



Resolución de Problemas y Algoritmos

Clase 17: Estrategias de resolución de problemas basadas en el uso de primitivas y división del problema



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: Tipos de datos en Pascal

¿Qué elementos de Pascal tienen un tipo asociado?

- una variable,
- una constante,
- una expresión,
- un parámetro,
- una función.

Tipo de Dato: define el conjunto de valores posibles que puede tomar un elemento de un programa (variable, expresión, parámetro o función), define las operaciones que pueden usarse sobre esos valores, y define una representación interna para su almacenamiento en memoria.

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

Program Ejemplo;
TYPE TipoElemento = Integer;
 TipoArch: FILE OF TipoElemento; Con archivos es obligatorio usar parámetros por referencia
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}

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

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 4

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**:

```

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 5

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 es **correcto**:

```

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

BIEN

Ahora A y F1 si son de tipos idénticos

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) 19/05/2016

Reflexión sobre temas vistos

Los siguientes temas, vistos en clases anteriores, están todos relacionados y son muy **importantes**:

- Diseño de la solución dividiendo el problema
- Funciones y procedimientos en Pascal
- Parámetros (por valor y por referencia)
- Entorno de referencia de los identificadores
- Visibilidad – Identificadores locales, globales, etc.

¿Alguna pregunta?

Esta **importancia** va más allá de RPA, en su vida profesional es muy probable que trabaje **en un equipo**.

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

Sobre el trabajo profesional futuro

Sin importar la dimensión del problema, existe una gran responsabilidad en la correcta resolución del mismo.

Considere por ejemplo las consecuencias negativas de una incorrecta resolución de un problema de pequeña escala como la *validación de la identidad del piloto del avión que usted está por abordar*, o la *validación de acceso a transferencias de su propia cuenta bancaria*.

Un sistema de gran escala (como *reserva y venta de pasajes*) estará formado por un conjunto de soluciones a problemas de pequeña escala (como controlar que una fecha sea correcta).

Permitir el ingreso y trabajar luego con fechas incorrectas puede tener malas consecuencias.

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

Primitivas en el desarrollo de software

- Si trabajo en grupo y tengo a cargo una parte, debo tener una manera de compartir esa parte de forma que los demás puedan usarla como una primitiva sin necesidad de conocer los detalles de cómo está hecha.
- Si trabajo solo y tengo que abordar un problema que no es trivial (el cual puede ser dividido en partes), y además, algunas de esas partes pueden "re-utilizarse", entonces también necesito una forma de implementar una primitiva.
- Al resolver un problema en el futuro, no solo dispondré de las primitivas predefinidas, si no también las que he construido antes o las de mis compañeros de trabajo.

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

Conceptos: estrategias "top-down" y "bottom-up"

Estrategia Top-down:

Por división del problema principal en subproblemas más simples hasta llegar a problemas que no necesitan dividirse.

Sub-problemas

Estrategia Bottom-up:

Por composición, resolviendo primero los subproblemas más simples hasta llegar a solucionar al problema principal.

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

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
- 3) **para cada parte implementar primitivas** en Pascal: como funciones o procedimientos

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

Problema propuesto

La universidad quiere premiar a sus buenos alumnos que quieran ir a visitar la Feria del Libro. Para aquellos alumnos que lo soliciten y cumplan los requisitos se les pagará la inscripción y el viaje.

Los requisitos son: ser alumno regular con un promedio mayor a 6, y en el año anterior: haber cursado 3 o más materias, haber aprobado por lo menos dos materias y no tener ninguna materia desaprobada. Se desea desarrollar una aplicación que a partir del archivo "inscriptos.alu" con los números de LU de los inscriptos, genere otro archivo "cumplen.alu" con los inscriptos que cumplen los requisitos.

Para esta aplicación se dispone del archivo "regulares-más-de6.alu" con los LU de los que son regulares y tienen promedio mayor a 6. También se tienen los archivos "cursadas.alu", "aprobadas.alu" y "desaprobadas.alu" que contienen pares (LU - código materia), con las cursadas, aprobadas y desaprobadas del año anterior.

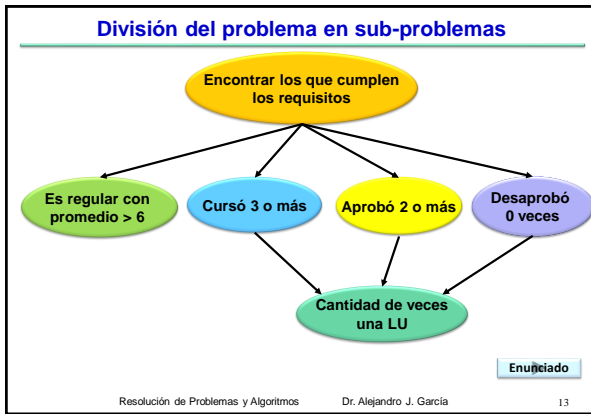
Se requiere usar archivos de texto para todos los casos. EJEMPLO

Se asume que no hay errores de carga en los archivos. ESQUEMA

ALGORITMO

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) 19/05/2016



Ejemplo / Caso de prueba

- Estos dos archivos son secuencias de LU
inscritos.alu: 55055 89100 99099 88008 77077 95095 81118
regulares-más-de6.alu: 89100 88008 77077 95095 81118
- Estos tres archivos son secuencias de pares LU código_materia
 cursadas.alu: 89100 11 88008 11 77077 22 95095 22 81118 33
 89100 33 88008 33 77077 44 89100 44 88008 44 77077 23
 aprobadas.alu: 88008 11 95095 22 81118 33 89100 33 88008
 33 77077 44 89100 44 88008 44 77077 23
 desaprobadas.alu: 55055 11 99099 22 88008 44
- Con estos datos quedan elegidos las siguientes LU:
 cumplen.alu: 89100 77077

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

Algoritmo general

Abrir "inscritos" para leer
 Crear "cumplen" para escribir
 Mientras hay elementos en "inscritos" hacer:

- leer una LU del archivo de inscritos
- Si pertenece (LU, regulares_mas_de6) y cantidad(cursadas, LU) ≥ 3 y cantidad(aprobadas, LU) ≥ 2 y cantidad(desaprobadas, LU) = 0 entonces: agregar esa LU al archivo "cumplen"

cerrar "inscritos"
 cerrar "cumplen"

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

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

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

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) 19/05/2016

Procedimientos y parámetros en Pascal

MultiplicarFracciones tiene 6 parámetros formales: 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 {multiplica dos fracciones}
  NumRes := Num1 * Num2;
  DenRes := Den1 * Den2;
END;
    
```

Los parámetros por referencia ¿siempre van al final? Respuesta: no.

¿Puede un procedimiento tener sólo parámetros por valor?

```

PROCEDURE BAJAR_LINEAS(cant: INTEGER);
var v: integer; {Deja "cant" líneas en blanco en la pantalla}
BEGIN
FOR v:=1 TO cant DO writeln;
END;
    
```

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

Procedimientos y parámetros en Pascal

¿ Puede un procedimiento tener sólo parámetros por referencia?

```

PROCEDURE PasarAmayuscula(VAR L: char);
begin {Si recibe una minúscula, cambia el valor por mayúscula}
  if (L >= 'a') and (L <= 'z')
  then L := chr(ord(L) - (ord('a') - ord('A')));
end; {Si no recibe una minúscula, el valor recibido no se cambia}
    
```

¿ Puede un procedimiento no tener parámetros?

```

PROCEDURE PAUSA;
BEGIN {muestra mensaje y espera por un ENTER del usuario}
  writeln(' pulse ENTER para continuar'); readln;
END;
    
```

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

Procedimientos vs. Funciones

¿Puede un procedimiento tener un único dato de salida?

```

PROCEDURE EsVocal (letra : char; var Es: boolean);
BEGIN {primitiva para identificar letras vocales}
  CASE letra OF {si letra es vocal retorna true en ES}
    'A','E','I','O','U', 'a','e','i','o','u': Es:=true;
    ELSE Es:=false; END; {o false en caso contrario}
END;
    
```

¿Qué diferencia hay con tener una función como esta?

```

FUNCTION EsVocal (letra : char): boolean;
BEGIN {primitiva para identificar letras vocales}
  CASE letra OF {si letra es vocal la función retorna true}
    'A','E','I','O','U', 'a','e','i','o','u': EsVocal:=true;
    ELSE EsVocal:=false; END {o false en caso contrario}
END;
    
```

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

Funciones y parámetros en Pascal

¿Puede una función no tener parámetros?

```

FUNCTION leer_letra: CHAR;
var aux: char; {Esta función sin parámetros lee del buffer }
BEGIN {hasta que el carácter leído sea una letra y la retorna}
  REPEAT
    read(aux)
  UNTIL (aux>='A') and (aux<='Z') or (aux>='a') and (aux<='z')
  leer_letra:= aux
END;
    
```

Llamada a la función:
ch:=leer_letra;

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

Funciones y parámetros en Pascal

¿ Puede una función tener parámetros por referencia?

```

TYPE TipoElemento= integer; ARCHI: FILE OF TipoElemento;

FUNCTION cantidad_elementos (VAR A: ARCHI): integer;
var aux: TipoElemento; cant: integer;
BEGIN {retorna la cantidad de elementos de un archivo}
  cant:= 0;
  reset(A);
  while not eof(A) do { por cada elemento leído suma uno}
    begin read(A,aux); cant:=cant+1; end;
  cantidad_elementos := cant;
  close(A);
END;
    
```

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

Funciones y parámetros en Pascal

¿ Puede una función tener parámetros por referencia?

```

{Función que retorna la cantidad de líneas de un archivo de texto}
FUNCTION cantidad_lineas(VAR A: text): integer;
var cant: integer;
BEGIN
  cant:= 0;
  reset(A);
  while not eof(A) do
    begin readln(A); cant:=cant+1; end;
  cantidad_lineas := cant;
  close(A);
END;
    
```

Recuerde que "text" es un tipo predefinido y estructurado (archivo)

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) 19/05/2016

¡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.



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) 19/05/2016