*Условный оператор, оператор выбора. Процедуры ввода-вывода. Логические операции в Паскале, таблицы истинности.

Инструкции ввода и вывода
Ввод данных

Ввод данных — это передача исходных данных программы в оперативную память компьютера для обработки.

Инструкцию ввода с клавиатуры можно записать в одной из форм:

read(xl, x2, ..., xN);

readln;

readln(xl, x2,...,xN);

где **x1, x2, xN** — список ввода, содержащий имена переменных допустимых типов.

При этом первые две инструкции, выполненные последовательно, эквивалентны третьей.

- инструкция **readln** при вводе с клавиатуры предпочтительнее **read,** т. к. полностью освобождает *буфер клавиатуры*. Инструкция **read** оставляет в буфере клавиатуры код клавиши <Enter>, что может привести к неправильной работе следующей команды ввода;
- в одной инструкции **read** или **readln** можно записать несколько переменных, разделенных запятыми. При выполнении программы значения, вводимые с клавиатуры, можно разделять пробелом либо символом табуляции (клавиша <Tab>) или нажимать клавишу <Enter> после ввода каждого из значений;
- инструкция **readln;** (без переменных) обычно записывается в конце программы и служит для создания паузы, которая длится до нажатия пользователем клавиши <Enter>;
- тип данных, вводимых во время работы программы, должен соответствовать типу переменной, указанной в инструкции

*Вывод данных

Вывод данных — это передача данных после обработки из оперативной памяти на устройство вывода (экран, принтер, файл на диске).

Инструкция вывода на экран записывается в одной из следующих форм:

write(y1, y2,. . . ,yN) ;

writeln;

writeln(y1, y2,...,yN);

где **y1, y2,... ,yN** — *список вывода*.

Первые две инструкции, выполненные последовательно, эквивалентны третьей.

- инструкции **write** и **writeln** предназначены для вывода констант различных типов, значений переменных или выражений. Число параметров — произвольно;
- из констант наиболее часто выводятся строки текста (вспомним, что строковые константы заключаются в апострофы);
- если в инструкции вывода записано выражение, то сначала будет произведено его вычисление, а затем выполнен вывод полученного результата.
- процедура вывода **writeln** аналогична **write.** Отличие заключается в том, что после вывода курсор автоматически переходит в начало новой строки;
- инструкция **writeln** (без параметров) переводит курсор в начало следующей строки. Таким способом можно, например, отделять результаты работы программы друг от друга одной или несколькими пустыми строками.

Пример: программа спрашивает имя пользователя и выводит на экран сообщение, что данный пользователь учит Паскаль.

```
program p1;
var a:string;
begin
writeln(‘Введите свое имя');
readln(a);
writeln(a,' учит Паскаль');
readln;
end.
```

*Формат вывода

Вывод значения вещественного типа осуществляется в форме с использованием степени десяти. Если такой вид результатов не устраивает пользователей, то программисту следует позаботиться о приемлемом *формате вывода*.

В инструкциях вывода имеется возможность записи выражения, определяющего *ширину поля вывода* для каждой выводимой переменной или константы:

write (yl: w:d, y2: w: d, . . ., yN: w: d);

writeln (yl: w: d, y2: w: d, . . ., yN: w: d);

где **w** задает общую ширину поля вывода; **d** — место под дробную часть; **w** и **d** константы или выражения целого типа. Параметр **d** указывается только для выражений вещественного типа.

если **w**, заданное программистом, мало, то при выводе ширина поля будет *увеличена*. Часто задают **w** = 0 для автоматического подбора ширины поля вывода;

если **w**, наоборот, слишком велико, то выводимое значение прижимается к правому краю поля вывода, а слева выводятся пробелы;

если мало **d**, то производится *округление*.

Пример. Найти сумму, разность, произведение и частное двух чисел.

```
program p2;
var a,b,r,s,p:integer; c:real;
begin
writeln('Vvedite dva chisla');
readln(a,b);
r:=a-b;  s:=a+b;
p:=a*b;  c:=a/b;
writeln('a-b=',r);
writeln('a+b=',s);
writeln('a*b=',p);
writeln('a/b=',c:6:3);
readln;
end.
```

*Условный оператор IF

Оператор **if** можно записать двумя способами.

Вариант 1

if Условие **then**

{ Действия, которые выполняются, если Условие истинно}

else

{ Действия, которые выполняются, если Условие ложно };

Вариант 2:

if Условие **then**

{ Действия, которые выполняются, если Условие истинно };

Во втором случае говорят о *сокращенной форме* условного оператора (алгоритмическая конструкция *обход)*.

Описание выполнения оператора if

Выполнение условного оператора начинается с вычисления условия. *Если* оно *истинно*, то выполняется *оператор, стоящий после* служебного слова **then.** Если условие *ложно*, то выполняется *оператор, стоящий после* служебного слова **else,** а в сокращенной форме условного оператора — выполняется следующая по порядку за оператором условия инструкция.

Следует знать:

- при вложенности операторов каждое **else** соответствует тому **then,** которое непосредственно ему *предшествует;*
- конструкций со степенью вложенности более 2—3 необходимо *избегать* из-за сложности их анализа при отладке программы;
- если в сложном условном операторе проверяемые **Условие1, Условие2...** не влияют друг на друга, т. е. последовательность их вычисления безразлична, в тексте программы их рекомендуется располагать в определенном порядке. Условие, с наибольшей вероятностью принимающее значение **true,** должно стоять на первом месте, с меньшей вероятностью — на втором, и т. д. Это ускорит выполнение программы.

Пример: пользователь вводит возраст, определить совершеннолетний ли он.

```
program p3;
var a:integer;
begin
writeln('Vvedite vozrast');
readln(a);
if a>=18 then writeln('sovershennoletnij')
        else writeln('nesovershennoletnij');
readln;
end.
```

Пример: Пользователь вводит число, если оно положительное, то увеличить его в два раза.

```
program p4;
var a:integer;
begin
writeln('Vvedite chislo');
readln(a);
if a>0 then a:=a*2;
writeln('chislo=', a);
readln;
end.
```

*Составной оператор

Составной оператор представляет собой группу из произвольного числа операторов, отделенных друг от друга точками с запятой. Группа ограничена ***операторными скобками*** **begin** и **end**.

Часто в условных операторах используют составной оператор **begin end.** Если между **begin** и **end** находится *только одна инструкция*, слова **begin** и **end** лучше не писать.

Обратите внимание — при использовании в программе составного оператора обычно стараются слово **end** располагать строго под словом **begin,** чтобы легче было проверять соответствие операторных скобок.

Пример: Пользователь вводит число. Если оно положительное, то уменьшите его в два раза и выведите на экран, в противном случае выведите сообщение, является ли число отрицательным или нулем.

```
program p5;
var a:real;
begin
writeln('Vvedite chislo');
readln(a);
if a>0 then
            begin
            a:=a/2;
            writeln('chislo=', a:6:3);
            end
        else if a<0 then writeln('chislo<0')
                        else writeln('chislo=0');
readln;
end.
```

*Оператор выбора

```
case Выражение-селектор of
  СписокКонстант1: { Инструкции, которые выполняются
   если Выражение-селектор совпадает с одной из
   констант из СписокКонстант1 };
  СписокКонстант2: {Инструкции, которые выполняются
  если Выражение-селектор совпадает с одной из констант
  из СписокКонстант2 } ;
  …
  СписокКонстантN: {Инструкции, которые выполняются
  если Выражение-селектор совпадает с одной из констант
  из СписокКонстантN };
 else { Инструкции, которые выполняются, если
  Выражение-селектор не совпало ни с одной из констант }
 end;
```

Описание выполнения оператора case

Выполнение оператора **case** начинается с вычисления *выражения-селектора*. Инструкции, помещенные между **begin** и **end,** выполняются в том случае, если значение выражения, стоящего после слова **case,** *совпадает* с константой из соответствующего списка. Если это не так, то выполняются инструкции, следующие после **else,** расположенные между **begin** и **end.** Если **else** отсутствует, выполняется оператор программы, следующий за **case.**

Список констант выбора может состоять из произвольного количества значений или диапазонов, отделенных друг от друга запятыми. Границы диапазона записываются двумя константами через разграничитель "..". Тип констант должен совпадать с типом селектора.

Обратите внимание — в конце оператора **case** стоит ключевое слово **end,** для которого нет парного слова **begin.** Оператор **end** располагают строго под **case.**

Следует знать:

- инструкция **case** является обобщением оператора **if** и используется для выбора одного из *нескольких* направлений дальнейшего хода программы;
- выбор последовательности инструкций программы осуществляется во время ее выполнения в зависимости от текущего значения *выражения-селектора*.
- выражение-селектор должно иметь *порядковый тип* (чаще всего – **integer**, реже – **char, boolean** или один из пользовательских типов). Использование *вещественного и строкового* типа *недопустимо*.
- константы выбора внутри одного оператора выбора должны быть различны, в противном случае выполняется первая "подходящая" ветвь.
- точку с запятой можно не ставить после последнего

Пример: Пользователь вводит номер месяца, необходимо определить и вывести на экран пору года.

```
program p6;
var a:byte;
begin
writeln('Введите номер месяца');
readln(a);
case a of
  12,1,2:writeln(‘зима');
  3..5:writeln(‘весна');
  6..8:writeln(‘лето');
  9..11:writeln(‘осень');
  else writeln(‘нет такого месяца')
end;
readln;
end.
```

*Логические выражения

Основные операции отношения

Операция	Название	Выражение	Результат
=	Равно	A = B	true, если A равно B
< >	Не равно	A ≠ B	true, если A не равно B
>	Больше	A > B	true, если A больше B
<	Меньше	A < B	true, если A меньше B
>=	Больше или равно	A >= B	true, если A больше или равно B
<=	Меньше или равно	A <= B	true, если A меньше или равно B
In	Принадлежность	A in B	true, если A находится в списке B

Операции отношения выполняют сравнение двух операндов и определяют, истинно значение отношения или ложно.

Используя отношения, при помощи знаков *логических операций* получают более сложные *логические выражения*. Результат проверки любого условия, каким бы сложным оно ни было, всегда имеет логический тип – **boolean.**

*Таблицы истинности основных логических операций

Операция	Действие	Выражение	A	B	Результат
not	Логическое отрицание	not A	True		False
			False		True
and	Логическое И	A and B	True	True	True
			True	False	False
			False	True	False
			False	False	False
or	Логическое ИЛИ	A or B	True	True	True
			True	False	True
			False	True	True
			False	False	False
xor	Исключающее ИЛИ	A xor B	True	True	false
			True	False	true
			False	True	true
			False	False	false

Приоритет операций

Если в логическом *(булевском)* выражении не использованы круглые скобки, то операции выполняются в порядке убывания приоритетов. Операции одинакового приоритета выполняются слева направо. Для задания явного порядка используются круглые скобки.

Обратите внимание, в Pascal логические операции имеют более высокий приоритет, чем операции отношения. Поэтому каждое *простое условие в логическом выражении обязательно заключается в скобки*.

Например, для целых m, mmax, n, nmax выражение

m > mmax and n > nmax

вызовет сообщение об ошибке, т. к. сначала будет выполняться операция **mmax and n.** Определим приоритет, расставив скобки, и получим правильное выражение:

(m > mmax) and (n > nmax).

- в языке Pascal нельзя записать двустороннее неравенство вида **1<x<2.** Необходимо воспользоваться логическим выражением **(x>1) and (x<2);**
- нельзя также записать **x=y=z.** Необходимо: **(x=y) and(x=z);**
- для записи условия, заключающегося в том, что **x** не лежит в диапазоне от −2 до 2, можно использовать:
- **not((x>-2)and(x<2))** **ИЛИ** (x<=-2)**or(x>=2)**
- условие принадлежности точки кругу единичного радиуса с центром в начале координат: **sqr (x) +sqr (y) <1.**

Логические выражения могут использоваться в инструкциях присваивания и вывода. Но, конечно, основное назначение логических выражений — организация ветвлений, когда в зависимости от значения логического выражения выполняется та или иная группа операторов.

*Операции *div и mod*

Целочисленное деление **div** (от англ. *division* — деление) отличается от обычной операции деления тем, что возвращает целую часть частного, а дробная часть отбрасывается. И операнды и результат операции должны быть целочисленные.

Например **13 div 3 = 4**.

Взятие остатка от деления **mod** (от англ. *modulus* — мера) вычисляет остаток, полученный при выполнении целочисленного деления. И операнды и результат операции должны быть целочисленные.

Например 13 mod 4=1

*Домашнее задание

Составить опорный конспект лекции по теме «Условный оператор, оператор выбора. Логические операции в Паскале, таблицы истинности» на основе презентации.

Составить программы:

1. Пользователь вводит два катета прямоугольного треугольника. Найти его гипотенузу.
2. Пользователь вводит два числа, если большее кратно меньшему, то найдите их сумму, в противном случае найдите их частное.
3. Пользователь вводит возраст, определить категорию: до 3-х лет –младенец, от 3-х до 7-дошкольник, от 7 до 18-школьник, от 18-до 60-трудоспособный, после 60-пенсионер.