Resolución de Problemas y Algoritmos

Clase 15 Lenguaje Pascal:
División de Problemas y uso de primitivas.

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

Tipos de datos en Pascal

(hasta ahora…) Tipo de Dato: Conjunto de valores posibles que puede tomar una variable….

¿Qué elementos de Pascal tienen un tipo asociado?
- **una variable,**
- **una expresión,**
- **un parámetro,**
- **una función.**

Tipo de Dato: define el conjunto de valores posibles que puede tomar una variable, expresión, parámetro o función; y también define las operaciones que pueden usarse sobre esos valores.

Observación importante

- **Los identificadores (constantes, tipos, variables, procedimientos, o funciones) del entorno global son visibles en todo el programa.**
- **En esta materia se permite el uso de constantes globales, procedimientos o funciones globales, y tipos globales..**

Pero…

- **En esta materia se PROHIBE el uso de VARIABLES GLOBALES (y NO LOCALES) en procedimientos y funciones.**
- **Siempre que tenga la necesidad de usarlas, tiene que usar un parámetro.**

Cabeza (o encabezado) y Cuerpo

```
PROCEDURE MultiplicarFracciones
          ( Num1, Den1, Num2, Den2: INTEGER;
            VAR NumRes,DenRes: INTEGER);
BEGIN
   NumRes := Num1 * Num2;
   DenRes := Den1 * Den2;
END;
```

cabeza / cuerpo

```
FUNCTION Potencia(Base,Exp:integer):integer;
VAR aux,P: integer;
BEGIN
  P := 1;
  FOR aux:= 1 TO Exp DO P := P * Base;
  Potencia:= P;
END;
```

cabeza / cuerpo

Llamadas entre procedimientos

```
PROGRAM LLAMADAS;

  PROCEDURE UNO (I: INTEGER);
  BEGIN
        writeln(' Estoy en uno ',I);
  END;

  PROCEDURE DOS (K: INTEGER);
  BEGIN
        writeln(' Estoy en dos' , K);
  END;

BEGIN
UNO(1);
DOS(2);
END.
```

¿Puedo llamar a DOS desde acá?

¿Puedo llamar a UNO desde acá?

¿Qué identificadores son visibles en cada uno de los procedimientos?

Dos llama a uno

```
PROGRAM LLAMADAS;

  PROCEDURE UNO (I: INTEGER);
  BEGIN
        writeln(' Estoy en uno ',I);
  END;

  PROCEDURE DOS (K: INTEGER);
  BEGIN
        writeln(' Estoy en dos ' , K);
        UNO(10);
  END;

BEGIN
UNO(1);
DOS(2);
END.
```

Uno llama a Dos: uso de "forward"

```
PROGRAM LLAMADAS;

  PROCEDURE   DOS    (K: INTEGER); FORWARD;

    PROCEDURE   UNO   (I: INTEGER);
    BEGIN
        writeln(' Estoy en uno ',I);
        DOS(10);
    END;

    PROCEDURE   DOS    (K: INTEGER);
    BEGIN
          writeln(' Estoy en dos  '  , K);
    END;
BEGIN
UNO(1);  DOS(2);
END.
```


Procedimientos y parámetros en Pascal

MultiplicarFracciones tiene 6 parámetros: 4 por valor (para recibir datos) y 2 por referencia (para devolver datos)

```
PROCEDURE  MultiplicarFracciones
          ( Num1, Den1, Num2, Den2: INTEGER;
            VAR NumRes,DenRes: INTEGER);
BEGIN
   NumRes := Num1 * Num2;
   DenRes := Den1 * Den2;
END;
```

¿Puede un procedimiento tener sólo parámetros por valor?

```
PROCEDURE BAJAR_LINEAS(cant:INTEGER);
var v:integer;
BEGIN
  FOR v:=1 TO cant DO writeln;
END;
```


Procedimientos y parámetros en Pascal

¿ Puede un procedimiento tener sólo parámetros por referencia?

```
PROCEDURE PasarAmayuscula(VAR L:char);
begin
L := chr(ord(L) - (ord('a') - ord ('A'));
end;
```

¿ Puede un procedimiento no tener parámetros?

```
PROCEDURE PAUSA;
BEGIN
  writeln(' pulse ENTER para continuar'); readln;
END;
```


¿Puede un proced. tener un único dato de salida?

```
PROCEDURE EsVocal (letra :char; var ES: boolean);
BEGIN
CASE letra OF
 'A','E','I','O','U', 'a','e','i','o','u': Es:=true;
ELSE Es:=false;
END;
```

¿Qué diferencia hay con tener una función?

```
FUNCTION EsVocal (letra :char): boolean;
BEGIN
CASE letra OF
 'A','E','I','O','U', 'a','e','i','o','u': EsVocal:=true;
ELSE EsVocal:=false;
END;
```

Funciones y parámetros en Pascal

¿Puede una función no tener parámetros?

```
FUNCTION  leer_letra:CHAR;
var aux: char;
BEGIN
 REPEAT
      read(aux)
 UNTIL (aux>='A') and (aux<='Z')
                or (aux>='a') and (aux<='z')
 leer_letra:= aux
END;
```


Funciones y parámetros en Pascal

¿ Puede una función tener parámetros por referencia?

```
FUNCTION  leer_letra(VAR error:boolean) :CHAR;
var aux: char;
BEGIN
 read(aux);
 IF (aux>='A') and (aux<='Z')
                or (aux>='a') and (aux<='z')
    then error:=FALSE
    else error:=TRUE;
 leer_letra:= aux
END;
```

Preguntas teóricas frecuentes

- ¿Cuándo un identificador es visible (referenciable) en un bloque? ¿Cuándo está oculto?
- ¿Qué elementos forman el entorno de referencia de un bloque?
- ¿Qué diferencia hay entre un parámetro por valor y uno por referencia?
- ¿Porqué es importante distinguir entre declarar un identificador y usar (referenciar) un identificador?
- ¿Qué diferencias hay entre implementar la primitiva **factorial** como una función o como un procedimiento?
- ¿Toda función puede ser transformada a un procedimiento? ¿Qué gana? ¿Qué pierde?

Problema propuesto como tarea

Realizar un programa que muestre el contenido de un archivo de enteros llamado 'mis-numeros.datos', luego solicite al usuario un elemento E, elimine todas las apariciones E, y vuelva a mostrar el contenido del archivo. Esta operación podría repetirse cuantas veces el usuario quiera.

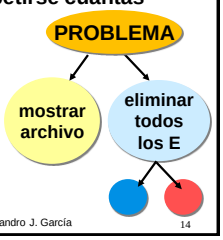
Repetir

mostrar archivo (primitiva)

solicitar elemento E

eliminar todos los E (primitiva)

hasta que el usuario lo decida



Mostrar archivo

- **Una parte del problema se resuelve con una primitiva para mostrar el archivo en pantalla.**

Algoritmo para la primitiva MostrarArchivo

Mientras no llegue al final del archivo hacer:

- **leer un elemento**
- **mostrar en pantalla el elemento y luego un separador**

problema


```
TYPE TipoElemento = Integer;
     TipoArch: FILE OF TipoElemento;
                          Con los archivos es obligatorio
                          usar parámetros por referencia
PROCEDURE mostrarA( VAR archi: TipoArch; separador:char);
Var elemento: TipoEelemento;
begin {... muestra el contenido de un archivo de números enteros ...}
   Reset(archi);
    while not eof(archi) do begin
      read(archi, elemento);
      write(elemento,' ', separador ,' ');
    end; {while}
   close(archi);
End;
```

problema

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:

a) Están declarados con el mismo identificador de tipo.

b) 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:

MAL

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

A y F1 NO SON DE TIPOS IDÉNTICOS

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:

a) Están declarados con el mismo identificador de tipo.

De esta forma ahora SI es correcto:

OK

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

Ahora A y F1 si son de tipos idénticos

```
PROGRAM pruebaMostrar;
CONST simbolo = ':' ;
TYPE TipoElemento = Integer;
     TipoArch: FILE OF TipoElemento;
VAR  F1: TipoArch;

PROCEDURE mostrarA( VAR archi: TipoArch; separador:char);
                …está más arriba…

begin
   assign(F1, 'mis-numeros.datos');
   mostrarA(F1, simbolo);
end.
```

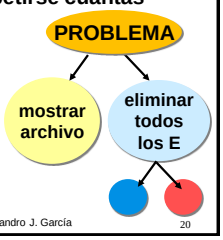
problema

Problema propuesto como tarea

Realizar un programa que muestre el contenido de un archivo de enteros llamado 'mis-numeros.datos', luego solicite al usuario un elemento E, elimine todas las apariciones E, y vuelva a mostrar el contenido del archivo. Esta operación podría repetirse cuantas veces el usuario quiera.

Repetir
- **mostrar archivo (primitiva)**
- **solicitar elemento E**
- **eliminar todos los E (primitiva)**

hasta que el usuario lo decida




```
PROGRAM pruebaMostrar;
CONST simbolo = ':' ;
TYPE TipoElemento = Integer; TipoArch: FILE OF TipoElemento;
VAR  F1: TipoArch;  Elem: TipoElemento;
PROCEDURE mostrarA( VAR archi: TipoArch; separador:char);
                …está más arriba…

PROCEDURE eliminarE( VAR archi:TipoArch; Ele: TipoElemento);
VAR temp: Tarchi;
begin
   assign(temp, 'temporario.auxiliar');
   CopiarSinLosEle(archi,Ele,temp);
   CopiarArchivoCompleto(temp,archi);
end;
```

TAREA: Complete el progama con los procedimientos que faltan.