Лабораторная № 1

Сдать оформленные работы до 18 февраля.

Для реализации проекта методы описать в header (заголовочном) файле (*.h). Реализацию методов поместить в файл *.cpp, функцию main - в файл main.cpp. Использовать исключения.

Реализовать стек (очередь)в классе на базе односвязного и дек на базе двухсвязного динамическиго списка. Использовать исключения.

Общее задание:

Для обработки стека(дека, очереди) реализовать методы в классе:

1. Создание списка (чтение данных из файла)
2. Добавить элемент
3. Удаление элемента
4. Удаление всех элементов
5. Вывод списка на консоль и в файл
6. Запросы на выполнение операций оформить в виде меню.

Примечание: Выделять память для одного элемента списка через конструктор.

Индивидуальные задания:

1.1. Очередь. Элемент списка структура. Структура содержит информацию о студентах, поля: фамилия, код, телефон (char*,float, int).
1.2.. Стек. Элемент списка структура. Структура содержит информацию об автомобилях, поля: модель, год выпуска, номер (char*,int,double).
1.3. Дек. Элемент списка структура. Структура содержит информацию о дилерах, поля: название, адрес (char*,S char*, double)
1.4. Очередь. Элемент списка структура. Структура содержит информацию о владельцах номеров телефонов, поля: номер телефона, фамилия. (short, char*, double)
1.5. Стек. Элемент списка структура. Структура содержит информацию о товарах, поля: название, цена. (char*, double, char*)
1.6. Дек. Элемент списка структура.. Структура содержит информацию о сотрудниках фирмы, содержит поля: фамилия, должность. (char*, char*, int)
1.7. Очередь. Элемент списка структура. Структура содержит данные об услугах Интернет, поля: фамилия, время разговора. (char*, int, double)
1.8. Стек. Элемент списка структура. Структура содержит информацию о рейсах авиакомпании, поля: номер рейса, название. (double, char*, int)
1.9. Дек. Элемент списка структура. Структура содержит информацию о компьютерах, поля: модель, объём HDD (int, long, char*)
1.10. Очередь. Элемент списка структура. Структура содержит информацию о туристических поездках, поля: стоимость в уе, название, (float, char*, int)
1.11. Стек. Элемент списка структура. Структура содержит информацию о книгах, поля: название, автор. (char*,S char*, double)
1.12. Дек. Элемент списка структура. Структура содержит информацию о CD, поля: название, стоимость (char*, long, double)
1.13. Очередь. Элемент списка структура. Структура содержит информацию о рейсах автобусах, поля: номер рейса, продолжительность (int, float, char*)
1.14. Стек. Элемент списка структура. Структура содержит информацию о книгах, поля: идентификатор, количество страниц. (long, short, char*)
1.15. Дек. Элемент списка структура. Структура содержит информацию о компьютерах, поля: модель, скорость (long, int, double)

Лабораторная № 2

Сдать оформленные работы до 21 февраля.

Для реализации проекта методы описать в header (заголовочном) файле (*.h). Реализацию методов поместить в файл *.cpp, функцию main - в файл main.cpp. Использовать исключения.

Реализовать метод классе со списком на базе односвязного или двухсвязного динамическиго списка.

Общее задание:

Для обработки списка:

1. Реализовать конструктор –создание списка из N элементов (чтение из файла)
2. Индивидуальный метод
3. Вывод списка на консоль
4. Реализовать деструктор (удаление списка)
5. Запросы на выполнение операций оформить в виде меню.

Индивидуальные задания:

1.Продублировать первый элемент в списке. Элемент списка структура. Структура содержит информацию о студентах, поля: фамилия, код. (char*,float)
2. Продублировать последний элемент в списке. Элемент списка структура. Структура содержит информацию об автомобилях, поля: модель, год выпуска. (char*,int)
3. Продублировать элемент в списке средний (один из двух, если четное количество элементов). Элемент списка структура. Структура содержит информацию о дилерах, поля: название, адрес (char*,S char*)
4. Удалить наибольший положительный элемент в списке. Элемент списка структура. Структура содержит информацию о владельцах номеров телефонов, поля: номер телефона, фамилия. (short, char*)
5. Удалить наименьший положительный элемент в списке. Элемент списка структура. Структура содержит информацию о товарах, поля: название, цена. (char*, double)
6. Удалить элементы с длиной строки K из списка. Элемент списка структура.. Структура содержит информацию о сотрудниках фирмы, содержит поля: фамилия, должность. (char*,char*)
7. Удалить наименьший отрицательный элемент в списке. Элемент списка структура. Структура содержит данные об услугах Интернет, поля: фамилия, время разговора. (char*, int)
8. Удалить средний элемент в списке (один из двух, если четное количество элементов). Элемент списка структура. Структура содержит информацию о рейсах авиакомпании, поля: номер рейса, название. (double, char*)
9. Вывести положительные элементы списка. Элемент списка структура. Структура содержит информацию о компьютерах, поля: модель, объём HDD (int, long)
10. Вывести отрицательные элементы списка. Элемент списка структура. Структура содержит информацию о туристических поездках, поля: стоимость в уе, название, (float, char*)
11. Вывести четные элементы списка. Элемент списка структура. Структура содержит информацию о CD, поля: название, стоимость (char*, long)
12. Вывести нечетные элементы списка. Элемент списка структура. Структура содержит информацию о рейсах автобусах, поля: номер рейса, стоимость(int, float)
13. Вывести элементы с нулями из списка. Элемент списка структура. Структура содержит информацию о книгах, поля: идентификатор, количество страниц. (long, short)
14. Вывести элементы сединицами из списка. Элемент списка структура. Структура содержит информацию о компьютерах, поля: модель, скорость (long, int)
15. Удалить наибольший отрицательный элемент в списке. Элемент списка структура. Структура содержит информацию о компьютерах, поля: модель, цена (long, __int8)

Лабораторная №3

Сдать оформленные работы до 6.03.

Для реализации проекта методы описать в header (заголовочном) файле (*.h). Реализацию методов поместить в файл function.cpp, функцию main - в файл main.cpp

Общее задание:

Классы, перегрузка операторов.

Реализовать обработку ошибок с использованием механизма исключений.

Реализовать классс с операторами для индивидуального задания:

а) *имеющий конструктор, конструктор копирования, деструктор.*

b) *Выполнить перегрузку операторов, написать функции для обработки +, -, *(умножение на число), /(деление на число), ++(постфиксный и префиксный вариант),-- (постфиксный и префиксный вариант),=,==, !=, >=,<=, [] - как членов класса.*

c) *Выполнить перегрузку операторов, написать функции для обработки двух **объектов** +, -, *, как **дружественные функции.**.*

d) *операции ввода–вывода: перегрузку операторов<<,>>, как дружественные функции.*

e) *Использовать исключения.*

Вывод результатов:

Вывести комментарии (операции) на консоль: содержимое объекта(ов) **до и после** операций и операцию (или выражение).

Меню не использовать!!!

Например, часть результатов(на консоль) для класса Point(точка):

….
A=2,3 B=3,5 C=A+B
Rezult: A=2,3 B=3,5 C=5,8

A=2,3 C=++A
Rezult: A=3,4 C=3,4

A=2,3 C=A++
Rezult: A=3,4 C=2,3

…

Индивидуальные задания:

3.1. Класс для хранения матриц переменной размерности (Тип: элемента double). Операция ++(сложение с единичной матрицей). +N(-N, *N, /N) – добавление числа N к каждому элементу матрицы(аналогично с вычитанием, умножением, делением). Операция “[]” – возвращение строки (или столбца) как матрицы размерности 1*N.

3.2. Класс 4D-точка. Координаты хранить в массиве. Операция : ~ -отражение относительно начала координат.

3.3. Класс для хранения N- мерной точки (использовать массив, Тип: элемента int). Операция : ~ - поразрядная инверсия элементов (изменить знак).

3.4. Класс для хранения строк переменной размерности.
Перегрузить только операции: +N,-N(удаление подстроки размерности N), ++(увеличение подстроки на символ, справа или слева), --(уменьшение строки на один символ справа или слева), *N(увеличение длины строки в N раз), *(для двух объектов - объединение для строк), /N(взятие подстроки размерности : длина строки\N),), +(преобразование к верхнему регистру), - (преобразование к нижнему регистру), /(пересечение строк), ~ - переворот строки.

3.5. Класс стек (использовать массив, Тип: элемента char). Операция ++ - добавить элемент, операция -- (удалить элемент) +(объединение двух стеков). Операция +N(-N, *N, /N) – добавление числа N к видимому элементу стека (аналогично с вычитанием, умножением, делением).

3.6. Класс для хранения комплексных чисел.

3.7. Класс для хранения векторов в N- мерном пространстве (использовать массивы, Тип: элемента double). Операция : ~ - направление вектора изменить на противоположное.

3.8. Класс ломаная на плоскости (хранить в массиве). Операция + : соединение ломаных ребром. Операция – удаление совпадающих ребер(если на концах ломанной). Операция ~ -отражение относительно начала координат. Операция ++ увеличить на одно ребро, длина ребра=1 (по аналогии опер. «--») Операции *N, /N - увеличение\уменьшение количества ребер в N раз . Операции *,/ над двумя объектами не делать. Операции сравнения – по общей длине ребер.

3.9. Класс Окружность. Операции +,- :объединение ,пересечение. Операция ~ -отражение относительно начала координат. Операция ++ прибавить к координатам и радиусам) единицу (по аналогии опер. «--») Операции *N, /N - увеличение\уменьшение радиусов(в N раз) . Операции *,/ над двумя объектами не делать. Операции сравнения - по площади.

3.10. Класс треугольная область (трегольник) на плоскости. Операции +,- :объединение ,пересечение .Операция ~ -отражение относительно начала координат. Операция ++ прибавить к координатам единицу (по аналогии опер. «--») Операции *N, /N - увеличение\уменьшение координат в N раз (или площади). Операции *,/ над двумя объектами не делать. Операции сравнения - по площади.

3.11. Класс отрезок на плоскости . Операция ~ -отражение относительно начала координат. Операция ++ прибавить к координатам единицу(по аналогии опер. «--») Операция *N увеличить длину в N раз. Операция /N - уменьшить длину в N раз. Операции сравнения - длине.

3.12. Класс Прямоугольная область на плоскости. Операции +,- :объединение ,пересечение. Операция ~ - отражение относительно начала координат. Операция ++ прибавить к координатам единицу или прямоугольник, длина одной стороны которого =1, (по аналогии опер. «--») Операция *N увеличить прямоугольную область в N раз. Операция /N - уменьшить прямоуг. область в N раз. Операции сравнения - по площади.

3.13. Класс текст (матрица, Тип: элементов - char). Операция +(добавить N строк до или после текста), операцию - (удалить N строк),^ (шифрование текста). Операция "+" – объединение для строк. Операция "–" разность (из первого текста вычесть строки, содержащиеся во втором). Операция "++" – добавить строку, операция "--" – удалить строку. Операция *N (продублировать текст N раз). Операция /N (сократить текст (количество строк) в N раз). Вместо операции, *(для двух объектов) использовать - & (пересечение для строк).

3.14. Класс очередь (использовать массив, Тип: элемента double). Операция ++ - добавляет случайный элемент, операция -- (удаляет элемент). Операция +(объединение двух очередей). Операция +N(-N) – добавляет(удаляет) N случайных элементов. Операция *N, (/N) умножение на число N (и соответственно деление на число N) крайнего элемента очереди.

3.15. Класс дек (использовать массив, Тип: элемента double). Операция ++ - добавить элемент(с начала или с конца), операция -- удалить элемент(с начала или конца) +(объединение двух деков). Операции: +N, -N (добавление\удаление N элементов с обоих концов); *N, /N – умножение\деление на число N концевых элементов.

3.16. Класс массив (Тип: элемента long).

Лабораторная № 4.

Сдать оформленные работы до 20.03.

Абстрактные классы. Наследование.

Общее задание:

Примерное определение классов:

```
class AbstractOBJ{
protected:

        const int Id;
        const string Name;
public:
        static int Count;
        AbstractOBJ (string aName){}
          AbstractOBJ (){}
        virtual  string GetName()=0 ;
        virtual int GetId()=0;
        …
        virtual void Show(ostream&)=0;
        ~ AbstractOBJ (){}
protected:
        //некоторые методы

}
class OBJ:public AbstractOBJ {
protected:

        …
public:
        OBJ (string aName){}
        void Show(ostream&);
…
        ~ OBJ(){}
};

class OBJ2: public OBJ
{
public:
```

```
        OBJ2 ():OBJ(){}…
        OBJ2 (string aName):OBJ(aName ){}…
        …
        ~ OBJ2 (){}
};
```

Для вывода результата создать минимум 3-и объекта :1-го- класса OBJи 2-го- класса OBJ2

Индивидуальные задачи.

3.1.Создать абстрактный класс. Создать класс студент (наследник абстрактного класса), имеющий имя (указатель на строку), курс и идентификационный номер. Определить конструкторы, деструктор и функцию вывода. Создать public-производный класс - студент-дипломник, имеющий тему диплома. Определить конструкторы по умолчанию и с разным числом параметров, деструкторы, функцию вывода. Определить функции переназначения названия диплома. Использовать статическую переменную для хранения количества созданных объектов классов наследников и константы для хранения идентификационных номеров в абстрактном классе, в классах наследниках.

3.2. Создать абстрактный класс. Создать класс комната, имеющая размеры (не наследник абстрактного класса). Определить конструктор и метод доступа. Создать класс квартира(наследник абстрактного класса), содержащий комнаты(класс комната содержится в классе однокомнатная квартира) и кухню (ее площадь), этаж. Определить конструкторы, методы доступа. Определить public-производный класс котедж (дополнительный параметр - название количество этажей). Определить конструкторы, деструктор и функцию вывода. Использовать статическую переменную для хранения количества созданных объектов классов наследников и константы для хранения идентификационных номеров в абстрактном классе, в классах наследниках.

3.3. Создать абстрактный класс. Создать класс машина(наследник абстрактного класса), имеющий марку (указатель на строку), число цилиндров, мощность. Определить конструкторы, деструктор и функцию вывода. Создать public-производный класс - грузовик, имеющий грузоподъемность кузова. Определить конструкторы по умолчанию и с разным числом параметров, деструкторы, функцию вывода. Определить функции переназначения марки и грузоподъемности. Использовать статическую переменную для хранения количества созданных объектов классов наследников и константы для хранения идентификационных номеров в абстрактном классе, в классах наследниках.

3.4. Создать абстрактный класс. Создать класс двигатель, имеющий мощность (не наследник абстрактного класса). Определить конструкторы и метод доступа. Создать класс машин(наследник абстрактного класса), содержащий класс двигатель. Дополнительно есть марка (указатель на строку), цена. Определить конструкторы и деструктор. Определить public- производный класс грузовик, имеющий дополнительно грузоподъемность. Определить конструкторы, деструкторы и функцию вывода. Использовать статическую переменную для хранения количества созданных объектов классов наследников и константы для хранения идентификационных номеров в абстрактном классе, в классах наследниках.

3.5. Создать абстрактный класс. Создать класс жидкость(наследник абстрактного класса), имеющий название (указатель на строку), плотность. Определить конструкторы, деструктор и функцию вывода. Создать public-производный класс - сок, имеющий процент натуральности. Определить конструкторы по умолчанию и с разным числом параметров, деструкторы, функцию вывода. Определить функции переназначения плотности и крепости. Использовать статическую переменную для хранения количества созданных объектов классов наследников и константы для хранения идентификационных номеров в абстрактном классе, в классах наследниках.

3.6. Создать абстрактный класс фигура. Создать класс точка имеющий название (указатель на строку), координаты, цвет. Определить конструкторы, деструктор и функцию вывода. Создать public-производный класс (от абстрактного класса) - многоугольник, имеющий площадь и содержит массив точек (использовать класс точка). Определить конструкторы по умолчанию и с разным числом параметров, деструкторы, функцию вывода. Определить функции переназначения координат и площади. Использовать статическую переменную для хранения количества созданных объектов классов наследников и константы для хранения идентификационных номеров в абстрактном классе, в классах наследниках.

3.7. Создать абстрактный класс. Создать класс процессор, имеющий мощность (МГц). Определить конструкторы и метод доступа. Создать класс компьютер(наследник абстрактного класса), содержащий класс процессор. Дополнительно есть марка (указатель на строку), цена. Определить конструкторы и деструктор. Определить public- производный класс компьютеров с монитором, имеющий дополнительно размер монитора. Определит конструкторы, деструкторы и функцию вывода. Использовать статическую переменную для хранения количества созданных объектов классов наследников и константы для хранения идентификационных номеров в абстрактном классе, в классах наследниках.

3.8. Создать абстрактный класс. Создать класс человек(наследник абстрактного класса), имеющий имя (указатель на строку), возраст, вес. Определить конструкторы, деструктор и функцию вывода. Создать

public-производный класс - совершеннолетний, имеющий номер паспорта. Определить конструкторы по умолчанию и с разным числом параметров, деструкторы, функцию вывода. Определить функции переназначения возраста и номера паспорта. Использовать статическую переменную для хранения количества созданных объектов классов наследников и константы для хранения идентификационных номеров в абстрактном классе, в классах наследниках.

3.9. Создать абстрактный класс. Создать класс колесо, имеющий радиус (не наследник абстрактного класса). Определить конструкторы и метод доступа. Создать класс машин(наследник абстрактного класса), содержащий класс колесо. Дополнительно есть марка (указатель на строку), цена. Определить конструкторы и деструктор. Определить public- производный класс грузовик, имеющий дополнительно грузоподъемность. Определить конструкторы, деструкторы и функцию вывода. Использовать статическую переменную для хранения количества созданных объектов классов наследников и константы для хранения идентификационных номеров в абстрактном классе, в классах наследниках

3.10. Создать абстрактный класс. Создать класс жесткий диск, имеющий объем (Мбайт) радиус (не наследник абстрактного класса). Определить конструкторы и метод доступа. Создать класс компьютер(наследник абстрактного класса), содержащий класс жесткий диск. Дополнительно есть марка (указатель на строку), цена. Определить конструкторы и деструктор. Определить public- производный класс компьютеров с монитором, имеющий дополнительные периферийные устройства. Определит конструкторы, деструкторы и функцию вывода. Использовать статическую переменную для хранения количества созданных объектов классов наследников и константы для хранения идентификационных номеров в абстрактном классе, в классах наследниках.

3.11. Создать абстрактный класс фигура. Создать класс окружность (наследник абстрактного класса), имеющий название (указатель на строку), координаты, площадь. Определить конструкторы, деструктор и функцию вывода. Создать public-производный класс - прямоугольник, имеющий дополнительные координаты. Определить конструкторы по умолчанию и с разным числом параметров, деструкторы, функцию вывода. Определить функции переназначения координат и площади. Использовать статическую переменную для хранения количества созданных объектов классов наследников и константы для хранения идентификационных номеров в абстрактном классе, в классах наследниках.

3.12. Создать абстрактный класс. Создать класс студент(наследник абстрактного класса), имеющий имя (указатель на строку), возраст, курс , группа. Определить конструкторы, деструктор и функцию вывода. Создать public-производный класс - школьник, имеющий класс (год обучения). Определить конструкторы по умолчанию и с разным числом параметров, деструкторы, функцию вывода. Определить функции переназначения возраста и класса. Использовать статическую переменную для хранения количества созданных объектов классов наследников и константы для хранения идентификационных номеров в абстрактном классе, в классах наследниках

3.13. Создать абстрактный класс. Создать класс машина(наследник абстрактного класса), имеющий марку (указатель на строку), число цилиндров, мощность. Определить конструкторы, деструктор и функцию вывода. Создать public-производный класс – легковая машина, имеющий вместимость пассажиров, объем багажника, наличие компьютера, спортивная или нет. Определить конструкторы по умолчанию и с разным числом параметров, деструкторы, функцию вывода. Определить функции переназначения марки и грузоподъемности. Использовать статическую переменную для хранения количества созданных объектов классов наследников и константы для хранения идентификационных номеров в абстрактном классе, в классах наследниках.

3.14. Создать абстрактный класс. Создать класс процессор, имеющий мощность (МГц) (не наследник абстрактного класса). Определить конструкторы и метод доступа. Создать класс компьютер(наследник абстрактного класса), содержащий класс процессор, дополнительно есть марка (указатель на строку), объем памяти, винчестер, цена, монитор (размер). Определить конструкторы и деструктор. Определить private-, public- производный класс компьютеров с внешними устройствами (массив структур), имеющий дополнительно размер монитора. Определить конструкторы, деструкторы и функцию вывода. Использовать статическую переменную для хранения количества созданных объектов классов наследников и константы для хранения идентификационных номеров в абстрактном классе, в классах наследниках.

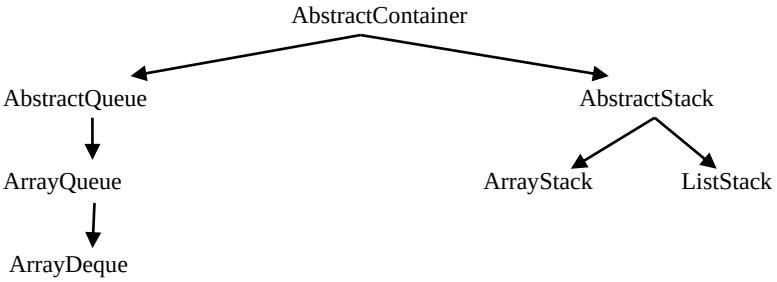
3.15. Создать абстрактный класс. Создать класс плавающее средство(наследник абстрактного класса), имеющий марку (указатель на строку), число весел, длина, количество парусов. Определить конструкторы, деструктор и функцию вывода. Создать public-производный класс – моторная лодка, имеющий марку и мощность двигателя. Определить конструкторы по умолчанию и с разным числом параметров, деструкторы, функцию вывода. Определить функции переназначения марки и грузоподъемности. Использовать статическую переменную для хранения количества созданных объектов классов наследников и константы для хранения идентификационных номеров в абстрактном классе, в классах наследниках.

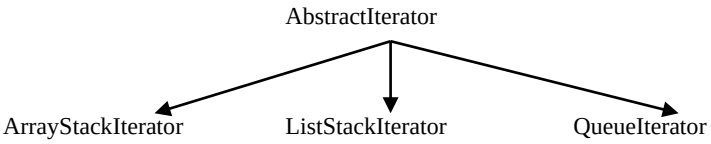
Лабораторная № 5

Сдать оформленные работы до 25.04.

Абстрактные классы, наследование.

Иерархия классов:





Абстрактные классы

```
// абстрактный базовый класс Итератор
class AbstractIterator
{
public:
        virtual ~AbstractIterator() {};
        virtual bool InRange() const = 0;          // индекс в допустимых пределах?
        virtual void Reset() = 0;                  // сбросить индекс в начало
        virtual int& operator *() const = 0;       // разыменование (чтение элемента)
        virtual void operator ++() = 0;                    // сдвиг на элемент
};
// абстрактный базовый класс Контейнер
class AbstractContainer
{
public:
        virtual ~AbstractContainer() {};
        virtual bool IsEmpty() const = 0;          // контейнер пуст
        virtual bool IsFull()  const = 0;          // контейнер полный
};
// абстрактный базовый класс Стек
class AbstractStack: public AbstractContainer
{
public:
        virtual void push(const int& n) = 0;       // втолкнуть в стек
        virtual int pop(void)  = 0;                // вытолкнуть из стека
};
// абстрактный базовый класс Очередь
class AbstractQueue: public AbstractContainer
{
public:
        virtual void push(const int& n) = 0; // втолкнуть в очередь
        virtual void del(int& n) = 0;                      // удалить из очереди
};
```

Конкретные классы

```
class ArrayStackIterator;          // предваряющее объявление
class QueueIterator;               // предваряющее объявление

// класс Стек на базе массива
class ArrayStack: public AbstractStack
{
protected:
        int             size;   // размерность массива
        int*    p;              // указатель на массив
        int             top;    // верхушка стека
public:
```

```
        ArrayStack(int _size);
        ArrayStack(ArrayStack &s);
        ~ArrayStack();
        void push(const int& n);            // втолкнуть в стек
        int pop(void);                      // вытолкнуть из стека
        bool IsEmpty() const;
        bool IsFull() const ;
        friend class ArrayStackIterator;
};

// класс стек очередь на базе списка - определить самостоятельно

// класс очередь
class ArrayQueue: public AbstractQueue
{
protected:
        int             size;               // размерность массива
        int*    p;                          // указатель на массив
        int             head;               // индекс первого занятого элемента
        int             n;                  // количество элементов в очереди
public:
        ArrayQueue(int _size);
        ArrayQueue(ArrayQueue &q);
        ~ArrayQueue();
        void push(const int& n);            // втолкнуть в очередь
        int del(void);                      // удалить из очереди
        bool IsEmpty() const;
        bool IsFull() const;
        friend class QueueIterator;
};

// класс Дек
class ArrayDequeue: public ArrayQueue
{
public:
        ArrayDequeue(int _size);
        ArrayDequeue(const ArrayDequeue &d);
        ~ArrayDequeue();
        virtual int pop(void);              // вытолкнуть из дека со стороны push
        virtual void ins(const int& n);             // вставить в дек со стороны del
};

// класс Итератор стека
class ArrayStackIterator: public AbstractIterator
{
        ArrayStack      &a;                 // ссылка на стек
        int             pos;                // текущая позиция итератора
        StackIterator();
public:
        ArrayStackIterator(const ArrayStack& _a);
        bool InRange() const;               // индекс в допустимых пределах
        void Reset();                       // сбросить индекс в начало
        int& operator *() const;            // разыменование (чтение элемента)
        void operator ++();                 // сдвиг на элемент
};

// класс Итератор очереди
class QueueIterator: public AbstractIterator
{
        ArrayQueue      &a;                 // ссылка на очередь
        int             pos;                // текущая позиция итератора
        QueueIterator();
public:
        QueueIterator(ArrayQueue& _a);
        bool InRange();                     // индекс в допустимых пределах
        void Reset();                       // сбросить индекс в начало
```

```
        int& operator *() const;            // разыменование (чтение элемента)
        void operator ++();                 // сдвиг на элемент
};
```

Замечания:
Интерфейсы и данные-члены можно изменить.

Индивидуальные варианты:

5.1. Используя абстрактные классы, реализовать конкретные классы: ArrayStack **(на массиве)** и ArrayStackIterator.
5.2. Используя абстрактные классы, реализовать конкретные классы: ListStack, ListStackIterator
5.3. Используя абстрактные классы, реализовать конкретные классы: ArrayDequeue, QueueIterator
5.4. Используя абстрактные классы, реализовать конкретные классы: ArrayQueue**(на массиве)**, QueueIterator
5.5. Используя абстрактные классы, реализовать конкретные классы: ArrayStack**(на массиве)** и ArrayStackIterator.
5.6. Используя абстрактные классы, реализовать конкретные классы: ArrayQueue**(на массиве)**, QueueIterator
5.7. Используя абстрактные классы, реализовать конкретные классы: ArrayStack**(на массиве)** и ArrayStackIterator.
5.8. Используя абстрактные классы, реализовать конкретные классы: ArrayDequeue**(на массиве)**, QueueIterator
5.9. Используя абстрактные классы, реализовать конкретные классы: ArrayQueue**(на массиве)**, QueueIterator
5.10. Используя абстрактные классы, реализовать конкретные классы: ListStack, ListStackIterator
5.11. Используя абстрактные классы, реализовать конкретные классы: ArrayQueue**(на массиве)**, QueueIterator
5.12. Используя абстрактные классы, реализовать конкретные классы: ArrayStack**(на массиве)** и ArrayStackIterator.
5.13. Используя абстрактные классы, реализовать конкретные классы: ArrayDequeue**(на массиве)**, QueueIterator
5.14. Используя абстрактные классы, реализовать конкретные классы: ArrayQueue**(на массиве)**, QueueIterator
5.15. Используя абстрактные классы, реализовать конкретные классы: ListStack, ListStackIterator

Лабораторная № 6

Сдать оформленные работы до 9.05.

Функции-шаблоны. Классы-шаблоны. Потоки
Для ввода-вывода данных использовать потоки.

Создать параметризованный класс данных (шаблон)- массив данных о студентах(с динамическим выделением памяти) - MASSIV". Этот класс предназначен для ввода, хранения и вывода информации. Переменными-членами класса являются количество записей и указатель на массив структур.
Данные о каждой записи хранятся в структуре.

Необходимо проверить шаблон для трех типов (классов): Element1, Element2, Element3, Для каждой структуры должен быть определен оператор вывода в текстовый поток (файл) (operator <<). Эти операторы понадобятся для реализации функций основных классов.
1) Для класса- шаблона " MASSIV" кроме обязательных конструктора, деструктора и т.д., реализовать основные функции для работы с массивом:

a) ввод из текстового потока (файла) массива данных (operator >>)
b) вывод в текстовый поток (файл) массива данных (operator <<)
c) вывод в бинарный поток (файл) массива данных
d) вывод на консоль массива данных перегрузка оператора
e) operator=
f) доступ к элементу (operator[])
g) создать па основе этих данных массив новый массив (функц. 1 согласно индивидуальному варианту).
h) Упорядочить полученный массив в порядке (функц. 3 согласно индивидуальному варианту).

2) Создать отдельную функцию шаблон, с параметром класс-шаблон MASSIV и элемент структуры (функц. 2 согласно индивидуальному варианту)

3) Реализовать вне класса- шаблона одну дружественную функцию и любой один метод класса шаблона.

Частичное описание класса шаблона:

```
template<class T>
class MASSIV
   {
   int count;
   T*M;
public:
   MASSIV ();
   MASSIV(int );
 MASSIV(MASSIV <T> & m);
   ~MASSIV();
   …
   };
```

Частичное описание функции main()

```
int main()
{
   MASSIV <Element1> M;
   cout<<M;
…
   MASSIV <Element2> K;
   cout<<K;
…
Element3 st // или Element1 st или Element2 st
   MASSIV <Element3> R;
```

Согласно пунктам в индивидуальном варианте в проекте должны быть 3-и функции:

X.function(K,M); //Функция внутри класса-шаблона

search_function2 (X , st); //функция-шаблон, первый параметр согласно индивидуальному варианту

X.Sortfunction3();//какой контейнер *X* (M, K, R) - указано в индивидуальном варианте

}

Важно:

В шаблоне MASSIV, обрабатывать только массив, поля структур не использовать!!

В main должны быть объекты MASSIV для типов :doble (char или float или long) и вызваны методы поиска и сортировки.

Индивидуальные варианты:

1. Element1 - «Студенты факультета», Element2 - «Студенты имеющие задолженности по сессии» и Element3 - «Студенты, сдавшие сессию».

Структуры:

struct Element1
{
char Name[50];
char Addr[50];
int Count;
};
	struct Element2
	{
	char Name[50];
	…
	};
		struct Element3
		{
		char Name[50];
		char Addr[50];
		int Count;
		…
		};

Функции:

1. Сформировать массив Element3, как разность Element1 и Element2, содержащий данные только о тех студентах, которые присутствуют в первом массиве и отсутствуют во втором.
2. Поиск в Element1 студентов, проживающих на одной улице
3. Сортировка Element2 по полю Name.

2. Element1 - «Студенты дневного отделения», Element2 - «Студенты-заочники» и Element3 - « Все студенты».

Структуры:

struct Element1
{
char Name[50];
int kurs;
int Otdelenie;
 	
};
	struct Element2
	{
	char Name[50];
	int kurs;
	…
	};
		struct Element3
		{
		char Name[50];
		int kurs;
		int Otdelenie;
		int day_ev
		
		…
		};

Функции:

1. Сформировать массив Element3, как объединение Element1 и Element2, содержащий данные только о тех студентах, которые присутствуют в обоих массивах.
2. Поиск в Element3 студентов К-го курса. К – ввести с консоли

3. Сортировка Element1 по полю Otdelenie.

3. Element1 - «Студенты факультета», Element2 - «Студенты и аспиранты - активисты» и Element3 - « Активные студенты с отличной успеваемостью».

Структуры:

struct Element1
{
char Name[50];
int SredniyBal;
 	
};
	struct Element2
	{
	char Name[50];
	char Space[50];//область деятельности (спорт, искусство, общественная жизньи т.д.)
	…
	};
		struct Element3
		{
		char Name[50];
		int SredniyBal;…
		};

Функции:

1. Сформировать массив Element3, как пересечение Element1 и Element2, содержащий данные только о тех студентах отличниках, которые присутствуют в первом массиве и присутствуют во втором.
2. Поиск в Element2 студентов активных в спорте
3. Сортировка Element2 по полю Space.

4. «Студенты, взявшие книги в библиотеке БГУ», Element2- «Читатели, вернувшие все книги») и Element3 «Студенты, не сдавшие книги».

Структуры:

struct Element1
{
char Name[50];
int Count;
char Addr[50];
};
	struct Element2
	{
	char Name[50];
	char Addr[50];
	…
	};
		struct Element3
		{ char Name[50];
		char Addr[50];
		double Price;
		int Count;
		};

Функции:

1. Создать па основе этих данных массив «разности» типа Element3, содержащий данные только о тех студентах, которые присутствуют в первом массиве и отсутствуют во втором.
2. Поиск в Element1 студентов взявших больше К книг. К – ввести с консоли.
3. Сортировка Element2 по полю Addr.

5. Element1 - «Студенты факультета», Element2 - «Студенты имеющие задолженности по сессии» и Element3 - «Студенты, сдавшие сессию».

Структуры:

struct Element1
{
char Name[50];
char Addr[50];
};
	struct Element2
	{
	char Name[50];
	int Count;
	…
	};
		struct Element3
		{
		char Name[50];
		char Addr[50];
		int Count;
		…
		};

Функции:

1. Сформировать массив Element3, как разность Element1 и Element2, содержащий данные только о тех студентах, которые присутствуют в первом массиве и отсутствуют во втором.
2. Поиск в Element3 студентов-однофамильцев
3. Сортировка Element3 по полю Count.

6. Element1 - «Студенты дневного отделения», Element2 - «Студенты-заочники» и Element3 - « Все студенты».

Структуры:

struct Element1
{
char Name[50];
int kurs;
int Otdelenie;
 	
};
	struct Element2
	{
	char Name[50];
	int kurs;
	int Otdelenie;
	…
	};
		struct Element3
		{
		char Name[50];
		int kurs;
		int Otdelenie;
		…
		};

Функции:

1. Сформировать массив Element3, как объединение Element1 и Element2.
2. Поиск в Element2 студентов одной специальности
3. Сортировка Element1 по полю kurs;.

7. Element1 - «Студенты факультета», Element2 - «Студенты и аспиранты - активисты» и Element3 - « Активные студенты с отличной успеваемостью».

Структуры:

struct Element1
{
char Name[50];
int SredniyBal;
 	
};
	struct Element2
	{
	char Name[50];
	char Space[50];//область деятельности (спорт, искусство, общественная жизнь и т.д.)
	};
		struct Element3
		{
		char Name[50];
		int SredniyBal;…
		};

Функции:

1. Сформировать массив Element3, как пересечение Element1 и Element2, содержащий данные только о тех студентах отличниках, которые присутствуют в первом массиве и присутствуют во втором.
2. Поиск в Element2 студентов активных в общественной жизни
3. Сортировка Element1 по полю SredniyBal;.

8. Element1 - «Студенты, взявшие книги в библиотеке БГУ», Element2- «Читатели, вернувшие все книги») и Element3 «Студенты, не сдавшие книги».

Структуры:

struct Element1
{
char Name[50];
char Addr[50];
 	
};
	struct Element2
	{
	char Name[50];
	int data;
	};
		struct Element3
		{ char Name[50];
		char Addr[50];
		double Price;
		};

Функции:

1. Создать па основе этих данных массив «разности» типа Element3, содержащий данные только о тех студентах, которые присутствуют в первом массиве и отсутствуют во втором.
2. Поиск в Element2 студентов – вернувших книги после даты D. D – ввести с консоли.
3. Сортировка Element2 по полю data;

9. Element1 - «Студенты факультета», Element2 - «Студенты имеющие задолженности по сессии» и Element3 - «Студенты, сдавшие сессию».

Структуры:

struct Element1
{
char Name[50];
int kurs;
};
	struct Element2
	{
	char Name[50];
	int Count;
	int kurs;
	…
	};
		struct Element3
		{
		char Name[50];
		int kurs;
		
		…
		};

Функции:

1. Сформировать массив Element3, как разность Element1 и Element2, содержащий данные только о тех студентах, которые присутствуют в первом массиве и отсутствуют во втором.
2. Поиск в Element1 студентов – старшекурсников
3. Сортировка Element3 по полю kurs;.

10. Element1 - «Студенты дневного отделения», Element2 - «Студенты-заочники» и Element3 - « Все студенты».

Структуры:

struct Element1
{
char Name[50];
int kurs;
int group;
int Otdelenie;
 	
};
	struct Element2
	{
	char Name[50];
	int kurs;
	int Otdelenie;
	…
	};
		struct Element3
		{
		char Name[50];
		int kurs;
		int Otdelenie;
		int group;
		
		};

Функции:

1. Сформировать массив Element3, как объединение Element1 и Element2, содержащий данные только о тех студентах, которые присутствуют в обоих массивах.
2. Поиск в Element1 студентов одной группы
3. Сортировка Element1 по полю group.

11. Element1 - «Студенты факультета», Element2 - «Студенты и аспиранты - активисты» и Element3 - « Активные студенты с отличной успеваемостью».

Структуры:

struct Element1
{
char Name[50];
int SredniyBal;
	struct Element2
	{
	char Name[50];
	char Space[50];//область
		struct Element3
		{
		char Name[50];
		int SredniyBal;

int kurs; };	деятельности (спорт, искусство, общественная жизнь и т.д.) int kurs; … };	int kurs; };

Функции:

1. Сформировать массив Element3, как пересечение Element1 и Element2, содержащий данные только о тех студентах отличниках, которые присутствуют в первом массиве и присутствуют во втором.
2. Поиск в Element3 студентов со средним баллом >К. К – ввести с консоли
3. Сортировка Element3 по полю SredniyBal;

12. Element1 - «Студенты, взявшие книги в библиотеке БГУ», Element2 - «Читатели, вернувшие все книги») и Element3 «Студенты, не сдавшие книги».

Структуры:

struct Element1
{ char Addr[50];
char Name[50];
double Price;
};
	struct Element2
	{
	char Name[50];
	char Addr[50];…
	};
		struct Element3
		{ char Name[50];
		char Addr[50];
		double Price;
		};

Функции:

1. Создать па основе этих данных массив «разности» типа Element3, содержащий данные только о тех студентах, которые присутствуют в первом массиве и отсутствуют во втором.
2. Поиск в Element3 студентов – задолжников на сумму>К. К – ввести с консоли.
3. Сортировка Element1 по полю Addr.

13. Element1 - «Студенты факультета», Element2 - «Студенты в академическом отпуске» и Element3 - «Студенты на обучении(фактически)».

Структуры:

struct Element1
{
char Name[50];//ФИО
int kurs;
};
	struct Element2
	{
	char Name[50]; //ФИО
	
	…
	};
		struct Element3
		{
		char Name[50]; //ФИО
		int kurs;
		…
		};

Функции:

1. Сформировать массив Element3, как разность Element1 и Element2, содержащий данные только о тех студентах, которые присутствуют в первом массиве и отсутствуют во втором.
2. Поиск в Element3 студентов – с фамилией Иванов(а)
3. Сортировка Element3 по полю kurs.

14. Element1 - «Студенты факультета», Element2 - «Студенты спортсмены и в сборной БГУ » и Element3 - « Студенты не спортсмены».

Структуры:

struct Element1
{
char Name[50];
char Sport[50];//спорт,
int SredniyBal;
 	
};
	struct Element2
	{
	char Name[50];
	char Sport[50];//спорт,
	
	};
		struct Element3
		{
		char Name[50];
		};

Функции:

1. Сформировать массив Element3, как разность Element1 и Element2, содержащий данные только о тех студентах отличниках, которые присутствуют в первом массиве и отсутствуют во втором.
2. Поиск в Element2 студентов из сборной.
3. Сортировка Element2 по полю Sport;.

15. Element1 - «Студенты дневного отделения», Element2 - «Студенты-заочники» и Element3 - « Все студенты».

Структуры:

struct Element1 { char Name[50]; int kurs; int Otdelenie; string tel;	struct Element2 { char Name[50]; int kurs; int Otdelenie; char Addr[50];…	struct Element3 { char Name[50]; int kurs; int Otdelenie; …

};	};	};

Функции:

1. Сформировать массив Element3, как объединение Element1 и Element2.
2. Поиск в Element2 студентов по адресу
3. Сортировка Element1 по полю tel.