PASCAL

- ## Basic Syntax:

1. Variables

```
var
A_Variable, B_Variable ... : Variable_Type;
```

2. Functions

```
Function Func_Name(params...) : Return_Value;
Procedure Proc_Name(params...);
```

3. Statements

```
readln (a, b, c);
s := (a + b + c)/2.0;
area := sqrt(s * (s - a)*(s-b)*(s-c));
writeln(area);
```

4. Comments

```
(* This is a multi-line comments
   and it will span multiple lines. *)

{ This is a single line comment in pascal }
```

- ## Data Types:

Type	Minimum	Maximum	Format
Integer	-2147483648	2147483647	signed 32-bit
Cardinal	0	4294967295	unsigned 32-bit
Shortint	-128	127	signed 8-bit
Smallint	-32768	32767	signed 16-bit
Longint	-2147483648	2147483647	signed 32-bit
Int64	-2^63	2^63 - 1	signed 64-bit
Byte	0	255	unsigned 8-bit
Word	0	65535	unsigned 16-bit
Longword	0	4294967295	unsigned 32-bit

Examples:

```
days, age = integer;
yes, true = boolean;
name, city = string;
fees, expenses = real;
PIE  =  3.141592;
```

- ## LOOPS:

1. WHILE-DO LOOP

   Syntax:

```
        while (condition) do S;
```

Example:

```
while number>0 do
begin
  sum := sum + number;
  number := number - 2;
end;
```

2. FOR-DO LOOP
   Syntax:

```
for < variable-name > := < initial_value > to [down to] <
final_value > do S;
```

   Example:

```
for i:= 1 to 10 do writeln(i);
```

3. REPEAT-UNTIL
   Syntax:

```
  repeat
    S1;
    S2;
    ...
    ...
    Sn;
until condition;
```

   Example:

```
repeat
  sum := sum + number;
  number := number - 2;
until number = 0;
```

4. NESTED LOOP
   Syntax:

```
for variable1:=initial_value1 to [downto]
final_value1 do
begin
```

```
      for variable2:=initial_value2 to [downto]
     final_value2 do   begin

     statement(s);

   end;

end;
```

Example:

```
program nestedPrime;
var
   i, j:integer;

begin
   for i := 2 to 50 do

   begin
      for j := 2 to i do
         if (i mod j)=0  then
            break; {* if factor found, not prime *}

      if(j = i) then
         writeln(i , ' is prime' );
   end;
end.
```

- **FUNCTIONS:**

```
function name(argument(s): type1; argument(s): type2;
...): function_type;
local declarations;

begin
  ...
  < statements >
  ...
  name:= expression;
end;
```

- **DATA STRUCTURES:**

1. Arrays:

```
type
  vector = array [ 1..25] of real;
var
  velocity: vector;
```

2. Pointers:

```
type
  Rptr = ^real;
  Cptr = ^char;
  Bptr = ^ Boolean;
  Aptr = ^array[1..5] of real;
  date-ptr = ^ date;
    Date = record
      Day: 1..31;
      Month: 1..12;
      Year: 1900..3000;
    End;
```

```
var
   a, b : Rptr;
   d: date-ptr;
```