



Universidad
de Huelva

ALGORITMOS Y ESTRUCTURAS DE DATOS I

- Septiembre 2001 -

Dpto. Ing. Electrónica, Sistemas Informáticos y Automática
E.P.S. La Rábida (Universidad de Huelva)

Apellidos :	DNI :
Nombre :	Especialidad :

Sección Informativa:

- El examen se compone de una sola parte de dos horas de duración y consta de 4 ejercicios.
- Solo se podrán realizar consultas durante los primeros 20 minutos, por lo que deberá leer atentamente el examen completo.
- Se agotará la convocatoria transcurridos los primeros 20 minutos.

1.- (2.5 Puntos)

Escribir un programa recursivo que liste todos los pares de enteros positivos que son la suma de un número, dado. Por ejemplo:

$$7 = 6 + 1, 5 + 2, 4 + 3$$

Nota: No se pueden repetir las parejas, por ejemplo, 6+1 y 1+6. El subprograma que se le pide puede ser llamado desde otra acción o función que también habría que escribir.

2.- (2.5 Puntos)

Escribir un subprograma que determine, mediante la utilización de una pila, si una expresión con paréntesis leída desde un fichero de acceso secuencial, está equilibrada (equilibrada es aquella que tienen tantos símbolos de apertura como de cierre de paréntesis). Por ejemplo, la expresión:

$$(x + y * (m / (n - p))) + (x / y) \quad \text{está equilibrada}$$

$$(x + y * (m / (n - p))) + (x / y) \quad \text{no está equilibrada}$$

Nota: Para la realización del ejercicio es obligatorio la utilización de *primitivas* o procedimientos y funciones propios del manejo de la estructura de datos *pila*.

3.- (1.5 puntos)

Implemente la declaración de tipos correspondiente a lista_circular, pila y cola con memoria dinámica.

4.- (3.5 puntos)

Localice y corrija los errores que encuentre en los siguientes subprogramas. (*Localizar* consiste en indicar en un párrafo de no más de dos líneas, el problema encontrado en el código. *Corregir* consiste en poner el código correcto de modo que cumpla su cometido).

a)

funcion Ultimo (list:punt):punt;

(* Localiza un puntero al último elemento de una lista simplemente enlazada *)

var

ult : punt;

fvar

inicio

reservar(ult);

si (Vacía(list)) →

ult := nulo (* El último de lista vacía lo consideramos nulo *)

no (vacía(list)) →

inicio

ult:=list;

mientras (ult^.siguiente <> nulo) hacer

ult:= ult^.siguiente;

fin

liberar (ult);

retorna (ult)

ffuncion

b)

accion Vaciar_Pila (var p: pila);

(* Vacía la pila liberando la memoria que ocupa *)

var

paux := pila

fvar

inicio

si p<>nulo →

inicio

paux:= p;

liberar (paux)

fin;

faccion

c) (En este apartado hay tres tipos de error).

accion Mezcla (f1, f2 : fichero de regs; long: entero; var f sal :fichero de regs);
(* Mezcla de los ficheros ordenados de acceso directo *f1* y *f2*, (ambos de longitud *long*), en un fichero de salida de acceso secuencial *f_sal* manteniéndolo ordenado. (El campo por el que se ordena el registro es *orden*). Los ficheros no han sido asignados ni abiertos *)

var

i,j:entero; d1,d2:regs;

fvar

inicio

```
abrir(f1,1,'entrada1.dat');
abrir(f2,1,'entrada2.dat');
abrir(f_sal,e,'salida.dat');
i:=1; (* contador fichero entrada 1 *)
j:=1; (* contador fichero entrada 2 *)
leer(f1,1,d1);
leer(f2,1,d2);
mientras (i <= FDF(f1)) y (j <= FDF(f2)) hacer
  inicio
    si d1.orden[i] <= d2.orden [j] →
      inicio
        escribir(f_sal,d1);
        i:=i+1;
      fin
    □ d1.orden > d2.orden →
      inicio
        escribir(f_sal,d2);
        j:= j+1;
      fin
  fin;
si (i <= long) →
  inicio
    mientras (i <= FDF(f1)) hacer
      inicio
        escribir(f_sal,d1);
        i:= i+1;
      fin
  fin
  □ (j<=long) →
    inicio
      mientras (j <= FDF(f2)) hacer
        inicio
          escribir(f_sal,d2);
          j:=j+1;
        fin
    fin
```

faccion



ALGORITMOS Y ESTRUCTURAS DE DATOS I

SOLUCIÓN DEL EXAMEN

- Septiembre 2001 -

Dpto. Ing. Electrónica, Sistemas Informáticos y Automática
E.P.S. La Rábida (Universidad de Huelva)

EJERCICIO 1

accion CalculaSumas ();

var

num: entero;

fvar

inicio

escribir ('Introduce el numero');
leer (num);
CalculaRecursivo (num,num-1,1);

faccion

accion CalculaRecursivo(numero:entero; p1:entero; p2:entero)

inicio

si $p1 \geq p2 \rightarrow$

inicio

escribir (p1); escribir ('+'); escribir (p2);
CalculoRecursivo (numero, p1-1,p2+1);

fin

$p1 < p2 \rightarrow$ nada,

faccion

EJERCICIO 2

accion Equilibrada (fich:fichero de caracter);

var

P:pila;
expr:caracter,
correcto:booleano;

fvar

inicio

correcto:=verdad;
abrir(fich,l, 'EXAMEN');
leer (fich,expr);
mientras no fdf(fich) y correcto hacer

inicio

si expr = ' (' \rightarrow push(expr, p)

expr = ' (' \rightarrow Correcto:= pop(expr,p)

(* si en algún momento no se puede sacar un paréntesis de la pila es porque le falta su homólogo *)

Otro caso \rightarrow nada;

leer (fich, expr);

fin;

si no correcto \rightarrow escribir ('no esta equilibrada');

si EsVacía (p) y correcto \rightarrow escribir ('está equilibrada')

no (EsVacía (p) y correcto) \rightarrow escribir ('no esta equilibrada');

faccion

EJERCICIO 3

a)

tipo

```
lista_circular = puntero a elemento;  
elemento = registro  
    numero:entero; (* tipo elemento *)  
    siguiente : lista_circular;  
fregistro;
```

ftipo

b)

tipo

```
pila = puntero a elemento;  
elemento = registro  
    numero:entero; (* tipo elemento *)  
    siguiente:pila;  
fregistro;
```

ftipo

c)

tipo

```
punt = puntero a elemento;  
elemento = registro  
    numero:entero (* tipo elemento *);  
    siguiente:punt;  
fregistro;
```

```
cola = registro  
    inicio,fin:punt;  
fregistro;
```

ftipo