Задание 1 для первого семестра 2 курса

Требуется разработать и реализовать несколько простейших классов на C++ и написать тесты для проверки работы этих классов.

Требования к реализации

0. Реализация должны состоять из нескольктх файлов. Нужно уметь использовать собственные заголовочные файлы, составлять простейшие make-файлы и использовать их для сборки, а также уметь собирать и отлаживать программу из командной строки (в том числе с использованием gdb).

1. Все реализации должны удовлетворять согласованным неформальным интерфейсам и иметь заголовочный .h файл с прокомментированным описанием методов класса. Разработка формального описания классов предоставляется студенту, при этом должны быть предусмотрены способы реакции и оповещения о некорректных или ошибочных ситуациях.

2. К заданию должны быть написаны тесты, т.е. отдельные программы которые вызывают требуемые функции с разнообразными наборами данных и (если возможно, то автоматически) проверяют правильность результата.

3. Поскольку детальная реализация предлагаемых задач может оказаться весьма объемной, то не нужно пытаться реализовать все возможные операции. Но при этом реализованная часть все же должна поддерживать некоторые вполне содержательные задачи (тесты) и должно быть ясно как при необходимости можно расширить реализацию до необходимого уровня доопределением дополнительных функций и операций.

Возможные варианты задания Данные формулировки задают только направление работы. Конкретизация постановок, уточнение и т.д. проводятся в рабочем порядке с преподавателем. Каждый должен выполнить некую реализацию в рамках задачи **All.** и одно из заданий **A**–**H**. Задачи стоит делать одновременно, чтобы по мере знакомства с конструкциями C++ постепенно расширять свои "проекты".

All. Евклидова геометрия в $\mathbf{R^2}$. Векторы, точки, отрезки, линии, фигуры. Построения "циркулем и линейкой".

Задание подразумевает создание системы "геометрических" классов, в которых координаты объектов спрятаны в private секциях, а на уровне интерфейса предоставляются естественные геометрические соотношения и операции по типу геометрии Евклида.

Построения "циркулем и линейкой". Сравнения объектов, пересечения, принадлежность. Преобразования (подобие, гомотетия, повороты, симметрии, инверсии).

Предполагается, что каждый студент выберет себе некоторое количество "школьных" задач на построение или проверку теорем, реализует необходимые классы геометрических объектов и напишет тесты для решения поставленных задач. Тест подразумевает функцию, которая реализует построение "циркулем и линейкой" для решения некоторой школьной задачи на построение и использует только методы из данной системы геометрических классов (не использует координаты объектов, длины и т.п.).

Кроме этого каждый реализованный геометрический класс дожен иметь метод, позволяющий его отрисовать, например, в смысле использования утилиты gnuplot. В этом случае, можно будет визуализировать последовательность и результат построений требуемых геометрических объектов.

Например, пусть поставлена задача построения треугольника по трем сторонам. Т.е. тестовая функция три объекта типа "отрезок", а на выходе должна дать объект типа "треугольник". При построении "циркулем и линейкой" у нас нет возможности узнать длину отрезков, мы можем только откладывать отрезок, равный данному (кстати, эта операция тоже нетривиально выполняется циркулем и линейкой). Поэтому можно предусмотреть конструктор объекта "окружность" с радиусом, равным данному отрезку. После этого мы создаем две окружности с радиусами равными двум данным отрезкам и центрами на концах третьего отрезка, получаем точки пересечения этих окружностей как применение отдельной операции пересечения окружностей, и создаем объект "треугольник" из найденых трех точек. Еще раз обращаю внимани, что циркуль и линейка не предполагают использование числовых значений длины, только манипуляции с отрезками, прямыми, точками, окружностями и т.д.

В качестве задач можно взять:

— построение треугольника по трем элементам (не только простейшим, но и более сложным, скажем, по двум углам и отрезку с длиной равной периметру и т.д.);

— построение касательных с разным фигурам;

— построение преобразованных фигур (растяжения, повороты и пр.)

— “проверка” разных утверждений (например, вписанный угол, опирающийся на диаметр прямой, реализация сравнения фигур на равенство как это понимается у Евклида, т.е. все сводится к проверке совпадению точек или принадлежности точек прямым или отрезкам);

— любые другие аналогичные задачи.

Не возбраняется коллективное использование геометрических классов, но не программ решающих конкретные задачи на построение. Эти программы каждый должен выбрать для себя сам и уметь также потом выполнять аналогичные построения сразу на семинаре.

Пример: Проверить, что вписанный угол, опирающийся на диаметр, является прямым. Решение: Строим произвольную окружность. Проводим прямую через ее центр. Находим отрезок пересечения этой прямой с окружностью — это диаметр. Проводим прямую через одну из крайних точек диаметра — получаем хорду. Строим отрезок другой хорды. Получаем вписанный угол. Строим прямой угол известным школьным построением. Сравниваем эти два угла (не используя понятие градусной меры). Здесь дополнительно возникает необходимость сравнивать геометрические объекты на “равенство”. В евклидовой геометрии под равенством фигур понимается их совпадение при наложении. Соответственно в нашем случае мы можем определить “равенство точек” и “равенство отрезков” (как совпадение их координат или длин в рамках заданной точности, при этом понятие длины присутствует только внутри нашей процедуры сравнения, на внешнем уровне длина как число не появляется), и далее уже определять равенства других фигур, используя сведение к равенству точек или отрезков.

Арифметика. Задача — построить класс, представляющий собой специальное число, и потом проверить некоторые вычислительные алгориты с такими числами. Для начала — простейшие арифметические операции, потом что-то более сложное — суммирование рядов, вычисление интегралов, решение уравнений и т.п. Функция печати этого числа в соответствующей наглядной форме.

Задачи этого типа могут оказаться разной сложности. В случае сложных задач надо вовремя остановиться. То есть, построить реализацию, которая может решать некоторые содержательные частные задачи, но не обязательно обладает полной функциональностью. В списке ниже более сложные задачи отмечены звездочкой.

A. Комплексная арифметика. Обычное комплексное число — действительная и мнимая части. Реализация операций, элементарных функций. Тест, например, решение квадратного уравнения, вычисление элементарных функций через ряды и т.п.

B. Рациональная арифметика. Число представляется как целочисленная пара — числитель и знаменатель. Арифметические операции, сокращение общего множителя, контроль переполнения знаменателя. Взаимодействие с вещественными числами (приближение).

C. Длинная целая арифметика. Реализация вычислений с целыми числами произвольной величины. Тут можно предложить несколько способов, основанных на общей идее динамического массива, т.е. массива, которые может менять свою длину при необходимости.

C0. Число есть char массив (элемент - одна десятичная цифра).

C1. Число есть битовый массив.

C2. Число есть полубайтовый массив, где каждый полубайт представляет десятичную цифру.

C3. Число есть массив int, где каждый элемент отвечает за несколько десятичных знаков числа (например, 9, т.е. содержит число от 0 до 999999999).

В перспективе интересно объединение с задачей **B.** для устранения переполнения знаменателя.

D1. Вещественная арифметика с фиксированной точкой. Число — это пара целых чисел (целая и дробная части). Арифметические операции и элементарные функции. Тесты — любые вычислительные процедуры и сравнение с результатами в double или float.

D2. Вещественная арифметика с фиксированной точкой. Число — это целое число, в котором часть разрядов отведена под целую часть, а часть под дробную.

Вариант: в разных числах это разделение может быть разным (т.е. это тоже переменная класса), в этом случае при операциях необходимо делать выравнивание.

Вариант: это разделение определяется в конструкторе и числа с разным разделением не могут участвовать в одной операции.

Арифметические операции и элементарные функции. Тесты — любые вычислительные процедуры и сравнение с результатами в double или float.

E. Интервальная арифметика. Число — это интервал, которому может принадлежать рабочее значение (min, max). Арифметические операции и элементарные функции. Соответственно результат любой операции — это тоже интервал для максимального и минимального возможных значений. Тесты — любые вычислительные процедуры и сравнение с результатами в double или float.

F. Арифметика с оценкой абсолютной погрешности. Число есть значение и модуль его абсолютной погрешности. Арифметические операции и элементарные функции. Тесты — любые вычислительные процедуры и сравнение с результатами в double или float. Это тот самый Number!

G. Арифметика с оценкой относительной погрешности. Число есть значение и его относительная погрешность. Арифметические операции и элементарные функции. Тесты — любые вычислительные процедуры и сравнение с результатами в double или float.

***H.** Арифметика с производной. Число есть пара чисел, которые интерпретируются как значение некоторой функции и ее производной. Соответственно результат операций для производной вычисляется по формулам производной суммы, разности, произведения, частного. Выполняя некоторый алгоритм с такими числами (например, суммируя ряд для синуса), мы должны одновременно получить и значение производной (т.е. значение косинуса). Есть некоторые тонкости при взаимодействии с обычными числами.

I. Алгебра многочленов. Сложение, вычистание, умножение, деление (с остатком). Интересно соединить с комплексными числами. Тест - нахождение корней. Действительные — численным поиском, комплексные — делением с остатком и выледением квадратичных сомножителей.

Внимание!

Некоторые задачи весьма сложны для полной реализации. Поэтому не старайтесь из сразу решать в полном объеме, а уделите существенное внимание именно разработке структуры ваших классов. Иначе может оказаться, что неверные решения, принятые в начале проектирования, не дадут вам возможность реализовать какие-то требования.

При реализации нужно использовать перегрузку операторов, разнообразные конструкторы, в частности, операторы присваивания и сравнения и конструкторы копирования.

Следует предусмотреть возможность проверки и обработки ситуаций, когда операция с объектами не может быть выполнена или не дает корректного результата (например, деление на 0, пересечение параллельных линий и т.п.)