UNIVERSIDAD CATÓLICA ANDRÉS BELLO
ESCUELA DE INGENIERÍA INFORMÁTICA

Guía Práctica

Estudiantes Asesores de Ingeniería

Algoritmos y Programación I – Rolando Andrade
Caracas, 07 de Noviembre de 2020

Pregunta 1.

Los números primos son aquellos números distintos a 1 que solo son divisibles por sí mismos y 1. Cree un programa que dado un número mayor a 1, imprima en pantalla si es primo o no, como en el siguiente ejemplo:

```
Introduce un numero: 5
El 5 es primo

Introduce un numero: 8
El 8 no es primo

Introduce un numero: 0
Hasta pronto

Process has ended with exit code 0
```

Bonus: Realice una función "esPrimo(x: Integer): Boolean" que reciba un número, y retorne verdadero si es primo, o falso de no serlo. Luego en el programa principal imprima si es primo o no dependiendo del resultado de la función.

Pregunta 2.

Los números primos son aquellos números distintos a 1 que solo son divisibles por sí mismos y 1. Cree un programa que dado un número mayor a 1, imprima en pantalla todos sus factores primos, y si es primo, indicarlo, como se ve en el ejemplo:

```
Introduce un numero: 5
El 5 es primo

Introduce un numero: 8
El 8 es divisible por 1, 2, 4 y 8.

Introduce un numero: 0
```

```
Hasta pronto

Process has ended with exit code 0
```

Pregunta 3.

Cree un procedimiento "tablaDeMultiplicar(x: Integer)" que reciba un número, e imprima la tabla de mutiplicar de dicho número del 1 hasta x, como en el ejemplo:

```
Introduce un numero: 5
5 * 1 = 5
5 * 2 = 10
5 * 3 = 15
5 * 4 = 20
5 * 5 = 25

Introduce un numero: 8
8 * 1 = 8
8 * 2 = 16
8 * 3 = 24
8 * 4 = 32
8 * 5 = 40
8 * 6 = 48
8 * 7 = 56
8 * 8 = 64

Introduce un numero: 0
Hasta pronto

Process has ended with exit code 0
```

Pregunta 4.

Determine que realiza el siguiente programa:

```
function UADEJHADADA(x: Integer): Integer;
var
  i, n: Integer;
begin
  n:=1;
  for i:= x downto 1 do
  begin
    n:=n*i;
  end;
  UADEJHADADA:=n;
end;
```

Puede apoyarse de corridas en frío de ser necesario.

Pregunta 5.

Dado un string de números, ordene los valores presentes de menor a mayor como en el ejemplo:

```
Introduce una secuencia de números: 987654321
Ordenado: 123456789

Introduce una secuencia de números: 46464679498
Ordenado: 44446667899

Introduce una secuencia de números: 0
Hasta pronto

Process has ended with exit code 0
```

Pregunta 6.

Dado un string de números, sume los strings con el algoritmo de suma de primaria:

```
     1 1 1
+  12345
+    678
-------------
   13023
```

No necesariamente el primer número es mayor al segundo, pero usted puede ordenarlos por tamaño si lo desea.

```
Introduce x: 12345
Introduce y: 678
Resultado 12345 + 678 = 13023

Introduce x: 678
Introduce y: 12345
Resultado 678 + 12345 = 13023

Introduce una secuencia de números: 0
Hasta pronto

Process has ended with exit code 0
```

Pregunta 7.

Estudie la sentencia mostrada, luego determine el valor de verdad (V o F) de todas las opciones disponibles en menos de 1 segundo por pregunta.

Sean a y b integers.

1. (a+b<b) or (not(a+b <> a) and (not(not(not(a*b*0<>0))))) or (a<>b)
    a. a=1, b=2
    b. a=6, b=5
    c. a=-1, b=-1
    d. a=5, b=5
    e. a=8, b=3
    f. a=-3, b=-3
    g. a=7, b=7
    h. a=0, b=0
2. (1<>1) and (a+b=a-b) and ((a+b<b) or (not(a+b <> a) and (not(not(not(a*b*0<>0))))) or (a<>b))
    a. a=4, b=3
    b. a=2, b=1
    c. a=-1, b=-1
    d. a=0, b=0
    e. a=7, b=10
    f. a=-14, b=-3
    g. a=10, b=11
    h. a=12, b=26
3. not(not(a+b<b) and not((not(a+b <> a) and (not(not(not(a*b*0<>0)))))) and not((a<>b))
    a. a=17, b=12
    b. a=41, b=2
    c. a=-1, b=-1
    d. a=5, b=5
    e. a=14, b=12
    f. a=-3, b=-3
    g. a=7, b=7
    h. a=0, b=0

Pregunta 8.

Dado el mes y el año de una fecha, imprima el calendario de ese mes como en el ejemplo:

```
Fecha: 11/2020

November 2020

Mo Tu We Th Fr Sa Su
                   01
02 03 04 05 06 07 08
09 10 11 12 13 14 15
16 17 18 19 20 21 22
23 24 25 26 27 28 29
30

Process has ended with exit code 0
```

Pregunta 9.

Evaluar expresiones lógicas puede ser aburrido. Existen 4 operaciones básicas: AND (Y), OR (O), NOT (N), IMPLICA (I) y DOBLE IMPLICA (D). Realice un programa que reciba una expresión y sus parámetros, y la evalúe como en el ejemplo:

```
Introduce expresión: pYq
Introduce valor de p [V-F]: V
Introduce valor de q [V-F]: V
El resultado es V

Introduce expresión: aONb
Introduce valor de a [V-F]: F
Introduce valor de b [V-F]: F
El resultado es V

Introduce expresión: rINs
Introduce valor de a [V-F]: V
Introduce valor de b [V-F]: F
El resultado es F

Introduce expresión: mANm
Introduce valor de a [V-F]: V
El resultado es F

Introduce expresión: SALIR

Process has ended with exit code 0
```

Cuando una expresión tiene varios elementos como por ejemplo:

```
Introduce expresión: pYqOp
Introduce valor de p [V-F]: V
Introduce valor de q [V-F]: F
El resultado es V
```

Las operaciones se hacen en orden de izquierda a derecha. Primero se hace la operación pYq y luego (pYq)Op. En caso de haber negadores, resolverlos siempre de primero.

Por ahora no considere el uso de paréntesis, cuando vea recursividad inténtelo.

Pregunta 10.

Realice una corrida en frío para determinar qué salida genera el programa:

```
function GCD (a: Integer; b: Integer): Integer;
var
   r: Integer;
begin
    while (a>0) do
    begin
        r := a;
        a := b mod a;
        b := r;
    end;
    GCD := b;
end;
```

Intente realizarla con los valores a=10, b=15; a=21, b=2; a=20, b=2; a=3, b=12.

Intente determinar qué hace la función, el por qué puede ser tan útil y cómo la usaría para determinar si dos números son primos relativos entre sí.

Pregunta 11.

Realice un programa que tenga los siguientes procedimientos o funciones:

- function leer(): vector; //lee y retorna un vector de números.
- procedure imprimir(v: vector); //muestra el vector.
- function sum(v: vector): Integer; //retorna la suma de todos los elementos del vector.

- function max(v: vector): Integer; //retorna el mayor elemento del vector.
- function min(v: vector): Integer; //retorna el menor elemento del vector.
- function buscar(v: vector; x: Integer): Integer; //retorna el índice con la posición del elemento buscado o -1 de no encontrarse.
- function voltear(v: vector): vector; //retorna el mismo vector volteado.
- function ordenar(v: vector; comparador: TComparador): vector; //retorna el vector ordenado según el comparador enviado.

```
Dame array de enteros de longitud 5: 1 2 3 4 5

El array es: 1 2 3 4 5
La suma es: 15
El máximo es: 5
El mínimo es: 1
Volteado es: 5 4 3 2 1
Ordenado ascendente es: 1 2 3 4 5
Ordenado descendente es: 5 4 3 2 1
Número a buscar: 2
Su posición es: 2

Process has ended with exit code 0
```

Pregunta 12.

2048 es un juego muy interesante, al mover el tablero de 4x4 hacia cualquier lado, suma los números potencia de 2 que sean iguales en la fila o en la columna y los envía lo más cercano a ese lado.

En un primer acercamiento se le pide crear un algoritmo para sumar los números de la fila y enviarlos a un lado como se muestra en el ejemplo:

```
Introduce fila: 2 2 2 2
Hacia la derecha: 0 0 4 4
Hacia la izquierda: 4 4 0 0

Introduce fila: 0 0 4 4
Hacia la derecha: 0 0 0 8
Hacia la izquierda: 8 0 0 0

Introduce fila: 0 16 4 2
Hacia la derecha: 0 16 4 2
Hacia la izquierda: 16 4 2 0

Introduce fila: 16 4 2 2
Hacia la derecha: 0 16 4 4
Hacia la izquierda: 16 4 4 0
```

```
Introduce fila: 16 0 0 16
Hacia la derecha: 0 0 0 32
Hacia la izquierda: 32 0 0 0

Introduce fila: 0 0 0 0
Hasta pronto

Process has ended with exit code 0
```

Puedes usar cualquiera de los métodos y funciones creadas en la pregunta anterior si lo crees necesario.

Pregunta 13.

Sean las siguientes funciones con nombres para confundir:

```
function suma(a: Integer; b: Integer): Integer;
begin
    suma := b - a;
end;

function resta(a: Integer; b: Integer): Integer;
begin
    resta := a + b;
end;

function Y(a: Boolean; b: Boolean): Boolean;
begin
    Y := a or b;
end;

function O(a: Boolean; b: Boolean): Boolean;
begin
    O := a and b;
end;
```

Determine las salidas para las siguientes sentencias:

4. O(Y(suma(resta(1,2), 5) > 6, resta(suma(5, 3), resta(2, 1)) < 0), true).
5. resta(resta(resta(1, 2), suma(1, 2)), resta(resta(1, 2), suma(1, 2))
6. Y(O(true, not(suma(1,2)>4), false)
7. not (suma(1,2)+resta(suma(resta(6,5), 5), 10) > 0)
8. suma(10, 20) + suma(5, 4) - resta(7, 3) + 40 + 10 * suma(2,5)
9. resta(1,0) * resta(1,1) * resta(1,resta(1,1)) *resta(resta(1,1),resta(1,1))

Si quieres inventa más sentencias y sigue practicando.

PD: Anidar funciones de esta manera es muy mala práctica, si te toca hacerlo en algún momento, divide la sentencia en partes más pequeñas y ve guardando el resultado en variables temporales.

Pregunta 14.

Ordena a mano los siguientes arreglos utilizando todos los algoritmos vistos en clase, guardando el estado y los cambios, por ejemplo:

Estado	Cambios
3 2 1 2 3 1 1 3 2	3 -> 2 = 2 3 1 2 -> 1 = 1 3 2 3 -> 2 = 1 2 3

Puedes usar cualquier otro tipo de columna guía para hacer tu corrida, lo importante es que dejes claro los cambios y el estado para contarlos y confirmar que hiciste el algoritmo correcto.

```
20 45 21 10 15 6 7 9 12 17 60 35
5 6 5 2 4 3 5 5 7 8 9 11 5 8
1 1 1 1 1 1
10 5 20 15 30 25 40 35
PACO CASA CORRO LARA HOLA ALO JOSE MARY
ABC CBD CSD DIN NAR ABC ACA
```

Pregunta 15.

Para cada variable intenta identificar a qué ámbito pertenece, por ejemplo:

```
program P;
var
    x, y, z : Integer;
procedure P1(x, w: Integer);
begin
    ...
    procedure P2(y, x, z: Integer);
    ...
    procedure P3(w, y, z: Integer);
end;
```

```
P usará x, y, z globales
P1 usará x, w de sus parámetros, e y, z globales
P2 usará las x, y, z de sus parámetros y la w de P1
P3 usará w, y, z de sus parámetros y x de P1
```

```
1.
program P;
var
    a, b, c, d: Integer;
procedure P1(a, x: Integer);
begin
    ...
    procedure P2(b, y, c: Integer);
    begin
        ...
        procedure P3(w, y, z: Integer);
        ...
    end;
    ...
    procedure P4(b, a, c: Integer);
end;


2.
program P;
var
    a, b, c, d, e, f: Integer;
procedure P1(a, x: Integer);
begin
    ...
    procedure P2(a, b, c, x, y, z: Integer);
    begin
        ...
        procedure P3(e, f, x: Integer);
        ...
    end;
    ...
    procedure P4(c, d, e: Integer);
    begin
        ...
        procedure P5(e, f: Integer);
        ...
    end;
end;
```

Pregunta 16.

Cree un programa que determine si una matriz de 2x2 es submatriz de otra matriz de 5x5. Una matriz es submatriz de otra si la misma la contiene, ejemplo:

```
    1 2 3
A = 4 5 6      B = 2 3       B es submatriz de A
    7 8 9          5 6
```

```
Introduce matriz A:
F1: 1 2 3 4 5
F2: 6 7 8 9 10
F3: 11 12 13 14 15
F4: 16 17 18 19 20
F5: 21 22 23 24 25

Introduce matriz B:
F1: 12 13
F2: 17 18

B es submatriz de A

Introduce matriz A:
F1: 1 2 3 4 5
F2: 6 7 8 9 10
F3: 11 12 13 14 15
F4: 16 17 18 19 20
F5: 21 22 23 24 25

Introduce matriz B:
F1: 12 14
F2: 17 19

B NO es submatriz de A

Introduce matriz A:
F1: 1 2 3 4 5
F2: 6 7 8 9 10
F3: 11 12 13 14 15
F4: 16 17 18 19 20
F5: 21 22 23 24 25

Introduce matriz B:
F1: 1 2
F2: 6 7

B es submatriz de A
```

Pregunta 17.

2048 es un juego muy interesante, aplique el algoritmo del problema 12 con las columnas y las filas y agregue un 2 en una casilla aleatoria como se muestra en el ejemplo:

```
2048
Arriba [u], abajo [d], derecha: [r], izquierda [l]

0 0 2 0
0 0 0 0
0 0 0 0
R: u

0 0 2 0
0 2 0 0
0 0 0 0
R: u

0 2 2 0
0 0 0 0
0 0 2 0
R: r

0 0 0 4
0 0 0 0
0 0 2 2
R: r

2 0 0 4
0 0 0 0
0 0 0 4
R: l

2 4 0 0
2 0 0 0
4 0 0 0
R: d

0 0 2 0
4 0 0 0
4 4 0 0
R:
```

¡Felicidades por uno de tus primeros juegos!

Pregunta 18.

Crea un repositorio en Github donde vayas subiendo las actividades de esta guía.

5 tips para ser mejor Ingeniero en informática

1. ***Actúa con prudencia**: Pese a que el calendario parezca cómodo, el tiempo pasa volando, empieza cuanto antes y poco a poco ve trabajando en tu proyecto, no esperes a última hora para empezar.*

2. ***Aplica los principios de la programación funcional**: Divide los problemas en problemas más pequeños que puedas manejar y entender.*

3. ***Pregúntate qué haría el usuario (no eres el usuario)**: Todos piensan distinto, debes ponerte en los zapatos del usuario de verdad para entender cómo actúa y qué errores comete para prevenirlos.*

4. ***Crea un estándar**: En un proyecto trabajan varias personas, cada una con sus gustos y preferencias, al principio del proyecto deben ponerse de acuerdo en cómo escribir el código, convenios para las variables o nombres de funciones/procedimientos, etc.*

5. ***La verdadera belleza está en la simplicidad:** Los tiempos han cambiado, no debes sufrir por resolver un problema de manera fea pero eficiente, ahora lo que importa es la simplicidad de tu código, que se pueda entender al leer, sin brincar de un lado a otro, líneas de 100 palabras o variables locas.*