Ayudantia n°6

Polimorfismo, vectores y Mapas.

Diego Mena

Polimorfismo

Gracias al polimorfismo, podemos hacer que dos objetos de diferentes clases, que están relacionadas por herencia, respondan de manera distinta al mismo comando. Esto se logra mediante el uso de la palabra clave virtual (que se coloca antes del tipo de retorno, como void o int, según la función). Al declarar un método como virtual, permitimos que el objeto responda de forma diferente cuando se llama a dicho método

```
Class Animal
{
private: ...
public:...
Virtual void sonido()
{
cout<<"bla bla"<<endl;
}
};
```

```
Class Perro: public Animal
{
private: ...
public:...
Virtual void sonido()
{
cout<<"Guau"<<endl;
}
};
```

```
Class Gato: public Animal
{
private: ...
public:...
Virtual void sonido()
{
cout<<"miau"<<endl;
}
};
```

```
int Main()
{
Perro* Obj= new Perro();
Gato* Obj1= new Gato();

Obj->Sonido(); //Guau
Obj1->Sonido(); //Miau
}
```

Vectores

Un vector es como un arreglo, debido a que de la misma manera almacena datos, el cambio de esto es que el vector no tiene un tamaño fijo, y tambien tiene una metodologia de Lifo (Last in First out).

```
#include <vector>

vector<tipo_de_dato>nombre;
vector<string>nombre;
vector<int>nombre;
vector<Clase*>nombre;
```

size(): retorna la cantidad de elementos, en este caso nos devolveria 3

empty(): retorna verdadero si está vacío y false en caso contrario

front(): retorna el valor del primer elemento, en este caso “1”.

back(): retorna el valor el ultimo elemento, en este caso “3”.

push_back(valor): agrega un elemento al final del vector, en este caso agregamos 7

pop_back(): elimina el último elemento del vector, aqui eliminaria el 7.

at(i): Retorna el valor del vector en una posición dada, ejemplo i=2;. retornara 3.

clear(): Elimina todo los elementos.

begin(): Retorna un iterador que apunta al primer elemento.

end(): Retorna un iterador que apunta a lo que va después del último elemento.

erase(rango): Elimina el o los elementos según un rango dado. Este rango se debe indicar con los iteradores vistos anteriormente.

```
Nombre_vector.erase(Nombre_vector.begin());
```


```
Nombre_vector.erase(Nombre_vector.end()-1);
```

Ejercicio

Se requiere desarrollar un sistema que permita gestionar la información de estudiantes. El sistema debe permitir ingresar tanto el Rut como nombre, además de registrar y calcular el promedio de las notas del estudiante.
Requisitos:

Clase Estudiante:

- Debe tener un rut, un nombre, y un vector de Floats para las notas.

- Funciones pedidas:
  - ingresarUnaNota(float nota): Añade una sola nota al vector.
  - ingresarMuchasNotas(vector<float> Lista): permite ingresar múltiples notas a la vez.
  - calcularPromedio(): debe calcular y devolver el promedio de las notas ingresadas.
  - lista(): debe mostrar la información y notas del estudiante.

}

Mapas

Un mapa contiene dos tipos de datos, uno el cual funcionara como llave para poder acceder al otro tipo de dato, esto se puede observar de la siguiente manera:

LLave	LLave	LLave
Cont.	Cont.	Cont.

```
#include <map>
//Tipo de elemento=TD

map<TD1, TD2>Nombre;
map<string, int>nombre;

Nombre[LLave]=Contenido;
Nombre[string]=Int;
```

Contacto {

+56 9 6070 8865

diego.mena2@mail.udp.cl

https://github.com/Dmena1/Ayudantia-prograAvanzada

}