Класс объявляется через ключевое слово class.

1. **Как в классе реализуется инкапсуляция?**

Доступ к полям может быть доступ либо извне (public поля), либо только

с помощью внутренних методов и friend функций (private поля), тем самым создаётся “черный ящик” из объекта и пользователь не имеет представления о реализации класса.

1. **Каким образом элементы класса получают атрибуты доступа?**

Элементы класса получают атрибуты доступа, либо по умолчанию (в зависимости от ключа класса) либо при использовании спецификаторов доступа:

public (общедоступный),

private (собственный),

protected (защищенный).

1. **Что такое объект класса и как выполнить объявление объекта класса в программе?**

Объект — это отдельная переменная, представитель класса, имеющий конкретное состояние и поведение, полностью определяемое классом. Объект объявляется, как и все стандартные переменные – указывается ключевое слово тип-название класса и название объекта.

1. **Что такое методы класса? Каковы особенности объявления и определения методов класса?**

Методы делают над данными некоторые операции (некоторую работу). Во многих случаях методы представляют доступ к данным класса.

1. **Что такое указатель this? Как выполняется вызов метода класса?**

\*this позволяет обращаться к полям объекта через новую переменную. Для применения метода к объекту нужно обращаться к нему через переменную.

1. **Какие функции называются конструкторами? Каков механизм передачи параметров конструктору?**

Функции, которые позволяют создать объект (или сразу заполнить его поля), параметры передаются на вход функции.

1. **Что такое конструктор по умолчанию и какую пользу приносит его наличие?**

Конструктор по умолчанию нужен чтобы создавать объект не получая на

вход никаких параметров, нужен для удобства создания переменных пользователем.

1. **Как выполняется инициализация объектов класса?**

Инициализация объекта класса осуществляет конструктор при его создании.

1. **Каким образом в программе можно создать объект класса, элементы-данные которого являются копиями элементов данных созданного ранее объекта?**

Для этого применяют конструктор копий, в котором данные из полей

объекта на входе функции копируются в новый экземпляр.

1. **Каким образом осуществляется перегрузка операций в языке С++?**

Указывается тип данных, которые должен вернуть оператор, ключевое слово operator, его символ и переменные на вход (справа от оператора). Изменяемый объект обычно должен находиться слева.

1. **Какие операции не могут быть перегружены?**

*Список операторов, которые нельзя перегружать:*

1) Оператор разрешения области (::)

2) Троичный или условный оператор (?:)

3) Доступ к элементу или оператор Dot (.)

4) Оператор указателя на элемент (.\*)

5) Оператор размера объекта (sizeof)

6) Оператор объектного типа (typeid)

7) static\_cast (оператор приведения)

8) const\_cast (оператор приведения)

9) reinterpret\_cast (оператор приведения)

10) dynamic\_cast (оператор приведения)

1. **Теряет ли операция при перегрузке свое исходное назначение?**

Не теряет, перегрузка вводит новую операцию для класса, не затрагивая её поведение для остальных переменных.

1. **Каков синтаксис объявления оператора-функции?**

*const* *void* operator = (*const* aeroflot *B*);

Пишется тип возвращаемого значения, ключевое слово operator, символ и значение не вход функции.

1. **Почему при перегрузке унарных операций оператор-функция не имеет параметров?**

Если унарная операция перегружается как функция-член, то она не должна иметь аргументов, так как в этом случае ей передается неявный аргумент-указатель this на текущий объект.

1. **Как различить префиксную и постфиксную операции инкремента при перегрузке?**

Перегруженные префиксные операторы возвращают объект после того, как он был инкрементирован или декрементирован. Префиксный — это противоположность постфиксному. Сначала оператор инкремента или декремента увеличивает или уменьшает операнд (переменную) и только после этого над ним производятся остальные операции, если они есть. *То есть, если у нас такая строчка: A = 5 \* —B; , то это будет читаться так: уменьшить значение переменной B на единицу, после чего умножить 5 на B и присвоить результат переменной A.*

1. **Какая функция называется другом класса?**

Которая не является компонентом класса, но описана в нём и имеет доступ к его защищенным полям.

1. **В чем состоят различия между дружественной функцией и методом класса?**

Метод должен обязательно принадлежать конкретному классу и при вызове относиться к конкретному объекту.

1. **В чем состоят особенности передачи параметров дружественной функции при перегрузке унарных и бинарных операций?**

При перегрузке унарных операций параметры берутся из одной переменной слева (пре-унарные) или справа (пост-унарные) от оператора, а при перегрузке бинарных операций из двух аргументов слева и справа от оператора.

1. **Почему нельзя перегрузить операции >> и << для ввода и вывода типов с помощью метода класса?**

Потому что левостоящая переменная в таком операторе — это поток, а для него нельзя изменить операции, т.к. его класс уже полностью определён, поэтому применяют перегрузку вне класса, т.к. тогда поток передаётся в параметрах.

# ВЫВОДЫ

Были закреплены основы работы с классами, а также использованы новые алгоритмы такие как, сортировка массивов класса, заполнение массива класса из файла, а также вывод в файл.