

# Programación y Diseño Algorítmico

F. Hidrobo, K. Tucci, M. Uzcátegui

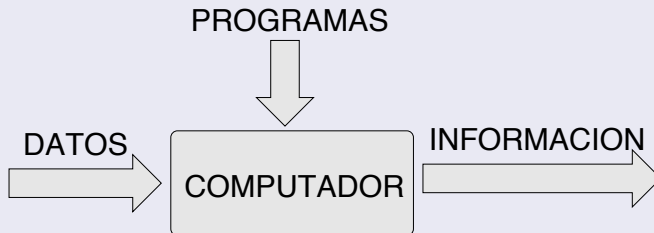
Universidad de Los Andes  
Facultad de Ciencias  
SUMA

{hidrobo,kay,maye}@ula.ve

## Objetivo

Este curso tiene como objetivo brindar los elementos necesarios para que el estudiante sea capaz de resolver adecuadamente problemas concretos haciendo uso del computador como herramienta de trabajo.

# Procesamiento de Información

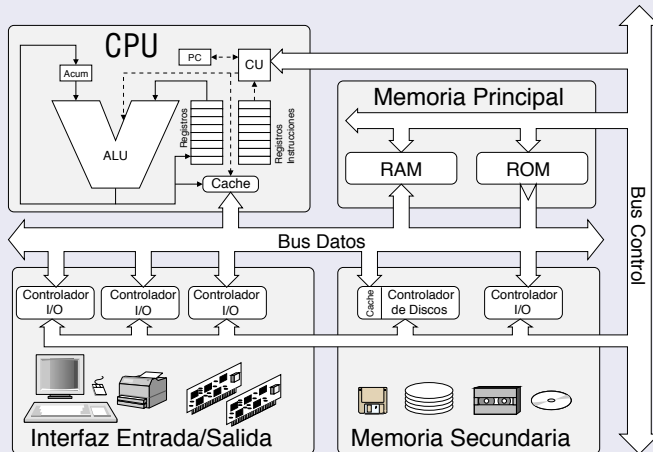


**Hardware:** Componentes físicos y dispositivos de E/S

**Programa:** Instrucciones a ser ejecutadas por un computador

**Software:** Programas escritos para un computador

# Computador



# Dispositivos E/S

Los dispositivos de E/S permiten la comunicación ente el usuario y el computador.

## Dispositivos de Entrada

- Teclado
- Ratón
- Lector de discos
- Scanner ...

## Dispositivos de Salida

- Impresora
- Monitor
- Red ...

# Memoria

## Memoria principal (primaria)

- Conjunto de celdas de memoria que se direccionan a través de un único nombre.
- Acceso directo por referencia para:  
**Carga y Recuperación de información**
- Residen los programas en ejecución

## Memoria auxiliar (secundaria)

- Permite almacenar gran cantidad de información
- Información persistente
- **Ejemplos:** Discos duros, CD Rom, DVD, ...

# Unidad Central de Procesamiento

## Unidad de Control

- Carga instrucciones en memoria
- Interpreta
- Devuelve el resultado de la ejecución

## Unidad Aritmética y Lógica

- Procesa operaciones aritméticas y lógicas
- Provee decisión a la Unidad de Control

# Solución de Problemas

Análisis  $\Rightarrow$  Diseño Algorítmico  $\Rightarrow$  Programa

**Análisis:** El problema debe ser claramente especificado y entendido.

**Diseño algorítmico:** Construcción de una solución del problema posible de ser ejecutada.

**Programa:** Traducción del algoritmo a un lenguaje de computación.

# Análisis

- Debe acotarse el campo de acción del problema
- Requiere la clara definición del problema
- Debe indicarse que va hacer el programa y cual va a ser el resultado.
- Debe detallarse las especificaciones de entrada y salida
- En la resolución de un problema complejo, se divide en varios sub problemas y seguidamente se vuelven a dividir los sub problemas en otros mas sencillos, hasta que puedan implementarse en el computador.

# Término Algoritmo

proviene del matemático persa, *Muhammad ibn Mūsā al – Khwārizmī* (780-850). Conocido como *Al – Khwārizmī*  
Escribió libros sobre geografía, astronomía y matemática.

En su obra *Artimética*, traducida al latín en el siglo 12 como

*Algoritmi de numero Indorum*

explica con detalle el funcionamiento del sistema decimal que usaban en la India y los algoritmos para calcular con este sistema de numeración.

Esta obra contribuyó a la difusión en occidente del sistema de numeración indio y al conocimiento del cero.

# Algoritmo

Se refiere a una secuencia ordenada y finita de pasos, exenta de ambigüedades, que seguidas en su orden lógico nos conduce a la solución de un problema específico.

Estar compuesto de pasos o instrucciones simples.

Seguir un orden preestablecido.

**Claridad:** Expresado en términos específicos, describir las situaciones excepcionales.

**Riguroso:** No tener ambigüedades.

**Efectivo:** Resolver el problema de forma general y en un número finito de pasos.

**Preciso:** Bajo las mismas condiciones el algoritmo siempre debe arrojar los mismos resultados.

# Algoritmo

En general las instrucciones básicas que se pueden dar en un algoritmo son:

**Solicitud:** Los datos provienen de alguna fuente externa al algoritmo

**Asignación:** Darle un valor específico a una variable

**Decisión:** Acción de seleccionar, de acuerdo a alguna condición, la siguiente instrucción

**Repetición:** Indica que una o varias instrucciones se deben efectuar más de una vez

**Generación:** proporcionar los resultados.

# Algoritmo

En general las instrucciones básicas que se pueden dar en un algoritmo son:

**Solicitud:** Los datos provienen de alguna fuente externa al algoritmo

**Asignación:** Darle un valor específico a una variable

**Decisión:** Acción de seleccionar, de acuerdo a alguna condición, la siguiente instrucción

**Repetición:** Indica que una o varias instrucciones se deben efectuar más de una vez

**Generación:** proporcionar los resultados.

# Pascal

- Lenguaje de alto nivel que usaremos en el curso.
- Fue diseñado y desarrollado por N. Wirth alrededor del 1968.
- Este lenguaje permite expresar principios de programación y de diseño de solución de problemas en forma abstracta y estructurada.
- Una instrucción Pascal equivale a varias instrucciones de lenguaje de máquina.

# Estructura

NombrePrograma.pas

```
Program NombrePrograma;
```

Declaraciones

```
Begin
```

    Secuencia de Instrucciones

```
End.
```

# Primer programa pascal

Hola mundo...

uno.pas

```
Program uno;  
Uses Crt;  
Begin  
  
    Write('Hola mundo...');  
End.
```

```
% fpc uno.pas  
% ./uno
```

# Primer programa pascal

Hola mundo...

uno.pas

```
Program uno;  
Uses Crt;  
Begin  
    clrscr;  
  
    Write('Hola mundo...');  
End.
```

```
% fpc uno.pas  
% ./uno
```

# Primer programa pascal

Hola mundo...

uno.pas

```
Program uno;  
Uses Crt;  
Begin  
  clrscr;  
  gotoxy(10,3);  
  Write('Hola mundo...');  
End.
```

```
% fpc uno.pas  
% ./uno
```

# Primer programa pascal

Hola mundo...

```
uno.pas
```

```
// Primer programa en pascal
```

```
Program uno;
```

```
Uses Crt;
```

```
Begin
```

```
  clrscr;
```

```
  gotoxy(10,3);
```

```
  Write('Hola mundo...');
```

```
End.
```

```
% fpc uno.pas
```

```
% ./uno
```

# Diseño descendiente

Técnica de diseño en la que se ataca primero el problema general, descomponiéndolo en problemas más pequeños, los cuales dependiendo de su complejidad se resolverán descomponiéndolos.

Este proceso continúa hasta que los problemas restantes son triviales.

# Representación

Los algoritmos pueden ser expresados de muchas maneras, incluyendo:

- lenguaje natural,
- **pseudocódigo**,
- diagramas de flujo,

Las descripciones en lenguaje natural tienden a ser ambiguas y extensas. El usar pseudocódigo y diagramas de flujo evita muchas ambigüedades del lenguaje natural. Dichas expresiones son formas más estructuradas para representar algoritmos.

**independientes de un lenguaje de programación específico**

# Pseudocódigo

- Es la descripción de un algoritmo que asemeja a un lenguaje de programación pero con algunas convenciones del lenguaje natural.
- Tiene varias ventajas entre las que se destaca el poco espacio que se requiere para representar instrucciones complejas.
- No está regido por ningún estándar.
- pseudo viene de falso y por ende es un código que no se aplica al proceso que debe realizar la maquina.

# Identificadores

- Los identificadores se usan para darle nombre a las cosas en el algoritmo.
- Un identificador está formado por una letra o el caracter `_` seguido o no por una secuencia de letras, dígitos y/o `_`.
- Son identificadores válidos `A345x` , `_234` , `mI_nomBrE` y `Peso_2_1` .
- No se puede incluir espacio en blanco entre las palabras en los identificadores.
- Los identificadores no pueden comenzar con un dígito.

## Palabras reservadas

Existen una serie de secuencias válidas como identificadores que no pueden ser usadas con tal fin debido a que tienen una función preestablecida.

Entre las palabras reservadas tenemos a todos identificadores predefinidos de

**constantes:** ( $\pi$ , ...),

**funciones:** (sin, cos, ln, exp, ...)

**procedimientos:** (obtener, devolver, ...).

**secuencias para estructuras de programación:** (si, sino, según, mientras, repita, desde, ...).

## Símbolos especiales

El conjunto de caracteres ASCII consta de 256 elementos algunos de los cuales por tener un significado especial en los algoritmos y en los lenguajes de programación se denominan símbolos especiales:

operadores aritméticos: + - \* / ÷ %

operadores lógicos:  $\neg$   $\wedge$   $\vee$

símbolos relacionales: = < >  $\neq$   $\geq$   $\leq$

símbolos de agrupación: { } ( ) [ ]

signos de puntuación: . , ; : ! ? "

Por Su uso está definido por la reglas sintácticas y semanticas del pseudocódigo y de los lenguajes de programación.

## Tipos de Datos

- Los datos en el computador son almacenados como secuencias binarias, es decir secuencias de ceros y unos.
- En el algoritmo y en los programas tales secuencias se interpretan asociando un tipo de dato a cada dato y a cada expresión.
- Los datos son valores de determinado tipo; se asignan a variables, se asocian a constantes o resultan de la evaluación de expresiones.
- Datos ordinales son aquellos datos que pueden ordenarse realizando una asociación uno a uno con el conjunto (o un subconjunto) de los números enteros.

# Tipos de Datos

\* : Ordinal



# Constantes

Valores asociados a un identificador que permanecen inalterados durante la solución del problema

- **Numéricas enteras**
- **Numéricas punto flotante**
- **Caracter**
- **Cadenas de caracteres**

# Variables

- Se asocian a un identificador y son utilizadas para almacenar datos que pueden ser modificados a medida que el algoritmo avanza.
- El identificador de cada variable tiene asociado un tipo de dato, el cual no debe cambiar a lo largo del algoritmo. Lo que puede cambiar es el valor contenido en la variable.
- Siempre se debe asignarle el valor a una variable antes de que esta intervenga en una expresión.
- El valor es lo que se utiliza en las expresiones una vez haya sido interpretado según el tipo correspondiente a esa variable.

# Cálculo de área de triángulo

## Planteamiento

Dado un triángulo, a partir de las dimensiones correspondientes a su altura y a la base, calcular el área del mismo.

# Cálculo de área de triangulo

Diseño

Entrada

$base, altura \in \mathcal{R}^+$

Salida

$area \in \mathcal{R}$

Pseudocódigo

Obtener( $base, altura$ )

$area \leftarrow \frac{base * altura}{2}$

Devolver( $area$ )

## Cálculo de área de triangulo

Código Pascal

areatriangulo.pas

```
Program areatriangulo;  
Uses Crt;  
Var  
    altura, base, area: Real;  
Begin  
    Read(altura,base);  
    area := altura * base/2;  
    Write(area);  
End.
```

```
% fpc areatriangulo.pas  
% ./areatriangulo
```

# Sentencias

Las instrucciones simples son aquellas que realizan una única acción y de ellas no depende directamente la ejecución o no de otras instrucciones. Pueden ser de dos tipos:

**Asignación:** La asignación es la instrucción que permite almacenar un valor en una variable, es decir, el resultado de evaluar una expresión es almacenado en el espacio de memoria reservado a una variable.

**Llamada a subprograma:** permite hacer uso de las librerías del sistema y de los módulos desarrollados por el programador.

# Asignación

Hay que destacar que el tipo de dato resultante de la evaluación de la expresión debe ser compatible con el tipo de dato de la variable, esto es, que el resultado de la expresión pueda ser almacenado e interpretado correctamente como el valor contenido en la variable.

## Pseudocódigo

```
IdVariable ← Valor  
IdVariable = Valor
```

## Código Pascal

```
IdVariable := Valor;
```

al asignarle un valor a una variable se pierde irremediablemente el valor que tenía asignado anteriormente

## Compatibilidad de Tipos

Variable	Expresión
$\mathcal{N}$	$\mathcal{N}$
$\mathcal{Z}$	$\mathcal{N}, \mathcal{Z}$
$\mathcal{R}$	$\mathcal{N}, \mathcal{Z}, \mathcal{R}$
$\mathcal{L}$	$\mathcal{L}$
$ch$	$ch$

Además de la compatibilidad, es importante garantizar, ya en el momento de la implementación del código del programa, que el resultado de la expresión no desborde la capacidad del tipo de dato de la variable

## Llamada a subprogramas

En caso de que exista un tipo de dato arrojado al ejecutar un subprograma este es determinado directamente por él.  
Los subprogramas básicos de entrada salida de datos son:

### Pseudocódigo

*Obtener*(lista de variables)    *Leer*(lista de variables)

---

*Devolver*(lista de valores)    *Escribir*(lista de valores)

### Código Pascal

*Read*(lista de variables);    *Readln*(lista de variables);

---

*Write*(lista de valores);    *Writeln*(lista de valores);

# Expresiones

- Las expresiones son estructuras utilizadas en los algoritmos para calcular nuevos valores.
- Su estructura está compuesta por uno o más operandos separados por operadores.
- Operandos y operadores pueden ser agrupados para darle claridad a una expresión o para forzar un orden específico de evaluación.
- Los operandos están conformados por expresiones (constantes, variables, funciones, o una combinación de ellos)

## Clasificación de los Operadores

- Pueden ser unarios y binarios según el número de operandos sobre los que actúan.
- De acuerdo al tipo de datos sobre los cuales operan tenemos:
  - Aritméticos:** que operan sobre valores numéricos;
  - Relacionales:** que comparan dos valores;
  - Lógicos:** que actúan sobre valores booleanos.

# Operadores

	Operación	Operador	A	B	Operación
<b>Asignación:</b>	Asignación	$\leftarrow A$	$:=$	$\mathcal{Z}$	$\mathcal{Z}, \mathcal{R}$
		$= A$		$\mathcal{R}$	$\mathcal{R}$
				$ch$	$ch, \mathcal{C}$
				$\mathcal{C}$	$\mathcal{C}$
				$\mathcal{L}$	$\mathcal{L}$

**Lógicos:**

Conjunción	$A \wedge B$	AND	$\mathcal{L}$	$\mathcal{L}$	$\mathcal{L}$
Disjunción	$A \vee B$	OR	$\mathcal{L}$	$\mathcal{L}$	$\mathcal{L}$
Negación	$\neg A$	NOT	$\mathcal{L}$		$\mathcal{L}$

## Aritméticos:

Suma	$A + B$	+	$\mathbb{Z}$ $\mathbb{Z}$ $\mathcal{R}$ $\mathcal{R}$	$\mathbb{Z}$ $\mathcal{R}$ $\mathbb{Z}$ $\mathcal{R}$	$\mathbb{Z}$ $\mathcal{R}$ $\mathcal{R}$ $\mathcal{R}$
Resta	$A - B$	-	$\mathbb{Z}$ $\mathbb{Z}$ $\mathcal{R}$ $\mathcal{R}$	$\mathbb{Z}$ $\mathcal{R}$ $\mathbb{Z}$ $\mathcal{R}$	$\mathbb{Z}$ $\mathcal{R}$ $\mathcal{R}$ $\mathcal{R}$
Multiplicación	$A * B$	*	$\mathbb{Z}$ $\mathbb{Z}$ $\mathcal{R}$ $\mathcal{R}$	$\mathbb{Z}$ $\mathcal{R}$ $\mathbb{Z}$ $\mathcal{R}$	$\mathbb{Z}$ $\mathcal{R}$ $\mathcal{R}$ $\mathcal{R}$
División	$A/B$	/	$\mathbb{Z}, \mathcal{R}$	$\mathbb{Z}, \mathcal{R}$	$\mathcal{R}$
Cociente	$A \div B$	DIV	$\mathbb{Z}$	$\mathbb{Z}$	$\mathbb{Z}$
Residuo	$A \% B$	MOD	$\mathbb{Z}$	$\mathbb{Z}$	$\mathbb{Z}$

## Relacionales:

Igualdad	$A = B$	=	$\mathbb{Z}, \mathbb{R}$ $ch, \mathbb{C}$ $\mathcal{L}$	$\mathbb{Z}, \mathbb{R}$ $ch, \mathbb{C}$ $\mathcal{L}$	$\mathcal{L}$ $\mathcal{L}$ $\mathcal{L}$
Desigualdad	$A \neq B$	<>	$\mathbb{Z}, \mathbb{R}$ $ch, \mathbb{C}$ $\mathcal{L}$	$\mathbb{Z}, \mathbb{R}$ $ch, \mathbb{C}$ $\mathcal{L}$	$\mathcal{L}$ $\mathcal{L}$ $\mathcal{L}$
Mayor	$A > B$	>	$\mathbb{Z}, \mathbb{R}$ $ch, \mathbb{C}$ $\mathcal{L}$	$\mathbb{Z}, \mathbb{R}$ $ch, \mathbb{C}$ $\mathcal{L}$	$\mathcal{L}$ $\mathcal{L}$ $\mathcal{L}$
Mayor o igual	$A \geq B$	>=	$\mathbb{Z}, \mathbb{R}$ $ch, \mathbb{C}$ $\mathcal{L}$	$\mathbb{Z}, \mathbb{R}$ $ch, \mathbb{C}$ $\mathcal{L}$	$\mathcal{L}$ $\mathcal{L}$ $\mathcal{L}$
Menor	$A < B$	<	$\mathbb{Z}, \mathbb{R}$ $ch, \mathbb{C}$ $\mathcal{L}$	$\mathbb{Z}, \mathbb{R}$ $ch, \mathbb{C}$ $\mathcal{L}$	$\mathcal{L}$ $\mathcal{L}$ $\mathcal{L}$
Menor o igual	$A \leq B$	<=	$\mathbb{Z}, \mathbb{R}$ $ch, \mathbb{C}$ $\mathcal{L}$	$\mathbb{Z}, \mathbb{R}$ $ch, \mathbb{C}$ $\mathcal{L}$	$\mathcal{L}$ $\mathcal{L}$ $\mathcal{L}$

# Funciones Básicas

Descripción	$f(x)$	$x$	$f(x)$
-------------	--------	-----	--------

## Potencias:

Cuadrática	$x^2$	SQR(x)	$\mathcal{R}$	$\mathcal{R}_+$
Radical	$\sqrt{x}$	SQRT(x)	$\mathcal{R}_+$	$\mathcal{R}_+$

## Trigonométricas:

Seno	$sen(x)$	SIN(x)	$\mathcal{R}$	$\mathcal{R}$
Coseno	$cos(x)$	COS(x)	$\mathcal{R}$	$\mathcal{R}$
Arco tangente	$tan^{-1}(x)$	ARCTAN(x)	$\mathcal{R}$	$\mathcal{R}$

## Exponenciales y Logarítmicas:

Exponencial	$e^x$	EXP(x)	$\mathcal{R}$	$\mathcal{R}_\oplus$
Logaritmo	$ln(x)$	LN(x)	$\mathcal{R}_\oplus$	$\mathcal{R}_+$

## Ordenamiento:

Siguiete	$sig(x)$	SUCC(x)	$\mathcal{Z}$	$\mathcal{Z}$
			$ch$	$ch$
			$\mathcal{L}$	$\mathcal{L}$
Anterior	$ant(x)$	PRED(x)	$\mathcal{Z}$	$\mathcal{Z}$

# Funciones Básicas

## Conversión:

Truncamiento	$\text{trunc}(x)$	TRUNC(x)	$\mathcal{Z}, \mathcal{R}$	$\mathcal{Z}$
Redondeo	$\text{redon}(x)$	ROUND(x)	$\mathcal{Z}, \mathcal{R}$	$\mathcal{Z}$
Ordinal	$\text{ord}(x)$	ORD(x)	$\mathcal{Z}, ch, \mathcal{L}$	$\mathcal{Z}$
Caracter	$\text{chr}(x)$	CHR(x)	$\mathcal{Z}$	ch
Impar	$\text{impar}(x)$	ODD(x)	$\mathcal{Z}$	$\mathcal{L}$

## Otras:

Valor Absoluto	$ x $	ABS(x)	$\mathcal{Z}$ $\mathcal{R}$	$\mathcal{Z}$ $\mathcal{R}$
Parte Entera	$\llbracket x \rrbracket$	INT(x)	$\mathcal{Z}, \mathcal{R}$	$\mathcal{R}$
Parte Decimal	$\text{frac}(x)$	FRAC(x)	$\mathcal{Z}, \mathcal{R}$	$\mathcal{R}$
Aleatorio	$\text{aleatorio}(x)$	RANDOM(x)	$\mathcal{Z}$	$\mathcal{Z}$

## Evaluación de Expresiones

$$2 - 6/2 * 3 - 1$$

¿Al evaluar la expresión anterior cuál es el resultado correcto?  
-7, -1, -8.

El problema es la ambigüedad en la forma de evaluar las expresiones en la vida cotidiana. En los algoritmos la evaluación de expresiones se efectúa según una reglas, llamadas **precedencia de operadores**, que asignan prioridades relativas.

## Precedencia de Operadores

1	Paréntesis	()
2	Operadores Unarios y Funciones	$\neg$ sin cos ln exp ...
3	Multiplicativos	* / ÷ % ^
4	Aditivos	+ - √
5	Relacionales	= ≠ < > ≤ ≥

El resultado de la evaluación de una expresión debe ser alojado en memoria mediante una instrucción de asignación o utilizando los subprogramas para el manejo de las operaciones de entrada y salida.

## Ejercicios

Sean  $A \leftarrow 4$ ,  $B \leftarrow 4$ ,  $C \leftarrow 2$ ,  $D \leftarrow 3$ ,  $E \leftarrow 2$   
Calcular el valor de  $X$  en cada caso

$$X \leftarrow A/B - C + D * E - A - C$$

$$X \leftarrow A/(B - C) + D * (E - A) - C$$

$$X \leftarrow A/B - C > D * E - A - C$$

## Ejercicio 1

$A \leftarrow 4, \quad B \leftarrow 4, \quad C \leftarrow 2, \quad D \leftarrow 3, \quad E \leftarrow 2$

$X \leftarrow A/B - C + D * E - A - C$

## Ejercicio 1

$A \leftarrow 4, \quad B \leftarrow 4, \quad C \leftarrow 2, \quad D \leftarrow 3, \quad E \leftarrow 2$

$X \leftarrow A/B - C + D * E - A - C$

$X \leftarrow 4/4 - 2 + 3 * 2 - 4 - 2$

## Ejercicio 1

$A \leftarrow 4, \quad B \leftarrow 4, \quad C \leftarrow 2, \quad D \leftarrow 3, \quad E \leftarrow 2$

$X \leftarrow A/B - C + D * E - A - C$

$X \leftarrow 4/4 - 2 + 3 * 2 - 4 - 2$

$X \leftarrow 1 - 2 + 3 * 2 - 4 - 2$

## Ejercicio 1

$A \leftarrow 4, B \leftarrow 4, C \leftarrow 2, D \leftarrow 3, E \leftarrow 2$

$$X \leftarrow A/B - C + D * E - A - C$$

$$X \leftarrow 4/4 - 2 + 3 * 2 - 4 - 2$$

$$X \leftarrow 1 - 2 + 3 * 2 - 4 - 2$$

$$X \leftarrow 1 - 2 + 6 - 4 - 2$$

## Ejercicio 1

$A \leftarrow 4, \quad B \leftarrow 4, \quad C \leftarrow 2, \quad D \leftarrow 3, \quad E \leftarrow 2$

$X \leftarrow A/B - C + D * E - A - C$

$X \leftarrow 4/4 - 2 + 3 * 2 - 4 - 2$

$X \leftarrow 1 - 2 + 3 * 2 - 4 - 2$

$X \leftarrow 1 - 2 + 6 - 4 - 2$

$X \leftarrow -1 + 6 - 4 - 2$

## Ejercicio 1

$A \leftarrow 4, \quad B \leftarrow 4, \quad C \leftarrow 2, \quad D \leftarrow 3, \quad E \leftarrow 2$

$$X \leftarrow A/B - C + D * E - A - C$$

$$X \leftarrow 4/4 - 2 + 3 * 2 - 4 - 2$$

$$X \leftarrow 1 - 2 + 3 * 2 - 4 - 2$$

$$X \leftarrow 1 - 2 + 6 - 4 - 2$$

$$X \leftarrow -1 + 6 - 4 - 2$$

$$X \leftarrow 5 - 4 - 2$$

## Ejercicio 1

$A \leftarrow 4, \quad B \leftarrow 4, \quad C \leftarrow 2, \quad D \leftarrow 3, \quad E \leftarrow 2$

$X \leftarrow A/B - C + D * E - A - C$

$X \leftarrow 4/4 - 2 + 3 * 2 - 4 - 2$

$X \leftarrow 1 - 2 + 3 * 2 - 4 - 2$

$X \leftarrow 1 - 2 + 6 - 4 - 2$

$X \leftarrow -1 + 6 - 4 - 2$

$X \leftarrow 5 - 4 - 2$

$X \leftarrow 1 - 2$

## Ejercicio 1

$A \leftarrow 4, B \leftarrow 4, C \leftarrow 2, D \leftarrow 3, E \leftarrow 2$

$X \leftarrow A/B - C + D * E - A - C$

$X \leftarrow 4/4 - 2 + 3 * 2 - 4 - 2$

$X \leftarrow 1 - 2 + 3 * 2 - 4 - 2$

$X \leftarrow 1 - 2 + 6 - 4 - 2$

$X \leftarrow -1 + 6 - 4 - 2$

$X \leftarrow 5 - 4 - 2$

$X \leftarrow 1 - 2$

$X \leftarrow -1$

## Ejercicio 1

$A \leftarrow 4, B \leftarrow 4, C \leftarrow 2, D \leftarrow 3, E \leftarrow 2$

$X \leftarrow A/B - C + D * E - A - C$

$X \leftarrow 4/4 - 2 + 3 * 2 - 4 - 2$

$X \leftarrow 1 - 2 + 3 * 2 - 4 - 2$

$X \leftarrow 1 - 2 + 6 - 4 - 2$

$X \leftarrow -1 + 6 - 4 - 2$

$X \leftarrow 5 - 4 - 2$

$X \leftarrow 1 - 2$

$X \leftarrow -1$

$X \in \mathcal{R}$

## Ejercicio 2

$A \leftarrow 4, \quad B \leftarrow 4, \quad C \leftarrow 2, \quad D \leftarrow 3, \quad E \leftarrow 2$

$X \leftarrow A/(B - C) + D * (E - A) - C$

## Ejercicio 2

$A \leftarrow 4, \quad B \leftarrow 4, \quad C \leftarrow 2, \quad D \leftarrow 3, \quad E \leftarrow 2$

$$X \leftarrow A / (B - C) + D * (E - A) - C$$

$$X \leftarrow 4 / (4 - 2) + 3 * (2 - 4) - 2$$

## Ejercicio 2

$A \leftarrow 4, \quad B \leftarrow 4, \quad C \leftarrow 2, \quad D \leftarrow 3, \quad E \leftarrow 2$

$$X \leftarrow A / (B - C) + D * (E - A) - C$$

$$X \leftarrow 4 / (4 - 2) + 3 * (2 - 4) - 2$$

$$X \leftarrow 4 / 2 + 3 * (2 - 4) - 2$$

## Ejercicio 2

$A \leftarrow 4, \quad B \leftarrow 4, \quad C \leftarrow 2, \quad D \leftarrow 3, \quad E \leftarrow 2$

$$X \leftarrow A / (B - C) + D * (E - A) - C$$

$$X \leftarrow 4 / (4 - 2) + 3 * (2 - 4) - 2$$

$$X \leftarrow 4 / 2 + 3 * (2 - 4) - 2$$

$$X \leftarrow 4 / 2 + 3 * -2 - 2$$

## Ejercicio 2

$A \leftarrow 4, \quad B \leftarrow 4, \quad C \leftarrow 2, \quad D \leftarrow 3, \quad E \leftarrow 2$

$$X \leftarrow A / (B - C) + D * (E - A) - C$$

$$X \leftarrow 4 / (4 - 2) + 3 * (2 - 4) - 2$$

$$X \leftarrow 4 / 2 + 3 * (2 - 4) - 2$$

$$X \leftarrow 4 / 2 + 3 * -2 - 2$$

$$X \leftarrow 2 + 3 * -2 - 2$$

## Ejercicio 2

$A \leftarrow 4, \quad B \leftarrow 4, \quad C \leftarrow 2, \quad D \leftarrow 3, \quad E \leftarrow 2$

$$X \leftarrow A / (B - C) + D * (E - A) - C$$

$$X \leftarrow 4 / (4 - 2) + 3 * (2 - 4) - 2$$

$$X \leftarrow 4 / 2 + 3 * (2 - 4) - 2$$

$$X \leftarrow 4 / 2 + 3 * -2 - 2$$

$$X \leftarrow 2 + 3 * -2 - 2$$

$$X \leftarrow 2 - 6 - 2$$

## Ejercicio 2

$A \leftarrow 4, \quad B \leftarrow 4, \quad C \leftarrow 2, \quad D \leftarrow 3, \quad E \leftarrow 2$

$$X \leftarrow A / (B - C) + D * (E - A) - C$$

$$X \leftarrow 4 / (4 - 2) + 3 * (2 - 4) - 2$$

$$X \leftarrow 4 / 2 + 3 * (2 - 4) - 2$$

$$X \leftarrow 4 / 2 + 3 * -2 - 2$$

$$X \leftarrow 2 + 3 * -2 - 2$$

$$X \leftarrow 2 - 6 - 2$$

$$X \leftarrow -4 - 2$$

## Ejercicio 2

$A \leftarrow 4, \quad B \leftarrow 4, \quad C \leftarrow 2, \quad D \leftarrow 3, \quad E \leftarrow 2$

$$X \leftarrow A / (B - C) + D * (E - A) - C$$

$$X \leftarrow 4 / (4 - 2) + 3 * (2 - 4) - 2$$

$$X \leftarrow 4 / 2 + 3 * (2 - 4) - 2$$

$$X \leftarrow 4 / 2 + 3 * -2 - 2$$

$$X \leftarrow 2 + 3 * -2 - 2$$

$$X \leftarrow 2 - 6 - 2$$

$$X \leftarrow -4 - 2$$

$$X \leftarrow 2$$

## Ejercicio 2

$A \leftarrow 4, \quad B \leftarrow 4, \quad C \leftarrow 2, \quad D \leftarrow 3, \quad E \leftarrow 2$

$$X \leftarrow A / (B - C) + D * (E - A) - C$$

$$X \leftarrow 4 / (4 - 2) + 3 * (2 - 4) - 2$$

$$X \leftarrow 4 / 2 + 3 * (2 - 4) - 2$$

$$X \leftarrow 4 / 2 + 3 * -2 - 2$$

$$X \leftarrow 2 + 3 * -2 - 2$$

$$X \leftarrow 2 - 6 - 2$$

$$X \leftarrow -4 - 2$$

$$X \leftarrow 2$$

$X \in \mathcal{R}$

## Ejercicio 3

$A \leftarrow 4, B \leftarrow 4, C \leftarrow 2, D \leftarrow 3, E \leftarrow 2$

$X \leftarrow A/B - C > D * E - A - C$

## Ejercicio 3

$A \leftarrow 4, \quad B \leftarrow 4, \quad C \leftarrow 2, \quad D \leftarrow 3, \quad E \leftarrow 2$

$X \leftarrow A/B - C > D * E - A - C$

$X \leftarrow 4/4 - 2 > 3 * 2 - 4 - 2$

## Ejercicio 3

$A \leftarrow 4, B \leftarrow 4, C \leftarrow 2, D \leftarrow 3, E \leftarrow 2$

$X \leftarrow A/B - C > D * E - A - C$

$X \leftarrow 4/4 - 2 > 3 * 2 - 4 - 2$

$X \leftarrow 1 - 2 > 3 * 2 - 4 - 2$

## Ejercicio 3

$A \leftarrow 4, B \leftarrow 4, C \leftarrow 2, D \leftarrow 3, E \leftarrow 2$

$X \leftarrow A/B - C > D * E - A - C$

$X \leftarrow 4/4 - 2 > 3 * 2 - 4 - 2$

$X \leftarrow 1 - 2 > 3 * 2 - 4 - 2$

$X \leftarrow 1 - 2 > 6 - 4 - 2$

## Ejercicio 3

$A \leftarrow 4, \quad B \leftarrow 4, \quad C \leftarrow 2, \quad D \leftarrow 3, \quad E \leftarrow 2$

$X \leftarrow A/B - C > D * E - A - C$

$X \leftarrow 4/4 - 2 > 3 * 2 - 4 - 2$

$X \leftarrow 1 - 2 > 3 * 2 - 4 - 2$

$X \leftarrow 1 - 2 > 6 - 4 - 2$

$X \leftarrow -1 > 6 - 4 - 2$

## Ejercicio 3

$A \leftarrow 4, \quad B \leftarrow 4, \quad C \leftarrow 2, \quad D \leftarrow 3, \quad E \leftarrow 2$

$X \leftarrow A/B - C > D * E - A - C$

$X \leftarrow 4/4 - 2 > 3 * 2 - 4 - 2$

$X \leftarrow 1 - 2 > 3 * 2 - 4 - 2$

$X \leftarrow 1 - 2 > 6 - 4 - 2$

$X \leftarrow -1 > 6 - 4 - 2$

$X \leftarrow -1 > 2 - 2$

## Ejercicio 3

$A \leftarrow 4, \quad B \leftarrow 4, \quad C \leftarrow 2, \quad D \leftarrow 3, \quad E \leftarrow 2$

$X \leftarrow A/B - C > D * E - A - C$

$X \leftarrow 4/4 - 2 > 3 * 2 - 4 - 2$

$X \leftarrow 1 - 2 > 3 * 2 - 4 - 2$

$X \leftarrow 1 - 2 > 6 - 4 - 2$

$X \leftarrow -1 > 6 - 4 - 2$

$X \leftarrow -1 > 2 - 2$

$X \leftarrow -1 > 0$

## Ejercicio 3

$A \leftarrow 4, \quad B \leftarrow 4, \quad C \leftarrow 2, \quad D \leftarrow 3, \quad E \leftarrow 2$

$X \leftarrow A/B - C > D * E - A - C$

$X \leftarrow 4/4 - 2 > 3 * 2 - 4 - 2$

$X \leftarrow 1 - 2 > 3 * 2 - 4 - 2$

$X \leftarrow 1 - 2 > 6 - 4 - 2$

$X \leftarrow -1 > 6 - 4 - 2$

$X \leftarrow -1 > 2 - 2$

$X \leftarrow -1 > 0$

$X \leftarrow F$

## Ejercicio 3

$A \leftarrow 4, B \leftarrow 4, C \leftarrow 2, D \leftarrow 3, E \leftarrow 2$

$X \leftarrow A/B - C > D * E - A - C$

$X \leftarrow 4/4 - 2 > 3 * 2 - 4 - 2$

$X \leftarrow 1 - 2 > 3 * 2 - 4 - 2$

$X \leftarrow 1 - 2 > 6 - 4 - 2$

$X \leftarrow -1 > 6 - 4 - 2$

$X \leftarrow -1 > 2 - 2$

$X \leftarrow -1 > 0$

$X \leftarrow F$

$X \in \mathcal{L}$

## Incremento y Decremento

Muchas veces el valor a una variable está relacionado con el valor que dicha variable tiene en el momento en el que se realiza la asignación.

Por ejemplo, transcurrido un año, hay que incrementar en uno la edad de cierta persona.

```
edad ← edad+1
```

Note que, a una variable sólo se le puede asignar valores. Como primero se evalúa la expresión y luego se procede a asignar a la variable `edad` el resultado de la evaluación, el valor de la `edad` con esta instrucción se incrementa en uno.

# Intercambio

Cuando queremos asignar el valor que tiene una variable en otra y viceversa

<b>caso 1</b>	<b>caso 2</b>	<b>caso 3</b>
$x \leftarrow y$	$y \leftarrow x$	$\text{aux} \leftarrow x$
$y \leftarrow x$	$x \leftarrow y$	$x \leftarrow y$
?	?	$y \leftarrow \text{aux}$

Este es un problema equivalente al de intercambiar el contenido de dos botellas y para resolverlo necesitamos una tercera botella en la que se almacena temporalmente el contenido de una de las dos botellas a intercambiar.

## Identificación de Variables

Tome  $a$  como la inicial de su apellido,  $b$  y  $c$ , como los dos últimos dígitos de su cédula.

Identifique los tipos posibles de cada una de las variables y determine el valor de cada una en cada paso del algoritmo.

### Algoritmo

```
obtener(a,b,c)
d ← a
b ← ord((sqr(b)-100*(c+1))>0)* 5
e ← b % ord(a)
f ← ch(ord(a)+1) = d
c ← e/b
```

## Identificación de Variables

Tome  $a$  como la inicial de su apellido,  $b$  y  $c$ , como los dos últimos dígitos de su cédula.

Identifique los tipos posibles de cada una de las variables y determine el valor de cada una en cada paso del algoritmo.

### Algoritmo

```
obtener(a,b,c)
d ← a
b ← ord((sqr(b)-100*(c+1))>0)* 5
e ← b % ord(a)
f ← ch(ord(a)+1) = d
c ← e/b
```

$a$	$b$	$c$	$d$	$e$	$f$
'U'	1	1			

## Identificación de Variables

Tome  $a$  como la inicial de su apellido,  $b$  y  $c$ , como los dos últimos dígitos de su cédula.

Identifique los tipos posibles de cada una de las variables y determine el valor de cada una en cada paso del algoritmo.

### Algoritmo

```
obtener(a,b,c)
d ← a
b ← ord((sqr(b)-100*(c+1))>0)* 5
e ← b % ord(a)
f ← ch(ord(a)+1) = d
c ← e/b
```

a	b	c	d	e	f
'U'	1	1			
'U'	1	1	'U'		

## Identificación de Variables

Tome  $a$  como la inicial de su apellido,  $b$  y  $c$ , como los dos últimos dígitos de su cédula.

Identifique los tipos posibles de cada una de las variables y determine el valor de cada una en cada paso del algoritmo.

### Algoritmo

```
obtener(a,b,c)
d ← a
b ← ord((sqr(b)-100*(c+1))>0)* 5
e ← b % ord(a)
f ← ch(ord(a)+1) = d
c ← e/b
```

a	b	c	d	e	f
'U'	1	1			
'U'	1	1	'U'		
'U'	0	1	'U'		

## Identificación de Variables

Tome  $a$  como la inicial de su apellido,  $b$  y  $c$ , como los dos últimos dígitos de su cédula.

Identifique los tipos posibles de cada una de las variables y determine el valor de cada una en cada paso del algoritmo.

### Algoritmo

```
obtener(a,b,c)
d ← a
b ← ord((sqr(b)-100*(c+1))>0)* 5
e ← b % ord(a)
f ← ch(ord(a)+1) = d
c ← e/b
```

a	b	c	d	e	f
'U'	1	1			
'U'	1	1	'U'		
'U'	0	1	'U'		
'U'	0	1	'U'	0	

## Identificación de Variables

Tome  $a$  como la inicial de su apellido,  $b$  y  $c$ , como los dos últimos dígitos de su cédula.

Identifique los tipos posibles de cada una de las variables y determine el valor de cada una en cada paso del algoritmo.

### Algoritmo

```
obtener(a,b,c)
d ← a
b ← ord((sqr(b)-100*(c+1))>0)* 5
e ← b % ord(a)
f ← ch(ord(a)+1) = d
c ← e/b
```

a	b	c	d	e	f
'U'	1	1			
'U'	1	1	'U'		
'U'	0	1	'U'		
'U'	0	1	'U'	0	
'U'	0	1	'U'	0	F

## Identificación de Variables

Tome a como la inicial de su apellido, b y c, como los dos últimos dígitos de su cédula.

Identifique los tipos posibles de cada una de las variables y determine el valor de cada una en cada paso del algoritmo.

### Algoritmo

```
obtener(a,b,c)
d ← a
b ← ord((sqr(b)-100*(c+1))>0)* 5
e ← b % ord(a)
f ← ch(ord(a)+1) = d
c ← e/b
```

a	b	c	d	e	f
'U'	1	1			
'U'	1	1	'U'		
'U'	0	1	'U'		
'U'	0	1	'U'	0	
'U'	0	1	'U'	0	F
'U'	error	0	'U'	0	F

## Identificación de Variables

Tome  $a$  como la inicial de su apellido,  $b$  y  $c$ , como los dos últimos dígitos de su cédula.

Identifique los tipos posibles de cada una de las variables y determine el valor de cada una en cada paso del algoritmo.

### Algoritmo

```

obtener(a,b,c)
d ← a
b ← ord((sqr(b)-100*(c+1))>0)* 5
e ← b % ord(a)
f ← ch(ord(a)+1) = d
c ← e/b
    
```

a	b	c	d	e	f
'U'	1	1			
'U'	1	1	'U'		
'U'	0	1	'U'		
'U'	0	1	'U'	0	
'U'	0	1	'U'	0	F
'U'	error	0	'U'	0	F

$a, d \in \text{ch}$      $b, c, e \in \mathbb{Z}$      $f \in \mathcal{L}$

# Potenciación

Dados dos números  $x, y$  calcular  $x^y$

$$5^2 = 5 \times 5 \quad 5^{-2} = \frac{1}{5^2} = \frac{1}{5 \times 5} \quad 5,25^{2,45}$$

$$-2^3 = -2 \times -2 \times -2 \quad -3^{-2} = \frac{1}{-3^2} = \frac{1}{-3 \times -3} \quad -3,124^{-2,12}$$

## Diseño

Entrada:

Salida:

Proceso:

Algoritmo:

## Potenciación

Dados dos números  $x, y$  calcular  $x^y$

$$5^2 = 5 \times 5 \quad 5^{-2} = \frac{1}{5^2} = \frac{1}{5 \times 5} \quad 5,25^{2,45}$$

$$-2^3 = -2 \times -2 \times -2 \quad -3^{-2} = \frac{1}{-3^2} = \frac{1}{-3 \times -3} \quad -3,124^{-2,12}$$

### Diseño

**Entrada:**  $x, y \in \mathcal{R}$

**Salida:**  $z \in \mathcal{R}$

**Proceso:**

**Algoritmo:**

## Potenciación

Dados dos números  $x, y$  calcular  $x^y$

$$5^2 = 5 \times 5 \quad 5^{-2} = \frac{1}{5^2} = \frac{1}{5 \times 5} \quad 5,25^{2,45}$$

$$-2^3 = -2 \times -2 \times -2 \quad -3^{-2} = \frac{1}{-3^2} = \frac{1}{-3 \times -3} \quad -3,124^{-2,12}$$

### Diseño

**Entrada:**  $x, y \in \mathcal{R}, x > 0$

**Salida:**  $z \in \mathcal{R}$

**Proceso:**

$$z = e^{y \ln(x)}, x > 0$$

**Algoritmo:**

# Potenciación

Dados dos números  $x, y$  calcular  $x^y$

$$5^2 = 5 \times 5 \quad 5^{-2} = \frac{1}{5^2} = \frac{1}{5 \times 5} \quad 5,25^{2,45}$$

$$-2^3 = -2 \times -2 \times -2 \quad -3^{-2} = \frac{1}{-3^2} = \frac{1}{-3 \times -3} \quad -3,124^{-2,12}$$

## Diseño

**Entrada:**  $x, y \in \mathcal{R}, x > 0$

**Salida:**  $z \in \mathcal{R}$

**Proceso:**

$$z = e^{y \ln(x)}, x > 0$$

**Algoritmo:**

*Obtener*( $x, y$ )

$z \leftarrow e^{y * \ln(x)}$

*Devolver*( $z$ )

# Potenciación

## Programa

```
Program potencia;  
Uses Crt;  
Var  
    x,y,z : Real;  
Begin  
    Readln(x,y);  
    z:=exp(y*ln(x));  
    writeln(z);  
End.
```

# Ecuación de Segundo Grado

## Planteamiento

Dada la ecuación

$$ax^2 + bx + c = 0$$

Encuentre la solución a dicha ecuación para valores de  $a, b, c \in \mathcal{R}$

Solución

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}, \quad a \neq 0$$

Donde realmente estamos indicando con el signo  $\pm$  dos posibles valores para la solución, por lo que tenemos que calcular  $x_1$  para el caso positivo y  $x_2$  para el caso negativo

# Ecuación de Segundo Grado

## Análisis

No nos informan nada respecto al tipo de dato de la solución por ello debemos asumir el caso más general: la cantidad subradical puede ser negativa por lo que el resultado pertenece al conjunto de los números complejos

$$d = b^2 - 4ac, \quad x = \begin{cases} \frac{-b}{2a} \pm \frac{\sqrt{d}}{2a}, & d \geq 0 \\ \frac{-b}{2a} \pm \frac{\sqrt{d}}{2a} i, & \text{oc} \end{cases}$$

Donde realmente estamos encontrando dos valores reales para cada solución, correspondientes a la parte real y a la parte imaginaria respectivamente

$$\begin{aligned}d &= b^2 - 4ac, \\v_1 &= -\frac{-b}{2a}, \\v_2 &= \frac{\sqrt{|d|}}{2a}\end{aligned} \quad x = \begin{cases} v_1 \pm v_2, & d \geq 0 \\ v_1 \pm v_2 i, & \text{oc} \end{cases}$$

$$d \geq 0 \begin{cases} x_1 = v_1 + v_2 \\ x_2 = v_1 - v_2 \end{cases}$$

$$d < 0 \begin{cases} x_1 = v_1 + v_2 i \\ x_2 = v_1 - v_2 i \end{cases}$$

Para el caso ( $d < 0$ ) debemos proporcionar la parte entera y la parte real de cada una de las soluciones

# Ecuación de Segundo Grado

## Diseño

**Entrada:**  $a, b, c \in \mathcal{R}, \quad a \neq 0$

**Salida:**

**Algoritmo:**

**Variables:**

# Ecuación de Segundo Grado

## Diseño

**Entrada:**  $a, b, c \in \mathcal{R}, \quad a \neq 0$

**Salida:**  $v1, v2 \in \mathcal{R}$

**Algoritmo:**

**Variables:**

# Ecuación de Segundo Grado

## Diseño

**Entrada:**  $a, b, c \in \mathcal{R}, \quad a \neq 0$

**Salida:**  $v1, v2 \in \mathcal{R}$

**Algoritmo:**

Obtener( $a, b, c$ )

**Variables:**

# Ecuación de Segundo Grado

## Diseño

**Entrada:**  $a, b, c \in \mathcal{R}, \quad a \neq 0$

**Salida:**  $v1, v2 \in \mathcal{R}$

**Algoritmo:**

Obtener( $a, b, c$ )

$d \leftarrow b * b - 4 * a * c$

**Variables:**

# Ecuación de Segundo Grado

## Diseño

**Entrada:**  $a, b, c \in \mathcal{R}, \quad a \neq 0$

**Salida:**  $v1, v2 \in \mathcal{R}$

**Algoritmo:**

Obtener( $a, b, c$ )

$d \leftarrow b * b - 4 * a * c$

$v1 \leftarrow -b / (2 * a)$

$v2 \leftarrow \sqrt{|d|} / (2 * a)$

**Variables:**

# Ecuación de Segundo Grado

## Diseño

**Entrada:**  $a, b, c \in \mathcal{R}, \quad a \neq 0$

**Salida:**  $v1, v2 \in \mathcal{R}$

**Algoritmo:**

Obtener( $a, b, c$ )

$d \leftarrow b * b - 4 * a * c$

$v1 \leftarrow -b / (2 * a)$

$v2 \leftarrow \sqrt{|d|} / (2 * a)$

Devolver( $v1 + \text{ord}(d \geq 0) * v2, \text{ord}(d < 0) * v2,$   
 $v1 - \text{ord}(d \geq 0) * v2, -\text{ord}(d < 0) * v2$ )

**Variables:**

# Ecuación de Segundo Grado

## Diseño

**Entrada:**  $a, b, c \in \mathcal{R}, \quad a \neq 0$

**Salida:**  $v1, v2 \in \mathcal{R}$

**Algoritmo:**

Obtener( $a, b, c$ )

$d \leftarrow b * b - 4 * a * c$

$v1 \leftarrow -b / (2 * a)$

$v2 \leftarrow \sqrt{|d|} / (2 * a)$

Devolver( $v1 + \text{ord}(d \geq 0) * v2, \text{ord}(d < 0) * v2,$   
 $v1 - \text{ord}(d \geq 0) * v2, -\text{ord}(d < 0) * v2$ )

**Variables:**  $d \in \mathcal{R}$

## Ecuación de Segundo Grado

ec2grado.pas

```
Program ec2grado;  
Uses Crt;  
Var  
    a,b,c,d,v1,v2: Real;  
Begin  
    Read(a,b,c);  
    d:= b*b - 4 *a * c;  
    v1:= -b/(2*a);  
    v2:= sqrt(abs(d))/(2*a);  
    Write(v1 + ord(d>=0)*v2, ord(d<0)*v2,  
          v1 - ord(d>=0)*v2, -ord(d<0)*v2);  
End.
```

## Selección simple

### Pseudocódigo

*Si Condición Entonces*  
*Bloque de Instrucciones*

### Código Pascal

*If Condición Then*  
*Instrucción;*

*If Condición Then*  
*Begin*  
*Instrucción 1;*  
*...*  
*End;*

# Selección simple

## Pseudocódigo

*Si* Condición *Entonces*  
    Bloque de Instrucciones 1  
*Sino*  
    Bloque de Instrucciones 2

## Código Pascal

*If* Condición *Then*  
    Bloque de Instrucciones 1  
*Else*  
    Bloque de Instrucciones 2;

## Selección múltiple

### Pseudocódigo

*Sequn* Valor Ordinal  
*rango 1* : Bloque de Instrucciones 1  
...  
*Sino*  
Bloque de Instrucciones *n*

### Código Pascal

*Case* Valor Ordinal *of*  
*rango 1* : Bloque de Instrucciones 1;  
...  
*Else*  
Bloque de Instrucciones *n*;

# Ecuación de Segundo Grado

## Diseño

**Entrada:**  $a, b, c \in \mathcal{R}, \quad a \neq 0$

**Salida:**

**Algoritmo:**

Obtener( $a, b, c$ )

$d \leftarrow b * b - 4 * a * c$

$v1 \leftarrow -b / (2 * a)$

$v2 \leftarrow \sqrt{|d|} / (2 * a)$

**Variables:**  $d \in \mathcal{R}$

# Ecuación de Segundo Grado

## Diseño

**Entrada:**  $a, b, c \in \mathcal{R}, \quad a \neq 0$

**Salida:**  $v1, v2 \in \mathcal{R}$

**Algoritmo:**

Obtener( $a, b, c$ )

$d \leftarrow b * b - 4 * a * c$

$v1 \leftarrow -b / (2 * a)$

$v2 \leftarrow \sqrt{|d|} / (2 * a)$

Si ( $d \geq 0$ ) Entonces

Devolver( $v1 + v2, v1 - v2$ )

**Variables:**  $d \in \mathcal{R}$

# Ecuación de Segundo Grado

## Diseño

**Entrada:**  $a, b, c \in \mathcal{R}, \quad a \neq 0$

**Salida:**  $v1, v2 \in \mathcal{R}$

**Algoritmo:**

Obtener( $a, b, c$ )

$d \leftarrow b * b - 4 * a * c$

$v1 \leftarrow -b / (2 * a)$

$v2 \leftarrow \sqrt{|d|} / (2 * a)$

Si ( $d \geq 0$ ) Entonces

    Devolver( $v1 + v2, v1 - v2$ )

Sino

    Devolver( $v1, v2, v1, -v2$ )

**Variables:**  $d \in \mathcal{R}$

## Ecuación de Segundo Grado

ec2if.pas

```
Program ec2if;  
Uses Crt;  
Var  
    a,b,c,d,v1,v2: Real;  
Begin  
    Read(a,b,c);  
    d:= b*b - 4 *a * c;  
    v1:= -b/(2*a);  
    v2:= sqrt(abs(d))/(2*a);  
    If (d>=0) Then  
        Write(v1+v2, v1-v2);  
    Else  
        Write(v1, v2, v1, -v2);  
End.
```

# Transformar Notas

## Planteamiento

Dado un valor numérico para una nota, encontrar la letra correspondiente según el código

Letra	Rango	Condición
A	[19,20]	Sobresaliente
B	[16,18]	Eximido
C	[10,15]	Aprobado
D	[5,9]	Reprobado
E	[1,4]	Deficiente
X		No Valida

# Transformar Notas

## Diseño

**Entrada:**  $nota \in \mathcal{Z}$

**Salida:**

**Algoritmo:**

**Variables:**

# Transformar Notas

## Diseño

**Entrada:**  $nota \in \mathcal{Z}$

**Salida:**  $letra \in ch$

**Algoritmo:**

**Variables:**

# Transformar Notas

## Diseño

**Entrada:**  $nota \in \mathcal{Z}$

**Salida:**  $letra \in ch$

**Algoritmo:**

Obtener( $nota$ )

Según  $nota$

19,20 :  $letra \leftarrow 'A'$

16 .. 18 :  $letra \leftarrow 'B'$

10 .. 15 :  $letra \leftarrow 'C'$

5 .. 9 :  $letra \leftarrow 'D'$

1 .. 4 :  $letra \leftarrow 'E'$

Sino

$letra \leftarrow 'X'$

Devolver( $letra$ )

**Variables:**

## Transformar Notas

notas.pas

```
Program notas;  
Uses Crt;  
Var  
    nota: Byte;  
    letra: Char;  
Begin  
    Read(nota);  
    Case (nota) of  
        19,20 : letra:= 'A';  
        16..18 : letra:= 'B';  
        10..15 : letra:= 'C';  
        5..9 : letra:= 'D';  
        1..4: letra:= 'E'  
    Else  
        letra:='X';  
    Write(letra);  
End.
```

## Repita para ...un número determinado de veces

Hace uso de una variable auxiliar que sirve de contador de ciclos de repetición, la cual puede ir en ascenso o descenso

### Pseudocódigo

*Para*  $IdVariable = Vi .. Vf$   
*Bloque de Instrucciones*

### Código Pascal

*For*  $IdVariable := Vi$  *To*  $Vf$  *Do*  
*Bloque de Instrucciones;*

---

*For*  $IdVariable := Vi$  *Downto*  $Vf$  *Do*  
*Bloque de Instrucciones;*

## Repita mientras

... solo si es necesario

El ciclo del lazo de repetición se inicia si la condición se cumple y se detiene cuando deja de cumplirse.

### Pseudocódigo

*Mientras Condición*  
*Bloque de Instrucciones*

### Código Pascal

*While Condición Do Begin*  
*Bloque de Instrucciones*  
*End;*

Si se necesita un contador de repeticiones éste debe controlarse en el bloque de instrucciones

Repita hasta

... al menos una vez

El ciclo del lazo de repetición se detiene si la condición se cumple.

### Pseudocódigo

*Repita*  
    *Bloque de Instrucciones*  
*Hasta* Condición

### Código Pascal

*Repeat*  
    *Bloque de Instrucciones*  
*Until* Condición;

Si se necesita un contador de repeticiones éste debe controlarse en el bloque de instrucciones

## Cálculo de promedio de notas

Calcular el promedio de  $n$  valores aleatorios correspondientes a las notas de un curso

$$\bar{x} = \sum_{i=1}^n \frac{x_i}{n}$$

Diseño

Entrada:

Salida:

Proceso:

## Cálculo de promedio de notas

Calcular el promedio de  $n$  valores aleatorios correspondientes a las notas de un curso

$$\bar{x} = \sum_{i=1}^n \frac{x_i}{n}$$

### Diseño

**Entrada:**  $n \in \mathbb{Z}, n > 0$

**Salida:**  $prom \in \mathbb{R}$

**Proceso:**

$$prom = \frac{1}{n} \sum_{i=1}^n x_i$$

## Cálculo de promedio de notas

### Diseño

#### Algoritmo:

```
Obtener( $n$ )  
 $prom \leftarrow 0$   
Para  $i \leftarrow 1..n$   
   $x \leftarrow \text{Aleatorio}(21)$   
   $prom \leftarrow prom + x$   
 $prom \leftarrow prom/n$   
Devolver( $prom$ )
```

**Variables:**  $x \in \mathcal{R}, x \in [0, 20]$

$i \in \mathcal{Z}$

# Cálculo de promedio de notas

## Diseño

### Algoritmo:

```
Obtener( $n$ )  
 $prom \leftarrow 0$   
Para  $i \leftarrow 1..n$   
   $x \leftarrow \text{Aleatorio}(21)$   
   $prom \leftarrow prom + x$   
 $prom \leftarrow prom/n$   
Devolver( $prom$ )
```

**Variables:**  $x, i \in \mathcal{Z}$ ,  $x \in [0, 20]$

$i \in \mathcal{Z}$

## Cálculo de promedio de notas

### Programa

```
Program promnotas;  
Uses Crt;  
Var  
    n,i,x : Integer;  
    prom : Real;  
Begin  
    Readln(n);  
    prom:=0;  
    For i:=1 to n do  
        Begin  
            x:=random(21);  
            prom:=prom +x;  
        End;  
    prom:=prom/n;  
    writeln(prom);  
End.
```

## Cálculo de producto de impares

Calcule el producto de los  $n$  primeros números impares.

$$1 \times 3 \times 5 \times 7 \dots$$

$$p = \prod_{i=1}^n 2i - 1$$

### Diseño

**Entrada:**  $n \in \mathcal{Z}$

**Salida:**  $p \in \mathcal{Z}$

**Algoritmo:**

*Obtener*( $n$ )

$p \leftarrow 1$

*Para*  $i \leftarrow 1..n$

$p \leftarrow p * (2 * i - 1)$

*Devolver*( $p$ )

**Variables:**  $i \in \mathcal{Z}$

## Cálculo de producto de impares

### Programa

```
Program prodimpar;  
Uses Crt;  
Var  
    n,p,i : Integer;  
Begin  
    Readln(n);  
    p:=1;  
    For i:=1 to n do  
        p:=p*(2*i-1);  
        writeln(p);  
End.
```

## Potencia de números enteros

Dados dos números enteros  $a, b$  calcular  $a^b$

$$\begin{aligned} 2^5 &= 2 \times 2 \times 2 \times 2 \times 2 & 5^{-2} &= \frac{1}{5^2} = \frac{1}{5 \times 5} \\ -2^3 &= -2 \times -2 \times -2 & -3^{-2} &= \frac{1}{-3^2} = \frac{1}{-3 \times -3} \end{aligned}$$

Diseño

Entrada:

Salida:

Proceso:

## Potencia de números enteros

Dados dos números enteros  $a, b$  calcular  $a^b$

$$\begin{aligned} 2^5 &= 2 \times 2 \times 2 \times 2 \times 2 & 5^{-2} &= \frac{1}{5^2} = \frac{1}{5 \times 5} \\ -2^3 &= -2 \times -2 \times -2 & -3^{-2} &= \frac{1}{-3^2} = \frac{1}{-3 \times -3} \end{aligned}$$

### Diseño

**Entrada:**  $a, b \in \mathcal{Z}$

**Salida:**  $p \in \mathcal{R}$

**Proceso:**

$$p = \prod_{i=1}^{|b|} a, \quad \begin{cases} 1 & b = 0 \\ \frac{1}{p}, & b < 0, \quad a \neq 0 \end{cases}$$

# Potencia de números enteros

## Diseño

### Algoritmo:

```
Obtener( $a, b$ )  
 $p \leftarrow 1$   
Para  $i \leftarrow 1..|b|$   
     $p \leftarrow p * a$   
Si  $((b < 0) \wedge (a \neq 0))$   
     $p \leftarrow 1/p$   
Devolver( $p$ )
```

Variables:  $i \in \mathbb{Z}$

## Potencia de números enteros

### Programa

```
Program potencia2;  
Uses Crt;  
Var  
    a,b,i : Integer;  
    p : real;  
Begin  
    Readln(a,b);  
    p:=1;  
    For i:=1 to abs(b) do  
        p:= p * a;  
    If ((b<0) and (a<>0)) Then  
        p:= 1/p;  
    writeln(p);  
End.
```

## Cálculo de la arco-tangente

Calcular los  $n$  primeros términos de la aproximación

$$\tan^{-1}(x) = \begin{cases} x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7} \cdots, & |x| < 1 \\ \pm \frac{\pi}{2} - \frac{1}{x} + \frac{1}{3x^3} - \frac{1}{5x^5} \cdots, & \begin{cases} + : x \geq 1 \\ - : x \leq -1 \end{cases} \end{cases}$$

### Análisis

$$\tan^{-1}(x) = \begin{cases} x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7} + \frac{x^7 x^2}{5+2} - \frac{x^9 x^2}{9+2} \cdots, & |x| < 1 \\ \pm \frac{\pi}{2} - \frac{1}{x} + \frac{1}{3x^3} - \frac{1}{(3+2)x^3 x^2} + \frac{1}{(5+2)x^5 x^2} \cdots, & \begin{cases} + : x \geq 1 \\ - : x \leq -1 \end{cases} \end{cases}$$

# Cálculo de la arco-tangente

## Diseño

**Entrada:**  $x \in \mathcal{R}, n \in \mathcal{Z}, n > 0$

**Salida:**  $atan \in \mathcal{R}$

**Procedimiento:**

$$atan = \begin{cases} \sum_{i=1}^n \frac{n_i}{d_i}, & |x| < 1 \\ \pm \frac{\pi}{2} + \sum_{i=2}^n \frac{n_i}{d_1 d_2 i}, & \begin{cases} + : x \geq 1 \\ - : x \leq -1 \end{cases} \end{cases} \quad \begin{cases} \begin{cases} n_1 = x \\ n_i = -n_{i-1}x^2 \\ d_1 = 1 \\ d_i = d_{i-1} + 2 \end{cases} \\ \begin{cases} n_1 = -1 \\ n_i = -n_{i_1} \\ d1_1 = 1 \\ d1_i = d1_{i-1} + 2 \\ d2_1 = x \\ d2_i = d2_{i-1}x^2 \end{cases} \end{cases}$$

## Cálculo de la arco-tangente

### Algoritmo

*Obtener*( $n, x$ )

$atan \leftarrow 0$

Si ( $|x| < 1$ )

$n1 \leftarrow x$

$d1 \leftarrow 1$

Para  $i \leftarrow 1..n$

$atan \leftarrow atan + n1/d1$

$n1 \leftarrow -n1 * x * x$

$d1 \leftarrow d1 + 2$

Sino

$n1 \leftarrow -1$

$d11 \leftarrow 1$

$d21 \leftarrow x$

Para  $i \leftarrow 2..n$

$atan \leftarrow atan + n1/(d11 * d21)$

$n1 \leftarrow -n1$

$d11 \leftarrow d11 + 2$

$d21 \leftarrow d21 * x * x$

Si ( $x \geq 1$ )

$atan \leftarrow atan + \pi/2$

Sino

$atan \leftarrow atan - \pi/2$

*Devolver*( $atan$ )

**Variables:**  $i \in \mathbb{Z}$

$n1, d1, d11, d21 \in \mathcal{R}$

## Cálculo de Área

Calcular usando aproximación de los trapecios y de simpson para el área bajo  $f(x) = \frac{1}{1-x}$  dividiendo el intervalo  $[a, b]$  en  $n$  segmentos.

$$\int_a^b f(x)dx \simeq \sum_{i=0}^{n-1} \frac{b-a}{2n} [f(x_i) + f(x_{i+1})]$$

$$\int_a^b f(x)dx \simeq \sum_{i=0}^{n-1} \frac{b-a}{6n} [f(x_i) + 4f(x_m) + f(x_{i+1})]$$

## Cálculo de Area

### Diseño

**Entrada:**  $a, b, n \in \mathbb{Z}$ ,  $a < b$ ,  $n > 1$

**Salida:**  $at, as \in \mathcal{R}$

**Procedimiento:**

$$at = \frac{b-a}{2n} \sum_{i=0}^{n-1} \left[ \frac{1}{1-x_i} + \frac{1}{1-x_{i+1}} \right]$$

$$as = \frac{b-a}{6n} \sum_{i=0}^{n-1} \left[ \frac{1}{1-x_i} + \frac{4}{1-x_m} + \frac{1}{1-x_{i+1}} \right]$$

$$x_0 = a \quad x_i = x_{i-1} + \frac{b-a}{n} \quad x_m = \frac{x_i + x_{i-1}}{2} \quad i \in [1, n]$$

## Cubos

## Método de Nicómano

Escriba un algoritmo que calcule los primeros  $n$  cubos mediante la conjetura de Nicómaco

- El primer impar: 1
- Sume los dos siguientes impares:  $3 + 5 = 8$ .
- Sume los tres siguientes impares:  $7 + 9 + 11 = 27$ .

### Análisis

$$k_1 = 1 \quad k_2 = k_1 + 2 \quad \dots$$
$$cubo_1 = k_1 \quad cubo_i = \sum_{j=1}^i k_j, \quad i \in [2, n]$$

# Cubos

# Método de Nicómano

## Diseño

**Entrada:**  $n \in \mathcal{Z}, n > 0$

**Salida:**  $cubo \in \mathcal{Z}$

**Procedimiento:**

$k = 1$

$cubo_1 = k$

$\left\{ \begin{array}{l} k = k + 2 \end{array} \right.$

$\left\{ \begin{array}{l} cubo_i = \sum_{j=1}^i k \quad i \in [2, n] \end{array} \right.$

**Algoritmo:**

*Obtener*( $n$ )

$k \leftarrow 1$

*Devolver*( $k$ )

*Para*  $i \leftarrow 2..n$

$cubo \leftarrow 0$

*Para*  $j \leftarrow 1..i$

$k \leftarrow k + 2$

$cubo \leftarrow cubo + k$

*Devolver*( $cubo$ )

**Variables:**  $k, i, j \in \mathcal{Z}$

# Subprogramas

Un subprograma es un programa que resuelve un problema en particular y que puede ser llamado desde otro programa.

Como todo programa, un subprograma tiene variables de entrada, variables de salida, variables internas y un cuerpo conformado por el conjunto de instrucciones que permiten ejecutar las tareas para las que fue diseñado.

Para cada subprograma debemos hacer un diseño algorítmico.

Los subprogramas pueden ser Funciones o Procedimientos.

En pascal los subprogramas se incluyen en la sección de Declaraciones: antes del **Begin** que indica el Inicio del Programa principal.

## Funciones

## Subprograma que retorna un valor único

### Pseudocódigo

*Función* *IdFunción* ( *ListaVariablesEntrada* ): *Tipo*  
    *Bloque de Instrucciones*  
*V*: *ListaVariablesInternas*

### Código Pascal

```
Function IdFunción ( ListaVariablesEntrada ): Tipo  
Var ListaVariablesInternas;  
Begin  
    Bloque de Instrucciones;  
End;
```

# Procedimientos

Subprograma que puede o no retornar valores

## Pseudocódigo

```
Procedimiento IdProced ( ListaVarEntrada, ListaVarSalida )  
    Bloque de Instrucciones  
V: ListaVarInternas
```

## Código Pascal

```
Procedure IdProced ( ListaVarEntrada, var ListaVarSalida )  
Var ListaVarInternas;  
Begin  
    Bloque de Instrucciones;  
End;
```

# Datos Simples

# Enumerados

Permite construir variables ordinales donde los valores asociados están declarados en orden.

## Pseudocódigo

$IdVariable \in ( ListaValores )$

## Código Pascal

$IdVariable : ( ListaValores );$

## Ejemplo

$Laborables \in ( Lunes, Martes, Miercoles, Jueves, Viernes )$

# Datos Simples

# Rangos

Variable ordinal donde los valores asociados están ordenados y pertenecen a un conjunto de datos definidos previamente.

## Pseudocódigo

*IdVariable*  $\in$  *v0..vf*

## Código Pascal

*IdVariable*: *v0..vf* ;

## Ejemplo

*Nota*  $\in$  *0..20*

## Tipos Definidos por el Usuario

En Pascal se ubican en la sección de Declaraciones antes de la declaración de Variables.

### Pseudocódigo

*IdTipo* ∈ *DescTipoConstruido*

### Código Pascal

*Type*  
*IdTipo* = *DescTipoConstruido* ;

# Datos Compuestos

# Vectores

Tipo de dato Compuesto de múltiples valores del mismo tipo.  
Los valores están indexados según un rango predefinido.

## Pseudocódigo

$IdVariable \in Arreglo[Vo..Vf] \in Tipo$

## Código Pascal

$IdVariable : Array[Vo..Vf] Of Tipo;$

# Datos Compuestos

# Cadenas

Arreglo de Caracteres de longitud definida.

Pseudocódigo

*IdVariable*  $\in \mathcal{C}[long]$

Código Pascal

*IdVariable* : *String*[*long*];

## Datos Compuestos

## Matrices

Arreglo de dos dimensiones.

En el caso más general podemos escribir arreglos multidimensionales.

### Pseudocódigo

$IdVariable \in Arreglo[V1o..V1f, V2o..V2f] \in Tipo$

### Código Pascal

$IdVariable : Array[V1o..V1f, V2o..v2f] Of Tipo;$

# Datos Simples

# Conjuntos

Variable ordinal donde los valores asociados están ordenados y pertenecen a un conjunto de datos definidos previamente.

## Pseudocódigo

$IdVariable \in \{v0..vf\}$

## Código Pascal

$IdVariable: Set\ Of\ v0..vf ;$

## Ejemplo

$Mes \in \{1..12\}$

# Datos Compuestos

# Registros

Tipo de Dato construido a partir de una lista de Variables.

## Pseudocódigo

$IdTipo \in \{ \text{Bloque de Declaraciones de Variables} \}$

## Código Pascal

$IdTipo = Record$   
    *Bloque de Declaraciones de Variables*  
 $end;$

## Ejemplo

$Alumno \in \{$   
    *nombre*  $\in \mathcal{C}$   
    *edad*  $\in \mathcal{Z}$   $\}$



### Potencias:

$$x^{-y} \equiv \frac{1}{x^y}$$

$$x^{yz} \equiv (x^y)^z$$

$$x^{y+z} \equiv x^y x^z$$

$$x^y \equiv e^{(y * \ln(x))}, x > 0$$

### Logarítmicas:

$$\ln(x) \equiv \log_e(x), x > 0$$

$$b^{\log_b(x)} \equiv \log_b(b^x) \equiv x, x > 0$$

$$\log_b(xy) \equiv \log_b(x) + \log_b(y), x, y > 0$$

$$\log_b(x) \equiv \frac{\log_a(x)}{\log_a(b)}, x, b > 0$$

$$\log_b(x^y) \equiv y \log_b(x), x > 0$$

$$\log_b(x/y) \equiv \log_b(x) - \log_b(y), x, y > 0$$

### Trigonómicas:

$$\sin^2(x) + \cos^2(x) \equiv 1$$

$$\sin(x \pm y) \equiv \sin(x)\cos(y) \pm \cos(x)\sin(y)$$

$$\cos(x \pm y) \equiv \cos(x)\cos(y) \mp \sin(x)\sin(y)$$

$$\tan(x) \equiv \frac{\sin(x)}{\cos(x)}, x \neq (2n+1)\pi/2, n \in \mathbb{Z}$$

$$\sec(x) \equiv \frac{1}{\cos(x)}, x \neq (2n+1)\pi/2, n \in \mathbb{Z}$$

$$\csc(x) \equiv \frac{1}{\sin(x)}, x \neq n\pi, n \in \mathbb{Z}$$

$$\cosh^2(x) - \sinh^2(x) \equiv 1$$

$$\sinh(x) \equiv \frac{1}{2}(e^x - e^{-x})$$

$$\cosh(x) \equiv \frac{1}{2}(e^x + e^{-x})$$

$$\tanh(x) \equiv \frac{\sinh(x)}{\cosh(x)}, x \neq 0$$

$$\sin^{-1}(x) + \cos^{-1}(x) \equiv \pi/2$$

$$\sin^{-1}(x) \equiv \tan^{-1}\left(\frac{x}{\sqrt{1-x^2}}\right), |x| < 1$$

$$\cos^{-1}(x) \equiv \tan^{-1}\left(\frac{\sqrt{1-x^2}}{x}\right), |x| < 1$$

$$\operatorname{ctan}^{-1}(x) \equiv \tan^{-1}(1/x), x \neq 0$$

$$\sec^{-1}(x) \equiv \cos^{-1}(1/x), |x| \geq 1, x \neq 0$$





$$\csc^{-1}(x) \equiv \sin^{-1}(1/x), |x| \geq 1, x \neq 0$$

$$\operatorname{csch}(x) \equiv \frac{1}{\sinh(x)}, x \neq 0$$

$$\operatorname{sech}(x) \equiv \frac{1}{\cosh(x)}$$

$$\operatorname{ctanh}(x) \equiv \frac{\cosh(x)}{\sinh(x)}$$

# Referencias

-  [JL03] Luis Joyanes Aguilar.  
Fundamentos de Programación: Algoritmos Estructuras de datos y Objetos, McGraw Hill. Tercera Edición. 2003
-  [SW93] William I. Salmon  
Introducción a la computación con Turbo Pascal. Estructuras y Abstracciones. Addison-Wesley Iberoamericana. 1993
-  [KW] John Konvalina, Stanley Wileman  
Programación con Pascal. Mc Graw - Hill
-  [GP] Peter Grogono  
Programación en Pascal. Addison - Wesley