1. Основные операторы языка Паскаль.

1.1 Оператор ввода данных с клавиатуры

Для ввода информации с клавиатуры используется оператор (встроенная процедура) *read*. Формат:

*read (*список переменных*);*

readln(a,b,c);

Переменные перечисляются через запятую. После выполнения программа останавливается и ожидает ввода информации с клавиатуры. Ввод осуществляется набором чисел, разделенных пробелом, и заканчивается нажатием клавиши ввода. Необходимо отметить, что вводимое значение должно соответствовать типам переменных и смыслу программы.

var b:real; a:char; … read(a,b);.

Если у нас есть несколько команд *read*, разделенных другими командами, то во время выполнения первого оператора *read* можно ввести все переменные. Для остальных команд программа не останавливается.

В отличие от оператора *read* оператор *readln* после ввода всех указанных в операторе данных осуществляет переход к следующей строке.

1.2 Оператор вывода данных на экран

Оператор вывода *write* служит для вывода информации на экран дисплея или принтера. Вид оператора:

write (список выражений);.

Выражения в списке разделяются запятыми. В результате выполнения оператора *write* могут вычисляться значения выражений.

*write ('y= ', 2*y);.*

Строки выводятся без всяких изменений, ограничивающие их апострофы не выводятся.

Наряду с оператором *write* используется оператор *writeln*. Отличие состоит в следующем: при выполнении оператора *write* информация выводится на экран и курсор остается в той же строке. Переход на следующую строку выполняется только тогда, когда выводимая строка полностью заполнена. В результате выполнения *writeln* после вывода информации курсор переводится на следующую строку. Результатом выполнения

write (1); writeln; write(-2,3);

будет

1

-2 3

Если не принять специальных мер, то значения вещественных переменных и выражений выводятся в форме с плавающей запятой. Для вывода вещественных чисел в форме с фиксированной точкой указывается формат после числа:

write(число:w:d);.

Здесь: w – общая ширина поля числа после вывода; d – количество цифр после запятой (последние цифры округляются).

writeln(- 73.1:6:1);.

Результат: (на экране клеток нет) будем считать, что одна клетка таблицы – одно знакоместо на экране

	-	7	3	.	1	

Если число литер в представлении выводимого значения оказывается меньше, чем w, то оно слева дополняется пробелами. Если w опускается, то назначается стандартная длина плоя, зависящая от конкретной ЭВМ.

Примеры операторов вывода:

1. Организация вывода значений переменных a, b, c **целого** типа – *var a,b,c:integer;*

read(a,b,c);

writeln(a,b,c);.

Ввод:
12 34 98
Вывод:
123498

writeln(a:4,b:5); write(c);.
Ввод:
12 -34 6
Вывод:
 12 -34
6

writeln('a=',a:4,' b=',b:5);write(c);.
Ввод:
555 -899 -8
Вывод:
a= 555 b= -899
-8

Если количество указанных позиций недостаточно, то происходит автоматическое увеличение поля до необходимых размеров.

2. Организация вывода ***вещественных*** чисел – *var a,b,c:real;.*
read(a,b,c);
writeln(a,b,c); (длина поля не указана).
Ввод:
222.677777 888.4444444 -6.7
Вывод:
2.2267777700E+02 8.884444440E+0.2-6.7000000000E+00
writeln(a:6:3,b:7:2,c:4:1);.
Ввод:
222.677777 888.4444444 -6.7
Вывод:
222.678 888.44-6.7

Если в операторе вывода указано общее число позиций w и не указано количество позиций после запятой, то числа выводятся в экспоненциальной форме с шириной поля w.

writeln(a:6,b:7,c:4);.
Ввод:
222.677777 888.4444444 -6.7
Вывод:
2.2E+02 8.9E+02-6.7e+00

3. Организация вывода значений **символьного** типа – *var a,b,c:char;.*
read(a,b,c);
writeln(a:6,b:8,c:4);.
Ввод (без пробелов, т. к. пробел – это символ):
pro
Вывод:
p r o

writeln(a,b,c);.
Ввод:
pro
Вывод:
pro

4. При выводе значений **логического** (булевского) типа на печать выводится *true* или *false*.

program ww;
var a,b,c:integer;

```
BEGIN
  readln(a,b,c);
  writeln(a<b:7,b>c:4);
END.
```

Ввод:

1 2 3

TRUEFALSE

1.3 Оператор присваивания.

Оператор присваивания имеет вид:

<имя переменной>**:=**выражение;

При вычислении значения выражения, стоящего в первой части оператора присваивания, учитывается тип данных и операции, которые над ним выполняются. Если в выражение входят данные целого типа, над которыми выполняются операции +,-,*, *div*, *mod*, то результат будет целочисленным. Если хотя бы один элемент данных относится к вещественному типу или встречается операция «деление», то результат будет вещественного типа. При этом будет сообщение об ошибке, указывающее на несоответствие типа.

1.4 Составной оператор и пустой оператор

Составной оператор - это последовательность произвольных операторов программы, заключенная в операторные скобки - зарезервированные слова

begin . . . end;

Составные операторы - важный инструмент Паскаля, дающий возможность писать программы по современной технологии структурного программирования (без операторов перехода GOTO).

Фактически, весь раздел операторов, обрамленный словами begin . . . end, представляет собой один составной оператор. Поскольку зарезервированное слово end является закрывающей операторной скобкой, оно одновременно указывает и конец предыдущего оператора, поэтому ставить перед ним символ «;» необязательно, и далее во всех примерах мы не будем этого делать. Наличие точки с запятой перед end в предыдущих примерах означало, что между последним оператором и операторной скобкой end располагается пустой оператор. Пустой оператор не содержит никаких действий, просто в программу добавляется лишняя точка с запятой. В основном пустой оператор используется для передачи управления в конец составного оператора.

Пример 1. Рассмотрим программу для вычисления значений выражений: $b = a^3$; $c = \sin(a) + \sqrt{|d|}$.

```
program    umn;
uses crt;
var a,b,c,d: real;
BEGIN
 clrscr;
  writeln('введите  a d');
  readln(a,d);
  b:=a*a*a;   c:=sin(a)+sqrt(abs(d));
  writeln('Ответ b=  ', b:6:2, '     c= ', c:8:1);
 readln;
END.
```