Курс *"Информатика и компьютерная техника"*

Апатова Н. В.
профессор, д.п.н., кафедра информационных систем

Тема 1. Устройство компьютера.

Основные понятия вычислительной техники. История вычислительной техники и языков программирования. Принципы программного управления.

Американский математик Джон фон Нейман предложил в конце 40-годов новую архитектуру компьютера, которая является основной уже в четырех поколениях вычислительной техники. Архитектура - это общее описание вычислительной системы, включая организацию памяти, команды, операции ввода-вывода и т.д.

Каждый компьютер, независимо от размеров и конфигурации, имеет указанные на схеме функциональные блоки : устройство управления, арифметико-логическое устройство, оперативную память и устройства ввода-вывода.

В современных компьютерах редко упоминают отдельно арифметико-логическое устройство и устройство управления. Они объединены словом "процессор".

Процессор - это устройство по переработке данных.

Дж. фон Нейман предложил несколько принципов работы компьютеров, называемых принципами программного управления. Главный из них - принцип единой линейной памяти. "Единой" означает, что и программа, и данные хранятся в одной памяти. Одна и та же ячейка памяти для одной задачи может хранить команду, а для другой - данные. "Линейной" означает, что все ячейки памяти пронумерованы, от 0 до некоторого большого числа N. Номер ячейки называется ее ***адресом***.

В различных компьютерах размеры ячеек были разными. Однако, в середине 60-х годов, когда фирма IBM выпустила семейство вычислительных машин третьего поколения, сложился стандарт единицы измерения памяти - байт.

Байт - это система из 8 элементов, каждый из которых может находиться в двух состояниях, обозначаемых 0 и 1. Элемент байта называется битом.

Каждый байт имеет номер - адрес. Байт может хранить числа и символы. Для запоминания символов используется код ASCII (American standart code for information interchange).

Второй принцип программного управления позволяет при работе программы передавать управление любой команде, а третий принцип позволяет выполнять различные операции (например, сложение) с командами (как с данными).

***Тема 2.* Язык программирования паскаль.**

Алфавит, структура программы, имена, числа, арифметические выражения. Примеры простейших программ, ввод и вывод данных. Логические выражения, таблицы истинности значений логических операций, запись условий на паскале.

Условный оператор, примеры использования условного оператора. Операторы цикла ("до", "пока", "пересчет"), использование операторов цикла при работе с простыми переменными. Скалярные типы данных, интервальный и перечислимый типы. Оператор варианта.

Программа на Паскале состоит из двух частей: описания используемых данных и операторов по их преобразованию. Вторая часть также называется программным блоком.

Общий вид программы:

program имя программы;
label { список меток };
const { список постоянных значений};
type { описания сложных типов данных};
var { описания данных программы};
begin { начало программного блока }
{алгоритм}
end. { конец программы}

Оператор присваивания. Алгоритм преобразования данных на Паскале состоит из операторов - укрупненных команд. Каждый оператор преобразуется транслятором в последовательность машинных команд. Основное преобразование данных, выполняемое компьютером - присваивание переменной нового значения. Общий вид оператора присваивания:

имя переменной := арифметическое выражение;

Неполный условный оператор имеет вид:

IF условие THEN оператор;

Полный условный оператор:

IF условие THEN оператор_1 ELSE оператор_2 ;

Оператор цикла "пока" имеет вид:

while условие ***do*** оператор;

Оператор цикла "до":

repeat оператор ***until*** условие;

Оператор прямого пересчета:

for i := n1 **to** n2 **do** оператор ;

Оператор обратного пересчета

for i := n2 **downto** n1 **do** оператор;

На языке Паскаль оператор варианта имеет вид:

case *индекс варианта* **of**
метка 1 **:** оператор 1;
метка 2 : оператор 2;
. . .
метка n : оператор n;
else оператор n + 1
end ;

***Тема 3.* Вычисления с заданной точностью.**

Рекуррентные вычисления, вычисление значения корня квадратного, вычисление чисел Фибоначчи. Вычисление значения функции с заданной точностью.

Вещественные числа, используемые в Паскале, занимают шесть байтов в памяти компьютера. Следовательно, длина числа (количество его цифр) ограничена размером предоставляемой памяти. При выводе вещественного числа в форме с плавающей точкой оно содержит 11 цифр - приближенное значение точного результата.

Приближенным числом а называется число, незначительно отличающееся от точного А и заменяющее последнее в вычислениях.

```
program {вычисление корня квадратного с заданной точностью};
var A, eps, x0, x1: real;
begin
write ('введите положительное число A =');
readln(A);
write ('введите точность вычислений - правильную десятичную дробь ');
readln(eps);
x0 := 1;
x1 := 1/2 * ( x0 + A/x0 );
while abs ( x1 - x0 ) > eps do
begin
x0 := x1;
x1 := 1/2 * ( x0 + A/ x0 )
end;
write (' корень квадратный из числа ' , A, ' = ' , x1)
end.
```

Тема 4. Алгоритмы обработки таблиц.

Массивы, суммирование, поиск наибольшего элемента в массиве, упорядочивание элементов массива по возрастанию. Алгоритмы поиска в массиве. Алгоритмы обработки матриц: нормы матриц, след матрицы, транспонирование матриц. Сложение и умножение матриц, умножение вектора на матрицу. Алгоритмы первичной статистической обработки данных.

Массив - именованный набор с фиксированным количеством однотипных данных.

Вычисление **суммы элементов** массива состоит из трех основных этапов :
1) ввод данных; 2) вычисление суммы; 3) печать результатов.

```
program E15;
const n = 7;
var a : array [ 1 .. n ] of real; S : real; i : integer;
begin
write(' вводите элементы массива - ' , n, ' вещественных чисел через пробел');
for i : = 1 to n do
read (a [ i ] );
S := 0;
for i := 1 to n do
S := S + a [ i ] ;
writeln;
write (' сумма элементов массива S = ' , S )
end.
```

При обработке массивов рассматривается следующий пример, связанный с вопросами ***экономики окружающей среды***: определение цены и оценка природных ресурсов. Адекватный учет цены/оценки природных ресурсов в проекте, получаемых в результате реализации проекта выгод, издержек и ущербов существенно влияет на решение о степени эффективности проекта. Последнее соотношение в неявном виде включает в себя экологическую информацию в виде экологических выгод и экологических затрат. Выделим отдельно экологическую составляющую в виде суммы экологических издержек и экологических выгод (E_t). Она может быть как положительной (проект дает большой природоохранный эффект), так и отрицательной (реализация проекта связана со значительным экологическим ущербом). Тогда формула имеет вид:

$$NPV = \sum_{t=0}^{n} \frac{B_t - C_t \pm E_t}{(1+r)^2}$$

Данное соотношение - основное для определения экономической эффективности проекта/программы с учетом экологической составляющей и фактора времени. Программа расчетов строится аналогично приведенному выше примеру.

Тема 5. Процедуры и функции.

Понятие подпрограммы. Процедуры без параметров, процедуры с параметрами, параметры-значения и параметры-переменные. Подпрограммы-функции. Рекурсия, рекурсивные процедуры и функции.

Подпрограмма - это специальным образом оформленный алгоритм, который может многократно использоваться при решении более общей задачи.

Процедура - подпрограмма, имеющая любое количество входных и выходных данных. Процедура может быть описана без параметров и с параметрами. Параметры - это данные из заголовка процедуры, как передаваемые ей на обработку, так и получаемые в виде результатов. Таким образом, с помощью

паарметров происходит обмен информацией между процедурой и вызывающей ее программой.

Процедуры без параметров.Описание процедуры имеет вид:

procedure имя;
{описание локальных переменных }
begin
{операторы}
end;

Процедуры с параметрами. Для удобства передачи данных в процедуру и получения из нее результата используются *формальные* и *фактические параметры*. **Формальные** - условные обозначения в описании процедуры - описываются в ее заголовке. **Фактические** - с которыми требуется выполнить процедуру - перечисляются при вызове процедуры. Формальные и фактические параметры должны соответствовать по количеству, типу и порядку следования. Формальные параметры, описанные в заголовке процедуры, больше нигде не описываются. Их описание похоже на описание данных в разделе переменных и может также содержать слово **var.** Слово var в заголовке процедуры ставится перед теми параметрами, имена которых соответствуют выходным данным. Фактические параметры, соответствующие формальным, перед которыми стоит слово var, могут быть только именами переменных. Перед именами формальных переменных, являющимися входными данными процедуры, можно слово var не указывать. Если перед формальным параметром в заголовке процедуры нет слова var, то ему может соответствовать формальный параметр, имеющий вид выражения соответствующего типа. Если для входных данных процедуры при описании формальных параметров указано слово var, то им также соответствуют фактические параметры - имена переменных.

Подпрограммы-функции. Подпрограмма, имеющая единственный результат, может быть оформлена как функция. Описание функции имеет вид:

function имя_функции (описание входных данных) : тип_результата;
{описания локальных переменных }
begin
{ операторы }
имя_функции := результат;
end;

Тема 6. Обработка строк.

Строковые и символьные данные, операции над строками. Функции и процедуры обработки строк. Пословный перевод с одного языка на другой. Разработка программы "частотный словарь".

Строка - это ограниченная апострофами последовательность любых символов.

Длина строки, обрабатываемой в Паскале, не должна превышать 255 символов (апострофы не считаются). Это связано с тем, что в конце строки, в

дополнительном байте, хранится ее длина - количество символов, а наибольшее целое число, которое может быть записано в байте - 255. Если требуется обработать текст, длина которого большее 255 знаков, то надо использовать массив строк.

Описание строки имеет вид:

var x: **string** [20];

Оператор присваивания для строковых данных имеет вид:

имя_строковой _переменной := строковое выражение;

Функция длины строки выдает количество символов строки:

length (строковое_выражение)

Функция копирования называется также "вырезкой". Она позволяет скопировать одну область памяти в другую. Для копирования необходимо указать строковое выражение, из значения которого выделяется часть, а также начальный номер символа и количество символов копируемой части :

copy (строковое выражение, начальный номер символа, количесвто символов)

Функция поиска определяет, с какой позиции (номера символа) одна строка (подстрока) содержится в другой (данной строке). Если такое вхождение подстроки в строку имеет место, то результат работы функции - номер символа в исходной строке, с которого начинается подстрока. Если вхождения нет, то результат - ноль. Аргументы функции могут быть строковыми выражениями.

pos (подстрока, исходная строка)

В одну строку можно вставить другую строку, указав номер символа, начиная с которого осуществляется вставка. Входные данные процедуры - вставляемая строка, исходная строка и целочисленное выражение, задающее позицию вставки. Строки также могут быть заданы строковыми выражениями. Результат работы процедуры помещается в исходную строку, строка при этом "расширяется". Если длина вставки совместно с длиной исходной строки превышает допустимую длину исходной строки, то вставка укорачивается справа до допустимой длины.

insert (вставляемая строка, исходная строка, целочисленное выражение);

. Часть строки можно удалить, строка при этом "сжимается". Для удаления необходимо указать строку (виде строкового выражения), начальный номер удаляемой части строки, количество удаляемых символов. Процедура удаления вызывается следующим образом:

delete (строка, начальный номер, количество символов);

Тема 7. Информационное моделирование, файлы.

Компьютерное моделирование, понятие объекта и его модели. Виды моделей: математическая, физическая, информационная. Принципы разработки моделей, технологическая цепочка построения компьютерной модели. Понятие о базах данных, связи между данными. Двумерные файлы, иерархические и сетевые структуры. Множества в Паскале. Структуры данных в Паскале: записи. Файловый тип данных, создание файла, запись в файл, чтение из файла, расширение файла. Особенности работы с файлами в турбо-паскале.

Информация, как сведения об объекте или явлении, проявляется в виде конкретных данных.

Человек может создавать и выполнять алгоритмы в реальной жизни, оперируя с реальными объектами. Компьютер всегда работает с моделью.

Модель - это образ объекта, его описание на некотором языке.

Существуют, как известно, и физические модели, являющиеся копией или аналогом объекта, и математические, описывающие в виде формул зависимости между различными свойствами объекта. Для компьютерного моделирования объект должен быть представлен в такой форме, чтобы можно было реализовать алгоритмы его обработки, т.е. на языке, понятном компьютеру. Следовательно, компьютер работает с *информационной моделью*, представленной как набор фактов и правил их обработки для данного объекта. Факты имеют вид данных, правила - последовательности команд алгоритма.

База данных - это совокупность данных и связей между ними.

Каждому элементу, рассматриваемому как объект, свойство или атрибут в информационной системе, может соответствовать несколько других объектов, свойств или атрибутов. Подобная структура имеет несколько уровней. Каждый ее элемент может быть связан с несколькими другими, находящимися на нижнем уровне и только с одним из более верхнего уровня. Такая структура называется **древовидной** или деревом. Каталоги, подкаталоги и содержащиеся в них файлы образуют древовидную структуру. Такую же структуру имеют практически все предприятия, министерства, ведомства и армия. Все, где есть единоначалие, может быть представлено в виде древовидной модели. Для представления деревьев в памяти компьютера существуют различные алгоритмы. Как уже известно из организации файлов в операционной системе (объединение их в каталоги, задание пути к файлу), поиск по дереву, т.е. данных, организованных в древовидные структуры, происходит быстро, намного быстрее, чем последовательный просмотр всех имеющихся данных. Базы данных с древовидной структурой называются **иерархическими**. Хранящиеся в них данные можно просматривать по-разному: двигаться от верхней точки дерева (корня) вниз по ветви до конечной точки (листа дерева), просматривать данные одного уровня, осуществить переход от данного нижнего уровня к верхнему. Возможности базы зависят от ее назначения.

Природоохранные мероприятия можно сгруппировать на основе построения структурно- целевой модели, т.е. дерева целей (рис. 1).

В исходном варианте структурно-целевой модели для трех отраслей, имеющихся на двух территориях запланированы природоохранные мероприятия; сокращенный вариант предполагает, что мероприятия первой отрасли есть на обеих территориях, второй отрасли — только на первой территории, третьей отрасли —

только на второй территории. На основе структурно-целевой модели можно построить реберный граф финансовых связей.

Производственные отношения между предприятиями часто имеют сложный характер, который отражается в сетевых структурах. **Сеть** - многоуровневая структура, каждый ее элемент может быть связан как с несколькими элементами нижнего уровня, так и с несколькими элементами верхнего уровня. Например, магазин получает товары от нескольких поставщиков. Один и тот же товар может быть поставлен разными поставщиками, в то же время один поставщик может поставлять несколько товаров. Если рассмотреть связи "товар-поставщик", то они представляют собой сеть.

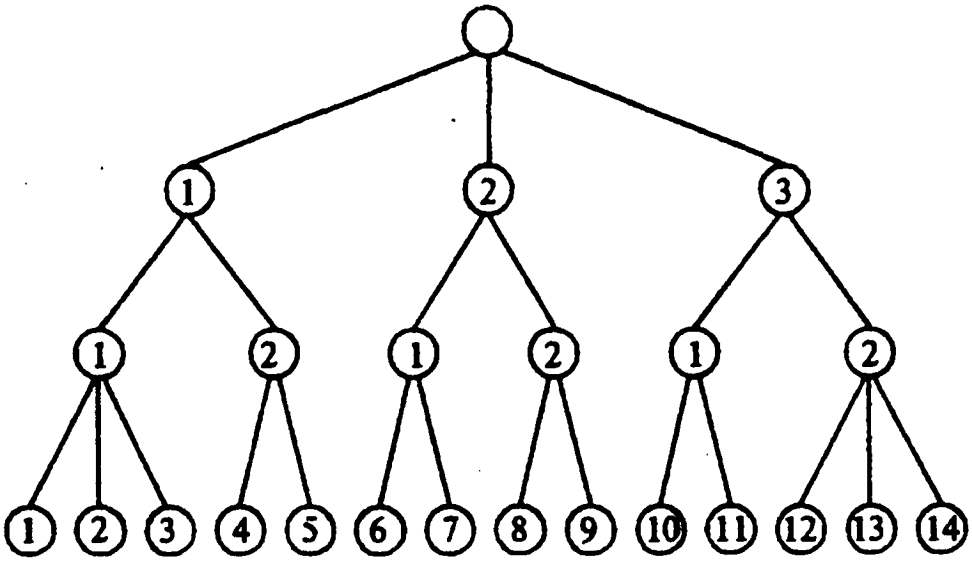

Рис. 1. Структурно-целевая модель группировки природоохранных мероприятий:

корень - цель программы;

1-ый уровень - отрасль;

2-ой уровень - территория;

3-ий уровень - мероприятия.

Реляционные базы данных. Для реализации различных связей между данными используются **таблицы**, изменяющиеся в зависимости от условий (отношений). Такую базу можно представить как набор столбцов одной длины, хранящихся в разных файлах и объединяемые в таблицы, или как двумерный файл (ведомость), которую можно разрезать на столбцы или группы столбцов, а потом по-разному склеить. Такой подход позволяет построить и древовидные, и сетевые модели. Объекты, к которым относятся атрибуты нескольких столбцов, всегда будут располагаться в одном и том же порядке, такой же порядок будет и у атрибутов. Каждой связи или последовательности связей соответствует строка таблицы.

Запрос к базе данных. Информационные системы делятся на документальные и фактографические. **Фактографические системы** отвечают на конкретные вопросы, выдавая имеющиеся сведения об объектах в различной комбинации значений их атрибутов. В запросе к фактографической системе точно указано, какие свойства

объектов служат критерием отбора данных из базы. Как правило, выдается список объектов с указанными в запросе свойствами.

Документальные системы предназначены для библиотек, они используют для каждого хранимого документа (статьи, книги, журнала) набор ключевых слов. Пользователь формирует запрос в виде набора таких слов, причем может указать, что слова должны быть все в одном документе, или хотя бы одно из них, или какого-то слова быть не должно.

Работа с файлами в паскале предполагает выполнение двух операций: запись в файл и чтение из файла. Запись в файл означает помещение в него новых данных. Файл размещается на носителе, как правило, на магнитном диске. Данное для занесения в файл формируется в оперативной памяти как значение некоторой переменной. Операцией записи в файл это данное копируется из оперативной памяти во внешнюю. Следовательно, форма представления данного, его тип и структура должны быть одинаковы и для записей файла. и для переменной , из которой это данное копируется.

Запись - последовательность байтов на носителе, ограниченная с двух сторон специальными признаками. Файл - это именованный набор однотипных данных на диске.

Описание файла. Описание файла может быть в разделе типов или в разделе переменных. Пусть файл f состоит из целых чисел. Его описание имеет вид:

var f : **file of integer**; a: **integer**;

где a - компонента файла, данного того же типа, что и записи файла. Тип данных файла указывается после слова **of** в описании, это может быть числовой или символьный тип, массив или запись. Сложный тип записи файла необходимо предварительно описать в разделе **type**.

Установление соответствия между логическим и физическим именами файла. Логическое имя - имя переменной из раздела **var,** под этим именем файл присутствует в программе. Физическое имя - имя из каталога оглавления диска.

Оператор установления соответствия между именами файлов:

assign (логическое имя файла, физическое имя);

Физическое имя заключено в апострофы, оно появится в оглавлении, в том же, где находится файл **turbo.exe**.

Открытие файла для операции "запись". Это действие выполняется оператором:

rewrite (f);

При открытии файла для занесения в негоданных на диске появляются две специальные записи : начало файла, содержащее физическое имя, и признак конца файла. Каждое открытие файла для записи означает создание файла. Если для операции "запись" открыть файл с уже имеющимися данными, то все данные файла пропадут. Поэтому открывать для записи можно только файлы с новыми именами (физическими).

При занесении данных в файл они будут размещаться между именем и признаком конца файла, каждое новое данное будет размещаться перед признаком конца файла. Файл может содержать произвольное количество данных. Ограничение размера файла никак в программе не оговаривается. В оперативной памяти достаточно одной области, совпадающей по формату с записями файла, а на диске файл может быть такого размера, сколько имеется свободного пространства на момент его создания.

Запись данных в файл производит оператор:

write (f , a) ;

Операция чтения. Для чтения данных из файла его необходимо описать, установить соответствие между логическим и физическим именем, открыть для чтения и считывать данные. Первые два шага - описание и установление соответсвтия имен - такие же, как и для операции записи. Если с файлом выполняются различные операции, то перед выполнением следующей его необходимо **закрыть** оператором

close (f) ;

Открытие файла для чтения производится оператором:

reset (f) ;

Читать данные из файла позволяет оператор:

read (f, a);

После создания файла и нескольких преобразований может быть неизвестно количество его записей. Поэтому при чтении данных из файла удобно использовать специальную функцию, контролирующую признак конца файла. Эта функция принимает значение "истина", если встречен признак конца файла, и "ложь", если прочитана другая запись. Т.к. при открытии файла для чтения уже считывается первая его запись, содержащая имя файла, то можно поставить контроль признака конца файла, даже не считав ни одной записи оператором ***read(f,a);***

Функция обработки признака конца файла :

eof (f)

Поскольку количество записей в файле неизвестно, то использовать при чтении данных файла цикл-пересчет нельзя, используется цик-пока. Его заголовок:

while not eof (f) ***do***

Тема 8. Компьютерная графика.

Графические средства персонального компьютера, графические примитивы. Построение изображений на экране дисплея, график функции. Деловая графика: круговые и линейные диаграммы, гистограммы.

При построении изображений используются так называемые графические примитивы - базовые конструкции, и их модификации. К графическим примитивам относятся : точка, отрезок, дуга.

Точка. Для выдачи точки на экран используется вызов процедуры

PutPixel (x, y, номер_цвета);

где x , y - координаты точки.

С помощью этого оператора можно построить график функции, например ***графики спроса и предложения.***

```
program IS_LM;
uses Graph, Crt;{подключение библиотек графических процедур и ДОС}
var flag: integer; {flag - переключатель графиков спроса и предложения}
procedure Init; { установка графического режима }
var gr, gm : integer;
begin
gr := 0; {автоматическое распознавание типа дисппея }
InitGraph (gr, gm, ' ' ); {файл egavga.bgi находится в одном каталоге с turbo.exe}
if GraphResult <> grOk then Halt (1); { обработка результата установки }
end;
```

```
function f ( x: real) : real;
begin
if flag = 1 then f := 1/x + 2{ вычисление функции спроса }
else f::= - 1/ (x - 5) +2 { вычисление функции предложения }
end;
procedure grafic;
var xmax, ymax, x1, y1 : integer; x, y, a, b: real;
begin
xmax := 640; ymax := 480; a := - 1; b := 5;
for x1 := 0 to xmax do
begin
x := a + x1 * ( b - a ) / xmax;
y := f ( x ); { вызов подпрограммы-функции }
y1 := trunc ( y * xmax / ( b - a ));
y1 := ymax div 2 - y1;
PutPixel ( x1, y1, flag + 3); { выдача точки зеленым цветом }
end
end;
begin { главная программа }
Init; { вызов процедуры установки графического режима}
flag := 1; {построение функции спроса бирюзовым цветом}
Grafic; { вызов процедуры построения графика}
flag := 2 {построение функции предложения малиновым цветом}
Grafic; { вызов процедуры построения графика}
repeat until keypressed
end.
```

Тема 9. Ссылки, списочные структуры.

Ссылочный тип данных. Списки, создание и просмотр списка, сортировка списка. Стеки и очереди, использование стеков при вызове подпрограмм. Двоичные деревья, просмотр и включение новых данных.

Тема 10. Арифметические и логические основы вычислительной техники.

Системы счисления, перевод чисел из одной системы счисления в другую, компьютерная арифметика. Логические основы вычислительной техники: основные логические элементы, одноразрядный двоичный сумматор.

Тема 11. Поколения вычислительной техники.

Понятие поколений вычислительной техники. Компьютеры 4-го поколения: супер-ЭВМ. Компьютеры 4-го поколения: персональные ЭВМ. Операционная система MS DOS: файлы. каталоги, основные команды. Проект вычислительной системы 5-го поколения, компьютеры потоков данных, нейрокомпьютеры.

Операционная система - это совокупность программ, которые контролируют, осуществляют помощь и организуют работу всех программ и ресурсов компьютера.

Самой распространенной в мире операционной системой является разработанная фирмой Microsoft дисковая операционная система MS-DOS для компьютеров, совместимых с IBM PC.

Процесс загрузки происходит следующим образом. **BIOS** - базовая система ввода-вывода, расположенная в ПЗУ (постоянном запоминающем устройстве) или в оперативной памяти, проверяет работоспособность памяти и подключенного к компьютеру оборудования. Затем она ищет программу начальной загрузки системы и передает ей управление. **Загрузчик операционной системы** - программа начальной загрузки - размещается всегда в первом секторе нулевой дорожки дискеты (BOOT - секторе). Если дискета не найдена, программа ищется на такой же дорожке твердого диска. Программа начальной загрузки считывается средствами BIOS в оперативную память, всегда в одно и то же место. BIOS запускает эту программу и она начинает переписывать в память (загружать) файлы ввода-вывода системы. **Системные файлы ввода-вывода** - программы дополнительных средств ввода-вывода, после загрузки они находятся постоянно в оперативной памяти.

Тема 12. Компьютерные сети.

Локальные сети. INTERNET. Язык HTML. Разработка WEB - страниц.

В компьютерной сети в узлах сетевой структуры находятся компьютеры, а соединяющие их линии могут быть или проводами, или использующими средства беспроволочной связи (напрмер, спутниковой) коммуникациями. При включении компьютера в сеть он получает название рабочей станции. Существующие компьютерные сети можно разделить на два класса : локальные сети и глобальные.Главный компьютер в сети, к которому имеют доступ рабочие станции, называется ***сервером.*** Он управляет сетью или ее участком

Локальная сеть используется для передачи данных на небольшие расстояния, например, в пределах одной комнаты, одного здания, или одного предприятия.

Объединенные в единую сеть локальные сети называются ***глобальной сетью.*** Охватывающая практически все страны мира компьютерная сеть носит имя **Internet.** Для связи используется телефонная сеть, подключение к которой компьютера осуществляется через специальное устройство - ***модем.***

Каждый узел сети имеет свой адрес. Адреса указываются для использования самой распространенной и доступной функции Internet - ***электронной почты.***

Сеть Internet объединяет несколько сетей, поэтому для согласования их работы необходимы специальные соглашения, называемые ***протоколом.***

Тема 13. Интеллектуальные компьютерные системы

Виды систем искусственного интеллекта, основные проблемы искусственного интеллекта. Представление знаний в системах искусственного интеллекта: продукционные правила, семантические сети, фреймы. Прямой и обратный логический вывод. Обработка естественного языка.

Интеллект можно представить как совокупность фактов и способов их применения для достижения цели.

Искусственный интеллект - программная система, имитирующая на компьютере мышление человека

Наиболее простым способом для представления знаний является система **продукционных правил (эвристик)**, когда знания имеют вид предложения "ЕСЛИ

... ТО ...". Часть правила "ЕСЛИ" называется посылкой, а часть "ТО" - выводом или действием. Посылка может состоять из нескольких фактов или условий, вывод - это факт или действие.

Для получения новых знаний в системе продукционных правил, или обоснования некоторого факта, используются прямой и обратный логические выводы.

Прямой вывод служит для прогноза и называется индуктивным. Он отвечает на вопрос "Что будет, если ... ?".

Обратный вывод называется дедуктивным и обосновывает или объясняет имеющиеся факты.

Для представления знаний из некоторой области научной или практической деятельности человека используется также **семантическая сеть** - сетевая структура, позволяющая связать различные понятия, установив связи между ними. Главное назначение семантической сети - правильно отразить *смысл* каждого понятия.

Третий способ представления знаний - создание **фреймов** (рамок). В центр фрейма помещается объект, вокруг него, с соответствующими связями - его атрибуты. Объединяясь через однотипные атрибуты, фреймы также образуют сеть. Отличием фрейма от семантической сети является то, что фрейм может также содержать процедуры (обращения к программам) обработки элементов фрейма.

Список литературы

1. Апатова Н.В. Подготовка пользователя современного компьютера. Учебное пособие. -Симферополь: СГУ, 1997 г.
2. Апатова Н.В. Основы алгоритмизации и программирования. Учебное пособие. - Симферополь: СГУ, 1998 г.
3. А.Ф.Верлань, Н.В.Апатова. Информатика. Учебник для учащихся 10-11 классов средней школы (на укр. языке). - Киев: КВАЗАР-МИКРО, 1998.
4. Вирт Н. Алгоритмы и структуры данных. - М.: Мир, 1989.