

módulo tablas

exporta

constante

maxnumdatos=1000

tipo tabla

{Tabla de frecuencias de enteros. Implementación limitada a tablas con un tamaño máximo de 1000 números enteros distintos.}

procedimiento inicializar(**sal** t:tabla)

{Crea una tabla vacía t de frecuencias}

procedimiento añadir(**e/s** t:tabla; **ent** e:entero; **sal** error:booleano)

{Modifica t incrementando en 1 la frecuencia de e.

La implementación limita a 1000 el n° de datos distintos, por tanto, si e no cabe en la tabla, devuelve error=verdad.}

función total(t:tabla) **devuelve** natural

{Devuelve el n° de enteros distintos en la tabla t.}

procedimiento info(**ent** t:tabla; **ent** n:natural;

sal e:entero; **sal** m:natural; **sal** error:booleano)

{Devuelve en e el entero que ocupa el n-ésimo lugar en la tabla t, en orden de frecuencias decrecientes, y m es su frecuencia.

Si no existe ese entero (i.e. n=0 OR n>total(t)), devuelve 0 en ambos y error=verdad.}

implementación

tipos

dato = **registro**

número:entero;

frecuencia:natural

freg;

elementos = **vector**[1..maxnumdatos] **de** dato

tabla = **registro**

elmto:elementos;

total:natural;

freg

{Guarda los datos ordenados por frecuencias de mayor a menor.}

procedimiento inicializar(**sal** t:tabla)

{Crea una tabla vacía t de frecuencias}

principio

t.total:=0

fin

procedimiento añadir(**e/s** t:tabla; **ent** e:entero; **sal** error:booleano)

{Modifica t incrementando en 1 la frecuencia de e.

La implementación limita a maxnumdatos el n° de datos distintos, por tanto, si e no cabe en la tabla, devuelve error=verdad.}

variables

i:natural:=0;

éxito:booleano:=falso;

aux:entero

principio

{buscar e en t}

mientrasQue not éxito and i<t.total **hacer**

i:=i+1;

éxito:=t.elmto[i].número=e

fmq;

si éxito **entonces**

{e ya estaba -> incrementar su frecuencia y recolocar}

t.elmto[i].frecuencia:=t.elmto[i].frecuencia+1;

mientrasQue i>1 andThen t.elmto[i].frecuencia>t.elmto[i-1].frecuencia **hacer**

aux:=t.elmto[i].número;

```

    t.elmto[i].número:=t.elmto(i-1).número;
    t.elmto[i-1].número:=aux;
    aux:=t.elmto[i].frecuencia;
    t.elmto[i].frecuencia:=t.elmto[i-1].frecuencia;
    t.elmto[i-1].frecuencia:=aux
fmq;
error:=falso
sino
{e no estaba -> añadirlo al final de la tabla, si cabe}
si t.total<maxnumdatos entonces
    t.total:=t.total+1;
    t.elmto[t.total].número:=e;
    t.elmto[t.total].frecuencia:=1;
    error:=falso
sino
    error:=verdad
fsi
fsi
fin

función total(t:tabla) devuelve natural
{Devuelve el n° de enteros distintos en la tabla t.}
principio
    devuelve t.total
fin

procedimiento info(ent t:tabla; ent n:natural;
                    sal e:entero; sal m:natural; sal error:booleano)
{Devuelve en e el entero que ocupa el n-ésimo lugar en la tabla t,
 en orden de frecuencias decrecientes, y m es su frecuencia.
 Si no existe ese entero (i.e. total(t)<n), devuelve 0 en ambos y error=verdad.}
principio
si (n>0) and (n<=total(t)) entonces
    e:=t.elmto[n].número;
    m:=t.elmto[n].frecuencia;
    error:=falso
sino
    e:=0;
    m:=0;
    error:=verdad
fsi
fin

fin

```

Ejemplo de uso del módulo:

procedimiento estadistica

{Lee una secuencia de enteros de un fichero y escribe en pantalla cada entero distinto leído junto con su frecuencia de aparición, en orden de frecuencias decrecientes.

Si la tabla se llena se muestra un error.}

importa tablas, cadenas, ficheros

variables

```
f:fichero de entero;
nombre:cadena;
t:tabla;
dato:entero;
orden,frec:natural;
error:booleano:=falso
```

principio

```
escribir("Nombre del fichero: ");
leer(nombre);
asociar(f,nombre);
iniciarlectura(f);
inicializar(t);
mientrasQue not finFichero(f) and not error hacer
    leer(f,dato);
    añadir(t,dato,error)
fmq;
disociar(f);
si error
    escribir("Error por saturación de la capacidad de la tabla utilizada.")
sino
    para orden:=1 hasta total(t) hacer
        info(t,orden,dato,frec,error);
        escribir("entero: ", dato, " frecuencia: ", frec)
    fpara
fsi
fin
```