Convenciones de Programación en Pascal

1. Introducción

La calidad de los programas está determinada por aspectos de fondo (corrección y eficiencia) y de forma (diseño, estilo y documentación).
La corrección es el objetivo principal del programa: debe verificar su especificación. El diseño atiende aspectos semánticos y metodológicos, como son la organización del programa, el desarrollo de las estructuras que lo forman, los subprogramas, tipos de datos, etcétera. Un buen diseño permite mantener fácilmente a un programa y reutilizarlo. El estilo y la documentación atienden aspectos formales como el uso de identificadores adecuados, de comentarios. Su objeto es hacer más legibles los programas.
En general recomendamos que los programas sean auto-explicativos, la idea es que mantener dos textos distintos para un mismo programa (código y la documentación) propicia los errores, debido a la dificultad de mantenerlos consistentes. Por lo tanto conviene integrar el programa y la explicación del mismo en un solo documento.

2. Estructura de un Programa en Pascal

El lenguaje Pascal fue desarrollado por Niklaus Wirth, con el propósito de ayudar a los estudiantes en el manejo de las técnicas de la programación estructurada, pero en la actualidad su aplicación es de propósitos generales.
La construcción de programas en Pascal se basa en módulos que guardan las siguientes reglas de construcción. Las **palabras reservadas** las podemos escribir tanto en **mayúscula** como en **minúscula**. Lo importante es adoptar una forma de escribirlas y hacerlo **siempre de la misma manera**.

Todas las palabras reservadas en minúscula	**Solo la primer letra de cada palabra reservada en mayúscula, el resto en minúscula**	**Todas las palabras reservadas en mayúscula**
program <identificador> **const** <definiciones de constantes> **type** <declaración de tipos de datos definidos por el usuario> **var** <definiciones de variables> **procedure** <definiciones de procedimientos> **function** <definiciones de funciones> **begin** <concatenación de sentencias> **end.**	**Program** <identificador> **Const** <definiciones de constantes> **Type** <declaración de tipos de datos definidos por el usuario> **Var** <definiciones de variables> **Procedure** <definiciones de procedimientos> **Function** <definiciones de funciones> **Begin** <concatenación de sentencias > **End.**	**PROGRAM** <identificador> **CONST** <definiciones de constantes> **TYPE** <declaración de tipos de datos definidos por el usuario> **VAR** <definiciones de variables> **PROCEDURE** <definiciones de procedimientos> **FUNCTION** <definiciones de funciones> **BEGIN** <concatenación de sentencias > **END.**

3. Declaración Constantes, Variables, Tipos y Acciones

Las constantes, variables, tipos, funciones y acciones usados en un programa deben ser expresivos, claros, definitivamente relevantes a la solución del problema, y legibles. Sus nombres deben ser significativos o mnemónicos. Aunque es posible escribir sus nombres de cualquier forma, es conveniente utilizar alguna convención.

Ejemplo	Identificador definido, y su uso
cont	Declaración de Variable. Deben comenzar siempre con una letra minúscula. Este estilo es el que usaremos.
variableLarga	Declaración de Variable. Este estilo es el que usaremos cuando usamos más de una palabra.
variable_larga	Declaración de Variable. "_" separa las palabras del identificador. No lo usaremos.
Blanco	Declaración de Constante. Deben comenzar siempre con una letra mayúscula.
Tpersona	Declaración de un tipo. La declaración de tipos comienza siempre con la letra T.
Ppersona	Declaración de un puntero. La declaración de punteros comienza siempre con la letra P.
Sumar(a,b)	Declaración de una acción. Una acción comienza en mayúscula.
PotenciaDos(x)	Declaración de una función. Una función comienza en mayúscula.

Hay que destacar que cuando se usan varias palabras para formar un identificador, éstas deben ser muy legibles: **estoSiSeVeBien**, **peroestocasiquenoseentiende**. ¿O si?
Los únicos identificadores que comienzan con una minúscula son las variables; todos los demás identificadores que no pueden cambiar comienzan con una mayúscula: constantes, tipos, acciones, funciones, etc.
No se debe restringir el tamaño de un identificador, sino más bien debe escogerlo para que sea significativo. Sin embargo, ante la posibilidad de escoger entre uno largo y otro corto, deberá escogerse el corto si tiene la misma expresividad del largo. Si queremos denotar la altura de una persona, es mejor usar el identificador **alt** que **alturaPersona**, siempre y cuando **alt** sea suficientemente claro en el contexto de programación. En general, cuando se trata de varias palabras, es mejor pegar las palabras es el estilo que se está imponiendo como estilo general cuando se trata de dos o tres palabras.
Es importante utilizar constantes para evitar el uso de números en las expresiones, por ejemplo: pago:= total * 1.1 * 0.25; pago:= total * ImpVentas * propina;
La expresión de la derecha es más clara y más simple de modificar.

En resumen: para declarar **variables** utilizaremos la notación **camelCase** y para **constantes**, **tipos**, **funciones** y **acciones** usaremos notación **PascalCase**.

4. PALABRAS RESERVADAS

Como mencionamos anteriormente, las **palabras reservadas** las podemos escribir tanto en **mayúscula** como en **minúscula**. Lo importante es adoptar una forma de escribirlas y hacerlo **siempre de la misma manera**.
El siguiente listado no es exhaustivo, incluso dependiendo del compilador pueden haber más.

Todas las palabras reservadas en minúscula	**Solo la primer letra de cada palabra reservada en mayúscula, el resto en minúscula**	**Todas las palabras reservadas en mayúscula**
and	And	AND
array	Array	ARRAY
begin	Begin	BEGIN
case	Case	CASE
const	Const	CONST
div	Div	DIV
do	Do	DO
else	Else	ELSE
end	End	END
file	File	FILE
for	For	FOR
function	Function	FUNCTION
if	If	IF
in	In	IN
not	Not	NOT
of	Of	OF
or	Or	OR
procedure	Procedure	PROCEDURE
program	Program	PROGRAM
readln	Readln	READLN
Read	Read	READ
record	Record	RECORD
repeat	Repeat	REPEAT
set	Set	SET
string	String	STRING
then	Then	THEN
to	To	TO
type	Type	TYPE
unit	Unit	UNIT
until	Until	UNTIL
uses	Uses	USES
var	Var	VAR
while	While	WHILE
with	With	WITH
Write	Write	WRITE
writeln	Writeln	WRITELN

5. Indentación

La indentación es la posición en que comienza una línea de código en el renglón. Es un anglicismo de uso común en informática. Por indentación se entiende mover un bloque de texto hacia la derecha insertando espacios o tabuladores para separarlo del texto adyacente. La indentación se utiliza para mejorar la legibilidad del código, teniendo en cuenta que los compiladores o intérpretes raramente los consideran. La indentación es uno de los factores más importantes para lograr mejores programas. Al indentar un programa adecuadamente se logra destacar su estructura de control. Son frecuentes las discusiones sobre cómo o dónde usar la indentación, si es mejor usar espacios en blanco o tabuladores. Para unificar la forma de escribir los algoritmos usaremos un estilo en particular, aunque no es el único.
Debe indentarse **dos (2) espacios** o **un (1) TAB** cada vez que se utilice una construcción anidada. Siempre debe indentarse una construcción anidada, y siempre debe indentarse en **dos espacios** o un TAB. La consistencia en la aplicación de esta regla permite a nuestro cerebro predecir la forma del programa, y hacer la lectura más sencilla.
La **declaración** de variables, constantes, tipos, acciones y funciones debe indentarse **dos (2) espacios** o **un (1) TAB**. El cuerpo principal del programa debe indentarse **dos (2) espacios** o **un (1) TAB**.

Ejemplo:

```
PROGRAM Operaciones;
VAR
  a, b: Real;
  sum, res, mul, di: Real;
BEGIN
   READLN (a);
   READLN (b);
   sum := a +b;
   res :=  a - b;
   mul := a * b;
   di := a / b;
   WRITELN ( 'Suma', sum, 'Resta', res, 'Multiplicación', mul, 'División', di)
END.
```

5.1. Indentación del IF

La indentación de IF – THEN que se utilizará en los programas será:

```
If <condición>
Then Begin
  <sentencias>
End
```

La indentación de IF – THEN – ELSE que se utilizará en los programas será:

```
If <condición>
Then Begin
  <sentencias>
End
Else Begin
   <sentencias>
End
```

5.2. Indentación del WHILE

La indentación de WHILE que se utilizará en los programas será:

```
While <condición de continuación> Do Begin
  <sentencias>
End
```

5.3. Indentación del REPEAT

La indentación de REPEAT que se utilizará en los programas será:

```
Repeat
  <sentencias>
Until <condición de terminación>
```

5.4. Indentación del FOR

La indentación de FOR que se utilizará en los programas será:

```
For i := a To n Do Begin
  <sentencias>
End
```

5.5. Indentación de PROCEDURE

```
Procedure <identificador> (lista de parámetros);
Const
  <definiciones de constantes>
Type
  <declaración de tipos de datos definidos por el usuario>
Var
  <definiciones de variables>
Procedure
  <definiciones de procedimientos>
Function
  <definiciones de funciones>
Begin
  <concatenación de sentencias>
End.
```

5.6. Indentación de FUNCTION

```
Function <nombre> (lista de parámetros): tipoDelValorDevuelto;
Const
  <definiciones de constantes>
Type
  <declaración de tipos de datos definidos por el usuario>
Var
  <definiciones de variables>
Procedure
  <definiciones de procedimientos>
Function
  <definiciones de funciones>
Begin
  <concatenación de sentencias>
End.
```

Nota: al menos una sentencia, del cuerpo de la función, debe ser una asignación que contenga en su parte izquierda el nombre de la función y en su parte derecha una expresión del mismo tipo que el especificado en la cabecera, es decir, **tipoDelValorDevuelto.**

6. Comentarios

Es difícil indicar exactamente cuándo incluir comentarios en el programa. Sin embargo, todos los comentarios deben cumplir con lo siguiente:

c1) Deben ser completos
c2) Deben ser válidos
c3) Deben ser pertinentes
c4) Deben ser claros
c5) Deben ser coherentes
c6) Deben ser concisos

En cuanto a la forma de los comentarios, es conveniente que estén alineados unos con otros. Si están dentro de un bloque, deben estar adecuadamente indentados, y siempre que sea posible debe dejarse un espacio separando a las llaves del comentario. Las llaves que enmarcan a un comentario deben estar alineadas. Por ejemplo:

```
VAR
  a, b : INTEGER; { factores de conversión }
  difDeA : REAL;   { diferencial en a }
```

Los comentarios deben explicar en relación al programa los siguientes asuntos:

- el cometido del programa. Esto debe reflejarse al comienzo del mismo.
- el papel de un identificador cuando no se explique por si mismo.
- en los subprogramas, si el parámetro es de entrada o de salida.

El cometido de un fragmento de programa se puede anticipar con un comentario:

```
Begin
  {lectura de datos}
   instrucciones para la lectura de los datos
  {cálculos}
   instrucciones para los cálculos
  {presentación de los resultados}
   instrucciones de salida de resultados
End.
```

Las propiedades requeridas en los puntos delicados del programa. Además de las precondiciones y las poscondiciones, a veces conviene anotar otras propiedades, intercaladas entre las demás instrucciones del programa, por ejemplo:

```
{x=Xo, y=Yo}
x := x+y;
{x=Xo+Yo, y=Yo}
y := x-y;
{x=Xo+Yo, y=Xo+Yo-Yo}
{x=Xo+Yo, y=Xo}
x := x-y;
{x=Xo+Yo-Xo, y=Xo}
{x=Yo, y=Xo}
```