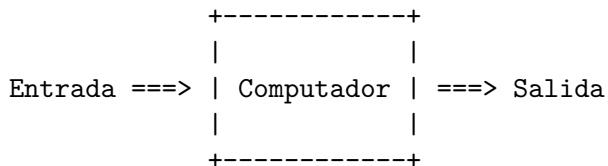


Componentes Básicos

Programación 1 - InCo

Modelo de Computación

Vemos al computador como un procesador de *datos*.



Organización de la Computadora

- Dispositivos de entrada
 - Teclado
 - Mouse
 - Pantalla Táctil
 - Micrófono
- Dispositivos de salida
 - Monitor
 - Impresora
 - Parlante
- Memoria: interna, externa.
- Unidad Central de Proceso (UCP)

Unidad de Control:

- Carga instrucciones en memoria (programa)
- Ejecuta las instrucciones

Unidad Aritmética Lógica

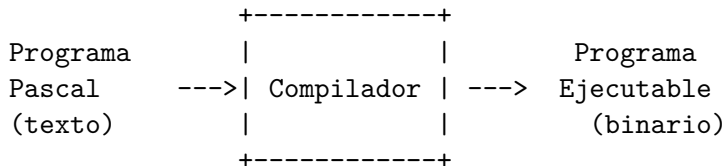
- Ejecuta operaciones aritméticas y lógicas.

Compilación

Lenguajes de alto nivel: pascal, java, C, python, etc.

La computadora no “*entiende*” los lenguajes de alto nivel.

Un **compilador** es un programa que traduce a código de máquina:



Al archivo que contiene el código Pascal se le llama también programa o código **fuente**.

Ejemplo de un programa pascal

```
program Triangulo;  
var  
    altura,base,area: real;  
begin  
  
    { ingresar datos }  
    readLn(altura,base);  
  
    { calcular area }  
    area := base * altura / 2;  
  
    { mostrar resultado }  
    writeLn(area)  
end.
```

Sintaxis de los identificadores

Diagrama sintáctico

identificador

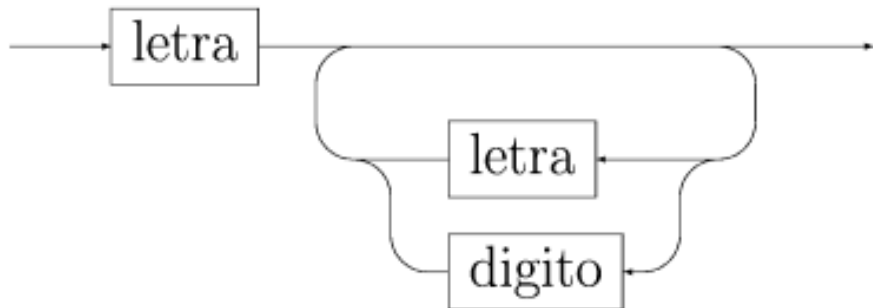


Figure 1: Identificador

Sintaxis de identificadores (BNF)

BNF:

identificador = letra { letra | digito }

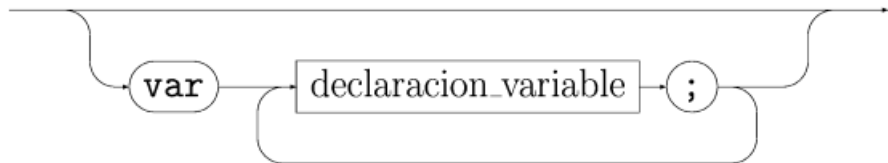
Lenguaje Natural:

Un identificador es una secuencia de caracteres alfanuméricos el primero de los cuáles debe ser alfabético.

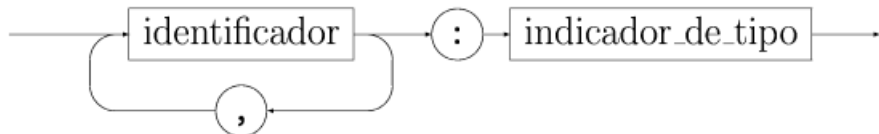
Sintaxis de la declaración de variables (Diagrama)

Diagrama

declaracion_de_variables



declaracion_variable



Sintaxis de la parte algorítmica

```
cuerpo = 'begin'  
        instruccion  
        { ';' instruccion }  
        'end' .
```

Observación: El punto y coma es un *separador* y no *finalizador* de instrucción.

En el ejemplo tenemos 3 tipos de instrucciones:

- Entrada: `ReadLn`
- Asignación: `area := base * altura / 2`
- Salida: `WriteLn`

Se verán otras a lo largo del curso.

Asignación

Es una instrucción que permite modificar el contenido de una variable:

```
identificador := expresion
```

identificador representa la variable que va a ser asignada.

expresión representa el valor que se asigna a la variable.

Sus tipos deben ser *compatibles*.

El Concepto de Tipo

- Los *tipos* permiten indicar la característica de los valores (datos) manipulados en un programa.
- Toda variable o constante tiene asociado un tipo.
- Esto ocurre también con los operadores, las funciones estándar y las definidas por el usuario.
- Poseer tipos permite detectar ciertos errores de construcción en el código (chequeo de tipos).

Por ejemplo:

3 + 4	tiene tipo correcto (integer)
3 + 'a'	tiene error de tipo

- En Pascal el chequeo de tipos se hace en tiempo de *compilación*.

- Los tipos según su estructura:
 - *elementales*: cada valor es simple e indivisible.
 - *estructurados*: cada valor está compuesto por varios valores.
- Otra clasificación:
 - estándar o predefinidos
 - definidos por el programador

Tipos Elementales Predefinidos

<code>integer</code>	Números enteros.
<code>real</code>	Números reales
<code>boolean</code>	El conjunto { <code>true</code> , <code>false</code> }
<code>char</code>	Los caracteres: letras, dígitos, símbolos, etc.

El tipo `integer`

Representa números enteros con o sin signo.

Ejemplos: `-32`, `0`, `+123`, `77`.

Es un tipo **acotado**. El máximo es `maxint`.

Expresiones aritméticas enteras

Una expresión se construye mediante: *constantes*, *variables*, *operadores* y *funciones*.

Operadores enteros

- Suma: +
- Resta: -
- Multiplicación: *
- División: `div`
- Módulo: `mod`.

El tipo real

Representa números “*reales*”.

Las constantes se pueden representar en notación *decimal* o notación *exponencial*.

decimal	exponencial
358.3	3.583E2
0.23	2.3e-1

Operadores aritméticos reales

operación	símbolo
suma	+
resta	-
multiplicación	*
división	/

El mismo símbolo es usado para denotar operaciones sobre tipos diferentes. Por ejemplo, los símbolos $+$, $-$ y $*$ denotan la suma, resta y multiplicación tanto de enteros como de reales, respectivamente.

Coerción

Argumentos que no son del tipo requerido por una función u operador son convertidos al tipo correcto.

Por ejemplo, es posible mezclar operandos de tipo entero y real en algunos operadores aritméticos. La *conversión* es automática.

Ejemplos:

expresión	conversión
$3 + 2.5$	$3.0 + 2.5$
$3.0 * 2$	$3.0 * 2.0$
$5 / 2.6$	$5.0 / 2.6$
$(5 + 2) / 2$	$7.0 / 2.0$

Coerción en asignaciones

En una asignación $x := e$ el tipo de la variable x y de la expresión e debe ser el mismo.

Excepción: *Es posible asignar un valor entero a una variable real.
La conversión de entero a real se realiza en forma automática.*

Precedencia de operadores

Para la evaluación de expresiones aritméticas se debe seguir este orden:

- 1 Evaluar expresiones parentizadas
- 2 Aplicar operaciones de multiplicación y división ($*$, $/$, div , mod). Si hay varias en secuencia, entonces evaluarlas de izquierda a derecha.
- 3 Aplicar operaciones de suma y resta ($+$, $-$). Si hay varias en secuencia, entonces evaluarlas de izquierda a derecha.

Funciones aritméticas estándar (predefinidas)

- `sqr(x)` - retorna el cuadrado de `x`.
- `sqrt(x)` - retorna la raíz cuadrada de `x`.
- `trunc(x)`, `round(x)` - conversión de real a entero.
- `abs(x)` - valor absoluto.

El tipo boolean

Constantes: `true`, `false`.

Operadores:

- `and` - conjunción
- `or` - disyunción
- `not` - negación

El tipo char

Cada valor del tipo `char` es un carácter simple.

Los literales se representan entre comillas simples.

- Letras mayúsculas y minúsculas: `'A'` `'B'` `'z'` `'h'`
- Dígitos: `'0'` `'1'` `'2'` `'3'`
- Símbolos: `'*'` `'@'` `'&'` `'^'`

Las funciones ord y chr

Los caracteres se representan internamente en la computadora con valores enteros. Es decir que a cada carácter le corresponde un entero.

- `ord` - toma un carácter y devuelve el entero correspondiente.
- `chr` - toma un entero y devuelve el carácter que representa.

Ejemplos:

- `ord('A')` es 65 y `chr(65)` es 'A'
- `ord('B')` es 66 y `chr(66)` es 'B'
- `ord('0')` es 48 y `chr(48)` es '0'
- `ord('1')` es 49 y `chr(49)` es '1'

La función `ord` se puede aplicar a todos los tipos ordinales (más adelante se verá).

Ejemplos de asignaciones

```
var
```

```
  i,k : integer;
```

```
  x,y : real;
```

```
  bb  : boolean;
```

```
  car : char;
```

```
begin
```

```
  i:= 4;
```

```
  x:= 2.3;
```

```
  y:= 1;                                (* conversión implícita *)
```

```
  k:= trunc(x);                          (* conversión explícita *)
```

```
  y:= (y + sqr(x)) / 2;
```

```
  x:= i + k;
```

```
  bb:= true;
```

```
  bb:= (y < x) or (3 >= k + sqrt(i));
```

```
  car:= chr(32);                          (* carácter espacio *)
```

```
  i:= ord(car) + 1;
```

```
  ...
```