



## Resolución de Problemas y Algoritmos

### Clase 16: Estrategias de resolución de problemas basadas en 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

¡Gracias a todos! ☺



**GIRSU UNS**  
Gestión Integral de Residuos Sólidos Urbanos

El plan se puso en marcha a mediados de marzo de 2015, con la institución funcionando a pleno, de modo tal de poder hacer los ajustes sobre la marcha y con el mayor flujo de alumnos. La recolección y reciclado, inicialmente la hace un servicio contratado que se lleva los residuos y paga por el material que recibe.

**Resultados:**

Fecha	Papel blanco (kg)	Papel y cartón (kg)	Envases - PET (kg)
7/4	800	250	100
15/4	700	300	50
5/5	1000	200	100
<b>Total Reciclado (1 mes)</b>	<b>2500</b>	<b>750</b>	<b>250</b>

**En un mes, se han reciclado 3,3 toneladas de papel y cartón lo que equivale aproximadamente al uso de 70 árboles y 250 kg de envases lo que equivale a unas 5000 botellas de 500 ml**

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

### Conceptos: diferentes clases de errores en programas

- Error de compilación:** es un error detectado por el compilador al momento que se está realizando la compilación de un código fuente, por eso también se llama *error en tiempo de compilación*.
- Error de ejecución:** ocurre cuando al momento de la ejecución del programa hay una situación anormal, y generalmente provoca que la ejecución se corte abruptamente. También se lo llama *error en tiempo de ejecución*. Ej: intentar abrir un archivo que no existe.
- Error lógico:** también llamado **error de programación (bug)**, es un error en la lógica del algoritmo o programa el cual causa que no se resuelva correctamente la tarea que debe hacer el programa.


Una tarea importante de un profesional es lograr que los programas no tengan errores. Para ello deben probarse los programas (*testing*) con los suficientes casos de prueba; y de encontrar un mal funcionamiento se debe buscar y eliminar los errores (*debugging*).

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

### Conceptos: debugging (depuración)

**Debugging** (depuración): se refiere al proceso metodológico de buscar y reducir el número de errores o defectos (*bugs*) de un programa, con el objetivo de lograr que el programa tenga el comportamiento esperado.

**Origen del término:** Los términos "bug" y "debugging" son atribuidos a la almirante [Grace Murray Hopper](#). En 1947 mientras trabajaba en un [Mark II](#) ella encontró una polilla atrapada en un relé impidiendo las operaciones de dicha computadora. Grace comentó que al sacar el "bug" (bicho) habían hecho un "debugging" (des-bichado) al sistema.

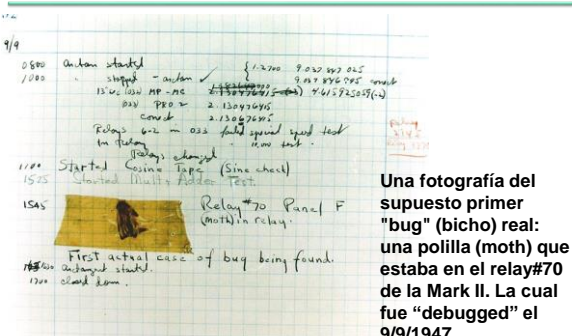


Grace Hopper frente al teclado de UNIVAC I (en 1960)

<http://en.wikipedia.org/wiki/Debugging>

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

### El primer "bug" en un programa



**Una fotografía del supuesto primer "bug" (bicho) real: una polilla (moth) que estaba en el relay#70 de la Mark II. La cual fue "debugged" el 9/9/1947.**

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

### Búsqueda de errores

- Realizar una traza es una herramienta de verificación y búsqueda de errores muy simple y muy útil (sobre todo en un examen).
- Pero a veces, al ejecutar un programa, este no realiza lo esperado.
- En los programas que tienen mucha salida en pantalla es bastante más fácil detectar en que parte está el error.
- Pero hay programas que tienen poca (o ninguna) salida en pantalla. Por ejemplo, considere un programa que lee de uno o más archivos y escribe el resultado en otro archivo. Aunque no tenga errores de compilación, podría ocurrir que al realizar ejecuciones de prueba ocurra algún **error de ejecución** y el programa se detenga abruptamente sin poder saber donde se produjo el error. También podría ocurrir que tenga **errores de programación** y en las ejecuciones de prueba dé resultados incorrectos. ¿Cómo encuentro los errores de ejecución o de programación?

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**

**Código que ayuda en la búsqueda de errores**

**Ejemplo:** recuerda el programa de la clase pasada, donde había que seleccionar de un archivo de inscriptos, alumnos que cumplieran ciertos requisitos, aquí una parte:

```
reset(inscriptos); rewrite(cumplen);
while not eof(inscriptos) do
begin
  read(inscriptos,LU);
  if pertenece(LU, reg_mas_d6)
  and (cantidad(cursadas,LU) >= 3)
  and (cantidad(aprobadas,LU) >= 2)
  and (cantidad(desaprobadas,LU) = 0)
  then write(cumplen,LU, ' ');
end;
```

Suponga que en una ejecución de prueba, el programa falla y no genera correctamente el archivo. ¿Cómo busco el error?

**Código que ayuda en la búsqueda de errores**

Si se está realizando una verificación (*testing*), o el programa funciona mal pero no se encuentra el lugar donde está el error, se puede hacer algo simple pero bastante poderoso que ayudará al *debugging*:

- agregar en ciertas partes convenientes del programa unos "writeln" que permitirán, para cada caso de prueba, mostrar en pantalla el camino real de la ejecución que sigue el programa.
- además, esos "writeln" pueden ir mostrando el contenido de las variables más importantes de cada procedimiento o función.

Por ejemplo:

```
writeln( 'Ingresé a función pertenece, buscando elemento', buscado);
```

```
.....
writeln( 'Salí de la función pertenece, con', resultado);
```

**Código que ayuda al "debugging"**

```
function pertenece (buscado: integer; var archivo: text):boolean;
{retorna true si el elemento buscado está en el archivo de texto}
var elemento: integer; encontre: boolean;
begin
  writeln(' DEBUG: buscando ',buscado); //borrar luego de verificar
  reset(archivo); encontre:=false;
  while not eof(archivo) and not encontre do
  begin
    read(archivo, elemento);
    encontre:=elemento=buscado;
  end;
  pertenece:= encontre;
  close(archivo);
  writeln(' DEBUG: ¿encontrado? ',buscado,'->',encontre); //borrar luego
end;
```

Estos "writeln();" mostrarán en pantalla para cada caso de prueba la ejecución real que sigue el programa y contenido de las variables más importantes de cada procedimiento o función.

**Código que ayuda al "debugging"**

```
function cantidad (var archivo: text; lu: integer):integer;
{retorna la cantidad de veces que está una LU en un archivo de pares LU materia}
var elemento,materia, cant: integer;
begin
  writeln(' DEBUG: buscando LU ',lu); // borrar luego de verificar
  reset(archivo); cant:=0;
  while not eof(archivo) do
  begin
    read(archivo, elemento);
    if elemento = lu then cant:=cant+1;
    read(archivo, materia);
  end;
  cantidad:=cant;
  close(archivo);
  writeln(' DEBUG:cantidad ',LU,'->', cantidad); //borrar luego
end;
```

Estos "writeln();" mostrarán en pantalla para cada caso de prueba la ejecución real que sigue el programa y contenido de las variables más importantes de cada procedimiento o función.

**Ejemplo de trabajo**

**Problema:** Escriba un programa para calcular la suma hasta el término K-ésimo, donde en cada término de posición N, el numerador es 2<sup>N</sup> y el denominador N! Por ejemplo para K = 6

$$\text{suma} = \frac{2}{1} + \frac{4}{2} + \frac{8}{6} + \frac{16}{24} + \frac{32}{120} + \frac{64}{720}$$

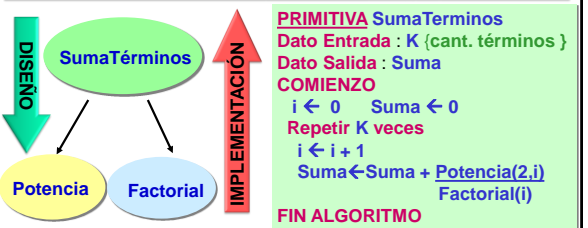
K=6

**Solución:** sumar (2<sup>N</sup> / N!) desde N=1 hasta N=k  
 En Pascal, ¿tengo una primitiva para N! (factorial)?  
 ¿tengo una primitiva para 2<sup>N</sup> (potencia) ?

**Como programador puedo construir nuevas primitivas.**

**Técnica:** Para escribir el algoritmo y el programa se sugiere descomponer el problema en sub-problemas.

**Descomposición del problema y primitivas**

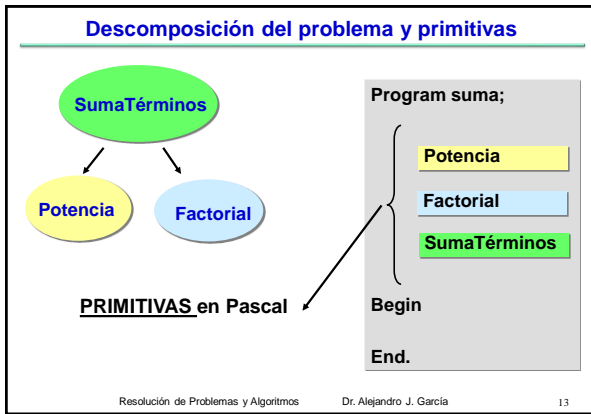


```
PRIMITIVA SumaTerminos
Dato Entrada : K {cant. términos }
Dato Salida : Suma
COMIENZO
i <- 0 Suma <- 0
Repetir K veces
i <- i + 1
Suma <- Suma + Potencia(2,i)
Factorial(i)
FIN ALGORITMO
```

```
PRIMITIVA Potencia
Datos Entrada : Base, Expo
Datos Salida : Pot
... calcula Base a la Expo...
```

```
PRIMITIVA Factorial
Datos Entrada : N {natural}
Datos Salida : Factorial de N
... calcula factorial de N...
```

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



```
PROGRAM sumaK;
{resuelve el problema de sumar los primeros "K" términos de la serie 2N / N! }
VAR cant: integer; resultado:real;

FUNCTION factorial (N:integer): integer;
{retorna el factorial del parámetro N}      {TAREA: COMPLETAR el cuerpo}

FUNCTION potencia (B,E:integer):integer;
{retorna B elevado a la E}                {TAREA: COMPLETAR el cuerpo}

PROCEDURE SumaTerminos ( K:integer; VAR suma: real );
VAR i: integer; {suma los primeros "K" términos de la serie 2N / N! }
BEGIN { el parámetro por referencia "suma" acumula y retorna el resultado}
  suma:= 0; {el parámetro por valor "K" indica cuantas iteraciones hacer}
  FOR i:=1 TO K DO suma := suma+ potencia(2,i) / factorial(i) ;
END;
BEGIN
writeln('ingrese cantidad de términos a sumar'); readln(cant);
sumaTerminos(cant, resultado);
writeln('la suma es: ', resultado);
END.
```

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

### Tarea (muy importante) propuesta

- Complete el programa sumaK y pase en la máquina.
- Realice algunas trazas en papel y luego ejecute para ver como funciona en la máquina.
- Puede poner algunos "writeln" para ver como se van llamando las funciones y como se modifican las variables. Por ejemplo:
 

```
writeln('ingreso a la función potencia');
writeln('salgo de sumaTermino con:', suma, tope);
```
- "sumaTerminos" también podría haber sido una función. Como tarea adicional haga otra versión del programa con una función para "sumaTerminos".

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

### Reflexión

En clase estuvimos reflexionando sobre las diferencias, ventajas y desventajas entre:

- a) ubicar a las funciones **factorial** y **potencia** en el entorno global,
- b) ubicar a las funciones **factorial** y **potencia** en el entorno local al procedimiento "SumaTerminos".

Analice las diferencias y reflexiones sobre las ventajas de cada una de las opciones.

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

### Observación sobre la traducción al castellano

El nombre en Inglés para los parámetros en Pascal es:

- **formal parameters**, traducido como "parámetros formales"
- **actual parameters** que se puede traducir al castellano como
  - 1) **parámetros reales**
  - 2) **parámetros efectivos** (yo prefiero esta última para no confundir con un parámetro de tipo real)

**IMPORTANTE** no hay que confundir "actual" en inglés con "actual" en castellano que se escriben igual (se pronuncian diferente) y son dos cosas diferentes.

Es bastante común ver mal traducido *actual parameters* como "parámetros actuales" pero no es correcto (ver a continuación).

<https://translate.google.com/?hl=en/es/The%20actual%20parameters%20are%20in%20the%20function%20call>

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

### Observación sobre "actual parameters"

**EN CASTELLANO:**  
**actual** *adj.* Presente. Activo, que obra. Que existe en el tiempo en que se habla.

**TRADUCCIÓN A INGLÉS:**  
**actual** *ADJ* 1. (= de ahora) [situación, sistema, gobernante] → current, present; [sociedad] → contemporary, present-day; [moda] → current, modern  
 en el momento actual → at the present moment

-----

**EN INGLÉS:**  
**actual** *adj* 1. existing in reality or as a matter of fact  
 2. real or genuine

**TRADUCCIÓN A CASTELLANO:**  
*The actual number is much higher than that* → El número real es mucho más grande  
*The film was based on actual events* → La película estaba basada en hechos reales  
*Let's take an actual case/example* → Tomemos un caso/ejemplo concreto

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

### IMPORTANTE: prohibición en RPA

En una función o procedimiento definida por el programador:

- está permitido usar **constantes, tipos, procedimientos, y funciones** que fueron declarados en su entorno local, global o en el entorno no-local.
- también puedo usar variables o parámetros del entorno local.

Sin embargo,...

- En RPA, en los procedimientos y funciones, **se PROHIBE y será considerado un error** el uso de **variables globales, o variables declaradas en un entorno no-local.**
- El uso de variables declaradas en otros entornos que no sea el local no es una buena pauta de programación y **lleva a cometer errores de programación que son muy difíciles de encontrar.**
- Vea los ejemplos que hay a continuación. Páselos a la máquina y reflexione sobre ellos.



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

```

program incorrecto;
var i: integer;
Procedure Linea;
begin
  For i:=1 to 25 DO write('-');
  writeln;
end; {linea}
MAL: usa una
variable global
Begin
linea;
write(' Ingrese un nro: '); Readln(i);
linea;
writeln('raíz de ',i,' es', SQRT(i):2:0);
end.
```

**En el programa incorrecto:**

- La variable "i" es usada en "linea" como global.
- Esto afecta al resultado esperado ya que "linea" modifica la variable i del programa.

```
-----
linea
Ingrese un nro: 16
-----
raíz de 25 es 5.00
```

- Solución:** crear una variable "i" local.
- Además, es evidente que la variable global "i" debería tener un nombre más significativo (vea el programa a continuación)**

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

```

program ahora_correcto;
var nro: integer;
Procedure Linea;
var i: integer;
begin
  For i:=1 to 25 DO write('-');
  writeln;
end; {linea}
Begin
linea;
write(' Ingrese un nro: '); Readln(nro);
linea;
writeln('raíz de ',nro,' es', SQRT(nro):2:0);
end.
```

variable global, solo usada en código del programa

variable local, solo usada en "linea"

```
-----
Ingrese un nro: 16
-----
raíz de 16 es 4.00
```

• Observe que ahora no hay error, ya que el procedimiento línea no modifica la variable del programa.

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

### IMPORTANTE: prohibición en RPA

En RPA, en los procedimientos y funciones, **se PROHIBE y será considerado un error** el uso de **variables globales, o variables declaradas en un entorno no local.**




• **Ayuda:** siempre que surja la necesidad de usar variables globales es porque tendría que usar o una constante, o una variable local o un parámetro.

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

### 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.

**Problema**



Repetir

mostrar archivo (primitiva)

solicitar elemento E

eliminar todos los E (primitiva)

hasta que el usuario lo decida

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

```

Program Eliminar; {una posible solución: completar lo que falta}
TYPE TipoElemento = Integer; TipoArch: FILE OF TipoElemento;
VAR F1: TipoArch; Elem: tipoElemento;

PROCEDURE mostrarArchivo ( VAR archi: TipoArch; separador: char);

PROCEDURE EliminarDeArchivo(E: TipoElemento; VAR original: TipoArch);
var auxiliar: TipoArch;

PROCEDURE pasar ( E: TipoElemento; VAR original, auxiliar: TipoArch);
{pasa todos los elementos que son distintos de E al archivo auxiliar}

PROCEDURE copiar (VAR auxiliar, original: TipoArch);
{copia todos los elementos del archivo auxiliar al original}

begin
assign(F1, 'mis-numeros.datos');
repeat
  mostrarArchivo(F1, ', ');
  writeln(' Ingrese elemento a eliminar'); readln(Elem);
  EliminarDeArchivo(Elem, F1);
until ..... // completar esta parte
end.
```

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