Estudios de Informática, Multimedia y Telecomunicaciones

Fundamentos de Programación

PEC1 – Primera Prueba de Evaluación Continuada

Apellidos: *Escribe aquí tus apellidos*

Nombre: *Escribe aquí tu nombre*

Indicaciones generales:

Razonad y justificad todas las respuestas.

Las respuestas incorrectas **no** disminuyen la nota.

Para dudas y aclaraciones sobre el enunciado, dirigíos al consultor responsable de vuestra aula de teoría. Para dudas sobre codificación, dirigíos al consultor responsable de vuestra aula de laboratorio.

Entrega:

1. Se ha de enviar la solución de los ejercicios 1...9 en un fichero llamado *ApellidosNombre_FP_PEC1* dirigido al buzón de entrega de actividades del aula de teoria.
2. Se ha de enviar la codificación de la pregunta 10 a través del **Corrector Automático**, que encontraréis en el aula de teoria.
3. Fecha límite para entregar la solución: **Lunes, 18 de Octubre de 2010** (a las 23:59 hores).

Es imprescindible respetar el formato y fecha de entrega. La no adecuación a estas especificaciones puede suponer la no evaluación de la PEC.

Ejercicio 1: Identificación de tipos [10%]

Objetivos: Mostrar conocimientos elementales sobre los tipos de datos, como manipularlos e identificar los tipos resultantes de efectuar operaciones entre objetos elementales del lenguaje algorítmico

Materiales: M1 y M2 hasta M2.1

Tarea: Dadas las siguientes definiciones de tipos, constantes y variables

```
tipo
        tBattery = {NiCd, NiMH, LiIon, LiSOCl2, LeadAcid}
        tSize = { D, C, AAA, AA}
ftipo

var
        voltage, power, energy : real;
        powerSource : tBattery;
        size: tSize;
        capacity :entero;
fvar
```

indicar el tipo resultante de evaluar las expresiones siguientes:

i) integerToReal(capacity)*voltage > power

El resultado es de tipo **booleano**.

ii) capacity = realToInteger(power / voltage)

El resultado es de tipo **booleano**.

iii) realToInteger (energy) **div** capacity

El resultado es de tipo **entero**.

iv) size = AAA **y** powerSource = NiMH

El resultado es de tipo **booleano**.

Ejercicio 2: Sintaxis de expresiones [10%]

Objetivos: Mostrar la habilidad en la confección de expresiones en lenguaje algorítmico. Utilización apropiada de objetos elementales del lenguaje algorítmico para la confección de las expresiones dadas unas declaraciones previas de objetos elementales.

Materiales: M1 y M2 hasta M2.1

Tarea: Dadas las siguientes definiciones de tipos, constantes y variables:

```
tipo
        tBattery = {NiCd, NiMH, LiIon, LiSOCl2, LeadAcid}
        tSize = { D, C, AAA, AA}
ftipo

var
        voltage, power, energy : real;
        powerSource : tBattery;
        size: tSize;
        capacity :entero;
fvar
```

Si las variables guardan datos relativos a una batería en particular, escribid el lenguaje algorítmico correspondiente a cada una de las descripciones siguientes:

i) Expresión que calcule cuantas baterías serian necesarias para disponer de una capacidad de al menos 10000 mAh, sabiendo que la variable *capacity* guarda la capacidad de una sola batería (el resultado final admite decimales).

```
10000.0 / integerToReal(capacity)
```

ii) Expresión para saber si el ratio entre *power* y *voltage* es superior al ratio entre *energy* y *capacity*

```
( power / voltage ) > ( energy / integerToReal(capacity) )
```

iii) Expresión para saber si una batería tiene un voltaje superior a 3.5 V, es de tipo LiIon o LiSOCl2, y su tamaño es D

```
(voltage>3.5) y (powerSource = LiIon o powerSource = LiSOCl2) y (size = D)
```

iv) Expresión para saber si una batería es de tamaño AAA, y su composición es NiCd.

```
(size = AAA)  y  (powerSource = NiCd)
```

Ejercicio 3: Evaluación de expresiones [10%]

Objetivos: Mostrar la habilidad en la manipulación de expresiones en lenguaje algorítmico. Dadas unas declaraciones previas de objetos elementales, saber explicar cómo efectuar su evaluación y proporcionar el resultado correcto.

Materiales: M1 y M2 hasta M2.1

Tarea: Dadas las definiciones de tipos, constantes y variables del ejercicio 2 y suponiendo que las variables tienen los valores siguientes:

voltage:=11.2, power:=100.0; energy := 25.7;
powerSource := LeadAcid;
size:=D;
capacity :=9850;

Calculad el resultado de las siguientes expresiones:

i) integerToReal(capacity)/1000.0 * voltage > power

9850.0/1000.0 * 11.2 > 100.0
9.850*11.2 > 100.0
110.32 > 100.0
CIERTO

ii) size = AAA **o** powerSource = NiMH **o no** (capacity > realToInteger (voltage * energy))

FALSO **o** FALSO **o no** CIERTO
FALSO **o** FALSO **o** FALSO
FALSO

iii) integerToReal(capacity)/(voltage*power)<integerToReal(capacity **div** realToInteger(energy))

9850.0/(11.2*100.0)<integerToReal(9850 div 25)
9850.0/1120<394.0
8.79<394.0
CIERTO

iv) powerSource = LeadAcid **y no** (size = AA) **y** (energy * energy < power * voltage)

CIERTO **y no** FALSO **y** 25.7*25.7 < 100.0*11.2
CIERTO **y** CIERTO **y** 660.49 < 1120.0
CIERTO **y** CIERTO **y** CIERTO
CIERTO

Ejercicio 4: Corrección de expresiones [10%]

Objetivos: Mostrar la habilidad en la manipulación de expresiones en lenguaje algorítmico. Dadas unas declaraciones previas de objetos elementales, razonar la corrección de las expresiones, argumentando los motivos de la incorrección si procede.

Materiales: M1 y M2 hasta M2.1

Tarea: Dadas las siguientes definiciones de tipos, constantes y variables.

```
const
        MINDISTANCE: real = 10.0;
fconst

tipo
        tInterfaceWireless = {Bluetooth, Wifi, GSM, ZigBee, WiMax}
ftipo

var
        interface : tInterfaceWireless;
        range:real;
        inComputer: booleano;
        ratio: entero;
fvar
```

Decir si son correctas o no las siguientes expresiones. En caso de que no lo sean, especificar el motivo o motivos.

i) interface = Bluetooth **o** interface = Wifi **o** interface < GSM+WiMax

Incorrecta, no se pueden hacer operaciones aritméticas sobre tipos enumerativos.

ii) MINDISTANCE = integerToReal(range) / ratio

Incorrecta: no se puede hacer la operación / con un entero (ratio) y además range no es entero.

iii) inComputer = range > MINDISTANCE **o** interface = tInterfaceWireless

Incorrecta: no se puede hacer una comparación con un tipo de datos.

iv) Bluetooth<Wifi<GSM

Incorrecta: Los operadores de comparación dan como resultado CIERTO o FALSO y no se pueden comparar con otros tipos.

Ejercicio 5: Algoritmo en lenguaje natural y especificaciones [10%]

Objetivos: Describir en palabras (lenguaje natural) un algoritmo y las especificaciones (precondiciones y postcondiciones). Sólo se pretende detectar la habilidad en identificar los contenidos de los diferentes elementos del algoritmo así como mostrar un conocimiento básico del algoritmo descrito.

Materiales: M1 y M2 hasta M2.2

Tarea: Se trata de que describais y especifiquéis un algoritmo a seguir para sacar dinero de un cajero automático (evidentemente no habrá solución única puesto que diferentes cajeros siguen diferentes secuencias de acciones).

{pre: disponer de una cuenta bancaria, disponer de una tarjeta de crédito asociada a la cuenta, disponer de dinero suficiente en la cuenta, disponer de un cajero automático con suficiente dinero en efectivo}

- Introducir la tarjeta en el cajero automático
- Seleccionar la opción para sacar dinero
- Introducir el código secreto
- Seleccionar el importe
- Pedir justificante de la operación
- Esperar a que el cajero haga la operación
- Recoger la tarjeta
- Recoger el dinero
- Recoger el justificante
- Guardar tarjeta, dinero y justificante en la cartera
- Seleccionar no hacer ninguna otra operación

{post: se ha obtenido del cajero la cantidad de dinero deseada, cantidad que ha sido descontada del total disponible en la cuenta y del total de dinero en efectivo del cajero.}

Ejercicio 6: Especificar las variables necesarias así como las Pre y Post-condiciones para resolver un problema.

Objetivos: Evaluar la habilidad de especificar un algoritmo. Dada una especificación informal de un problema, especificar las variables, la Pre y Post-condición para resolverlo.

Materiales: M1 y M2 hasta M2.2

Tarea: Especificar el problema siguiente: “Se desea calcular el tiempo medio por kilómetro en minutos y segundos empleado por un corredor de maratón, dado el tiempos total en horas, minutos, y segundos “. Considerad que la distancia recorrida es de 42,195 Km.

En concreto, se pide que se definan las variables necesarias para representar los datos del problema (apartado var ... fvar de un algoritmo) y que deis la precondición y la postcondición.

```
var
        hoursM:      entero;
        minutesM:    entero;
        secondsM:    entero;
        minutesKm:   entero;
        secondsKm:   entero;
fvar

{Pre: hoursM = HOURS y minutesM = MINUTES  y seconsdM = SECONDS}

        CalcularTiempoPorKm

{Post: minutesKm = realToInteger((hoursM*3600+minutesM*60+secondsM) / 42,195)
div 60 y  secondsKm = realToInteger((hoursM*3600+minutesM*60+secondsM) /
42,195) mod 60}
```

Ejercicio 7: Comprensión de algoritmos[10%]

Objetivos: Evaluar conocimientos básicos de algorítmica. Estructura general de un algoritmo y el uso de estructuras algorítmicas básicas. Se plantean preguntas de interpretación dado un algoritmo en lenguaje algorítmico.

Materiales: M1 y M2 hasta M2.3

Tarea: Considerando el siguiente algoritmo:

{Pre: n=N}

```
algoritmo numero
var
    n,a,b: entero;
fvar

n:=readInteger();
b:=0;

mientras n≠0 hacer
      a:=n mod 10;
      b:=(b*10)+a;
      n:=n div 10;
fmientras

writeInteger(b);

falgoritmo
```

se pide:

i) Explicar qué hace el algoritmo

Calcular los dígitos del número de entrada y crear un nuevo número con los mismos dígitos colocados en orden inverso.

ii) En caso de que dentro del bucle **mientras** cambiemos el orden de las instrucciones de la siguiente manera:

```
b:=(b*10)+a;
a:=n mod 10;
n:=n div 10;
```

indicar como tendríamos que modificar (si es que es necesario) el algoritmo para que continue funcionando de la misma forma.

{Pre: n=N}

```
algoritmo numero
var
    n,a,b: entero;
fvar

n:=readInteger();
b:=0;
a:=0;


mientras n≠0 hacer
     b:=(b*10)+a;
     a:=n mod 10;
     n:=n div 10;

fmientras
b:=(b*10)+a;
writeInteger(b);

falgoritmo
```

Ejercicio 8: Modificación del comportamiento de un algoritmo [10%]

Objetivos: Evaluar conocimientos básicos de algorítmica. La evaluación va un paso más allá de la interpretación básica en plantear modificaciones del mismo. Se piden modificaciones en el comportamiento de un algoritmo ya dado para que éste pueda contemplar situaciones diferentes.

Materiales: M1 y M2 hasta M2.3

Tarea: Considerando el algoritmo del ejercicio 7,

i) ¿Cómo se tendría que modificar el algoritmo para aplicarlo a la parte decimal de un número real (la parte fraccionaria que va después del punto) de la misma manera que se aplicaba a un entero (sólo se debe transformar la parte decimal, no todo el número real)?

Solución – Extraer la parte decimal y proceder como en el caso de enteros.

```
algoritmo numero
var
    n,a,b: entero;
    r: real;
fvar
```

```
r:=readReal();
r:=r-integerToReal(realToInteger(r));{Se queda con la parte
decimal}

mientras r-integerToReal(realToInteger(r))≠ 0 hacer
      r:= r * 10.0;
fmientras
n:=realToInteger(r);
b:=0;

mientras n≠0 hacer
      a:=n mod 10;
      b:=(b*10)+a;
      n:=n div 10;
fmientras

writeInteger(b);

falgoritmo
```

ii) ¿Cómo se tendría que modificar el algoritmo original para considerar la posibilidad de que el número entero de entrada fuese negativo?

```
algoritmo numero
var
    n,a,b: entero;
    sign: entero;
fvar
n:=readInteger();
si n<0 entonces
      sign := -1;
sino
      sign := 1;
fsi
b:=0;

mientras n≠0 hacer
      a:=n mod 10;
      b:=(b*10)+a;
      n:=n div 10;
fmientras

b:= b * sign;

writeInteger(b);

falgoritmo
```

Ejercicio 9: Bucles [10%]

Objetivos: Evaluar conocimientos de algorítmica poniendo énfasis en las composiciones iterativas. Se pide reescribir una iteración empleando **para** ... **fpara** en lugar de **mientras** ... **fmientras**, así como alguna pequeña modificación para consolidar el conocimiento que se tiene del algoritmo reescrito.

Materiales: M1 y M2 hasta M2.3

Tarea: Considerando el algoritmo del ejercicio 7,

i) reescribir el mismo algoritmo, pero –esta vez– utilizad la estructura **para** ... **fpara** en lugar del **mientras** ... **fmientras**, si es que es possible. Si no lo es, explicar los motivos.

No es posible hacer la sustitución porque no conocemos el número de cifras del entero y por tanto no sabemos el límite de un posible contador para el bucle **para....fpara**. Sería necesario utilizar primero otro **mientras....fmientras** para obtener dicho número de cifras y consecuentemente, el algoritmo resultante no seria eficiente respecto a la versión inicial.

ii) Indicar todas las posibles ordenaciones diferentes de las sentencias de dentro del **mientras...fmientras** que permitan que el algoritmo siga dando el mismo resultado sin tener que modificar nada más.

```
...
mientras n≠0 hacer
      a:=n mod 10;
      n:=n div 10;
      b:=(b*10)+a;
fmientras
...
```

Fijaros que la asignación `a:=n mod 10` se ha de ejecutar antes de modificar el valor de la variable n, ya que la expresión la utiliza. Y `b:=(b*10)+a` se ha de ejecutar antes de modificar la variable *a* ya que la utiliza.

Ejercicio 10: [10%]

Objetivos: Dado un listado en lenguaje algorítmico, traducirlo a C. Se aconseja poner asignaciones y comparaciones, que en C requieren utilizar adecuadamente el = y el ==.

Materiales: M1 y M2 hasta M2.3 así como la traducción de lenguaje algorítmico a C

Tarea: Se pide que codifiquéis el algoritmo siguiente en lenguaje C, lo compiléis y lo probéis. Los resultados se deberán mostrar con dos cifras decimales.

{Pre: En la entrada estándar hay una secuencia de la forma: `x1 y1 x2 y2` , que representan las coordenadas iniciales y finales de una línea de pendiente menor de 45º, respectivamente.}

```
algoritmo coordenadas
var
      dx, dy, p, end: entero;
      x1, x2, y1, y2, x, y: real;

      x1 := readReal();
      y1 := readReal();
      x2 := readReal();
      y2 := readReal();

      si x1>=x2 entonces
            dx := realToInteger(x1 - x2);
      sino
            dx := realToInteger(x2 - x1);
      fsi

      si y1>=y2 entonces
            dy := realToInteger(y1 - y2);
      sino
            dy := realToInteger(y2 - y1);
      fsi

      p := 2 * dy - dx;

      si x1 > x2 entonces
            x := x2;
            y := y2;
            end := realToInteger(x1);
      sino
            x := x1;
            y := y1;
            end := realToInteger(x2);
      fsi

      writeReal(x);
      writeReal(y);
```

```
      mientras x < integerToReal(end) hacer
            x := x + 1.0;

            si p < 0 entonces
                  p := p + 2 * dy;
            sino
                  y := y + 1.0;
                  p := p + 2 * (dy - dx);
            fsi
            writeReal(x);
           writeReal(y);
      fmientras
falgoritmo
```

{Post: Se ha escrito por la salida estándar la secuencia que corresponde a la lista de puntos `(x,y)` que hay en la línea entre `(x1,y1)` y `(x2,y2)`.}

Por ejemplo, si la entrada de coordenadas corresponde a los valores 10.0 10.0 20.0 15.0, la salida corresponderá a los puntos (escritos en columna para mejorar su legibilidad, no es necesario hacerlo así en el código resultante):

10.00 10.00
11.00 11.00
12.00 11.00
13.00 12.00
14.00 12.00
15.00 13.00
16.00 13.00
17.00 14.00
18.00 14.00
19.00 15.00
20.00 15.00

```
/* PAC1: Algoritmo coordenadas */
#include <stdio.h>


int main (void)
{
    int   dx,dy,p,end;
    float x1,x2,y1,y2,x,y;

    scanf("%f", &x1);
    scanf("%f", &y1);
    scanf("%f", &x2);
    scanf("%f", &y2);

    if (x1>=x2)
        dx = (int)(x1-x2);
    else
        dx = (int)(x2-x1);

    if (y1>=y2)
        dy = (int)(y1-y2);
    else
```

```
        dy = (int)(y2-y1);

    p = 2 * dy - dx;

    if (x1>x2){
        x = x2;
        y = y2;
        end = (int)(x1);
    }else{
        x = x1;
        y = y1;
        end = (int)(x2);
    }

    printf("%.2f ", x);
    printf("%.2f ", y);

    while (x<(float)(end)){
          x = x + 1.0;
          if (p < 0)
             p = p + 2 * dy;
          else{
             y = y + 1.0;
             p = p + 2 * (dy-dx);
          }
          printf("%.2f ", x);
          printf("%.2f ", y);
    }
    return 0;
}
```