Лабораторная работа №4. Композиция, наследование, полиморфизм

Практические задания

На оценку «**удовлетворительно**» необходимо выполнить **все обязательные задания** и набрать **не менее 60 баллов**.

На оценку «**хорошо**» необходимо выполнить все обязательные и часть дополнительных заданий и набрать **не менее 250 баллов**.

На оценку «**отлично**» необходимо выполнить все обязательные и часть дополнительных заданий и набрать **не менее 350 баллов**.

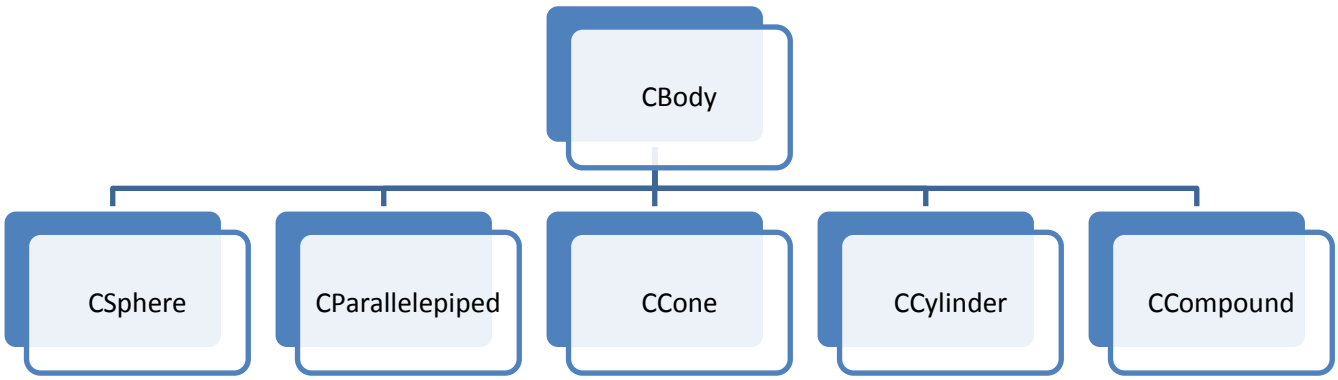
Внимание, дополнительные задания принимаются только после успешной защиты обязательных заданий.

Обязательные задания

Задание 1

Вариант 1 – 100 баллов

Разработайте классы следующей иерархии объемных тел[1]:



Каждое объемное тело обладает некоторым объемом, однородной плотностью и массой. Исключение составляет класс CCompound, моделирующий составное тело, состоящее из нескольких непересекающихся

[1] Для устранения дублирования кода может потребоваться вынесение одинаковой реализации методов простых тел в дополнительный базовый класс (например, CSolidBody)

объемных тел, плотность которого не является однородной. Средняя плотность составного тела может быть вычислена как отношение массы составляющих его тел к их суммарному объему.

Масса простого тела вычисляется через произведение его плотности и объема. **Масса составного тела** вычисляется через массу составляющих его частей.

Класс	**Описание**	**Свойства и методы**
CBody	Абстрактный класс объемных тел	• Плотность (при наличии составных тел метод также будет абстрактным) • Объем (абстрактный метод) • Масса (при наличии составных тел метод также будет абстрактным) • Получение полной информации об объемном теле в виде строки (абстрактный метод)
CSphere	Сфера	• Радиус
CParallelepiped	Параллелепипед	• Ширина • Высота • Глубина
CCone	Конус	• Радиус основания • Высота
CCylinder	Цилиндр	• Радиус основания • Высота
CCompound	Составное тело	• Добавление тела внутрь составного тела

При реализации метода добавления тела внутрь составного тела необходимо обрабатывать ситуацию с возможным зацикливанием, т.е. прямым или опосредованным добавлением составного объекта внутрь себя самого[2].

Программа должна считывать информацию об имеющемся наборе объемных тел из стандартного потока ввода (предусмотреть возможность ввода информации о составных объемных телах) и сохранять их в vector[3]. Из имеющихся тел программа должна вывести информацию **о теле с наибольшей массой**, а также о теле, которое будет **легче всего весить, будучи полностью погруженным в воду**[4] (плотность воды принять равной 1000 кг/м^3).

Возможна сдача работы без поддержки составных объемных тел. В этом случае работа будет принята с коэффициентом 0,7.

В комплекте с программой должны обязательно поставляться файлы, позволяющие проверить ее работу автоматически. Без них работа будет принята с коэффициентом 0.7 (коэффициент применяется к остальным коэффициентам).

---

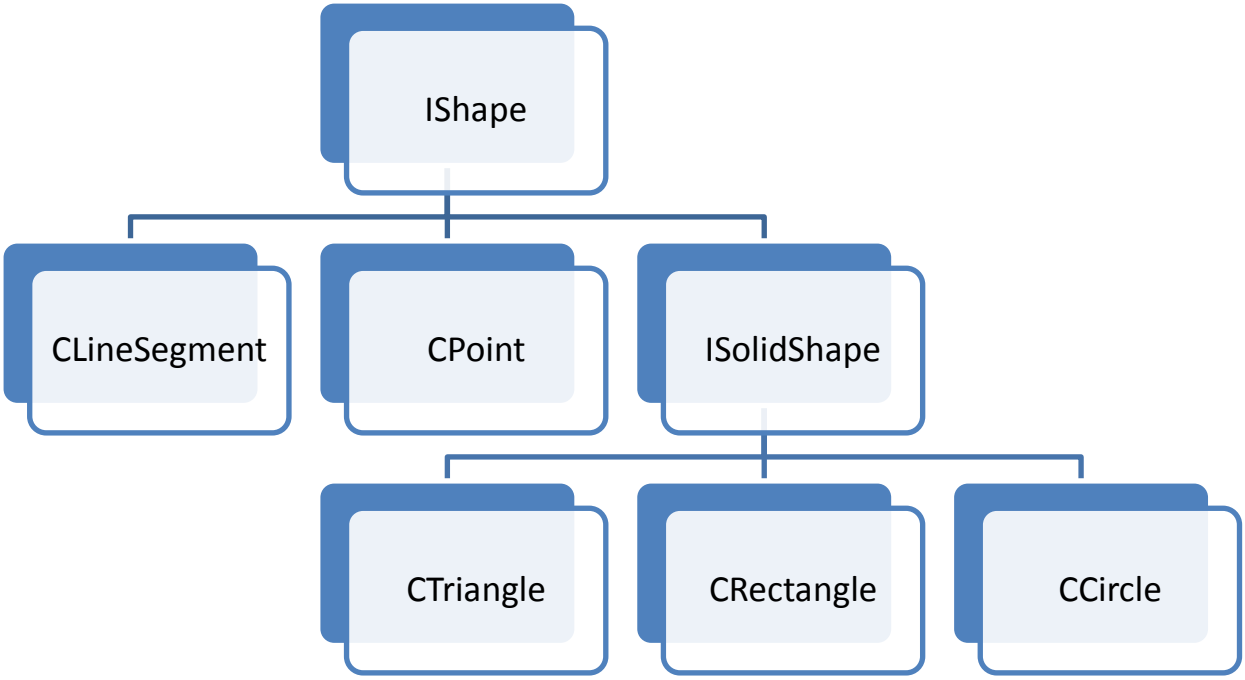
[2] Для решения данной задачи можно при добавлении объекта предварительно проверить при помощи оператора **dynamic_cast**, является ли добавляемый объект составным. В случае положительного результата проверки следует рекурсивно проверить, содержится ли в добавляемом составном объекте текущий объект (может быть полезным хранение ссылок на внешний составной объект).

[3] В данном случае контейнер vector должен хранить указатели на CBody. Лучше всего воспользоваться умными указателями типа std::shared_ptr (boost::shared_ptr при его отсутствии), unique_ptr или контейнером boost::ptr_vector.

[4] Легче всего в воде будет весить тело, для которого величина $F_{\text{тяжести}} - F_{\text{архимеда}} = (\rho_{\text{тела}} - \rho_{\text{воды}})gV$ будет минимальной. Это не обязательно будет тело с наименьшей массой.

Вариант 2 – 80 баллов

Разработайте классы следующей иерархии геометрических фигур:



Класс	Описание	Свойства
IShape	Базовый интерфейс геометрических фигур	Площадь фигуры (чисто виртуальный метод) Периметр фигуры (чисто виртуальный метод) Строковое представление фигуры (чисто виртуальный метод)
ISolidShape	Базовый интерфейс для всех замкнутых фигур	Строка, задающая цвет заливки внутренней области, например «Red»
CPoint	Точка	Координаты точки Площадь и периметр точки принимаются равными нулю
CLineSegment	Отрезок прямой	Точки, задающие координаты концов отрезка Длина отрезка прямой Площадь отрезка прямой принимается равной нулю, а периметр – равным длине отрезка.
CRectangle	Прямоугольник	Точка, задающая координаты верхнего левого угла прямоугольника Ширина Высота
CTriangle	Треугольник	Координаты трех точек в вершинах треугольника Отрезки, задающие три стороны треугольника
CCircle	Окружность	Точка, задающая координаты центра Радиус

При задании ряда классов фигур используйте композицию, для определения новых фигур на основе уже имеющихся. Например, отрезок прямой линии может быть реализован в виде композиции двух точек.

В базовом интерфейсе **IShape** должен быть метод std::string ToString()const возвращающий строковое представление фигуры. Не забудьте в базовом классе определить виртуальный деструктор.

Программа считывает в **std::vector** информацию о фигурах из стандартного **потока ввода** в следующем формате:
<тип фигуры> <параметры фигуры>, например:
Rectangle <20, 30>, 20x10

После считывания фигур необходимо вывести их в стандартный поток вывода **дважды**: сначала **в порядке возрастания площади**, а затем **в порядке убывания периметра**. При выводе фигур в стандартный поток вывода должна также выводиться площадь фигуры и периметр. Например:

Circle <10, 10>, R=10, S=314.15927, P=62,831853

Поместить в **std::vector** объекты разных типов не получится, однако вместо хранения самих объектов в нем следует хранить указатели на IShape. Вместо хранения сырых указателей в массиве предпочтительнее воспользоваться услугами умных указателей std::shared_ptr<IShape> или std::unique_ptr<IShape>, либо контейнером boost::ptr_vector<IShape>. В противном случае перед выходом из программы необходимо будет удалять созданные объекты вручную.

В комплекте с программой должны обязательно поставляться файлы, позволяющие проверить ее работу автоматически. Без них работа будет принята с коэффициентом 0.7.

Дополнительные задания

Задание 2 – 200 баллов

Спроектируйте класс **CPerson** (человек), обладающий следующими характеристиками: пол, возраст, имя, рост и вес. Определите в нем методы для изменения возраста (только в большую сторону), имени, роста (только в большую сторону) и веса.

Спроектируйте класс **CUniversity** (университет), обладающий названием, а также методом для изменения названия.

Создайте производный от CPerson класс **CStudent** (Студент), имеющий в качестве дополнительных характеристик номер года обучения (1-5), а также университет, в котором происходит обучение студента. Студент должен обладать методами, позволяющими изменить номер года обучения (в большую сторону), а также перевести студента в другой университет. Классу «Студент» не должна даваться возможность изменения названия университета[5].

Программе при запуске должны передаваться через параметры командной строки имена файлов, содержащих информацию об университетах и студентах. В ходе работы с программой пользователь должен иметь следующие возможности:

- Вывести список университетов
- Изменить название произвольного университета. При этом совпадение с названием одного из имеющихся университетов не допускается.

[5] Для инициализации и изменения данных об университете студенту должна передаваться константная ссылка на университет. Благодаря этому, из класса CStudent будет невозможно вызвать неконстантные методы университета. Отметим следующие очевидные, хотя и не всегда принимаемые во внимание новичками, особенности:

- Университет не является составной частью студента, поэтому класс «Студент» должен хранить поле типа указатель или ссылка на внешний объект типа «Университет», а не само значение типа «Университет».
- Поскольку в C++ ссылка всегда ссылается на один и тот же объект, то хранение студентом ссылки на университет не позволит перевести студента в другой университет. Поэтому единственно возможным вариантом хранения информации об университете студентом является использование указателя на константные данные.
- Студент сразу после своего создания обучается в некотором университете. Следовательно, константная ссылка на университет должна передаваться конструктору студента. Как, впрочем, и номер года обучения.
- Поскольку студент хранит указатель на объект «Университет», необходимо следить за тем, чтобы объект «Университет» не разрушился раньше, чем объект «Студент». В противном случае, студент будет ссылаться на уже несуществующий университет и попытка обращения к университету через запрос к студенту вызовет неопределенное поведение.

- Удалить произвольный университет. При этом происходит и удаление всех студентов, обучающихся в нем.
- Вывести список студентов произвольного университета.
- Добавить новый университет с уникальным названием.
- Вывести список студентов. Информация о студенте включает его имя, пол, возраст, рост, вес, а также название университета и номер года обучения.
- Изменить сведения о студенте (кроме пола). Возраст, рост и номер года обучения могут быть изменены только в большую сторону или оставлены без изменения. Должна иметься возможность перевода студента в другой из имеющихся университетов.
- Удалить произвольного студента
- Добавить произвольного студента

При выходе из программы, программа должна предлагать сохранить изменения в оригинальных файлах.

Бонус в 120 баллов за создание классов «Рабочий» и «Компания»

Спроектируйте класс «Компания», характеризующаяся названием и адресом Интернет сайта (может быть пустой строкой). Должны иметься методы, позволяющие изменить название компании и адрес ее сайта. Решите следующие архитектурные вопросы насчет создания данного класса:

1. Должен ли класс «Компания» быть спроектирован полностью независимым от класса «Университет», фактически, дублируя код доступа к названию?
2. Должен ли класс «Компания» быть унаследован от класса «Университет»? Если да, то какой тип наследования является наиболее предпочтительным: открытое, закрытое или защищенное?
3. Следует ли при создании класса «Компания» вместо наследования отдать предпочтение композиции?

Сделайте свой выбор, реализовав класс «Компания» соответствующим образом.

Разработайте класс **CWorker** (рабочий), расширяющий возможности обычного человека принадлежностью к некоторой компании, названием занимаемой должности, а также возможностью перехода из одной компании в другую, а также смены должности.

При выполнении данного бонусного функционала требуется также расширить возможности приложения, которое должно загружать информацию о компаниях и рабочих и предоставлять следующие действия по работе с ними:

- Вывод списка имеющихся компаний. Информация о компании включает ее название и адрес Интернет сайта.
- Вывод списка имеющихся рабочих. Информация о рабочем включает сведения о нем, как об обычном человеке, а также название компании и занимаемой должности
- Вывод списка рабочих произвольной компании.
- Редактирование сведений о компании (название и адрес веб сайта). Совпадение с уже имеющимися названиями компаний не допускается.
- Удаление компании. При этом происходит удаление всех служащих данной компании из списка
- Добавление новой компании
- Редактирование сведений об имеющемся рабочем. Ограничения при редактировании персональных данных подобны ограничениям при редактировании таких данных студента. Должна иметься возможность перевода рабочего на другое место работы и/или должность
- Удаление рабочего
- Добавление рабочего

При выходе из программы пользователю должно предлагаться сохранить сведения об имеющихся студентах, рабочих, компаниях и университетах в оригинальных файлах.