ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ОРЕНБУРГСКИЙ ГОСУДАРСТВЕННЫЙ АГРАРНЫЙ УНИВЕРСИТЕТ»

МЕТОДИЧЕСКИЕ УКАЗАНИЯ ДЛЯ ОБУЧАЮЩИХСЯ ПО ОСВОЕНИЮ ДИСЦИПЛИНЫ

Б1.Б.16 ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ

Направление подготовки *09.03.01 Информатика и вычислительная техника*
Профиль образовательной программы **"***Автоматизированные системы обработки информации и управления***"**
Форма обучения *заочная*

СОДЕРЖАНИЕ

1. КОНСПЕКТ ЛЕКЦИЙ

1.1 Лекция № 1 (2 часа)

Тема: «Эволюция методологий программирования»

1.1.1 Вопросы лекции:

1. Первое поколение языков программирования, развитие алгоритмических абстракций, модуль как единица построения программных систем, третье поколение языков.

2. Зарождение объектной модели, четвертое поколение языков.

3. Объектные и объектно-ориентированные языки программирования.

4. Парадигмы программирования

1.1.2 Краткое содержание вопросов:

1 Первое поколение языков программирования, развитие алгоритмических абстракций, модуль как единица построения программных систем, третье поколение языков.

В эволюции методологий разработки ПО можно выделить несколько периодов. На заре программирования каждая программа разрабатывалась вручную одним разработчиком, который и был полным ее хозяином. Возникшие по мере усложнения программ трудности в их программировании и отладке стимулировали развитие систем автоматизации программирования. Эти системы включали средства компиляции, а также ограниченный набор средств отладки и документирования. На этом этапе все средства были ориентированы на работу с текстами программ.

Второй период начался с момента, когда в связи с увеличением размеров программ стало резко возрастать число людей, разрабатывающих программу. Это потребовало организации коллективной работы на основе применения структурных методов и структурных методологий. Упорядочивая структуру и дисциплинируя разработку, они помогали стандартизации и систематизации разработки ПО и его сопровождения. Все это позволяло представить процесс создания программ как инженерную дисциплину, которая опиралась на формализованное понятие "жизненный цикл ПО". При этом в жизненном цикле центр тяжести начал перемещаться с процессов программирования и отладки программ на процесс проектирования общей структуры ПО и структуры его компонент, наряду с чем продолжалось активное развитие средств автоматизации программирования и отладки. Этот период может считаться временем, когда были сформулированы основные принципы методологии SE, понимаемые в смысле технологии разработки ПО или техники ПО , и начата реализация этих принципов.

Третий период в эволюции методологии характеризуется автоматизацией структурных методологий, т. е. созданием инструментальной поддержки трудоемких работ по проектированию ПО в целях максимального высвобождения времени разработчиков. Именно с этим периодом связывается появление первой генерации CASE-средств. Последние принесли новую жизнь структурным методологиям созданием автоматизированных графических средств для выпуска различных схем, диаграмм, экранных и бумажных изображений, наличием словарей данных и хранилищ, появлением средств анализа и контроля структуры и текстов, генераторов документов, генераторов программных кодов и др.

Следующий период связан с объединением средств автоматизации структурных методологий со средствами автоматизации программирования и отладки программ. По существу, речь идет о начале создания интегрированных систем поддержки полного жизненного цикла разработки ПО, который реализуется во второй генерации CASE-средств. В течение этого периода в рамках второй генерации CASE-средств активно развиваются и используются методы повторной разработки ПО. Имеющиеся достижения и результаты, по мнению специалистов, свидетельствуют о достаточно полной реализации принципов методологий SE. В то же время проблемы, возникающие при создании современных программных систем, требуют перехода к более развитой методологии создания ПО.

С учетом сказанного последний период можно охарактеризовать как переход от методологии SE к методологии IE. На практике такой переход происходит достаточно плавно и постепенно путем ориентации методологии SE на централизованно-информационный подход к разработке ПО.

2. Зарождение объектной модели, четвертое поколение языков.

Методы структурного проектирования помогают упростить процесс разработки сложных систем за счет использования алгоритмов как готовых строительных блоков. Аналогично, методы объектно-ориентированного проектирования созданы, чтобы помочь разработчикам применять мощные выразительные средства объектного и объектно-ориентированного программирования, использующего в качестве блоков классы и объекты.

Объектно-ориентированный анализ и проектирование отражают эволюционное, а не революционное развитие проектирования; новая методология не порывает с прежними методами, а строится с учетом предшествующего опыта. К сожалению, большинство программистов в настоящее время формально и неформально натренированы на применение только методов структурного проектирования. Разумеется, многие хорошие

проектировщики создали и продолжают совершенствовать большое количество программных систем на основе этой методологии. Однако алгоритмическая декомпозиция помогает только до определенного предела, и обращение к объектно-ориентированной декомпозиции необходимо. Более того, при попытках использовать такие языки, как C++ или Ada, в качестве традиционных, алгоритмически ориентированных, мы не только теряем их внутренний потенциал - скорее всего результат будет даже хуже, чем при использовании обычных языков C и Pascal. Дать электродрель плотнику, который не слышал об электричестве, значит использовать ее в качестве молотка. Он согнет несколько гвоздей и разобьет себе пальцы, потому что электродрель мало пригодна для замены молотка.

3. Объектные и объектно-ориентированные языки программирования.

- Smalltalk
- C++
- Common Lisp Object System (CLOS)
- Ada
- Eiffel
- Java
- Object Pascal

4. Парадигмы программирования.

Парадигма программирования — это совокупность подходов, методов, стратегий, идей и понятий, определяющая стиль написания программ.

Парадигма программирования в современной индустрии программирования очень часто определяется набором инструментов программиста (язык программирования и операционная система).

Парадигма программирования представляет (и определяет) то, как программист видит выполнение программы. Например, в объектно-ориентированном программировании программист рассматривает программу как набор взаимодействующих объектов, тогда как в функциональном программировании программа представляется в виде цепочки вычисления функций.

1.2 Лекция № 2 (2 часа)

Тема: «Понятие объекта»

1.2.1 Вопросы лекции:

1. Состояние объекта.

2. Поведение объекта.

3. Идентичность объектов.

1.2.2 Краткое содержание вопросов:

1. Состояние объекта.

Состояние объекта характеризуется перечнем (обычно статическим) всех свойств данного объекта и текущими (обычно динамическими) значениями каждого из этих свойств.

К числу свойств объекта относятся присущие ему или приобретаемые им характеристики, черты, качества или способности, делающие данный объект самим собой. Например, для лифта характерным является то, что он сконструирован для поездок вверх и вниз, а не горизонтально.

Все свойства имеют некоторые значения. Эти значения могут быть простыми количественными характеристиками, а могут ссылаться на другой объект. Состояние лифта может описываться числом 3, означающим номер этажа, на котором лифт в данный момент находится.

Простые количественные характеристики (например, число 3) являются «постоянными, неизменными и непреходящими», тогда как объекты «существуют во времени, изменяются, имеют внутреннее состояние, преходящи и могут создаваться, уничтожаться и разделяться».

Тот факт, что всякий объект имеет состояние, означает, что всякий объект занимает определенное пространство (физически или в памяти компьютера).

Все объекты в системе инкапсулируют некоторое состояние, и все состояние системы инкапсулировано в объекты.

2. Поведение объекта.

Поведение — это то, как объект действует реагирует; поведение выражается в терминах состояния объекта и передачи сообщений.

Иными словами, поведение объекта — это его наблюдаемая и проверяемая извне деятельность.

Операцией называется определенное воздействие одного объекта на другой с целью вызвать соответствующую реакцию. В чисто объектно-ориентированном языке, таком как Smalltalk, принято говорить о передаче сообщений между объектами. В языках типа C++ мы говорим, что один объект вызывает функцию-член другого. В основном понятие сообщение совпадает с понятием операции над объектами, хотя механизм передачи различен. В объектно-ориентированных языках операции, выполняемые над данным объектом, называются методами и входят в определение класса объекта. В C++ они называются функциями-членами.

Состояние объекта представляет суммарный результат его поведения.

Наиболее интересны те объекты, состояние которых не статично: их состояние изменяется и запрашивается операциями.

3. Идентичность объекта.

«Идентичность — это такое свойство объекта, которое отличает его от всех других объектов».

Они отмечают, что «в большинстве языков программирования и управления базами данных для различения временных объектов их именуют, тем самым путая адресуемость и идентичность. Большинство баз данных различают постоянные объекты по ключевому атрибуту, тем самым, смешивая идентичность и значение данных». Источником множества ошибок в объектно-ориентированном программировании является неумение отличать имя объекта от самого объекта.

1.3 Лекция № 3 (2 часа)

Тема: «Представление объектов и классов»

1.3.1 Вопросы лекции:

1 Реализация поведения объектов на примере добавления функций-членов в структуры.

2 Структура как вырожденный класс.

3 Структура объявления класса.

4 Доступ к членам класса.

5 Поля данных класса как механизм реализации состояния объекта.

6 Функции члена класса как механизм реализации поведения объекта.

7 Спецификаторы доступа для обеспечения инкапсуляции.

8 Средства управления жизнью объекта.

1.3.2 Краткое содержание вопросов:

1 Реализация поведения объектов на примере добавления функций-членов в структуры.

Пользователям, по-видимому, понадобится широкий набор операций над объектами типа Screen: возможность перемещать курсор, проверять и устанавливать области экрана и рассчитывать его реальные размеры во время выполнения, а также копировать один объект в другой. Все эти операции можно реализовать с помощью функций-членов.

Функции-члены класса объявляются в его теле. Это объявление выглядит точно так же, как объявление функции в области видимости пространства имен. (Напомним, что глобальная область видимости – это тоже область видимости пространства имен.

2 Структура как вырожденный класс.

Конструкция struct языка C не была принята в языке программирования Java потому, что класс выполняет все то же самое, что может делать структура, и даже более того. Структура группирует несколько полей данных в один общий объект, тогда как класс связывает с полученным объектом операции, а также позволяет скрывать поля данных от пользователей объекта. Иными словами, класс может инкапсулировать encapsulate) свои данные в объекте, доступ к которому осуществляется только через его методы.

3 Структура объявления класса.

Классы и объекты в C++ являются основными концепциями объектноориентированного программирования — ООП. Объектно-ориентированное программирование — расширение структурного программирования, в котором основными концепциями являются понятия классов и объектов. Основное отличие языка программирования C++ от C состоит в том, что в C нет классов, а следовательно язык C не поддерживает ООП, в отличие от C++.

Чтобы понять, для чего же в действительности нужны классы, проведём аналогию с каким-нибудь объектом из повседневной жизни, например, с велосипедом. Велосипед — это объект, который был построен согласно чертежам. Так вот, эти самые чертежи играют роль классов в ООП. Таким образом классы — это некоторые описания, схемы, чертежи по которым создаются объекты. Теперь ясно, что для создания объекта в ООП необходимо сначала составить чертежи, то есть классы. Классы имеют свои функции, которые называются методами класса. Передвижение велосипеда осуществляется за счёт вращения педалей, если рассматривать велосипед с точки зрения ООП, то механизм вращения педалей — это метод класса. Каждый велосипед имеет свой цвет, вес, различные составляющие — всё это свойства. Причём у каждого созданного объекта свойства могут различаться. Имея один класс, можно создать неограниченно количество объектов (велосипедов), каждый из которых будет обладать одинаковым набором методов, при этом можно не задумываться о внутренней реализации механизма вращения педалей, колёс, срабатывания системы торможения, так как всё это уже будет определено в классе. Разобравшись с назначением класса, дадим ему грамотное определение.

4 Доступ к членам класса.

Поддержка свойства инкапсуляции в классе дает два главных преимущества. Вопервых, класс связывает данные с кодом. И во-вторых, класс предоставляет средства для управления доступом к его членам. Именно эта, вторая преимущественная особенность и будет рассмотрена в данной статье.

В языке C#, по существу, имеются два типа членов класса: открытые и закрытые, хотя в действительности дело обстоит немного сложнее. Доступ к открытому члену свободно осуществляется из кода, определенного за пределами класса. А закрытый член класса доступен только методам, определенным в самом классе. С помощью закрытых членов и организуется управление доступом.

Ограничение доступа к членам класса является основополагающим этапом объектно-ориентированного программирования, поскольку позволяет исключить неверное использование объекта. Разрешая доступ к закрытым данным только с помощью строго определенного ряда методов, можно предупредить присваивание неверных значений этим данным, выполняя, например, проверку диапазона представления чисел. Для закрытого члена класса нельзя задать значение непосредственно в коде за пределами класса. Но в то же время можно полностью управлять тем, как и когда данные используются в объекте. Следовательно, правильно реализованный класс образует некий "черный ящик", которым можно пользоваться, но внутренний механизм его действия закрыт для вмешательства извне.

5 Поля данных класса как механизм реализации состояния объекта.

Как уже говорилось выше, в современных объектно-ориентированных языках программирования каждый объект является значением, относящимся к определённому классу. Класс представляет собой объявленный программистом составной тип данных, имеющий в составе:

Поля данных

Параметры объекта (конечно, не все, а только необходимые в программе), задающие его состояние. Поля данных объекта называют свойствами объекта. Физически свойства представляют собой значения (переменные, константы), объявленные как принадлежащие классу.

Методы

Процедуры и функции, связанные с классом. Они определяют действия, которые можно выполнять над объектом такого типа, и которые сам объект может выполнять.

6 Функции члена класса как механизм реализации поведения объекта.

Функция-член класса может содержать спецификатор virtual. Такая функция называется виртуальной. Спецификатор virtual может быть использован только в объявлениях нестатических функций-членов класса.

Если некоторый класс содержит виртуальную функцию, а производный от него класс содержит функцию с тем же именем и типами формальных параметров, то обращение к этой функции для объекта производного класса вызывает функцию, определённую именно в производном классе. Функция, определённая в производном классе, вызывается даже при доступе через указатель или ссылку на базовый класс. В таком случае говорят, что функция производного класса подменяет функцию базового класса. Если типы функций различны, то функции считаются разными, и механизм виртуальности не включается. Ошибкой является различие между функциями только в типе возвращаемого значения.

7 Спецификаторы доступа для обеспечения инкапсуляции.

Начиная с этой лекции мы будем подробно рассматривать принципы объектно-ориентированного программирования только на примере языка C#. Язык C# является полноценным объектно-ориентированным языком программирования со строгим контролем типов. Мы будем рассматривать объектно-ориентированное программирование только для языка C#, поскольку этот язык поддерживает все основные конструкции объектно-ориентированного подхода за исключением множественного наследования.

В C# объектно-ориентированное программирование поддерживается с помощью классов. Класс - это пользовательский тип данных, который включает в себя как данные, так и программный код в виде функций. Данные хранимые в классе называются полями, а функции класса называются методами. Также класс может содержать свойства и индексаторы, которые в определенном смысле объединяют поля и методы. После того, как класс описан, можно создавать переменные с типом этого класса.

8 Средства управления жизнью объекта.

Методология управления, менеджмента – это логическая схема управленческой деятельности, предполагающая взаимосвязанное понимание целей, ориентиров, а также средств и способов их достижения. Это еще и умение видеть, распознавать, понимать, оценивать и учитывать зависимости, которые и раскрывают содержание проблем, и подсказывают пути их решения.

Специалисты выделяют следующие компоненты, характеризующие содержание методологии управления: подходы, парадигмы, проблемы, приоритеты, ориентиры, критерии, альтернативы, процедуры выбора, средства и методы управления, а также ограничения.

Подход – наиболее принципиальный компонент методологии, определяющий выбор и использование остальных ее компонентов. Среди подходов в управлении особо выделяется системный подход. Весьма популярны в управлении программно-целевой и проектный подходы. Для современного менеджмента характерен маркетинговый подход, ориентированный на потребителя, кроме того, для него свойственны кибернетический, информационный, гуманистический подход – при этом заметен акцент на преимущественном использовании отдельных наук, типов ресурсов и соответствующей методологии в целом. Многие исследователи выделяют интеграционный и сетевой подходы. Принимая на вооружение тот или иной подход, можно говорить о науке управления и об управлении как искусстве; выбранный подход в таком случае обычно конкретизируется через те или иные принципы управления.

Парадигма (от греч. paradeigma пример, образец) – исходная концептуальная схема, система понятий, отражающая осмысление существенных черт действительности, модель постановки проблем и их решения, выбора соответствующих методов, господствующая в научном сообществе в течение определенного исторического периода и знаменующая собой определенный этап в развитии теории (управленческий рационализм Ф.Тейлора, функциональная дифференциация А. Файоля и М. Вебера и др.) В современной парадигме управления, несмотря на разнообразие конкретных формулировок, предпочтение отдается человеческой личности, с учетом процессов глобализации и с акцентом на управление знаниями, на сетевые, партнерские принципы взаимодействия.

2. МЕТОДИЧЕСКИЕ МАТЕРИАЛЫ ПО ВЫПОЛНЕНИЮ ЛАБОРАТОРНЫХ РАБОТ

2.1 Лабораторная работа № 1 (2 часа).

Тема: «Понятие объекта»

2.1.1. Вопросы к занятию:

1. Состояние объекта.
2. Поведение объекта.
3. Идентичность объектов.

2.1.2. Краткое описание проводимого занятия:

Задания для проведения текущего контроля успеваемости

1. TRadiogroup в Delphi – это:

1) группа зависимых переключателей;
2) независимый переключатель;
3) группа независимых переключателей;

4) кнопка зависимого переключателя;
5) нет верного ответа.

2. Свойство Columns типа Integer задает в Delphi:
1) число столбцов;
2) номер выделенного элемента;
3) число выделенных компонентов на форме;
4) размеры выделенного компонента;
5) раскрыт ли список.

3. Методы объекта определяют его в Delphi:
1) поведение;
2) свойства;
3) родителя;
4) атрибуты;
5) потомка.

2.2 Лабораторная работа № 2 (2 часа).

Тема: «Разработка Visual Basic-приложений. Создание программного интерфейса пользователя»

2.2.1. Вопросы к занятию:

1. Visual приложение.
2. Создание программного интерфейса пользователя.

2.2.2. Краткое описание проводимого занятия:

Задания для проведения текущего контроля успеваемости

1. Проект в Visual Basic – это:
1) один или несколько программных модулей
2) одна или несколько экранных форм
3) совокупность частей, составляющих Windows-приложение
4) интерфейс программы + программный код

2. Visual Basic хранит каждый проект в файле с расширением:
1) *.vbp
2) *.frm
3) *.bas
4) *.frx

3. Программный код проекта Visual Basic:
1) существует сам по себе
2) привязан к отдельным объектам и не оторван от формы
3) привязан к отдельным объектам и не связан с формой
4) привязан только к форме

2.3 Лабораторная работа № 3 (2 часа).

Тема: «Визуальная модель Delphi»

2.3.1. Вопросы к занятию:

1. Процедурное программирование
2. Объектно-ориентированное программирование
3. Компонентная модель, наследственность

2.3.2. Краткое описание проводимого занятия:

Задания для проведения текущего контроля успеваемости

1. Библиотека компонент Delphi. Визуальные и невизуальные компоненты.

2. Иерархия классов Delphi. Краткая характеристика основных классов Delphi и их назначение.

3. Стандартные события (события мыши, клавиатуры, системные события) визуальных компонент в Delphi.

4. Организация текстового диалога. Обзор стандартных окон и стандартных компонент в Delphi.

5. Работа с многострочным текстом. Компонент TMemo, классы Tstrings, TStringList в Delphi.

6. Обзор стандартных компонент управления (выключатели, переключатели, списки, контейнеры). Их взаимодействие.

2.4 Лабораторная работа № 4 (2 часа).

Тема: «Природа классов»

2.4.1. Вопросы к занятию:

1 Интерфейс и реализация

2 Отношение между классами

3 Ассоциация

4 Наследование. Множественное наследование

2.4.2. Краткое описание проводимого занятия:

Задания для проведения текущего контроля успеваемости

1. Базовый класс для всех компонентов в Delphi:
   1) TObject;
   2) TControl;
   3) TComponent;
   4) TPersistent;
   5) TWinControl.
2. В каком окне представлен список классов и объектов в VBA?
   1) окне макета формы;
   2) окне программного кода;
   3) проводнике объектов;
   4) окне свойств.
3. Методы объекта определяют его в Delphi:
   1) поведение;
   2) свойства;
   3) родителя;
   4) атрибуты;
   5) потомка.

2.5 Лабораторная работа № 5 (2 часа).

Тема: «Представление объектов и классов»

2.5.1. Вопросы к занятию:

1 Доступ к членам класса.

2 Поля данных класса как механизм реализации состояния объекта.

3 Функции члена класса как механизм реализации поведения объекта.

4 Спецификаторы доступа для обеспечения инкапсуляции.

2.5.2. Краткое описание проводимого занятия:

Задания для проведения текущего контроля успеваемости

1. Жизненный цикл объекта.

2. Конструкторы и деструкторы.

3. Порядок вызова конструкторов и деструкторов при наследовании.

2.6 Лабораторная работа № 6 (2 часа).

Тема: «Наследование как средство организации иерархий классов»

2.6.1. Вопросы к занятию:

1 Одиночное наследование.
2 Множественное наследование.
3 Пространства имен.

2.6.2. Краткое описание проводимого занятия:

Задания для проведения текущего контроля успеваемости

1. Понятие класса в Delphi. Отличие класса Delphi от записей Pascal.

2. Свойства и методы базового класса Delphi TObject.

3. Создание и уничтожение экземпляра класса в Delphi.

4. Понятие свойства класса. Синтаксис свойств и их достоинства в Delphi.