УВАЖАЕМЫЕ СТУДЕНТЫ! Изучите приведенную лекцию, законспектируйте основные тезисы, дайте ответы на контрольные вопросы.

Ответы на вопросы, фотоотчет, предоставить преподавателю на e-mail: r.bigangel@gmail.com до 13.02.2023.

При возникновении вопросов по приведенному материалу обращаться по следующему номеру телефона: (072)111-37-59, (Viber, WhatsApp), vk.com: https://vk.com/daykini

ВНИМАНИЕ!!! При отправке работы, не забывайте указывать ФИО студента, наименование дисциплины, дата проведения занятия (по расписанию).

Лекция № 37
Тема «Событийно-ориентированное и структурное программирование»

План лекции:

1. Алгоритмическое или процедурное
2. Структурное программирование
3. Объектно-ориентированное
4. Событийно-ориентированное программирование

Алгоритмическое программирование

Основная идея алгоритмического программирования – разбиение программы на последовательность модулей, каждый из которых выполняет одно или несколько действий. При этом выполнение модуля всегда начинается с первой команды и заканчивается самой последней, чтобы нельзя было попасть на команды модуля извне и передать управление из модуля на другие команды минуя заключительную. Алгоритм на выбранном языке записывается с помощью команд описания данных, вычисления значений и управления последовательностью выполнения программы. Текст программы представляет собой линейную последовательность операторов.

Используется для решения не сложных задач, когда программа состоит из нескольких сот строк кода.

Структурное программирование

При создании средних по размеру приложений (несколько тысяч строк исходного кода) используется структурное программирование. Его **идея** заключается в том, что структура программы должна содержать структуру решаемой задачи, чтобы алгоритм решения был явно виден из исходного текста. В качестве средства создания программы используется **подпрограмма** – набор операторов, выполняющих нужное действие и не зависящих от других частей исходного кода. Программа разбивается на множество мелких подпрограмм (занимающих до 50 операторов – критический порог для быстрого понимания цели программы), каждая из которых выполняет одно из действий, предусмотренных исходным заданием. Комбинируя эти программы, удается формировать итоговый алгоритм уже не из простых операторов, а из законченных блоков кода, имеющих определенную смысловую нагрузку, причем обращаться к таким блокам можно по названиям. Получается, что подпрограммы – это новые операторы или операции языка, определяемые программистом. Подпрограммы бывают двух видов процедуры и функции. Процедура просто выполняет группу операторов. Функция - выполняет группу операторов, вычисляет некоторое значение и передает его обратно в главную программу. При этом значение имеет определенный тип.

Возможность использования подпрограмм относит язык программирования к классу **процедурных** языков.

Наличие подпрограмм позволяет вести проектирование и разработку приложения **сверху вниз** – такой подход называется *нисходящим проектированием* . При этом задача разбивается на большое число мелких подзадач, каждая из которых решается своей процедурой или функцией (декомпозиция задачи). При этом проектирование программы идет по

принципу сверху вниз: сначала определяются необходимые для решения про граммы модули, их входы и выходы, а затем уже эти модули разрабатываются. Такой под ход вместе с локальными именами переменных позволяет разрабатывать проект силами большого числа программистов.

Подпрограммы могут быть **вложенными** , если она вызывается не только из главной программы, но и из любых подпрограмм. Допускается вызов подпрограммы из самой себя – такой прием называется **рекурсией** .

Структура подпрограммы: заголовок с параметрами, тело подпрограммы (операторы, которые будут выполняться при ее вызове), завершение подпрограммы.

Параметры, которые указываются в заголовке программы, могут быть

– формальными , которые нужны только для описания тела программы;

– фактическими - конкретные значения, которые указываются в момент вызова подпрограммы.

как доказал Э. Дейкстра, любой алгоритм можно реализовать, используя лишь три управ ляющие конструкции:

– последовательное выполнение,

– ветвление

– цикл

Не должно быть безусловных переходов!

Правила композиции, используемые при структурном подходе к составлению алгоритмов:

а) альтернативный выбор

б) цикл

с) подпрограмма

Объектно-ориентированный подход **к программированию** - это подход к разработке программного обеспечения, основанный на объектах.

Реальные объекты окружающего мира обладают тремя базовыми характеристиками: они имеют набор свойств, способны разными методами изменять эти свойства и реагировать на события, возникающие как в окружающем мире, так и внутри самого объекта. **Объект** – совокупность свойств (структур данных, характерных для этого объекта), методов их обработки (подпрограмм изменения свойств) и событий, на которые данный объект может реагировать и, которые приводят, как правило, к изменению свойств объекта.

Класс – новый тип, совокупность объектов имеющих идентичную структуру, отличающихся только значениями. Каждый конкретный объект называют **экземпляром класса** . Описание нового класса похоже на описание новой структуры данных, только к полям (свойствам) добавляются методы – подпрограммы.

Важнейшая характеристика класса – возможность создания на его основе новых классов с **наследованием** всех свойств и методов добавления собственных. Класс, не имеющий предшественника, называют базовым. Например, класс «животные» имеет свойства «название», «размер», методы «идти», «размножаться». Созданный на его основе класс «кошка» наследует все его свойства и методы, к которым дополнительно добавляется свойство «окраска» и метод «пить».

Свойство объектов переопределять методы наследуемого класса называется **полиморфизмом** . Например, в большинстве случаев методы базового класса у классов-наследников приходится переопределять – объект класса «кошка» выполняет метод «идти» совсем не так как «амеба». Все переопределяемые методы по названию (написанию) будут совпадать с методами базового объекта. Компилятор по типу объекта (его классу) распознает, какой конкретный метод надо использовать. И не вызовет для класса «амеба» метод «идти» класса «кошка».

***Система программирования* — это система для разработки новых программ на конкретном языке программирования.**

Современные системы программирования обычно предоставляют пользователям **мощные и удобные средства разработки программ.** В них входят:

– компилятор или интерпретатор;

– интегрированная среда разработки;

– средства создания и редактирования текстов программ;

– обширные библиотеки стандартных программ и функций;

– отладочные программы, т.е. программы, помогающие находить и устранять ошибки в программе;

– "дружественная" к пользователю диалоговая среда;

– многооконный режим работы;

– мощные графические библиотеки; утилиты для работы с библиотеками

– встроенный ассемблер;

– встроенная справочная служба;

– другие специфические особенности.

Популярные системы программирования — ***Turbo Basic*** , ***Quick Basic*** , ***Turbo Pascal*** , ***Turbo C*** .

В последнее время получили распространение системы программирования, ориентированные на создание *Windows-приложений*:

– пакет Borland Delphi (Делфи) — блестящий наследник семейства компиляторов Borland Pascal, предоставляющий качественные и очень удобные средства визуальной разработки. Его исключительно быстрый компилятор позволяет эффективно и быстро решать практически любые задачи прикладного программирования.

– пакет Microsoft Visual Basic — удобный и популярный инструмент для создания Windows-программ с использованием визуальных средств. Содержит инструментарий для создания диаграмм и презентаций.

– пакет Borland C++ — одно из самых распространённых средств для разработки DOS и Windows приложений.

Событийно-ориентированное программирование

С активным распространением системы Windows и появлением визуальных RAD-сред (Rapid Application Development – быстрая разработка приложений, например Borland Delphi и Borland C++ Builder) широкую популярность приобрел событийный подход к созданию программ — событийно-ориентированное программирование.

Идеология системы Windows основана на событиях. Щелкнул человек на кнопке, выбрал пункт меню, нажал на клавишу или кнопку мыши — в Windows генерируется соответствующее сообщение, которое отсылается окну соответствующей программы.

Структура программы, созданной с помощью событийного программирования, следующая. Главная часть представляет собой один бесконечный цикл, который опрашивает Windows, следя за тем, не появилось ли новое сообщение. При его обнаружении вызывается подпрограмма, ответственная за обработку соответствующего события (обрабатываются не все события, их сотни, а только нужные), и подобный цикл опроса продолжается, пока не будет получено сообщение «Завершить работу».

События могут быть пользовательскими, возникшими в результате действий пользователя, системными, возникающими в операционной системе (например, сообщения от таймера), и программными, генерируемыми самой программой (например, обнаружена ошибка и ее надо обработать).

Событийное программирование является развитием идей нисходящего проектирования, когда постепенно определяются и детализируются реакции программы на различные события.

Контрольные вопросы:

1. Алгоритмическое или процедурное программирование

2. Структурное программирование
3. Объектно-ориентированное программирование
4. Событийно-ориентированное программирование