1.

Определить класс CCompexVector для работы с векторами комплексных чисел. Внутри класса вектор должен быть реализован с помощью указателя на тип complex.

Длина вектора задается в конструкторе класса. В классе должны быть определены необходимые конструкторы (в том числе конструктор копирования и перемещения), деструктор, операторы присваивания (копированием и перемещением), сложения, вычитания, скалярного умножения, «, инкремент ++ и декремент −− (справа и слева), увеличивающий и уменьшающий (если это возможно) длину вектора на 1. Последние операторы в префиксной форме должны обрабатывать (дублировать или удалять) первый элемент вектора, в постпрефиксной – последний.

При сложении и вычитании размер результата – это минимум из размеров исходных векторов. При скалярном умножении недостающие координаты считать нулевыми.

В этом классе должна быть создана [чисто] виртуальная Функция вывода данных класса в файл вида

virtual int output()=0;

От класса CCompexVector надо породить два класса CData0 и CData1, в первом из которых переопределить функцию

int output(); как функцию вывода данных класса в файл в одну строку, а во втором данную функцию определить как функцию вывода данных класса в столбец (т.е. по одному элементу данных в одну строку). В файле с исходными данными в каждой строке задаются данные для одного экземпляра класса, порожденного от CCompexVector. Данные задачи задаются в виде:

I FileName Data

где I = 0 или I= 1, FileName – имя выходного файла, Data – все данные одного объекта, разделенные пробелами без указания длины вектора. Для каждой строки исходного файла надо создать объект класса CData0, если I == 0, либо экземпляр класса CData1, если I == 1 и заполнить его данными Data. Имя выходного файла следует занести в соответствующее поле созданного класса. Указатели на созданные объекты надо поместить в массив указателей на базовый класс CCompexVector **arr. Каждый новый объект должен создаваться функцией вида

CCompexVector *CreateData(const char *str, CFabricData **f);. где f – массив фабрик для создания I-го дочернего класса от CCompexVector. Далее надо в цикле для каждого объекта из массива arr вызвать функцию output().

Также надо написать разумный тест на все реализованные функции класса. Тест должен содержать сложение объектов, на которые указывают сохраненные в массиве arr указатели и их циклический сдвиг.

2.

Определить класс CIntN для работы с целыми беззнаковыми числами, состоящими из N десятичных цифр.

Внутри класса число должно быть реализовано с помощью указателя на тип char. Число N задается в конструкторе класса.

В классе должны быть определены необходимые конструкторы (в том числе конструктор копирования и перемещения), деструктор, операторы присваивания (копированием и перемещением), сложения, вычитания, «, инкремент ++ и декремент −− (справа и слева как умножение и деление на 10).

При сложении и вычитании количество значащих десятичных цифр результата (N) может отличаться от N аргументов.

В этом классе должна быть создана [чисто] виртуальная Функция вывода данных класса в файл вида

virtual int output()=0;

От класса CIntN надо породить два класса CData0 и CData1, в первом из которых переопределить функцию

int output(); как функцию вывода данных класса в файл в одну строку, а во втором данную функцию определить как функцию вывода данных класса в столбец (т.е. по одному элементу данных в одну строку). В файле с исходными данными в каждой строке задаются данные для одного экземпляра класса, порожденного от CIntN . Данные задачи задаются в виде:

I FileName Data

где I = 0 или I= 1, FileName – имя выходного файла, Data – данные одного объекта (последовательность десятичных цифр без пробелов) без указания количества десятичных цифр. Для каждой строки исходного файла надо создать объект класса CData0, если I == 0, либо экземпляр класса CData1, если I == 1 и заполнить его данными Data. Имя выходного файла следует занести в соответствующее поле созданного класса. Указатели на созданные объекты надо поместить в массив указателей на базовый класс CIntN **arr. Каждый новый объект должен создаваться функцией вида

CIntN *CreateData(const char *str, CFabricData **f);. где f – массив фабрик для создания I-го дочернего класса от CIntN. Далее надо в цикле для каждого объекта из массива arr вызвать функцию output().

Также надо написать разумный тест на все реализованные функции класса. Тест должен содержать сложение объектов, на которые указывают сохраненные в массиве arr указатели и их циклический сдвиг.

3.

Определить класс CRat для работы с вектором несократимых дробей вида p_i/q_i, где p_i — целое, q_i -— натуральное. Длина вектора задается в конструкторе класса. В классе должны быть определены необходимые конструкторы (в том числе конструктор копирования и перемещения), деструктор, операторы присваивания (копированием и перемещением), сложения, вычитания, деления, «, инкремент ++ и декремент −− (справа и слева), дублирующий и удаляющий (если это возможно) первый (последний) элемент вектора. Размер результата при сложении, вычитании, умножении и делении – это минимум из размерив исходных векторов.

В этом классе должна быть создана [чисто] виртуальная Функция вывода данных класса в файл вида

virtual int output()=0;

От класса CRat надо породить два класса CData0 и CData1, в первом из которых переопределить функцию

int output(); как функцию вывода данных класса в файл в одну строку, а во втором данную функцию определить как функцию вывода данных класса в столбец (т.е. по одному элементу данных в одну строку). В файле с исходными данными в каждой строке задаются данные для одного экземпляра класса, порожденного от CRat. Данные задачи задаются в виде:

I FileName Data

где I = 0 или I= 1, FileName – имя выходного файла, Data – все данные одного объекта, разделенные пробелами без указания длины вектора. Для каждой строки исходного файла надо создать объект класса CData0, если I == 0, либо экземпляр класса CData1, если I == 1 и заполнить его данными Data. Имя выходного файла следует занести в соответствующее поле созданного класса. Указатели на созданные объекты надо поместить в массив указателей на базовый класс CRat **arr. Каждый новый объект должен создаваться функцией вида

CRat *CreateData(const char *str, CFabricData **f);. где f – массив фабрик для создания I-го дочернего класса от CRat. Далее надо в цикле для каждого объекта из массива arr вызвать функцию output().

Также надо написать разумный тест на все реализованные функции класса. Тест должен содержать сложение объектов, на которые указывают сохраненные в массиве arr указатели и их циклический сдвиг.

4.

Определить класс CString для работы со строками, длина которых хранится в самом классе (строки произвольной длины). Длина строки задается в конструкторе класса. Внутри класса строка должна быть реализована с помощью указателя (char *).

В классе должны быть определены необходимые конструкторы (в том числе конструктор копирования и перемещения), деструктор, операторы присваивания (копированием и перемещением), присваивания обычной строки переменной типа CString (например, Cstring str; str="aaaa";), «, инкремент ++ и декремент −− (справа и слева: добавление символа ! в конец строки и обрезание последнего символа, если это возможно), сложения (конкатенация), умножения (слева и справа) строки на беззнаковое целое число (оно равносильно сложению строки с собой нужное число раз),

В этом классе должна быть создана [чисто] виртуальная Функция вывода данных класса в файл вида

virtual int output()=0;

От класса CString надо породить два класса CData0 и CData1, в первом из которых переопределить функцию

int output(); как функцию вывода данных класса в файл в одну строку, а во втором данную функцию определить как функцию вывода данных класса в столбец (т.е. по одному элементу данных в одну строку). В файле с исходными данными в каждой строке задаются данные для одного экземпляра класса, порожденного от CString. Данные задачи задаются в виде:

I FileName Data

где I = 0 или I= 1, FileName – имя выходного файла, Data – данные одного объекта (строка без указания длины). Для каждой строки исходного файла надо создать объект класса CData0, если I == 0, либо экземпляр класса CData1, если I == 1 и заполнить его данными из введенной строки. Имя выходного файла следует занести в соответствующее поле созданного класса. Указатели на созданные объекты надо поместить в массив указателей на базовый класс CString **arr. Каждый новый объект должен создаваться функцией вида

CString *CreateData(const char *str, CFabricData **f);. где f – массив фабрик для создания I-го дочернего класса от CString. Далее надо в цикле для каждого объекта из массива arr вызвать функцию output().

Также надо написать разумный тест на все реализованные функции класса. Тест должен содержать сложение объектов, на которые указывают сохраненные в массиве arr указатели и их циклический сдвиг.

5.

Определить класс CMatrix для работы с квадратной матрицей целых чисел. Внутри класса матрица должна быть реализована с помощью указателя (int **). Размер матрицы задается в конструкторе класса.

В классе должны быть определены необходимые конструкторы (в том числе конструктор копирования и перемещения), деструктор, операторы присваивания (копированием и перемещением), «, инкремент ++ и декремент −− (справа и слева: добавление нулевой строки и нулевого столбца в начало матрицы (вычеркивания, если возможно, первой строки и первого столбца), сложения, вычитания матриц, умножения матрицы (слева и справа) на число. При сложении и вычитании размер результата – это минимум из размеров исходных матриц, лишние строки матрицы большего размера игнорируются.

В этом классе должна быть создана [чисто] виртуальная Функция вывода данных класса в файл вида

virtual int output()=0;

От класса CMatrix надо породить два класса CData0 и CData1, в первом из которых переопределить функцию

int output(); как функцию вывода данных класса в файл в одну строку (построчно), а во втором данную функцию определить как функцию вывода данных класса как квадратную матрицу. В файле с исходными данными в каждой строке задаются данные для одного экземпляра класса, порожденного от CMatrix. Данные задачи задаются в виде:

I FileName Data

где I = 0 или I= 1, FileName – имя выходного файла, Data – все данные одного объекта, разделенные пробелами, т.е. элементы матрицы без указания размера. Для каждой строки исходного файла надо создать объект класса CData0, заполнить его данными Data. Имя выходного файла следует занести в соответствующее поле созданного класса.

Указатели на созданные объекты надо поместить в массив указателей на базовый класс CMatrix **arr. Каждый новый объект должен создаваться функцией вида

CMatrix *CreateData(const char *str, CFabricData **f);. где f – массив фабрик для создания I-го дочернего класса от CMatrix. Далее надо в цикле для каждого объекта из массива arr вызвать функцию output().

Также надо написать разумный тест на все реализованные функции класса. Тест должен содержать сложение объектов, на которые указывают сохраненные в массиве arr указатели и их циклический сдвиг.

6.

Определить класс CStack для работы со стеком целых чисел ограниченного размера. Глубина стека задается в конструкторе класса. В классе должны быть определены необходимые конструкторы (в том числе конструктор копирования и перемещения), деструктор, операторы присваивания (копированием и перемещением), «, инкремент ++ и декремент −− (справа и слева, работают как push и pop), сложения (результатом является стек из содержимого обоих стеков, при этом глубины суммируется, использовать push и pop), функции push (добавление числа в стек, если это возможно, возвращает информацию об успешности) и pop (удаление вершины, если это возможно, с сохранением вытесненной вершины, возвращает информацию об успешности),

В этом классе должна быть создана [чисто] виртуальная Функция вывода данных класса в файл вида

virtual int output()=0;

От класса CStack надо породить два класса CData0 и CData1, в первом из которых переопределить функцию

int output(); как функцию вывода данных класса в файл в одну строку, а во втором данную функцию определить как функцию вывода данных класса в столбец (т.е. по одному элементу данных в одну строку). В файле с исходными данными в каждой строке задаются данные для одного экземпляра класса, порожденного от CStack. Данные задачи задаются в виде:

I FileName Data

где I = 0 или I= 1, FileName – имя выходного файла, Data – все данные стека, разделенные пробелами без указания глубины, которая определяется как минимально возможная. Для каждой строки исходного файла надо создать объект класса CData0, если I == 0, либо экземпляр класса CData1, если I == 1 и заполнить его данными Data. Имя выходного файла следует занести в соответствующее поле созданного класса. Указатели на созданные объекты надо поместить в массив указателей на базовый класс CStack **arr. Каждый новый объект должен создаваться функцией вида

CStack *CreateData(const char *str, CFabricData **f);. где f – массив фабрик для создания I-го дочернего класса от CStack. Далее надо в цикле для каждого объекта из массива arr вызвать функцию output().

Также надо написать разумный тест на все реализованные функции класса. Тест должен содержать сложение объектов, на которые указывают сохраненные в массиве arr указатели и их циклический сдвиг.

7.

Определить класс CPriority_queue для работы с приоритетной очередью целых чисел ограниченного размера. Глубина очереди задается в конструкторе класса. В классе должны быть определены необходимые конструкторы (в том числе конструктор копирования и перемещения), деструктор, операторы присваивания (копированием и перемещением), «, декремент ——(справа и слева, работает как pop), сложения с целым числом (работает как push к копии очереди), функции push (добавление числа в очередь, при этом максимальное число должно встать в голову очереди) и pop (удаление вершины, если это возможно, с сохранением вытесненной вершины, возвращает информацию об успешности),

В этом классе должна быть создана [чисто] виртуальная Функция вывода данных класса в файл вида

virtual int output()=0;

От класса CPriority_queue надо породить два класса CData0 и CData1, в первом из которых переопределить функцию

int output(); как функцию вывода данных класса в файл в одну строку, а во втором данную функцию определить как функцию вывода данных класса в столбец (т.е. по одному элементу данных в одну строку). В файле с исходными данными в каждой строке задаются данные для одного экземпляра класса, порожденного от CPriority_queue . Данные задачи задаются в виде:

I FileName Data

где I = 0 или I= 1, FileName – имя выходного файла, Data – все данные одного объекта, разделенные пробелами без указания глубины, которая определяется как минимально возможная. Для каждой строки исходного файла надо создать объект класса CData0, если I == 0, либо экземпляр класса CData1, если I == 1 и заполнить его данными Data. Имя выходного файла следует занести в соответствующее поле созданного класса. Указатели на созданные объекты надо поместить в массив указателей на базовый класс CPriority_queue **arr. Каждый новый объект должен создаваться функцией вида

CPriority_queue *CreateData(const char *str, CFabricData **f);. где f – массив фабрик для создания I-го дочернего класса от CPriority_queue. Далее надо в цикле для каждого объекта из массива arr вызвать функцию output().

Также надо написать разумный тест на все реализованные функции класса. Тест должен содержать сложение с числом объектов, на которые указывают сохраненные в массиве arr указатели и их циклический сдвиг.

8. Определить класс CSet для работы с битовым множеством целых чисел. Диапазон чисел задается в конструкторе класса. Внутри класса множество должно быть реализовано с помощью указателя. Принадлежность числа множеству означает, что бит, соответствующей этому числу, равен 1, в случае нулевого бита число не принадлежит множеству.

В классе должны быть определены необходимые конструкторы (в том числе конструктор копирования и перемещения), деструктор, операторы присваивания (копированием и перемещением), «, инкремент ++ и декремент −− (справа и слева),), увеличивающий и уменьшающий (если это возможно) верхнюю границу диапазона, сложения (объединение множеств), вычитания (пересечения). При сложении верхняя граница диапазона результата равна максимуму из соответствующих границ слагаемых, нижняя – минимуму, при вычитании верхняя – минимуму, нижняя – максимуму.

В этом классе должна быть создана [чисто] виртуальная Функция вывода данных класса в файл вида

virtual int output()=0;

От класса CSet надо породить два класса CData0 и CData1, в первом из которых переопределить функцию

int output(); как функцию вывода данных класса в файл в одну строку, а во втором данную функцию определить как функцию вывода данных класса в столбец (т.е. по одному элементу данных в одну строку). В файле с исходными данными в каждой строке задаются данные для одного экземпляра класса, порожденного от CSet. Данные задачи задаются в виде:

I FileName Data

где I = 0 или I= 1, FileName – имя выходного файла, Data – все данные одного объекта (диапазон и числа множества в любом порядке), разделенные пробелами. Для каждой строки исходного файла надо создать объект класса CData0, если I == 0, либо экземпляр класса CData1, если I == 1 и заполнить его данными Data. Имя выходного файла следует занести в соответствующее поле созданного класса. Указатели на созданные объекты надо поместить в массив указателей на базовый класс CSet **arr. Каждый новый объект должен создаваться функцией вида

CSet *CreateData(const char *str, CFabricData **f);. где f – массив фабрик для создания I-го дочернего класса от CSet. Далее надо в цикле для каждого объекта из массива arr вызвать функцию output().

Также надо написать разумный тест на все реализованные функции класса. Тест должен содержать сложение объектов, на которые указывают сохраненные в массиве arr указатели и их циклический сдвиг.

9.

Определить класс CQueue для работы с (кольцевой) очередью целых чисел ограниченного размера. Максимальный размер очереди задается в конструкторе класса. В классе должны быть определены необходимые конструкторы (в том числе конструктор копирования и перемещения), деструктор, операторы присваивания (копированием и перемещением), «, инкремент ++ и декремент −− (справа и слева, работают как push и pop), сложения (результатом является очередь из содержимого обеих очередей, при этом глубины суммируется, использовать push и pop), функции push (добавление числа в очередь, если это возможно, возвращает информацию об успешности) и pop(удаление вершины, если это возможно, с сохранением вытесненной вершины, возвращает информацию об успешности). При сложении исходные очереди не должен изменяться.

В этом классе должна быть создана [чисто] виртуальная Функция вывода данных класса в файл вида

virtual int output()=0;

От класса CQueue надо породить два класса CData0 и CData1, в первом из которых переопределить функцию

int output(); как функцию вывода данных класса в файл в одну строку, а во втором данную функцию определить как функцию вывода данных класса в столбец (т.е. по одному элементу данных в одну строку). В файле с исходными данными в каждой строке задаются данные для одного экземпляра класса, порожденного от CQueue. Данные задачи задаются в виде:

I FileName Data

где I = 0 или I= 1, FileName – имя выходного файла, Data – все данные одного объекта, разделенные пробелами без указания глубины, которая определяется как минимально возможная. Для каждой строки исходного файла надо создать объект класса CData0, если I == 0, либо экземпляр класса CData1, если I == 1 и заполнить его данными Data. Имя выходного файла следует занести в соответствующее поле созданного класса. Указатели на созданные объекты надо поместить в массив указателей на базовый класс CQueue **arr. Каждый новый объект должен создаваться функцией вида

CQueue *CreateData(const char *str, CFabricData **f);. где f – массив фабрик для создания I-го дочернего класса от CQueue. Далее надо в цикле для каждого объекта из массива arr вызвать функцию output().

Также надо написать разумный тест на все реализованные функции класса. Тест должен содержать сложение объектов, на которые указывают сохраненные в массиве arr указатели и их циклический сдвиг.

10.

Определить класс CList для работы со списком целых чисел. В классе должны быть определены необходимые конструкторы (в том числе конструктор копирования и перемещения), деструктор, операторы присваивания (копированием и перемещением), сложения (результатом является список из сумм элементов). «, инкремент ++ и декремент —— (справа и слева, дублирование последнего элемента, если это возможно, и удаление первого) функция push_back (добавление числа в конец списка) При сложении длина результата – это минимум исходных длин.

В этом классе должна быть создана [чисто] виртуальная Функция вывода данных класса в файл вида

virtual int output()=0;

От класса CList надо породить два класса CData0 и CData1, в первом из которых переопределить функцию

int output(); как функцию вывода данных класса в файл в одну строку, а во втором данную функцию определить как функцию вывода данных класса в столбец (т.е. по одному элементу данных в одну строку). В файле с исходными данными в каждой строке задаются данные для одного экземпляра класса, порожденного от CList. Данные задачи задаются в виде:

I FileName Data

где I = 0 или I= 1, FileName – имя выходного файла, Data – все данные одного объекта, разделенные пробелами без указания длины списка. Для каждой строки исходного файла надо создать объект класса CData0, если I == 0, либо экземпляр класса CData1, если I == 1 и заполнить его данными Data. Имя выходного файла следует занести в соответствующее поле созданного класса. Указатели на созданные объекты надо поместить в массив указателей на базовый класс CList **arr. Каждый новый объект должен создаваться функцией вида

CList *CreateData(const char *str, CFabricData **f);. где f – массив фабрик для создания I-го дочернего класса от CList. Далее надо в цикле для каждого объекта из массива arr вызвать функцию output().

Также надо написать разумный тест на все реализованные функции класса. Тест должен содержать сложение объектов, на которые указывают сохраненные в массиве arr указатели и их циклический сдвиг.

11.

Определить класс CMultiSet для работы с мультимножеством целых чисел в диапазоне от N_1 до N_2, диапазон задается в конструкторе класса. В качестве данных класса использовать массив целых чисел, значение элемента которого равно количеству числа в мультимножестве. В классе должны быть определены необходимые конструкторы, (в том числе конструктор копирования и перемещения), деструктор, операторы присваивания (копированием и перемещением), «, инкремент ++ и декремент —— (справа и слева, результат – увеличение или уменьшение одного из диапазонов), операторы сложения (объединение множеств), умножения (пересечения), при этом диапазоны исходных множеств могут быть различны.

В этом классе должна быть создана [чисто] виртуальная Функция вывода данных класса в файл вида

virtual int output()=0;

От класса CMultiSet надо породить два класса CData0 и CData1, в первом из которых переопределить функцию

int output(); как функцию вывода данных класса в файл в одну строку, а во втором данную функцию определить как функцию вывода данных класса в столбец (т.е. по одному элементу данных в одну строку). В файле с исходными данными в каждой строке задаются данные для одного экземпляра класса, порожденного от CMultiSet. Данные задачи задаются в виде:

I FileName Data

где I = 0 или I=1, FileName – имя выходного файла, Data – все данные одного объекта, разделенные пробелами: начало и конец диапазона и (в любом порядке) числа, входящее в мультимножество. Для каждой строки исходного файла надо создать объект класса CData0, если I == 0, либо экземпляр класса CData1, если I == 1 и заполнить его данными Data. Имя выходного файла следует занести в соответствующее поле созданного класса. Указатели на созданные объекты надо поместить в массив указателей на базовый класс CMultiSet **arr. Каждый новый объект должен создаваться функцией вида

CMultiSet *CreateData(const char *str, CFabricData **f);. где f – массив фабрик для создания I-го дочернего класса от CMultiSet. Далее надо в цикле для каждого объекта из массива arr вызвать функцию output().

Также надо написать разумный тест на все реализованные функции класса. Тест должен содержать сложение объектов, на которые указывают сохраненные в массиве arr указатели и их циклический сдвиг.

12.

Определить класс CPoly2 для работы с полиномом от 2 переменных степени не выше N (по каждой из переменных), N задается в конструкторе класса. Внутри класса полином должен быть реализован с помощью указателя (int **).

В классе должны быть определены необходимые конструкторы (в том числе конструктор копирования и перемещения), деструктор, операторы присваивания (копированием и перемещением), «, инкремент ++ и декремент −− (справа и слева: дифференцирование по переменным x и y). операторы сложения, вычитания.

В этом классе должна быть создана [чисто] виртуальная Функция вывода данных класса в файл вида

virtual int output()=0;

От класса CPoly2 надо породить два класса CData0 и CData1, в первом из которых переопределить функцию

int output(); как функцию вывода данных класса в файл в одну строку по возрастанию мономов $x^i y^j$ относительно лексикографического порядка (i, j), а во втором данную функцию определить как функцию вывода данных класса в строку по лексикографическому возрастанию пары (j, i). В файле с исходными данными в каждой строке задаются данные для одного экземпляра класса, порожденного от CPoly2. Данные задачи задаются в виде:

I FileName Data

где I = 0 или I= 1, FileName – имя выходного файла, Data – все данные одного объекта (коэффициенты многочлена, без указания степени N, которая определяется как минимально возможная). Для каждой строки исходного файла надо создать объект класса CData0, если I == 0, либо экземпляр класса CData1, если I == 1 и заполнить его данными Data. Имя выходного файла следует занести в соответствующее поле созданного класса. Указатели на созданные объекты надо поместить в массив указателей на базовый класс CPoly2 **arr. Каждый новый объект должен создаваться функцией вида

CPoly2 *CreateData(const char *str, CFabricData **f);. где f – массив фабрик для создания I-го дочернего класса от CPoly2. Далее надо в цикле для каждого объекта из массива arr вызвать функцию output().

Также надо написать разумный тест на все реализованные функции класса. Тест должен содержать сложение объектов, на которые указывают сохраненные в массиве arr указатели и их циклический сдвиг.

13.

Определить класс CArr для работы с упорядоченным по возрастанию массивом вещественных чисел. Длина массива задается в конструкторе класса. Внутри класса массив должен быть реализована с помощью указателя (int *).

В классе должны быть определены необходимые конструкторы (в том числе конструктор копирования и перемещения), деструктор, операторы присваивания (копированием и перемещением), «, инкремент ++ и декремент —— (справа и слева), увеличивающий и уменьшающий (если это возможно) длину массива путем дублирования (удаления) первого элемента (для префиксной формы) и последнего (для постпрефиксной), сложения (массив, полученный слиянием слагаемых).

В этом классе должна быть создана [чисто] виртуальная Функция вывода данных класса в файл вида

virtual int output()=0;

От класса CArr надо породить два класса CData0 и CData1, в первом из которых переопределить функцию

int output();

как функцию вывода данных класса в файл в одну строку, а во втором данную функцию определить как функцию вывода данных класса в столбец (т.е. по одному элементу данных в одну строку). В файле с исходными данными в каждой строке задаются данные для одного экземпляра класса, порожденного от CArr. Данные задачи задаются в виде:

I FileName Data

где I = 0 или I= 1, FileName – имя выходного файла, Data – все данные одного объекта, разделенные пробелами без указания длины массива. Для каждой строки исходного файла надо создать объект класса CData0, если I == 0, либо экземпляр класса CData1, если I == 1 и заполнить его данными Data. Имя выходного файла следует занести в соответствующее поле созданного класса. Указатели на созданные объекты надо поместить в массив указателей на базовый класс CArr **arr. Каждый новый объект должен создаваться функцией вида

CArr *CreateData(const char *str, CFabricData **f);. где f – массив фабрик для создания I-го дочернего класса от CArr. Далее надо в цикле для каждого объекта из массива arr вызвать функцию output().

Также надо написать разумный тест на все реализованные функции класса. Тест должен содержать сложение объектов, на которые указывают сохраненные в массиве arr указатели и их циклический сдвиг.

14.

Определить классы CVect для работы с массивом векторов на плоскости и CAngl для работы с массивом углов между векторами. Длина массивов задается в конструкторах классов.

В классах должны быть определены необходимые конструкторы (в том числе конструктор копирования и перемещения), деструктор, операторы присваивания (копированием и перемещением), «, инкремент ++ и декремент −− (справа и слева), увеличивающие и уменьшающие (если это возможно) длину массива путем дублирования (вычеркивания) последнего элемента, операторы сложения (CVect и CAngl, возвращающий CVect, а также CAngl и CVect, возвращающий CVect), вычитания (CVect из CVect, возвращающий CAngl).

В классе CVect должна быть создана [чисто] виртуальная Функция вывода данных класса в файл вида

virtual int output()=0;

От класса CVect надо породить два класса CData0 и CData1, в первом из которых переопределить функцию

int output(); как функцию вывода данных класса в файл в одну строку, а во втором данную функцию определить как функцию вывода данных класса в столбец (т.е. по одному элементу данных в одну строку). В файле с исходными данными в каждой строке задаются данные для одного экземпляра класса, порожденного от CVect. Данные задачи задаются в виде:

I FileName Data

где I = 0 или I= 1, FileName – имя выходного файла, Data – все данные одного объекта, разделенные пробелами (без указания размера массива).

Для каждой строки исходного файла надо создать объект класса CData0, если I == 0, либо экземпляр класса CData1, если I == 1 и заполнить его данными Data. Имя выходного файла следует занести в соответствующее поле созданного класса. Указатели на созданные объекты надо поместить в массив указателей на базовый класс CMultiSet **arr. Каждый новый объект должен создаваться функцией вида

CMultiSet *CreateData(const char *str, CFabricData **f);. где f – массив фабрик для создания I-го дочернего класса от CVect. Далее надо в цикле для каждого объекта из массива arr вызвать функцию output().

Также надо написать разумный тест на все реализованные функции класса. Тест должен содержать нахождение знакопеременной суммы объектов, на которые указывают сохраненные в массиве arr указатели и их циклический сдвиг.

15.

Определить класс CArr для работы с вектором (массивом) целых чисел. Внутри класса массив должен быть реализован с помощью указателя. Длина вектора задается в конструкторе класса.

В классе должны быть определены необходимые конструкторы (в том числе конструктор копирования и перемещения), деструктор, операторы присваивания (копированием и перемещением), «, инкремент ++ и декремент —— (справа и слева), увеличивающие и уменьшающие (если это возможно) длину вектора путем добавления в начало вектора числа 0 (удаления первого числа), сложения вектора с целым числом (добавление числа в конец массива), унитарного минуса - (удаление последнего элемента, если это возможно).

При сложении массива с числом и применении минуса исходный массив не должен изменяться.

От класса CArr надо породить два класса CData0 и CData1, в первом из которых переопределить функцию

int output(); как функцию вывода данных класса в файл в одну строку, а во втором данную функцию определить как функцию вывода данных класса в один столбец. В файле с исходными данными в каждой строке задаются данные для одного экземпляра класса, порожденного от CArr. Данные задачи задаются в виде:

I FileName Data

где I = 0 или I= 1, FileName – имя выходного файла, Data – все данные одного объекта (без указания размера массива). Для каждой строки исходного файла надо создать объект класса CData0, если I == 0, либо экземпляр класса CData1, если I == 1 и заполнить его данными Data. Имя выходного файла следует занести в соответствующее поле созданного класса. Указатели на созданные объекты надо поместить в массив указателей на базовый класс CArr **arr. Каждый новый объект должен создаваться функцией вида

CArr *CreateData(const char *str, CFabricData **f);. где f – массив фабрик для создания I-го дочернего класса от CArr. Далее надо в цикле для каждого объекта из массива arr вызвать функцию output().

Также надо написать разумный тест на все реализованные функции класса. Тест должен содержать сложение объектов, на которые указывают сохраненные в массиве arr указатели с заданным числом и их циклический сдвиг.