

Resolución de Problemas y Algoritmos

Clase 15: Primitivas como procedimientos. Parámetros por valor y por referencia.



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

Repaso de la clase pasada sobre funciones

En Pascal una función tiene:

1. Un **nombre** (con el cual se la invocará desde una expresión).
2. **Parámetros formales** (entre los cuales estarán los datos de entrada).
3. **Tipo del resultado** (determinará en que expresión podrá ser usada)
4. **Variables locales** (que son propias de la función).
5. Sentencias (también llamado **"cuerpo"** de la función).
6. **Asignación** de una expresión al **nombre** de la función (al menos una vez). Es la forma de retornar un valor.

```

FUNCTION Potencia (Base, Exponente: integer) : integer;
{retorna base elevado a la exponente}
VAR aux, resultado: integer; // variables auxiliares
BEGIN
  resultado := 1;
FOR aux:= 1 TO Exponente DO resultado := resultado * Base;
  Potencia:= resultado;
END;
    
```

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

Conceptos: retornar el valor de una función

- En toda función, es necesario que al menos una vez se ejecute una **asignación** que en su parte izquierda tenga el **nombre** de la función.
- Es la forma que tiene la función de retornar un valor.
- Si esto no ocurre se considera un **error de programación**.
- A continuación se incluyen ejemplos correctos y otros con errores.

```

FUNCTION Cubo (N:integer):integer;
BEGIN
  Cubo:= N*N*N;
END;
    
```

CORRECTO

```

FUNCTION Cubo (N:integer):integer;
VAR aux: integer;
BEGIN
  aux:= N*N*N;
END;
    
```

MAL

INCORRECTO:
error de programación,
falta la asignación
de valor a la función.

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

Conceptos: retornar el valor de una función

- En toda función, es necesario que al menos una vez se ejecute una **asignación** que en su parte izquierda tenga el **nombre** de la función.

```

FUNCTION EsMayuscula (letra :char): boolean;
BEGIN
  IF ('A' <= letra) and (letra <= 'Z' )
  THEN EsMayuscula:=true ELSE EsMayuscula:=false
END;
    
```

CORRECTO

```

FUNCTION EsMayuscula (letra :char): boolean;
BEGIN
  IF ('A' <= letra) and (letra <= 'Z' )
  THEN EsMayuscula:=true
END;
    
```

MAL

Error de programación:
cuando no es mayúscula no se ejecuta la asignación de valor a la función.
Ejemplo de prueba: 'a'.

Importante: si existe aunque sea un solo caso de prueba para el cual la función no retorna valor, entonces hay un **error de programación**.

```

function esMayuscula (letra :char): boolean;
begin
  esMayuscula:= ('A' <= letra) and (letra <= 'Z' )
end;
    
```

CORRECTO

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

Conceptos: retornar el valor de una función

```

FUNCTION EsMayuscula (letra :char): boolean;
BEGIN
  EsMayuscula:=false;
  IF ('A' <= letra) and (letra <= 'Z' )
  THEN EsMayuscula:=true
END;
    
```

CORRECTO

```

FUNCTION dia_mes (mes,anio :integer) :integer;
BEGIN (retorna la cantidad de días de un mes)
  dia_mes:=0;
  CASE MES OF
    11,4,6,9: dia_mes :=30;
    2: IF EsBisiesto(anio) // llama a la función "EsBisiesto"
    THEN dia_mes := 29 ELSE dia_mes := 28;
    1,3,5,7,8,10,12: dia_mes :=31;
  END;
END;
    
```

CORRECTO

Puede haber muchas asignaciones que dan valor a la función. Si se ejecuta más de una, la función retornará el valor asignado por la última en ejecutarse.

Importante: el nombre de una función **NO ES UNA VARIABLE**

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

Conceptos: retornar el valor de una función

Importante: el nombre de una función no es una variable. Algunas veces se crea una confusión porque se le "asigna" un valor. Pero no puede usarse como una variable.

```

function esMayuscula (c :char): boolean;
begin
  esMayuscula:= ('A' <= c) and (c <= 'Z' )
end;
    
```

program ...
...
resultado:= esMayuscula(ch);
IF resultado = true
then ...

El nombre de una función a la izquierda del := se usa para darle valor a la función.

El nombre de una función usado en la expresión a la derecha del := es usado para llamar a la función

identificador := expresión

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) 13/05/2016

Conceptos: retornar el valor de una función

```

FUNCTION Cubo (N:integer):integer;
VAR aux,P: integer;
BEGIN
  Cubo := 1;
  FOR aux:= 1 TO 3
  DO Cubo := Cubo * N;
END;
    
```

INCORRECTO: error de programación. "Cubo" no es una variable, es el nombre de la función. Si usa el nombre de la función en una expresión entonces: ¡está llamando a la función!

MAL

identificador := expresión

El nombre de una función a la **izquierda** del := se usa para **darle valor** a la función.

El nombre de una función usado en la expresión a la **derecha** del := es usado para **llamar** a la función

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

Conceptos: retornar el valor de una función

```

FUNCTION Cubo (N:integer):integer;
VAR aux,P: integer;
BEGIN
  P:= 1;
  FOR aux:= 1 TO 3
  DO P := P * N;
  Cubo:= P;
END;
    
```

CORRECTO: Una forma de evitar el error de programación anterior es usar una variable local. "P" si es una variable y puede usarse para almacenar y usar su valor.

OK

identificador := expresión

El nombre de una función a la **izquierda** del := se usa para darle valor a la función.

El nombre de una función usado en la expresión a la **derecha** del := es usado para llamar a la función

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

Problema planteado

Problema: Escribir una función que retorne el dígito más significativo (dms) de un número entero. Realice además un programa de prueba que use esa función.

Recuerde la metodología propuesta: ejemplos, solución, algoritmo y finalmente Pascal.

Ejemplos: dms(326)=3; dms(3)=3; dms(-14)=1; dms(0)=0

Solución:

Tomar el valor absoluto N del número ingresado. Cuando N tiene un dígito, el dms(N) es N. Si N tiene más de 1 dígito, se cumple la propiedad que $dms(N) = dms(N \div 10)$. Ej: $dms(326) = dms(32) = dms(3)$.

Por lo tanto divido N sucesivamente por 10 hasta llegar a tener un número de un dígito.

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

Conceptos: parámetros por valor

```

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(num);
  writeln('el D.M.S. de', N, 'es', D);
END.
    
```

Parámetros por valor: reciben una copia de los valores de los efectivos

El parámetro "N" de la función es por valor, entonces aunque en la función le asigne nuevos valores, estos cambios no afectan a la variable "N" del programa.

Copiar la traza del pizarrón ☺

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

Conceptos: Parámetros por valor

PARÁMETROS

- Formales { por valor.
- Efectivos { si corresponde a un parámetro formal por valor, puede ser una de estas tres opciones:
 - un valor
 - una expresión
 - una variable

Ejemplos:

Parámetro formal por valor

```

FUNCTION digito_mas_significativo(N:integer): digito;
    
```

Parámetros efectivos: valores, expresiones o variables

```

{...llamadas a la función...}
d:=digito_mas_significativo(123);
d:=digito_mas_significativo(num div 2);
d:=digito_mas_significativo(num);
    
```

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

Conceptos: parámetros por valor

```

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
  D:=digito_mas_significativo(236);
  writeln('el D.M.S. de 236 es', D);
  N:=123;
  D:=digito_mas_significativo(N*7+N);
  writeln('el D.M.S. de', N*7+N,' es', D);
END.
    
```

Parámetros por valor: reciben una copia de los valores de los efectivos

El parámetro efectivo puede ser tanto una variable, un valor, o una expresión (siempre que sea de un tipo asignación compatible con el del parámetro formal).

Hacer la traza ☺

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) 13/05/2016

Conceptos: Procedimientos (procedure)

```
PROCEDURE Asteriscos( cant : INTEGER );
{ Imprime "cant" asteriscos consecutivos }
VAR i : INTEGER ;
BEGIN
    FOR i := 1 TO cant DO write('*');
END ;
```

Parámetros formales (cant)

Variables Locales (i)

- En Pascal, además de las funciones, se pueden construir primitivas como "procedimientos" (PROCEDURE).
- No tienen un tipo asociado.
- Tampoco retornan obligatoriamente un valor.
- Al igual que las funciones pueden tener parámetros y también variables locales.
- Su invocación se realiza como una sentencia.

```
Asteriscos(10) ; // imprime 10 * en pantalla
num:=5; asteriscos(num); // imprime 5 * en pantalla
read(num); asteriscos(num+SQR(num)); // imprime * en pantalla
```

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

```
PROGRAM ejemplos;
VAR tope,i: integer;
PROCEDURE Asteriscos( cant : INTEGER );
{ Imprime "cant" asteriscos consecutivos }
VAR i : INTEGER ;
BEGIN
    FOR i := 1 TO cant DO write('*');
END ;
PROCEDURE Pausa;
BEGIN (Muestra mensaje y espera ENTER)
Asteriscos(40); writeln;
Writeln ('Presione ENTER para continuar'); Readln;
END ;
BEGIN
Pausa;
writeln('ingrese tope'); readln(tope);
FOR i:= 1 to tope DO begin Asteriscos(i); writeln; end;
END.
```

Variables globales (tope, i)

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

Obs: no tiene parámetros (PROCEDURE Pausa)

Llamadas a procedimiento (Pausa, writeln('ingrese tope'), readln(tope), FOR i:= 1 to tope DO begin Asteriscos(i); writeln; end;)

Parámetro efectivo (tope)

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

Conceptos: diferencias entre...

Funciones	Procedimientos
<ul style="list-style-type: none"> Se invocan desde una expresión Al regresar de la invocación se sigue ejecutando la sentencia de la llamada. Tiene un tipo asociado Aunque no tenga parámetros devuelve un valor que se usa en la expresión que la llama. 	<ul style="list-style-type: none"> Se invocan como una sentencia. Al regresar de la invocación se ejecuta la sentencia siguiente a la llamada.

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

Cabeza (o encabezado) y Cuerpo

```
PROCEDURE Asteriscos( cant : INTEGER );
{ Imprime "cant" asteriscos consecutivos }
VAR i : INTEGER ;
BEGIN
    FOR i := 1 TO cant DO write('*');
END ;
FUNCTION Potencia(Base,Exp:integer):integer;
{calcula Base elevado a la Exp}
VAR aux,P: integer;
BEGIN
    P := 1;
    FOR aux:= 1 TO Exp DO P := P * Base;
    Potencia:= P;
END;
```

Cabeza: Declaración de parámetros formales y cuerpo de comentarios.

Cuerpo: Declaración de variables locales y cuerpo de código.

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

Ejemplo

Problema: Escriba una primitiva para multiplicar dos fracciones.

$$\frac{N1}{D1} \times \frac{N2}{D2} = \frac{N1 \times N2}{D1 \times D2} \quad \frac{1}{3} \times \frac{5}{6} = \frac{5}{18}$$

- Se propone representar una fracción con su numerador y denominador en forma separada; y construir una primitiva "multiplicar fracciones" que retorne el numerador y el denominador del resultado.
- La primitiva tendrá 4 datos de entrada: los 2 numeradores y los 2 denominadores.
- y además 2 datos de salida: el numerador (N) y el denominador (D) del resultado .
- El cálculo de N y D será el siguientes:

```
N := N1 * N2;
D := D1 * D2;
```

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

Conceptos: Parámetros por referencia

Los **parámetros formales** pueden ser:

- por valor:** sólo permiten recibir valores y se los utiliza para entrada de datos.
- por referencia:** cuando se antepone la palabra VAR. En este caso, se crea una referencia con el parámetro efectivo, y por lo tanto, permite salida de datos.

```
PROCEDURE MultFrac (N1,D1,N2,D2:integer; VAR N, D:integer);
BEGIN
    N := N1 * N2;
    D := D1 * D2;
END;
```

4 parámetros formales por valor (N1, D1, N2, D2)

2 parámetros formales por referencia (VAR N, D)

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) 13/05/2016

Parámetros por valor

Parámetros por referencia

```

PROGRAM Prueba;
VAR num1, den1, num2, den2, Nres, Dres :Integer;
PROCEDURE MultFrac (N1,D1,N2,D2:integer; VAR N, D:integer);
BEGIN
  N := N1 * N2;
  D := D1 * D2;
END;
BEGIN
write('Ingrese 2 fracciones:');
readln(num1,den1,num2,den2);
MultFrac (num1,den1,num2,den2, Nres, Dres );
writeln('Fracción resultado: ',Nres,'/',Dres);
END.
```

Sugerencia: pase a la máquina y ejecute.

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

Traza con parámetros por referencia

Prueba	
Num1	1
Den1	3
Num2	5
Den2	6
Nres	
Dres	

Ingrese 2 fracciones:
1 3 5 6

(1) Cuando comienza la ejecución de prueba, se crean 6 variables. Luego de la ejecución de `readln(num1,den1,num2,den2);` se asignan valores a las cuatro primeras variables. Considere el caso de prueba: 1 3 5 6 (que representa a 1/3 y 5/6)

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

Traza con parámetros por referencia

Prueba		MultFrac	
Num1	1	N1	1
Den1	3	D1	3
Num2	5	N2	5
Den2	6	D2	6
Nres		N	
Dres		D	

Ingrese 2 fracciones:
1 3 5 6

(2) Al llamar al procedimiento `MultFrac` se crean los parámetros por valor (N1,D1,N2,D2) y los parámetros por referencia (N y D). Luego se copian los valores para los parámetros por valor (indicado en la figura con la flecha punteada de simple punta). Además, se crea una referencia (indicada en la figura con una flecha de doble punta) entre el parámetro N y la variable `Nres` del programa, y entre el parámetro D y la variable `Dres`.

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

Traza con parámetros por referencia

Prueba		MultFrac	
Num1	1	N1	1
Den1	3	D1	3
Num2	5	N2	5
Den2	6	D2	6
Nres	5	N	
Dres	18	D	

Ingrese 2 fracciones:
1 3 5 6

Todo cambio que haga sobre un parámetro formal por referencia, afectará directamente al parámetro efectivo vinculado.

(3) Al ejecutarse las asignaciones del cuerpo del procedimiento, se modifican los parámetros N y D, y al ser estos por referencia modifican directamente a las variables `Nres`, y `Dres` que estaban en los parámetros efectivos de la llamada al procedimiento.

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

Traza con parámetros por referencia

Prueba	
Num1	1
Den1	3
Num2	5
Den2	6
Nres	5
Dres	18

Ingrese 2 fracciones:
1 3 5 6
Fracción resultado: 5 / 18

Todo cambio que haga sobre un parámetro formal por referencia, afectará directamente al parámetro efectivo vinculado.

(4) Al finalizar la ejecución del procedimiento, la memoria usada por el procedimiento se liberará. Pero dado que los valores asignados a los parámetros por referencia modificaron las variables `Nres` y `Dres`, entonces el resultado no se pierde y es mostrado en pantalla

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

Conceptos: Parámetros por valor y por referencia

PARÁMETROS

- 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**
 - si corresponde a un **parámetro formal por referencia**, debe ser siempre:
 - una **variable**

Ejemplos:

```

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

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) 13/05/2016

Conceptos: compatibilidad de tipos entre parámetros

- En los parámetros por valor, se copia el valor del parámetro efectivo y se asigna este valor al parámetro formal. Cualquier modificación que realice sobre el formal no afectará nunca al valor que tiene el efectivo.
- El valor de un parámetro efectivo pasado por valor **debe ser de asignación-compatible** al tipo del parámetro formal.
- En los parámetros por referencia se crea un referencia entre el formal y el efectivo. Todo cambio en el formal afecta y cambia al efectivo.
- Si un procedimiento o función tiene un parámetro formal pasado por referencia, entonces el tipo del parámetro formal **debe ser idéntico** al tipo del parámetro efectivo.

```

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 un poco 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.)

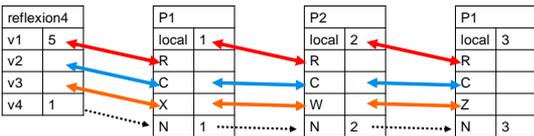
Parte de la traza

Estado de la traza antes de llamar a P1

reflexion4	
v1	5
v2	
v3	
v4	1

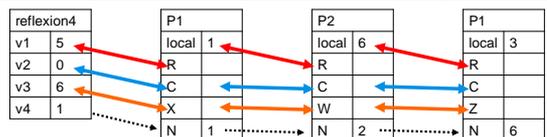
En esta página y las siguientes se muestran algunas partes de la traza del programa reflexion4. Sugerencia: realice su propia traza completa y compare.

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

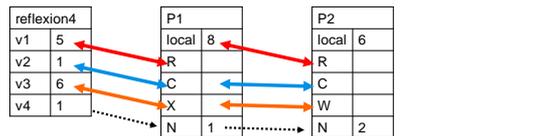


Parte de la traza

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

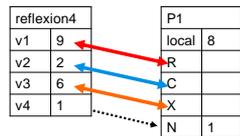


Luego de ejecutar "C:=C+1;" en P2: observe los cambios en v2 y v3



Parte de la traza

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



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 P2 no se hiciera C:=0?

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) 13/05/2016