Федеральное государственное бюджетное
образовательное учреждение высшего образования
"Кемеровский государственный университет"
Новокузнецкий институт (филиал)

Факультет информатики, математики и экономики
Кафедра математики, физики и математического моделирования

А.Д. Ульянов

ПРОГРАММИРОВАНИЕ НА ЯЗЫКАХ ВЫСОКОГО УРОВНЯ

Методические рекомендации по выполнению внеаудиторной самостоятельной работы для обучающихся по направлениям подготовки
02.03.03 Математическое обеспечение и администрирование информационных систем, профиль «Программное и математическое обеспечение информационных технологий»

Часть 1

Новокузнецк

2020

Ульянов А.Д.

Программирование на языках высокого уровня: методические рекомендации по выполнению внеаудиторной самостоятельной работы для студентов факультета информатики, математики и экономики, обучающихся по направлению подготовки 02.03.03 Математическое обеспечение и администрирование информационных систем (уровень бакалавриата): в 2 ч. Ч 1. / А.Д. Ульянов; Новокузнецкий ин-т (фил.) Кемеров. гос. ун-та. – Новокузнецк : НФИ КемГУ, 2020 – 12 с.

В настоящих методических указаниях для студентов представлены материалы по выполнению внеаудиторной самостоятельной работы по дисциплине «Программирование на языках высокого уровня»: основные теоретические сведения в форме конспектов лекций с примерами решения типовых задач по темам «Жизненный цикл программного продукта», «Стандартная библиотека STL», «Основы объектно-ориентированного программирования (ООП). Классы и объекты», «Перегрузка операций», «Наследование и полиморфизм», «Обобщенное программирование», «Обработка исключений», «Паттерны проектирования». Также представлены банк задач для контрольных работ, методические рекомендации по решению и оформлению, оценивание работ в бально-рейтинговой системе и список основной и дополнительной литературы.

Методические рекомендации предназначены для наиболее рациональной организации внеаудиторной самостоятельной работы студентов при подготовке к выполнению контрольных работ и теста.

Рекомендовано
на заседании кафедры
математики, физики и математического моделирования
22 октября 2020г.
Заведующий кафедрой

 / Е.В. Решетникова

Текст представлен в авторской редакции

ОГЛАВЛЕНИЕ

ПРЕДИСЛОВИЕ

Настоящие методические рекомендации адресованы студентам, получающим квалификацию бакалавр по направлению подготовки 02.03.03 Математическое обеспечение и администрирование информационных систем, профиль «Программное и математическое обеспечение информационных технологий» и направлены на оказание помощи студентам в подготовке к выполнению контрольных работ по темам «Основные принципы объектно-ориентированного программирования (ООП). Классы и объекты», «Перегрузка операций» и «Наследование и полиморфизм» дисциплины «Программирование на языках высокого уровня».

Задача программирования ЭВМ существует со дня создания первого компьютера. С тех самых пор сообщество программистов опытным и аналитическим путем пытается создать наиболее эффективный способ написания программ. Задача меняет свои начальные условия с каждым новым витком развития ЭВМ. Это происходит благодаря возрастающей производительности компьютеров, что предоставляет программистам новые возможности. При этом старые эффективные решения теряют актуальность.

В последние десятилетия, с появлением последних поколений ЭВМ эффективность выполнения программ постепенно уступает приоритет понятности и читаемости программного кода. Высокий уровень производительности современных компьютеров позволяет разрабатывать и запускать достаточно ёмкие и многофункциональные программные продукты, выводя ранок предоставляемых ИТ услуг на принципиально новый уровень. Вместе с тем возрастает и объем программного кода, усложняется структура взаимосвязей внутренних сущностей программ, увеличивается число разработчиков и управленцев.

Таким образом, при проектировании и разработке ПО представляется актуальной проблема декомпозиции облака данных программной модели в отдельные объекты, которые объединяют данные и их функциональность в единую сущность (структуры данных). В крупных программных решениях число типов таких сущностей может измеряться сотнями и тысячами, что труднообозримо для отдельно взятого программиста. Поэтому необходимо не только группировать данные, но и скрывать часть зависимых объектов и функциональности от разработчиков разных уровней. Т.е. разработчики объектов низкого уровня, отладив и описав открытые интерфейсы своих объектов, должны предоставлять разработчикам прикладного уровня только те функции и данные объектов, которые могут быть им полезны (ООП, инкапсуляция).

Одновременно с тем, различные объекты системы могут иметь схожее поведение и представление, а различаться между собой частично. С целью сокращения объема дублирующегося кода, такие объекты по функциональности и включаемым данным следует выстроить в иерархию, и организовать копирование похожих свойств согласно введенной иерархии (ООП, наследование). В случае, если на прикладном уровне программы не требуется учитывать специфику каждого конкретного объекта, а лишь обобщенно использовать методы, общие для всех "родственных" объектов, то такие объекты должны быть приведены к общему для всех интерфейсу (ООП, полиморфизм). Это явно демонстрирует общие черты объектов программисту прикладного уровня.

Перечисленные требования к программному обеспечению особенно актуальны для больших коллективов программистов. Тем более в условиях современного рынка труда, где программисты могут менять рабочее место с достаточной быстротой, особенно необходимо иметь возможность внедрять новых разработчиков в процесс программирования и поддержки. Особенно этому способствует возрастающая "читаемость" и понятность программного кода, соответствующего предъявленным требованиям. Для руководства отсутствие зависимости календарного планирования от ротации состава исполнителей - неоценимая помощь.

Безусловно, накопление промежуточных структур данных и функций увеличивает число выполняемых команд процессора для выполнения некоторой функциональности. Но сегодня, благодаря выросшей производительности компьютеров, этим можно пренебречь в угоду понятной внутренней структуре. Особо следует отметить снижение объема документации проектов, поскольку понятная структура проекта позволяет не пояснять запутанные явления внутри кода.

Отметим главное, что в современных рыночных условиях важнее время, потраченное на разработку проекта, чем общая производительность первых версий проекта.Поэтому большинство языков высокого уровня имеют поддержку ООП. Среди всех таких языков следует выделить язык C++, как наиболее популярный и, при этом, наиболее похожий на большую часть языков (Си-подобные языки).

Данные методические материалы позволяют студенту подготовиться к практическим занятиям и выполнению контрольных работ по соответствующим темам. Методические рекомендации могут оказаться полезными при написании курсовых и выпускных квалификационных работ.

1. ОСНОВНЫЕ ПРИНЦИПЫ ОБЪЕКТНО-ОРИЕНТИРОВАННОГО ПРОГРАММИРОВАНИЯ (ООП). КЛАССЫ И ОБЪЕКТЫ

1.1. Теоретические сведения

Центральным понятием объектно-ориентированного программирования является класс. Классом называют пользовательский тип данных, объединяющий близкие по смыслу данные и методы работы с ними (функции, вложенные в класс). Примером могут быть данные о векторе в трехмерном пространстве. Вектор состоит из 3 координат по каждой из осей. Таким образом, тип для вектора должен включать описание 3-х вещественных полей данных и методы трансформации координат.

Тип данных это описание структуры данных. Любой тип допускает множество операций, определенных для него. Класс, являясь пользовательским типом, может содержать определения и реализации операций с собственными полями. Такой подход позволяет концентрировать возможные действия с данным типом в рамках одного модуля, что приводит к лучшей читаемости программного кода.

Класс, являясь типом, подразумевает создание собственных экземпляров. Чтение и запись значений возможны только для экземпляров классов, а не в их описание. Вызов методов класса тоже производится в контексте экземпляра класса. Важным следствием из этого является наличие предопределенной переменной ***this*** в каждом методе. Эта переменная является указателем на экземпляр класса, для которого вызван данный метод. Она обеспечивает доступ к полям и методам объекта (контексту) в процессе выполнения метода.

Отметим, что методы и поля класса могут быть помечены квалификатором ***static***. Они будут определены вместе с описанием класса, и обращение к ним возможно с использованием всего лишь имени класса, а не созданного экземпляра. Но при этом имя ***this*** не существует в контексте статических методов.

Большинство классов создается с целью создания их экземпляров. Создание экземпляра класса всегда сопровождается вызовом специального метода класса – конструктора. Это метод, имя которого совпадает с именем класса. Конструктор класса может быть перегружен, но назначение его остается неизменным – он служит для распределения памяти полей создаваемого экземпляра.

После окончания жизненного цикла объекта его необходимо разрушить. Для корректной утилизации памяти колей класса необходимо наличие деструктора. Деструктор – это метод класса, который вызывается при удалении объекта из памяти. Перегрузка деструктора невозможна. Имя деструктора начинается с символа ‘~’, при этом оставшаяся часть метода совпадает с именем класса. Действия, описываемые внутри деструктора должны быть направлены на освобождение памяти полей класса.

Класс может иметь поля пользовательских типов. Такое отношение классов называется композицией. При создании такого класса, его поле необходимо некоторым образом инициализировать и утилизировать. Для этого удобнее всего использовать конструктор и деструктор.

1.2.Банк заданий к контрольной работе №1 по разделу «Основные принципы объектно-ориентированного программирования (ООП). Классы и объекты»

1. Разработать класс на языке C++.
2. Разработать класс, которые будет являться полем класса, разработанного ранее.
3. Реализовать сценарии создания, поведения и утилизации полученного объекта класса.

1.3.Особенности оценивания контрольной работы в балльно-рейтинговой системе

Контрольная работа по разделу «Основные принципы объектно-ориентированного программирования (ООП). Классы и объекты» является промежуточной формой контроля знаний студентов и представляет собой письменное выполнение определенных заданий. Она предназначена для проверки знаний студентов по учебной дисциплине «Программирование на языках высокого уровня», а также служит для закрепления полученных знаний, умений и навыков. В контрольной работе студентам предлагаются задачи, сформулированные на основании материала, изложенного в лекциях проработанного на практических занятиях или самостоятельно изученного студентами. Перед тем как приступить к выполнению контрольной работы, студентам следует ознакомиться с теоретическим материалом и разобраться с разобранными в нем типовыми задачами.

Варианты контрольной работы состоят из трех заданий, которые представлены в разделе "Банк заданий к контрольной работе №1 по разделу "Основные принципы объектно-ориентированного программирования (ООП). Классы и объекты". Варианты отличаются тематикой разрабатываемого класса. Тематика определяется каждым субъектом уникально, и закрепляется за ним на все оставшиеся контрольные работы.

Система оценивания заданий контрольной работы №1 представлена в таблице 1.1.

Таблица 1.1 Оценивание контрольной работы №1 в БРС.

Критерии оценивания заданий	Количество баллов
Логично и последовательно выполнены все шаги решения, рассуждения имеют четкое обоснование.	8
Ход решения задания верный, но аргументация неполная.	6
Нарушена логическая цепочка рассуждений, решение неполное.	4
Максимальное количество баллов за контрольную работу №1	8

Оформление контрольной работы должно соответствовать Правилам оформления учебных работ студентов[1], принятым в НФИ КемГУ.

2. ПЕРЕГРУЗКА ОПЕРАЦИЙ

2.1.Теоретические сведения

Для любого пользовательского типа могут быть определены не только методы работы с данными, но и перегружены операции. Иными словами, существуют средства, позволяющие разрабатывать более понятный код для пользовательских с более явной семантикой. Средства являются возможностью перегрузки операций сложения, умножения, индексации и т.д.

На языке C++ существует 2 подхода к перегрузке операций: первый позволяет переопределять операции как методы класса; второй – перегружать операторы с использованием функций – друзей.

При перегрузке операций используется ключевое слово ***operator***, после которого следует обозначение самой операции. Полный список операций, для которых возможна перегрузка можно прочитать в литературе. Например, для операции ‘+’ нужно описать метод класса с именем: TMyClassTMyClass::operator + (TMyClass&arg). Данный метод будет вызван при использовании оператора ‘+’для 2х экземпляров класса TMyClass. Притом первый аргумент операции будет объектом, у которого вызван этот метод, а второй – аргументом данного метода.

Другой способ перегрузки метода заключается в использовании дружественной функции. Для этого функцию необходимо объявить в интерфейсе класса с квалификатором ***friend***. Однако реализация функции будет находится в том же модуле, но за пределами пространства имен класса. Вид функции будет похож на метод перегрузки, но вместо ***this*** будет использован дополнительный её аргумент: TMyClassoperator + (TMyClassarg1, TMyClassarg2). При этом дружественная функция имеет возможность доступа к закрытым полям класса.

2.2 Банк заданий к контрольной работе №2 по разделу «Перегрузка операций»

1. Разработать класс, для хранения числовых данных.
2. Продумать алгоритмы операций для полей класса и перегрузить основные арифметические операторы.
3. Разработать сценарии использования поученных объектов.

2.3 Особенности оценивания контрольной работ в балльно-рейтинговой системе

Контрольная работа по разделу «Перегрузка операций» является промежуточной формой контроля знаний студентов и представляет собой письменное выполнение определенных заданий. Она предназначена для проверки знаний студентов по учебной дисциплине «Программирование на языках высокого уровня», а также служит для закрепления полученных знаний, умений и навыков. В контрольной работе студентам предлагаются задачи, сформулированные на основании материала, изложенного в лекциях проработанного на практических занятиях или самостоятельно изученного студентами. Перед тем как приступить к выполнению контрольной работы, студентам следует ознакомиться с теоретическим материалом и разобраться с разобранными в нем типовыми задачами.

Варианты контрольной работы состоят из трех заданий, которые представлены в разделе "Банк заданий к контрольной работе №2 по разделу "Перегрузка операций". Варианты отличаются тематикой разрабатываемого класса. Тематика определяется каждым субъектом уникально, и закрепляется за ним на все оставшиеся контрольные работы.

Система оценивания заданий контрольной работы №2 представлена в таблице 2.1.

Таблица 2.1 Оценивание контрольной работы №2 в БРС.

Критерии оценивания заданий	Количество баллов

Логично и последовательно выполнены все шаги решения, рассуждения имеют четкое обоснование.	8
Ход решения задания верный, но аргументация неполная.	6
Нарушена логическая цепочка рассуждений, решение неполное.	4
Максимальное количество баллов за контрольную работу №1	8

Оформление контрольной работы должно соответствовать Правилам оформления учебных работ студентов[1], принятым в НФИ КемГУ.

3. НАСЛЕДОВАНИЕ И ПОЛИМОРФИЗМ

3.1.Теоретические сведения

Отличие класса от обычной структуры данных заключается в возможности определения отношения наследования. Наследование одного класса другим позволяет последнему использовать открытые и защищенные методы и данные базового класса. В частности, отношение открытого наследования указывает на общность объектов. Это позволяет присвоить указатель объекта-потомка указателю объекта предка, так как потомок частично является своим предком.

При определении конструктора класса-наследника необходимо указывать явно, какой из конструкторов класса-предка будет предшествовать данному конструктору. Все конструкторы вызываются согласно иерархии наследования – от базового к листу. Деструкторы будут вызваны в обратном порядке.

Методы и поля каждого класса входят в одну из 3х секций: ***public***, ***private***, ***protected***. Открытые методы доступны для вызова из любого пространства имен. Методы privateмогут быть вызваны только методами данного класса. Методы privateмогут быть вызваны в иерархии наследования. Это относится и к полям данных.

Любой указатель на экземпляр класса-наследника может быть присвоен указателю родительского класса. Но доступны для внешнего вызова останутся только имена методов класса-предка (интерфейс). Методы класса могут быть перекрыты в классе-наследнике. Способ замещения определяется наличием квалификатора ***virtual***. Если метод виртуальный, то при вызове будет выбран последний перекрывающий метод. В противном случае, будет вызван метод класса, на указатель которого присвоен экземпляр.

Особо отметим, что виртуальные методы, при вызове в конструкторе класса не рассматриваются как виртуальные. Они будут вызваны как статические, так как при вызове конструктора предка еще не сформирована и не инициализирована часть объекта, определенная в наследниках.

3.2 Банк заданий к контрольной работе №3 по разделу «Наследование и полиморфизм»

1. Разработать класс-контейнер для хранения битовое информации.
2. Разработать класс-наследник, реализующий строку.
3. Разработать сценарии использования обоих классов.

3.3.Особенности оценивания контрольной работы в балльно-рейтинговой системе

Контрольная работа по разделу «Наследование и полиморфизм» является промежуточной формой контроля знаний студентов и представляет собой письменное выполнение определенных заданий. Она предназначена для проверки знаний студентов по учебной дисциплине «Программирование на языках высокого уровня», а также служит для закрепления полученных знаний, умений и навыков. В контрольной работе студентам предлагаются задачи, сформулированные на основании материала, изложенного в лекциях проработанного на практических занятиях или самостоятельно изученного студентами. Перед тем как приступить к выполнению контрольной работы, студентам следует ознакомиться с теоретическим материалом и разобраться с разобранными в нем типовыми задачами.

Варианты контрольной работы состоят из трех заданий, которые представлены в разделе "Банк заданий к контрольной работе №3 по разделу "Наследование и полиморфизм". Варианты отличаются тематикой разрабатываемого класса. Тематика определяется каждым субъектом уникально, и закрепляется за ним на все оставшиеся контрольные работы.

Система оценивания заданий контрольной работы №3 представлена в таблице 3.1.

Таблица 3.1 Оценивание контрольной работы №3 в БРС.

Критерии оценивания заданий	Количество баллов
Логично и последовательно выполнены все шаги решения, рассуждения имеют четкое обоснование.	8
Ход решения задания верный, но аргументация неполная.	6
Нарушена логическая цепочка рассуждений, решение неполное.	4
Максимальное количество баллов за контрольную работу №1	8

Оформление контрольной работы должно соответствовать Правилам оформления учебных работ студентов[1], принятым в НФИ КемГУ.

4. СПИСОК РЕКОМЕНДУЕМОЙ УЧЕБНОЙ ЛИТЕРАТУРЫ

Основная учебная литература:

1. Абрамян, М.Э. Введение в стандартную библиотеку шаблонов C++. Описание, примеры использования, учебные задачи: учебник по курсу «Стандартная библиотека C++» для студентов направления 02.03.02 «Фундаментальная информатика и информационные технологии» (бакалавриат) / М.Э. Абрамян ; Южный федеральный университет. – Ростов-на-Дону ; Таганрог : Южный федеральный университет, 2017. – 179 с. : ил. – Режим доступа: по подписке. – URL: http://biblioclub.ru/index.php?page=book&id=499454 – Библиогр. в кн. – ISBN 978-5-9275-2374-0. – Текст : электронный.

2. Лямин, А. В. Языки программирования C/C++ : учебное пособие / А. В. Лямин, Е. Н. Череповская. — Санкт-Петербург : НИУ ИТМО, 2017. — 71 с. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/110458 — Режим доступа: для авториз. пользователей.

Дополнительная учебная литература:

1. Мариус, Б. Решение задач на современном C++ / Б. Мариус ; перевод с английского А. Н. Киселева. — Москва : ДМК Пресс, 2019. — 302 с. — ISBN 978-5-97060-666-7. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/123704 — Режим доступа: для авториз. пользователей.

Литература для оформления учебный работ:

Правила оформления учебных работ студентов : учебно-методическое пособие / И.А. Жибинова, А.Е. Аракелян, О.В. Соколова, Ю.Н. СоинаКутищева. – Новокузнецк : НФИ КемГУ, 2018. – 124 с. – Текст : непосредственный