Algoritmos: Módulos y parámetros Arrays
Practica dirigida

Enunciado. Generar un orden de méritos (Mejores notas) de un aula de alumnos de la cual se conoce sus códigos, y cuyas calificaciones deben ser generadas en forma aleatoria en la escala de 05 a 20.

1. Diseño de Algoritmos: Pseudocodigo

```
Algoritmo. Orden de Merito
const M=7, N=2
tipo
    array[1..M;1..N] de entero: arrAlumno
    array[1..M] de entero: arrNota
var
   arrNota: notaAlu
   arrAlumno: alumno = {       //inicializar codigos de alumnos
                       {130, 0},
                       {111, 0},
                       {115, 0},
                       {100, 0},
                       {160, 0},
                       {120, 0},
                       {140, 0},
                       }
inicio
      //ingresar notas de alumnos usando numeros aleatorios
      llamar_a leerArray(alumno, M)
      //mostrar lista de alumnos con notas s/ordenar (datos originales)
      llamar_a escribirArray(alumno; M, N)

      //extraer notas de alumno en un vector de notas
      llamar_a copiarArray(alumno, notaAlu, M)
      //ordenar vector de notas
      llamar_a ordenar(notaAlu, M)
      //mostrar notas en orden ascendente
      llamar_a escribirArray(notaAlu, M)

      //mostrar orden merito: 3 mejores calificaciones (indice 7, 6, y 5)

      //buscar alumno que tiene maxima nota: indice 7
      llamar_a buscar(alumno, notaAlu[7], M)
      //buscar alumno que tiene maxima nota: indice 6
      llamar_a buscar(alumno, notaAlu[6], M)
      //buscar alumno que tiene maxima nota: indice 5
      llamar_a buscar(alumno, notaAlu[5], M)
fin


//definir prototipos
procedimiento leerArray(var arrAlumno: alu; E entero: items)
var
   entero: i
inicio
      desde(i<-1 hasta items)hacer
                alu[i,2] <- aleatorio() * 15+5
      fin_desde
fin_procedimiento
```

```
real funcion aleatorio()
var
   const RAND_MAX = 32767
inicio
      devolver (rand() / RAND_MAX)
fin_funcion


//extraer notas de la matriz de alumnos y copiarlos en un vector
procedimiento copiarArray(var arrAlumno: origen, arrNota: destino;
                                                          E entero: filas)
var
   entero: i
inicio
      desde(i<-1 hasta filas)hacer
            destino[i] <- origen[i][2]
      fin_desde
fin_procdimiento


//subrutina: escribirArray - vector
procedimiento escribirArray(var arrNota: nota; E entero: items)
var
   entero: i
inicio
      desde(i<-1 hasta items)hacer
            escribir(nota[i])
      fin_desde
fin_procedimiento


//sobrecarga de subrutina escribirArray - matriz
procedimiento escribirArray(var arrAlumno: alu; E entero: fila, columa)
var
   entero: i, j
inicio
      desde(i<-1 hasta fila)hacer
            desde(j<-1 hasta columna)hacer
                  escribir(alu[i,j])
            fin_desde
      fin_desde
fin_procedimiento


//ordenar vector de notas
procedimiento ordenar(var arrNota: nota; E entoero: items)
var
      entero: i, j
inicio
      desde(i<-1 hasta items-1)hacer
            desde(j<- hasta items-i)hacer
                  si(nota[j] > nota[j+1])entonces
                      llamar_a intercambio(nota[j], nota[j+1])
                  fin_si
            fin_desde
      fin_desde
fin_procedimiento
```

```
//proceso de intercambio
procedimiento intercambio(S entero: a, b)
var
   entero: aux
inicio
      aux <- a
      a <- b
      b <- aux
fin_procedimiento


//sub-rutina buscar nota de alumno unica o nota repetida
procedimiento buscar(var arrAlumno: alu; E entero: nota, filas)
var
   entero: i, k
inicio
      k=0
      desde(i<-1 hasta filas)hacer
            si(alu[i,2] = nota)entonces
                  escribir("Alumno: ", alu[i,1])
                  escribir("Nota  : ", alu[i,2])
                  k <- k+1
            fin_si
      fin_desde
      escribir("Encontrados: ", k)
fin_procedimiento

//sobrecarga de subrutina: buscar el primero que se encuentra
entero funcion buscar(var arrAlumno: alu; E entero: nota, filas)
var
   entero: i, k
inicio
      k = -1
      desde(i <- 1 hasta filas)hacer
            si( alumno[i,2] = nota)entonces
                k <- i
            fin_si
      fin_desde
      devolver (k)
fin_funcion
```

2. Comprobación de Algoritmos: C++

```
//Algoritmo. Orden de Merito
#include <cstdlib>
#include <iostream>

void leerArray( int [][2], int );
float aleatorio(void);
void copiarArray(int [][2], int [], int );
void ordenar(int [], int );
void intercambio(int *, int *);
void escribirArray( int [], int );
void escribirArray( int [][2], int, int );
```

```
void buscar(int [][2], int, int );

int main(int argc, char *argv[]){
const int M=7, N=2;
/*tipo
    array[1..M;1..N] de entero: arrAlumno
    array[1..M] de entero: nota
var
*/
   int notaAlu[M];
   int alumno[][N] = {
                        {130, 0},
                        {111, 0},
                        {115, 0},
                        {100, 0},
                        {160, 0},
                        {120, 0},
                        {140, 0}
                        };
//inicio
      //generar verdaderos aleatorios:
      srand(time(0));
      //ingresar notas de alumnos usando numeros aleatorios
      //llamar_a
      leerArray(alumno, M);
      //mostrar lista de alumnos con notas s/ordenar (datos originales)
      puts("Listado alumnos:");
      //llamar_a
      escribirArray( alumno, M, N);

      //extraer notas de alumno en un vector de notas
      //lamar_a
      copiarArray(alumno, notaAlu, M);
      //ordenar vector de notas
      //llamar_a
      ordenar(notaAlu, M);
      //mostrar notas en orden ascendente
      puts("\nNotas en orden ASC:");
      //llamar_a
      escribirArray(notaAlu, M);

      //mostrar orden de merito: 3 mejores calificaciones (indice 7, 6, y 5)
      puts("\nOrden de Merito:");
      //buscar alumno que tiene  1ra maxima nota: indice 7
      //llamar_a
      buscar(alumno, notaAlu[M-1], M);
      //buscar alumno que tiene 2da maxima nota: indice 6
      //llamar_a
      buscar(alumno, notaAlu[M-2], M);
      //buscar alumno que tiene 3ra maxima nota: indice 5
      //llamar_a
      buscar(alumno, notaAlu[M-3], M);

      system("PAUSE");
      return EXIT_SUCCESS;
}//fin
```

```
//definir prototipos: importante - Los array en C++ son base 0
void leerArray( int alu[][2], int items){
//var
   int i;
//inicio
      for(i=0; i < items; i++){//hacer
                alu[i][1] = int(aleatorio() * 15+5);
      }//fin_desde
}//fin_procedimiento


float aleatorio(void){
//var
   //const int RAND_MAX = 32767;
//inicio
      return (float(rand()) / RAND_MAX);
}//fin_funcion


void copiarArray(int origen[][2], int destino[], int filas){
//var
   int i;
//inicio
      for(i=0; i < filas; i++){//hacer
            destino[i] = origen[i][1];
      }//fin_desde
}//fin_procedimiento


//subrutina: escribirArray - vector
void escribirArray(int nota[], int items){
//var
   int i;
//inicio
      for(i=0; i < items; i++){//hacer
          printf("elemento: %d\n", nota[i]);
      }//fin_desde
}//fin_procedimiento


//sobrecarga de subrutina escribirArray - vector
void escribirArray( int alu[][2], int fila, int columna){
//var
   int i, j;
//inicio
      for(i=0; i < fila; i++){//hacer
          printf("elemento: ");
          for(j=0; j < columna; j++){//hacer
                printf("%d ", alu[i][j]);
          }//fin_desde
          std::cout << std::endl;
      }//fin_desde
}//fin_procedimiento
```

```
void ordenar(int nota[], int items){
//var
      int i, j;
//inicio
      for(i=0; i < items-1; i++){//hacer
          for(j=0; j < items-1; j++){//hacer
                  if(nota[j] > nota[j+1]){//entonces
                      //llamar_a
                      intercambio(&nota[j], &nota[j+1]);
                  }//fin_si
          }//fin_desde
      }//fin_desde
}//fin_procedimiento



void intercambio(int *a, int *b){
//var
    int aux;
//inicio
      aux = *a;
      *a = *b;
      *b = aux;
}//fin_procedimiento



//sub-rutina buscar nota de alumno unica o nota repetida
void buscar(int alu[][2], int nota, int filas){
//var
   int i, k;
//inicio
      k=0;
      for(i=0; i < filas; i++){//hacer
          if( alu[i][1] == nota){//entonces
              printf("Alumno: %d ", alu[i][0]);
              printf("Nota  : %d\n", alu[i][1]);
              k = k+1;
          }//fin_si
      }//fin_desde
      printf("Encontrados: %d\n", k);
}//fin_procedimiento
```