|  |  |
| --- | --- |
|  | **ESCUELA POLITÉCNICA NACIONAL - ESFOT**  **PROGRAMACIÓN. TECNOLOGÍA SUPERIOR EN DESARROLLO DE SOFTWARE.**  **Examen 2B**  **Quito, 14 de septiembre de 2021.**  **NOMBRE:JEAN TITUAÑA** |

**Instrucciones:**

Duración: Este examen es individual y tiene una duración de dos horas y treinta minutos.

Instrucciones:

Envío: Se realizará en este archivo de Microsoft Word, los problemas pueden ser realizado en Visual Studio, Visual Studio Code o Repl.it, pero se debe copiar el código en este documento, y enviarlo como archivo **de Word** y enviar a través de la tarea del aula virtual antes de que se cumplan el tiempo desde del inicio de la prueba.

El problema 1 y 2 valen 0.5 puntos y el problema 3 vale 1 punto.

**Problemas:**

**1**. Una pequeña aerolínea acaba de comprar una computadora para su nuevo sistema de reservaciones automatizado. Se le ha pedido a usted que desarrolle el nuevo sistema. Usted va a escribir una aplicación para asignar asientos en cada vuelo del único avión de la aerolínea (capacidad: 10 asientos). Su programa debe mostrar el siguiente menú de alternativas: Por favor escriba 1 para "Primera Clase" y Por favor escriba 2 para "Económico". Si la persona escribe 1, su programa debe asignarle un asiento en la sección de primera clase (asientos 1 a 5). Si la persona escribe 2, su programa debe asignarle un asiento en la sección económica (asientos 6 a 10). Su programa deberá entonces imprimir un pase de abordaje, indicando el número de asiento de la persona y si se encuentra en la sección de primera clase o económica del avión.

Use un arreglo unidimensional para representar la tabla de asientos del avión. Inicialice todos los elementos del arreglo con 0 para indicar que todos los asientos están vacíos. A medida que se asigne cada asiento, establezca los elementos correspondientes del arreglo en 1 para indicar que ese asiento ya no está disponible.

Desde luego que su programa nunca deberá asignar un asiento que ya haya sido asignado. Cuando esté llena la sección de primera clase, su programa deberá preguntar a la persona si acepta ser colocada en la sección económica (y viceversa). Si la persona acepta, haga la asignación de asiento apropiada. Si no acepta, imprima el mensaje "El próximo vuelo sale en 3 horas".

#include <iostream>

using namespace std;

const int N = 10;

void primeraclase();

void clase\_economica();

void primeraclase\_llena();

void economico\_lleno();

bool economico\_disponibles();

bool primera\_disponibles();

void MostrarSillas();

int asientos[N] = {};

int primera = 0;

int economico = 5;

bool ok = true;

int main()

{

int clase;

while(ok)

{

cout << "Sistema de Reservaciones" << endl;

cout << "Escriba 1 para Primera Clase " << endl;

cout << "Escriba 2 para Economico " << endl;

cout << "Escriba 3 para Mostrar asientos " << endl;

cout << "Escriba 4 para Salir " << endl;

cin >> clase;

if(clase == 1)

{

primeraclase();

}

else if(clase == 2)

{

clase\_economica();

}

else if(clase == 3)

{

cout << "Asientos " << endl;

cout << "0 Vacio " << endl;

cout << "1 Lleno " << endl;

MostrarSillas();

}

else if(clase == 4)

{

ok = false;

}

else{

cout << "Opcion No valida! " << endl;

}

}

return 0;

}

void primeraclase()

{

if(primera < 5)

{

asientos[primera] = 1;

cout << "Asiento asignado " << endl;

}

else{

primeraclase\_llena();

}

primera++;

}

void clase\_economica()

{

if(economico < 10)

{

asientos[economico] = 1;

cout << "Asiento asignado" << endl;

}

else{

economico\_lleno();

}

economico++;

}

void primeraclase\_llena()

{

int opcion;

bool disponible;

bool exito = true;

while(exito)

{

cout << "Primera Clase Llena! " << endl;

cout << "Desea ser colocado en la clase economica" << endl;

cout << "1.-Si" << endl;

cout << "2.-No" << endl;

cin >> opcion;

if(opcion == 1)

{

disponible = economico\_disponibles();

if(disponible == false)

{

clase\_economica();

}

else{

cout << "Primera Clase LLena!" << endl;

cout << "Proximo vuelo en 3 horas " << endl;

ok = true;

exito = false;

}

}

if(opcion == 2)

{

cout << "Proximo vuelo en 3 horas" << endl;

exito = false;

}

}

}

void economico\_lleno()

{

int opcion;

bool disponible;

bool exito = true;

while(exito)

{

cout << "Economico Lleno! " << endl;

cout << "Desea ser colocado en la primera clase" << endl;

cout << "1.-Si" << endl;

cout << "2.-No" << endl;

cin >> opcion;

if(opcion == 1)

{

disponible = primera\_disponibles();

if(disponible == false)

{

primeraclase();

}

else{

cout << "Clase Economica LLena!" << endl;

cout << "Proximo vuelo en 3 horas " << endl;

ok = true;

exito = false;

}

}

if(opcion == 2)

{

cout << "Proximo vuelo en 3 horas" << endl;

ok = true;

exito = false;

}

}

}

bool economico\_disponibles()

{

bool Lugar = true;

for(int i = 5; i < 10; i++)

{

if(asientos[i] == 0)

{

        Lugar = false;

    }

  }

return Lugar;

}

bool primera\_disponibles()

{

bool Lugar = true;

for(int i = 0; i < 5; i++)

{

if(asientos[i] == 0)

{

        Lugar = false;

    }

  }

return Lugar;

}

void MostrarSillas()

{

cout << endl;

for(int i=0 ; i<N ; i++)

{

cout << "[" << asientos[i] << "]";

}

cout << endl << endl;

}

**2.** Una fábrica de tornillos nos ha solicitado desarrollar un programa que permita obtener e imprimir las estadísticas de una determinada muestra de tornillos. Para ello nos indican que cada tornillo viene caracterizado por su longitud y su diámetro y que ambas medidas deben almacenarse en un registro con dos campos. Desarrollar un programa en C que:

a) Solicite al usuario los datos de una muestra de tornillos (longitud y diámetro) y los almacene en un vector

b) A partir de los datos introducidos calcule y muestre por pantalla la media de las longitudes y la media de los diámetros.

Nota: El tamaño de la muestra se definirá como una constante denominada TAMANIO\_MUESTRA con 5 elementos.

#include <iostream>

using namespace std;

const int TAMANIO\_MUESTRA = 5;

typedef struct

{

int longitud;

int diametro;

} tTornillo;

int main() {

int i;

float sumaL = 0;

float sumaD = 0;

float mediaL;

float mediaD;

tTornillo muestra[TAMANIO\_MUESTRA];

for(int i = 0; i < TAMANIO\_MUESTRA; i++)

{

cout << "Tornillo " << i + 1 << endl;

cout << "Ingrese la longitud: ";

cin >> muestra[i].longitud;

cout << "Ingrese el diametro: ";

cin >> muestra[i].diametro;

sumaL += muestra[i].longitud;

sumaD += muestra[i].diametro;

}

mediaL = (sumaL/TAMANIO\_MUESTRA);

mediaD = (sumaD/TAMANIO\_MUESTRA);

cout << endl << "La media de las longitudes es: " << mediaL <<endl;

cout << "La media de los diametros es: " << mediaD << endl;

return 0;

}

**/\*Código del programa\*/**

**3.** Se quiere implementar una aplicación para gestionar canciones. El número total de canciones no puede superar un cierto valor máximo N, siendo el número de obras existentes en cada momento variable. La información que se quiere almacenar para cada canción consiste en:

Número de canción (entero positivo)

Título de la canción (cadena)

Nombre del autor (cadena)

Genero de la canción (Salsa, Merengue, Baladas, Boleros, Electrónica o Pop ).

Las canciones se mantendrán ordenadas por el número de canción.

Se pide:

Redactar las declaraciones adecuadas.

Implementar un subprograma que cargue en la lista la información de un archivo de texto canciones.txt sobre canciones ordenadas por número de canción. Para cada canción, el archivo contiene cuatro líneas, que indican, en este orden, el número de canción, el título, el nombre del autor y el género. Termina con ‐1 como número de canción.

Implementar un subprograma que, dados una lista de canciones y el número de una canción, elimine de la lista la canción con dicho número, si es que existe, y devuelva un valor que indique si la operación ha tenido éxito. Suponer que la lista de canciones está ordenada en orden creciente por el número de canción y que no existen números de canciones repetidos. Si la operación tiene éxito (se suprime la canción): No deben quedar “huecos” en la lista.

También debe permitir:

* agregar nuevas canciones y
* mostrarlas ordenadas por el título de la canción o por el autor.

**/\*Código del programa\*/**

#include <iostream>

#include <string>

using namespace std;

#include <fstream>

#include <iomanip>

const int N = 100;

typedef struct{

int numero;

string nombre;

string autor;

string genero;

}tCancion;

typedef struct {

tCancion canciones[N];

int contador;

} tListaCanciones;

int menu\_canciones();

void cargar\_canciones(tListaCanciones& lista, bool& ok);

void guardar\_canciones(const tListaCanciones lista);

void leercancion(tCancion& cancion);

void insertarcancion(tListaCanciones& lista, tCancion cancion, bool& ok);

void eliminarcancion(tListaCanciones& lista, int pos, bool& ok);

string nombreCompleto(tCancion cancion);

void listado\_canciones(const tListaCanciones lista);

void ordenarBurbuja(tListaCanciones& lista, tCancion& cancion);

int main() {

  tListaCanciones lista;

  tCancion cancion;

  bool exito;

  int op, pos;

  cargar\_canciones(lista, exito);

  if (!exito)

{

cout << "No se ha podido cargar la lista!" << endl;

  }

  else {

    do {

      op = menu\_canciones();

      switch (op) {

      case 1:

      {

        leercancion(cancion);

        insertarcancion(lista, cancion, exito);

        if (!exito) {

          cout << "Lista llena: imposible insertar" << endl;

        }

      }

      break;

      case 2:

      {

        cout << "Posición: ";

        cin >> pos;

        eliminarcancion(lista, pos - 1, exito);

        if (!exito) {

          cout << "Elemento inexistente!" << endl;

        }

      }

      break;

      case 3:

      {

      }

      break;

      case 4:

      {

        listado\_canciones(lista);

ordenarBurbuja(lista,cancion);

      }

      }

    } while (op != 0);

    guardar\_canciones(lista);

  }

  return 0;

}

int menu\_canciones()

{

int opc;

cout << " " <<"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

cout << " |" << setw(54) << " |" << endl;

cout << " |" << setw(30) << " CANCIONES " <<setw(24) <<"|"<< endl;

cout << " |" << setw(54) << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|" << endl;

cout << " |" << setw(54) << " |" << endl;

cout << " |" << "1 AÑADIR CANCION" << setw(36) << " |" << endl;

cout << " |" << "2 ELIMINAR CANCION" << setw(34) << " |" << endl;

cout << " |" << "3 LISTADO DE CANCIONES" << setw(31) << " |" << endl;

cout << " |" << "0 SALIR" << setw(46) << " |" << endl;

cout << " |" << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|" << endl << endl;

do

{

cout << " OPCION: ";

cin >> opc;

if (opc < 0 || opc > 3)

{

cout << endl << "ERROR! La opcion ingresada no es valida... Ingrese nuevamente." << endl << endl;

}

} while (opc < 0 || opc > 3);

  return opc;

}

void cargar\_canciones(tListaCanciones& lista, bool& ok) {

  tCancion cancion;

  ifstream archivo;

  char aux;

  lista.contador = 0;

  archivo.open("canciones.txt");

  if (!archivo.is\_open()) {

    ok = false;

  }

  else {

    ok = true;

    archivo >> cancion.numero;

    while ((cancion.numero != -1) && (lista.contador < N))

{

getline(archivo,cancion.nombre);

getline(archivo,cancion.autor);

getline(archivo,cancion.genero);

      lista.canciones[lista.contador] = cancion;

archivo >> cancion.numero;

    }

  }

archivo.close();

}

void guardar\_canciones(const tListaCanciones lista) {

  ofstream archivo;

  archivo.open("pacientes.txt");

  for (int i = 0; i < lista.contador; i++) {

    archivo << lista.canciones[i].numero << endl;

    archivo << lista.canciones[i].nombre << endl;

    archivo << lista.canciones[i].autor << endl;

    archivo << lista.canciones[i].genero << endl;

  }

  archivo << -1;

  archivo.close();

}

void leercancion(tCancion& cancion)

{

  cin.ignore();

cout << "Numero de cancion: ";

  cin >> cancion.numero;

  cout << "Nombre: ";

  getline(cin, cancion.nombre);

  cout << "Autor: ";

  getline(cin, cancion.autor);

  cout << "Genero: ";

  getline(cin, cancion.genero);

}

void insertarcancion(tListaCanciones& lista, tCancion cancion, bool& ok) {

  ok = true;

  if (lista.contador == N) {

    ok = false;

  }

  else {

    lista.canciones[lista.contador] = cancion;

    lista.contador++;

  }

}

void eliminarcancion(tListaCanciones& lista, int pos, bool& ok)

{

  if ((pos < 0) || (pos > lista.contador - 1))

{

    ok = false;

  }

  else {

    ok = true;

    for (int i = pos; i < lista.contador - 1; i++)

{

      lista.canciones[i] = lista.canciones[i + 1];

    }

    lista.contador--;

  }

}

string nombreCompleto(tCancion cancion)

{

  return cancion.nombre;

}

void listado\_canciones(const tListaCanciones lista)

{

  for (int i = 0; i < lista.contador; i++)

{

cout << i + 1 << ": ";

cout << setw(6) << right << lista.canciones[i].numero << " ";

cout << setw(30) << left << nombreCompleto(lista.canciones[i]);

cout << setw(8) << setprecision(2) << lista.canciones[i].autor << " ";

cout << setw(3) << left << lista.canciones[i].genero<< "";

cout<<endl;

   }

}

void ordenarBurbuja(tListaCanciones& lista, tCancion& cancion) {

  int i = 0;

  bool inter = true;

  tCancion temp;

  if (lista.contador == 0) {

    cout << endl << "ERROR! No existen productos en el registro" << endl << endl;

  }

  else {

    while ((i < lista.contador - 1) && inter) {

      inter = false;

      for (int j = lista.contador - 1; j > i; j--) {

        if (lista.canciones[j].nombre < lista.canciones[j - 1].nombre)

{

          temp = lista.canciones[j];

          lista.canciones[j] = lista.canciones[j - 1];

          lista.canciones[j - 1] = temp;

          inter = true;

        }

      }

      if (inter) {

        i++;

      }

    }

    cout << endl << ">> Se ha ordenado el registro de acuerdo a los nombres de los productos! " << endl << endl;

    listado\_canciones(lista);

  }

}

canciones.txt

1

Walk This Way

Aerosmith

Rock

2

Welcome To The Jungle

Gun N´ Roses

Rock

3

My Way

Frank Sinatra

Pop

4

Blinding Lights

The Weeknd

R&B/Soul

-1