Manual de Pascal

Este manual está diseñado para introducirlo a los conceptos fundamentales de Pascal, un lenguaje que equilibra simplicidad y estructura. A lo largo del contenido, exploraremos desde lo más básico, como el control de flujo y la asignación de variables, hasta aspectos más complejos como el manejo de punteros, memoria dinámica y registros definidos por el usuario. Usted se familiarizará con la sintaxis clara y estricta de Pascal, la declaración de variables y procedimientos, y el uso eficiente de tipos de datos y operadores. Con esta guía, podrá adquirir las herramientas necesarias para escribir programas claros y eficientes en Pascal.

Conceptos Fundamentales

1. Uso de comandos para controlar el flujo del programa.

Pascal utiliza estructuras de control que permiten alterar el flujo secuencial del programa. Estas incluyen condicionales, bucles y sentencias de salto.

Si se quieren ejecutar varias instrucciones, se deben agrupar con `begin ... end`:

```
begin
  writeln('Mayor de edad');
  writeln('Puede votar');
end
```

Para más información mirar la seccíon de Estructuras de Flujo.

2. Empleo de asignaciones a variables para manipular el estado del sistema.

En Pascal, las variables se utilizan para guardar y modificar valores durante la ejecución del programa. Esto permite llevar control sobre el estado del sistema, como por ejemplo usando contadores, banderas (Boolean), acumuladores, entre otros.

Las asignaciones se hacen con el operador `:=`, y las variables deben ser declaradas antes de usarse, indicando su tipo.

```
var
  contador: Integer;
begin
  contador := 0;
  contador := contador + 1;
  writeln('Contador: ', contador);
end.
```

3. Forma de definir una ejecución secuencial de instrucciones

En Pascal, las instrucciones se ejecutan en orden secuencial, es decir, una tras otra tal como están escritas en el programa. Este flujo solo cambia si se utilizan estructuras como condicionales o bucles.

```
begin
  writeln('Iniciando');
  writeln('Hola Mundo');
  writeln('Finalizando');
end.
```

Este código ejecuta tres instrucciones en el orden en que aparecen.

Aspectos de Léxico y Sintaxis

1. Sintaxis de forma libre

Tokens

En Pascal existen los tokens, que hacen referencia a palabras que tienen un significado para el compilador. Pueden ser ya sea palabras reservadas, identificadores, una constante, una literal o un simbolo.

```
writeln('Hola mundo!');
```

En ese ejemplo, los tokens son: writeln, los paréntesis, el string 'Hola mundo!' y el punto y coma ;

Comentarios

Pascal permite tres estilos de comentarios:

```
{ Comentario con llaves }

(* Comentario con paréntesis y asterisco *)

// Comentario de una sola línea (solo en Free Pascal y Delphi)
```

Espacios en Blanco y saltos de línea

Pascal ignora los espacios en blanco y los saltos de línea entre instrucciones. Sin embargo, se debe usar ; para indicar el final de una instrucción.

```
a := 5; b := 10; writeln(a + b);
```

Es equivalente a

```
a := 5;
b := 10;
writeln(a + b);
```

2. Definiciones globales, funciones, variables y tipos de datos.

Funciones y procedimientos

En Pascal se pueden crear funciones (que devuelven un valor) y procedimientos (que no devuelven nada).

```
{Función}
function sumar(a, b: Integer): Integer;
begin
  sumar := a + b;
end;
```

```
{Procedimiento}
procedure saludar;
begin
  writeln('Hola mundo');
end;
```

Variables

En Pascal, las variables deben declararse antes de usarse. Para eso se usa la palabra clave var, seguida del nombre de la variable y su tipo.

```
var
  nombre: String;
  edad: Integer;
```

Definiciones globales

En Pascal, las definiciones globales se colocan antes del bloque principal `begin ... end.`. Estas pueden incluir constantes, funciones, procedimientos y variables que estarán disponibles para todo el programa.

```
const {constantes globales}
  PI = 3.14;

var {variables globales}
  nombre: String;
  edad: Integer;
```

Tipos de Datos

En Pascal existen los siguientes tipos de datos básicos:

- **Boolean:** Puede tomar los valores binarios True o False.
- **Integer:** Números enteros, positivos o negativos. Ejemplo: -5, 0, 42.
- **Real:** Números con decimales. Ejemplo: 3.14, -0.5.
- **Char:** Representa un solo carácter. Ejemplo: 'A' o '9'.
- **String:** Representa una secuencia de caracteres. Ejemplo: 'Hola mundo'.

Tipos de Datos Básicos

1. Enteros

Tipo	Descripción	Tamaño (bytes)
`Integer`	Entero común, con signo	2 o 4
`Byte`	Entero sin signo (0 a 255)	1
`ShortInt`	Entero pequeño con signo (-128 a 127)	1
`LongInt`	Entero largo con signo	4
`Word`	Entero sin signo (0 a 65535)	2

2. Caracteres

Tipo	Descripción	Tamaño (bytes)
`Char`	Un solo carácter ASCII	1
`WideChar`	Carácter Unicode (UTF-16)	2

WideChar es soportado en Free Pascal y Delphi para trabajar con Unicode.

3. Flotantes

Tipo	Descripción	Tamaño (bytes)
`Real`	Flotante básico	6 (obsoleto)
`Single`	Precisión simple (7 dígitos)	4
`Double`	Precisión doble (15 dígitos)	8
`Extended`	Precisión extendida (19-20 dígitos)	10

4. Especiales

Pascal no tiene un tipo void, pero puede simularse con procedimientos que no devuelven valor.

Declaraciones e identificadores

1. Uso de palabras claves

Pascal no utiliza palabras clave como static, extern, register o typedef, propias de lenguajes como C o C++. En su lugar, controla el comportamiento del programa de forma estructurada y explícita, sin necesitar esos modificadores.

- No existe static, pero se puede simular comportamiento persistente usando variables globales.
- No hay extern, ya que Pascal no requiere declarar funciones externas del mismo modo que C.
- Los tipos personalizados se crean con la palabra clave type.

```
type
  Edad = Integer;
```

2. Restricciones en la formación de nombres de variables.

Pascal tiene reglas estrictas para la formación de nombres (identificadores):

- Deben comenzar con una letra (A–Z, a–z).
- Pueden incluir letras y dígitos (0–9), pero no caracteres especiales como @, #, $, etc.
- No pueden contener espacios ni guiones. El uso del guion bajo _ no está permitido en Pascal estándar, aunque sí está soportado por algunos compiladores modernos, como `Free Pascal y Delphi`.
- No pueden usar palabras reservadas del lenguaje (begin, if, then, etc.).

Ejemplos válidos:

```
total
contador1
nombre_usuario
```

Ejemplos inválidos:

```
1numero     // comienza con un número
total$      // contiene carácter no válido
if          // palabra reservada
```

3. Diferenciación entre variables globales y locales.

Globales: se declaran al inicio del programa, fuera de cualquier función o procedimiento. Están disponibles en todo el programa.

Locales: se declaran dentro de una función o procedimiento, y solo existen dentro de ese bloque.

```
var
  global: Integer; {variable global}

procedure ejemplo;
var
  local: Integer;  {variable local}
begin
  local := 10;
  global := 5;
end;
```

Estructuras de Control de Flujo

Condicionales

- `if ... then ... else`

```
if edad >= 18 then
    writeln('Mayor de edad') { print }
else
begin
    writeln('Menor de edad');
    writeln('No puede votar');
end;
```

- Selección múltiple `case ... of`

```
case opcion of
    1: writeln('Inicio');
    2: writeln('Opción dos');
else
    writeln('Otra opción');
end;
```

Ciclos:

- for ascendente `for ... to`

```
for i := 1 to 5 do
    writeln('Valor de i: ', i);
```

- for descendente `for ... downto`

```
for i := 5 downto 1 do
    writeln('Contando hacia atrás: ', i);
```

- `while ... do`

```
while x < 3 do
begin
    writeln(x);
    x := x + 1;
end;
```

Sentencias de salto:

- Instrucción `exit`: Salir de una función

```
begin
    writeln('Inicio del programa');
    exit;
    writeln('Esto no se ejecuta');
end.
```

- Instrucción `goto`: Saltar a una etiqueta como un jump en ensamblador.

```
    if x = 1 then
        goto salir;
    writeln('Esto no se imprime');
salir:
    writeln('Saltamos directamente aquí');
```

Operadores

1. Aritméticos

Pascal permite el uso de los operadores aritméticos más comunes:

Operador	Descripción
`+`	Suma dos operandos
`-`	Resta entre dos operandos
`*`	Multiplica ambos operandos

Operador	Descripción
`/`	Divide y devuelve un resultado de tipo Real
`div`	División entera: devuelve solo el cociente
`mod`	Módulo: devuelve el residuo de la división entera

2. Relacionales

Se utilizan para comparar valores, y se usan en expresiones condicionales:

Operador	Descripción
`=`	Igualdad
`<>`	Diferente
`<`	Menor que
`>`	Mayor que
`<=`	Menor o igual que
`>=`	Mayor o igual que

3. Lógicos

Se utilizan para trabajar con valores booleanos:

Operador	Descripción
`and`	Verdadero si ambos operandos son verdaderos
`or`	Verdadero si al menos uno es verdadero
`not`	Niega el valor lógico
`xor`	Verdadero si exactamente uno es verdadero

4. Manejo de bits

Pascal incluye operadores lógicos que también pueden actuar a nivel de bits (especialmente en Free Pascal):

Operador	Descripción
`and`	AND bit a bit
`or`	OR bit a bit
`xor`	XOR bit a bit
`shl`	Shift Left: desplaza bits a la izquierda
`shr`	Shift Right: desplaza bits a la derecha

5. Asignaciones

Pascal utiliza := como operador de asignación. A diferencia de Go o C, no tiene operadores compuestos como +=, *=.

Operador	Descripción
`:=`	Asigna el valor de la derecha a la variable de la izquierda

6. Manejo de punteros

Pascal permite trabajar con punteros usando los siguientes operadores:

Operador	Descripción
`^`	De referencia: accede al valor al que apunta el puntero
`@`	Devuelve la dirección de una variable
`=`	Compara si dos punteros apuntan a la **misma dirección**
`<>`	Compara si dos punteros apuntan a **diferentes direcciones**

Funciones

1. Declaraciones y definiciones con tipos de retorno

En Pascal, las funciones se declaran utilizando la palabra clave function, seguida del nombre, los parámetros y el tipo de dato que devolverá. Para funciones que no devuelven valor (similar a void en otros lenguajes), se usa procedure.

```
// function NombreFuncion(parámetros): TipoDeRetorno;

{Retorna un Integer}
function cuadrado(x: Integer): Integer;
begin
  cuadrado := x * x;
end;

{Sin valor de retorno}
procedure mostrarMensaje;
begin
  writeln('Hola mundo');
end;
```

2. Parámetros por valor y referencia

Por defecto, los parámetros en Pascal se pasan por valor, lo que significa que se hace una copia del argumento.

Si se quiere pasar un parámetro por referencia (para modificar su valor original), se debe usar la palabra clave var.

```
procedure porValor(x: Integer);
begin
  x := x + 1;  // solo cambia la copia
end;

procedure porReferencia(var x: Integer);
begin
  x := x + 1;  // modifica el valor original
end;
```

3. Uso de return para devolver valores

Pascal no utiliza la palabra return como en otros lenguajes. En su lugar, el valor que se desea devolver se asigna al nombre de la función.

```
function sumar(a, b: Integer): Integer;
begin
  sumar := a + b;  // valor de retorno
end;
```

Punteros y Arreglos

1. Declaración y manipulación de punteros

En Pascal, un puntero es una variable que almacena la dirección de memoria de otra variable. Se declara con el símbolo ^ y se asigna con la función @.

```
type
  PEntero = ^Integer;

var
  puntero: PEntero;
  numero: Integer;
begin
  numero := 42;
  puntero := @numero;  // asignar dirección de memoria
end.
```

Desreferenciación

Para acceder o modificar el valor apuntado, se utiliza el operador ^:

```
puntero^ := 100;   // modifica el valor original
writeln(puntero^); // imprime 100
```

Puntero nulo

En Pascal, el equivalente a un puntero nulo es nil. Se usa para indicar que el puntero no apunta a nada:

```
puntero := nil;
```

2. Declaración y manipulación de arreglos

En Pascal, un arreglo es una colección de elementos del mismo tipo. Se declara con un rango y el tipo de dato:

Declaración:

```
var
  numeros: array[1..5] of Integer;
```

Asignar valores:

```
numeros[1] := 10;
numeros[2] := 20;
```

Acceso por indices

```
writeln(numeros[1]);  // imprime 10
```

3. Aritmética de punteros

En Pascal no se puede usar p + 1 como en C. Para mover un puntero al siguiente o anterior valor, se usan las funciones Inc y Dec.

Función	Descripción
`Inc(p)`	Avanza el puntero al siguiente dato
`Dec(p)`	Retrocede el puntero al dato anterior

Inc y Dec solo funcionan con punteros en compiladores como Free Pascal. En Pascal estándar no están permitidos con punteros.

4. Punteros a funciones

En Pascal se pueden usar punteros a funciones, pero esta característica solo está disponible en compiladores como Free Pascal o Delphi. Sirve para guardar una función dentro de una variable y luego llamarla.

```
type
  TOperacion = function(a, b: Integer): Integer;

function sumar(a, b: Integer): Integer;
begin
  sumar := a + b;
end;

var
  operacion: TOperacion;

begin
  operacion := @sumar;
  writeln(operacion(2, 3));  // imprime 5
end.
```

Memoria dinámica

Función	Qué hace	Equivalente en C	¿Pascal estándar?
`New(ptr)`	Reserva memoria para una variable apuntada	`malloc()`	☑ Sí
`Dispose(ptr)`	Libera memoria reservada con `New`	`free()`	☑ Sí
`GetMem(ptr, size)`	Reserva memoria de tamaño específico	`malloc()`	❌ No
`FreeMem(ptr)`	Libera memoria reservada con `GetMem`	`free()`	❌ No
`AllocMem(size)`	Reserva e inicializa a cero	`calloc()`	❌ No
`ReallocMem(ptr, size)`	Redimensiona la memoria y preserva el contenido	`realloc()`	❌ No

Solo New y Dispose son parte del Pascal estándar. Las demás están disponibles en compiladores como Free Pascal y Delphi.

```
procedure New(
  var P: Pointer
);

procedure New(
  var P: Pointer;
```

```
  Cons: TProcedure
);
```

```
procedure Dispose(
  P: Pointer
);

procedure Dispose(
  P: TypedPointer;
  Des: TProcedure
);
```

```
begin
  New (P);
  P^:='Hello, World !';
  Dispose (P);
  { P is undefined from here on !}
  New(T,Init);
  T^.i:=0;
  Dispose (T,Done);
end.
```

Estructuras Definidas por el usuario

1. Estructuras de datos para agrupar diferentes tipos de datos

En Pascal, se pueden agrupar varios valores de distintos tipos usando registros (record). Son similares a las estructuras (struct) en C.

```
type
  Persona = record
    nombre: String;
    edad: Integer;
    activo: Boolean;
  end;

var
  p: Persona;
begin
  p.nombre := 'Ana';
  p.edad := 25;
  p.activo := True;
end.
```

2. Uniones (union) para compartir memoria entre distintos tipos

En Pascal no existe el tipo union como en C, pero su comportamiento puede simularse utilizando registros variantes (variant records).

```
type
  TUnion = record {variant record}
    caso: Integer;
    case Integer of
      0: (i: Integer);
      1: (r: Real);
      2: (c: Char);
  end;
```

Esta característica puede variar según el compilador, por lo que se recomienda el uso de Free Pascal. Para más información, se puede consultar la documentación oficial de Free Pascal [3].

3. Enumeraciones (enum) para definir conjuntos de valores simbólicos

Pascal permite definir enumeraciones usando la palabra clave `type`.

```
type
  Dia = (Lunes, Martes, Miercoles, Jueves, Viernes);

var
  hoy: Dia;
begin
  hoy := Martes;
end.
```

Ambiente de ejecución

En Pascal, no existe una función llamada main como en Go o C. En su lugar, la ejecución comienza en el bloque principal del programa, justo después de la palabra clave `begin`, y termina en el `end.`

Todo lo que esté dentro de ese bloque se ejecuta en orden, a menos que el flujo sea alterado por estructuras como condicionales o bucles.

```
program MiPrograma;

begin
  writeln('Inicio del programa');
  // Aquí continúa la ejecución del código
end.
```

Referencias

[1] Free Pascal Team, Free Pascal Language Reference Guide. [Online]. Available: https://www.freepascal.org/docs-html/ref/ref.html

[2] Free Pascal Team, System Unit Reference. [Online]. Available: https://www.freepascal.org/docs-html/rtl/system/index.html

[3] Free Pascal Team, Variant Records. [Online]. Available: https://www.freepascal.org/docs-html/ref/refsu15.html

[4] TutorialsPoint, Pascal Programming Language. [Online]. Available: https://www.tutorialspoint.com/pascal/index.htm

[5] Wikibooks contributors, Pascal Programming. [Online]. Available: https://en.wikibooks.org/wiki/Pascal_Programming