Nombre:	
C.I.:	

TECNÓLOGO EN INFORMÁTICA

PRINCIPIOS DE PROGRAMACIÓN

Segundo Parcial

2009

Ejercicio 1 (16 puntos)

1) ¿Cuantas funciones main() puede tener un programa C?
2) ¿Cuál es el propósito de una sentencia return?
3) ¿Qué le dice la siguiente declaración al compilador?

float complex_magnitude (float freal, float fimag)

4) ¿Porqué se necesitan punteros para intercambiar valores usando una función?

Ejercicio 2 (14 puntos)

Considere el siguiente programa:

```
int main ()
{
        int x=1;int y=10;
        while (x >=0 && y >=5)
                if (x % 2 == 1)
                {   x++;
                    y--;
                }
        return 0;
}
```

Se pide:

1) **Escribir un programa equivalente utilizando la estructura de control do-while en lugar de while.**
2) **Escribir un programa equivalente utilizando la estructura de control for en lugar de while.**

Ejercicio 3 (30 puntos)

Se desea crear un programa para manejar y mostrar ciertos datos de un Video Club.

El programa manejara la información referente a socios y a películas.

El sistema permitirá dar de alta socios, de los cuales interesa conocer el nombre, el apellido, la cédula de identidad y un número de socio.

También el sistema deberá permitir dar de alta películas. La información referente a las películas que interesa saber son el titulo y un número de película.

El programa deberá dar una opción para que el usuario pueda listar a todos los socios del sistema ordenados por número de socio (de forma ascendente), y otra opción en la que pueda listar todas las películas disponibles ordenadas de forma decreciente según el nombre de la película.

Asuma que todos los datos de tipo cadena de caracteres son ingresados en mayúscula.

El siguiente es el menú principal del programa y muestra las siguientes opciones:

1-Alta Socio
2-Alta película
3-Listar Socios
4-Listar Películas
5-Salir

Cada vez que se elija una opción, el sistema deberá de responder al usuario con las acciones que correspondan y luego retornar al menú principal.
El sistema podrá almacenar una cantidad máxima de socios dada por la constante **MAX_SOCIOS**, y una cantidad máxima de películas dada por **MAX_PELICULAS**. Todos los datos que se declaren de tipo cadena de caracteres tendrán una longitud máxima de largo **MAX_CADENA**. El sistema deberá impedir que se den de alta más socios o más películas que la cantidad máxima permitida.

Se pide:

a) Definir en c los tipos Socio y Película utilizando estructuras.

```
struct Socio{                    struct Pelicula{
….                                    …
}                                 }
```

b)Escribir la función
insOrd(struct Socio socios[],int cantsocios, struct Socio nuevoSocio);
/* Precondición: el arreglo socios debe estar ordenado de forma creciente según el campo numero socio de la estructura.
Poscondición: Se devuelve el arreglo socios ordenado en forma creciente según el número de socio conteniendo al nuevo socio
*/

Que recibe como parámetros un arreglo de tipo Socio con un tamaño igual a MAX_SOCIOS,
un entero cantsocios con la cantidad de socios que tiene el arreglo socios,
(En caso de que cantsocios==MAX_SOCIOS, la función no debe hacer ninguna modificación sobre el arreglo.)
y por ultimo el parámetro nuevoSocio, que es el nuevo socio que será insertado (en caso de que cantsocios<MAX_SOCIOS) de forma ordenada en el arreglo socios.
La función inserta de forma ordenada (ordenada de forma creciente según el numero de socio) en el arreglo de socios al nuevo socio.

c) Escribir en pseudocódigo un algoritmo que resuelva completamente todo el programa descrito al comienzo del ejercicio.

d) Escribir completamente el programa que resuelva el problema planteado al comienzo del ejercicio.

Nota:
-Se pueden hacer uso de las funciones y estructuras de las partes anteriores.
-Se puede utilizar la función
void ordenarCrecientePeliculas(struct Pelicula peliculas[]);
que ordena de forma decreciente un arreglo de películas según el campo nombre de la estructura películas.
- Para la parte d) se pueden utilizar funciones y procedimientos auxiliares, pero todos ellos deben ser implementados.

Soluciones

Ej2)
a)

```
int main ()
{
        int x=1;int y=10;
       if(x >=0 && y >=5){
         do{
                if (x % 2 == 1)
                {  x++;
                   y--;
                }

         } while (x >=0 && y >=5)
        }//fin si

        return 0;
}
```

b)

```
int main ()
{
        int x=1;int y=10;
        for (;x >=0 && y >=5;)
                if (x % 2 == 1)
                {  x++;
                   y--;
                }
```

```
        return 0;
}
```

Ej3)

```
#include <cstdlib>
#include <iostream>

using namespace std;
//DECLARACION DE CONSTANTES
const int MAX_CADENA=30;
const int MAX_SOCIOS=30;
const int MAX_PELICULAS=30;
const int MAX_CEDULA=10;

//DECLARACIÓN DE TIPOS
typedef struct Socio{
   char nombre[MAX_CADENA];
   char apellido[MAX_CADENA];
   char ci[MAX_CEDULA];
   unsigned int nro;

};


typedef struct Pelicula{
   char titulo[MAX_CADENA];
   unsigned int nro;
};

//Prototipos Funciones Auxiliares
void insOrd(Socio socios[],int cantsocios, Socio nuevoSocio);
/* Precondición: el arreglo socios debe estar ordenado de forma creciente según el
campo numero socio de la estructura.
Poscondición: Se devuelve  el arreglo socios ordenado en forma creciente según el
número de socio conteniendo al nuevo socio
*/

void ordenarCrecientePeliculas(Pelicula peliculas[], int cantpeliculas);
//ordena de forma creciente un arreglo de películas
//según el campo nombre de la estructura películas.

void altaSocio(Socio[],int &cantsocios);
void altaPelicula(Pelicula[],int &cantpeliculas);
void listarSocios(Socio[],int cantsocios);
void listarPeliculas(Pelicula[], int cantpeliculas);


//Definicion de Funciones
void burbujaCrecienteSocios(Socio s[], int tam_arreglo)
  {
```

```
   //Ordena de forma ascendente
   int temp2, j, i;
   Socio temp;
   for(i = (tam_arreglo-1); i >= 0; i--)
   {
    temp2= (tam_arreglo-1);
    for(j = 1; j <= temp2; j++)
    {
     if(s[j-1].nro > s[j].nro)
     {

      temp = s[j-1];
      s[j-1] = s[j];
      s[j] = temp;
     }
    }
   }
  }

//Definicion de Funciones
void burbujaCrecientePeliculas(Pelicula s[], int tam_arreglo)
  {
   //Ordena de forma ascendente
   int temp2, j, i;
   Pelicula temp;
   for(i = (tam_arreglo-1); i >= 0; i--)
   {
    temp2= (tam_arreglo-1);
    for(j = 1; j <= temp2; j++)
    {
     if(strcmp(s[j-1].titulo,s[j].titulo)>0)
     {

      temp = s[j-1];
      s[j-1] = s[j];
      s[j] = temp;
     }
    }
   }
  }



void ordenarCrecientePeliculas(Pelicula peliculas[], int cantpeliculas){
    burbujaCrecientePeliculas(peliculas,cantpeliculas);
}

void  insOrd(Socio socios[],int cantsocios, Socio nuevoSocio){
        socios[cantsocios++]=nuevoSocio;
        burbujaCrecienteSocios(socios,cantsocios);

 }
```

```
void altaSocio(Socio socios[],int &cantsocios){
    if(cantsocios<MAX_SOCIOS){
        Socio s;
        printf("Nombre\n");
        scanf("%s",&s.nombre);
        //fflush(stdin);
        printf("Apellido\n");
        scanf("\n%s",&s.apellido);
        printf("Cedula\n");
        scanf("\n%s",&s.ci);
        printf("Nro. Socio\n");
        scanf("%d",&s.nro);
        insOrd(socios,cantsocios,s);
        cantsocios++;
    }
}

void altaPelicula(Pelicula peliculas[],int &cantpeliculas){
    if(cantpeliculas<MAX_PELICULAS){
        Pelicula p;
        printf("Titulo\n");
        scanf("%s",&p.titulo);
        //fflush(stdin);
        printf("Nro.\n");
        scanf("%d",&p.nro);
        peliculas[cantpeliculas]=p;
        cantpeliculas++;
    }
}

void listarSocios(Socio socios[],int cantsocios){
   if(cantsocios>0){
    for(int i=0;i<cantsocios;i++){
      printf("%s\n",socios[i].nombre);
      printf("%s\n",socios[i].apellido);
      printf("%s\n",socios[i].ci);
      printf("%d\n",socios[i].nro);
      printf("----------------------------------------\n");
    }
    fflush(stdin);
    char c;
    scanf("%c",&c);//para pode leer el listado
   }
}

void listarPeliculas(Pelicula peliculas[], int cantpeliculas){
    if(cantpeliculas>0){
     ordenarCrecientePeliculas(peliculas,cantpeliculas);
     for(int i=(cantpeliculas-1);i>=0;i--){
         printf("%s\n",peliculas[i].titulo);
         printf("%d\n",peliculas[i].nro);
         printf("----------------------------------------\n");
     }
```

```
    fflush(stdin);
    char c;
    scanf("%c",&c);//para pode leer el listado
   }

}

//Programa Principal

int main(int argc, char *argv[])
{
   int cantsocios=0,cantpeliculas=0;
   Socio socios[MAX_SOCIOS];
   Pelicula peliculas[MAX_PELICULAS];
   char opcion;
   do{
    system("cls");
    printf("1-Alta Socio\n2-Alta pelicula\n3-Listar Socios\n4-Listar Peliculas\n5-
Salir\n");
    fflush(stdin);
    scanf("%c",&opcion);
    switch(opcion){
        case '1': altaSocio(socios,cantsocios);break;
        case '2': altaPelicula(peliculas,cantpeliculas);break;
        case '3': listarSocios(socios,cantsocios);break;
        case '4': listarPeliculas(peliculas,cantpeliculas);break;
        case '5': break;
        //default: system("cls");printf("Comando Incorrecto.\n");
    }

   }while(opcion!='5');
   return EXIT_SUCCESS;
}
```