Національний технічний університет «Харківський політехнічний інститут»

Навчально-науковий інститут механічної інженерії і транспорту
Кафедра «Інтегровані технології машинобудування» ім. М.Ф. Семка

Третяк Т.Є.

КОНСПЕКТ ЛЕКЦІЙ

з дисципліни «Інформатика»

Харків

Семестр 1

Модуль 1. Аппаратные и программные средства компьютерной системы. Среда программирования Turbo Pascal. Среда объектно-ориентированного программирования Delphi.

Лекция 1. Аппаратно-программный комплекс на архитектурной платформе IBM PC. Общие понятия информатики.

Основные вопросы:

1. Аппаратные средства компьютерной системы (базовые, периферийные).
2. Программные средства компьютерной системы.
3. Понятие информации и данных, основные операции с данными.
4. Представление данных в компьютере.

1. Аппаратные средства компьютерной системы подразделяются на базовые и периферийные. К ***базовым аппаратным средствам*** относятся системный блок, монитор и клавиатура. К ***периферийным аппаратным средствам*** относятся манипуляторы, устройства ввода и вывода, мультимедийные устройства, дополнительные устройства хранения данных, коммуникационные устройства, блоки бесперебойного питания и др.

Системный блок является центральным блоком компьютера. Он включает ряд обязательных компонентов: центральный процессор (CPU), оперативную память (RAM), накопитель на жестком магнитном диске (HDD), видеоплату, материнскую плату, устройства чтения и перезаписи DVD-дисков и Blue ray-дисков и др.

2. ***Программные средства компьютерной системы*** подразделяются на 3 уровня:

- аппаратные (POST, BIOS);
- системные (операционные системы, драйверы устройств, сервисные программы для обслуживания дисков, антивирусы, файловые менеджеры, программы работы с сетями);
- прикладные (тектовые и графические процессоры, базы данных, алгоритмические языки программирования).

3. В информатике под ***информацией*** понимается сообщение, снижающее степень неопределенности знаний о состоянии предметов или явлений и помогающее решить поставленную задачу. Изменение некоторой физической величины во времени, обеспечивающее передачу сообщений, называется ***сигналом***. Зарегистрированные сигналы называются ***данными***.. Однако данные не тождественны информации. Для получения информации необходим ***метод обработки данных***. Т.о. ***информация*** – это продукт взаимодействия данных и адекватных им методов их обработки.

Над данными можно выполнять различные ***операции***, состав которых определяется решаемой задачей. Основными из низ являются:

- ***сбор данных*** – накопление данных с целью обеспечения достаточной их полноты для принятия решений;
- ***формализация данных*** – приведение данных, поступающих из разных источников, к единой форме, что позволяет сделать их сопоставимыми между собой;
- ***фильтрация данных*** – отсеивание данных, в которых нет необходимости для принятия решений, что повышает их достоверность и адекватность;
- ***сортировка данных*** – упорядочение данных по заданному признаку с целью удобства использования;

- ***защита данных*** – комплекс мер, направленных на предотвращение утраты, воспроизведения и модификации данных.
- ***архивация данных*** – организация хранения данных в удобной и легкодоступной форме, снижающей затраты на хранение и повышающей общую надежность информационного процесса.
- ***транспортировка данных*** – приём и передача данных между удаленными участниками информационного процесса.
- ***преобразование данных*** – перевод данных из одной формы или структуры в другую.

4. В информатике используется в основном три ***системы счисления***:

- двоичная (binary);
- десятичная (decimal);
- шестнадцатиричная (hexadecimal).

Для представления информации (как числовой, так и не числовой) в памяти компьютера используется ***двоичная система счисления***. Наименьшей единицей хранения информации является ***бит***, который может принимать значения 0 или 1. Биты организуются в более крупные образования – ***ячейки памяти***. Каждая ячейка памяти имеет свой ***адрес***. Минимальной адресуемой (пересылаемой между компонентами компьютера) единицей информации является ***байт***, состоящий, как правило, из 8 бит. Самый младший бит байта имеет номер 0 и располагается справа, самый старший бит байта имеет номер 7 и располагается слева. Два байта со смежными адресами образуют ***слово (word)*** разрядностью 16 бит.

Лекция 2. Основные сведения об операционных системах. Файловая структура хранения данных.

Основные вопросы:

1. Назначение и классификация операционных систем.

2. Понятие файла, виды файлов.

3. Назначение файловой системы хранения данных, понятие каталога.

1. ***Операционная система (ОС)*** – это программа, которая используется для управления компьютером, его ресурсами и запуска прикладных программ на выполнение. На устройствах хранения информации ОС организует файловую систему.

В зависимости от алгоритма управления процессором, операционные системы делятся на:

- однозадачные и многозадачные;
- однопользовательские и многопользовательские;
- однопроцессорные и многопроцессорные системы;
- локальные и сетевые.

2. ***Файл*** – это набор взаимосвязанных данных, под одним именем хранящихся на диске. Файлы бывают двух видов:

- текстовые;
- двоичные (бинарные).

Текстовые файлы предназначены для чтения пользователем, состоят из строк символов. В них хранятся тексты программ, командных файлов ОС.

В двоичных (бинарных) файлах используется машинное представление данных. К ним относятся выполняемые коды программ, драйверы устройств.

3. Операционная система управляет файлами с помощью файловой системы. ***Файловая система*** включает в себя систему каталогов и системы размещения файлов на диске, она определяет возможности манипулирования данными – создания, записи, чтения, поиска, модификации, удаления, восстановления удаленных файлов.

Имена файлов регистрируются в каталогах. ***Каталог*** – это специальное место на диске, в котором хранятся имена файлов, сведения об их размере, времени последнего обновления, атрибуты (свойства) файлов и др. На диске формируется многоуровневая древовидная структура каталогов.

Лекция 3. Особенности работы в среде программирования Turbo Pascal.

Основные вопросы:
1. Основные сведения о среде Turbo Pascal.
2. Этапы решения задачи в среде Turbo Pascal.

1. ***Язык Turbo Pascal*** является структурированным языком высокого уровня, на котором можно написать программу практически неограниченного размера и любого назначения.

Интегрированная среда программирования Turbo Pascal сочетает в себе возможности:

- редактора текстов;
- компилятора;
- отладчика.

Она поддерживает систему меню, оконный интерфейс, управление конфигурацией системы и контекстную систему подсказки. Turbo-среда загружается запуском файла ***turbo.exe***.

2. Любая задача, связанная с программированием, проходит ряд этапов. Можно выделить следующие ***этапы решения задачи*** в среде Turbo Pascal:

1) вызвать команду меню ***File/Open*** для чтения с диска и загрузки в окно редактора уже существующего файла программы или команду меню ***File/New*** для открытия нового файла;
2) набрать текст новой программы или отредактировать текст уже существующей программы;
3) вызвать команду меню ***File/Save*** для записи содержимого текущего окна редактора на диск в файл с тем же именем или команду меню ***File/Save as*** для записи содержимого окна редактора в файл с другим именем;
4) вызвать команду меню ***Compile/Compile*** для компиляции текста программы;
5) вызвать команду меню ***Run/Run*** для запуска программы на выполнение;
6) вызвать команду меню ***File/Exit*** для выхода из Turbo-среды.

Лекция 4. Алфавит и структура Pascal-программы.

Основные вопросы:
1. Алфавит языка Turbo Pascal.
2. Идентификаторы и зарезервированные слова языка.
3. Общая структура Pascal-программы.

1. ***Алфавит языка программирования*** - это набор символов, разрешенных к использованию и воспринимаемых компилятором.

Алфавит языка Turbo Pascal включает:

1) латинские строчные и прописные буквы (*A, B, … , Z* и *a, b, … , z*);
2) цифры от 0 до 9;

3) символ подчеркивания “_”;
4) символ “пробел”;
5) символы с кодами ASCII от 0 до 31 (управляющие коды);
6) специальные символы (+, -, *, /, =, <, >, [,], ., ,, (,), :, ;, ^, @, {, }, $, #, ’);
7) составные символы (<=, >=, <>, :=, (*, *), (., .), ..).

Символы с номерами от 128 до 255 из расширенного кода ASCII (в частности алфавит кириллицы) и некоторые другие символы из основного набора клавиатуры (!, % и др.) не входят в алфавит языка, но могут использоваться в тексте программы в виде значений символьных и строковых констант или в тексте комментариев.

2. ***Идентификаторы*** – это имена типов, переменных, констант, меток, процедур, функций и модулей. Имена конструируются из латинских букв, цифр и символов подчеркивания. Они могут состоять из любого числа перечисленных символов, но должны начинаться с буквы.

Turbo Pascal имеет большое количество ***зарезервированных слов*** (*begin, end, do, var* и др.). Эти слова не могут быть использованы в качесве идентификаторов, т.к. за ними закреплены строго определенные в языке понятия .

3. ***Общая структура Pascal-программы*** в соответстви со стандартом ANSI Pascal имеет следующий вид:

program имя программы;
uses
  список подключаемых модулей;
label
  список меток переходов в основном блоке программы;
const
  описание констант;
type
  описание пользовательских типов;
var
  описание переменных;
Описание пользовательских подпрограмм (процедур и функций);
begin
  основной блок (тело) программы
end.

При этом все блоки программы, кроме основного, могут отсутствовать.

Язык Turbo Pascal предоставляет большую гибкость в организаци текста программы, чем стандарт языка. В Turbo Pascal все блоки программы, кроме основного, могут отсутствовать; все блоки, кроме блока uses и основного блока, могут повторяться и располагаться в любом порядке, если выполняется условие: все блоки описаний данных должны располагаться выше блоков, в которых используются эти данные.

Лекция 5. Типы данных, константы и переменные языка Turbo Pascal.

Основные вопросы:
1. Система типов языка, простые и сложные типы.
2. Константы и переменные, их описание и использование.
3. Оператор присваивания и совместимость типов.
4. Арифметические выражения в языке Turbo Pascal.

1. Стандартные ***типы*** языка Turbo Pascal подразделяются на

- ***простые***, определяющие тип только одного отдельного значения;

- ***сложные***, определяющие некоторую структуру из компонентов простых типов.

Базовый набор простых типов включает:

1) числовые типы:
   - короткое целое без знака – ***byte*** (1 байт);
   - короткое целое со знаком – ***shortint*** (1 байт);
   - целое без знака – ***word*** (2 байта);
   - целое со знаком – ***integer*** (2 байта);
   - длинное целое со знаком – ***longint*** (4 байта);
   - вещественное – ***real*** (6 байт);
2) логический тип – ***boolean***;
3) символьный тип – ***char***;
4) строковый тип – ***string***, ***string[n]***;
5) адресный тип (указатель) – ***pointer***;
6) перечислимый тип;
7) ограниченный тип (диапазон).

Все эти типы могут учавствовать в определени сложных типов. ***Базовый набор сложных типов*** включает:

1) массив – ***array … of …***;
2) множество – ***set of …***;
3) файлы – ***text***, ***file***, ***file of …***;
4) запись – ***record***;
5) объект – ***object***;
6) ссылка – ^***базовый тип***.

2. ***Константы*** описываются в программе в блоке ***const*** следующим образом:

const
имя константы=значение (или значение выражения);

Константы могут принимать участие в выражениях для других констант. Значения констант нельзя изменять в теле программы.

Переменные описываются в программе в блоке ***var*** следующим образом:

var
имя переменной:имя типа;

Свои значения переменные получают в теле программы, и их можно изменять.

В языке имеется возможность объявлять переменные и тут же записывать в них стартовые значения. Такие переменные называются ***переменными со стартовыми значениями*** или ***типизированными константами***.

Типизированные константы описываются в программе в блоке ***const*** следующим образом:

const
имя типизированной константы: имя типа=значение (или значение выражения);

Типизированные константы не могут принимать участие в выражениях для других констант. Значения типизированных констант можно изменять в теле программы.

3. Описанные в программе переменные могут получать свои значения в теле программы с помощью операци присваивания. ***Операция присваивания*** имеет следующий вид:

имя переменной:=значение;

где составной символ ":=" называется ***оператором присваивания***.

Значением может быть:

- константа;
- переменная;
- вызов функции;
- арифметическое или логическое выражение.

Turbo Pascal, являясь языком с сильной системой типов, требует соблюдения определенных правил совместимости типов переменных и значений слева и справа от оператора присваивания. Типы считаются ***совместимыми по присваиванию***, если:

- тип переменной и тип значения совпадают;
- тип переменной является вещественным, а тип значения – целым;
- тип переменной является строковым, а тип значения – символьным.

4. ***Арифметическое выражение*** – это совокупность операндов, соединенных знаками арифметических операций.

В качестве ***операндов*** могут быть использованы следующие конструкци:

- целые и вещественные константы;
- целые и вещественные переменные;
- вызовы функций;
- другие арифметические выражения, заключенные в круглые скобки.

В языке Turbo Pascal предопределены следующие ***арифметические операции***:

- - - унарный минус (1 приоритет);
- + - унарный плюс (1 приоритет);
- * - умножение (2 приоритет);
- / - деление (2 приоритет);
- *div* – целочисленное деление (2 приоритет);
- *mod* – остаток от деления нацело (2 приоритет);
- + – сложение (3 приоритет);
- - - вычитание (3 приоритет).

Вычисление арифметических выражений производится в следующем порядке:

1) вычисляются аргументы функций, а затем сами функции;
2) выполняются операции 1 приоритета;
3) выполняются операции 2 приоритета;
4) выполняются операции 3 приоритета.

При этом выражения, заключенные в скобки, выполняются в первую очередь, операци одного приоритета выполняются слева направо.

Лекция 6. Управляющие структуры языка Turbo Pascal.

Основные вопросы:

1. Простой и составной операторы.

2. Операторы передачи управления:

а) оператор условного перехода IF…THEN…ELSE;

б) оператор безусловного перехода GOTO;

в) операторы безусловного выхода из программных блоков EXIT и HALT.

3. Логические выражения в языке Turbo Pascal.

4. Оператор варианта CASE.

5. Операторы цикла:

а)Оператор цикла с параметром FOR…DO.

б) Операторы цикла с предусловием WHILE…DO и постусловием REPEAT…UNTIL.

1. ***Простой оператор*** в программе – это единое неделимое предложение, выполняющее некоторое лействие. Два последовательных оператора разделяются символом “;”.

Если действие мыслится как единое, но реализуется несколькими различными операторами, то они могут быть представлены как ***составной оператор*** – последовательность операторов, объединенная операторными скобками *begin … end*.

2. Для программной реализации разветвляющихся вычислительных процессов в языке Turbo Pascal предусмотрены специальные ***операторы передачи управления***:

а) ***оператор условного перехода*** имеет следующий вид:

if условие ***then*** оператор 1 ***else*** оператор 2;

Условием может быть:

- логическая константа;
- логическая переменная;
- логическое выражение.

Оператор 1 выполняется, если значение условия истинно, а оператор 2 - если условие ложно. Операторы 1 и 2 могут быть простыми или составными. Альтернатива ***else*** может отсутствовать.

б) ***оператор безусловного перехода*** имеет следующий вид:

goto метка;

Оператор передает управление к оператору, помеченному данной меткой. Меткой может быть описанный в блоке ***label*** программы идентификатор или число от 0 до 9999. Метка от оператора должна отделяться символом “:”.

в) ***операторы безусловного выхода из программных блоков*** имеют следующий вид:

exit; - оператор завершает работу программного блока (основной программы, процедуры или функции), в котором он вызывается;

halt; - оператор завершает работу основной программы независимо от того, в каком программном блоке (основной программе, процедуре или функции) он вызывается.

3. ***Логические выражение*** – это совокупность операндов, соединенных знаками логических операций.

В качестве ***операндов*** могут быть использованы следующие конструкци:

- логические константы;
- логические переменные;
- логические отношения - два операнда, соединенные знаками операций отношения (<, <=, >, >=, =, <>);
- другие логические выражения, заключенные в круглые скобки.

В языке Turbo Pascal предопределены следующие ***логические операции***:

- *not* – логическое “не” (отрицание) (1 приоритет);
- *and* - логическое “и” (2 приоритет);
- *or* - логическое “или” (3 приоритет);
- *xor* - логическое исключающее“или” (3 приоритет).

Вычисление логических выражений производится в следующем порядке:

1) вычисляются операнды логических отношений, а затем сами логические отношения;

2) выполняются операции 1 приоритета;
3) выполняются операции 2 приоритета;
4) выполняются операции 3 приоритета.

При этом выражения, заключенные в скобки, выполняются в первую очередь, операци одного приоритета выполняются слева направо.

4. ***Оператор варианта*** необходим в тех случаях, когда в зависимости от значений некоторой переменной (или выражения) необходимо выполнить те или иные операторы. Он имеет следующий вид:

case управляющая переменная (или выражение) ***of***
набор значений11: оператор 1;
набор значений 2: оператор 2;
……
набор значений N: оператор N
[***else*** альтернативный оператор]
end;

Тип управляющей переменной (или выражения) может быть только одним из перечислимых типов. Набор значений – это конктетные значения управляющей переменной (или выражения), при которых необходимо выполнить соответствующий оператор, игнорируя остальные варианты.

Если значение управляющей переменной (или выражения) не совпадает ни с одним из значений в вариантах, то выполняется альтернативный оператор. Операторы 1, 2, … , N могут быть простыми или составными. Альтернатива ***else*** может отсутствовать.

5. Для программной реализации циклических вычислительных процессов в языке Turbo Pascal предусмотрены специальные ***операторы цикла***:

а) ***оператор цикла с параметром*** используется для организации циклов с заранее известным числом повторений и имеет следующий вид:

for параметр цикла:=младшее значение ***to*** старшее значение ***do*** оператор;

или

for параметр цикла:=старшее значение ***downto*** младшее значение ***do*** оператор;

Тип параметра цикла может быть только одним из перечислимых типов.

Оператор, представляющий собой тело цикла, выполняется циклически до тех пор, пока параметр цикла не превзойдет (в цикле с ***to***) или не станет меньше (в цикле с ***downto***) своего конечного значения. Оператор может быть простым или составным.

б) ***оператор цикла с предусловием*** используется для организации циклов с заранее неизвестным числом повторений и имеет следующий вид:

while условие ***do*** оператор;

Условием может быть:

- логическая константа;
- логическая переменная;
- логическое выражение.

Оператор, представляющий собой тело цикла, выполняется циклически до тех пор, пока значение условия истинно. Оператор может быть простым или составным;

оператор цикла с постусловием используется для организации циклов с заранее неизвестным числом повторений и имеет следующий вид:

repeat
оператор 1;
оператор 2;
……
оператор N
until условие;

Условием может быть:

- логическая константа;
- логическая переменная;
- логическое выражение.

Набор операторов 1, 2, … , N, представляющий собой тело цикла, выполняется циклически до тех пор, пока значение условия ложно.

Лекция 7. Сложный тип массив языка Turbo Pascal.

Основные вопросы:
1. Тип массив и переменная типа массив.
2. Операции, допустимые над массивами.

1. ***Массив*** – это упрядоченная последовательность однотипных данных (компонентов), обозначенных одним идентификатором.

Компоненты этой последовательности называются ***элементами*** массива. Массив имеет определенный ***размер***, определяющий, сколько элементов в нем хранится. Обратиться к отдельному элементу массива можно с помощью его ***индекса*** (***индексов***).

Сложный ***тип массив*** описывается в программе в блоке ***type*** следующим образом:

type
имя типа массива=***array*** [диапазон(ы) индексов] ***of*** тип элементов;

Диапазон(ы) индексов указывает(ют) значения индексов первого и последнего элементов массива. Тип индексов может быть только одним из перечислимых типов (кроме *longint*). Тип элементов может быть одним из простых типов, массивом (или записью.

Переменная типа массив описывается в программе в блоке ***var*** стандартным образом. Доступ к отдельному элементу переменной типа масив осуществляется по имени элемента, которое состоит из имени переменной и индекса(ов) этого элемента, заключенного(ых) в квадратные скобки.

2. Над совместимыми (описанными в одной группе переменных или одним именем типа) массивами из элементов любых типов допустима ***операция присваивания***, которая копирует поэлементно один массив в другой.

Над массивами из элементов символьного типа допустимы также ***операции объединения (конкатенации)*** и ***сравнения***.

Лекция 8. Ввод и вывод данных и файловая система в Turbo Pascal.

Основные вопросы:
1. Понятие логического и физического файла, файловые типы языка Turbo Pascal.
2. Общие процедуры для работы с файлами.
3. Операторы ввода и вывода данных в текстовые файлы.

1. ***Физический файл*** – это последовательность данных, расположенных вне рабочей памяти программы (на диске, на экране компьютера, на принтере, в другом компьютере, в компьютерной сети). Физический файл в программе определяется строкой с его именем.

Логический файл – это переменная одного из файловых типов, определенных в языке. Turbo Pascal поддерживает 3 ***файловых типа***:

1) ***text*** – текстовые файлы;
2) ***file*** – типизированные файлы;
3) ***file of …*** - бестиповые файлы.

После объявления в программе переменной файлового типа она может служить средством доступа к любым физическим файлам.

2. В языке Turbo Pascal определены ***общие процедуры для работы с файлами*** любых типов:

- ***assign***(логический файл, имя физического файла); - связывает логический файл с конкретным физическим файлом;
- ***reset***(логический файл); - открывает логический файл для чтения;
- ***rewrite***(логический файл); - открывает логический файл для записи;
- ***close***(логический файл); - закрывает открытый логический файл;

и др.

3. ***Ввод данных*** в программу из текстового файла осуществляется операторами:

- ***read***(логический файл, список ввода); - считывает данные в текущей строке;
- ***readln***(логический файл, список ввода); - считывает данные в текущей строке и переводит позицию считывания в начало следующей строки, даже если в теущей строке остались непрочитанные данные.

Вывод данных в текстовый файл осуществляется операторами:

- ***write***(логический файл, список вывода); - выводит данные в текущую строку, не закрывая ее;
- ***writeln***(логический файл, список вывода); - выводит данные в текущую строку и закрывает ее.

Лекция 9. Процедуры и функции в языке Turbo Pascal.

Основные вопросы:
1. Назначение процедур и функций, особенности их структуры и вызова.
2. Глобальные и локальные данные.
3. Формальные и фактические параметры процедур и функций.
4. Категории формальных параметров процедур и функций.
5. Включаемые процедуры и функции.

1. В языке Turbo Pascal имеется возможность создания пользовательских подпрограмм (процедур и функций), входящих в состав основной программы и работающих под ее управлением.

Процедура – это подпрограмма, предназначенная для выполнения некоторой законченной последовательности действий. Процедура имеет следующий вид:

procedure имя процедуры (список входных и выходных формальных параметров);
label
 список локальных меток переходов в теле процедуры;
const
 описание локальных констант;
type

описание локальных пользовательских типов;
var
описание локальных переменных;
Описание вложенных пользовательских подпрограмм (процедур и функций);
begin
тело процедуры
end;

Заголовок процедуры обязателен. Допускается описание процедуры без параметров.

Вызов процедуры в основной программе или другой подпрограмме имеет следующий вид:

имя процедуры (список входных и выходных фактических параметров);

Функция – это подпрограмма, предназначенная для вычисления некоторого значения, присваиваемого имени функции. Функция имеет следующий вид:

function имя функции (список входных формальных параметров):тип значения функции;
label
список локальных меток переходов в теле функции;
const
описание локальных констант;
type
описание локальных пользовательских типов;
var
описание локальных переменных;
Описание вложенных пользовательских подпрограмм (процедур и функций);
begin
операторы тела функции;
……
имя функции:=значение;
……
операторы тела функции;
end;

Вызов процедуры в основной программе или другой подпрограмме в составе некоторого выражения имеет следующий вид:

имя функци (список входных фактических параметров);

2. ***Глобальные данные*** (метки, константы, типы, переменные) объявляются в соответствующих блоках основной программы. Они доступны как основной программе, так и подпрограммам, входящим в ее состав.

Локальные данные (метки, константы, типы, переменные) объявляются в соответствующих блоках подпрограмм и доступны только им.

3. ***Формальные параметры*** процедур и функций указывают, с какими параметрами следует обращаться к подпрограмме (их тип, количество и позиция в списке).

Фактические параметры процедур и функций замещают все вхождения в подпрограмму формальных параметров и именно они фактически участвуют в работе подпрограммы пр ее вызове.

4. Язык Turbo Pascal допускает использование 4-х ***категорий формальных параметров*** процедур и функций:

- параметров-значений;
- параметров-переменных;
- параметров-констант;
- параметров-процедур и параметров-функций.

5. В языке Turbo Pascal имеется возможность включать в текст программы на этапе ее компоновки уже отлаженные фрагменты, хранящиеся во внешних дисковых файлах. Для этой цели служит специальная директива компиляции, имеющая следующий вид:

{$I имя вклюмаемого файла}.

Эта директива замещает в программе включаемый внешний фрагмент. Наиболее целесообразно оформлять включаемые фрагменты в виде ***включаемых процедур и функций***.

Лекция 10. Модули в языке Turbo Pascal.

Основные вопросы:

1. Назначение и особенности модулей.

2. Структура модулей.

1. В языке Turbo Pascal имеется возможность создания пользовательских модулей.

Модуль – это специальным образом оформленная библиотека определений пользовательских констант, типов, переменных, процедур и функций, которые могут использоваться в основной программе при подключении к ней модуля в блоке ***uses***.

Модуль, в отличие от программы, не может быть запущен на выполнение самостоятельно, он может только участвовать в построени программы или другого модуля. Перед использованием модуль компилируется на диск независимо от основной программы. Имя модуля, указываемое в его заголовке, должно соответствовать имени файла, в котором хранится текст модуля.

2. Модуль имеет следующий вид:

unit имя модуля;
interface {начало раздела объявлений}
uses
  список используемых при объявлениях модулей;
const
  описание библиотечных констант;
type
  описание библиотечных типов;
var
  описание библиотечных переменных;
Заголовки библиотечных подпрограмм (процедур и функций);
implementation {начало раздела реализации}
uses
  список используемых при реализации модулей;
label
  список меток переходов в блоке инициализации модуля;
const
  описание внутренних констант;
type
  описание внутренних типов;

var
описание внутренних переменных;
Описание внутренних подпрограмм (процедур и функций);
Описание библиотечных подпрограмм (процедур и функций);
begin
блок (раздел) инициализаци модуля
end.

Заголовок модуля обязателен. Все блоки внутри разделов объявлений и реализации модуля могут отсутствовать или, за исключением блока ***uses***, повторяться. Блок инициализаци также может отсутствовать, но слово ***end***. должно сохраняться.

Лекция 11. Особенности работы в среде программирования Delphi.

Основные вопросы:
1. Основные характеристики Delphi.
2. Элементы экрана Delphi.
3. Понятие объекта и события в Delphi.

1. ***Среда программирования Delphi*** является одной из ведущих систем программирования, используемых для разработки современных прикладных программных продуктов, и в первую очередь приложений Windows.

Систему Delphi относят к категории RAD-систем программирования (RAD – Rapid Application Development - быстрая разработка приложений). В основе таких систем лежит технология визуального проектирования и событийного программирования.

Система Delphi базируется на использовани языка программирования Object Pascal, который является логическим продолжением и развитием классического языка программирования Pascal. Среда загружается запуском файла ***delphi32.exe***. Программа, находящаяся в Delphi в стадии разработки, называется ***проектом***.

2. Экран Delphi помимо стандартных элементов окна Windows включает:

- ***палитру компонентов*** - содержит большое количество вкладок с заготовками объектов приложения;
- ***окно формы*** – является заготовкой окна приложения, на котором размещаются объекты;
- ***окно списка объектов*** – содержит список используемых объектов в виде древовидной структуры;
- ***окно инспектора объектов*** – содержит вкладки для настройки свойств объектов и закрепления за объектами событий, на которые они должны реагировать;
- ***окно кода*** – содержит вкладку со сформированным средой Delphi шаблоном программного кода приложения, в котором пользователь программирует в виде отдельных процедур реакцию объектов на закрепленные за ними события.

3. ***Объекты*** приложения - это форма приложения и элементы, расположенные на ней. Каждый тип объектов обладает определенным набором ***свойств***. Свойства объекта могут изменяться в ходе работы приложения при воздействи пользователя на объект. Такие действия, производимые пользователем в отношении объекта, называются ***событиями***. Для каждого типа объектов существует определенный набор событий, на которые он может реагировать. Реакция объекта на закрепленные за ним события программируется пользователем в виде отдельных процедур (***обработчиков событий***).

Лекция 12. Разработка первого проекта Delphi.

Основные вопросы:

1. Создание стартовой формы приложения Delphi, основные компоненты формы.

2. Базовый набор событий Delphi.

3. Понятие процедуры обработки события.

4. Создание программного кода приложения.

1. При разработке проекта Delphi система создает группу связанных между собой программных файлов. Поэтому рекомендуется каждый вновь создаваемый пользователем проект сохранять в отдельной папке.

Разработку проекта желательно начинать с его сохранения. Предварительно создав для проекта новую папку, необходимо вызвать команду меню ***File/Save as*** для сохранения файла модуля формы ***unit1.pas***, а затем команду меню ***File/Save Project as*** для сохранения главного файла описания проекта ***project1.dpr***. Остальные связанные с проектом файлы сохраняются автоматически.

Разработка проекта Delphi включает в себя два взаимосвязанных компонента:

- создание стартовой формы приложения;
- создание программного кода приложения.

Создание стартовой формы приложения включает:

- перенос с палитры компонентов на заготовку окна приложения заготовок необходимых объектов приложения (компонентов формы);
  при этом заготовки ***основных компонентов формы*** размещенны на вкладке ***Standard (Стандартные)*** палитры компонентов, ими являются:
  - ***Label (Надпись)***;
  - ***Edit (Текстовое поле)***;
  - ***Button (Командная кнопка)***;
    и др.
- настройку с помощью вкладки ***Properties (Свойства)*** окна инспектора объектов свойств формы и свойств объектов, размещенных на ней;
- закрепление с помощью вкладки ***Events (События)*** окна инспектора объектов за объектами событий, на которые они должны реагировать.

2. Для каждого типа объектов существует определенный набор событий, на которые он может реагировать. В Delphi каждому событию присвоено имя. ***Базовый набор событий Delphi*** включает следующие события:

- ***OnClick*** – происходит при щелчке кнопкой мыши;
- ***OnDblClick*** – происходит при двойномщелчке кнопкой мыши;
- ***OnMouseMove*** – происходит при перемещении мыши;
- ***OnKeyPress*** – происходит при нажатии клавиши клавиатуры;
- ***OnCreate*** – происходит при создани объекта;
  и др.

3. Реакцией объекта на закрепленное за ним событие должно быть некоторое действие. Это действие программируется пользователем в окне кода в виде отдельной ***процедуры обработки события.*** При запуске приложения эта процедура выполняется в момент совершения события над объектом.

4. Среда Delphi автоматически формирует шаблон программного кода приложения, добавляя в него шаблоны процедур обработки предусмотренных пользователем событий над объектами приложения. Поэтому для пользователя с***оздание программного кода приложения*** заключается лишь в дополнении шаблонов этих процедур.

Лекция 13. Структура проекта Delphi.

Основные вопросы:
1. Набор модулей, составляющих проект Delphi.
2. Назначение и структура главного модуля проекта.
3. Назначение и структура модуля формы проекта.

1. ***Проект Delphi*** представляет собой группу связанных между собой программных модулей, используя которые компилятор создает выполняемый код приложения (файл ***project1.exe***). В простейшем случае проект состоит из следующих ***модулей***:

- главного файла описания проекта ***project1.dpr;***
- файла модуля формы ***unit1.pas***;
- файла ресурсов ***project1.res***;
- файла описания формы ***unit1.pas***;
  и др.

2. ***Главный модуль проекта*** (главный файл описания проекта) ***project1.dpr*** содержит инструкции, с которых начинается выполнение приложения. Он полностью формируется средой Delphi и имеет следующий вид:

```
program Project1;
uses
  Forms,
  Unit1 in 'Unit1.pas' {Form1};
{$R *.res}
begin
  Application.Initialize;
  Application.CreateForm (TForm1, Form1);
  Application.Run;
end.
```

В тексте модуля
- директива компиляциии {$R *.res} подключает файл ресурсов ***project1.res***, который содержит ресурсы приложения (пиктограммы, курсоры, битовые образы и др.);
- метод ***Initialize*** запускает инициализацию приложения;
- метод ***CreateForm*** инициализирует форму приложения;
- метод ***Run*** запускает приложение на выполнение.

3. ***Модуль формы проекта unit1.pas*** содержит описание формы приложения и поддерживающих ее работу процедур. Его шаблон формируется средой Delphi при создани пользователем нового проекта и имеет следующий вид:

```
unit Unit1;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls;
type
  TForm1 = class(TForm)
  private
    { Private declarations }
  public
    { Public declarations }
  end;
```

```
var
  Form1: TForm1;
implementation
{$R *.dfm}
end.
```

В тексте модуля

- в блоке ***type*** описывается тип-класс формы TForm1 на базе родительского класса TForm;
- директива компиляци {$R *.dfm} подключает файл описания формы ***unit1.dfm***, который формируется средой Delphi на основе внешнего вида формы и содержит описание свойств формы и свойств объектов, расположенных на ней.

В процессе разработки пользователем проекта в блоке ***type*** в описание типа-класса формы средой Delphi будут добавлены описание объектов, расположенных на форме, и заголовки процедур обработки предусмотренных пользователем событий над объектами приложения, а в разделе ***implementation*** - шаблоны этих процедур, дополняемые пользователем.

Лекция 14. Сохранение проекта Delphi. Особенности компиляции и выполнения программы.

Основные вопросы:

1. Сохранение и компиляция проекта Delphi.

2. Сообщения об ошибках, предупреждения и подсказки на этапе компиляции программы.

3. Сообщения об ошибках времени выполнения и исключениях во время выполнения программы.

1. Каждый новый проект рекомендуется сохранять в предварительно созданной отдельной папке. Для ***сохранения проекта*** необходимо вызвать команду меню ***File/Save as*** для сохранения файла модуля формы ***unit1.pas***, а затем команду меню ***File/Save as*** для сохранения главного файла описания проекта ***project1.dpr***. Остальные связанные с проектом файлы сохраняются автоматически.

Для ***компиляции проекта*** после его сохранения необходимо вызвать команду меню ***Project/Compile Project***. Процесс компиляции состоит из двух этапов. На первом этапе выполняется проверка текста программы на отсутствие ошибок, на втором – генерируется выполнимый код программы (файл ***project1.exe***).

2. Процесс и результат компиляции проекта отражаются в диалоговом окне ***Compiling***. В это окно компилятор выводит количество ошибок, предупреждений и подсказок. Сами сообщения об ошибках, предупреждения и подсказки находятся в нижней части окна редактора кода.

При обнаружении в программе ***ошибок (Errors)*** на первый план выводится окно файла, в котором находится ошибка, указанная в первом сообщени об ошибках (обычно это файл ***unit1.pas***). После исправления ошибки выполняется повторная компиляция.

При обнаружении в программе неточностей, которые не являются ошибками, компилятор выводит ***предупреждения (Warnings)*** и ***подсказки (Hints)***.

3. Для ***запуска проекта на выполнение*** после его компиляци необходимо загрузить файл ***project1.exe*** или в среде Delphi вызвать команду меню ***Run/Run***. Во время выполнения программы могут возникать ***ошибки времени выполнения (run-time errors),*** или ***исключения(exceptions)***. В большинстве случаев их причиной являются неверные исходные данные.

Модуль 2. Среда объектно-ориентированного программирования Delphi. Текстовый процессор Microsoft Word. Табличный процессор Microsoft Excel.

Лекция 1. Особенности языки программирования Object Pascal.

Основные вопросы:
1. Типы данных языка Object Pascal.
2. Функции преобразования типов данных.
3. Управляющие структуры языка Object Pascal.

1. ***Базовый набор простых типов*** языка Object Pascal включает:

1) числовые типы:
   - короткое целое без знака – ***byte*** (1 байт);
   - короткое целое со знаком – ***shortint*** (1 байт);
   - целое без знака – ***word*** (2 байта);
   - целое со знаком – ***smallint*** (2 байта);
   - длинное целое без знака – ***longword*** (4 байта);
   - длинное целое со знаком – ***longint (=integer)***(4 байта);
   - 64 битное целое - ***int64*** (8 байт);
   - 48-битное вещественное - ***real48*** (6 байт);
   - вещественное одинарной точности – ***single*** (4 байта);
   - вещественное двойной точности – ***double (=real)*** (8 байт);
   - вещественное расширенной точности – ***extended*** (10 байт);
   - вещественное компактное – ***comp*** (8 байт);
   - вещественное денежное – ***currency*** (8 байт);
2) логические типы:
   - ***boolean*** (1 байт);
   - ***bytebool*** (1 байт);
   - ***wordbool*** (2 байта);
   - ***longbool*** (4 байта);
3) символьные типы:
   - ***ansichar (=char)***;
   - ***widechar***;
4) строковые типы:
   - ***shortstring;***
   - ***ansistring (=string)***;
   - ***wideistring***;
5) адресный тип (указатель) – ***pointer***;
6) перечислимый тип;
7) ограниченный тип (диапазон);
8) тип-вариант - ***variant***.

Базовый набор сложных типов включает:

7) массив – ***array … of …***;
8) множество – ***set of …***;
9) файлы – ***textfile***, ***file***, ***file of …***;
10) запись – ***record***;
11) объект – ***object***;
12) ссылка – ^***базовый тип;***
13) тип-класс - ***class***.

2. Функции преобразования типов данных наиболее часто используются в операторах ввода и вывода информации. В языке Object Pascal предопределены следующие ***функции преобразования типов***:

- ***IntToStr*** (*n*) – переводит целое число *n* в его строковое изображение;
- ***FloatToStr*** (*n*) – переводит вещественное число *n* в его строковое изображение;
- ***FloatToStrF*** (*n, f, k, m*) – переводит вещественное число *n* в его строковое изображение с учетом выбранного способа изображения; при этом
  f – формат изображения (например, *ffFixed* – формат числа с фиксированной точкой);
  k – общее количество цифр в числе;
  m - количество цифр в числе после запятой;
- ***StrToInt*** (*s*) – возвращает целое число, изображением которого является строка *s*;
- ***StrToFloat*** (*s*) – возвращает вещественное число, изображением которого является строка *s*.

3. ***Управляющие структуры*** языка Object Pascal (оператор условного перехода ***if…then…else***, оператор безусловного перехода ***goto***, операторы безусловного выхода из программных блоков ***exit*** и ***halt***, оператор варианта ***case***, оператор цикла с параметром ***for…do***, операторы цикла с предусловием ***while…do*** и постусловием ***repeat…until***) без изменений наследуются из языка Turbo Pascal.

Лекция 2. Особенности разработки программ с разветвляющейся структурой в среде Delphi.

Основные вопросы:

1. Способы добавления на форму Delphi списка возможных вариантов.

2. Способы формирования элементов списка возможных вариантов на форме Delphi.

1. Для ***добавления*** на форму Delphi ***списка возможных вариантов*** на вкладке ***Standard (Стандартные)*** палитры компонентов необходимо выбрать заготовку одного из объектов:

- ***ListBox (Список)***;
- ***ComboBox (Комбинированный список)***;
- ***Radiogroup (Группа радиокнопок)***.

2. ***Элементы списка*** могут быть ***сформированы*** двумя способами:

1) во время настройки формы (в том случае, если содержимое списка известно до запуска приложения на выполнение);
2) во время работы приложения (в том случае, если содержимое списка зависит от хода выполнения приложения).

Для формирования элементов списка ***во время настройки формы*** необходимо в окне инспектора объектов выбрать свойство ***Items*** объекта, отвечающего за список, и в редакторе списка строк набрать элементы списка в отдельных строках.

Для формирования элементов списка ***во время работы приложения*** необходимо в тексте программы вызывать метод ***Add*** свойства ***Items*** объекта, отвечающего за список, столько раз, сколко элементов должно присутствовать в списке. В качестве параметра этого метода задается отдельная добавляемая строка списка.

Во время работы приложения номер выбранного пользователем элемента списка возвращает свойство ***ItemIndex*** объекта, отвечающего за список.

Лекция 3. Особенности разработки программ с циклической структурой в среде Delphi.

Основные вопросы:

1. Добавление на форму Delphi поля для ввода или вывода двумерного массива.

2. Настройка свойств сетки Delphi.

3. Добавление на форму Delphi поля для ввода или вывода одномерного массива с элементами срокового типа.

1. Для ***добавления*** на форму Delphi ***поля для ввода или вывода двумерного массива*** на вкладке ***Additional (Дополнительные)*** палитры компонентов необходимо выбрать заготовку объекта ***StringGrid (Сетка)***. Содержимое ячеек сетки станет доступным только во время работы приложения, поэтому во время настройки формы оно не формируется.

2. При использовани объекта ***StringGrid (Сетка)***, как правило, требуется ***настройка*** его следующих ***свойств*** с помощью:

- ***ColCount*** – количество столбцов сетки;
- ***RowCount*** - количество строк сетки;
- ***FixedCols*** – количество зафиксированных слева столбцов сетки (они выделяются цветом и при горизонтальной прокрутке сетки остаются на месте);
- ***FixedRows*** - количество зафиксированных сверху строк сетки (они выделяются цветом и при вертикальной прокрутке сетки остаются на месте);
- ***DefaultColWidth*** – ширина столбцов сетки;
- ***DefaultRowHeight*** – высота строк сетки;
- ***Width*** - ширина сетки;
- ***Height*** - высота сетки;
- ***Options.goEditing*** – признак допустимости редактирования содержимого ячеек сетки;
- ***Options.goTab*** - признак допустимости использования клавиши [*Tab*] для перемещения курсора в следующую ячейку сетки.

В тексте программы можно обращаться к конкретной ячейке сетки с помощью свойства ***Cells*** (соответствующего сетке двумерного массива), указав после его имени в квадратных скобках номер столбца и номер строки ячейки. Свойство ***Cells*** отсутствует в окне инспектора объектов, т.к. имеет смысл только в тексте программы.

3. Для ***добавления*** на форму Delphi ***поля для ввода или вывода одномерного массива с элементами срокового типа*** на вкладке ***Standard (Стандартные)*** палитры компонентов необходимо выбрать заготовку объекта ***Memo (Список)***.

Элементы списка могут быть ***сформированы*** тремя способами:

1) во время настройки формы заполнением списка непосредственно на ней;
2) во время настройки формы выбором в окне инспектора объектов свойства ***Lines*** объекта, отвечающего за список, и формированием его в редакторе списка строк;
3) во время работы приложения вызовом метода ***Add*** свойства ***Lines*** объекта, отвечающего за список, столько раз, сколко элементов должно присутствовать в списке (в качестве параметра этого метода задается отдельная добавляемая строка списка).

В тексте программы можно обращаться к конкретному элементу списка с помощью свойства ***Lines***, указав после его имени в квадратных скобках номер элемента.

Лекция 4. Графические возможности Delphi.

Основные вопросы:

1. Особенности вычерчивания на холсте Delphi линий и контуров.
2. Особенности заливки на холсте Delphi областей, ограниченных контурами.
3. Вывод текста на холсто Delphi.
4. Вычерчивание прямой и ломанной линий на холсте Delphi.
5. Вычерчивание эллипса, окружности, их дуг или секторов на холсте Delphi.
6. Вычерчивание прямоугольника и многоугольника на холсте Delphi.
7. Заливка замкнутых областей на холсте Delphi.
8. Окрашивание точки на холсте Delphi.

1. Среда Delphi позволяет разрабатывать приложения, выводящие графику на поверхность объекта (формы или компонента ***Image***). ***Поверхности объекта*** соответсвует свойство ***Canvas*** – абстрактный ***холст***, состоящий из ***пикселов***. Положение пиксела характеризуется его горизонтальной (*x*) и вертикальной (*y*) координатами, представляющими собой значения целого типа.

Методы холста обеспечивают вывод графических примитивов на его поверхность. Эти методы используют внутренние ***свойства холста - Pen (Карандаш)*** и ***Brush (Кисть)***, которые позволяют задавать характеристики выводимых графических примитивов.

Вид линий и контуров, которые вычерчиваются на холсте, определяют следующие внутренние ***свойства карандаша***:

- ***Color (Цвет линии)*** - может принимать значение одной из именованных констант, предопределенных в Delphi для обозначения цветов;
- ***Width (Толщина линии)*** - задается количеством пикселов;
- ***Style (Стиль линии)*** - может принимать значение одной из именованных констант, предопределенных в Delphi для обозначения стилей линии:

  psSolid – сплошная линия;

  psDash – пунктирная линия из длинных штрихов;

  psDot – пунктирная линия из коротких штрихов;

  psDashDot – пунктирная линия из чередующихся длинного и короткого штрихов;

  psDashDotDot – пунктирная линия из чередующихся одного длинного и двух коротких штрихов;

  psClear – линия не отображается;
- ***Mode (Режим отображения линии)*** - может принимать значение одной из именованных констант, предопределенных в Delphi для обозначения режимов отображения линии:

  pmBlack – черный цвет линии независимо от значения свойства ***Color***;

  pmWhite – белый цвет линии независимо от значения свойства ***Color***;

  pmCopy – цвет линии определяется значением свойства ***Color***;

  pmNotCopy – цвет линии является инверсным по отношению к цвету, определяемому значением свойства ***Color***;

  pmNot – цвет точки линии является инверсным по отношению к цвету точки холста, в которую выводится точка лини.

2. ***Вид заливки областей, ограниченных контурами,*** на холсте определяют следующие внутренние ***свойства кисти***:

- ***Color (Цвет заливки)*** - может принимать значение одной из именованных констант, предопределенных в Delphi для обозначения цветов;

- ***Style (Стиль заливки)*** - может принимать значение одной из именованных констант, предопределенных в Delphi для обозначения стилей заливки:
  bsSolid – сплошная заливка;
  bsClear – заливка не отображается;
  bsHorizontal – горизонтальная штриховка;
  bsVertical – вертикальная штриховка;
  bsFDiagonal – диагональная штриховка с наклоном линий вперед;
  bsBDiagonal – диагональная штриховка с наклоном линий назад;
  bsCross – горизонтально-вертикальная штриховка в клетку;
  bsDiagCross – диагональная штриховка в клетку.

3. Для ***вывода текста*** на холст используется метод

TextOut (*x, y, Текст*);

где *x, y* – координаты точки холста, от которой выполняется вывод текста;

Текст – переменная или константа строкового типа, значение которой выводится.

Этот метод использует внутреннее ***свойство холста*** – ***Font (Шрифт)***, которое позволяет задавать характеристики шрифта, используемого при выводе текста.

Характеристики шрифта определяют следующие внутренние ***свойства шрифта***:

- ***Name (Название шрифта)*** - может принимать значение одного из названий шрифтов, предопределенных в Windows;
- ***Size (Размер шрифта)*** - задается количеством пунктов;
- ***Style (Стиль шрифта)*** - может принимать значение одной из именованных констант, предопределенных в Delphi для обозначения стилей шрифта:
  fsBold – полужирный шрифт;
  fsItalic – курсив;
  fsUnderline – подчеркнутый шрифт;
  fsStrikeOut – перечеркнутый шрифт;

  это свойство является множеством, может заключаться в квадратные скобки и состоять из нескольких разделенных запятыми значений приведенных выше именованных констант, что позволяет комбинировать необходимые стили;
- ***Color (Цвет символов)*** - может принимать значение одной из именованных констант, предопределенных в Delphi для обозначения цветов.

4. Для ***вычерчивания прямой линии*** на холсте используется метод

LineTo (*x, y*);

где *x, y* – координаты точки холста, до которой вычерчивается линия от текущей позиции карандаша.

Перемещение позиции карандаша в необходимую точку выполняет метод

MoveTo (*x, y*);

где – координаты новой позиции карандаша.

Для ***вычерчивания ломаной линии*** на холсте используется метод

PolyLine (*Массив координат точек*);

где *Массив координат точек* – переменная типа массив записей типа ***TPoint***, содержащих координаты *x* и *y* точек, которые соединяет ломаная линия.

5. Для ***вычерчивания эллипса или окружности*** на холсте используется метод

Ellipse (*x1, y1, x2, y2*);

где *x1, y1* – координаты левого верхнего угла прямоугольника (или квадрата), внутри которого вычерчивается эллипс (или окружность);

x2, *y2* – координаты правого нижнего угла прямоугольника (или квадрата), внутри которого вычерчивается эллипс (или окружность).

Для ***вычерчивания дуг эллипса или окружности*** на холсте используется метод

Arc (*x1*, *y1*, *x2*, *y2*, *x3*, *y3*, *x4*, *y4*);

где *x1*, *y1* – координаты левого верхнего угла прямоугольника (или квадрата), внутри которого вычерчивается дуга эллипса (или окружности);

x2, *y2* – координаты правого нижнего угла прямоугольника (или квадрата), внутри которого вычерчивается дуга эллипса (или окружности);

x3, *y3* – координаты начальной точки дуги эллипса (или окружности);

x4, *y4* – координаты крнечной точки дуги эллипса (или окружности).

Для ***вычерчивания секторов эллипса или окружности*** на холсте используется метод

Pie (*x1*, *y1*, *x2*, *y2*, *x3*, *y3*, *x4*, *y4*);

где *x1*, *y1* – координаты левого верхнего угла прямоугольника (или квадрата), внутри которого вычерчивается сектор эллипса (или окружности);

x2, *y2* – координаты правого нижнего угла прямоугольника (или квадрата), внутри которого вычерчивается сектор эллипса (или окружности);

x3, *y3*, *x4*, *y4* – координаты конечных точек прямых, являющихся границами сектора эллипса (или круга), начальные точки прямых совпадают с центром эллипса (или окружности).

6. Для ***вычерчивания прямоугольника*** на холсте используется метод

Rectangle (*x1*, *y1*, *x2*, *y2*);

где *x1*, *y1* – координаты левого верхнего угла прямоугольника;

x2, *y2* – координаты правого нижнего угла прямоугольника.

Для ***вычерчивания прямоугольника со скругленными углами*** на холсте используется метод

RoundRec (*x1*, *y1*, *x2*, *y2*, *x3*, *y3*);

где *x1*, *y1* – координаты левого верхнего угла прямоугольника;

x2, *y2* – координаты правого нижнего угла прямоугольника;

x3, *y3* – размеры эллипса, одна четверть которого используется для вычерчивания скругленного угла прямоугольника.

Существует два метода, которые вычерчивают прямоугольник не карандашом, а кистью.

Для ***вычерчивания закрашенного прямоугольника*** кистью используется метод

FillRect (*Координаты углов прямоугольника*);

где *Координаты углов прямоугольника* – структура типа ***TRect***, содержащая координаты *x1* и *y1* левого верхнего угла прямоугольника и координаты *x2* и *y2* правого нижнего углов прямоугольника.

Для ***вычерчивания контура прямоугольника*** кистью используется метод

FrameRect (*Координаты углов прямоугольника*);

где *Координаты углов прямоугольника* – структура типа ***TRect***, содержащая координаты *x1* и *y1* левого верхнего угла прямоугольника и координаты *x2* и *y2* правого нижнего углов прямоугольника.

Структура типа ***TRect*** может быть заполнена при помощи функции

Rect (*x1*, *y1*, *x2*, *y2*).

Для ***вычерчивания многоугольника*** на холсте используется метод

Polygon (*Массив координат точек*);

где *Массив координат точек* – переменная типа массив записей типа ***TPoint***, содержащих координаты *x* и *y* точек, которые являются вершинами многоугольника.

7. Для ***заливки замкнутых областей*** на холсте используется метод

FloodFill (*x, y, Цвет, Способ заливки*);

где *x, y* – координаты точки холста, от которой выполняется заливка;

Цвет – цвет, который используется при определении границы заливаемой области;

Способ заливки – указывает, как именно по этому цвету определяется граница, он может принимать значение одной из именованных констант, предопределенных в Delphi для обозначения способов заливки:

fsSurface –заливка всех точек области, в которых цвет равен цвету, заданному третьим параметром метода;

fsBorder - заливка всех точек области, в которых цвет не раен цвету, заданному третьим параметром метода.

8. Информацию о цвете каждой точки холста содержит внутреннее ***свойство холста Pixels (Пикселы)***, которое представляет собой двумерный массив элементов типа ***TColor***.

Для ***окрашивания точки*** на холсте используется вызов

Pixels[*x, y*]***:=****Цвет*;

где *x, y* – координаты окрашиваемой точки холста;

Цвет – новый цвет окрашиваемой точки холста, он может принимать значение одной из именованных констант, предопределенных в Delphi для обозначения цветов.

Лекция 5. Ввод и вывод данных в Delphi.

Основные вопросы:

1. Вывод на стартовую форму Delphi окна ввода.

2. Способы вывода на стартовую форму Delphi окна сообщения.

1. О***кно ввода*** – это стандартное диалоговое окно, которое выводится на форму во время работы приложения и предназначено для ввода в программу информации от пользователя.

Для ***вывода окна ввода*** в тексте программы используется функция

InputBox (*Заголовок*, *Подсказка*, *Значение по умолчанию*);

где *Заголовок* – текст , который выводится в строке заголовка окна вывода;

Подсказка – текст, который выводится внутри окна ввода;

Значение по умолчанию – текст, который находится в поле ввода по умолчанию.

Функция ***InputBox*** возвращает в программу значение строкового типа, оно представляет собой текст, который пользователь ввел в поле ввода во время работы приложения до закрытия окна ввода, или текст, который находился в поле ввода по умолчанию.

2. О***кно сообщения*** – это стандартное диалоговое окно, которое выводится на форму во время работы приложения и предназначено для вывода программой информации для пользователя. Существует два способа вывода окна сообщения

Для ***вывода*** упрощенного ***окна сообщения*** в тексте программы используется процедура

ShowMessage (*Сообщение*);

где *Сообщение* – текст, который выводится внутри окна сообщения.

Для ***вывода*** более сложного ***окна сообщения*** в тексте программы используется функция

MessageDlg (*Сообщение, Тип сообщения, Список кнопок, Контекст справки*);

где *Сообщение* – текст, который выводится внутри окна сообщения;

Тип сообщения – может принимать значение одной из именованных констант, предопределенных в Delphi для обозначения типов сообщения:

mtWarning – сообщение-Внимание;
mtError – сообщение-Ошибка;
mtInformation – сообщение-Информация;
mtConfirmation – сообщение-Подтверждение;
mtCustom – обычное сообщение;

каждому типу сообщения, за исключением обычного сообщения, соответствует определенный значок, который выводится внутри окна сообщения рядом с текстом;

Список кнопок – список кнопок, которые выводятся внизу окна сообщения, он является множеством, заключается в квадратные скобки и состоит из нескольких разделенных запятыми значений именованных констант, предопределенных в Delphi для обозначения кнопок:

mbOk – кнопка *Ok*;
mbCancel – кнопка *Cancel*;
mbAbort – кнопка *Abort*;
mbRetry – кнопка *Retry*;
mbIgnore – кнопка *Ignore*;
mbYes – кнопка *Yes*;
mbNo – кнопка *No*;
mbAll – кнопка *All*;
mbHelp – кнопка *Help*;

Контекст справки – раздел справочной системы, который должен появляться на экране при нажатии пользователем клавиши [*F1*] (если вывод справки не предусмотрен, значение параметра должно быть равным 0).

Функция ***MessageDlg*** возвращает в программу значение целого типа, оно представляет собой номер кнопки, которая была нажата пользователем во время работы приложения до закрытия окна сообщения. Значение функции может быть сравнено впоследствии в программе со значением одной из именованных констант, предопределенных в Delphi для обозначения номеров кнопок:

mrOk=1 – номер кнопки *Ok*;
mrCancel=2 – номер кнопки *Cancel*;
mrAbort=3 – номер кнопки *Abort*;
mrRetry =4 – номер кнопки *Retry*;
mrIgnore=5 – номер кнопки *Ignore*;
mrYes=6 – номер кнопки *Yes*;
mrNo=7 – номер кнопки *No*;
mrAll=8 – номер кнопки *All*.

Лекция 6. Создание консольного приложения в среде Delphi.

Основные вопросы:

1. Назначение и особенности создания консольного приложения.

2. Операторы ввода и вывода данных в консольном приложении.

1. ***Консольное приложение*** – это программа, предназначенная для работы в операционной системе MS DOS (или окне DOS), для которой устройством ввода является клавиатура, а устройством вывода – монитор.

Работа с в среде Delphi может быть удобна пользователям, предпочитающим программирование в среде Turbo Pascal, или для отладки сложных вычислительных программ.

Для ***создания консольного приложения*** необходимо:

1) закрыть окно формы и окно кода с вкладкой файла ***unit1.pas***;
2) вызвать команду меню ***Project/View Source*** для открытия окна кода с вкладкой файла ***project1.dpr***, текст которого имеет упрощенный вид:

```
program Project1;
uses
  Forms;
{$R *.res}
begin
  Application.Initialize;
  Application.Run;
end.
```

3) откорректировать текст файла, приведя его к следуещему виду:

```
program Project1;
{$APPTYPE CONSOLE}
uses
  Forms;
{$R *.res}
begin
end.
```

В тексте файла директива компиляции {$APPTYPE CONSOLE} – команда компилятору генерировать программу как консольное приложение;

4) дополнить текст файла необходимыми программными блоками;
5) сохранить текст файла под своим именем.

При запуске консольного приложения на выполнение на экране появляется стандартное окно DOS, в которое вводятся и выводятся данные программы.

2. Ввод и вывод данных в консольном приложении осуществляется с помощью операторов ***read(ln)*** и ***write(ln)***, работа которых в языке Object Pascal идентична их работе в языке Turbo Pascal.

Лекция 7. Объявление и использование в среде Delphi пользовательских подпрограмм и модулей. Подключение к проекту дополнительной формы.

Основные вопросы:

1. Правила объявления и использования в среде Delphi пользовательских подпрограмм.

2. Правила объявления и использование в среде Delphi пользовательских модулей.

3. Подключение к проекту Delphi дополнительной формы.

1. ***Объявление и использование*** в языке Object Pascal ***пользовательских подпрограмм*** выполняется таким же образом, как и в языке Turbo Pascal.

Для того, чтобы в модуле формы ***unit1.pas*** можно было использовать пользовательскую подпрограмму, необходимо поместить ее в раздел реализации модуля ***implementation*** перед подпрограммой, которая будет ее вызывать. В разделе объявлений модуля ***interface*** заголовок пользовательской подпрограммы не указывается, т.к. она является ***внутренней*** для модуля формы ***unit1.pas***.

2. ***Объявление и использование*** в языке Object Pascal ***пользовательских модулей*** выполняется по тем же правилам, что и в языке Turbo Pascal.

Пользовательская подпрограмма, помещенная в пользовательский модуль, является ***библиотечной***, т.е. может использоваться в другом модуле. В этом случае в разделе объявлений пользовательского модуля ***interface*** необходимо указать заголовок пользовательской подпрограммы, а саму подпрограмму поместить в раздел реализации пользовательского модуля ***implementation***.

Для ***создания пользовательского модуля*** необходимо:

1) закрыть окно формы и окно кода с вкладкой файла ***unit1.pas***;
2) вызвать команду меню ***File/New/Unit*** для открытия окна кода с вкладкой файла ***unit1.pas***, текст шаблона которого имеет упрощенный вид:

```
unit Unit1;
interface
implementation
end.
```

3) дополнить текст шаблона файла описанием пользовательских подпрограмм;
4) сохранить текст модуля под своим именем.

Для того, чтобы в модуле формы ***unit1.pas*** можно было использовать подпрограмму из пользовательского модуля, необходимо:

1) вызвать команду меню ***Project/Add to Project*** для подключения пользовательского модуля к проекту;
2) подключить пользовательский модуль к модулю формы ***unit1.pas***, указав его имя в блоке ***uses*** раздела объявлений ***interface.***

3. Для ***подключения к проекту дополнительной формы*** необходимо:

1) вызвать команду меню ***File/New/Form*** для добавления на экране окна дополнительной формы и открытия в окне кода дополнительной вкладки файла ***unit2.pas***;
2) подключить модуль дополнительной формы ***unit2.pas*** к модулю формы ***unit1.pas,*** указав его имя ***unit2*** в блоке ***uses*** раздела объявлений ***interface.***

Во время работы приложения для вывода на передний план окна дополнительной формы используется метод формы ***Show***, который делает форму видимой. Для закрытия окна дополнительной формы используется метод формы ***Close***. При этом следует учитывать, что основная форма проекта является родительской, а дополнительная форма – дочерней. Поэтому закрытие дополнительной формы выполняет возврат к основной форме, а закрытие основной формы приводит к закрытию приложения.

Лекция 8. Введение в объектно-ориентированное программирование.

Основные вопросы:

1. Понятие класса, объекта и метода Delphi.

2. Механизмы инкапсуляции, наследования и полиморфизма объектов Delphi.

1. ***Объектно-ориентированное программирование (ООП)*** – это методика разработки программ, в основе которой лежит понятие объекта как некоторой структуры, описывающей объект реального мира, его поведение. Задача, решаемая с использованием методики ООП, описывается в терминах объектов и операций над ними, а программа при таком подходе представляет собой набор объектов и связей между ними.

Язык Object Pascal, поддерживая концепцию объектно-ориентированного программирования, дает возможность определения сложного типа данных – класса.

Класс – это сложная структура, включающая, помимо описания данных, описание процедур и функций, которые могут быть выполнены над представителем этого класса – ***объектом*** (другими словами, над переменной-объектом такого типа-класс).

Данные класса называются его ***полями***, а процедуры и функции класса – его м***етодами***.

2. ***Инкапсуляци***я – это скрытие полей объекта с целью обеспечения доступа к ним только посредством методов класса В этом случае поля объекта становятся его ***свойствами***.

Наследование – это возможность определять новые классы посредством добавления полей, свойств и методов к уже существующим классам. При этом новый, производный класс (потомок) наследует поля, свойства и методы своего базового (родительского) класса.

Полиморфизм – это возможность использования одинаковых имен для методов, входящих в описание различных классов. Концепция полиморфизма обеспечивает в случае применения метода к объекту использование именно того метода, который соответсвует классу этого объекта.

Лекция 9. Особенности работы в текстовом процессоре Microsoft Word. Управление документами Word.

Основные вопросы:

1. Режимы работы в Word.
2. Документы Word в окнах и панелях.
3. Создание документов на базе шаблонов.
4. Разработка компоновки страницы документа.
5. Установка атрибутов текста и абзацев, использование табуляций.
6. Действия с блоками текста - вырезка, копирование, вставка и удаление.

1. Для работы с документами Word предусмотрено несколько стандартных ***режимов работы***:

- обычный режим;
- режим структуры документа;
- режим разметки страницы;
- режим полного экрана;
- режим схемы документа (режим электронного документа);
- режим предварительного просмотра.

Для переключения между режимами используются соответствующие команды меню ***View (Вид)***.

2. В Word для каждого документа открывается отдельное окно и добавляется кнопка на Панели задач. Для открытия дополнительного окна для документа, разделения окна на две панели, а также для переключения между окнами используются соответствующие команды меню ***Window (Окно)***.

3. Каждый документ в Word основывается на некотором ***шаблоне*** – документе, определяющем базовую структуру для других документов. В Word предусмотрено несколько десятков ***шаблонов*** разных типов.

Для создания нового документа стандартного вида используется команда меню ***File (Файл)/New (Создание документа)***, предоставляющая выбор шаблона для документа.

4. Для настройки параметров страницы документа используется команда меню ***File (Файл)/Page Setup (Параметры страницы)***, предоставляющая выбор размера бумаги, ориентации страницы, полей страницы, а также способа вертикального выравнивания текста на странице.

5. Для установки атрибутов текста используется команда меню ***Format (Формат)/Font (Шрифт)***, предоставляющая выбор типа, размера, стиля и цвета шрифта. Для установки атрибутов абзаца используется команда меню ***Format (Формат)/Paragraph (Абзац)***, предоставляющая выбор отступов и интервалов в абзаце, а также способа горизонтального выравнивания текста в абзаце. Для установки табулостопов используется команда меню ***Format (Формат)/Tabs (Табуляция)***, позволяющая вводить новые позиции табуляци или изменять существующие.

6. Для выполнения действий с блоками текста (вырезки, копирования, вставки и удаления блоков) используются соответствующие команды меню ***Edit (Правка)***.

Лекция 10. Работа с таблицами в Word. Импорт в документ Word внешних объектов – рисунков и математических формул.

Основные вопросы:
1. Способы создания таблиц в документе Word.
2. Форматирование данных в ячейках таблицы.
3. Разработка компоновки таблицы.
4. Типы иллюстраций в Word.
5. Импорт рисунков в документ Word.
6. Редактирование иллюстраций.
7. Ввод в документ Word математических формул.
8. Установка атрибутов текста формул.

1. В Word предусмотрено несколько ***способов создания таблиц*** в документе:
- с помощью команды меню ***Table (Таблица)/Insert Table (Вставить),*** предоставляющей выбор параметров таблицы;
- с помощью кнопки ***Insert Table (Добавить таблицу)*** на стандартной панели инструментов с последующим выбором параметров таблицы;
- с помощью команды меню ***Table (Таблица)/Draw Table (Нарисовать таблицу),*** выполняющей переход в режим рисования таблицы “карандашом”.

2. Для изменения ориентации текста в ячейке таблицы используется команда меню ***Format (Формат)/Text Direction (Направление текста)***.

Выбор способа вертикального выравнивания текста в ячейке осуществляется таким же образом, как для текста на странице. Выбор способа горизонтального выравнивания текста в ячейке осуществляется же образом, как для текста в абзаце.

3. Для применения к таблице одного из стандартных форматов используется команда меню ***Table (Таблица)/AutoFormat (Автоформат)***, предоставляющая выбор фомата для таблицы. Тип автоподбора ширины столбцов таблицы устанавливается на этапе создания таблицы с помощью команды меню ***Table (Таблица)/Insert Table (Вставить)***. Для объединения ячеек таблицы используется команда меню ***Table (Таблица)/Merge Cells (Объединить ячейки)***. Для разделенения ячеек таблицы используется команда меню ***Table (Таблица)/Split Cells (Разбить ячейки)***. Вставка в таблицу новых столбцов или строк выполняется командой меню ***Table (Таблица)/Insert Cells (Вставить/ячейки)***. Удаление столбцов или строк выполняется командами меню ***Table (Таблица)/Delete Column (Удалить/столбец)*** или ***Table (Таблица)/Delete Row (Удалить/строку)***.

4. Основные типы иллюстраций, используемые в Word – графические объекты и рисунки.

Графические объекты создаются непосредственно в документе инструментами рисования Word. Они включают:

- автофигуры;
- фигурный текст Word.

Для создания ***автофигур*** используется меню ***AutoShapes (Автофигуры)*** в панели инструментов ***Drawing (Рисование)***. Для создания ***фигурного текста Word*** используется кнопка ***Insert WordArt (Добавить объект WordArt)*** на панели инструментов ***Drawing (Рисование)***.

Рисунки создаются импортом изображений из внешнего файла или другого приложения. Они включают:

- точечные рисунки;
- рисунки типа метафайлов;
- сканированные изображения;
- картинки ClipArt.

5. Для импорта рисунков в документ Word используется команда меню ***Insert (Вставка)/Object (Объект)***. При этом необходимо сделать выбор: сохранить ли изображение как часть документа Word (внедрить) или создать в документе ссылку на внешний файл с изображением. ***Внедренный объект*** представляет собой независимую копию данных. ***Связанный объек***т сохраняется в исходном файле стандартного для данного вида объекта формата.

6. Для редактированя графических объектов и рисунков используются средства, собранные на панелях инструментов ***Drawing (Рисование)*** и ***Picture (Настройка изображения)***.

7. Для создания математических формул в Word и других приложениях Office предназначен ***редактор формул Equation Editor***. Для запуска редактора формул в Word используется команда меню ***Insert (Вставка)/Object (Объект).***

На панели инструментов редактора формул располагаются кнопки для вставки более чем 150 математических символов, а также кнопки для вставки шаблонов скобок, дробей корней, сумм, интегралов, произведений, матриц, подстрочных и надстрочных символов.

8. Для установки атрибутов текста формул (типа, размера и стиля шрифта) используются соответствующие команды меню редактора формул.

Лекция 11. Особенности работы в табличном процессоре Microsoft Excel.

Основные вопросы:

1. Основные элементы экрана Excel.
2. Работа с книгами и листами Excel.
3. Стили адресации ячеек таблицы Excel.
4. Форматирование данных в ячейках таблицы.
5. Форматы данных ячеек.
6. Разработка компоновки таблицы.

1. Экран Excel помимо стандартных элементов окна Windows включает:

- ***окно имен*** - содержит адрес выделенной ячейки (или диапазона ячеек) рабочего поля экрана;
- ***строку формул*** - содержит формулу, выражающую зависимость значения выделенной ячейки рабочего поля экрана от значения другой ячейки (или значений ячеек диапазона);
- ***рабочее поле*** – прямоугольная сетка, разграфленная на строки и столбцы, содержит строку заголовков столбцов и столбец заголовков строк.

2. Главная единица информации в Excel – ***книга (workbook)***. Книга представляет собой физический дисковый файл с расширением ***.xls***, используемый для обработки ихранения данных Excel. Каждая книга может состоять из нескольких ***листов (worksheet)***. Листы служат для непосредственной оргагнизаци и анализа данных.

3. Для обозначения ***столбцов*** таблицы Excel используются латинские буквы и их комбинации. Для обозначения ***строк*** таблицы Excel используются цифры. Пересечения строк и столбцов образуют ***ячейки*** – логические поля, предназначенные для ввода и хранения различной информации (текстовых и числовых данных, формул, графики).

Существует два ***стиля адресации ячеек*** таблицы Excel:

- ***A1*** – при котором указывается обозначение столбца и обозначение строки;
- ***R1C1*** – при котором указывается номер строки и номер столбца.

Для адресации ***диапазона ячеек*** необходимо указать адрес ячейки верхнего левого угла диапазона и затем через двоеточие адрес ячейки правого нижнего угла диапазона.

4. Для форматирования данных в ячейке таблицы используется команда меню ***Format (Формат)/Cells (Ячейки)***, предоставляющая на соответствующих вкладках выбор типа, размера, стиля и цвета шрифта, способов горизонтального и вертикального выравнивания текста в ячейке, а также ориентации содержимого ячейки.

5. В Excel предопределены следующие ***форматы данных*** ячеек:

- общий формат числа;
- числовой формат;
- денежный формат;
- финансовый формат;
- процентный формат;
- дробный формат;
- экспоненциальный (научный) формат;
- дополнительный формат;
- формат даты;
- формат времени
- текстовый формат.

Выбор необходимого формата ячейки выполняется с помощью команды меню ***Format (Формат)/Cells (Ячейки)*** на соответствующей вкладке**.**

6. Для установки ширины столбцов или высоты строк таблицы используются команды меню ***Format (Формат)/Column (Столбец)/Width (Ширина)*** или ***Format (Формат)/Row (Строка)/Height (Высота)***.

Для автоподбора ширины столбцов или высоты строк используются команды меню ***Format (Формат)/Column (Столбец)/AutoFit (Автоподбор ширины)*** или ***Format (Формат)/Row (Строка)/AutoFit (Автоподбор высоты)***.

Лекция 12. Формулы и функции в Excel.

Основные вопросы:

1. Ввод формул в ячейки таблицы Excel.

2. Группы формул в Excel.

3. Виды ссылок в Excel.

4. Ввод стандартных функций Excel с помощью Мастера функций.

1. ***Формула*** является основным средством для манипулирования данными Excel и их анализа. Формула может ссылаться на ячейки текущего листа, ячейки других листов той же книги или ячейки листов другой книги.

Формула имеет следующий вид:

=операнды.

Операндами могут быть:

- константы;
- вызовы функций;
- ссылки на ячейки (или их диапазоны).

Операнды разделяются одним или несколькими ***операторами*** – символами, которые комбинируют операнды или управляют ими.

Если формула содержит ссылку на ячейку, то ячейка, содержащая формулу, называется ***зависимой ячейкой***, ее значение зависит от значения другой, ***влияющей ячейки***.

2. Формулы в Excel подразделяются на 4 ***группы***:

1) арифметические;
2) сравнения;
3) текстовые;
4) адресные.

Формулы каждой группы имеют свой набор операторов и особенности использования.

3. В формулах Excel используется два ***вида ссылок*** на ячейки:

- абсолютные;
- относительные.

Абсолютная ссылка указывает всегда на одну и ту же ячейку. При копировании формулы с абсолютной ссылкой в другую ячейку в своем новом положени она ссылается на ту же исходную ячейку.

Относительная ссылка указывает на ячейку, определенным образом расположенную относительно ячейки с формулой. При копировании формулы с относительной ссылкой в другую ячейку в своем новом положени она ссылается на новую ячейку, положение которой определяется относительно нового положения формулы.

4. В Excel определено большое количество стандартных формул, именуемых ***функциями***. Функции выполняют вычисления, используя значения входных данных – ***аргументов***.

Вызов функции имеет следующий вид:

имя функции (аргумент 1; аргумент 2; …).

Для облегчения процедуры создания функции пользователь может воспользоваться инструментом Excel, который называется Мастером функций. Для вызова Мастера функций используется команда меню ***Insert (Вставка)/Function (Функция)***, предоставляющая выбор типа функции и имени функции, а также возможность ввода значений для аргументов функции.

Лекция 13. Работа со списками в Excel.

Основные вопросы:
1. Создание структуры списка.
2. Сортировка данных списка.
3. Отбор данных списка с помощью автофильтра.

1. ***Список*** – это набор строк, содержащий связанные данные. Список может использоваться как небольшая база данных, в которой строки выступают в качестве ***записей***, а столбцы являются ***полями***. Первую строку списка при этом Excel воспринимает в качестве ***заголовков столбцов***.

2. Для сортировки данных списка используется команда меню ***Data (Данные)/Sort (Сортировка)***, предоставляющая выбор полей, по которым список должен быть отсортирован по возрастанию или убыванию значений ячеек.

3. Для отбора данных списка с помощью автофильтра используется команда меню ***Data (Данные)/Filter (Фильтр)/AutoFilter (Автофильтр)***, предоставляющая выбор полей, по которым необходимо выполнить фильтрование данных, а также критерия фильтрования.

Лекция 14. Представление данных Excel в виде диаграммы.

Основные вопросы:
1. Создание диаграмм Excel с помощью Мастера диаграмм.
2. Настройка элементов диаграммы.

1. ***Диаграммы*** являются одним из самых эффективных способов анализа данных.

Для облегчения процедуры создания и настройки диаграммы пользователь может воспользоваться инструментом Excel, который называется Мастером диаграмм. Для вызова Мастера диаграмм используется команда меню ***Insert (Вставка)/Chart (Диаграмма)***, предоставляющая выбор типа диаграммы, ее вида в пределах типа, а также выбор ряда параметров, определяющих оформление различных элементов диаграммы.

2. Настройка элементов диаграммы, как правило, включает установку флажков для отображения основных линий сетки по осям *x* и *y*, легенды на соответствующих вкладках.

Лекция 15. Надстройка Solver (Поиск решения) среды Microsoft Excel.

Основные вопросы:
1. Нахождение корней уравнений с помощью надстройки Solver (Поиск решения).
2. Решение задач оптимизации функций с помощью надстройки Solver (Поиск решения).

1. Для нахождения корней уравнения с помощью надстройки Solver (Поиск решения) используется команда меню ***Сервис/Поиск решения***, предоставляющая выбор целевой ячейки, значение которой проверяется в процессе поиска (оно должно стремиться к нулю), а также влияющей ячейки, значение которой изменяется в процессе поиска.

2. Для нахождения экстремумов функции с помощью надстройки Solver (Поиск решения) используется команда меню ***Сервис/Поиск решения***, предоставляющая выбор целевой ячейки, значение которой проверяется в процессе поиска (оно должно стремиться к максимуму или минимуму), а также влияющей ячейки, значение которой изменяется в процессе поиска.