INSTITUTO TECNOLÓGICO DE COSTA RICA

TAREA 2 - PASCAL

Ariel Herrera
Saúl Zamora

profesor
M. Sc. Saúl Calderón Ramírez

August 24, 2016

I. Datos históricos

El lenguaje de programación Pascal está nombrado en honor al matemático francés Blaise Pascal y fue desarrollado por Niklaus Wirth. Antes de su trabajo en Pascal, Wirth desarrolló *Euler* y *ALGOL W* y luego trabajó en lenguajes similares a pascal como *Modula-2* y *Oberon*.

Pascal fue el lenguaje de alto nivel mayormente utilizado en el desarrollo de *Apple Lisa* y también durante los primeros años de *Macintosh*.

II. Importancia y usos

Inicialmente, Pascal era muy utilizado para la enseñanza de la programación. Generaciones de estudiantes usaron Pascal como un lenguaje introductorio durante la universidad. Variantes de Pascal han sido utilizadas frecuentemente para todo desde proyectos de investigación orientados a juegos de PC y sistemas embebidos.

Object Pascal es utilizado hoy en día para el desarrollo de aplicaciones Windows pero también tiene la habilidad de compilar el mismo código a Mac, iOS y Android. Otra versión multiplataforma llamada *Free Pascal*, junto con el IDE *Lazarus*, es popular entre los usuarios Linux ya que ofrece el tipo de desarrollo de *escribe el código una vez y compilalo donde sea*.

III. Tipos de datos

Al igual que en otros lenguajes, un tipo en Pascal define un rango de valores que una variable es capaz de almacenar y también define que operaciones pueden ser ejecutadas sobre dicha variable.

- *integer*: números enteros
- *real*: números de punto flotante
- *boolean*: valores de falso o verdadero
- *char*: un caracter de un set ordenado de caractéres
- *string*: grupo o hilera de caractéres
- *subrango de tipos*: se pueden contruir subrangos de cualquier tipo de dato ordinal

```
var
  x : 1..10;
  y : 'a'..'z';
```

- *sets*: en contraste con otros lenguajes de tu tiempo, Pascal soporta tipos de datos set

```
var
  Set1 : set of 1..10;
  Set2 : set of 'a'..'z';
```

- *declaraciones de tipos*: los tipos pueden ser definidos usando otros tipos por medio de declaraciones

```
type
  x = integer;
  y = x;
```

O declaraciones más complejas:

```
type
  a = array[1..10] of integer;
  b = record
        x : integer;
        y : char
      end;
  c = file of a;
```

- *tipo archivo*: como se mostro en el código anterior, en Pascal, los archivos son secuencias de componentes
- *punteros*: Pascal soporta el uso de punteros

```
type
  pNode = ^Node;
  Node  = record
            a : integer;
            b : char;
            c : pNode  {extra semicolon not strictly
                 required}
            end;
var
  NodePtr : pNode;
  IntPtr  : ^integer;
```

IV. Expresiones

Las expresiones en Pascal se dan en asignaciones o pruebas. Las expresiones producen un valor de un cierto tipo. Son construidas con dos componentes: los operadores y los operandos. Usualemente los operadores son binarios; requieren dos operandos. También hay operadores unarios, que requieren un único operando.

TABLE I
Precedencia de operadores

Operador	Precedencia	Categoría
Not, @	Más alta	Operadores unarios
/ div mod and shl shr as <<>>	Segunda	Múltiples operadores
+ - or xor	Tercera	Operadores de adición
= <><><= >= in is	Más baja	Operadores relacionales

V. Almacenamiento

A. Almacenamiento de pila

En los progrmas en Pascal, el almacenamiento de pila se ubica luego del almacenamiento estático y crece hacia abajo, hacia el fondo del área de memoria en el que se encuentra ubicado. La ubicación de almacenamiento de pila es automática cada vez que se invoca un subprograma y la recuperación también es automática cuando el subprograma termina.

B. Almacenamiento de montículo

En Pascal, el manejo del montículo o *heap* es hecho explícitamente por el programa, este se asignado cuando es necesario y deasignado para recuperación durante la ejecución del programa.

VI. Estructuras de control

Pascal es un lenguaje estructurado, lo que significa que el control de flujo está estructurado en estatutos estándares, usualmente sin el comando *goto*.

```
while a <> b do  WriteLn('Waiting');

if a > b then WriteLn('Condition␣met')   {no semicolon
     allowed!}
          else WriteLn('Condition␣not␣met');

for i := 1 to 10 do  {no semicolon for single statements
     allowed!}
  WriteLn('Iteration:␣', i);

repeat
  a := a + 1
```

```
until a = 10;

case i of
  0 : Write('zero');
  1 : Write('one');
  2 : Write('two');
  3,4,5,6,7,8,9,10: Write('?')
end;
```

VII. Características principales

Pascal es un lenguaje de programación estructurado fuertemente tipado, lo cual impleca que:

- El código está dividido en funciones o procedimientos, de esta forma, se facilita el uso de la programación estructurada en oposición al antiguo modo de programación monolítica.
- El tipo de dato de todas las variables debe ser declarado previamente para que su uso sea posible.

VIII. Características distintivas

- En Pascal, el tipo de una variable se fija en su definición, la asignación de variables de valores de tipo incompatible no está autorizada. Esto previene errores comúnes donde variables son usadas incorrectamente porque el tipo es desconocido; y también evita la necesidad de la *notación húngara*, en la que se ponen prefijos a los nombre de las variables para indicar su tipo.

IX. Ventajas y desventajas

A. Ventajas

- Pascal no permite asignaciones dentro de las expresiones y utiliza sintáxis distintas para asignaciones y comparaciones evitando de esta manera muchos bugs.
- El tipo de la variable está fijado en su definición.
- El programa tiene dos partes definidas: la declarativa y la ejecutiva.
- Facilidad de uso.

B. Desventajas

- Durante los años 80 y principio de los 90, Pascal fué criticado por no producir código industrial.
- En la actualidad, es prácticamente obsoleto en comparación con otros lenguajes más modernos y poderosos.

X. Ejemplo

Ejemplos de programas en Pascal:

- Hola mundo

```
PROGRAM HolaMundo (OUTPUT);
BEGIN
  WriteLn(' Hola Mundo!')
  { como la siguiente instruccin no es ejecutable "
      end."
  no se requiere ; aunque puede ponerse seg n las
  preferencias del programador }
END.
```

- Suma

```
PROGRAM Suma (INPUT, OUTPUT);

VAR
  Sumando1, Sumando2,Suma:INTEGER;

BEGIN
  Writeln('ingrese un numero:');
  ReadLn(Sumando1);
  Writeln('ingrese otro numero:');
  ReadLn(Sumando2);
  Suma:=Sumando1 + Sumando2;
  WriteLn ('La suma es:',Suma);
  Write ('Pulse [Intro] para finalizar...');
  readkey
END.
```

- Raíz cuadrada

```
PROGRAM Raiz (INPUT, OUTPUT);
(* Obtener la ra z cuadrada de un n mero real x
    cualquiera.*)

VAR
  Valor, Resultado: REAL;
BEGIN
  WriteLn ('** Calcular la ra z cuadrada **');
  Write ('Introduzca el valor: '); ReadLn (Valor);
(* Ra z cuadrada del valor absoluto de x para evitar
    ra ces imaginarias *)
  Resultado := sqrt (abs (Valor));
  Write ('La ra z cuadrada de', Valor, ' es ');
  IF Valor < 0 THEN (* Si es negativo, el resultado es
      imaginario *)
    WriteLn (Resultado ,'i')
  ELSE
    WriteLn (Resultado);
  Write ('Pulse [Intro] para finalizar...');
END.
```

- Bucles

```
PROGRAM MultiplosDe3 (INPUT, OUTPUT);

VAR
  Numero, Cnt: INTEGER;

BEGIN
  Cnt := 0;
  Write  ('Entra el primer n mero de la serie: ');
      ReadLn (Numero);
  WHILE Numero <> 0 DO
  BEGIN
    IF (Numero MOD 3) = 0 THEN
      INC (Cnt);
    Write ('Dame otro numero (0 para terminar): ');
        ReadLn (Numero);
  END;
  WriteLn ('La cantidad de m ltiplos de 3 ingresados
      es', Cnt);
  Write ('Pulse [Intro] para finalizar...');
END.
```

- Funciones y recursividad

```
PROGRAM CalcularFactorial (INPUT, OUTPUT);

(* Funci n que calcula el factorial de n (n!) de forma
    recursiva. *)
  FUNCTION Factorial (CONST N: INTEGER): INTEGER;
  BEGIN
    IF N > 1 THEN
      Factorial := N * (Factorial (N - 1))
    ELSE
      Factorial := N;
  END;

VAR
  Base: INTEGER;
BEGIN
  Write ('Valor de N:'); ReadLn (Base);
  WriteLn ('N! =', Factorial (Base));
  Write ('Pulse [Intro] para finalizar...');
END.
```

- Vectores

```
PROGRAM NotasDeAlumnos;
uses crt;
Type
vecalumnos = array [1..40] of string;
var
Nombre, Apellido: vecalumnos;
Nota: array [1..40] of real;
Begin
clrscr; /*Limpia pantalla*/
For i:= 1 to 40 do
  begin
   write( Ingrese Nombre:   );
   readln(Nombre[i]);
   write( Ingrese Apellido:   );
   readln(Apellido[i]);
   write( Ingrese Nota:   );
   readln(Nota[i]);
  end;
For i:= 1 to 40 do
  begin
  write(Nombre[i],      ,Apellido[i]);
  if (Nota[i] >=7) then
  writeln(   aprob  )
  else
  writeln(   no aprob  );
  end;
writeln(    );
Write ('Pulse [Intro] para finalizar...');
Readln;
end.
```

XI. Referencias

References

[1] Pascal (programming language) (2016). . In *Wikipedia*. Retrieved from https://en.wikipedia.org/wiki/Pascal_(programming_language)

[2] Pascal (lenguaje de programacin) (2011). . In Wikipedia. Retrieved from https://es.wikipedia.org/wiki/Pascal_(lenguaje_de_programacin)

[3] 12 expressions. (2015, November 14). Retrieved August 25, 2016, from http://www.freepascal.org/docs-html/ref/refch12.html

[4] Storage management. Retrieved August 25, 2016, from https://people.cs.clemson.edu/~turner/courses/cs428/summer98/section1/webct/content/pz/ch5/ch5_4.html