

INTRODUCCIÓN A LA PROGRAMACIÓN

- Febrero 2002 -

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

Duración : 2 horas

Dispone de 45 min. para realizar preguntas sobre el enunciado.

Cada problema deberá resolverlo en hojas separadas.

1.- (3,5 puntos) Suponga que tiene las 48 cartas de una baraja desde el 1 al 12 de cada palo dispuestas en una cola, además tendremos 4 pilas (vacías) una por cada palo de la baraja. **Diseñe un algoritmo** que haciendo uso de los módulos que vienen a continuación simule el juego del solitario de la siguiente forma:

Se sacará una carta de la baraja y si es oros se pondrá en la pila de oros, pero siempre en orden del 1 al 12, es decir que primero se pondrá la carta del 1, después la carta del dos...

En el caso de que no se pudiera poner en la pila que corresponde a su palo se volvería a guardar en la cola de la baraja.

Todo este proceso se repite hasta que no quede ninguna carta en la baraja.

Se tienen disponibles los siguientes módulos:

Modulo Cartas;

exporta

tipo carta;

(* información sobre una carta de la baraja *)

función numero (c:carta): entero;

(* dada una carta c devolverá el numero de la misma, 1,2,3,4,5,6,7,8,9,10,11 o 12 *)

función palo (c:carta): carácter;

(* dada una carta c devolverá el palo que corresponde a la carta, O = oros, C = copas, E=espadas, B= bastos *)

fexporta

fmodulo

Modulo pilas;

importa Cartas fimporta

exporta

tipo pila;

(* para guardar cartas *)

acción inicializarp(var p: pila);

(* antes de poder sacar o meter cartas en p es obligatoria la llamada a esta acción, pone la pila p como si estuviera vacía *)

función vaciap (p:pila): booleano;

(* devolverá cierto si la pila p está vacía, en caso contrario devolverá falso *)

acción sacarp (var p: pila; var c: carta);

(* Pondrá en c la carta que lleva menos tiempo almacenada en p *)

acción meterp (var p: pila; c:carta);

(* guardará en la pila p la carta c *)

función topep (p: pila): carta;

(* devolverá la carta que lleva menos tiempo en la pila p, pero sin quitarla de p *)

fexporta

fmodulo

Modulo colas;
importa Cartas fimporta
exporta

tipo cola; (* para guardar cartas*)

acción inicializarc (var col : cola);

(* antes de poder sacar o meter cartas en col es obligatoria la llamada a esta acción , pone la cola col como si estuviera vacía *)

función vaciac (col:cola): booleano;

(* devolverá cierto si la cola col está vacía, en caso contrario devolverá falso*)

acción sacarc (var col: cola; var c: carta);

(* Pondrá en c la carta que lleva más tiempo almacenada en col *)

acción meterc (col: cola; c:carta);

(* guardará en la cola col la carta c *)

función topec (col: cola): carta;

(* devolverá la carta que lleva más tiempo en la cola col, pero sin quitarla de col *)

acción cargar cola (var col:cola);

(* cargará la cola col con todas las cartas de la baraja de forma aleatoria *)

fexporta

fmodulo

2.- (2 puntos) Suponga que tiene almacenada en una tabla una serie de números enteros. Diseñe una acción que permita calcular el mínimo número entero almacenado en una tabla, así como el número de veces que aparece dicho mínimo. La acción a implementar tendrá la siguiente cabecera:

acción minimo (t: datos; n:entero; var num:entero; var nveces: entero);

(* guarda en num el valor mínimo existente entre las casillas num y n (ambas inclusive) de la tabla t y guarda en nveces el número de veces que aparece dicho mínimo entre esas celdas *)

(*tipo datos = tabla[1..50] de entero *)

Ej: Supongamos que la tabla t tiene los siguientes elementos

20	7	1	9	22	7	9	7	14	7	3	7
1	2	3	4	5	6	7	8	9	10	11	12

Valor mínimo: 7
 Veces que aparece: 3

Para calcular el mínimo valor que hay entre las casillas 4 y 10 la llamada a realizar sería la siguiente:

m:=4;

minimo (t,10,m,nveces);

Una vez ejecutado m vale 7 y nveces vale 3

3.- (1.5 puntos) Implementar la acción "**Rellenar**" de manera que dada una tabla de 40 enteros, la rellene con la serie de números que se puede obtener a partir de la siguiente expresión:

$$T[n] = \begin{cases} T[n-1] + T[n-2] + n & \text{si } n > 2 \\ 2 & \text{si } n = 2 \\ 1 & \text{si } n = 1 \end{cases}$$

Un ejemplo de tabla (con 10 enteros) es el siguiente:

1	2	3	4	5	6	7	8	9	10
1	2	6	12	23	41	71	120	200	330

(1.5 puntos) Construir la función "**Comprobar_inversa**" de manera que dada una tabla de 40 enteros nos devuelva verdadero si contiene una serie construida mediante la expresión anterior, pero en orden inverso.

Un ejemplo de tabla (con 10 enteros) invertida es el siguiente:

1	2	3	4	5	6	7	8	9	10
330	200	120	71	41	23	12	6	2	1

INTRODUCCIÓN A LA PROGRAMACIÓN

SOLUCIÓN DEL EXAMEN

- Febrero 2002 -

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

PROBLEMA 1

algoritmo uno;

importa cartas, pilas, colas fimporta

var

baraja : cola;
oros, copas, espadas, bastos : pila;
car : carta;
pal: carácter

fvar

inicio

inicializarc (baraja);
cargar_cola (baraja);
inicializarp (oros);
inicializarp (copas);
inicializarp (espadas);
inicializarp (bastos);

mientras no vaciac(baraja) hacer

inicio

sacarc (baraja, car);
pal := palo(car);
caso pal de
"O" : meter_en_palo (oros, car, baraja)
"C" : meter_en_palo (copas, car, baraja)
"E" : meter_en_palo (espadas, car, baraja)
"B" : meter_en_palo (bastos, car, baraja)

fcaso;

fin;

falgoritmo

acción meter_en_palo (var p:pila; car: carta; var bar:baraja);

inicio

si vaciap(p) →

si numero(car)=1 → meterp(p,car)

numero(car)<> 1 → meterc(bar,car)

□ no vaciap(p) →

si numero(car) = numero(topep(p)) + 1 → meterp(p,car)

□ numero(car)<> numero(topep(p)) + 1 → meterc(bar,car);

facción

PROBLEMA 2

```
accion minimo(t: datos; n:entero; var num:entero; var nveces: entero);  
var  
    menor,i: entero;  
fvar  
inicio  
    menor:=t[num];  
    para i:= num+ 1 hasta n hacer  
        inicio  
            si t[i] < menor → menor:=t[i]  
            □ t[i] >= menor → nada;  
        fin  
  
    nveces:=0;  
    para i:= num hasta n hacer  
        inicio  
            si t[i]=menor → nveces:=nveces+1  
            □ t[i]<>menor → nada  
        fin;  
  
    num:=menor;  
faccion
```

PROBLEMA 3.

a)

```
accion rellenar(var t:tabla[1..40] de entero);  
var  
    n:entero;  
fvar  
inicio  
    t[1]:=1;  
    t[2]:=2;  
    para n:=3 hasta 40 hacer t[n]:= t[n-1] + t[n-2] + n;  
faccion
```

b)

```
funcion comprobar_inversa(t:tabla[1..40] de entero): booleano;  
var  
    n:entero;  
    tmpt:tabla[1..40] de entero;  
fvar  
inicio  
    rellenar(tmpt);  
    n:=1;  
    mientras (n<=40) y (tmpt[n]=t[41-n]) hacer n:=n+1;  
  
    retorna (n=41);  
ffuncion
```