**Конспект урока на тему   
«Языки программирования.  
Основы объектно-ориентированного программирования»**

**Цель урока:**

* *рассмотреть этапы развития языков программирования*
* *ввести основные понятия ООП*
* *познакомить с интегрированной средой программирования VB*

**План урока:**

1. **Теоретическая основа урока**

**Языки программирования. Компиляторы и интерпретаторы.** Каждый процессор имеет свою систему команд. Компьютер способен понять только последовательность команд, понятных процессору, - машинный код. Первоначально программы для компьютеров писались с использованием машинного кода. Программирование в машинном коде – трудоемкий процесс, в ходе которого трудно избежать ошибок. Упростить этот процесс можно, если автоматизировать работу, поручив часть ее самому компьютеру. Поэтому сегодня для записи программ используются языки программирования. Язык программирования - это формальный язык для записи алгоритмов в виде, допускающем их автоматическую подготовку к выполнению на компьютере. Для преобразования программы в машинный код служит специальное программное средство – транслятор.  
Трансляторы делятся на две группы  по их работе – компиляторы интерпретаторы. Интерпретатор преобразует команды исходного текста программы в машинные команды и немедленно их выполняет. Можно сказать, что программа выполняется по стокам исходного текста. При этом происходит проверка правильности написания строк программы с точки зрения правил языка. При обнаружении ошибок специальный отладчик сообщает об ошибке, исполнение программы останавливается. Такой режим очень удобен для программиста, потому что результаты труда сразу же видны в деле. Однако интерпретация программ – довольно медленный процесс. Он заметно усложняется, если программа состоит из нескольких модулей. Кроме того, для запуска созданной программы на конкретном компьютере необходимо, чтобы на нем была установлена программа-интерпретатор.

Компилятор просматривает текст программы (иногда несколько раз – такие компиляторы называют многопроходными) и создает последовательность данных, которая называется объектным кодом. Объектный код еще не является полным аналогом программы. Необходимо дополнительный этап, который называется редактированием связей (или компоновкой). На этом этапе происходит объединение объектного кода программы и объектного кода подпрограмм, взятых из внешних библиотек. Результатом этого этапа является так называемый исполнимый код – он представляет собой набор машинных команд, реализующих алгоритм, записанный в программе. Исполнимый код может запускаться  автономно на любом компьютере подходящей платформы. Код, прошедший компиляцию, работает в десятки раз быстрее, чем его эквивалентная программа в режиме интерпретации Операцию компоновки кода выполняет отдельная программа, которая называется редактором связей или компоновщиком. В современных системах программирования компоновщик часто объединяют с компилятором, так что для программиста оба этапа сливаются в один. Двухступенчатый процесс облегчает создание больших и сложных программ.

И компиляторы, и интерпретаторы имеют свои достоинства. Граница между ними постепенно стираются. Некоторые системы разработки программ содержат в своем составе, как компилятор, так и интерпретатор для поддерживаемого языка программирования. В некоторых случаях используются смешанные технологии.

**Уровни языков**

Язык программирования содержит три основных компонента: алфавит, синтаксис и семантику. Эти компоненты определяют правила записи программ. Алфавит языка – это набор символов, которые можно применять в инструкциях языка программирования. Другие символы допустимы только в особых случаях, например в строковых константах. Синтаксис языка определяет правила построения операторов. Семантика – это смысловое содержание операторов языка программирования. Семантические правила определяют действия, описываемые различными операторами, и, в итоге сущность всего алгоритма.

Языки программирования можно разделить на две группы – языки высокого уровня и языки низкого уровня (машинные).  
К языкам низкого уровня относится язык Ассемблер, в котором программа пишется на уровне машинных кодов. Инструкция языка ассемблера описывает ровно одну машинную команду. И наоборот: каждой команде в системе команд процессора соответствует  инструкция языка (мнемоника). По сравнению с машинным кодом язык ассемблер имеет ряд преимуществ, облегчающих труд программист:

* Символические мнемоники запоминаются легче, чем шестнадцатеричные коды команд.
* Для регистров и областей памяти также можно использовать символические имена.
* Нет необходимости работать с физическими адресами памяти.
* Числовые константы и строки представляются в программе в привычном виде.

Программировать на языке ассемблера намного проще, чем в машинном коде, но все рано сложно. Поэтому были созданы другие языки программирования, в которых каждая инструкция (оператор) языка преобразуется в группу машинных. Эти языки ориентируются не на систему команд процессора, а на способ мышления, присущий человеку. Языки удобные для людей, называют языками высокого уровня. Достоинства языков высокого уровня:

* Машинная независимость.
* Использование естественных обозначений.
* Эффективное представление этапов обработки данных средствами языка.
* Готовые библиотеки стандартных подпрограмм для выполнения часто встречающихся действий.

Все языки программирования высокого уровня делятся на процедурные, логические и объектно-ориентированные. Несмотря на различия между языками, все они позволяют написать программу любого назначения.  
Каждый язык программирования имеет свое название. История этих названий восходит к моменту создания языков. Правила программ в некоторых языках неоднократно менялись, но названия языков остались без изменения.

**Поколения языков**  
**1 этап.**

Операционное программирование. (ЭВМ 1-го поколения с 1945-1959 год).  ЭВМ того времени понимали только цифровые команды, и программы состояли из множества строк, состоящих из цифр, интерпретируемых центральным процессором. Например, команда 05 825 631 трактовалась как сложение двух чисел (код 05), записанных в ячейки с номерами 825 и 631. Производительность труда программистов того времени была очень невелика, так как вручную было необходимо распределить все переменные программы в оперативной памяти.

**2 этап.**

Мало отличается от первого. Он связан с ЭВМ 2-го поколения. Появились языки программирования типа Ассемблер и автокод. Теперь команда сложения записывалась с использованием служебных слов – ADD (сложить) PR1, ZET, где ADD – код команды, PR1, ZET – имена ячеек. Перевод программы (трансляция), записанных таким образом в цифровое представление, а только такое понимает ЭВМ, осуществляется с помощью специальных программ, называемых ассемблерами. Технология работы программиста: программа собирается из мелких деталей, отдельных операций и имеет достаточно простую структуру, решаемые задачи в основном расчетные.

**3 этап.**

Развиваются языки программирования высокого уровня. В них реализуются новые идеи: подпрограммы и раздельная компиляция (Фортран 2); блочная структура и типы данных (Алгол 60); описание данных и работа с файлами (Кобол); обработка списков и указателей (Лисп). В следующих версиях языков продолжается развитие: PL/1 (Фортран+Алгол+Кобол), Алгол 68 (приемник Алгол 60), Паскаль (развитие Алгол 60), Simula (классы абстрактные данные).   
Возможности языков программирования обеспечивают поддержку нисходящей технологии конструирования программ. Суть нисходящего конструирования программ – разбиение большой задачи на подзадачи, которые могут рассматриваться отдельно.

Основные правила применения данной технологии:

* формализованное  и  строгое описание программистом входов функций и выходов всех модулей программы и системы;
* согласованная разработка структур данных и алгоритмов;
* ограничение на размер модулей;

Восходящая технология конструирования программ – решение складывается «из отдельных кирпичиков», из известных решений подзадач. Таким образом, данной технологией оговаривается определенный принцип декомпозиции и иерархическая структура программы. Важнейшей составляющей этой технологии является структурное программирование (языки программирования Паскаль, Модула –2). Пионером структурного программирования был Э. Дейкстра, который в 1965 году предположил, что оператор GOTO может быть исключен из языков программирования.   
Характерные черты структурного стиля программирования:

* простота и ясность (программа легко читается и анализируется);
* использование только базовых конструкций;
* отсутствие сетевых структур в программе;
* отсутствие многоцелевых функциональных блоков;
* отсутствие неоправданно сложных арифметических и логических конструкций;
* расположение в строке программы не более одного оператора языка программирования;
* содержательность имен переменных.

При этом процесс нисходящей разработки программы может продолжаться до тех пор, пока не будет, достигнут уровень «атомарных» блоков, т.е. базовых конструкций.  
К нисходящей технологии следует отнести и то, что называется модульным программированием. Достаточно независимые фрагменты задачи оформляются как модули. Создаются библиотеки модулей, определяется механизм включения модулей в разрабатываемую программу. Модуль должен иметь строго определенный интерфейс и скрытую часть,  одну точку входа и одну точку выхода.Структурная технология предоставила в распоряжение разработчиков строгие, формализованные методы описания программ и принимаемых технических решений. При этом использовалась наглядная графическая техника (схемы, диаграммы). Программы имели последовательную структуру, идеи Э. Дейкстры были реализованы в полной мере, что определило новый этап в развитии технологии программирования.

**4 этап.**

Связан с применением объектно-ориентированных языков 4-го поколения. В основе объектно-ориентированного программирования (ООП) лежит идея объединения в одной структуре данных и действий, которые производятся с этими данными. (В терминологии ООП такие действия называются методами). При таком подходе организация данных и программная реализация действий над ними оказываются гораздо сильнее связаны, чем при традиционном структурном программировании.  
ООП базируется на трех основных понятиях:

* **Инкапсуляция** – комбинирование данных с процедурами и функциями, которые манипулируют этими данными. В результате получается новый тип данных – объект.
* **Наследование** – это возможность использования уже определенных объектов для построения иерархии объектов производных от них. Каждый из «наследников» наследует описание данных «прародителя» и доступ к методам их обработки.
* **Полиморфизм** – это возможность определения единого по имени действия (процедуры или функции), применимого одновременно ко всем объектам иерархии наследования, причем каждый объект иерархии может «заказывать» особенность реализации этого действия над «самим собой».

ООП может заметно упростить написание сложных программ, придать им гибкость. Одним из его главных преимуществ можно назвать возможность расширять область их применения, не переделывая программу, а лишь добавляя в нее новые уровни иерархии.  
Первым языком с элементами ООП был язык Симула-67. В Турбо-Паскале, начиная с версии 5.5, появились средства ООП. Итогом развития Турбо-Паскаля в этом направлении стало создание фирмой Borland системы программирования Delphi (Делфи). Использование этой системы, в частности, даёт возможность легко и быстро программировать сложный графический интерфейс.

**5 этап.**

ЭВМ будущего 5-го поколения называют машинами «искусственного интеллекта». Прототипы языков для этих машин были созданы много раньше их физического появления. Это языки Лисп и Пролог. Эти языки относятся к языкам логического программирования.    
Язык Пролог разработан в 70-ых годах во Франции. Его появление связано с первыми работами в области создания искусственного интеллекта (экспертных систем, программ-переводчиков, интеллектуальных игр и пр.). Базируется на логической модели знаний.   
  
**Основы  ООП**

Системы ООП дают возможность визуализировать процесс создания графического интерфейса разрабатываемого приложения, то есть позволяют создать объекты и задавать значения их свойств с помощью диалоговых окон системы программирования.  Взаимодействия объектов между собой и их изменения описываются с помощью программного кода. Создание программного кода базируется на исполнении алгоритмических структур различных типов.   Основной единицей в ООП является объект, который объединяет в   себе как описывающие его данные, так и средства обработки этих   
данных, т.е. программные объекты обладают свойствами, могут,  использовать методы и реагирует на события.

* Классы объектов

Классы объектов являются шаблонами, определяющими наборы свойств, методы и событий. По этим шаблонам создаются объекты. Каждый из классов обладает специфическим набором свойств, методов и событий.

* Экземпляры классов

Объект, созданный по "шаблону" класса объектов, является экземпляром класса и наследует весь набор свойств, методов и событий данного класса. Каждый экземпляр класса имеет уникальное для данного класса имя. Экземпляры обладают одинаковым набором свойств, но значения свойств у них могут отличаться.

* Свойства объектов

Семейство объектов представляет собой объект, содержащий несколько объектов, экземпляров одного класса.

**Объекты**

|  |  |  |
| --- | --- | --- |
| Свойства Каждый объект обладает набором свойств. Значение свойств можно изменить в программном коде.        Объект. Свойство = Значение свойства | Методы Для того чтобы объект выполнил какую-либо операцию, необходимо применить метод, которым он обладает. Методы имеют аргументы, которые позволяют задать параметры выполняемых действий.    Объект. Метод арг 1: = зн, арг 2: = знач. | События Событие представляет собой действие, распознаваемое объектом. Событие может создаваться пользователем или быть результатом взаимодействия других программных объектов. В качестве реакции на событие вызывается определенная процедура. |

**Интегрированная среда Visual Basic**

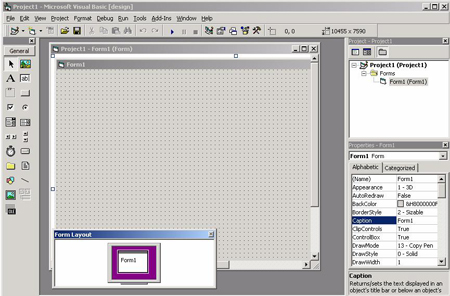
Система программирования Visual Basic состоит из текстового редактора для написания текста (кода) программ и конструктора форм. Программист пишет исходный код программы на одном из диалектов языка программирования Basic, выполняя разработку интерфейса программы с помощью конструктора форм. Для создания интерфейса Visual Basic предоставляет готовые объекты, обладающие определенными свойствами, которые можно изменять. Код программы и ее интерфейс объединяется в общий проект, который запускается на выполнение непосредственно из среды программирования Visual Basic или как исполняемый файл, компилируемый данной средой.  
Проект, созданный с помощью Visual Basic состоит из нескольких файлов:  
Основной файл проекта имеет расширение **.vbp** и содержит реестр файлов, необходимых для создания исполняемого файла приложения.  
Текстовый файл с расширением **.frm** сохраняет описание формы проекта и программный код, относящийся к ней. Таких файлов в проекте может быть несколько.  
Код, не связанный с конкретной формой или элементом управления, помещается в стандартный модуль с расширением **.bas**. Могут быть модули и другого типа.  
При вызове Visual Basic открывается окно диалога мастера проектов. Оно имеет три вкладки следующего назначения:  
**New** - создание нового проекта, предлагается на выбор несколько шаблонов, выбираем             STANDART.EXE.  
**Existing** - открыть существующий проект, появляется окно открытия файла.  
**Recen**t - открыть один из последних проектов.  
Окно Visual Basic содержит строку меню, панели инструментов и различные окна.  
Окно конструктора форм находится в центре экрана и содержит форму (**Form**). В новый проект сразу добавляется форма с именем *Form1*. Команда **Object** меню **View** открывает окно, если оно было закрыто и выводит его поверх других окон.  
Окно с набором управляющих элементов (**ToolBox**) находится обычно слева от формы. В процессе проектирования вы выбираете нужные элементы и перемещаете их на форму.  
Окно свойств (**Properties**) содержит список свойств и их значений для выделенного объекта. Объект можно выбрать также в списке под строкой заголовка.  
Окно проводника (**Project Explorer**) отображает все составные части проекта: формы и модули, не связанные с формами. Здесь же имеются кнопки для переключения между окном формы и редактором.  
Окно редактора кода (**Code**) содержит текст программы.  
Окно макета формы (**Form Layout**) показывает как будет выглядеть текущая форма на экране в режиме выполнения (см. Рис. 1)  


Рис. 1: Окно программы **Visual Basic**

Окна перемещаются по экрану за полосу заголовка. Любое из окон можно убрать с экрана и вернуть их обратно с помощью команд меню**View**или кнопок на панели инструментов.  
Строка меню Visual Basic содержит стандартные заголовки меню: **File**(Файл), **Edit** (Правка), **View** (Вид), **Window** (Окно), **Help** (Справка). Кроме того, имеются меню, обеспечивающие доступ к функциям программирования, например **Project** (Проект), **Tools** (Инструменты).

**Команды меню File**  
New Project - создать новый проект.  
Open Project - открыть существующий проект.  
Riove Project - закрыть текущий проект.  
Save Project - сохранить проект.  
Save Project As - сохранить проект под новым именем.  
Make <имя проекта>.exe - создать исполняемый файл текущего проекта.  
  
**Команды меню Edit (редактирование текста и формы)**  
Undo (Откат) - отменить последнюю операцию.  
Cut (Вырезать) - удалить выделенную область, скопировав ее в буфер обмена.  
Copy (Копировать) - копировать выделенную область в буфер обмена.  
Paste (Вставить) - вставить содержимое буфера обмена.  
Delete (Удалить) - удалить выделенную область или текущий элемент.  
Select All (Выделить все) - выделить все содержимое текущего окна.  
  
**Команды меню View**  
Code - открыть окно кода, относящегося к текущей форме или модулю.  
Object - открыть окно текущей формы.  
Project Explorer - открыть окно проводника проекта.  
Properties Window - открыть окно свойств.  
Form Layout Window - открыть окно макета формы.  
ToolBox - открыт окно коллекции инструментов.  
ColorPalette - открыть окно, позволяющее выбрать цвет текущего элемента или формы.

**2. Домашнее задание:**п.4.1,4.3,4.4 - читать