EXAMEN DE SISTEMAS INFORMÁTICOS INDUSTRIALES
(SOLUCIÓN PRÁCTICAS)

La duración del examen es de 2 horas**.** **SEPTIEMBRE 2016**

En este examen se solicita al alumno realizar una función para ordenar un vector en C y a continuación, mediante el uso de clases en C++, diseñar una serie de métodos para solicitar datos, ordenarlos y mostrar el mayor y menor número del vector.

1. En primer lugar se pide escribir un programa en C que permita ordenar un vector mediante el método de la burbuja. Esta función debe completarse dentro del fichero clase.cpp donde ya se encuentra su prototipo. Para poder probar su funcionamiento se proporciona ya un código en la opción 1 del menú del main (examen.cpp). El pseudocódigo del método de ordenación burbuja se muestra en la siguiente hoja.

   (4.00 puntos)
2. A continuación se solicita al alumno que complete el constructor, destructor y diferentes métodos de la clase cuya definición se encuentra en el fichero clase.h (que no debe modificarse). En el fichero *examen.cpp* se proporciona un menú ya realizado con las diferentes opciones pero con las llamadas a los métodos comentadas. Usar dicho menú para comprobar el funcionamiento el programa.
   a. Constructor y destructor. Deben inicializarse las variables de la clase y liberar memoria en caso de ser necesario.

      (0.75 puntos)
   b. Método *introducir_datos*. Se solicitará al usuario el número de datos enteros que va a introducir almacenandolos en *longitud_vector*. A continuación se reservará dinámicamente memoria para almacer dichos datos en *vector*. Y finalmente se pedirán los datos para almacenarlos en el vector.

      (2.00 puntos)
   c. Método *ordenar*. Se ordenará el vector llamando a la función realizada en el apartado 1 y se indicará en la variable *ordenado*. Si no hay vector que ordenar se indicará por pantalla.

      (1.00 puntos)
   d. Métodos *mayor* y *menor*. Estos métodos devolverán el mayor y menor número del vector únicamente si el vector contiene datos y está ordenado. En caso contrario se indicará por pantalla y se devolverá *NULL*.

      (1.50 puntos)
   e. Método liberar. Inicializará de nuevos las variables para poder volver a introducir un nuevo vector de datos.

      (0.75 puntos)

El programa debe compilar y se valorará su funcionamiento así como la claridad del código y los comentarios.

Pseudocodigo del método de ordenación burbuja:

```
ALGORITMO burbuja
para i=1 hasta N-1
        para j=N-1 hasta i
                si (v[j-1] > v[j])
                        intercambiar v[j-1] y v[j]
                fsi
        fpara
fpara
FIN ALGORITMO
```

Se facilita el código del fichero **clase.h** que no debe modificarse:

```
#ifndef __CLASE_H__
#define __CLASE_H__

void ordenar_burbuja(int vec[], int n);

class clase{
      protected:
                int *vector;
                int longitud_vector;
                int ordenado;
      public:
                 clase();//Constructor
                 ~clase(void); //Destructor
                 void introducir_datos();
                 void ordenar();
                 int mayor();
                 int menor();
                 void liberar();
};
#ifndef    __CLASE_CPP__
#else
#endif
#endif
```

Se facilita el código del fichero **clase.cpp** que el alumno deber completar.

```
#define __CLASE_CPP__
#include <iostream>
#include <math.h>
#include "clase.h"
#include <cstdlib>
#include <stdio.h>
#include <string.h>
#include <locale.h>
using namespace std;

void ordenar_burbuja(int vec[], int n) {
}
clase::clase() {
}
clase::~clase(void) {
}
```

Se facilita el código del fichero **examen.cpp** que el alumno debe completar:

```
#define __EXAMEN_CPP__
#include <iostream>
#include "clase.h"
#include <cstdlib>
#include <stdio.h>
#include <string.h>
#include <locale.h>
using namespace std;
int main(void){
    int opcion_menu,i;
    int vec[7];
    clase miclase;
do {
 printf("\n- MENU -\n--------\n");
 printf("1: Probar la funcion en C del algoritmo de ordenacion\n");
 printf("2: Solicitar numeros desde la clase\n");
 printf("3: Ordenar vector de numeros desde la clase\n");
 printf("4: Mostrar numero mayor del vector\n");
 printf("5: Mostrar numero menor del vector\n");
 printf("6: Liberar clase\n");
 printf("0: SALIR\n");
 printf("Introduzca opcion del menu: ");
 scanf("%d",&opcion_menu);
 switch(opcion_menu){
        case 1:
            printf("Introduzca 7 valores enteros:\n");
            for (i=0; i<7; i++) {
                    printf("Introduce el elemento %d: ",i);
                    scanf("%d", &vec[i]);
              }
            ordenar_burbuja(vec,7);
            printf("Vector ordenado.\n");
            for (i=0; i<7; i++) {
                    printf("Elemento %d: %d\n",i,vec[i]);
                }
            break;
        case 2:
            //printf("Solicitando numeros para el vector.\n");
            //miclase.introducir_datos();
            break;
        case 3:
            //printf("Ordenando vector.\n");
            //miclase.ordenar();
            break;
        case 4:
            //printf("El numero mas grande del vector es:
%d\n",miclase.mayor());
            break;
        case 5:
            //printf("El numero mas pequeño del vector es:
%d\n",miclase.menor());
            break;
        case 6:
            //miclase.liberar();
            //printf("La memoria ha sido liberada y ya puede pedirse un nuevo
vector.\n");
            break;
        default:
            break;
                }
    } while(opcion_menu!=0);
}
```

Ejemplo de funcionamiento:

```
examen.exe

- MENU -
--------
1: Probar la funcion en C del algoritmo de
ordenacion
2: Solicitar numeros desde la clase
3: Ordenar vector de numeros desde la clase
4: Mostrar numero mayor del vector
5: Mostrar numero menor del vector
6: Liberar clase
0: SALIR
Introduzca opcion del menu: 1
Introduzca 7 valores enteros:
Introduce el elemento 0: 5
Introduce el elemento 1: 9
Introduce el elemento 2: 1
Introduce el elemento 3: 3
Introduce el elemento 4: 8
Introduce el elemento 5: 3
Introduce el elemento 6: 4
Vector ordenado.
Elemento 0: 1
Elemento 1: 3
Elemento 2: 3
Elemento 3: 4
Elemento 4: 5
Elemento 5: 8
Elemento 6: 9
```

```
examen.exe
- MENU -
--------
1: Probar la funcion en C del algoritmo de
ordenacion
2: Solicitar numeros desde la clase
3: Ordenar vector de numeros desde la clase
4: Mostrar numero mayor del vector
5: Mostrar numero menor del vector
6: Liberar clase
0: SALIR
Introduzca opcion del menu: 2
Solicitando numeros para el vector.
Cuantos numeros enteros va a introducir:
4
Introduce el elemento 1: 8
Introduce el elemento 2: 1
Introduce el elemento 3: 7
Introduce el elemento 4: 4

- MENU -
--------
1: Probar la funcion en C del algoritmo de
ordenacion
2: Solicitar numeros desde la clase
3: Ordenar vector de numeros desde la clase
4: Mostrar numero mayor del vector
5: Mostrar numero menor del vector
6: Liberar clase
0: SALIR
Introduzca opcion del menu: 4
El vector no esta ordenado.
El numero mas grande del vector es: 0

- MENU -
--------
1: Probar la funcion en C del algoritmo de
ordenacion
2: Solicitar numeros desde la clase
3: Ordenar vector de numeros desde la clase
4: Mostrar numero mayor del vector
5: Mostrar numero menor del vector
6: Liberar clase
0: SALIR
Introduzca opcion del menu: 5
El vector no esta ordenado.
El numero mas pequeño del vector es: 0

- MENU -
--------
1: Probar la funcion en C del algoritmo de
ordenacion
2: Solicitar numeros desde la clase
3: Ordenar vector de numeros desde la clase
4: Mostrar numero mayor del vector
5: Mostrar numero menor del vector
6: Liberar clase
0: SALIR
Introduzca opcion del menu: 3
Ordenando vector.
```

```

- MENU -
--------
1: Probar la funcion en C del algoritmo de
ordenacion
2: Solicitar numeros desde la clase
3: Ordenar vector de numeros desde la
clase
4: Mostrar numero mayor del vector
5: Mostrar numero menor del vector
6: Liberar clase
0: SALIR
Introduzca opcion del menu: 4
El numero mas grande del vector es: 8

- MENU -
--------
1: Probar la funcion en C del algoritmo de
ordenacion
2: Solicitar numeros desde la clase
3: Ordenar vector de numeros desde la
clase
4: Mostrar numero mayor del vector
5: Mostrar numero menor del vector
6: Liberar clase
0: SALIR
Introduzca opcion del menu: 5
El numero mas pequeño del vector es: 1

- MENU -
--------
1: Probar la funcion en C del algoritmo de
ordenacion
2: Solicitar numeros desde la clase
3: Ordenar vector de numeros desde la
clase
4: Mostrar numero mayor del vector
5: Mostrar numero menor del vector
6: Liberar clase
0: SALIR
Introduzca opcion del menu: 6
La memoria ha sido liberada y ya puede
pedirse un nuevo vector.

- MENU -
--------
1: Probar la funcion en C del algoritmo de
ordenacion
2: Solicitar numeros desde la clase
3: Ordenar vector de numeros desde la
clase
4: Mostrar numero mayor del vector
5: Mostrar numero menor del vector
6: Liberar clase
0: SALIR
Introduzca opcion del menu: 0
```

Solución:

examen.cpp

```
#define __EXAMEN_CPP__
#include <iostream>
#include "clase.h"
#include <cstdlib>
#include <stdio.h>
#include <string.h>
#include <locale.h>
using namespace std;

int main(void){
    int opcion_menu,i;
    int vec[7];
    clase miclase;
do
 {
 printf("\n- MENU -\n--------\n");
 printf("1: Probar la funcion en C del algoritmo de ordenacion\n");
 printf("2: Solicitar numeros desde la clase\n");
 printf("3: Ordenar vector de numeros desde la clase\n");
 printf("4: Mostrar numero mayor del vector\n");
 printf("5: Mostrar numero menor del vector\n");
 printf("6: Liberar clase\n");
 printf("0: SALIR\n");
 printf("Introduzca opcion del menu: ");
 scanf("%d",&opcion_menu);
 switch(opcion_menu){
        case 1:
            printf("Introduzca 7 valores enteros:\n");
            for (i=0; i<7; i++) {
                      printf("Introduce el elemento %d: ",i);
                      scanf("%d", &vec[i]);
               }
            ordenar_burbuja(vec,7);
            printf("Vector ordenado.\n");
            for (i=0; i<7; i++) {
                      printf("Elemento %d: %d\n",i,vec[i]);
                }
            break;
        case 2:
            printf("Solicitando numeros para el vector.\n");
            miclase.introducir_datos();
            break;
        case 3:
            printf("Ordenando vector.\n");
            miclase.ordenar();
            break;
        case 4:
            printf("El numero mas grande del vector es: %d\n",miclase.mayor());
            break;
        case 5:
            printf("El numero mas pequeño del vector es: %d\n",miclase.menor());
            break;
        case 6:
            miclase.liberar();
            printf("La memoria ha sido liberada y ya puede pedirse un nuevo vector.\n");
            break;
        default:
            break;
                }
    } while(opcion_menu!=0);
}
```

clase.cpp

```
#define __CLASE_CPP__
#include <iostream>
#include <math.h>
#include "clase.h"
#include <cstdlib>
#include <stdio.h>
#include <string.h>
#include <locale.h>
```

```
using namespace std;

void ordenar_burbuja(int vec[], int n)
{
        int aux;
        int i,j;

        for (i=1; i<n; i++)
                for (j=n-1; j>=i; j--)
                        if (vec[j-1] > vec[j])
                        {
                                aux = vec[j-1];
                                vec[j-1] = vec[j];
                                vec[j] = aux;
                        }
}

clase::clase(){ // Constructor
longitud_vector=0;
ordenado=0;
vector=NULL;
}
clase::~clase(void)
{
    if(vector!=NULL)
        delete []vector;
}

void clase::introducir_datos()
{
    int i;
    printf("Cuantos numeros enteros va a introducir: \n");
    scanf("%d",&longitud_vector);
    vector = (int *)malloc(longitud_vector*sizeof(int));
    for(i=0;i<longitud_vector;i++)
    {
        printf("Introduce el elemento %d: ",i+1);
        scanf("%d", &vector[i]);
    }
}
void clase::ordenar()
{
    if(longitud_vector!=0)
        {
            ordenar_burbuja(vector,longitud_vector);
            ordenado=1;
        }
    else
        printf("No hay vector que ordenar.\n");
}
int clase::mayor()
{
    if(longitud_vector!=0 && ordenado==1)
        return(vector[longitud_vector-1]);
    else
    {
        printf("El vector no esta ordenado.\n");
        return(NULL);
    }
}
int clase::menor()
{
    if(longitud_vector!=0 && ordenado==1)
        return(vector[0]);
    else
     {
        printf("El vector no esta ordenado.\n");
        return(NULL);
     }
}
void clase::liberar(){
    longitud_vector=0;
    ordenado=0;
    if(vector!=NULL)
        delete []vector;
}
```