Pascal

Daniel Delgado, *Estudiante, ITCR,* Wilbert Gonzales, *Estudiante, ITCR,*
Anthony Leandro, *Estudiante, ITCR,* and Bryan Mena, *Estudiante, ITCR*

1. Datos Historicos

Pascal fue creado en 1970 por Niklaus Wirth. Inicialmente estaba destinado a ser usado para enseñar a programar aunque su uso no se limita a esto, se han creado varias aplicaciones y videojuegos para PC como por ejemplo Dev-C++ y Skype los cuales fueron desarrollados, en parte, con Delphi (Object Pascal, una versión un poco más moderna de Pascal que implementa la Orentación a Objetos) además de sistemas embebido (sistema diseñado para realizar pocas funciones especificas en un sistema de manera constante) y fue ampliamente utilizado para desarrollo del computador Apple Lisa, Pascal estaba diseñado de manera que compilar mentalmente un programa se hiciera fácil esto debido a que el acceso a computadoras en aquella época era muy costoso. Cabe mencionar que la implementación de Orientación a Objetos de Pascal fue inicialmente desarrollada por Apple en 1986 pero fue hasta 1993 que Pascal Standards Committee publica la extención de OO para Pascal

Figura 1. Niklaus Wirth.

2. Tipos de Datos

En Pascal se utiliza la palabra *type* cuando se va a realizar una declaración de un tipo para asignarle la semantica al tipo (tipo de dato, rango de dato, operaciones posibles). Además, es importante recalcar que no todas las distribuciones de pascal soportan los mismos tipos de datos, por ejemplo, pascal estandar no posee soporte para cadenas de caracteres, aunque FreePascal lo hace

Nombre	Tipo de Dato	Ejemplo
String	Texto	'Lenguajes'
Integer	Números Enteros	6, 3545
Real	Números Decimales	3.14, 15.6
Boolean	Valor Booleano	TRUE, FALSE
Character	Un único carácter	'V', 'B'
Enumerados	–	ver cuadro1
Record	Datos heterogeneos	ver cuadro1

Cuadro 1. Código de tipo de datos enumerados y de registro en Pascal

```
type
//TipoMes Enumerado
TipoMes = (Ene, Feb, Mar, Abr, May,
Jun, Jul, Ago, Sep, Oct, Nov, Dec)
//TipoFecha y Cita Record
TipoFecha = record
Mes : TipoMes;
Dia : 1..3; //Subrangos, Revisar seccion 2.3
A\˜no : 1900..2000;
end;
Cita = record
Fecha : TipoFecha;
Hora : 0..2400;
end;
```

2.1. Enumerated

Define un rango de elementos a los que se refieren los identificadores(Ver cuadro 2). Es posible definir subrangos de ellos. Puesto que son ordenados, pueden compararse entre sí. Funciones aplicables a un identificador de un tipo Enumerated:[1]

1. Ord: Cantidad de ocurrencias de un identificador
2. Pred y Succ: Predecesor y sucesor de un identificador

Cuadro 2. Código de declaración en enumerated

```
type
<Nombre Tipo> =
    (<identificador>,...,<identificador>);
```

1. Datos tomados de www.gnu-pascal.de. Ver Refencias [7]

2.2. Arrays

Cuadro 3. Código de declaración para un array

```
type
typename =
        array [<Rango>] of <Tipo Dato>;
```

2.3. Punteros

Como en C o C++ un puntero es una dirección a memoria, de manera similar es necesario indicar a que tipo de dato se apuntará, en pascal esto se hace poniendo el carácter ^ antes del tipo de variable a la que se desea apuntar, luego se debe asignar un espacio en memoria que será donde el puntero viva, esto se hace con el comando New(Nombre_Puntero), aclarando que el puntero es de tipo puntero a un tipo de dato, para asignarle un valor al espacio de memoria al cual se esta apundando se necesita utilizar de nuevo el carácter ^ y asignar el valor, como ejemplo proponemos una lista implementada por medio de punteros:

Cuadro 4. Código de una lista con punteros

```
type
        punteroNodo = ^Nodo;
        Nodo = record
        dato : integer;
        siguiente : punteroNodo;
end;
```

2.4. Subrangos

Puede asociarsele con un "slice", toma un objeto y recupera la información desde Menor_Valor..Mayor_Valor, a continuación se presenta un ejemplo se subrangos con un tipo Enumerated:

Cuadro 5. Ejemplo de subrangos

```
type
Meses = (Enero, Febrero, Marzo, Abril
Mayo, Junio, Julio, Agosto,
Septiembre, Octubre, Noviembre,
Diciembre);
Subrango = Enero..Abril;
```

2.5. Puntos Importantes[2]

- Cuando se declara un string se le asigna una longitud máxima, de no asignarsele una el valor por defecto es de 255
- Para el nombre de las variables se siguen estos principios:
  *Debe empezar con una letra o un underscore(_)
  *No puede tener espacios en blanco
  *Los carácteres permitidos son letras, números o underscore

A continuación se presentan algunos de los rangos para Integers, igualmente cabe la aclaración que no todos los compiladores ni distribuciones de pascal son iguales, por lo tanto pueden existir inconsistencias en los siguientes datos:

Tipo	Rango	Bits
Byte	0..255	8
Shortint	-128..127	8
Integer	-32768..32767	16
Word	0..65535	16
Cardinal	0..2147483647	32
Longint	-2147483648..2147483647	32
Single	$1{,}5x10^{-45}..3{,}4x10^{38}$	32
Real	$2{,}9x10^{-39}..1{,}7x10^{38}$	48
Double	$5x10^{-324}..1{,}7x10^{308}$	64
Extended	$3{,}4x10^{-4932}..1{,}1x10^{4932}$	80
Comp	-9223372036854775809.. 9223372036854775807	64

3. Estructuras de control y Expresiones

En Pascal se utilizan bloques de codigo, inician con un *begin* y terminan con un *end*. Como dato curioso en Pascal *end* no necesariamente termina en ";", al contrario cuando se utiliza para terminar un archivo de pascal se utiliza un ".".
Ejemplo:

Cuadro 6. Código Hola Mundo en Pascal

```
program Hola;
begin
        writeln("Hola Mundo");
end.
```

3.1. Estructura típica de un programa

Cuadro 7. Estructura típica de un programa en Pascal

```
PROGRAM <Nombre del programa> (FileList);

CONST
        <Declarar Constantes>

TYPE
        <Declarar Tipos>

VAR
        <Declarar Variables>

BEGIN
        <Expresiones Ejecutables>
END.
```

2. Datos Obtenidos de *Pascal Data Types and Variables*, Computer Science Department at New York University

3.2. Constantes

Definidas en el bloque de constantes al inicio del programa, se utilizan identificadores para referenciarlas, el valor que almacenan no podrá ser cambiado a lo largo del programa

Cuadro 8. Declaración de Constantes

```
const
        Identificador1 = value;
        Identificador2 = value;
        Identificador3 = value;
        //...
```

3.3. Variables

Cuadro 9. Declaración de Variables

```
var
        Identificador1 : DataType1;
        Identificador2 : DataType2;
        Identificador3 : DataType3;
        //...
```

Cabe mencionar que los identificadores pueden ser "listas", o sea, varios identificadores separados por comas que serán declarados del mismo tipo

3.4. Tipos

Cuadro 10. Declaración de Tipos

```
type
        Identificador1 = <Type Definition>;
        Identificador2 = <Type Definition>;
        //...
```

3.5. Identación y Puntuación

Es necesario utilizar puntuación para decirle al compilador cuando un *statement* termina. Se utiliza ; en las siguientes situaciones:

- Despues de la declaración del programa
- Despues de cada definición de constante
- Despues de cada definición de variable
- Despues de cada definición de tipo
- Despues de cada *statement*

3.6. Comentarios en el códogo

A lo largo del tiempo en Pascal se han implementado diferentes maneras de comentar en el códogo, el comentario clásico inicia con (* y termina con *), lo anterior para comentarios de una sola línea,para comentarios multilínea se utilizan los {}, otras versiones de Pascal, por ejemplo Delphi, utilizan // para comentarios de una línea

3.7. Recursión

Como es necesario para un lenguaje de programación en Pascal es posible utilizar la recursión, importante recordar utilizar una condición de parada en la recursión para evitar que esta se vuelva infinita, seguidamente se presenta un ejemplo del uso de la recursión para resolver una suma:

Cuadro 11. Recursión en Pascal

```
function Suma (num : integer) : integer;
begin
        if num = 1 then
                Suma := 1
        else
                Suma := Suma(num-1) + num
end;
```

3.8. Estatutos *IF*

Cuadro 12. Sintaxis de un IF

```
var X, Y: Integer;
BEGIN
Writeln ("Digite dos numeros: ");
Readln (X,Y);
if (X >Y) then
        Writeln (X, " Es mayor que ",Y)
else
        Writeln (X, " Es menor que ", Y);
end.
```

3.9. Estatutos *CASE*

Cuadro 13. Sintaxis de un Case

```
Case <Caso> OF
<Caso con valor 1>: <Hacer ... 1>;
<Caso con valor 2>: <Hacer ... 2>;
...
Else
        <Hacer ... n>
End;
```

3.10. Estatutos *WHILE*

Cuadro 14. Sintaxis de un While

```
While (Conditional Expression) do
        <Hacer...>;
```

3.11. Estatutos *REPEAT*

Parecido al estatuto while, la diferencia recae en que primero realiza los estatutos declarados en su bloque y luego pregunta la condición booleana (Hacer luego preguntar)

Cuadro 15. Sintaxis de un Repeat

```
Repeat
        <Hacer...>;
Until (Conditional Expression)
```

3.12. Estatutos *FOR*

El estatuto for es utilizado cuando se conoce el numero exacto de repeticiones de un ciclo. Existen dos maneras de utilizarlo, desde el inicio hasta el final o desde el final hasta el inicio

Cuadro 16. Sintaxis de un FOR

```
//Inicia el for desde el
//valor inicial hasta el valor final
FOR <Variable> := <Inicio> TO <Final> DO
        <Hacer...>;

//Inicia el for desde el
//valor final hasta el valor incial
FOR <Variable> := <Final> DOWNTO <Inicio> DO
        <Hacer...>;
```

En 2005 Delphi implementa una nueva forma de realizar un *for* (similar al *for* en Python). Ejemplo

Cuadro 17. Sintaxis de un FOR en Delphi

```
begin
for <varName> in <ObjIterable> do
        <Hacer...>
end;
```

3.13. Estatutos *WITH*

Utilizado para accesar a las propiedades de un Record sin tener que referenciar al Record constantemente. Ejemplo:

Cuadro 18. Sintaxis de un WITH

```
with date do
        if month = 12 then
                begin
                        month := 1 ;
                        year := year + 1
                end
        else
                month := month+1
```

El código anterior remplaza el siguiente, eliminando la necesidad de referirse al Record cada vez que se va a accesar a alguno de sus campos.[3]

```
if date.month = 12 then
        begin
                date.month := 1 ;
                date.year := date.year+1
        end
else
        date.month := date.month+1
```

3. Ejemplos tomados del documento ISO 7185:1990 ver [6]

3.14. Procedures

Cuadro 19. Sintaxis de un Procedure

```
procedure <Nombre>(Lista de parametros);
        <Declaracion de constantes locales>
        <Declaracion de variables locales>
        begin
                <Hacer...>
        end;
```

Para llamar un procedure se utiliza su nombre más los datos para satisfacer la lista de parametros declarados.

3.15. Funciones

Cabe aclarar que funciones y Procedures no son lo mismo, procedures son un conjunto de instrucciones que se ejecutan cuando se llaman, las funciones son lo mismo, la diferencia recae en que las funciones tienen valores de retorno mentras que los procedures no.

Cuadro 20. Sintaxis de una función

```
function <Nombre>(Parametros):<TipoRetorno>;
        <Declaracion de constantes locales>
        <Declaracion de variables locales>
        begin
                <Hacer...>;
                <Nombre>: = return <Valor>;
        end;
```

3.16. Operadores Booleanos

Operador	Significado
and	Las dos condiciones evaluadas tienen que ser verdad
and then	Igual que *and* solo que garantiza que se evalueen las expresiones en el orden dado
or	Al menos una de las condiciones debe ser verdad
or else	Igual que *or* solo que garantiza que se evalueen las expresiones en el orden dado
not	Niega el resultado
xor	*or* exclusivo

3.17. Operadores condicionales

Es importante mencionar que la asignación en Pascal es ":=", y, como se muestra en el siguiente cuadro la condicional de igualdad es "="

Operador	Significado
=	Igualdad
>	Mayor que
<	Menor que
>=	Mayor Igual que
<=	Menor Igual que
<>	Diferente

3.18. Operadores Aritmeticos

Operador	Operación	Operandos	Tipo Resultado
+	Suma	Integer o Real	Integer si los dos operandos son integer, real de otra forma
-	Resta	Integer o Real	Integer si los dos operandos son integer, real de otra forma
$*$	Multiplicación	Integer o Real	Integer si los dos operandos son integer, real de otra forma
/	División	Integer o Real	Real
div	División truncada	Integer	Integer
mod	Modulo	Integer	Integer

3.19. Prioridad de algunas operaciones

1. not
2. $*$, $/$, div, mod, and
3. $+$, $-$, or
4. $<$ $>$ $<=$ $>=$ $=$ $<>$

4. Características

- Case Insensitive
- Lenguaje fuertemente tipado(Una variable es y será del tipo que se le declara y esto no se puede cambiar)
- Algunos compiladores solo le dan importancia a los primeros 32 carácteres (aproximadamente) de un identificador

5. Características y Ventajas

- Legible, es un lenguaje con una sintaxis que se asemeja mucho al inglés, esto hace que el código en Pascal sea mucho más fácil de aprender y de leer
- Variedad de estructuras de control y tipos de datos
- Versiones más modernas soportan la orientación a objetos

6. Desventajas

- Popularidad: Pascal no es un lenguaje tan popular como lo puede ser Java u otros, esto conduce a que librerias y otros aditivos sean escasos para solucionar cierto tipo de problemas
- Multiples Versiones sin un standard bien definido: esto hace que migrar aplicaciones de un ambiente a otro sea muy dificil incluso imposible, aun con versiones más modernas como Delphi no hay un standard definido

Referencias

[1] N. Wirth *Recollections about the development of Pascal.* In The second ACM SIGPLAN conference on History of programming languages (HOPL-II). ACM, New York, NY, USA, 333-342. DOI=http://ezproxy.itcr.ac.cr:2075/10.1145/154766.155378

[2] N. Wirth. 1975. *An assessment of the programming language PASCAL.* In Proceedings of the international conference on Reliable software. ACM, New York, NY, USA, 23-30. DOI=http://ezproxy.itcr.ac.cr:2075/10.1145/800027.808421

[3] N. Wirth. 1976. Comment on a note on dynamic arrays in PASCAL. SIGPLAN Not. 11, 1 (January 1976), 37-38. DOI=http://ezproxy.itcr.ac.cr:2075/10.1145/987324.987330

[4] N. Wirth *The Programming Language Pascal* Acta Informatica, Vol. 1, Fasc. 1, 1971 pp. 35-63

[5] B. Kernighan 1981 *Why Pascal is Not My Favorite Programming Language* Recuperado de: http://www.lysator.liu.se/c/bwk-on-pascal.html

[6] Pascal ISO 7185:1990 Recuperado de: https://www.iso.org/obp/ui/#iso:std:iso:7185:ed-2:v1:en

[7] *The GNU Pascal Manual* Recuperado de: http://www.gnu-pascal.de/gpc/