

RESOLUCIÓN DE PROBLEMAS Y ALGORITMOS

CLASE 16

Estructura de bloques, entornos de referencia, y visibilidad de identificadores.

Luciano H. Tamargo
<http://cs.uns.edu.ar/~lt>
 Depto. de Ciencias e Ingeniería de la Computación
 Universidad Nacional del Sur, Bahía Blanca
 2016

```
0 1 1 1 0 0
1 0 0 1 1
1 0 1 1 0
0 1 1 1 0
0 1 1 1 0
1 0 0 1 1
0 1 1 1 0
0 1 1 1 0
1 0 0 1 1
0 1 1 1 0
1 0 0 1 1
1 1 1
0 0
1
```

CONCEPTOS: PARAMETROS POR VALOR Y POR REFERENCIA

PARAMETROS

- Formales
 - por valor.
 - por referencia.
- Efectivos
 - si corresponde a un **parámetro formal por valor**, puede ser una de estas tres opciones:
 * un valor
 * una expresión
 * una variable
 y debe ser de **tipo asignación-compatible con el tipo del parámetro formal.**
 - si corresponde a un **parámetro formal por referencia**, debe ser siempre:
 * una variable
 y debe ser de **tipo idéntico al tipo del parámetro formal.**

Ejemplos:

```
PROCEDURE MultFrac(N1,D1,N2,D2:integer;VAR N,D:integer);
  Param. formales {(...llamadas...)}
  MultFrac (1,2,3,4, N,D);
  MultFrac (N,D, 2+2, trunc(2.3)+1, N1, D1);
  Param. efectivos
```

```
program Reflexion4; {El objetivo de este programa es hacer una traza y reflexionar sobre el pasaje de parámetros por referencia. }
var v1,v2,v3,v4:integer; {quedó compacto para que entre en una "hoja"}
procedure P3 (var R, C, Z:integer; N:integer);
  var local: integer;
  begin writeln('Entro a P3 con ', R:9, N:9);
        local:= 3; N:= local+N; R:=N; C:=0; Z:=R;
        writeln('Salgo de P3 con ', local, R:9, C:9, Z:9, N:9); end;
procedure P2 (var R, C, W:integer; N:integer);
  var local: integer;
  begin writeln('Entro a P2 con ', R:9, N:9);
        local:= 2; P3 (local,C,W,N+1); R:=local+N; C:=C+1;
        writeln('Salgo de P2 con ', local, R:9, C:9, W:9, N:9); end;
procedure P1 (var R, C, X:integer; N:integer);
  var local: integer;
  begin writeln('Entro a P1 con ', R:9, N:9);
        local:= 1; P2 (local,C,X,N+1); R:=local+N; C:=C+1;
        writeln('Salgo de P1 con ', local, R:9, C:9, X:9, N:9); end;
begin v1:=5; v4:=1; P1(v1,v2,v3,v4); write('finalizo con', v1,v2,v3,v4); end.
```

Tarea:
 Primero haga una traza en papel (bien prolija) y luego ejecute en su computadora para comparar. (Puede agregar mas "writeln"s.)

PARTE DE LA TRAZA

Estado de la traza antes de llamar a P1

reflexion4	
v1	5
v2	
v3	
v4	1

En esta página y las siguientes se muestran algunas partes de la traza del programa reflexion4. Sugerencia: realice su propia traza completa y compare.

Estado de la traza luego de ejecutar "local:= 3" en P3

reflexion4		P1		P2		P3	
v1	5	local	1	local	2	local	3
v2							
v3							
v4	1						

PARTE DE LA TRAZA

Luego de ejecutar "X:= R" en P3: observe los cambios en v2 y v3, (y local de P2)

reflexion4		P1		P2		P3	
v1	5	local	1	local	2	local	3
v2							
v3							
v4	1						

Luego de ejecutar "C:=C+1;" en P2: observe los cambios en v2 y v3

reflexion4		P1		P2	
v1	5	local	8	local	6
v2	1				
v3	6				
v4	1				

PARTE DE LA TRAZA

Luego de ejecutar "C:=C+1;" en P1: observe los cambios en v2 y v3

reflexion4		P1	
v1	9	local	8
v2	2		
v3	6		
v4	1		

Estado final de las variables globales.

reflexion4	
v1	9
v2	2
v3	6
v4	1

PREGUNTAS PARA REFLEXIONAR

- Las siguientes preguntas son sobre el programa "reflexion4", (antes de responderlas tiene que hacer la traza)
 - (fácil): ¿cuáles son parámetros por referencia?
 - La variable v2 no tiene valor al ser usada en el parámetro efectivo de la llamada a P1, ¿es un error de programación?
 - La variable v1 si tiene valor ¿es un error? ¿es mejor?
 - ¿Qué ocurriría si en P3 no se hiciera C:=0?

Resolución de Problemas y Algoritmos - 2016

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.
```

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

Resolución de Problemas y Algoritmos - 2016

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 - 2016

CONCEPTOS: BLOQUE E IDENTIFICADORES

```
PROGRAM MIPROGRAMA;
CONST Meses=12 VAR N:REAL;
FUNCTION F(R:real):real;
VAR meses: integer; BIEN
  N, PI: REAL; BIEN
  R: integer; MAL
BEGIN... sentencias... END;

PROCEDURE P(Var X:char);
CONST PI=3.14; BIEN
VAR N:REAL; BIEN
BEGIN... sentencias... END;

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

Identificadores que puede tener un BLOQUE:

1. identificadores de constantes
2. identificadores de tipos
3. identificadores de variables
4. identificadores de parámetros
5. identificadores de procedimientos
6. 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 - 2016

"EL LIMITE ESTA DADO POR SU IMAGINACIÓN"

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

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

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

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

Resolución de Problemas y Algoritmos - 2016

CONCEPTOS: TECNICA 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 - 2016

Resolución de Problemas y Algoritmos

DIVISION DE UN PROBLEMA EN SUB-PROBLEMAS

Metología: Para resolver un problema complejo se propone:

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

```

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

Resolución de Problemas y Algoritmos - 2016 13

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 - 2016 14

DIVISION DE UN PROBLEMA EN SUB-PROBLEMAS

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 - 2016 15

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 - 2016 16

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 - 2016 17

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 - 2016 18

Resolución de Problemas y Algoritmos

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

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

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

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.
    
```

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

Resolución de Problemas y Algoritmos Dr. Alejandro J. García

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 - 2016 23

CONCEPTOS: DECLARACION VS. REFERENCIA

Es importante distinguir entre:

1. 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);
1. 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 - 2016 24

EJEMPLOS DE DECLARACION 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

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

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 (Ejemplos de identificadores predefinidos: 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 - 2016 27

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 - 2016 28

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 - 2016

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, etc. (todos los elementos predefinidos provistos por Pascal).

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: todos los elementos predefinidos provistos por Pascal.

Resolución de Problemas y Algoritmos - 2016 30

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 - 2016 31

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 - 2016 32

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 - 2016 33

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 - 2016 34

EJEMPLO DE IDENTIFICADOR OCULTO

```

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

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 - 2016 35

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 qué 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 - 2016 36