Ayudantía 7

Repaso LAB

Benjamín Aceituno

Programación Avanzada S09

Veamos un ejercicio tipo lab, paso por paso.

Desarrolla un sistema de gestión de estudiantes y cursos en el que cada estudiante puede matricularse en múltiples cursos, y cada curso puede tener varios estudiantes inscritos. Para implementar este sistema, crea dos clases: Estudiante y Curso.

Clase Estudiante:

Atributos: Nombre, Rut, Vector de cursos matriculados

Implemente el método constructor.

matricularCurso(string curso): Método que añade el nombre del curso al vector cursosMatriculados.

mostrarCursos(): Método que imprime todos los cursos en los que el estudiante está matriculado.

Ahora con el vector nos ahorramos un problema, que es implementarlo en el constructor con this, el vector se inicializa automáticamente como vacío, recuerden

```cpp
// Clase Estudiante
class Estudiante {
public:
    string nombre;
    int rut;
    vector<string> cursosMatriculados;

    Estudiante(string nombre, int rut) {
        this->nombre = nombre;  // asignación de los atributos usando this
        this->rut = rut;
        // el vector cursosMatriculados se inicializa automáticamente como vacío
    }
```

Que pasaría si tuvieramos una clase solo con vectores?

```
// Clase Biblioteca
class Biblioteca {
public:
    vector<Libro> librosDisponibles;
    vector<Lector> lectoresRegistrados;

    // constructor vacío
    Biblioteca() {}
```

Aquí tenemos una clase biblioteca que almacena vectores de la clase libro.

Ocupamos un constructor vacío, ya que los vectores se inicializan automáticamente.

matricularCurso(string curso)

Siempre que queremos ingresar algo, o en este caso, matricular un curso, usamos el push_back, ya que estamos añadiendo un elemento al vector
Recuerden! está recibiendo un parámetro, y se especifica que recibe un parámetro de cierto tipo, osea de tipo string curso, en este caso sería el nombre del curso

```
// método para matricularse en el curso, ingresa un string curso a un vector de cursos.
void matricularCurso(string curso) {
    cursosMatriculados.push_back(curso);
}
```

mostrarCursos()

Este un método que muestra, osea, va a imprmir los contenidos del vector, y para esto hay que recorrer el vector y mostrar su contenido.

```
// método para mostrar los cursos en los que el estudiante está matriculado
void mostrarCursos() {
cout << "Cursos matriculados por " << nombre << ": ";
for (int i = 0; i < cursosMatriculados.size(); i++) {
    cout << cursosMatriculados[i] << ". ";
}
cout << endl;
```

Clase Curso

Atributos:

nombreCurso, codigoCurso, estudiantesMatriculados: un vector que almacena punteros a los objetos de la clase Estudiante.

Métodos:

Constructor

matricularEstudiante(Estudiante* estudiante): Método que añade al estudiante (como puntero) al vector estudiantesMatriculados y lo matricula en el curso.

mostrarEstudiantes(): Método que imprime los nombres de los estudiantes matriculados en el curso.

La clase curso tiene un vector de objetos de la clas estudiante, que igualmente no se inicializa

```
class Curso {
public:
    string nombreCurso;
    string codigoCurso;
    vector<Estudiante*> estudiantesMatriculados; // vector de clases estudiante

    Curso(string nombreCurso, string codigoCurso) {
        this->nombreCurso = nombreCurso;
        this->codigoCurso = codigoCurso;
        // el vector estudiantesMatriculados se inicializa automáticamente como vacío
    }
```

matricularEstudiante(Estudiante* estudiante)

Esto recibirá como parámetro a un estudiante, pero ojo, es un objeto del tipo estudiante, este debe ser creado en el main. El estudiante es ingresado a un vector con push_back.
Ahora estudiante apunta a matricular curso, que es uno de los métodos anteriores, que recibia como parámetro el nombre del curso, entonces el estudiante se matriculará a un curso con un nombre en particular.

```
// Método para matricular un estudiante en el curso
void matricularEstudiante(Estudiante* estudiante) {
    estudiantesMatriculados.push_back(estudiante);
    estudiante->matricularCurso(nombreCurso);
}
```

Para que esto no sea tan confuso se explicará a detalle

Se crean los estudiantes

```
Estudiante* estudiante1 = new Estudiante("Diego Mena", 21471976);
Estudiante* estudiante2 = new Estudiante("Kanye West", 21733733);
Estudiante* estudiante3 = new Estudiante("Vicente Diaz", 21733734);
Estudiante* estudiante4 = new Estudiante("Pepe Miguel", 21733735);
Estudiante* estudiante5 = new Estudiante("John Duart", 21733736);
```

Se crean los cursos

```
// Crear cursos dinámicamente con new
Curso* curso1 = new Curso("Programación", "CIT1000");
Curso* curso2 = new Curso("Progamación Avanzada", "CIT1010");
```

curso apuntará a matricular estudiante, recibirá un estudiante como parámetro

```
curso1->matricularEstudiante(estudiante1);
curso1->matricularEstudiante(estudiante2);
curso1->matricularEstudiante(estudiante3);
curso1->matricularEstudiante(estudiante4);
curso1->matricularEstudiante(estudiante5);
curso2->matricularEstudiante(estudiante1);
```

```
// Método para matricular un estudiante en el curso
void matricularEstudiante(Estudiante* estudiante) {
    estudiantesMatriculados.push_back(estudiante);
    estudiante->matricularCurso(nombreCurso);
}
```

```
// método para matricularse en el curso, ingresa un string curso a un vector de cursos.
void matricularCurso(string curso) {
    cursosMatriculados.push_back(curso);
}
```

No nos olvidemos de mostrarEstudiantes()

```
    // Método para mostrar los estudiantes matriculados en el curso
    void mostrarEstudiantes() {
    cout << "Estudiantes matriculados en " << nombreCurso << ": ";
    for (int i = 0; i < estudiantesMatriculados.size(); i++) {
        cout << estudiantesMatriculados[i]->nombre << ". ";
    }
    cout << endl;
}
```

Simplemente recorremos los estudiantes matriculados con un for en base a su size.

Diferencia entre mostrarEstudiantes y mostrarCursos?

```
    // método para mostrar los cursos en los que el estudiante está matriculado
    void mostrarCursos() {
    cout << "Cursos matriculados por " << nombre << ": ";
    for (int i = 0; i < cursosMatriculados.size(); i++) {
        cout << cursosMatriculados[i] << ". ";
    }
    cout << endl;
}
```

mostrarCursos va a mostrar los cursos en los que el estudiante en particular está matriculado, entonces estudiante debe llamar a ese método.

```
estudiante1->mostrarCursos();
```

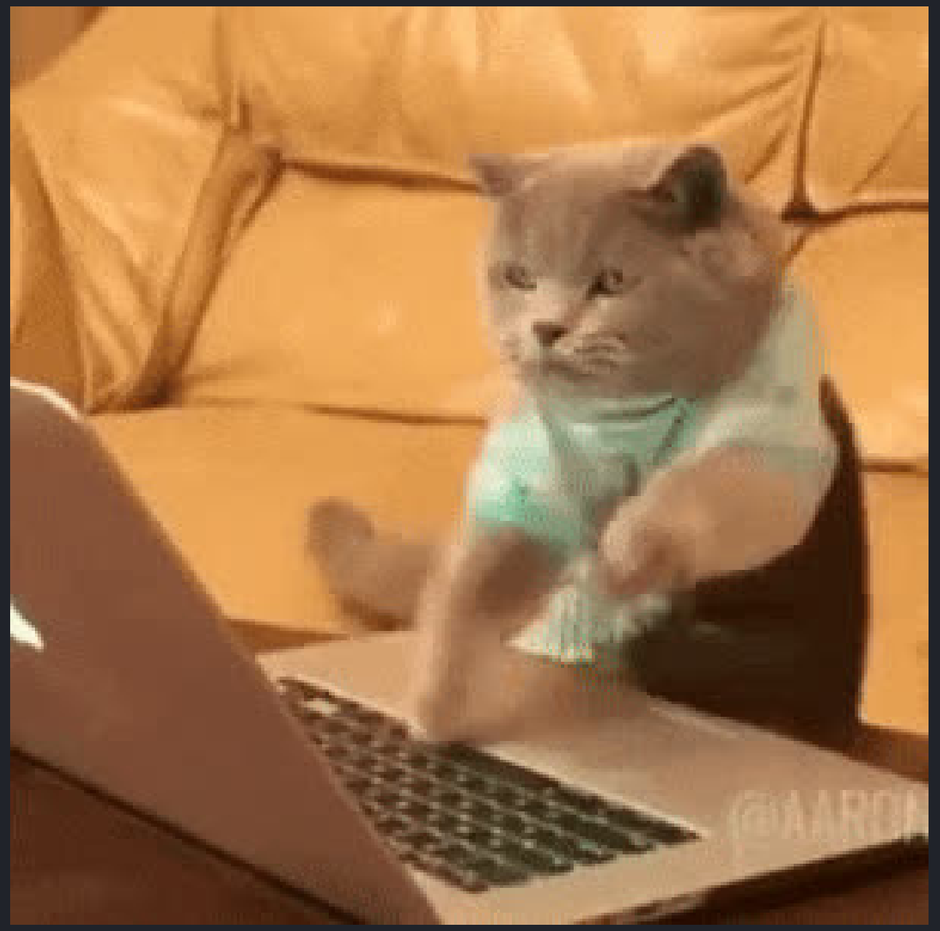
Diferencia entre mostrarEstudiantes y mostrarCursos?

```
    // Método para mostrar los estudiantes matriculados en el curso
    void mostrarEstudiantes() {
    cout << "Estudiantes matriculados en " << nombreCurso << ": ";
    for (int i = 0; i < estudiantesMatriculados.size(); i++) {
        cout << estudiantesMatriculados[i]->nombre << ". ";
    }
    cout << endl;
}
```

mostrarEstudiantes mostrará los estudiantes matriculados en vierto curso, entonces el curso debe llamar a ese método.

```
curso1->mostrarEstudiantes();
```

Ahora revisaremos el código en el computador...



Ahora veamos un ejercicio con map, paso por paso

Desarrolle un sistema para una tienda de CDs, la tienda manejará mapas de CDs, clientes y CDs prestados. La tienda debe ser capaz de agregar CDs con su ID y nombre, registrar a los clientes con su rut y nombre, prestar los CDs que tenga, pero un cliente no puede pedir más de 3 CDs prestados, mostrar los CDs y clientes, además de devolver CDs.

Clase Tienda
Atributos: CDs, Clientes, CDs Prestados (Todo en Map)

Implemente el método constructor.

agregarCD(int id, string titulo): Recibe como parámetro id del cd y su título, agrega un CD para que este sea prestado

registrarCliente(string rut, string nombre): Recibe como parámetro el rut y el nombre del cliente, registra un cliente debe partir con 0 CDs prestados

void prestarCD(string rutCliente, int idCD): Recibe como parámetro el rut del cliente y el id del CD, debe verificar que el cd y el cliente existen y el cliente no puede pedir más de 3 CDs.

void devolverCD(string rutCliente, int idCD, string titulo): Recibe como parámetro el rut del cliente, el id del cd y el título del CD, debe verificar que el cd y el cliente existen, descontar el CD al usuario y volver a agregar el CD.

void mostrarCDs(): Muestra todos los CDs.

void mostrarClientes(): Muestra todos los clientes y los CDs que le prestaron

Atributos y constructor

Como los atributos son solo mapas el constructor es vacío

```cpp
#include <iostream>
#include <map>
#include <string>

using namespace std;

class Tienda {
public:
    map<int, string> cds;
    map<string, string> clientes;
    map<string, int> cdsPrestados;

    Tienda() {}
```

agregarCD y registrarCliente

Le pasamos los parámetros que queremos agregar al map, eso hará que se ingrese la llave y valor al mapa dependiendo del parámetro que le pasemos.

```
void agregarCD(int id, string titulo) { // Agrega cd, iguala los parámetros que le pasemos a un mapa
    cds[id] = titulo;
}

void registrarCliente(string rut, string nombre) { // Agrega un cliente, iguala los parámetros que le pasamos a un mapa
    clientes[rut] = nombre;
    cdsPrestados[rut] = 0; // CDs prestados asociados a ese rut parten en 0
}
```

prestarCD

Verifica si existe o no usando find == end, si se cumple esa condición significa que no está, porque como vimos anteriormente, en el end no hay nada

```
void prestarCD(string rutCliente, int idCD) { // Prestar cd, para esto queremos saber el rut de quien se lo queremos prestar y el id del cd
    if (clientes.find(rutCliente) == clientes.end()) { // Si el cliente no existe
        cout << "El cliente con RUT " << rutCliente << " no está registrado." << endl; // El cliente no está registrado
        return;
    }

    if (cds.find(idCD) == cds.end()) { // Si el id del cd no existe
        cout << "El CD con ID " << idCD << " no está disponible." << endl; // El cd no está disponible
        return;
    }

    if (cdsPrestados[rutCliente] >= 3) { // Verificador para que el cliente no sea pesao y se lleve más de 3 cds
        cout << "El cliente con RUT " << rutCliente << " ya tiene 3 CDs prestados, no sea pesao, dejele a los demás..." << endl;
        return;
    }

    cdsPrestados[rutCliente]++; // Si se cumple toda nornalmente va aumentar la cantidad de cds prestados que tiene el rut asociado
    cout << "El CD '" << cds[idCD] << "' ha sido prestado al cliente " << clientes[rutCliente] << "." << endl; // Mostrar por pantalla
    cds.erase(idCD); // se elimina el cd porque lo tomaron prestado, osea no está disponible si queremos tomarlo prestado
}
```

devolverCD

Se descuentan los cds prestados del rut del cliente que se pasa como parámetro, además se agrega el cd que se devolvió a los cds.

```
void devolverCD(string rutCliente, int idCD, string titulo) { // metodo que recibe el rut cliente que quiere devolver, la
    if (clientes.find(rutCliente) == clientes.end()) { // Si no se encuentra el rut...
        cout << "El cliente con RUT " << rutCliente << " no está registrado." << endl;
        return;
    }

    if (cdsPrestados[rutCliente] == 0) { // Si el cliente quiere devolver un cd pero tiene 0 cdss asociados
        cout << "El cliente con RUT " << rutCliente << " no tiene CDs prestados." << endl;
        return;
    }

    cdsPrestados[rutCliente]--; // Si va todo normal se disminuye la cantidad de cds prestados que tiene el rut asociado
    cds[idCD] = titulo; // volvemos a agregar el cd de la misma manera que los añade la funcion del principio del codigo
    cout << "El cliente " << clientes[rutCliente] << " ha devuelto el CD '" << titulo << "'." << endl;
}
```

mostrarCDs y mostrarClientes

Se ocupa el iterator con un for, usando la sintaxis para recorrer un map, desde begin hasta que sea distinto de end

```
void mostrarCDs() { // Queremos mostrar los cds...
    cout << "CDs disponibles en la tienda:" << endl;
    map<int, string>::iterator it; // iterador
    for (it = cds.begin(); it != cds.end(); it++) { // Mostramos los cds de principio a fin
        cout << "ID: " << it->first << ", Título: " << it->second << endl; // Ocupamos el iterador
    }
}

void mostrarClientes() { // Queremos mostrar los clientes y los cds que le prestaron
    cout << "Clientes registrados en la tienda:" << endl;
    map<string, string>::iterator it; // iterador
    for (it = clientes.begin(); it != clientes.end(); it++) { // Mostramos los clientes de principio a fin
        cout << "RUT: " << it->first << ", Nombre: " << it->second << ", CDs prestados: " << cdsPrestados[it->first] << endl; // Ocupamos el iterador
    }
}
```

main()

```cpp
int main() {

    Tienda tienda; // Clase tienda, donde por alguna razon prestan cds... rarisimo

    tienda.agregarCD(1, "In Rainbows - Radiohead");
    tienda.agregarCD(2, "Schlagenheim - black midi");
    tienda.agregarCD(3, "Drukqs - Aphex Twin");
    tienda.agregarCD(4, "Remain in Light - Talking Heads");
    tienda.agregarCD(5, "Suck it and See - Arctic Monkeys");

    tienda.registrarCliente("21471976-9", "Charly");
    tienda.registrarCliente("21733733-2", "Gustavo");
    tienda.registrarCliente("21133071-6", "Vicente");

    tienda.prestarCD("21471976-9", 1);
    tienda.prestarCD("21471976-9", 2);
    tienda.prestarCD("21471976-9", 3);
    tienda.prestarCD("21471976-9", 4);
    tienda.prestarCD("21133071-6", 5);


    tienda.mostrarCDs();
    tienda.mostrarClientes();

    tienda.devolverCD("21471976-9", 1, "In Rainbows - Radiohead");

    tienda.mostrarCDs();

    tienda.prestarCD("21733733-2",1);

    return 0;
}
```

Output

```
El CD 'In Rainbows - Radiohead' ha sido prestado al cliente Charly.
El CD 'Schlagenheim - black midi' ha sido prestado al cliente Charly.
El CD 'Drukqs - Aphex Twin' ha sido prestado al cliente Charly.
El cliente con RUT 21471976-9 ya tiene 3 CDs prestados, no sea pesao, dejele a los demás...
El CD 'Suck it and See - Arctic Monkeys' ha sido prestado al cliente Vicente.
CDs disponibles en la tienda:
ID: 4, Título: Remain in Light - Talking Heads
Clientes registrados en la tienda:
RUT: 21133071-6, Nombre: Vicente, CDs prestados: 1
RUT: 21471976-9, Nombre: Charly, CDs prestados: 3
RUT: 21733733-2, Nombre: Gustavo, CDs prestados: 0
El cliente Charly ha devuelto el CD 'In Rainbows - Radiohead'.
CDs disponibles en la tienda:
ID: 1, Título: In Rainbows - Radiohead
ID: 4, Título: Remain in Light - Talking Heads
El CD 'In Rainbows - Radiohead' ha sido prestado al cliente Gustavo.
```

Ahora revisaremos el código en el computador...

Contacto

+569 86031881

benjamin.aceituno@mail.udp.cl

benja.mp4

https://github.com/benjamp4/prog.av-2024-2