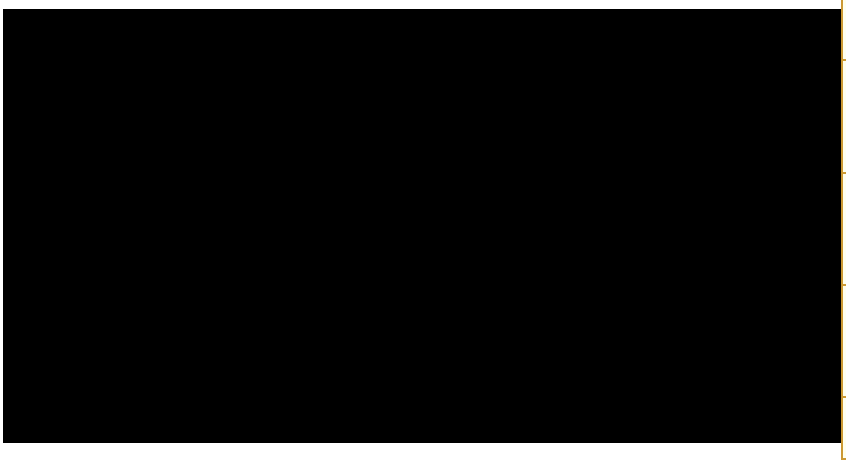
Capítulo 7. Lenguajes de programación

7.1 Conceptos generales

hardware *software*

González



- **compilado**

C ++

programa fuente

análisis léxico

-
-
-

```
200 FOR I=1 TO N STEP 1
210 LET A(I)=0: REM Inicializa la matríz
220 NEXT I
```

```
[L45] [T16] I=1 [T17] N [T18] 1 \
[L46] [T09] X(I)=0 \
[L47] [T19] I \
```

Análisis sintáctico

parsing

<Línea progAlg < í ero LTDíne

```
Nombre de la variable Tipo Dirección
  X   N  3A2F
```

código,

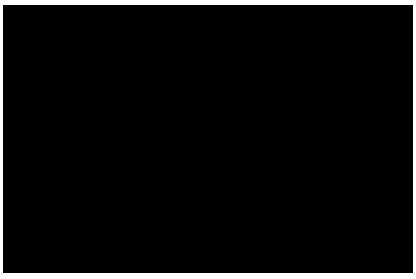
tokens

```
Línea 200
LOA X N+1 Emplea el registro índice como
contador
SIG CMP X N Compara con N
BGT FIN Al terminar vete a FIN
Línea 205
CLR A Pone a cero el acumulador
STD A D Y Copia el contenido del
acumulador en Y+índice
Línea 210
INC X Incrementa el índice
BRN SIG Continúa el bucle
FIN
```

optimización

linkado

editor de ligaduras



7.2 Metodologías de programación

software

-
-
-
-
- *software*

software

Ingeniería del Software

Ingeniería del Software **ciclo de vida**

-
-
-
-
-
-

Refinamiento Progresivo

-
-
-
-

"Descomposición Funcional

métodos

formales

7.3 Tipos de lenguajes

Alto Nivel

Bajo Nivel

lenguaje máquina

Ensamblador

COBOL, FORTRAN Y ALGOL60

declarativos **imperativos**

Prolog.

Hola Mundo

En ensamblador de x86 para GNU/Linux (nasm)

En C++

Anexo - Ejemplo de un lenguaje:

Pascal

El **Pascal** recibe su nombre en honor del matemático y filósofo francés Blas Pascal (1623-1662). Se desarrolló entre los años 1968 y 1971.

En contraste con FORTRAN, COBOL O ALGOL 60, el Pascal es resultado del trabajo de una sola persona, Niklaus Wirth (1934-), profesor universitario (actualmente jubilado) de Zurich (Suiza). A partir del Algol60 diseñó un lenguaje adecuado para enseñar computación a los estudiantes universitarios. Según su creador su desarrollo se debe a dos motivos principales. El primero es el de proporcionar un lenguaje adecuado para enseñar a programar de forma sistemática, a partir de unos cuantos conceptos fundamentales que se reflejen de forma clara y natural en el lenguaje. El segundo es desarrollar implementaciones de este lenguaje que funcionen de forma fiable y eficiente sobre los ordenadores disponibles actualmente.

Enlace recomendado:
Curso de Pascal. Centro de Cálculo, Universidad de Zaragoza (PDF)

Partes de un programa en Pascal

Un programa se subdivide en varias partes:

- La cabecera, suele ser opcional, en ella se indica el nombre del programa
- Definiciones y declaraciones, son opcionales y pueden categorizarse como:

- Definiciones de tipos
- Definiciones de constantes
- Declaraciones de variables
- Declaraciones de etiquetas
- Definiciones de subprogramas

La cabecera

Independientemente de lo largo o pequeño que sea, todo programa debe comenzar con una cabecera de programa. Esta cabecera es la primera línea, comienza con la palabra clave `Program` seguida del nombre y terminada por `;`.

Definiciones y declaraciones

Lo normal es que un programa necesite datos. Un dato es cualquier información que sea manipulada por un programa y necesite un espacio de almacenamiento en el ordenador al ejecutarse el programa. Los datos que no cambian nunca se denominan constantes, mientras que los que cambian se conocen como variables. Para reservar espacio en la memoria se han de declarar mediante una declaración de variables. Los identificadores deben comenzar con un carácter alfabético o el símbolo de subrayado. El primer carácter puede estar seguido de hasta 126

adicionales, alfabéticos, numéricos o de subrayado.

Tipos

En Pascal además de dar identificadores para las variables se debe de indicar el tipo de los datos que se usará para esa variable. El tipo de un dato tiene dos propósitos, una asignar cuánta memoria se va a asignar a un dato de tipo particular. El otro es prevenir un mal emparejamiento de tipos.

En Pascal, los tipos pueden categorizarse en predefinidos y definidos por el usuario.

Los tipos predefinidos son:

- Boolean, admite los valores FALSE y TRUE
- Char, son caracteres del conjunto de códigos ASCII
- Integer, son números en el rango -32768 a 32767
- Byte, es un subrango del tipo Integer, de 0 a 255
- Real, son números reales en el rango 1E-38 a 1E+38

Constantes

Se usan para asociar un valor fijo a un identificador. Este puede estar definido por el usuario o puede ser literal que describe un valor. Por otra parte las constantes definidas por el usuario, llamadas constantes con nombre deben definirse en la parte de definición de constante, como por ejemplo, `const PUERTO = 98`. Cons es la palabra reservada correspondiente. A su vez la constante se puede definir con tipo, como por ejemplo, Const PUERTO: Integer 98;

Variables

Cuando se declara una variable, se le asigna memoria, suministrándole un lugar para poner un tipo de dato. Una declaración también suministra un nombre para ese lugar. La palabra clave es Var, pudiendo aparecer a continuación uno o más identificadores de variables, separados por comas y los tipos se separan con dos puntos (:), a continuación se muestran ejemplos,

```
Var tipo_interes
cant_nominal
principal : Real;
año : Integer;
mes : Byte;
inchr : Char;
salida : Boolean;
Var     tipo_interes,     cant_nomina,
principal: Real;
```

Constantes con tipo

Pueden usarse para ahorrar memoria y como variables inicializadas. A continuación se da un ejemplo:

```
Const TEXTO="T";
```

```
MAYUSCULA="U";
peso : Byte = 80;
tipo_entrada : Char = MAYUSCULA;
```

Operadores

Gran parte de la manipulación de datos en un programa se ejecuta mediante operadores. Los operadores se clasifican en tres categorías, aritméticos, lógicos y relacionales.

El término expresión se refiere a una combinación de una o más constantes literales o con nombre o identificadores de variables, con uno o más operadores.

Cuando aparece más de un operador hay un orden de precedencia, y si es al mismo nivel la evaluación se efectúa de izquierda a derecha. Las precedencias son:

1 - (menos unario)
2 not
3 *, /,div,mod,and,shl,shr
4 *,-,or,xor
5 =,<,<„<=,=,in

Sentencias

La parte de sentencias es en donde se define la lógica de un programa, es donde se manipulan los datos para obtener el resultado deseado por el usuario del programa. Las sentencias se pueden clasificar en los tipos:

- Sentencias de asignación
- Sentencias compuestas
- Sentencias de control de la lógica

Una sentencia de asignación consta de un identificador de variable (o función) seguido del operador de la asignación (:=) seguido de una expresión. A continuación se indican ejemplos,

```
salida:=FALSE
interes:=10
pago:=principal/periodo
```

Una sentencia compuesta se ha de identificar en el principio con Begin y al final con end;. Una sentencia de este tipo puede estar formada por todos los tipos de sentencias incluyendo otras sentencias compuestas.

Sentencias de control de lógica

- *Bucle while*. Ejecuta una sentencia repetidamente hasta que la condición del bucle da FALSE. La forma es, `While condicion Do sentencia`

```
While condicion Do
 Begin
   sentencia 1
   sentencia 2
   ......
```

```
 End;
```

- *Bucle Repeat-Until.* Se usa cuando es necesario que el bucle se ejecute al menos una vez. En este caso la verificaciónde la condición ofrece esa posibilidad. La forma del bucle es,

```
 Repeat
  sentencia 1
  sentencia 2
  ......
 Until condicion
```

- *Bucle For-Do*. Es ideal para los casos en que está predeterminado el número de veces que se ejecutará una sentencia. La forma es, `For contador:= exp1 To exp2 Do sentencia`

```
 For contador:=exp1 To exp2 do
  Begin
   sentencia1
   sentencia2
   .......
  End;
```

- *If-Then-Else*. Esta sentencia ejecuta una, muchas o ninguna sentencia dependiendo de la evaluación de la condición establecida. La forma general es,

```
If condicion Then sentencia1 [Else sentencia2]
```

- Sentencia Case. La forma de esta sentencia es,

```
Case selector Of
 Constante1 : sentencia1;
 Constante2 : sentencia2;
 Constanten : sentencian;
Else
 sentencia;
```

Cuando se ejecuta una sentencia Case, el valor del selector, que es una variable y puede ser cualquier tipo escalar excepto Real, se usa para determinar cuál, si la hubiera de las, sentencias del Case se ejcuta. Las constantes asociadas con la sentencia deben de ser del mismo tipo que la variable selector

Subprogramas en Pascal

Los subprogramas son como pequeños programas dentro de un programa, incluso se parecen a los programas por comenzar con una cabecera, ir seguida de las declaraciones y una parte de sentencias. Los subprogramas se encuentran en la parte de declaración del programa, sin embargo operan en su propio entorno, separados pero no siempre aislados del programa principal. Un subprograma puede declarar variables que son sólo accesibles desde el subprograma, puede usar variables del programa principal y puede usar una mezcla de los dos.

Los subprogramas son de dos tipos: procedimientos y funciones.

Procedimientos

Los procedimientos comienzan por la palabra reservada Procedure seguida de un identificador y un punto y coma. Después de la cabecera viene la parte de declaración. Al ejecutar un procedimiento se crean en memoria las constantes y variables de la parte de declaración, pero después de que se haya terminado el procedimiento desaparecen, lo cual supone un ahorro de memoria.

A continuación de la declaración están las sentencias, entre las palabras clave Begin y End y un punto y coma. Los procedimientos se activan en el programa principal dando su nombre como una sentencia simple.

Con frecuencia los procedimientos necesitan que se les dé uno o más valores, que se conocen como parámetros. Estos se incluyen en la cabecera, dentro de los paréntesis que siguen al identificador. Las declaraciones de parámetros son como las declaraciones de variables en la forma y propósito. Cada identificador se separa por punto y coma, cada uno se ha de definir como un tipo con nombre. A continuación se muestra un procedimiento que repite un carácter en pantalla, el número de veces que se desee,

```
Procedure Repite(caracter; Car,
conta: Integer);
 Var i:Integer;
 Begin
  For i:=1 to conta Do
       Write(caracter);
 End;
```

y así para invocar que aparezca la letra A 80 veces seguida en pantalla,

```
Repite ("A",80)
```

donde se han pasado dos parámetros. Estos parámetros pueden ser de otro tipo, como variables, expresiones, como por ejemplo,

```
caracter:="A"; conta:=80;

Repite(LowerCase(caracter),conta);
```

donde LowerCase es una función que saca la letra minúscula de la indicada.

A su vez los parámetros se pueden pasar por valor o referencia. Si se pasa por valor, se transfiere una copia del valor del parámetro actual al procedimiento, el procedimiento no accede a la variable [illegible]

queda sin modificar el valor actual del parámetro original.

Si se desea pasar un parámetro por referencia, el parámetro actual debe ser una variable o una constante con tipo. Delante de la declaración del parámetro formal hay que poner la palabra clave `Var`.

Seguidamente se da un ejemplo para cada una de las dos formas de paso de parámetros,

a) Por valor

```
Procedure incrementa(mas_uno:Integer)

 Begin
  mas_uno:=mas_uno+1;
      WriteLn(mas_uno);
 End;
```

y al ejecutar las instrucciones siguientes,

```
avance:=1;
incrementa(avance);
    WriteLn(avance)
```

la salida sería de 2 y 1.

b) Por referencia

```
Procedure                incrementa(Var
mas_uno:Integer)

 Begin
  mas_uno:=mas_uno+1;
      WriteLn(mas_uno);
 End;
```

y al ejecutar las instrucciones siguientes,

```
avance:=1;
incrementa(avance);
    WriteLn(avance)
```

la salida sería de 2 y 2.

Funciones

La diferencia entre un procedimiento y una función es que el identificador de una función asume un valor cuando la función ha terminado y devuelve este valor a la rutina que la llamó en sustitución del nombre de la función. En un sentido amplio el nombre de la función es también una variable. Todo lo indicado previamente respecto a la definición de los procedimientos se aplica a las funciones. La palabra clave es `Function`. A continuación se indica un ejemplo de función que convierte un parámetro de tipo Char a letra minúscula, si dicho parámetro está en el rango de la A a la Z.,

```
Function
miniscula(caracter:Char);Char
```

```
 Begin
  If caracter in  "A".."Z" Then

minuscula:=Char(Ord(caracter)+32)
  Else
       minuscula:=caracter;
 End;
```

Otras posibilidades

El Pascal ofrece muchas más posibilidades, como por ejemplo la definición de tipos por el usuario, o el tipo Set (conjunto), de gran interés dado que no aparecen en la mayoría de los lenguajes. Otro tipo a destacar es el denominado puntero. La recursividad es una característica de mucha utilidad, es decir que cierta instrucción se puede llamar a si misma. Aquí no se tratan estas posibilidades aunque se usan ampliamente en la programación a escala avanzada en Pascal.

Bibliografía (Disponible en la Biblioteca Universitaria):

- Appleby, Doris, Lenguajes de programación : paradigma y práctica / Doris Appleby, Julius J. Vandekopple 1998
- Berger, Marc. Graficación por computador con Pascal / Marc Berger. -- Argentina, etc. : Addison-Wesley Iberoamericana, cop. 1991
- Biondi, Joëlle, Algorítmica y lenguajes / Joëlle Biondi, Gilles Clavel 1985
- Burns, Alan Sistemas de tiempo real y lenguajes de programación -- Madrid : Addison-Wesley, cop. 2003
- Cueva Lovelle, Juan Manuel...[et al.]. Introducción a la programación estructurada y orientada a objetos con Pascal -- Oviedo : Departamento de Matemáticas
- Cuevas Agustín, Gonzalo. Teoría de la información, codificación y lenguajes Córdoba, Argentina : SEPA, Sociedad para Estudios Pedagogicos Argentinos , cop. 1985
- Lenguajes HTML, JAVA Y CGI : el diseño de páginas web para internet a su alcance. -- Madrid : Abeto, 1996
- Findlay, W. Pascal : programación metódica W. Findlay, D. A. Watt. -- [2ª ed.]. -- Madrid : Rueda, 1984
- Grogono, Peter, Programación en PASCAL. -- Ed. revisada. -- Argentina, etc. : Addison-Wesley, cop. 1986
- Sánchez Dueñas, G. Compiladores e intérpretes.Un enfoque pragmático 2ª ed. Díaz de Santos, Madrid 1989
- Algunos programas de uso común en Pascal. -- Madrid : McGraw-Hill, D.L. 1982
- Lenguajes HTML, JAVA Y CGI : el diseño de páginas web para internet a su alcance. -- Madrid : Abeto, 1996
- TRUCS et astuces pour turbo Pascal. -- Paris : Micro-Application, 1986

Enlaces relacionados con la temática del capítulo:

- Artículos y tutoriales. Lenguaje Pascal
- Bjarne Stroustrup's homepage!
- El Máquinas
- Hello World. ACM

- Metodología y Tecnología de la Programación. Fac. Informática. Univ. Murcia
- Raúl Monge Anwandter. Univ. F. Sanat María. Chile
- Metodologías de Desarrollo de Software. Fac. Informática. Univ. Murcia
- Metodología y Tecnología de la Programación I. Vicente López. ETSI. U A. Madrid
- The Computer Language Benchmarks Game
- The Languaje Guide
- Sitios donde se originaron los lenguajes de programación (mapa en Google Maps).
- Vilecha.com. Tú albergue en la red

[redacted] fresqui [redacted] del.icio.us [redacted] meename

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

C | Us Magazine - Cancer-Stricken Michael C. Hall:

powered by Google™

Loading...

[redacted]

Rafael Menéndez-Barzanallana Asensio
Departamento Informática y Sistemas. Universidad de Murcia

Actualizado 2009/08/14

[redacted]

[redacted]