

RESOLUCIÓN DE PROBLEMAS Y ALGORITMOS

CLASE 10

Resolución de Problemas

0 1 1 0 0
 1 0 0 1 1
 1 0 1 1 0
 0 1 1 1 0
 0 1 1 0 0
 1 0 0 1 1
 1 0 1 1 0
 0 1 1 1 0
 1 0 0 1 1
 1 1 1
 0 0
 1

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

- PANEL DE GRADUADOS
 - 19/09 – Lunes!
 - 18hs.
 - Aula 4 de Palihue
- SE VIENE EL PARCIAL!!
 - Miércoles 28/09.

0 1 1 0 0
 1 0 0 1 1
 1 0 1 1 0
 0 1 1 1 0
 0 1 1 0 0
 1 0 0 1 1
 1 0 1 1 0
 0 1 1 1 0
 1 0 0 1 1
 1 1 1
 0 0
 1

Resolución de Problemas y Algoritmos - 2016

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:** es una situación anormal que ocurre al momento de la ejecución del programa, y generalmente provoca que la ejecución se corte abruptamente. También se lo llama *error en tiempo de ejecución*. Ejemplo: intentar leer de un archivo vacío.
- **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.

Resolución de Problemas y Algoritmos - 2016

CONCEPTOS: DIFERENTES CLASES DE ERRORES EN PROGRAMAS

- Un **profesional** debe lograr que los programas no tengan errores.
- Para ello **deben probarse los programas** (en Inglés **testing**) con los suficientes casos de prueba.
- Si se detecta un mal funcionamiento con algún caso de prueba, entonces se deben **buscar y eliminar los errores (debugging)**.

0 1 1 0 0
 1 0 0 1 1
 1 0 1 1 0
 0 1 1 1 0
 0 1 1 0 0
 1 0 0 1 1
 1 0 1 1 0
 0 1 1 1 0
 1 0 0 1 1
 1 1 1
 0 0
 1

Resolución de Problemas y Algoritmos - 2016

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.

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

- El término "**debugging**" es atribuido a la Almirante Grace Murray Hopper.
- En 1947 mientras trabajaba en una Mark II encontró una **polilla** atrapada en un relé que impedía las operaciones de dicha computadora.
- Grace dijo que al sacar el "bug" (**bicho**) habían hecho un "debugging" (**desbichado**) del sistema.
- En una entrevista Grace comentó que el término "bug" como sinónimo de error ya era usado en la jerga (ver más sobre esto en [software-bug](#)).

Resolución de Problemas y Algoritmos - 2016

EL PRIMER "BUG" EN UN PROGRAMA

9/9

0800 Anton started
 1000 support - anton ✓ { 12700 9.000 497 025 count
 1300 low MP-MC 2.13047415 7.017 896 095 count
 020 PRO 2 2.13047415
 count 2.13047415
 Relays are in 022 field speed test
 in relay 11.000 inch.

1100 Started cosine tape (Sine check)
 1525 Started Multi-Header Test.

1545  Relay #70 Panel F (moth) in relay.

First actual case of bug being found
 1700 anton started.
 1700 clock done.

Una fotografía del supuesto primer "bug" (bicho) real: una polilla (moth) que estaba en el relé #70 de la Mark II. La cual fue "debugged" el 9/9/1947 a las 15:45hs.

Resolución de Problemas y Algoritmos - 2016

Resolución de Problemas y Algoritmos

EJEMPLO PROPUESTO

Problema: escriba un programa que busque el menor elemento en un archivo de enteros "numeros.enteros" el cual ya fue creado.

Algoritmo buscar el menor

Abrir el archivo para leer
 Leer el primer elemento
 Guardar el primero como el menor encontrado hasta ahora
 Repetir mientras no sea fin de archivo (EOF)
 Leer un elemento nuevo del archivo
 Si el elemento recién leído es menor al menor encontrado hasta ahora entonces
 el elemento recién leído es el nuevo menor encontrado
 Mostrar el elemento menor
 Cerrar el archivo.
fin.

Ejemplos significativos / casos de prueba
 Archivo = 1, 2, 3 Archivo = 8, 3, 4, 3, 4
 Archivo = vacío Archivo = 8
 Archivo = 3, 2, 1

Realizaremos el programa en el pizarrón

```
PROGRAM buscar_menor;
TYPE
mis_nums = FILE OF integer; //tipo nuevo creado por el constructor
VAR
numeros: mis_nums; //manejador del archivo de enteros
menor, leído: integer; //el menor encontrado y el último leído
BEGIN
assign(numeros, 'numeros.enteros'); //vincula nombre con el
manejador
reset(numeros); //abre el archivo para leer
IF eof(numeros) THEN
writeln( ' archivo vacío, no hay menor' )
ELSE
BEGIN
read(numeros,menor); //guardo el primero como el menor
encontrado
//ahora miro el resto del archivo para
ver si hay otro menor...
WHILE NOT eof(numeros) DO //mientras no sea el fin del
archivo
BEGIN
... //el resto del programa fue desarrollado en el pizarrón
```

TAREA PROPUESTA

Problema: escriba un programa que indique cual es y cuantas veces aparece el menor elemento en un archivo de enteros "numeros.enteros" el cual ya fue creado.

Ejemplos significativos / casos de prueba

Archivo = -9, 2, 3, -9, 2, -9 el menor es el -9 y está tres veces
 Archivo = 8, 8, 4, 3, 4, 3 el menor es el 3 y está dos veces
 Archivo = vacío no existe el menor
 Archivo = 8 el menor es el 8 y está una vez
 Archivo = 3, 3, 3, 1 el menor es el 1 y está una vez

Algoritmo cuantas veces el menor

Tarea para practicar
 (no deje de mostrarle a un docente su propuesta)
fin.

Resolución de Problemas y Algoritmos - 2016

Problema: escriba un programa para copiar el contenido de "mis-numeros.datos", a otro archivo "mis-numeros.copia".

```
PROGRAM incorrecto; {VERSION INCORRECTA}
TYPE archi_enteros = FILE OF integer;
VAR F1, F2: archi_enteros;
BEGIN
assign(F1, 'mis-numeros.datos');
assign(F2, 'mis-numeros.copia');
F1:=F2;
close(F2); close(F1);
END.
```

Esto no copia los archivos
 Esta asignación hace que los dos manejadores se refieran al mismo archivo: "mis-numeros.copia"

Resolución de Problemas y Algoritmos - 2016

Problema: escriba un programa para copiar el contenido de "mis-numeros.datos", a otro archivo "mis-numeros.copia".

```
PROGRAM copias; {VERSION CORRECTA}
TYPE
archi_enteros = FILE OF integer;
VAR
F1, F2: archi_enteros; elemento: integer;
BEGIN
assign(F1, 'mis-numeros.datos');
reset(F1); //F1 para lectura
assign(F2, 'mis-numeros.copia');
rewrite(F2); //F2 para escritura
while not eof(F1) do
begin //mientras no se acabe F1
read(F1, elemento); //leo de a un elemento de F1
write(F2, elemento); //escribo cada elemento
leído en F2
end;
close(F1); close(F2);
END.
```

Resolución de Problemas y Algoritmos - 2016

ELIMINAR UN ELEMENTO DE UN ARCHIVO

Problema: escriba un programa que solicite al usuario un elemento y elimine todas las apariciones de ese elemento del archivo de enteros "numeros.enteros"

Ejemplos significativos / casos de prueba

Archivo = 1, 2, 3 eliminar el 2 Archivo = 1, 3
 Archivo = 1, 2, 3 eliminar el 5 Archivo = 1, 2, 3
 Archivo = 8, 3, 4, 3, 4 eliminar el 4 Archivo = 8, 3, 3
 Archivo = vacío eliminar el 2 Archivo = vacío
 Archivo = 8 eliminar el 8 Archivo = vacío

Realizaremos el algoritmo y programa en el pizarrón

Resolución de Problemas y Algoritmos - 2016

Resolución de Problemas y Algoritmos

ELIMINAR UN ELEMENTO DE UN ARCHIVO

Problema: escriba un programa que solicite al usuario un elemento y **elimine todas las apariciones de ese elemento** del archivo de enteros "numeros.enteros"

Algoritmo eliminar un elemento

Pedir y leer el elemento a eliminar

Realizaremos el algoritmo y programa en el pizarrón

Abrir el archivo original para leer, y abrir un archivo auxiliar para escribir

mientras no sea fin de archivo original

Leer un elemento del original

Si el elemento recién leído es distinto al que quiero eliminar entonces escribir el recién leído al archivo auxiliar

Cerrar ambos archivos

Como fue explicado en el problema anterior

Abrir auxiliar para leer y original para escribir

Copiar todos los elementos del auxiliar al original.

fin.

Resolución de Problemas y Algoritmos - 2016 13

PROBLEMA PROPUESTO

Problema: Escribir un programa para determinar si un número natural N es primo.

Definición: un número es primo si es un entero positivo mayor que 1, que es divisible solamente por si mismo y la unidad.

Ejemplos (futuros casos de prueba):

2, 3, 7, 11 y 2003: son primos

4 no es primo es divisible por 2

2001 no es primo, es divisible por 3

121 no es primo, es divisible por 11

Una solución: si N tiene un divisor distinto de N o 1 entonces NO es primo, de lo contrario es primo. Entonces para hacer un algoritmo puedo asumir que es primo hasta que se demuestre lo contrario (esto es, voy viendo uno a uno si encuentro un divisor)

ALGORITMO PARA "ESPRIMO"

ALGORITMO EsPrimo

COMIENZO

leer el número N

SI ($N > 1$) **ENTONCES**

EsPrimo \leftarrow Verdadero { asumo que es primo }

SINO EsPrimo \leftarrow Falso

Divisor \leftarrow 2 { el primer divisor a considerar es 2 ... }

REPETIR MIENTRAS (Divisor $<$ N) y (esPrimo = verdadero)

SI N es divisible por Divisor **ENTONCES** (usando módulo: $N // \text{Divisor} = 0$)

esPrimo \leftarrow falso { Si Divisor divide a N , NO ES PRIMO }

Divisor \leftarrow Divisor + 1 { paso al próximo posible Divisor }

fin repetir mientras

FIN ALGORITMO

- Tarea 1: Implemente en Pascal este algoritmo usando WHILE.
- Tarea 2: Luego implemente usando REPEAT-UNTIL.

Resolución de Problemas y Algoritmos - 2016 15