**Методические указания к выполнению практических заданий по курсу «Объектно-ориентированное программирование»**

Цикл состоит из 8 практических заданий, охватывающий разделы курса «Объектно-ориентированное программирование».

Для каждой работы приводится требования, предъявляемые к каждому проекту, который требуется реализовать. Формулируются цель работы и ограничения, в рамках которых эта цель должна быть реализована.

В качестве исходных данных для программирования в объектно-ориентированное парадигме (ООП) выступает объектная модель (ОМ) предметной области (ПрО), дополненная объектной библиотекой (ООБ) конструкторских классов (КК). ОМ в данном практическом цикле представлен в виде диаграммы классов (ДК), которую следует реализовать или разработать на основе описания задачи, заполнив классы ПрО необходимыми атрибутами (полями) и методами (операциями), обоснованно распределенными по областям видимости. В качестве КК на ДК используются стандартные классы библиотеки С++ и классы из ООБ Qt. На ДК используются следующие обозначения:



Диаграммы классов, представленные в заданиях задают **концепцию** построения приложений. На них могут быть не прорисованы некоторые конструкторские классы из ООБ. Здесь можно проявить творческий подход и использовать классы из соответствующих объектных библиотек. В отчетах по практическим работам следует представлять **полные диаграммы классов** со всеми используемыми классами иуказанием отношений между ними.

**Некоторые сведения о диаграммах классов**

Диаграмма классов занимает центральное место в логической модели ПО и используется для представления статической структуры модели системы в терминологии классов объектно-ориентированного программирования. Эта диаграмма может рассматриваться и строиться, как дальнейшее развитие концептуальной модели проектируемой системы.

Диаграмма классов представляет собой некоторый граф, вершинами которого являются элементы представляющие классы, дуги задают различные типы структурных отношений, которыми они связаны.

Класс (class) обозначает множества объектов, которые обладают одинаковой структурой, поведением и отношениями с объектами из других классов. Графически класс изображается в виде прямоугольника, который дополнительно может быть разделен горизонтальными линиями на разделы или секции. В этих разделах могут указываться имя класса, атрибуты (переменные) и операции (методы).

Обязательным элементов обозначения класса является его имя. На начальных этапах разработки диаграммы отдельные классы могут обозначаться простым прямоугольником с указанием только имени соответствующего класса. По мере проработки отдельных компонентов диаграммы, описания классов дополняются атрибутами и операциями.

Каждому атрибуту класса приписывается один из кванторов видимости. Квантор видимости может принимать одно из трех возможных значений и, соответственно, отображается при помощи специальных символов:

1. Символ "+" обозначает атрибут с областью видимости типа общедоступный *public).* Атрибут с этой областью видимости доступен или виден из любого другого класса пакета, в котором определена диаграмма.
2. Символ "#" обозначает атрибут с областью видимости типа защищенный *(protected).* Атрибут с этой областью видимости недоступен или невиден для всех классов, за исключением подклассов данного класса.
3. Символ "-" обозначает атрибут с областью видимости типа закрытый (*private).* Атрибут с этой областью видимости недоступен или невиден для всех классов без исключения.

Вместо условных графических обозначений можно записывать соответствующее ключевое слово: *public, protected, private.*

Кроме внутреннего устройства или структуры классов на диаграмме указываются различные отношения между классами. Базовыми отношениями между классами или связями являются:

1. отношение зависимости (dependency relationship),
2. отношение ассоциации (association relationship),
3. отношения реализации (realization relationship):

- отношение агрегации (aggregation relationship),

- отношение композиции (composition relationship),

1. отношение обобщения (inherits from relationship).

Каждое из этих отношений имеет собственное графическое представление на диаграмме, которое отражает взаимосвязи между объектами соответствующих классов.

**Отношение зависимости** (dependency relationship).

Отношение зависимости является наиболее общей формой отношения. Оно имеет место всегда, когда изменение спецификации одного класса (класса-источника) может повлиять на работу другого класса (класса-клиента), но не наоборот. Отношение зависимости графически изображается пунктирной линией со стрелкой, направленной от класса-клиента зависимости к независимому классу или классу-источнику.

**Отношение ассоциации** (association relationship).

Отношение ассоциации имеет место, когда объекты одного класса связаны с объектами другого класса таким образом, что можно перемещаться от объектов одного класса к другому. Отношение ассоциации обозначается сплошной линией, стрелка указывает на порядок следования классов.

**Отношение агрегации** (aggregation relationship).

Является частным случаем отношения ассоциации. Отношение агрегации имеет место между несколькими классами в том случае, если один из классов представляет собой некоторую сущность, включающую в себя в качестве составных частей другие сущности. Это отношение по своей сути описывает декомпозицию или разбиение сложной системы на более простые составные части, которые также могут быть подвергнуты декомпозиции, если в этом возникнет необходимость в последующем. Графически отношение агрегации изображается сплошной линией, один из концов которой представляет собой не закрашенный внутри ромб, который указывает на класс, представляющий собой "целое". Остальные классы являются его "частями".

**Отношение композиции** (composition relationship).

Является частным случаем отношения агрегации. Это отношение служит для выделения специальной формы отношения "часть-целое", при которой составляющие части в некотором смысле находятся внутри целого, и эти части существуют до тех пор, пока существует целое и прекращают свое существование вместе с целым. Графически отношение композиции изображается сплошной линией, один из концов которой представляет собой закрашенный внутри ромб, который указывает на класс, представляющий собой "целое". Остальные классы являются его "частями".

**Отношение обобщения** (inherits from relationship).

Является обычным таксономическим отношением между более общим элементом (родителем или предком) и более частным или специальным элементом (дочерним или потомком). Отношение описывает иерархическое строение классов и на следование их свойств и поведения. При этом предполагается, что класс-потомок обладает всеми свойствами и поведением класса-предка, а также имеет свои собственные свойства и поведение, которые отсутствуют у класса-предка. На диаграммах отношение обобщения обозначается сплошной линией с треугольной стрелкой на одном из концов. Стрелка указывает на более общий класс (класс-предок или суперкласс), а ее отсутствие – на более специальный класс (класс-потомок или подкласс).

**Требования к реализации практических работ**

Для выполнения практических заданий нет ограничений для выбора языка, среды и объектной библиотеки. Однако

1. проект должен быть реализован в точном соответствии с концепцией приведенных ДК, если таковые имеются (сохранение всех отношений указанных типов между классами). Допускаются расширение состава или замена используемых КК с сохранением их функционального назначения;

2. не допускается смена языка программирования от работы к работе;

3. для языков, в которых не отделяются описание и реализация в дополнении к проекту должны быть приведены С++ образные заголовочные файлы, содержащие протоколы классов проекта (протокол класса должен быть максимально возможно обозрим).

**Отклонение от этих требований за собой снижение оценки по работе.**

**ВАЖНО**

Основными классами, разрабатываемым в цикле из 8 работ являются МАССИВ, КОМПЛЕКСНОЕ ЧИСЛО, ПОЛИНОМ, ФУНКЦИЯ. В каждой последующей работе за основу берется уже класс из предыдущей без каких-либо косметических изменений. Для этих классов требуется:

1. использовать от работы к работе последнюю версию без изменений или при необходимости дополненную;

2. изменения допустимы для исправления указанных при оценке работы замечаний;

3. любые изменения или дополнения должны быть отображены и обоснованы в отчете.

При первом использовании указанных классов в отчете формируется таблица:

Таблица 1. Первичный протокол класса НАЗВАНИЕ

|  |  |  |  |
| --- | --- | --- | --- |
| Атрибуты | | | |
| идентификатор | тип | область видимости | семантическое описание |
|  |  |  |  |
| Методы | | | |
| идентификатор | область видимости | | семантическое описание |
|  |  | |  |

При последующих использованиях формируется таблица другого вида:

Таблица 2. Первичный протокол класса НАЗВАНИЕ

|  |  |  |  |
| --- | --- | --- | --- |
| Атрибуты (старые) | | | |
| идентификатор | тип | область видимости | семантическое описание |
|  |  |  |  |
| Атрибуты (удаленные) | | | |
|  |  |  |  |
| Атрибуты (новые) | | | |
| идентификатор | тип | область видимости | семантическое описание |
|  |  |  |  |
| Методы (старые) | | | |
| идентификатор | область видимости | | семантическое описание |
|  |  | |  |
| Методы (удаленные) | | | |
|  |  | |  |
| Методы (новые) | | | |
| идентификатор | область видимости | | семантическое описание |
|  |  | |  |

Рекомендуемая среда разработки QtCreator и использование библиотеки классов Qt (язык программирования С++). На приведенных диаграммах классов в качестве конструкторских используются классы из этой библиотеки.

**Требования к оформлению отчета**

Отчет по практической работе выполняется 1 на бригаду и состоит из пояснительной записки в бумажном или электронном виде, к которому прилагается архив исходных файлов проекта.

Отчет должен содержать:

1. Титульный лист.

2. Задание на практическую работу.

3. Спецификации классов (понятий или предметов), которые требуется разработать - таблицы 1 и 2 для всех основных классов.

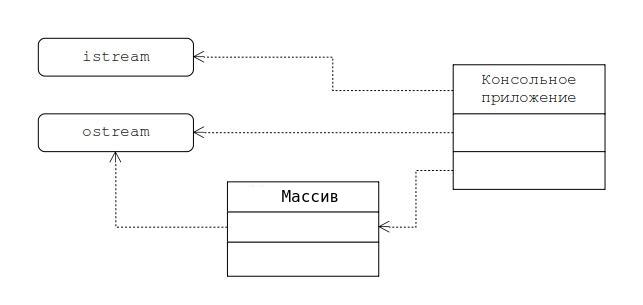
4. Диаграмму классов, дополненную атрибутами и методами.

5. Описание контрольного примера с исходными и ожидаемыми (расчетными) данными.

6. Скриншоты работы программы на контрольных примерах.

7. Выводы по выполнению работы.

**Практическая работа №1**

**Рис.1. Диаграмма классов работы №1**

Разработать класс "Массив" произвольной длины, способный хранить объекты абстрактного типа **number**. Класс должен уметь:

- создавать 0 массив (без элементов),

- создавать массив с N элементами и заполнять их заданными значениями,

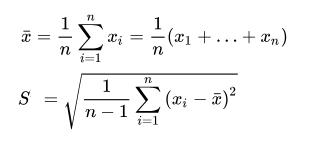
- корректное уничтожение объектов класса,

- изменять размерность массива,

- вводить элементы массива из стандартного потока ввода (объект cin класса istream),

- выводить элементы массива в стандартный поток вывода (объект cout класса ostream),

- вычислять среднее и среднеквадратичное отклонение (СКО) по формулам



- сортировать элементы массива по возрастанию и убыванию.

Создать консольное приложение согласно представленной на рис.1 диаграмме классов, предназначенное для работы с массивом **вещественных чисел**. Приложение должно включать основной модуль (функция main), модуль «application» и модуль «array».

В **основном модуле** консольного приложения (*для языка С++ - это модуль с функцией main*) должен создаваться объект класса "Консольное приложение" и вызываться его метод, который предоставляет пользователю **меню команд** приложения.

Модуль **«application»** должен содержать спецификацию класса "Консольное приложение" и реализацию его методов. Один из его методов должен выводить в консоль меню команд приложения, включающее:

- команду, инициирующую ввод с консоли значений, задающих объект массива (до ввода в программе должен быть задан массив по умолчанию);

- команду, инициирующую расчеты среднего и СКО элентов массива и вывод результатов расчета;

- команду, инициирующую сортировку элементов массива по возрастанию или убыванию;

- команду, инициирующую изменения размерности массива;

- команду, инициирующую изменения значения выбранного элемента массива;

- команду, инициирующую вывод в консоль элементов текущего объекта массива;

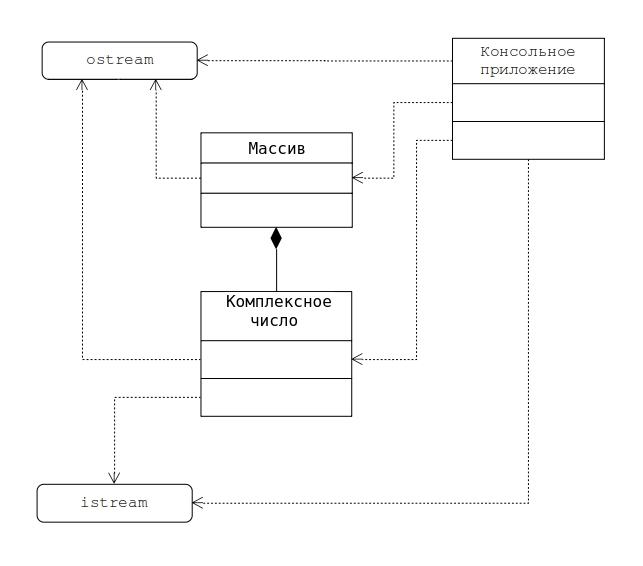
- команду выхода из приложения.

Модуль **«array»** должен содержать спецификацию класса "Массив" и реализацию его методов, необходимых для достижения цели разрабатываемого приложения. Описание класса должно использовать вместо типа double (вещественное число, заданное в условии) абстрактный тип ***number***, описание которого должно задаваться в отдельном заголовочном файле number.h с помощью оператора **typedef double number** (для С++).

**ВАЖНО.** Основное требование к реализации класса **«array»** заключается в том, что она должна быть инвариантна (одна и та же для различных вариантов использования, т.е. конкретизированных в каждом случае использования типов **number**) ко множеству применимых объектов. Здесь задано множество определения элементов матрицы как вещественное, а может быть и множества целых, комплексных, рациональных и т.д. Для различных множеств при сохранении функциональности реализация класса должна быть одна и та же. Если для различных множеств менять реализацию, то теряется смысл ООП как парадигмы.

Требуется реализовать и отладить программу, удовлетворяющую сформулированным требованиям и заявленным целям. Разработать контрольные примеры и оттестировать на них программу. Оформить отчет, сделать выводы по работе.

**Практическая работа №2**

**Рис.2. Диаграмма классов работы №2**

Создать консольное приложение, реализующее функции перечисленные в описании работы №1, но на множестве комплексных чисел.

Приложение должно включать основной модуль, модуль «application», модуль «array» и модуль «complex».

Для этого в проект лабораторной работы №1 следует добавить модуль с **собственным** (не брать из ООБ!) описанием и реализацией класса комплексных чисел (например, TComplex). Класс TComplex должен быть встроен в проект согласно диаграмме классов на рис.2. **При этом основной модуль, модуль «application» и модуль «array» не должны изменяться**. Изменения вносятся лишь в заголовочный файл number.h, где

**typedef double number;**

следует заменить на

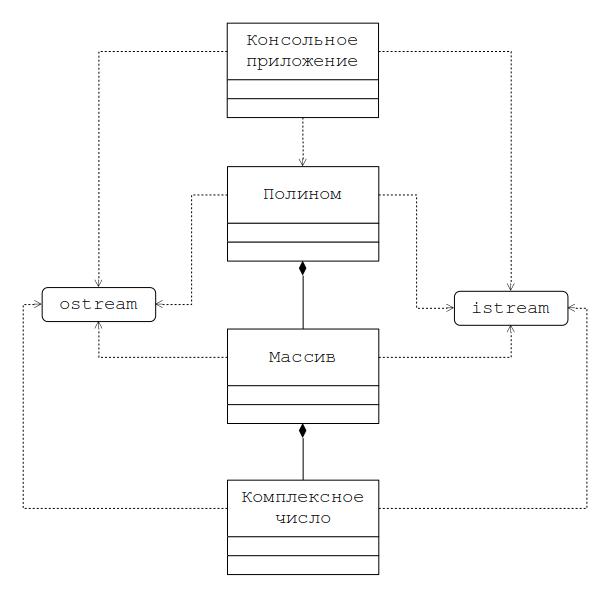
**#include «complex.h»**

**typedef TComplex number;**

В классе **TComplex** следует определить только те члены класса и спецификации, которые необходимы для совместимости модулей проекта и реализации отношений, приведенных в ДК объектной модели.

Реализовать и отладить программу, удовлетворяющую сформулированным требованиям и заявленной цели. Разработать контрольные примеры и оттестировать на них программу. Оформить отчет, сделать выводы по работе.

**Практическая работа №3**

**Рис.3. Диаграмма классов работы №3**

Разработать класс "Полином" произвольной степени.

*p(x) = anxn + an-1xn-1 + ... + a1x + a0, (1)*

где *an ,an-1 ,a1 ,a0 , x* - объекты абстрактного типа **number**.

Класс должен уметь:

- создавать полином 0-ой степени (0 полином),

- создавать полином N-ой степени, задаваемый значением коэффициента *an* и N значениями своих корней,

*p(x) = an(x-rn)(x-rn-1) ... (x-r1) (2)*

- хранить корни в массиве,

- вычислять коэффициенты при степенях х и хранить их в массиве, обеспечивая к ним доступ только на чтение,

- вводить коэффициент *an* и корни полинома из стандартного потока ввода (объект cin класса istream),

- выводить элементы массива в стандартный поток вывода (объект cout класса ostream) в форме 1 и 2 по выбору,

- вычислять значение полинома в заданной точке х.

Создать консольное приложение согласно представленной на рис.3 диаграмме классов, предназначенное для работы с полиномом на множестве **комплексных чисел**. Приложение должно включать основной модуль (функция main), модуль «application» и модули «polinom», «array», «complex».

В **основном модуле** консольного приложения (*для языка С++ - это модуль с функцией main*) должен создаваться объект класса "Консольное приложение" и вызываться его метод, который предоставляет пользователю **меню команд** приложения.

Модуль **«application»** должен содержать спецификацию класса "Консольное приложение" и реализацию его методов. Один из его методов должен выводить в консоль меню команд приложения, включающее:

- команду, инициирующую ввод с консоли значений, задающих объект полином (до ввода в программе должен быть задан полином по умолчанию);

- команду, инициирующую изменение коэффициента *an* или одного из выбранных по индексу корней;

- команду, инициирующую вычисления значения полинома в заданной точке;

- команду, инициирующую изменения размерности массива;

массива;

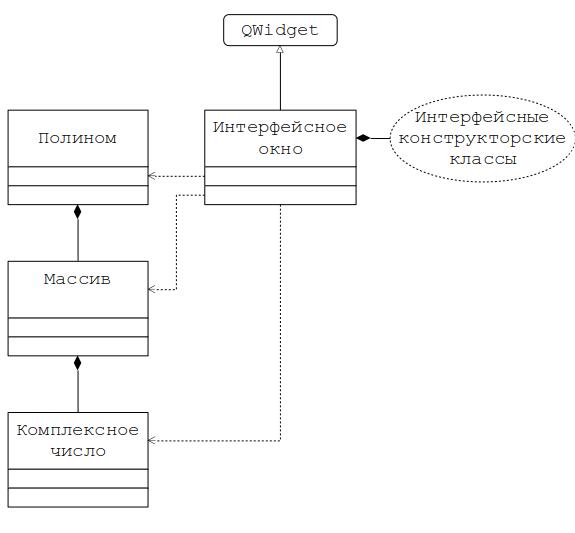
- команду, инициирующую вывод в консоль текущего объекта полинома в форме 1 или 2;

- команду выхода из приложения.

Модуль **«polynom»** должен содержать спецификацию класса "Полином" и реализацию его методов, необходимых для достижения цели разрабатываемого приложения.

Реализовать и отладить программу, удовлетворяющую сформулированным требованиям и заявленной цели. Разработать контрольные примеры и оттестировать на них программу. Оформить отчет, сделать выводы по работе.

**Практическая работа №4**

**Рис.4. Диаграмма классов работы №4**

Создать GUI приложение, реализующее функции перечисленные в описании работы №3, но на множестве рациональных чисел.

Приложение должно включать основной модуль, модуль «interface», модуль «polynom», модуль «array», модуль «complex» и файл number.h.

При необходимости расширения функциональности реализованных ранее классов следует только **дополнять** их протоколы **без каких-либо изменений** уже существовавшей реализации.

Реализовать и отладить программу, удовлетворяющую сформулированным требованиям и заявленным целям. Разработать контрольные примеры и оттестировать на них программу. Оформить отчет, сделать выводы по работе.

**Практическая работа №5**

|  |
| --- |
| Клиентская часть |
|  |
| Серверная часть |
|  |

**Рис.5. Диаграмма классов работы №5**

Создать распределенное приложение, включающее клиентскую и серверную части, взаимодействующие посредством сетевого обмена сообщениями.

Клиентская часть представляет собой GUI приложение, реализующее интерфейс аналогичный работе №4.

Серверная часть представляет собой консольное приложение, предназначенное для выполнения перечисленных в меню работы №3 функций над полиномом с комплексными элементами.

Диаграммы классов для клиентского и серверного приложений представлены на рис.5.

Реализовать и отладить программу, удовлетворяющую сформулированным требованиям и заявленным целям. Разработать контрольные примеры и оттестировать на них программу. Оформить отчет, сделать выводы по работе.

**\*)** Класс "Коммуникатор" на основе сетевого протокола Udp с использованием конструкторских классов библиотеки классов Qt и пример его использования прилагаются к работе в виде исходных кодов.

**Практическая работа №6**

Создать распределенное приложение, включающее клиентскую и серверную части, взаимодействующие посредством сетевого обмена сообщениями.

Клиентская часть должна отличаться от аналогичной в работе №5 тем, что позволяет разделить работу с полиномами на множествах вещественных и комплексных чисел.

Серверная часть представляет собой приложение, реализованное в работе №5, но в нем классы «Полином» и «Массив» параметризуются. Параметром этих классов-шаблонов будет используемый ранее абстрактный тип **number**.

Реализовать и отладить программу, удовлетворяющую сформулированным требованиям и заявленным целям. Разработать контрольные примеры и оттестировать на них программу. Оформить отчет, сделать выводы по работе.

**Практическая работа №7**

Сначала немного теории. Бесконечно дифференцируемую в точке x0 функцию действительной переменной *f(x)* можно разложить в ряд по степеням двучлена (x−x0):

*f(x)=f(x0)+f′(x0)(x−x0)/1!+f''(x0)(x−x0)2/2!+ ...*

Этот ряд называют рядом Тейлора. В случае *x0=0* полученный степенной ряд:

*f(x)=f(0)+f′(0)x/1!+f''(0)x2/2!+ ...*

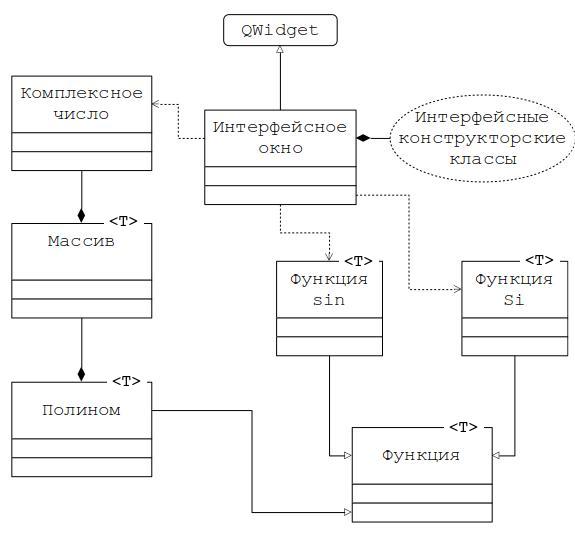
называют рядом Маклорена.

В правой части выражения мы видим полином бесконечной степени. Если степень полинома ограничить, то будет существовать погрешность вычисления функции, т.е.

f(x) ≈ pn(x)

С точки зрения ООП, если функция и полином - это классы, то последнее выражение означает, что между ними существует отношение обобщения. Полином обобщает функцию.

Аналогично понятие функции вообще обобщает понятия конкретных функций синус (*sin)* и интегральный синус (*Si=sin(x)/x*).



**Рис.7. Диаграмма классов работы №7**

Разработать GUI приложение, выполняющее вычисления функций синус и интегральный синус (пользователь выбирает) с переменной точностью (выбор пользователем степени полинома разложения в ряд Макларена) в задаваемой точке комплексной плоскости.

Для реализации шаблонов классов функций использовать шаблон класса «Полином» работы №6 без изменения состава атрибутов и методов, используя при необходимости только изменение области видимости некоторых членов класса.

Реализовать и отладить программу, удовлетворяющую сформулированным требованиям и заявленным целям. Разработать контрольные примеры и оттестировать на них программу. Оформить отчет, сделать выводы по работе.

**Практическая работа №8**

Разработать GUI приложение на основе работы №7 для построения графиков функций синус и интегральный синус на множестве вещественных чисел.

Интерфейс программы дополнить выбором диапазова изменения аргумента функций.

Реализовать и отладить программу, удовлетворяющую сформулированным требованиям и заявленным целям. Разработать контрольные примеры и оттестировать на них программу. Оформить отчет, сделать выводы по работе.