Pascal (programming language)

ver.0.1 (Nov. 2025), License: Public domain
konstantin.pankov@fsight.ru

Keywords

Turbo Pascal:	nil	**+Free Pascal:**
absolute	not	dispose
and	object	exit
array	of	false
asm	operator	new
begin	or	true
case	packed	
const	procedure	**+Object Pascal:**
constructor	program	as
destructor	record	class
div	reintroduce	dispinterface
do	repeat	except
downto	self	exports
else	set	finalization
end	shl	finally
file	shr	initialization
for	string	inline
function	then	is
goto	to	library
if	type	on
implementation	unit	out
in	until	packed
inherited	uses	property
inline	var	raise
interface	while	resourcestring
label	with	threadvar
mod	xor	try

Internal types

Integers

ShortInt	8-bit	-128..127
SmallInt	16-bit	-32768..32767
Integer	16/32-bit	Smallint or Longint
Longint	32-bit	-2147483648 .. 2147483647
Int64	64-bit	
Byte	8-bit	
Word	16-bit	
LongWord	32-bit	DWord, Cardinal
QWord	64-bit	

Floating-points

Real	32/64-bit	
Double	64-bit	

Booleans

Boolean	8-bit	True, False

Characters

Char	8-bit	AnsiChar
WideChar	16-bit	
String		msg: string[16];

Custom types

Enumerations: (name1, name2, ..., nameN)

```
type
    EnumType = (zero, one, twenty := 20, twentyone);
```

Subranges: start..end

```
type
    digit10 = 0..9;
```

Arrays: array[start..end] of type

```
type
    real100 = array[0..100] of real;
var
    a, b: real100;
    c: array[0..4] of integer;
```

Records:

```
RecordTypeName = record
    name1: type1;
    ...
    nameN: typeN;
end;
```

```
type
    Coord = record
        x: integer;
        y: integer;
        n: real;
    end;
var
    FirstCoord: Coord;
begin
    FirstCoord.x := 1;
    FirstCoord.y := 2;
    FirstCoord.n := 42
end.
```

Pointers

New creates a new instance
Dispose releases the instance

```
var
    r: real;
    p: ^real;
begin
    r := 3.14;
    p := @r;
    p^ := 1.618
end.
```

```
var
    p: ^integer;
begin
    new(p);
    ...
    dispose(p);
end.
```

Literals

'c' for characters, 'xxx' for strings

```
const
    chr = 'c'; // character literal
    Msg = 'string literal';
begin
    write('LF is #10 and'#13#10'CR is #13';)
end.
```

Variables

```
var identifier_list: type;
```

```
var
    n: integer;
    m: integer;
begin
    write('n: ');
    readln(n);
    write('m: ');
    readln(m);
    writeln(n, ' * ', m, ' = ', n * m)
end.
```

Constants

```
const constant_name[: type] = value;
```

```
const
    Count: integer = 10;
var
    i: integer;
begin
    for i := 0 to Count - 1 do
        writeln(i)
end.
```

Pascal (programming language)

ver.0.1 (Nov. 2025), License: Public domain
konstantin.pankov@fsight.ru

Comments

```
(* This is an old style comment *)
{ This is a Turbo Pascal comment }
// This is a Delphi single line comment
```

```
program Hello; // single line comment
begin
    {   multiline
        comment
        block   }
    writeln('Hello, pascal!')
end.
```

Operators

Assignment

```
a := b;
```

Relational

```
 =     equal
 <>    not equal
 <     less than
 >     greater than
 <=    less than or equal
 >=    greater than or equal
```

Binary arithmetic

```
 +     addition
 -     substraction
 *     multiplication
 /     Division
 div   integer division
 mod   Modulo division
```

Unary arithmetic

```
 +     positive value
 -     negative value
```

Logical

```
 not   bitwise unary negation
 and   bitwise and
 or    bitwise or
 xor   bitwise exclusive or
 shl   bitwise shift to the left
 shr   bitwise shift to the right
```

Truth

```
 not   logical negation
 and   logical and
 or    logical or
 xor   logical xor
```

String

```
 +     string concatenation
```

The priority of the operators

```
 not @ + -                          first (highest)
 * / div mod and shl shr            second
 + - or xor                         third
 = <> < > <= >=                     Fourth (lowest)
```

Conditional statements

```
if expression then statement_1 [else statement_2]
```

```
var
    x : integer;
begin
    read(x);
    if x > 10 then
        x := x mod 10;
    writeln(x)
end.
```

```
case selector of
  value_1: statement_2;
  ...
  value_n: statement_n;
  [else default_statement]
end;
```

```
procedure PrintErrorCode(i: integer);
begin
    case i of
        1: writeln('Access denied');
        2: writeln('Out of range');
        else writeln('No error')
    end
end;
```

Iteration statements (Loops)

```
for counter := initial_value to|downto final_value
do statement
```

```
var
    i: integer;
begin
    for i := 1 to 10 do
    begin
        write('*')
    end
end.
```

```
while expression do statement
```

```
var
    i : integer := 10;
begin
    while i <> 0 do
    begin
        writeln('while test');
        i := i - 1
    end
end.
```

```
repeat statement do expression
```

```
var
    x : integer;
    i : integer;
begin
    i := 0;
    readln(x);
    repeat
    { begin is not required }
        x := x div 2;
        i := i + 1
    { end is not required }
    until x <= 1;
    writeln(i)
end.
```

Pascal (programming language)

ver.0.1 (Nov. 2025), License: Public domain
konstantin.pankov@fsight.ru

Procedures

```
procedure proc_name(parameter_list);
    { local declarations }
begin
    { procedure body }
end;
```

```
program HelloProc;

procedure SayHello;
begin
    writeln('Hello, world!')
end;

begin
    SayHello
end.
```

Functions

```
function func_name(parameter_list) : return_type;
    { local declarations }
begin
    { procedure body }
end;
```

```
function Pow(Base: integer; Exp: integer): integer;
var
    Res: integer;
begin
    Res := 1;
    while Exp > 0 do
    begin
        if Odd(Exp) then
            Res := Res * Base;
        Base := Base * Base;
        Exp := Exp Div 2
    end;
    Pow := Res
end;
```

Jump statements

Break – Exit current loop construct

```
for i := 1 to 10 do
begin
    if i = 5 then
        Break;
    writeln('i = ', i);
end;
```

Continue - Continue with next loop cycle

```
for i := 1 to 10 do
begin
    if i = 5 then
        Continue; // skip 7
    writeln('i = ', i);
end;
```

exit - Break out of a routine

```
procedure MyProc;
begin
    { ... }
    if Error = True then exit;
    { ... }
end;
```

halt – Stop program execution

```
    halt(1); { Stop with exit code 1 }
```

goto – Stop program execution

```
label c1;
var a: array [1..10,1..10] of integer;
...
    var found := False;
    for var i := 1 to 10 do
        for var j := 1 to 10 do
            if a[i,j]=k then
            begin
                found := True;
                goto c1;
            end;
c1: writeln(found);
```

Files

Types:

```
t: text; // text file
tf: file of integer; // typed binary file
f: file; // untyped binary
```

```
program HelloFile;
{$I-} // I/O checking off
var f: text; // Text file
begin
    assign(f, 'hello.txt'); // Assign filename
    rewrite(f); // Create file for writing
    // append(f) opens existing file for append
    write(f, 'Hello, pascal!');
    close(f) // Close file
end.
{ --- Open existing file for reading --- }
program ReadFile;
{$I-}
var
    line: string;
    f: text;
begin
    assign(f, 'readfile.pas');
    reset(f); // Open existing file
    while not eof(f) do
    begin
        readln(f, line);
        writeln(line)
    end;
    close(f)
end.
{ --- Create typed file for writing --- }
program TypedFile;
type
  Person = record
    ...
  end;
var
  people: array[1..3] of Person;
  i: integer;
  f: file of Person;
begin
...
  assign(f, 'people.dat');
  rewrite(f);
  for i := 1 to 3 do
  begin
    write(f, people[i]);
  end;
  close(f);
end.
```