**ТЕЗИСЫ ЛЕКЦИЙ**

***Лекция № 1***

**Тема:** ***Подготовка и решение задач на компьютере. Этапы подготовки и решения задач на компьютере***

**ПЛАН:**

1. Этапы подготовки и решения задач на компьютере.

2. Типы обработки данных.

3. Этапы разработки программы.

4. Понятие алгоритма. Свойства алгоритма.

5. Системы программирования.

6. Качество программ.

**ТЕЗИСЫ**

Подготовка и решение задач на компьютере имеет следующие этапы:

постановка задачи и разработка технического задания;

разработка:

а) информационной модели решения задачи;

б) математической модели решения задачи;

в) алгоритма для решения задачи;

г) программы для решения задачи на компьютере;

документирование разработки и выпуск отчета;

счет по разработанной программе и анализ результатов.  
Обработка данных в компьютере может быть:

* числовая: решение задач численными методами;
* нечисловая: накопление, хранение, поиск, сортировка данных.

Математическая модель - это описание метода решения задачи. Этот этап включает разработку или выбор численных методов или методов нечисловой обработки данных.

Разработка алгоритма решения задачи (алгоритмизация) - один из самых сложных и ответственных этапов. Он включает формализацию процесса обработки данных и представление его в форме, позволяющей упростить дальнейшее кодирование алгоритма, т. е. его запись на одном из алгоритмических языков.

Для решения задачи на компьютере надо разработать программу.

Программирование - это теория и практика создания программ для компьютера. Программист (разработчик или аналитик) - это специалист, создающий программное изделие и соответствующую документацию.

Разработка программы включает:

1) подготовку тестовых исходных данных;

2) написание текста программы на алгоритмическом языке;

3) перенос программ и данных на машинные носители компьютера;

4) отладку и тестирование программы.

Отладка - это поиск и исправление ошибок программы и алгоритма.

Тестирование программы - это проверка правильности ее функционирования во всем диапазоне допустимых значений исходных данных.

Документирование включает разработку и выпуск технического отчета.

**Алгоритм. Основные свойства. Способы представления алгоритмов**

Алгоритм - это точное предписание по выполнению некоторого процесса обработки данных, который через разумное конечное число шагов приводит к решению задачи данного типа для любых допустимых вариантов исходных данных.

Слово "алгоритм" произошло от имени средневекового узбекского матема­тика IX в. Абу Джафара Мохамеда ибн Мусы аль-Хорезми. Он разработал правила выполнения четырех арифметических действий в десятичной системе счисления.

Алгоритм обладает тремя основными свойствами: детерминированности, массовости и результативности.

Детерминированность (определенность) означает, что метод решения задачи должен быть дан в алгоритме способом, не допускающим различных толкований человеком, транслятором, компьютером.

Массовость означает, что алгоритм должен быть пригоден для любых допустимых вариантов исходных данных.

Результативность означает, что в итоге выполнения алгоритма должен быть получен результат за конечное число шагов.

Алгоритм может быть представлен словесным описанием, в виде схем и на алгоритмическом языке, в том числе на машинном.

**Системы программирования. Качество программ**

Система программирования - это язык и средства разработки прикладных программ.

Существующие системы программирования делятся на компилирующие и интерпретирующие в соответствии с типом их трансляторов. Трансляторы - это программы, которые переводят текст, с языка программирования на машинный. Они могут быть двух типов: компиляторы и интерпретаторы. Компи­лятор - это транслятор, в котором трансляция и выполнение программы разне­сены во времени: сначала весь текст транслируется, без выполнения програм­мы, полученной в результате ее трансляции, а затем полученная программа выполняется. Интерпретаторы - это трансляторы, в которых текст транслируется и выполняется пооператорно, т. е. трансляция и выполнение операторов программы совмещены во времени.

Жизненный цикл программ в общем случае состоит из трех фаз: разработки, использования и сопровождения (продолжающейся разработки).

ЛИТЕРАТУРА:

1. Семакин И.Г. Основы алгоритмизации и программирования : учебник для спо / И.Г. Семакин, А.П. Шестаков. - 2-е изд., стереотип. - М. : Академия, 2011.
2. Голицына О.Л. Основы алгоритмизации и программирования : учебн. пособие для спо / О.Л. Голицына, И.И. Попов. - 3-е изд., исправ. и доп. - М. : ФОРУМ, 2008.
3. Фризен И.Г. Офисное программирование : учебн. пособие / И.Г. Фризен. - 2-е изд. - М. : Дашков и К, 2010.
4. Хорев П.Б. Технологии объектно-ориентированного программирования : учебн. пособие для вузов / П.Б. Хорев. - 2-е изд., стереотип. - М. : Академия, 2008.

***Лекция № 2***

**Тема:** **История развития языков программирования**

**ПЛАН:**

1. Понятия язык программирования и машинный язык.
2. Появление языков программирования.

**ТЕЗИСЫ**

Язык программирования - это совокупность набора символов (алфавит) системы, правил образования (синтаксис) и истолкования конструкций из символов (семантика) для задания алгоритмов с использованием символов естественного языка.

Машинный язык - это такой язык, который компьютер воспринимает непосредственно, то есть это язык машинных команд данной модели компьютера.

В 1955 г. появился первый язык высокого уровня Фортран.

На языке высокого уровня исходная программа состояла из последовательности операторов.

Затем в 1960 г. появился Алгол (Algoritmic language - алгоритмический язык), также ориентированный на научное применение, в него было введено множество новых понятий, подхваченных позднейшими языками, например, понятие блочной структуры.

Также при поддержке фирмы IBM появился язык Кобол (Cobol - сокращенное от английских слов Comnon business oriented language - общепринятый деловой ориентированный язык). Он был ориентирован на решение экономических задач, а точнее - на обработку информации.

Язык Бейсик, напоминающий Фортран, но более простой, быстро стал очень популярным. Достоинствами Бейсика являются удобные средства ввода, отладки и испытания программ, а также возможность доступа ко всем основным ресурсам компьютера. Его отличает простота конструкций и возможность осуществления диалогового режима работы с ЭВМ.

В 1967-1968 гг. появился язык PL/1 (Programming language - универсальный программно-ориентированный). Этот язык, как языки программирования СИ, Ада и Паскаль, может использоваться как для научных задач, так и для задач управления. Он очень мощный, но и очень сложный, используется лишь в высших учебных заведениях и научно-исследовательских центрах.

В 1970 г. профессор Никлаус Вирт создал в Цюрихском политехническом университете язык Паскаль (Pascal).

В 1980 г. появился язык Ада. Это самый новый и самый мощный из языков программирования, он унаследовал качества языков Паскаль и Алгол-68 и дополнительно приобрел многие другие качества: системное программирование, параллельность и т.д.

Языки Лисп и Пролог - это основные языки для задач, связанных с искуственным интеллектом. Липс оперирует списками (цепная последовательность элементов), а Пролог - деревьями (логическими разветвлениями).

Существует огромное множество специализированных языков, позволяющих эффективно решать задачи в некоторых областях: моделирования (языки Симула, Симкрит и GPSS), управления аппаратурой (ФОРТ), для написания системных программ (СИ), написания баз данных (Кодасил), обучения программированию (Лого, Робик, алгоритмический язык А.П.Ершова) и другие.

Язык Си появился в начале 70-х, автор – Деннис Ричи, Bell Laboratories. Это язык системного программирования, он разрабатывался для реализации ОС Unix (альтернатива языку ассемблера), но быстро превратился в язык общего пользования.

Компактные программы, эффективный объектный код, но сложный синтаксис. Программирование на C требует хорошей подготовки («много темных закоулков»), довольно трудно читать чужие тексты программ.

**C++**

В начале 80-х Бьярн Страуструп дополнил язык C объектно-ориентированными средствами по образцу языка Simula 67. При этом была существенно улучшена семантика.

**C# (читается: «Си шарп»)**

Язык для разработки платформы .NET Framework и для программирования под эту платформу. Платформа .NET – среда управляемого выполнения программ фирмы Microsoft.

ЛИТЕРАТУРА:

1. Семакин И.Г. Основы алгоритмизации и программирования : учебник для спо / И.Г. Семакин, А.П. Шестаков. - 2-е изд., стереотип. - М. : Академия, 2011.
2. Голицына О.Л. Основы алгоритмизации и программирования : учебн. пособие для спо / О.Л. Голицына, И.И. Попов. - 3-е изд., исправ. и доп. - М. : ФОРУМ, 2008.
3. Фризен И.Г. Офисное программирование : учебн. пособие / И.Г. Фризен. - 2-е изд. - М. : Дашков и К, 2010.
4. Хорев П.Б. Технологии объектно-ориентированного программирования : учебн. пособие для вузов / П.Б. Хорев. - 2-е изд., стереотип. - М. : Академия, 2008.

***Лекции № 3-4***

**Тема:** ***Данные и основные операторы в Pascal***

**ПЛАН:**

1. Характеристика Pascal.

2. Структура языка Pascal.

3. Алфавит языка Pascal.

4. Типы данных

5. Классификация операторов.

6. Встроенные в Pascal функции.

**ТЕЗИСЫ**

Паскаль удовлетворяет требованиям структурного программирования: он имеет операторы для реализации управляющих структур программирования.

Достоинства Паскаля:

1. он ориентирован на структурное программирование, имеет развитые средства контроля и достаточно прост в изучении;
2. язык имеет хороший состав типов и структур данных;
3. трансляторы с Паскаля есть во всех распространенных ПК;
4. конкретные реализации языка дают возможность использовать все аппаратные средства ПК;
5. на основе языка Паскаль разработана Delphi - одна из современных систем визуального программирования.

Недостатки языка Паскаль: в нем нет операции возведения в степень. Из-за богатства возможностей язык не прост для освоения в полном объеме при первоначальном изучении.

**Структура языка Паскаль**

Алгоритмический язык - это система правил для описания процесса обработки данных. Он включает алфавит, из символов которого формируются слова, выражения и операторы (предложения).

Алфавит - это набор допустимых символов языка. Слова (идентификаторы, константы) формируются из симво­лов по определенным правилам.

Выражения - это группы слов, имеющих определенный смысл.

Операторы - это предложения для описания некоторого дей­ствия в процессе обработки данных. Из операторов формируются программы.

Структура различных языков программирования однотипна. Они должны иметь средства для:

1. определения свойств объектов программы (ее данных);
2. определения обработки, т. е. получения новых значений объектов;
3. организации процесса обработки данных (разветвлений процесса обработки  
   данных и циклической обработки);
4. ввода-вывода данных, т. е. обмена данными между ОП и внешними устрой­  
   ствами (магнитными дисками, портами, принтером);
5. формирования и использования модулей (программ, процедур и функций) и  
   библиотек модулей (системных и пользовательских, разработанных про­  
   граммистом).

**Алфавит языка Паскаль**

Все конструкции языка формируются из символов его алфавита по определенным в языке правилам. Алфавит языка Паскаль включает буквы, цифры и специальные символы.

Буква : : = А | В | С | ... |x|Y|z|a|b|c| ...|y|z

Цифра ::=0 | 1 | 2 | 3 | 4 | 5 | б | 7 | 8 |9

Специальные символы : : = арифметические операции |

логические операции | операции отношения |

операции над множествами | ограничители | ключевые слова Арифметические операции :: = +|-|\*|/|DIV| MOD

Логические операции : : = NOT AND | OR | XOR | SHL | SHR

Операции отношения : : = = | <> | < | <= | >= | > | IN

Операции над множествами : : = + | - | \* | IN

Ограничители и разделители ::=(|)|[|]|{|}|(\*|\*)|

.| ..| , | ; | : | ' |:=

Символ ::= означает "равно по определению". Символ | - означает "или".

Кроме латинских букв в значениях строковых и символьных данных можно использовать большие (строчные) и маленькие (прописные) буквы русского алфавита.

В текстах программ допускаются комментарии и пробелы.

Комментарий - это текст, поясняющий программу и не влияющий на процесс ее выполнения. Комментарии ограничиваются символами {и} или {\*и\*}.

**Классификация данных**

Данные - это формализованное представление информации. Они обрабатываются программами. Данные могут быть в виде констант и переменных.

Константы - это конкретные значения данных. Например: 2,5, Маша.

Переменные - это данные, которые в процессе выполнения программы могут принимать различные значения.

**Встроенные процедуры и функции языка Паскаль**

Паскаль содержит 8 стандартных модулей, в том числе: System, Crt, Dos, Graph, Overlay. Для использования их процедур и функций надо после оператора PROGRAM записать оператор

Uses список-имен-используемых-модулей.

Например:

Program KR ;

Uses Crt, Graph;

Подключение модуля System производится по умолчанию, без указания его имени в списке Uses. В модуле System располагаются наиболее часто используемые процедуры и функции языка. Основные группы процедур и функций модуля System:

1. процедуры управления выполнением программы;
2. функции преобразования типов;
3. процедуры и функции для переменных ординального типа;
4. математические функции;
5. процедуры и функции для работы со строками;
6. процедуры и функции управления динамической памятью;
7. процедуры ввода-вывода и работы с файлами и каталогами.

Процедуры и функции модуля Crt предназначены для работы с экраном: его очистки, управления положением курсора на экране, управления клавиатурой, определения цветов фона и символов, управления звуком и окнами экрана.

Процедуры и функции модуля Graph предоставляют пользователю графические возможности языка. Процедуры и функции модуля Overlay позволяют организацию работы с программами оверлейной структуры.

Процедуры и функции модуля DOS предназначены для обслуживания прерываний, проверки состояния диска, обработки файлов, управления процессами и операционной средой.

**Структура программы на языке Паскаль**

Программа на языке Паскаль состоит из заголовка программы и блока (тела программы). Заголовок программы или подпрограммы - это один из операторов:

PROGRAM - для программы;

PROCEDURE - для Процедуры;

FUNCTION - для функции.

Блок программы и подпрограммы на языке Паскаль состоит из двух частей:

1)декларативной - описания (объявления) данных программы и текстов под­  
программ (разделы описаний);

2)выполнимой - описания действий, которые надо выполнить над данными  
(раздел операторов).

Блок включает все, кроме заголовка программы или подпрограммы. В общем случае блок может состоять из шести разделов различных типов. Все они, кроме раздела операторов, необязательны. Каждый раздел описаний начинается ключевым словом:

LABEL - для определения меток программы;

CONST - для определения констант;

TYPE - для определения типов данных;

VAR - для определения переменных;

PROCEDURE и FUNCTION - для определения процедур и функций;

BEGIN - определяет начало раздела операторов программы.

**ЛИТЕРАТУРА:**

1. Семакин И.Г. Основы алгоритмизации и программирования : учебник для спо / И.Г. Семакин, А.П. Шестаков. - 2-е изд., стереотип. - М. : Академия, 2011.
2. Голицына О.Л. Основы алгоритмизации и программирования : учебн. пособие для спо / О.Л. Голицына, И.И. Попов. - 3-е изд., исправ. и доп. - М. : ФОРУМ, 2008.
3. Фризен И.Г. Офисное программирование : учебн. пособие / И.Г. Фризен. - 2-е изд. - М. : Дашков и К, 2010.
4. Хорев П.Б. Технологии объектно-ориентированного программирования : учебн. пособие для вузов / П.Б. Хорев. - 2-е изд., стереотип. - М. : Академия, 2008.

***Лекции № 5-6***

***Тема:******Объектно-ориентированное программирование***

**ПЛАН:**

1. Введение в объектно-ориентированное программирование.
2. Основные понятия визуального программирования.
3. Особенности ООП.
4. Структурное и модульное программирование.

**ТЕЗИСЫ**

Объектно-ориентированное программирование (ООП) - это метод программирования, при использовании которого главными элементами программ являются объекты.

В языках программирования понятие объекта реализовано как совокупность свойств (структур данных, характерных для данного объекта), методов их обработки (подпрограмм изменения их свойств) и событий, на которые данный объект может реагировать и, которые приводят, как правило, к изменению свойств объекта.

Объединение данных и свойственных им процедур обработки в одном объекте, называется инкапсуляцией и является одним из важнейших принципов ООП.

Класс - это шаблон, на основе которого может быть создан конкретный программный объект, он описывает свойства и методы, определяющие поведение объектов этого класса.

Каждый конкретный объект, имеющий структуру этого класса, называется экземпляром класса.

Наследование - такая организация классов, которая предусматривает создание новых классов на базе существующих и позволяет классу потомку иметь (наследовать) все свойства класса – родителя.

Полиморфизм - означает, что рожденные объекты обладают информацией о том, какие методы они должны использовать в зависимости от того, в каком месте цепочки (дерева классов) они находятся, иными словами это концепция, реализующая "множество методов в одном интерфейсе".

Модульность - это такая организация объектов, когда они заключают в себе полное определение их характеристик, никакие определения методов и свойств не должны располагаться вне его, это делает возможным свободное копирование и внедрение одного объекта в другие.

В объектно-ориентированном анализе для описания динамического поведения объектов принято использовать модель состояний.

Событие - это переход объекта из одного состояния в другое.

Далее, выделяются четыре аспекта события:

метка - уникальный идентификатор события.

значение - текстовое сообщение о сути произошедшего.

предназначение - модель событий, которая принимает событие.

данные - данные, которые переносятся от одного объекта к другому.

Первый ряд примеров событий доставляет собственно сам жизненный цикл объекта:

создание объекта;

уничтожение объекта.

Сутью структурного программирования является возможность разбиения программы на составляющие элементы.

Распространены две методики (стратегии) разработки программ, относящиеся к структурному программированию: программирование "сверху вниз" и программирование "снизу вверх".

Программирование "сверху вниз", или нисходящее программирование – это методика разработки программ, при которой разработка начинается с определения целей решения проблемы, после чего идет последовательная детализация, заканчивающаяся детальной программой. Является противоположной методике программирования «снизу вверх».

При нисходящем проектировании задача анализируется с целью определения возможности разбиения ее на ряд подзадач. Затем каждая из полученных подзадач также анализируется для возможного разбиения на подзадачи. Процесс заканчивается, когда подзадачу невозможно или нецелесообразно далее разбивать на подзадачи.

В данном случае программа конструируется иерархически - сверху вниз: от главной программы к подпрограммам самого нижнего уровня, причем на каждом уровне используются только простые последовательности инструкций, циклы и условные разветвления.

Программирование "снизу вверх", или восходящее программирование – это методика разработки программ, начинающаяся с разработки подпрограмм (процедур, функций), в то время когда проработка общей схемы не закончилась. Является противоположной методике программирования «сверху вниз».

Достоинства структурного программирования:

1) повышается надежность программ (благодаря хорошему структурированию при проектировании, программа легко поддается тестированию и не создает проблем при отладке);

2) повышается эффективность программ (структурирование программы позволяет легко находить и корректировать ошибки, а отдельные подпрограммы можно переделывать (модифицировать) независимо от других);

3) уменьшается время и стоимость программной разработки;

4) улучшается читабельность программ.

Технология структурного программирования при разработке серьезных программных комплексов, основана на следующих принципах:

- программирование должно осуществляться сверху вниз;

- весь проект должен быть разбит на модули (подпрограммы) с одним входом и одним выходом;

- подпрограмма должна допускать только три основные структуры – последовательное выполнение, ветвление (if, case) и повторение (for, while, repeat).

- недопустим оператор передачи управления в любую точку программы (goto);

- документация должна создаваться одновременно с программированием в виде комментариев к программе.

Структурное программирование эффективно используется для решения различных математических задач, имеющих алгоритмический характер.

Модульное программирование - это организация программы как совокупности небольших независимых блоков (модулей), структура и поведение которых подчиняется определенным заранее правилам.

Модулем (в модульном программировании) называется множество взаимосвязанных подпрограмм (процедур) вместе с данными, которые эти подпрограммы обрабатывают.

Модульное программирование предназначено для разработки больших программ.

**ЛИТЕРАТУРА:**

1. Семакин И.Г. Основы алгоритмизации и программирования : учебник для спо / И.Г. Семакин, А.П. Шестаков. - 2-е изд., стереотип. - М. : Академия, 2011.
2. Голицына О.Л. Основы алгоритмизации и программирования : учебн. пособие для спо / О.Л. Голицына, И.И. Попов. - 3-е изд., исправ. и доп. - М. : ФОРУМ, 2008.
3. Фризен И.Г. Офисное программирование : учебн. пособие / И.Г. Фризен. - 2-е изд. - М. : Дашков и К, 2010.
4. Хорев П.Б. Технологии объектно-ориентированного программирования : учебн. пособие для вузов / П.Б. Хорев. - 2-е изд., стереотип. - М. : Академия, 2008.

***Лекции № 7-8***

***Тема: Циклы***

**ПЛАН:**

* + - 1. Понятие цикл
      2. Цикл со счетчиком For и его синтаксис.
      3. Оператор цикла с предусловием while.
      4. Оператор цикла с постусловием repeat.
      5. Сравнение работы операторов цикла.

**ТЕЗИСЫ**

В программирование цикл — это многократное повторение определенных инструкций.

Циклы состоят из заголовка и тела. Заголовок содержит в себе условия, которые определяют работу цикла, а тело – повторяющиеся действия.

В ЯП **Pascal** есть три вида циклов:

*цикл с параметром;*

*цикл с предусловием;*

*цикл с постусловием.*

Их алгоритмы выполнения различны, но есть и общее: после выполнения тела цикла, проверяется условие, и в зависимости от него работа цикла заканчивается, либо снова выполняется тело.

For — цикл с параметром

Цикл с параметром, известный также как цикл со счетчиком, используется при известном количестве повторений. Он имеет две формы записи:

1. For <счетчик>:=< начальное значение> To <конечное значение> Do <тело цикла>;
2. For <счетчик>:=<начальное значение> Downto <конечное значение> Do <тело цикла>;

*Счетчик* – это переменная порядкового типа. Начальное и конечное значение должны быть того же типа, что и счетчик. Тело выполняется до тех пор пока условие истинно.

Оператор цикла с предусловием while

Оператор цикла while используется в программе, если надо провести некоторые повторные вычисления (цикл), однако число повторов заранее неизвестно и определяется самим ходом вычисления.

В общем виде оператор выглядит так:

**while** условие **do**

**begin**

{последовательность операторов};

**end**;

где условие – выражение логического типа.

Оператор цикла while работает следующим образом:

1. Вычисляется условие.
2. Если условие ложно, то вход в цикл не выполняется и управление передается оператору, следующему непосредственно за операторами тела цикла.
3. Если же условие истинно, то происходит вход в цикл и однократное выполнение операторов. Как только достигнут конец тела цикла, управление передается на его заголовок, где снова вычисляется его условие.

Оператор цикла с постусловием repeat

Инструкция repeat как и инструкция while, используется в программе, если надо провести некоторые повторяющиеся вычисления (цикл), однако число повторений не известно и определяется самим ходом вычислений.

В общем виде оператор выглядит так:

**repeat**

{последовательность операторов};

**until** условие;

где условие – выражение логического типа.

Оператор работает следующим образом:

1. Выполняются инструкции следующие за слово repeat.
2. Вычисляется значение условия. Если условие ложно, то повторно выполняются инструкции цикла. Если же условие истинно, то выполнение цикла заканчивается.

Таким образом, операторы находящиеся между repeat и until выполняются до тех пор, пока условие ложно.

**ЛИТЕРАТУРА:**

1. Семакин И.Г. Основы алгоритмизации и программирования : учебник для спо / И.Г. Семакин, А.П. Шестаков. - 2-е изд., стереотип. - М. : Академия, 2011.
2. Голицына О.Л. Основы алгоритмизации и программирования : учебн. пособие для спо / О.Л. Голицына, И.И. Попов. - 3-е изд., исправ. и доп. - М. : ФОРУМ, 2008.
3. Фризен И.Г. Офисное программирование : учебн. пособие / И.Г. Фризен. - 2-е изд. - М. : Дашков и К, 2010.
4. Хорев П.Б. Технологии объектно-ориентированного программирования : учебн. пособие для вузов / П.Б. Хорев. - 2-е изд., стереотип. - М. : Академия, 2008.

***Лекции № 9-11***

***Тема: Массивы***

**ПЛАН:**

* + - 1. Понятие одномерный массив.
      2. Инициализация одномерного массива.
      3. Способы заполнения одномерного массива.
      4. Алгоритмы работы с одномерными массивами.
      5. Сортировка одномерного массива.
      6. Поиск элементов в одномерном массиве.
      7. Многомерные массивы.

**ТЕЗИСЫ**

Массив – это упорядоченный набор однотипных элементов, имеющих общее имя.

**Объявление** **массива**.

Перед использованием массив, как и любая переменная, должен быть объявлен в разделе объявления переменных. В общем виде объявление массива выглядит так:

имя: array [нижний\_индекс..верхний\_индекс] of тип.

Способы заполнения одномерного массива

1. Оператор присваивания.

2) Оператором ввода с клавиатуры.

3) Заполнение массива с использованием генератора случайных чисел.

4) Заполнение по формуле.

**Вывод массива.**

For i:=1 to 10 do writeln(a[i]);

Вывод пятого элемента массива записывается так : write(a[5]);

#### Алгоритмы работы с массивами

1) Сумма элементов массива.

2) Сумма положительных чисел.

3) Сумма и количество положительных чисел.

5) Поиск наибольшего (наименьшего) элемента в массиве.

**Ввод массива.**

Под вводом массива понимается ввод значений элементов массива. Как и вывод массива, ввод удобно реализовать с помощью оператора цикла for. Чтобы пользователь программы знал, ввода какого элемента массива ожидает программа, следует организовать вывод подсказок перед вводом очередного элемента массива. В подсказке обычно указывают индекс элемента массива.

**Поиск в массиве.**

При решении многих задач возникает необходимость установить, содержит ли массив определенную информацию или нет. Например, определить есть ли в массиве нулевые элементы. Задачи такого типа называются поиском в массиве.

Для организации поиска в массиве могут быть использованы различные алгоритмы. Наиболее простой – это алгоритм простого перебора. Поиск осуществляется последовательным сравнением элементов массива с образцом.

## Многомерные массивы

Для определения позиции элемента в двумерном массиве необходимы два индекса.

Любой двумерный массив есть матрица, а матрица есть таблица. Поэтому удобно описывать двумерные массивы путем указания границ изменения индексов (номеров) строк и столбцов.

Например, таблица символов M × N, где M – число строк и N – число столбцов, может быть описана:

var TAB : array [1..M, 1..N] of char;

|  |
| --- |
| **Общая форма записи**  VAR <имя> : ARRAY [тип индекса строки, тип индекса столбца]  OF <тип компонент>; |

|  |
| --- |
| **Общая форма записи**  TYPE <тип строки> = ARRAY [тип индекса] OF <тип компонент>;  <тип столбца> = ARRAY [тип индекса] OF <тип строки>;  VAR <переменная массива> : <тип столбца>; |

Эти два вида определения массивов задают и два способа обращения к элементам массива: TAB [I, J] – в первом случае и TAB [I][J] – во втором.

**ЛИТЕРАТУРА:**

1. Семакин И.Г. Основы алгоритмизации и программирования : учебник для спо / И.Г. Семакин, А.П. Шестаков. - 2-е изд., стереотип. - М. : Академия, 2011.
2. Голицына О.Л. Основы алгоритмизации и программирования : учебн. пособие для спо / О.Л. Голицына, И.И. Попов. - 3-е изд., исправ. и доп. - М. : ФОРУМ, 2008.
3. Фризен И.Г. Офисное программирование : учебн. пособие / И.Г. Фризен. - 2-е изд. - М. : Дашков и К, 2010.
4. Хорев П.Б. Технологии объектно-ориентированного программирования : учебн. пособие для вузов / П.Б. Хорев. - 2-е изд., стереотип. - М. : Академия, 2008.

***Лекция № 12***

***Тема: Обработка строк***

**ПЛАН:**

1. Структура строки символов.
2. Операции по обработке строк.
3. Процедуры редактирования строк.
4. Процедуры преобразования строк.

**ТЕЗИСЫ**

## Структурная организация

Строковый тип - массив, элементами которого являются данные литерного типа CHAR.

Элементы этого массива пронумерованы, начиная с 0. Сами символы располагаются, начиная с элемента с индексом 1.

Так как все элементы массива одинаковы по размеру, то под динамическую длину, как и под символ, отводится 1 байт. Поэтому строка символов не может иметь длину большую 255.

Строка длиной 0 - пустая строка.

В определении типа, если нужно зарезервировать строку длиной менее 255 символов, в квадратных скобках указывается необходимая длина.

В структурном программировании на языке Паскаль при использовании данных строкового типа этот тип не определяется в разделе нестандартных типов, а при описании переменных используется непосредственное определение строкового типа.

## Множество значений

Включает все значения, которые могут быть в структуре заданной в определении типа.

## Множество операций

Со строкой символов можно работать, как с одномерным массивом из символов в пределах заполненных элементов этого массива. Т.е. индексы находятся в диапазоне от 0 по значение ORD(<имя переменной типа строка>[0]).

### Операция конкатенации

Это бинарная операция со знаком +. Здесь операнды и результат любого строкового типа. В результате выполнения операции к строке, являющейся первым операндом, приклеивается вторая строка.

### Операции сравнения (отношения)

Это бинарные операции со знаками =|<>|<|>|<=|>= . Операнды любого строкового типа. Результат типа BOOLEAN. Сравнение производится посимвольно в соответствии с кодами символов, начиная с левого края строк, первое несовпадение определяет результат операции (отсутствие символа - код 0).

Для определения результата операции сравнения можно использовать два следующих правила:

1. две строки равны, если их динамические длины равны и попарно совпадают все соответствующие символы обеих строк;
2. первая строка считается меньше второй, если после первых попарно совпадающих символов код символа первой строки меньше кода символа второй строки или в первой строке отсутствует символ.

### Операция определения адреса

Унарная операция со знаком @. Операнд - переменная строкового типа, результат - указатель на участок оперативной памяти, в котором располагается переменная (адрес нулевого байта строки).

### Подпрограммы

#### Функции

а) определение текущей (динамической длины) строки:

LENGTH(<исходная строка>:STRING):INTEGER;

б) выделение из строки подстроки (получение частичной копии):

COPY(<исходная строка>:STRING;  
 <номер начальной позиции>,  
 <количество копируемых символов>:INTEGER)

:STRING;

в) определение позиции первого вхождения подстроки в строку (если подстрока не входит в строку, то результат 0):

POS(<подстрока>,<строка>:STRING):BYTE;

г) конкатенация (слияние) группы строк:

CONCAT(<первая строка>,

<вторая строка>,

...

<последняя строка>:STRING)

:STRING;

#### Процедуры редактирования строк

а) удаление последовательности символов строке:

б) вставка подстроки в строку:

#### Процедуры преобразования строк

а) преобразование численного значения в строку символов;

б) преобразование строки символов в число.

**ЛИТЕРАТУРА:**

1. Семакин И.Г. Основы алгоритмизации и программирования : учебник для спо / И.Г. Семакин, А.П. Шестаков. - 2-е изд., стереотип. - М. : Академия, 2011.
2. Голицына О.Л. Основы алгоритмизации и программирования : учебн. пособие для спо / О.Л. Голицына, И.И. Попов. - 3-е изд., исправ. и доп. - М. : ФОРУМ, 2008.
3. Фризен И.Г. Офисное программирование : учебн. пособие / И.Г. Фризен. - 2-е изд. - М. : Дашков и К, 2010.
4. Хорев П.Б.Технологии объектно-ориентированного программирования : учебн. пособие для вузов / П.Б. Хорев. - 2-е изд., стереотип. - М. : Академия, 2008.

***Лекции № 13-15***

***Тема: Работа с файлами***

**ПЛАН:**

1. Виды файлов.
2. Общие правила работы с файлами.
3. Организация работы с файлами прямого доступа.
4. Алгоритм сортировки файла прямого доступа.
5. Организация работы с последовательными файлами.

**ТЕЗИСЫ**

Различают три вида файлов:

* последовательные;
* прямого доступа;
* текстовые или символьные.

Эти виды определяют способы организации работы с файлами.

**Общие правила работы с файлами**

При работе с файлами любого вида необходимо придерживаться некоторых общих правил:

a) в разделе нестандартных типов определяется файловый тип (для нетипизированных файлов это делать не надо, так как для указания этого типа используется имя FILE);

б) файл описывается в разделе описания переменных путем указания имени файловой переменной и типа файла (таким образом определяется имя логического файла);

в) в разделе операторов до использования файла файловая переменная связывается с конкретным физическим файлом:

assign (<имя файловой переменной, называемое именем логического файла>, '<имя физического файла>');

г) перед выполнением операций с файлом, он открывается для использования с помощью одной из команд (обращения к процедуре).

reset(<имя файловой переменной>) - открытие существующего файла для работы.

rewrite(<имя файловой переменной>) - открытие для формирования (записи) нового файла.

append(<имя файловой переменной типа text>) – открытие для дозаписи в конец существующего текстового файла.

При открытии файла, он монополизируется, т.е. становится доступным только данной программе и недоступным всем другим программам;

д)начинается непосредственная работа с файлом по чтению и записью Она строго определяется видом файла. При этом обязательно используются стандартные процедуры чтения и записи READ и WRITE. Для текстовых файлов добавляются процедуры READLN и WRITELN;

е)после окончания использования, файл должен быть закрыт с помощью обращения к процедуре close(<имя файловой переменной>). При выполнении этой процедуры ожидается завершение внешним устройством всех запланированных операций чтения и записи, после чего файл демонополизируется, т.е. становится недоступным данной программе и доступным всем остальным.

**Организация работы с файлами прямого доступа**

Файлы прямого доступа характерны тем, что в любой момент времени можно получить доступ к любому существующему компоненту файла - связать буфер файла с любым существующим компонентом файла.

Все компоненты файла прямого доступа автоматически пронумерованы, начиная с 0. Номер компонента файла типа longint, т.е. теоретически в файле прямого доступа может быть свыше 2 млн. компонентов (правда, пока таких внешних устройств не существует).

В файлах прямого доступа читать можно любой существующий компонент файла, а записывать в файл можно либо на место любого существующего компонента, либо в конец файла.

С файлами прямого доступа можно работать в двух режимах:

* работа с существующим файлом прямого доступа. Открывается режим с помощью процедуры reset(<имя логического файла>). Попытка открыть этот режим для несуществующего файла приводит к ошибке ввода-вывода;
* создание нового файла прямого доступа . Открывается режим с помощью процедуры rewrite(<имя логического файла>). Если до открытия в этом режиме файл существовал, то информация в нем уничтожается.

При работе с файлами прямого доступа можно использовать следующие процедуры и функции (во всех нижеприведенных подпрограммах f – это имя логического файла):

*procedure write(var f:<имя файлового типа>; r:<тип компонента файла>)* - процедура записи в текущий компонент файла значения выражения, указанного при обращении к процедуре.

*procedure read(var f:<имя файлового типа>;var r:<тип компонента файла>)* - процедура чтения текущего компонента файла.

*function eof(var f:<имя файлового типа>):Boolean* – с помощью этой функции можно проверить связан ли буфер с признаком конец файла. Если результат true, то буфер связан с этим признаком, если – false, то буфер не связан с признаком конец файла, а находится на каком-то существующем компоненте файла;

*procedure seek(var f:<имя файлового типа>; n:longint)* - процедура установки на n-ый компонент файла прямого доступа f. При выполнении процедуры буфер файла связывается с n-ым компонентом, если такой компонент существует. В случае, если файл содержит меньшее количество компонентов, произойдет ошибка ввода-вывода ( в этом случае при включенной системе прерываний по вводу-выводу программа снимется с решения);

*procedure reset(var f:<имя файлового типа>) -* процедура установки на начало файла - буфер связывается с первым компонентом файла. Обращения seek(f,0) и reset(f) в процессе работы с файлом прямого доступа эквивалентны;

*procedure truncate(var f:<имя файлового типа>)* - усекает файл, начиная с текущей позиции. После выполнения процедуры в текущую позицию, с которой был связан буфер, будет записан признак конца файла;

*function filepos(var f:<имя файлового типа>):longint* - функция определения текущей позиции файла;

*function filesize(var f:<имя файлового типа>):longint* -функция определения количества компонентов в файле (признак конца файла не учитывается).

**Алгоритм сортировки файла прямого доступа**

***Сортировка файла методом пузырька (обменная сортировка)***

***Спецификация подпрограммы***

1. Назначение: сортировка файла прямого доступа из целых чисел по убыванию (файл обязательно существует; количество компонентов в файле произвольное).Файловый тип определяется ранее подпрограммы следующим образом type tf=file of integer;
2. Имя: sortfilepuz
3. Вид: процедура
4. Перечень параметров:

**Организация работы с последовательными файлами**

Под последовательными файлами понимают такие файлы, в которых искать необходимую информацию можно только последовательно читая компоненты файла, начиная с начала файла, а записывать новые компоненты только в конец файла.

**ЛИТЕРАТУРА:**

1. Семакин И.Г. Основы алгоритмизации и программирования : учебник для спо / И.Г. Семакин, А.П. Шестаков. - 2-е изд., стереотип. - М. : Академия, 2011.
2. Голицына О.Л. Основы алгоритмизации и программирования : учебн. пособие для спо / О.Л. Голицына, И.И. Попов. - 3-е изд., исправ. и доп. - М. : ФОРУМ, 2008.
3. Фризен И.Г. Офисное программирование : учебн. пособие / И.Г. Фризен. - 2-е изд. - М. : Дашков и К, 2010.
4. Хорев П.Б.Технологии объектно-ориентированного программирования : учебн. пособие для вузов / П.Б. Хорев. - 2-е изд., стереотип. - М. : Академия, 2008.

***Лекция № 16***

***Тема: Записи и множества***

**ПЛАН:**

1. Запись.
2. Описание типа Запись.
3. Оператор With и его структура.
4. Множество.
5. Пустое множество.

**ТЕЗИСЫ**

## Записи

Запись – это структура данных, состоящая из фиксированного числа элементов, называемых полями. При этом поля могут быть различных типов.

При выборе элементов применяется конкретное имя, которое дается каждому элементу и не может вычисляться, в отличие от массивов.

Общий вид описания типа «запись»:

Type T = **Record**

<список полей>

**end**;

При обращении к компонентам указывается имя переменной – записи, и через **сочленяющую точку** – имя поля.

В языке Паскаль разрешается использовать тип Record при описании других структур данных и наоборот.

В записях может использоваться не только фиксированная, но и вариантная часть, следующая за фиксированной (здесь не рассматривается).

При обращении к некоторым компонентам записей при большом количестве вложений могут получаться весьма длинные имена. Для сокращенной записи составных имен используется оператор With, имеющий следующую структуру:

**With** <список переменных записи> **Do** <оператор>;

В качестве переменных записи используются имена полей, определенных при описании данной записи, например:

Begin

…

Z[1].FIO:='Иванов Иван Иванович';

With Z[1].Dr do

Begin

D:=5;

M:=’Май’;

G:=’1975’;

End;

…

Оператор With R1,R2,…,Rn do S; эквивалентен записи

With R1 do

With R2 do

…

With Rn S;

## 

## Множества

Множество в математике – это произвольный набор объектов, понимаемый как единое целое. Два множества, отличающиеся только порядком следования элементов, считаются одинаковыми, например:

{A, B, C}, {A, C, B}, {B, C, A}

и т.д., то есть элементы множества не упорядочены.

На вид объектов и их число не накладываются никакие ограничения, но в языке Турбо-Паскаль это понятие существенно уже: в качестве базовых типов допускаются дискретные типы не более чем с 256 различными значениями, то есть типы byte, char, boolean, перечисляемый и диапазон.

Описание типа:

Туре <тип\_множество> = **SET OF** <базовый\_тип>

Var <список\_переменных>: **SET OF** <базовый\_тип>

В качестве констант в этом типе используется изображение или **конструктор множества**, который строится из списка элементов, заключенных в квадратные скобки.

Существует единственное **пустое множество**, которое принадлежит всем типам множеств и обозначается как [ ].

Пусть есть описание:

Var A, B, C: Set of 0..9;

D, E: Set of '0'..'9';

F, G: Set of Boolean;

Тогда:

A:=[1, 3, 5];

D:=['3', '6'..'9'];

F:=[5, 7]; {неверная строка – «не то» множество}

**ЛИТЕРАТУРА:**

1. Семакин И.Г. Основы алгоритмизации и программирования : учебник для спо / И.Г. Семакин, А.П. Шестаков. - 2-е изд., стереотип. - М. : Академия, 2011.
2. Голицына О.Л. Основы алгоритмизации и программирования : учебн. пособие для спо / О.Л. Голицына, И.И. Попов. - 3-е изд., исправ. и доп. - М. : ФОРУМ, 2008.
3. Фризен И.Г. Офисное программирование : учебн. пособие / И.Г. Фризен. - 2-е изд. - М. : Дашков и К, 2010.
4. Хорев П.Б.Технологии объектно-ориентированного программирования : учебн. пособие для вузов / П.Б. Хорев. - 2-е изд., стереотип. - М. : Академия, 2008.

***Лекция № 17***

***Тема: Введение в Delphi***

**ПЛАН:**

1. Delphi. Характеристика продукта.
2. Компилятор в машинный код.
3. Объектно-ориентированная модель программных компонент.
4. Быстрая разработка работающего приложения из прототипов.
5. Возможности Delphi.
6. Назначение языка.
7. Открытая компонентная архитектура.
8. Структурное объектно-ориентированное программирование.

**ТЕЗИСЫ**

**Delphi** - это комбинация нескольких важнейших технологий:

1. Высокопроизводительный компилятор в машинный код
2. Объектно-ориентированная модель компонент
3. Визуальное (а, следовательно, и скоростное) построение приложений из программных прототипов
4. Масштабируемые средства для построения баз данных

Компилятор, встроенный в **Delphi**, обеспечивает высокую производительность, необходимую для построения приложений в архитектуре “клиент-сервер”.

В процессе построения приложения разработчик выбирает из палитры компонент готовые компоненты как художник, делающий крупные мазки кистью.

Еще до компиляции он видит результаты своей работы - после подключения к источнику данных их можно видеть отображенными на форме, можно перемещаться по данным, представлять их в том или ином виде.

***Объектно-ориентированная модель программных компонент***

Основной упор этой модели в **Delphi** делается на максимальном реиспользовании кода.

Никаких ограничений по типам объектов, которые могут создавать разработчики, не существует.

Действительно, все в **Delphi** написано на нем же, поэтому разработчики имеют доступ к тем же объектам и инструментам, которые использовались для создания среды разработки.

***Быстрая разработка работающего приложения из прототипов***

Cреда **Delphi** включает в себя полный набор визуальных инструментов для скоростной разработки приложений (RAD - rapid application development), поддерживающей разработку пользовательского интерфейса и подключение к корпоративным базам данных.

VCL - библиотека визуальных компонент, включает в себя стандартные объекты построения пользовательского интерфейса, объекты управления данными, графические объекты, объекты мультимедиа, диалоги и объекты управления файлами, управление DDE и OLE.

Визуальные компоненты в **Delphi** обладают большей гибкостью.

***Масштабируемые средства для построения баз данных***

Объекты БД в **Delphi** основаны на SQL и включают в себя полную мощь Borland Database Engine. В состав **Delphi** также включен Borland SQL Link, поэтому доступ к СУБД Oracle, Sybase, Informix и InterBase происходит с высокой эффективностью.

***Клиент-серверная версия Delphi***

Она адресована корпоративным разработчикам, желающим разрабатывать высокопроизводительные приложения для рабочих групп и корпоративного применения.

Клиент-серверная версия включает в себя следующие особенности:

* SQL Links: специально написанные драйвера для доступа к Oracle, Sybase, Informix, InterBase
* Локальный сервер InterBase: SQL-сервер для Windows 3.1. СУБД для разработки в корпоративных приложений на компьютере, не подключенном к локальной сети.
* ReportSmith Client/server Edition: генератор отчетов для SQL-серверов
* Team Development Support: предоставляет версионный контроль при помощи PVCS компании Intersolve (приобретается отдельно) или при помощи других программных продуктов версионного контроля
* Visual Query Builder - это средство визуального построения SQL-запросов
* лицензия на право распространения приложений в архитектуре клиент-сервер, изготовленных при помощи **Delphi**
* исходные тексты всех визуальных компонент

***Некоторые особенности Delphi***

*Локальный сервер InterBase* - следует заметить, что этот инструмент предназначен только для автономной отладки приложений. В действительности он представляет из себя сокращенный вариант обработчика SQL-запросов InterBase, в который не включены некоторые возможности настоящего сервера InterBase. Отсутствие этих возможностей с лихвой компенсируется преимуществом автономной отладки программ.

*Team Development Support* - средство поддержки разработки проекта в группе. Позволяет существенно облегчить управление крупными проектами. Это сделано в виде возможности подключения такого продукта как Intersolve PVCS 5.1 непосредственно к среде **Delphi**.

*Высокопроизводительный компилятор в машинный код* - в отличие от большинства Паскаль-компиляторов, транслирующих в p-код, в **Delphi** программный текст компилируется непосредственно в машинный код, в результате чего **Delphi**- приложения исполняются в 10-20 раз быстрее (особенно приложения, использующие математические функции). Готовое приложение может быть изготовлено либо в виде исполняемого модуля, либо в виде динамической библиотеки, которую можно использовать в приложениях, написанных на других языках программирования.

***Открытая компонентная архитектура***

Благодаря такой архитектуре приложения, изготовленные при помощи **Delphi**, работают надежно и устойчиво. **Delphi** поддерживает использование уже существующих объектов, включая DLL, написанные на С и С++, OLE сервера, VBX, объекты, созданные при помощи **Delphi**.

***Библиотека визуальных компонент***

Эта библиотека объектов включает в себя стандартные объекты построения пользовательского интерфейса, объекты управления данными, графические объекты, объекты мультимедиа, диалоги и объекты управления файлами, управление DDE и OLE.

***Структурное объектно-ориентированное программирование***

**Delphi** использует структурный объектно-ориентированный язык (Object Pascal), который сочетает с одной стороны выразительную мощь и простоту программирования, характерную для языков 4GL, а с другой стороны эффективность языка 3GL.

**ЛИТЕРАТУРА:**

1. Семакин И.Г. Основы алгоритмизации и программирования : учебник для спо / И.Г. Семакин, А.П. Шестаков. - 2-е изд., стереотип. - М. : Академия, 2011.
2. Голицына О.Л. Основы алгоритмизации и программирования : учебн. пособие для спо / О.Л. Голицына, И.И. Попов. - 3-е изд., исправ. и доп. - М. : ФОРУМ, 2008.
3. Фризен И.Г. Офисное программирование : учебн. пособие / И.Г. Фризен. - 2-е изд. - М. : Дашков и К, 2010.
4. Хорев П.Б.Технологии объектно-ориентированного программирования : учебн. пособие для вузов / П.Б. Хорев. - 2-е изд., стереотип. - М. : Академия, 2008.

***Лекция № 18***

***Тема: Стандартные компоненты Delphi***

**ПЛАН:**

1. Палитра компонентов
2. Инспектор свойств объектов
3. Особенности сохранения объектов

**ТЕЗИСЫ**

***Стандартные компоненты***

**TMainMenu**  позволяет Вам поместить главное меню в программу. При помещении TMainMenu на форму это выглядит, как просто иконка. Иконки данного типа называют "невидимыми компонентом", поскольку они невидимы во время выполнения программы. Создание меню включает три шага: (1) помещение TMainMenu на форму, (2) вызов Дизайнера Меню через свойство Items в Инспекторе Объектов, (3) определение пунктов меню в Дизайнере Меню.

**TPopupMenu**  позволяет создавать всплывающие меню. Этот тип меню появляется по щелчку правой кнопки мыши.

**TLabel** служит для отображения текста на экране. Вы можете изменить шрифт и цвет метки, если дважды щелкнете на свойство Font в Инспекторе Объектов. Вы увидите, что это легко сделать и во время выполнения программы, написав всего одну строчку кода.

**TEdit** - стандартный управляющий элемент Windows для ввода. Он может быть использован для отображения короткого фрагмента текста и позволяет пользователю вводить текст во время выполнения программы.

**TMemo -**  иная форма TEdit. Подразумевает работу с большими текстами. TMemo может переносить слова, сохранять в Clipboard фрагменты текста и восстанавливать их, и другие основные функции редактора. TMemo имеет ограничения на объем текста в 32Кб, это составляет 10-20 страниц. (Есть VBX и “родные” компоненты Delphi, где этот предел снят).

**TButton**  позволяет выполнить какие-либо действия при нажатии кнопки во время выполнения программы. В Delphi все делается очень просто. Поместив TButton на форму, Вы по двойному щелчку можете создать заготовку обработчика события нажатия кнопки. Далее нужно заполнить заготовку кодом (подчеркнуто то, что нужно написать вручную):

procedure TForm1.Button1Click(Sender: TObject);

begin

**MessageDlg('Are you there?',mtConfirmation,mbYesNoCancel,0);**

end;

**TCheckBox** отображает строку текста с маленьким окошком рядом. В окошке можно поставить отметку, которая означает, что что-то выбрано. Например, если посмотреть окно диалога настроек компилятора (пункт меню Options | Project, страница Compiler), то можно увидеть, что оно состоит преимущественно из CheckBox’ов.

**TRadioButton** позволяет выбрать только одну опцию из нескольких. Если Вы опять откроете диалог Options | Project и выберете страницу Linker Options, то Вы можете видеть, что секции Map file и Link buffer file состоят из наборов RadioButton.

**TListBox**  нужен для показа прокручиваемого списка. Классический пример ListBox’а в среде Windows - выбор файла из списка в пункте меню File | Open многих приложений. Названия файлов или директорий и находятся в ListBox’е.

**TComboBox** во многом напоминает ListBox, за исключением того, что позволяет водить информацию в маленьком поле ввода сверху ListBox. Есть несколько типов ComboBox, но наиболее популярен выпадающий вниз (drop-down combo box), который можно видеть внизу окна диалога выбора файла.

**TScrollbar -** полоса прокрутки**,** появляется автоматически в объектах редактирования, ListBox’ах при необходимости прокрутки текста для просмотра.

**TGroupBox** используется для визуальных целей и для указания Windows, каков порядок перемещения по компонентам на форме (при нажатии клавиши TAB).

**TPanel** - управляющий элемент, похожий на TGroupBox, используется в декоративных целях. Чтобы использовать TPanel, просто поместите его на форму и затем положите другие компоненты на него. Теперь при перемещении TPanel будут передвигаться и эти компоненты. TPanel используется также для создания линейки инструментов и окна статуса.

**TScrollBox** представляет место на форме, которое можно скроллировать в вертикальном и горизонтальном направлениях.

Основное для понимания Инспектора Объектов состоит в том, что он используется для изменения характеристик любого объекта, брошенного на форму. Кроме того, и для изменения свойств самой формы.

***Сохранение программы***

Вы приложили некоторые усилия по созданию программы и можете захотеть ее сохранить. Это позволит загрузить программу позже и снова с ней поработать.

Первый шаг - создать поддиректорию для программы. Лучше всего создать директорию, где будут храниться все Ваши программы и в ней - создать поддиректорию для данной конкретной программы. Например, Вы можете создать директорию MYCODE и внутри нее - вторую директорию TIPS1, которая содержала бы программу, над которой Вы только что работали.

После создания поддиректории для хранения Вашей программы нужно выбрать пункт меню File | Save Project. Сохранить нужно будет два файла. Первый - модуль (unit), над которым Вы работали, второй - главный файл проекта, который "владеет" Вашей программой. Сохраните модуль под именем MAIN.PAS и проект под именем TIPS1.DPR.

**ЛИТЕРАТУРА:**

1. Семакин И.Г. Основы алгоритмизации и программирования : учебник для спо / И.Г. Семакин, А.П. Шестаков. - 2-е изд., стереотип. - М. : Академия, 2011.
2. Голицына О.Л. Основы алгоритмизации и программирования : учебн. пособие для спо / О.Л. Голицына, И.И. Попов. - 3-е изд., исправ. и доп. - М. : ФОРУМ, 2008.
3. Фризен И.Г. Офисное программирование : учебн. пособие / И.Г. Фризен. - 2-е изд. - М. : Дашков и К, 2010.
4. Хорев П.Б.Технологии объектно-ориентированного программирования : учебн. пособие для вузов / П.Б. Хорев. - 2-е изд., стереотип. - М. : Академия, 2008.

***Лекция № 19***

***Тема: События в Delphi***

**ПЛАН:**

1. События в Delphi.
2. Обработка сообщений Windows в Delphi.

**ТЕЗИСЫ**

Объекты из библиотеки визуальных компонент (VCL) Delphi, равно как и объекты реального мира, имеют свой набор свойств и свое поведение - набор откликов на события, происходящие с ними. Список событий для данного объекта, на которые он реагирует, можно посмотреть, например, в Инспекторе Объектов на странице событий.

Поведение объекта определяется тем, какие обработчики и для каких событий он имеет. Создание приложения в Delphi состоит из настройки свойств используемых объектов и создания обработчиков событий.

Простейшие события, на которые иногда нужно реагировать - это, например, события, связанные с мышкой (они есть практически у всех видимых объектов) или событие Click для кнопки TButton. Предположим, что вы хотите перехватить щелчок левой кнопки мыши на форме.

Событийное программирование есть не только в Windows, и данную черту можно реализовать не только в операционной системе.

### Обработка сообщений Windows в Delphi

Всегда возникнет необходимость дополнения или изменения свойств и поведения объектов. В этом случае, так же, как и при создании своих собственных компонент в Delphi, часто требуется обрабатывать сообщения Windows. Поскольку Object Pascal является развитием и продолжением Borland Pascal 7.0, то это выполняется сходным с BP способом.

**ЛИТЕРАТУРА:**

1. Семакин И.Г. Основы алгоритмизации и программирования : учебник для спо / И.Г. Семакин, А.П. Шестаков. - 2-е изд., стереотип. - М. : Академия, 2011.
2. Голицына О.Л. Основы алгоритмизации и программирования : учебн. пособие для спо / О.Л. Голицына, И.И. Попов. - 3-е изд., исправ. и доп. - М. : ФОРУМ, 2008.
3. Фризен И.Г. Офисное программирование : учебн. пособие / И.Г. Фризен. - 2-е изд. - М. : Дашков и К, 2010.
4. Хорев П.Б.Технологии объектно-ориентированного программирования : учебн. пособие для вузов / П.Б. Хорев. - 2-е изд., стереотип. - М. : Академия, 2008.

***Лекции № 20-21***

***Тема: Создание баз данных в Delphi***

**ПЛАН:**

1. Утилита Database Desktop.
2. Настройка BDE.
3. Алиасы.
4. Системная информация утилиты настройки BDE (BDECFG).

**ТЕЗИСЫ**

### Утилита Database Desktop

Database Desktop - это утилита, во многом похожая на Paradox, которая поставляется вместе с Delphi для интерактивной работы с таблицами различных форматов локальных баз данных - Paradox и dBase, а также SQL-серверных баз данных InterBase, Oracle, Informix, Sybase (с использованием SQL Links). Исполняемый файл утилиты называется DBD.EXE, расположен он, как правило, в директории, называемом DBD (при установке по умолчанию). Для запуска Database Desktop просто дважды щелкните по ее иконке.

Имя поля в таблице формата ***Paradox*** представляет собой строку, написание которой подчиняется следующим правилам:

1. Имя должно быть не длиннее 25 символов.
2. Имя не должно начинаться с пробела, однако может содержать пробелы. Однако, если Вы предполагаете в будущем переносить базу данных в другие форматы, разумнее будет избегать включения пробелов в название поля. Фактически, в целях переносимости лучше ограничиться девятью символами в названии поля, не включая в него пробелы.
3. Имя не должно содержать квадратные, круглые или фигурные скобки [], () или {}, тире, а также комбинацию символов “тире” и “больше” (->).
4. Имя не должно быть только символом #, хотя этот символ может присутствовать в имени среди других символов. Хотя Paradox поддерживает точку (.) в названии поля, лучше ее избегать, поскольку точка зарезервирована в Delphi для других целей.

Имя поля в таблице формата ***dBase*** представляет собой строку, написание которой подчиняется правилам, отличным от Paradox:

1. Имя должно быть не длиннее 10 символов.
2. Пробелы в имени недопустимы.

**Настройка BDE**

***Borland Database Engine (BDE)***, создание и редактирование *алиасы* - механизм, облегчающий связь с базами данных. Кроме того, мы изучим, как конфигурировать ODBC драйверы.

### Сущность BDE

Мощность и гибкость Delphi при работе с базами данных основана на низкоуровневом ядре - процессоре баз данных Borland Database Engine (BDE). Его интерфейс с прикладными программами называется Integrated Database Application Programming Interface (IDAPI).

Все инструментальные средства баз данных Borland - Paradox, dBase, Database Desktop - используют BDE. Все особенности, имеющиеся в Paradox или dBase, “наследуются” BDE, и поэтому этими же особенностями обладает и Delphi.

### Алиасы

Таблицы сохраняются в базе данных. Некоторые СУБД сохраняют базу данных в виде нескольких отдельных файлов, представляющих собой таблицы (в основном, все локальные СУБД), в то время как другие состоят из одного файла, который содержит в себе все таблицы и индексы (InterBase).

### Системная информация утилиты настройки BDE (BDECFG)

**System:** Определяет память и технические установки для таблиц в формате Paradox.

**Date:** Определяет установки, используемые при конвертации строковых значений в дату и обратно.

**Time:** Определяет установки, используемые при конвертации строковых значений во время и обратно.

**Number:** Описывает трактовку чисел BDE.

**ЛИТЕРАТУРА:**

1. Семакин И.Г. Основы алгоритмизации и программирования : учебник для спо / И.Г. Семакин, А.П. Шестаков. - 2-е изд., стереотип. - М. : Академия, 2011.
2. Голицына О.Л. Основы алгоритмизации и программирования : учебн. пособие для спо / О.Л. Голицына, И.И. Попов. - 3-е изд., исправ. и доп. - М. : ФОРУМ, 2008.
3. Фризен И.Г. Офисное программирование : учебн. пособие / И.Г. Фризен. - 2-е изд. - М. : Дашков и К, 2010.
4. Хорев П.Б.Технологии объектно-ориентированного программирования : учебн. пособие для вузов / П.Б. Хорев. - 2-е изд., стереотип. - М. : Академия, 2008.