



## Resolución de Problemas y Algoritmos

### Clase 14: Estructura de bloques, entornos de referencia, y visibilidad de identificadores.



**Dr. Alejandro J. García**  
http://cs.uns.edu.ar/~ajg



Departamento de Ciencias e Ingeniería de la Computación  
Universidad Nacional del Sur  
Bahía Blanca - Argentina

### Conceptos: Pascal es estructurado por bloques

```
PROGRAM MIPROGRAMA;  
CONST ..... TYPE.... VAR .....  
FUNCTION F(X:real):real;  
CONST ..... TYPE....  
VAR .....  
PROCEDURE ...  
FUNCTION ...  
BEGIN ...sentencias...END;
```

- En Pascal, un **programa** constituye un **bloque** compuesto por:
  - constantes, tipos, variables,
  - funciones, procedimientos,
  - y sentencias.
- Cada **procedimiento** o **función** también constituye un **bloque** por:
  - **parámetros**
  - constantes, tipos, variables,
  - procedimientos, funciones,
  - y sentencias.

Resolución de Problemas y Algoritmos Dr. Alejandro J. García 2

### Conceptos: Pascal es estructurado por bloques

```
PROGRAM MIPROGRAMA;  
CONST ..... TYPE.... VAR .....  
FUNCTION F(X:real):real;  
CONST ..... TYPE....  
VAR .....  
PROCEDURE ...  
FUNCTION ...  
BEGIN ...sentencias...END;
```

```
PROCEDURE P(Var X: char);  
CONST..... TYPE....  
VAR .....  
PROCEDURE ...  
FUNCTION ...  
BEGIN...sentencias...END;
```

```
BEGIN ...sentencias...  
END.
```

Este programa Tiene 7 bloques

Resolución de Problemas y Algoritmos Dr. Alejandro J. García 3

### “El límite está dado por su imaginación”

```
PROGRAM PROGRAMA1;  
PROCEDURE...  
FUNCTION ...  
PROCEDURE...  
PROCEDURE...  
FUNCTION ...  
PROCEDURE...
```

{...puede incluir todos los procedimientos o funciones que quiera...}

```
BEGIN ... END.
```

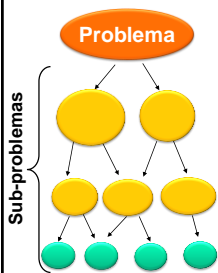
```
PROGRAM PROGRAMA2;  
PROCEDURE ...  
FUNCTION ...  
PROCEDURE ...  
BEGIN ..... END;  
BEGIN ..... END;
```

{..y en cada bloque, todo el "anidamiento" que quiera}

```
BEGIN... END.
```

Resolución de Problemas y Algoritmos Dr. Alejandro J. García 4

### Conceptos: Técnica de resolución de problemas

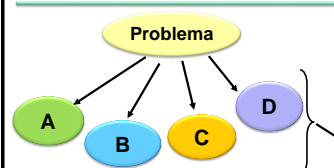


**Técnica: Resolución de un problema por descomposición en problemas más simples.**

- Cuando se intenta resolver un problema complejo puede abordarse la solución descomponiendo (dividiendo) el problema en partes (sub-problemas) más simples.
- De tal manera que la solución de las partes permita resolver el problema original.
- Si los sub-problemas siguen siendo complejos pueden también dividirse hasta llegar a problemas que no necesitan dividirse.

Resolución de Problemas y Algoritmos Dr. Alejandro J. García 5

### División de un problema en sub-problemas



```
Program SOLUCIÓN;  
Function A  
Procedure B  
Function C  
Procedure D  
Begin  
...  
End.
```

**Metología:** Para resolver un problema complejo se propone:

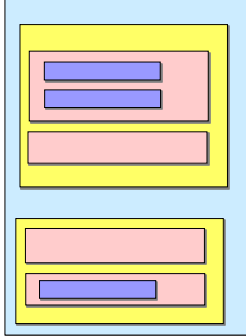
- 1) **dividir** en subproblemas,
- 2) **resolver** cada parte y luego para cada parte **implementar primitivas** en Pascal: como funciones o procedimientos

Resolución de Problemas y Algoritmos Dr. Alejandro J. García 6

El uso total o parcial de este material está permitido siempre que se haga mención explícita de su fuente:  
 “Resolución de Problemas y Algoritmos. Notas de Clase”. Alejandro J. García. Universidad Nacional del Sur. (c) 2015

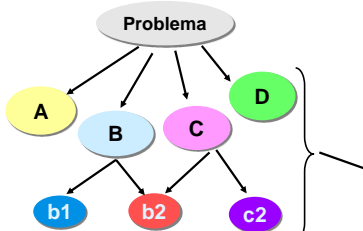
### Pascal: estructurado por bloques

- En un programa pueden incluirse tantos procedimientos y funciones como se desee.
- Cada uno de ellos puede a su vez tener sus bloques internos y así siguiendo.
- Esto permite implementar cualquier división del problema en sub-problemas que se diseñe.**



Resolución de Problemas y Algoritmos Dr. Alejandro J. García 7

### División del problema en subproblemas



El programa puede reflejar la división del problema realizada en el diseño. En Pascal no hay límite en cantidad o anidamiento de bloques.

```

Program SOLUCIÓN;
  Function A
  Function b2
  Procedure B
  Procedure b1
  Function C
  Procedure c2
  Procedure D
Begin ... End.
    
```

Resolución de Problemas y Algoritmos Dr. Alejandro J. García 8

### Conceptos: bloque e identificadores

- Identificadores que puede tener un BLOQUE:**
  - identificadores de constantes
  - identificadores de tipos
  - identificadores de variables
  - identificadores de parámetros (en proc. y fn.)
  - identificadores de procedimientos
  - identificadores de funciones
- Dentro de un mismo bloque no puede haber dos identificadores iguales para distintos elementos.
- Dos elementos pueden tener el mismo identificador si pertenecen a **diferentes bloques**.

Resolución de Problemas y Algoritmos Dr. Alejandro J. García 9

### Conceptos: bloques e identificadores

- Cada procedimiento y función determina un nuevo **bloque**.
- En cada bloque se puede tanto **declarar nuevos** identificadores como **usar** identificadores.
- En esta clase se introducen las reglas que definen **cuales identificadores son visibles para un bloque** (i.e., pueden usarse) aunque estén declarados en otros bloques del programa.
- A continuación se mostrará un programa en Pascal (llamado simple) con el objetivo de ejemplificar los **nuevos conceptos** que surgen de utilizar procedimientos y funciones.
- El programa no resuelve ningún problema en particular, está construido desde un punto de vista didáctico para mostrar la mayor cantidad de declaración y uso de identificadores.

Resolución de Problemas y Algoritmos Dr. Alejandro J. García 10

### Ejemplo: bloques e identificadores

```

PROGRAM simple; {para entender los conceptos}
Const Pi= 3.14; type Tdig=0..9; var A, B, C:CHAR;
PROCEDURE P1 (A:REAL);
var B: REAL; F2: Tdig;
begin B:= A; WRITE(B) end; {P1}
PROCEDURE P2 (A:REAL);
var B, MIA: real;
FUNCTION F2 (A:REAL):REAL;
var B, DE_F2: REAL;
begin B:= A; F2:= B + Pi; end; {F2}
begin B:= A; WRITE(F2(A)); P1(B) end; {P2}
BEGIN
P2(5); P1(10);
END.
    
```

Escriba en sus notas este programa (mientras lo copiamos en el pizarrón)

Resolución de Problemas y Algoritmos Dr. Alejandro J. García 11

### Preguntas sobre el programa "simple"

- ¿puedo llamar a P1 desde las sentencias de P2?
- ¿puedo llamar a F2 desde las sentencias de P2?
- ¿puedo llamar a F2 desde las sentencias de P1?
- ¿puedo llamar a P1 desde las sentencias de F2?
- ¡ HAGA AHORA SUS PREGUNTAS !** (y copie las de sus compañeros)
- Pregunta más general:** ¿desde qué lugar del programa puedo llamar a una función o procedimiento?
- ¿en qué bloques puedo usar la variable "MIA"?
- ¿y la variable DE\_F2?
- ¿en qué bloques puedo usar una variable?
- Todas las respuestas en la teoría que sigue a continuación...**

Resolución de Problemas y Algoritmos Dr. Alejandro J. García 12

El uso total o parcial de este material está permitido siempre que se haga mención explícita de su fuente: "Resolución de Problemas y Algoritmos. Notas de Clase". Alejandro J. García. Universidad Nacional del Sur. (c) 2015

**Ejemplo: bloques (demarcados con un recuadro)**

```
PROGRAM simple; {para entender los conceptos}
Const Pi = 3.14; type Tdig=0..9; var A, B, C:CHAR;
PROCEDURE P1 (A:REAL);
var B: REAL; F2: Tdig;
begin B:= A; WRITE(B) end; {P1}
PROCEDURE P2 (A:REAL);
var B, MIA: real;
FUNCTION F2(A:REAL):REAL;
var B, DE_F2: REAL;
begin B := A; F2 := B + Pi; end; {F2}
begin B:=A; WRITE(F2(A)); P1(B) end; {P2}
BEGIN
P2(5); P1(10);
END.
```

Resolución de Problemas y Algoritmos Dr. Alejandro J. García 13

**Repaso: diferentes elementos de un programa**

```
program simple; {para entender los conceptos}
const Pi = 3.14; type Tdig =0..9; var A, B, C: CHAR;
PROCEDURE P1 (A : REAL);
var B: REAL; F2: Tdig;
begin B:= A; WRITE (B) end; {P1}
PROCEDURE P2 (A : REAL);
var B, MIA: real;
FUNCTION F2 (A : REAL):REAL;
var B, DE_F2 : REAL;
begin B := A; F2 := B + Pi ; end; {F2}
begin B:=A; WRITE ( F2( A )); P1 ( B ) end; {P2}
begin
P2 (5); P1 (10);
end .
```

(1) Palabras reservadas  
(2) Símbolos y valores.  
(3) Comentarios.  
(4) Identificadores.

Resolución de Problemas y Algoritmos Dr. Alejandro J. García 14

**Repaso: diferentes elementos de un programa**

```
program simple; {para entender los conceptos}
const Pi = 3.14; type Tdig =0..9; var A, B, C: CHAR;
PROCEDURE P1 (A : REAL);
var B: REAL; F2: Tdig;
begin B:= A; WRITE (B) end; {P1}
PROCEDURE P2 (A : REAL);
var B, MIA: real;
FUNCTION F2 (A : REAL):REAL;
var B, DE_F2 : REAL;
begin B := A; F2 := B + Pi ; end; {F2}
begin B:=A; WRITE ( F2( A )); P1 ( B ) end; {P2}
begin
P2 (5); P1 (10);
end .
```

(1) Palabras reservadas: tienen un significado propio y el programador no puede cambiarlo.

Resolución de Problemas y Algoritmos Dr. Alejandro J. García 15

**Repaso: diferentes elementos de un programa**

```
program simple; {para entender los conceptos}
const Pi = 3.14; type Tdig =0..9; var A, B, C: CHAR;
PROCEDURE P1 (A : REAL);
var B: REAL; F2: Tdig;
begin B:= A; WRITE (B) end; {P1}
PROCEDURE P2 (A : REAL);
var B, MIA: real;
FUNCTION F2 (A : REAL):REAL;
var B, DE_F2 : REAL;
begin B := A; F2 := B + Pi ; end; {F2}
begin B:=A; WRITE ( F2( A )); P1 ( B ) end; {P2}
begin
P2 (5); P1 (10);
end .
```

(2) Símbolos y valores: tienen un significado propio y el programador no puede cambiarlo.

Resolución de Problemas y Algoritmos Dr. Alejandro J. García 16

**Repaso: diferentes elementos de un programa**

```
program simple; {para entender los conceptos}
const Pi = 3.14; type Tdig =0..9; var A, B, C: CHAR;
PROCEDURE P1 (A : REAL);
var B: REAL; F2: Tdig;
begin B:= A; WRITE (B) end; {P1}
PROCEDURE P2 (A : REAL);
var B, MIA: real;
FUNCTION F2 (A : REAL):REAL;
var B, DE_F2 : REAL;
begin B := A; F2 := B + Pi ; end; {F2}
begin B:=A; WRITE ( F2( A )); P1 ( B ) end; {P2}
begin
P2 (5); P1 (10);
end .
```

(4) Identificadores: tienen el significado que quiera el programador. Algunos son predefinidos.

Observación: a los predefinidos es posible cambiarle su significado.

Resolución de Problemas y Algoritmos Dr. Alejandro J. García 17

**Conceptos: declaración vs. referencia**

Es importante distinguir entre:

- La **declaración de un identificador** de constante, tipo, variable, parámetro, función, o procedimiento. **Ejemplos:**  
`CONST pi=3.14; TYPE Tdig =0..9; VAR precio: real;`  
`PROCEDURE recargo(precio,rec: real; var monto: real);`
- La **referencia o el uso de un identificador**. **Ejemplos:**  
`recargo(24,incremento,precio);`  
`a_pagar := precio+intereses(round(precio));`

En cada bloque, se declaran identificadores; y además, se hace referencia (usan) identificadores.  
 A continuación se muestra para el programa "simple"  
 (1) en primer lugar donde se **declaran** identificadores  
 y (2) en segundo lugar donde se **usan** identificadores.

Resolución de Problemas y Algoritmos Dr. Alejandro J. García 18

El uso total o parcial de este material está permitido siempre que se haga mención explícita de su fuente:  
 "Resolución de Problemas y Algoritmos. Notas de Clase". Alejandro J. García. Universidad Nacional del Sur. (c) 2015

### Ejemplos de declaración de identificadores

```

PROGRAM simple; {para entender los conceptos}
Const Pi = 3.14; type Tdig = 0..9; var A, B, C :CHAR;
PROCEDURE P1(A : REAL);
var B: REAL; F2: Tdig;
begin B:= A; WRITE(B) end; {P1}
PROCEDURE P2(A : REAL);
var B, MIA: real;
FUNCTION F2(A : REAL):REAL;
var B, DE_F2: REAL;
begin B:= A; F2:= B + Pi; end; {F2}
begin B:= A; WRITE(F2(A)); P1(B) end; {P2}
BEGIN
P2(5); P1(10);
END.
    
```

Resolución de Problemas y Algoritmos Dr. Alejandro J. García 19

### Ejemplos de uso de identificadores

```

PROGRAM simple; {para entender los conceptos}
Const Pi = 3.14; type Tdig = 0..9; var A, B, C :CHAR;
PROCEDURE P1(A : REAL);
var B: REAL; F2: Tdig;
begin B:= A; WRITE(B) end; {P1}
PROCEDURE P2(A : REAL);
var B, MIA: real;
FUNCTION F2(A : REAL):REAL;
var B, DE_F2: REAL;
begin B:= A; F2:= B + Pi; end; {F2}
begin B:= A; WRITE(F2(A)); P1(B) end; {P2}
BEGIN
P2(5); P1(10);
END.
    
```

Resolución de Problemas y Algoritmos Dr. Alejandro J. García 20

### Concepto: Entorno de referencia para un bloque B

**El entorno de referencia de un bloque B** está formado por los siguientes cuatro entornos:

1. El **entorno local**: conjunto de identificadores (parámetros formales, constantes, tipos, variables, el nombre de los procedimientos y funciones) **declarados** dentro del **bloque B**.
2. El **entorno global**: conjunto de identificadores **declarados** en el bloque del programa principal.
3. El **entorno no-local**: conjunto de identificadores **declarados** en los bloques que contienen al **bloque B**, exceptuando al global.
4. El **entorno predefinido**: conjunto de identificadores ya **declarados** por el compilador de Pascal y disponible para todo programa (Ej: maxint, char, write, eof).

Ejemplo: considere el programa simple mostrado antes, indique cuales son sus bloques y el entorno de referencia de cada bloque.

Resolución de Problemas y Algoritmos Dr. Alejandro J. García 21

### Ejemplos de entornos de referencia

- El programa “simple” tiene 4 bloques: P1, P2, F2 y el bloque del programa “simple”.
- A continuación se muestran los entornos de referencia para cada uno de estos bloques.
- Observe que el entorno predefinido y el entorno global es siempre el mismo para todos.

Resolución de Problemas y Algoritmos Dr. Alejandro J. García 22

### Ejemplos: bloques del programa “simple”

**Entorno de referencia para el Bloque “F2”**

- Entorno local: A, B, DE\_F2
- Entorno no-local: A, B, MIA, F2 (declarados en P2)
- Entorno global: Pi, Tdig, A, B, C, P1, P2
- Entorno predefinido: maxint, char, write, ...

**Entorno de referencia para el Bloque “P2”**

- Entorno local: A, B, MIA, F2
- Entorno no-local: (vacío, no tiene)
- Entorno global: Pi, Tdig, A, B, C, P1, P2
- Entorno predefinido: maxint, char, write, ...

Resolución de Problemas y Algoritmos Dr. Alejandro J. García 23

### Ejemplos: bloques del programa “simple”

**Entorno de referencia para el Bloque “P1”**

- Entorno local: A, B, F2
- Entorno no-local: (vacío, no tiene)
- Entorno global: Pi, Tdig, A, B, C, P1, P2
- Entorno predefinido: (el mismo siempre para todos)

**Entorno de referencia para el Bloque “simple”**

- Entorno no-local: (vacío, no tiene)
- Entorno global (y también local): Pi, Tdig, A, B, C, P1, P2
- Entorno predefinido: (el mismo siempre para todos)

Resolución de Problemas y Algoritmos Dr. Alejandro J. García 24

El uso total o parcial de este material está permitido siempre que se haga mención explícita de su fuente:  
 “Resolución de Problemas y Algoritmos. Notas de Clase”. Alejandro J. García. Universidad Nacional del Sur. (c) 2015

**Conceptos: identificadores ocultos**

Cuando se hace **referencia a un identificador**:

1. primero se busca en su entorno de referencia local,
2. luego en su entorno de referencia no local,
3. luego en su entorno de referencia global,
4. y finalmente en el entorno de referencia predefinido

Por lo anterior, **si hay identificadores iguales en diferentes entornos uno oculta al otro**.

1. Un identificador de nombre N en un entorno local **oculta** a todo identificador del mismo nombre N en otro entorno (no-local, global, predefinido)
2. Uno no-local N **oculta** a otro N global o predefinido,
3. Un identificador global N **oculta** a uno predefinido N

Resolución de Problemas y Algoritmos Dr. Alejandro J. García 25

**Ejemplo de identificador oculto**

```
PROGRAM PruebaDMS;
TYPE digito = 0..9;
VAR N, D :Integer;

FUNCTION digito_mas_significativo(N:integer): digito;
BEGIN
  if N < 0 then N:=~1*N;
  while (N >= 10) do N:=N div 10;
  digito_mas_significativo:= N;
END;

BEGIN
  write('Ingrese un número:');
  readln(N);
  D:=digito_mas_significativo(N);
  writeln('el D.M.S. de', N, 'es', D);
END.
```

El nombre del parámetro puede ser igual a uno de una variable global.

Aunque tengan el mismo nombre, los cambios del parámetro N no afectarán a la variable global N, ya que el parámetro oculta a la variable global.

Sugerencia: copie el programa y ejecute en la máquina para ver la traza real en pantalla.

Resolución de Problemas y Algoritmos Dr. Alejandro J. García 26

**Conceptos: identificador visible y alcance de un identificador**

- Un identificador es **referenciable** en un bloque, si es parte de su entorno de referencia y no está oculto.
- Un identificador es **visible**, si es referenciable.
- El **alcance** de un identificador D, son aquellas sentencias (o bloques) del programa donde el identificador D es visible.

Ejercicios propuestos:

- Para cada uno de los cuatro bloques del programa **simple**, encuentre los identificadores visibles (referenciables).
- Indique el alcance del identificador P1 y el alcance de la variable MIA.

Resolución de Problemas y Algoritmos Dr. Alejandro J. García 27

**Ejemplos**

- La constante **Pi** es visible (referenciable) en todos los bloques (ya que está en todos los entornos por ser parte del entorno global). Lo mismo ocurre con el procedimiento **P1** y el tipo **Tdig**.
- La función **F2** es visible en **P2** y en **F2**.
- La variable **"de\_f2"** solamente es visible en **F2**.

Resolución de Problemas y Algoritmos Dr. Alejandro J. García 28

**Reflexión final**

Como tarea que ayudará a comprender mejor los conceptos que se han compartido en lo anterior, se sugiere que en cada uno de los ejercicios y problemas de los prácticos reflexione sobre:

- Los identificadores declarados y usados en cada bloque.
- El entorno de referencia de cada bloque.
- ¿En que caso es interesante usar identificadores del entorno predefinido, cuando del entorno global y cuando del entorno local? ¿La misma respuesta vale para tipos, constantes, variables o primitivas?

Resolución de Problemas y Algoritmos Dr. Alejandro J. García 29

**Archivos como parámetros**

- Las **funciones** y los **procedimientos** pueden recibir datos de tipo FILE (archivos) como parámetros.
- **Ejemplos:**  

```
MostrarContenidoArchivo( archivo1);
IF Pertenece_elemento( E, archivo1) then ...
Copiar_contenido( archivo1, archivo_nuevo);
```
- En Pascal, es **obligatorio** que un **parámetro** de tipo **archivo** sea un parámetro **por** referencia.
- Dado que los parámetros por referencia deben ser de tipos idénticos, debe crearse un identificador de tipo archivo.

Resolución de Problemas y Algoritmos Dr. Alejandro J. García 30

El uso total o parcial de este material está permitido siempre que se haga mención explícita de su fuente:  
**"Resolución de Problemas y Algoritmos. Notas de Clase". Alejandro J. García. Universidad Nacional del Sur. (c) 2015**

**Repaso: compatibilidad entre parámetros**

En un parámetro POR REFERENCIA, el tipo del parámetro formal **debe ser idéntico** al tipo del parámetro efectivo. Estos es, se cumple que:

- Están declarados con el mismo identificador de tipo.
- Los identificadores de tipo son diferentes ( ej: **T1** y **T2**) pero han sido definidos como equivalentes por una declaración de la forma **T1 = T2**.

De esta forma es **incorrecto**:

```

VAR F1: FILE OF Integer;
...
PROCEDURE ejemplo( VAR A: FILE OF Integer);
...
ejemplo(F1);
    
```

**MAL**

**A y F1 NO SON DE TIPOS IDÉNTICOS**

Resolución de Problemas y Algoritmos Dr. Alejandro J. García 31

**Repaso: compatibilidad entre parámetros**

En un parámetro POR REFERENCIA, el tipo del parámetro formal **debe ser idéntico** al tipo del parámetro efectivo. Estos es, se cumple que:

- Están declarados con el mismo identificador de tipo.

De esta forma ahora **SI** es correcto:

```

TYPE TipoArchi = FILE OF Integer;
VAR F1: TipoArchi
...
PROCEDURE ejemplo( VAR A: TipoArchi );
...
ejemplo(F1);
    
```

**OK**

**Ahora A y F1 si son de tipos idénticos**

Resolución de Problemas y Algoritmos Dr. Alejandro J. García 32

**Program Ejemplo;**

```

TYPE TipoElemento = Integer;
TipoArch: FILE OF TipoElemento;
VAR F1: TipoArch;
PROCEDURE mostrara ( VAR archi: TipoArch; separador: char);
Var elemento: TipoElemento;
begin {... muestra el contenido de un archivo de números enteros usando un "separador" enviado por parámetro...}
Reset(archi);
while not eof(archi) do begin
read(archi, elemento); write(elemento, ' ', separador, ' ');
end; {while}
close(archi);
End;
begin {programa}
assign(F1, 'mis-numeros.datos');
mostrarA(F1, ',');
end. {fin del programa}
    
```

Con archivos es obligatorio usar parámetros por referencia

Resolución de Problemas y Algoritmos Dr. Alejandro J. García 33

**¡Ahora a practicar y hacer experiencia!**

Hay muchos conceptos nuevos para poner en práctica, y muchos problemas en los prácticos esperando por ser resueltos.

Disfrute mientras trabaja y recuerde que siempre estamos felices que vengan a compartir con nosotros sus soluciones, o sus dudas.

😊

Resolución de Problemas y Algoritmos Dr. Alejandro J. García 34

El uso total o parcial de este material está permitido siempre que se haga mención explícita de su fuente:  
 “Resolución de Problemas y Algoritmos. Notas de Clase”. Alejandro J. García. Universidad Nacional del Sur. (c) 2015