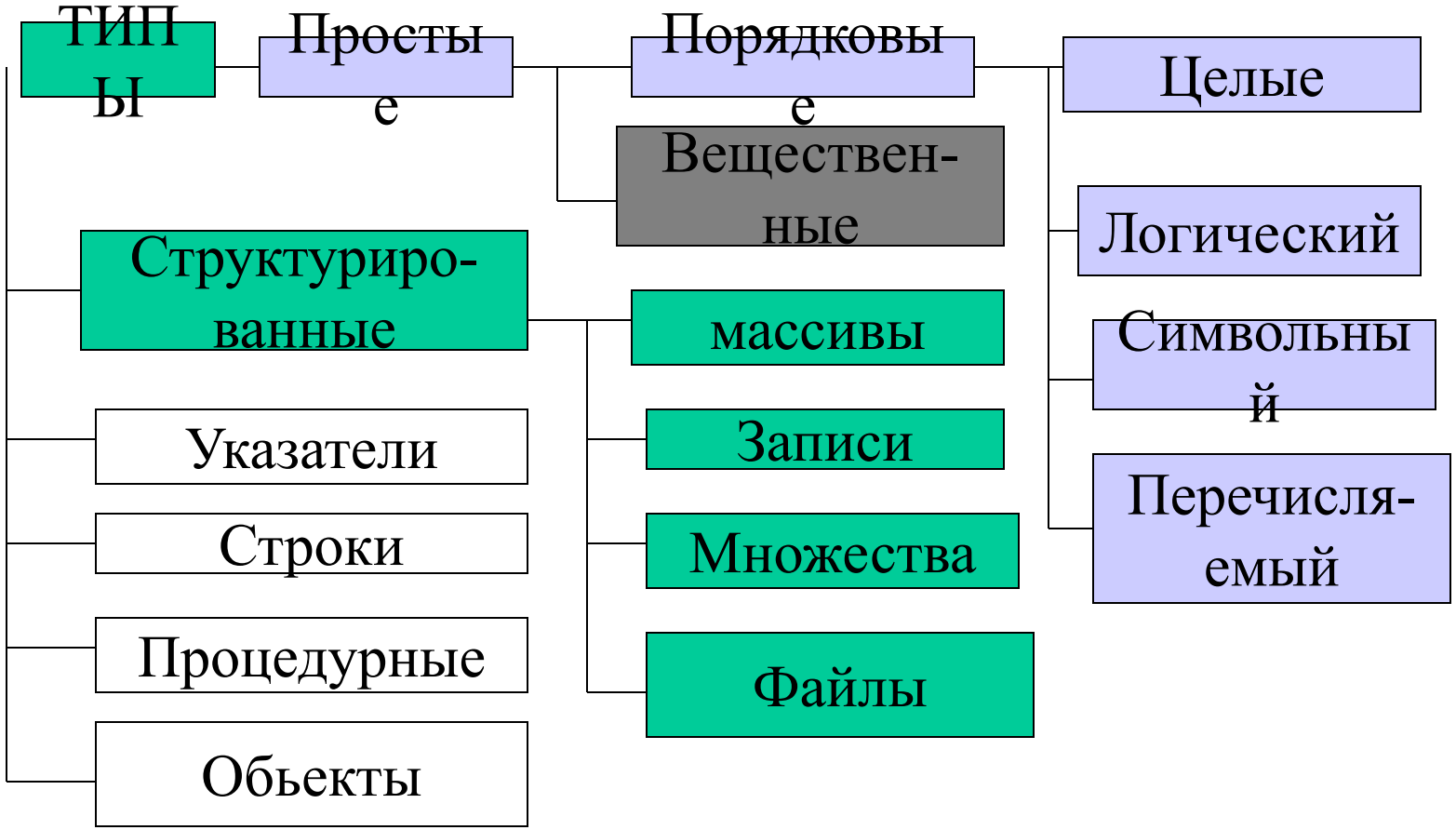
ТИПЫ ДАННЫХ
(ПРОСТЫЕ ТИПЫ)


ТИПЫ
Простые
Порядковые
Целые
Вещественные
Логический
Структурированные
массивы
Символьный
Указатели
Записи
Перечисляемый
Строки
Множества
Процедурные
Файлы
Обьекты

<u>Концепция типа данных.</u>

Все данные, используемые в программе, должны быть предварительно определены,

т.е. для каждого данного надо обозначить

- · **имя,**
- · **характер и диапазон изменения значений,**
- · **требуемую память для размещения,**
- · **набор допустимых к ним операций.**

Для определения данных можно использовать 2 способа объявления данных в программе:

- **посредством задания их значений.**
- **путем задания их типов - для данных изменяющих свои значения при выполнении программы.**

В любом алгоритмическом языке каждая константа, переменная, выражение или функция бывают определенного типа.

В языке ПАСКАЛЬ существует правило: тип явно задается в описании переменной или функции, которое предшествует их использованию.

Обязательное описание типа приводит к избыточности в тексте программ, но такая избыточность является важным вспомогательным средством разработки программ и рассматривается как необходимое свойство современных алгоритмических языков высокого уровня.

Тип определяет:

- возможные значения переменных, констант, функций, выражений, принадлежащих к данному типу;
- внутреннюю форму представления данных в ЭВМ;
- операции и функции, которые могут выполняться над величинами, принадлежащими к данному типу.

Простые порядковые типы (целые, логический, символьный, перечисляемый). Имеют общие свойства:

1. все возможные значения порядкового типа представляют собой ограниченное упорядоченное множество;
2. к любому порядковому типу могут быть применены стандартные функции:

- **Ord.** В качестве результата возвращает порядковый номер конкретного значения в данном типе; **для целых – само целое число,** для **логического типа – 0,1, для символьного типа 0-255 , для перечисляемого типа 0-65535.**
- **Pred** и **Succ**, которые возвращают предыдущее и последующее значения соответственно; **c:=‘5’, pred(c) –4, succ(5) –6.**
- **Low** и **High,** которые возвращают наименьшее и наибольшее значения величин данного типа.

Если K:integer, то **Low(k) - -32768, High(k) – 32767**

Целые типы.

Целые типы определяют константы, переменные и функции, значения которых реализуются множеством целых чисел, допустимых в данной ЭВМ.

В Турбо Паскале используются **пять целочисленных типов данных** , наиболее распространенным среди которых является тип **INTEGER**

Целые типы

тип	диапазон значений	требуемая память
Shortint	-128 .. 127	1 байт
Integer	-32768 .. 32767	2 байта
Longint	-2147483648 .. 2147483647	4 байта
Byte	0 .. 255	1 байт
Word	0 .. 65535	2 байта

Над целыми можно выполнять следующие **арифметические операции:**

сложение (+),

вычитание (-),

умножение (*),

деление (/),

деление нацело (div),

получение остатка от деления (mod).

Результат арифметической операции над целыми операндами есть величина целого типа.

Результат выполнения операции деления целых величин есть целая часть частного.

Результат выполнения операции получения остатка от деления - остаток от деления целых.

<u>Например:</u>

деление нацело (div).

получение остатка от деления (mod).

17 div 2 = 8

3 div 5 = 0

17 mod 2 = 1

3 mod 5 = 3.

Над целыми можно выполнять следующие **операции отношения**:

- **равенство =,**
- **неравенство <>,**
- **больше или равно >=,**
- **меньше или равно <=,**
- **больше >, меньше < .**

Операции отношения, примененные к целым операндам, дают результат логического типа **TRUE или FALSE** (истина или ложь).

К аргументам **целого типа** применимы следующие стандартные (встроенные) функции,

1. результат выполнения имеет **целый тип**:

Abs(X) - абсолютное значение X

Sqr(X), - X в квадрате

Succ(X) - X+1

Pred(X) - X-1.

2. результат выполнения имеет **действительный тип**:

Sin(X) - синус X

Cos(X) - косинус X

ArcTan(X) - арктангенс угла, заданного в радианах,

Ln(X) - логарифм натуральный

Exp(X) - экспоненту X

Sqrt(X) - корень квадратный X.

Целую величину можно проверить на нечетность при помощи функции **Odd(X).** Результат имеет значение истина, если аргумент нечетный, и значение ложь, если аргумент четный:

X=5 Odd(X)=TRUE

X=4 Odd(X)=FALSE

Для быстрой работы с целыми числами определены процедуры:

Inc(X)	**X:=X+1**
Inc(X,N)	**X:=X+N**
Dec(X)	**X:=X-1**
Dec(X,N)	**X:=X-N**

Примеры объявления целочисленных данных:

```
const
 step=1;
 mm:word=65500;

var
 a:integer;
 c,d:byte;
 f:shortint;
```

<u>Логический тип (Boolean)</u>

Определяет данные, которые могут принимать логические значения **TRUE и FALSE.**

Логический тип определен таким образом, что **FALSE < TRUE.**

К булевским операндам можно применять все операции отношения.

Для данных **логического типа** определены следующие логические операции

NOT - отрицание: NOT(True)=false;

OR - логическое сложение(ИЛИ):

True OR False = True;

AND - логическое умножение(И):

False AND True = False;

XOR - исключающее ИЛИ:

True XOR True = False; True XOR False = True;

В ТУРБО ПАСКАЛЬ введены еще разновидности логического типа:

ByteBool, WordBool и LongBool, которые занимают в памяти ЭВМ

один, два и четыре байта соответственно.

<u>Пример объявления логических данных:</u>

```
const
a=true;
var
b:boolean;
```

<u>Символьный тип</u> (Char)

Значение символьной переменной или константы - это **один** символ из допустимого набора.

Символьная константа может записываться в тексте программы тремя способами:

- как один символ, заключенный в апострофы, например: 'А' 'а' 'Ю' 'ю';
- с помощью конструкции вида #K, где K - код соответствущего символа, при этом значение K должно находиться в пределах 0..255;
- с помощью конструкции вида ^C, где C - код соответствующего управляющего символа.

К величинам символьного типа **применимы все операции отношения**.

Для величин символьного типа определены две функции преобразования **Ord(C) Chr(K).**

Ord(C) определяет порядковый номер символа С в наборе символов,

Chr(K) определяет по порядковому номеру К символ, стоящий на К-ом месте в наборе символов.
Порядковый номер имеет **целый тип**.
К аргументам символьного типа применяются функции, которые определяют предыдущий и последующий символы: **Pred(C) Succ(C).**

Pred('F') = 'E' ; Succ('Y') = 'Z' .

При отсутствии предыдущего или последующего символов значение соответствующих функций не определено.

Для литер из интервала 'a'..'z' применима функция **UpCase(C)**, которая переводит эти литеры в верхний регистр 'A'..'Z'.

Пример объявления переменной и константы символьного типа:

```
var
   cimv:char;

const
   vsym='A';
```

Перечисляемый тип данных

Представляет собой ограниченную упорядоченную последовательность скалярных констант.

Значение каждой константы задается ее именем.

Имена отдельных констант отделяются друг от друга запятыми, а вся совокупность констант, составляющих данный перечисляемый тип, заключается в круглые скобки.

Обьединяется в одну группу в соответствии с каким - либо признаком вся совокупность значений, составляющих перечисляемый тип.

Например,

перечисляемый тип **Rainbow(РАДУГА)** объединяет скалярные значения:
RED, ORANGE, YELLOW, GREEN, LIGHT_BLUE, BLUE, VIOLET (КРАСНЫЙ, ОРАНЖЕВЫЙ, ЖЕЛТЫЙ, ЗЕЛЕНЫЙ, ГОЛУБОЙ, СИНИЙ, ФИОЛЕТОВЫЙ).

Traffic_Light (СВЕТОФОР) объединяет скалярные значения **RED, YELLOW, GREEN (КРАСНЫЙ, ЖЕЛТЫЙ, ЗЕЛЕНЫЙ).**

Перечисляемый тип описывается в разделе описания типов, который начинается со служебного слова type, например:

```
type
  Rainbow = (RED, ORANGE, YELLOW, GREEN,
  LIGHT_BLUE, BLUE, VIOLET);
```

Каждое значение является константой своего типа и может принадлежать только одному из перечисляемых типов, заданных в программе. Так, перечисляемый тип **Traffic_Light** не может быть определен в одной программе с типом **Rainbow**, так как оба типа содержат одинаковые константы.
Описание переменных, принадлежащих к скалярным типам, которые объявлены в разделе описания типов, производится с помощью имен типов.

Например:

type Traffic_Light= (RED, YELLOW, GREEN);
var

Section: Traffic_Light;

Это означает, что переменная **Section** может принимать значения **RED, YELLOW или GREEN.**

Переменные перечисляемого типа могут быть описаны в разделе описания переменных, например:

var

Section: (RED, YELLOW, GREEN);

При этом имена типов отсутствуют, а переменные определяются совокупностью значений, составляющих данный перечисляемый тип.

К переменным перечисляемого типа может быть применим оператор присваивания:

Section:= YELLOW;

Упорядоченная последовательность значений, составляющих перечисляемый тип, автоматически нумеруется, начиная с нуля, поэтому к перечисляемым переменным и константам могут быть применены операции отношения и стандартные функции

Pred, Succ, Ord.

Переменные и константы перечисляемого типа не могут быть элементами списка ввода или вывода.

Интервальный тип (тип-диапазон).

Отрезок любого порядкового типа может быть определен как интервальный или ограниченный тип.

Отрезок задается диапазоном от минимального до максимального значения констант, разделенных двумя точками.

В качестве констант могут быть использованы константы, принадлежащие к целому, символьному, логическому или перечисляемому типам.

Скалярный тип, на котором строится отрезок, называется базовым типом.

Минимальное и максимальное значения констант называются нижней и верхней границами отрезка, определяющего интервальный тип.

Нижняя граница должна быть меньше верхней. Над переменными, относящимися к интервальному типу, могут выполняться все операции и применяться все стандартные функции, которые допустимы для соответствующего базового типа.

При использовании в программах интервальных типов данных может осуществляться контроль за тем, чтобы значения переменных не выходили за границы, введенные для этих переменных в описании интервального типа.

type

Dig1 =’0’..’9’;

Dig2=48..57;

```
Type
Days=(mo,tu,we,th,fr,sa,su);
WeekEnd=sa..su;
Var
w:WeekEnd;
Begin
…..
W:=sa;
…
End;
```

Ord(w) – вернет значение 5,

Pred(w) – приведет к ошибке.

High(x)- возвращает максимальное значение типа-диапазона, к которому принадлежит x.

Low(x) - возвращает минимальное значение типа-диапазона, к которому принадлежит x.

```
Var
k:integer;
Begin
WriteLn(Low(k),'…',High(k));
End.
```

Результат –32768 …32767

Вещественные типы

В отличие от порядковых типов, значения которых всегда сопоставляются с рядом целых чисел и, следовательно, представляются в ПК абсолютно точно, значения вещественных типы определяют произвольное число лишь с некоторой конечной точностью, зависящей от внутреннего формата числа.

Длина	Назва-ние	Кол-во значащих цифр	Диапазон десятичного порядка.
4	Single	7… 8	
6	Real	11…12	-39… +38
8	Double	15…16	-324…+308
10	Extended	19…20	-4951…+4932
8	Comp	19…20	-2*10 в 63 степ.+1… +2*10 в 63 степ.-1

Вещественное число занимает от 4 до 10 смежных байт и имеет следующую структуру в памяти ПК.

S	**E**	**M**

S- знаковый разряд числа

E –экспоненциальная часть

M- мантисса

Мантисса m имеет длину

от 23 (для SINGLE) до 63 (для EXTENDED)

двоичных разрядов, что и обеспечивает точность

7.. .8 для SINGLE и 19.. .20 для EXTENDED десятичных цифр.

Десятичная точка (запятая) подразумевается перед левым (старшим) разрядом мантиссы, но при действиях с числом ее положение сдвигается влево или вправо в соответствии с двоичным порядком числа, хранящимся в экспоненциальной части, поэтому действия над вещественными числами называют арифметикой с плавающей точкой (запятой).

Доступ к типам SINGLE, DOUBLE и EXTENDED возможен только при особых режимах компиляции.

Эти типы рассчитаны на аппаратную поддержку арифметики с плавающей точкой и для их эффективного использования в состав ПК должен входить арифметический сопроцессор.

Компилятор Турбо Паскаля позволяет создавать программы, работающие на любых ПК (с сопроцессором или без него) и использующие любые вещественные типы. В процессе запуска Турбо Паскаль проверяет состав аппаратных средств и выявляет наличие или отсутствие сопроцессора.

В некоторых случаях бывает необходимо отключить автоконтроль. Для этого перед запуском Турбо Паскаля следует дать такую команду ДОС:

set 87=N

команда

set 87=Y

напротив, включает автоконтроль - эта команда активна по умолчанию.

Арифметический сопроцессор всегда обрабатывает числа в формате EXTENDED, а три других вещественных типа в этом случае получаются простым усечением результатов до нужных размеров и применяются в основном для экономии памяти.

Тип REAL оптимизирован для работы без сопроцессора. Если Ваш ПК оснащен сопроцессором, использование типа REAL приведет к дополнительным затратам времени на **преобразование REAL к EXTENDED.** Поэтому никогда не используйте REAL на ПК с сопроцессором, т.к. дополнительные затраты времени на преобразование типов могут свести на нет все преимущества сопроцессора. При разработке программ, критичных ко времени счета, следует заменять его типами SINGLE или DOUBLE: по сравнению с типом REAL скорость вычислений на машинах с сопроцессором в этом случае увеличивается в 2...3 раза. Если в ПК нет арифметического сопроцессора, скорость обработки данных всех вещественных типов приблизительно одинакова.

Особое положение в Турбо Паскале занимает тип **COMP**, который трактуется как вещественное число без экспоненциальной и дробной частей. **Фактически, COMP - это «большое» целое число со знаком**, сохраняющее 19...20 значащих десятичных цифр (во внутреннем представлении COMP занимает 8 смежных байт). В то же время в выражениях COMP полностью совместим с любыми другими вещественными типами: над ним определены все вещественные операции, он может использоваться как аргумент математических функций и т.д. Наиболее подходящей областью применения типа COMP являются бухгалтерские расчеты: денежные суммы выражаются в копейках или центах и действия над ними сводятся к операциям с достаточно длинными целыми числами.