Ayudantía 6

Vector y Map

Benjamín Aceituno

Programación Avanzada S09

Antes de enseñar qué es vector y map les enseñaré qué es el polimorfismo, un concepto que forma parte de herencia. Es esencial en polimorfismo y nos ayuda a optimizar código y memoria en C++.

Polimorfismo

El polimorfismo en C++ sirve para hacer responder a distintas clases con una misma función, osea, que cada una funcione de forma distinta al mismo mensaje. Se debe ocupar virtual en la clase base para ocupar esta caracteristica.

Virtual

Permite que las funciones compartidas de la clase
base puedan ser modificadas y de esta manera se
puedan distinguir cuando se usan

Ejemplo:

Tenemos la clase base Animal y las clases derivadas Gato y Perro, las tres clases como tal pueden hacer un sonido, usando virtual en la clase Animal podemos hacer que se llame al método virtual de la clase base a través de las clases Perro y Gato.

```cpp
#include <iostream>
using namespace std;
class Animal
{
    protected:
        string nombre;
    public:
        Animal(string nombre)
        {
            this->nombre=nombre;
        }
        virtual void Sonido() // Este método va con virtual para que sea sobreescrito por los demás métodos
        {
            cout<<"WAAAAAAAAA"<<endl; // Si no se usa virtual no se sobreescribirá y se llamará a este método
        }
};

class Perro : public Animal
{
    protected:
        string color;
    public:
        Perro(string nombre, string color) : Animal(nombre)
        {
            this->color=color;
        }
        void Sonido() // Si se usa este método, sobreescribirá el método de la clase base
        {
            cout<<"El perro con nombre: "<<nombre<<", es de color: "<<color<<" dice: guau guau soy un perro"<<endl;
        }
};
```

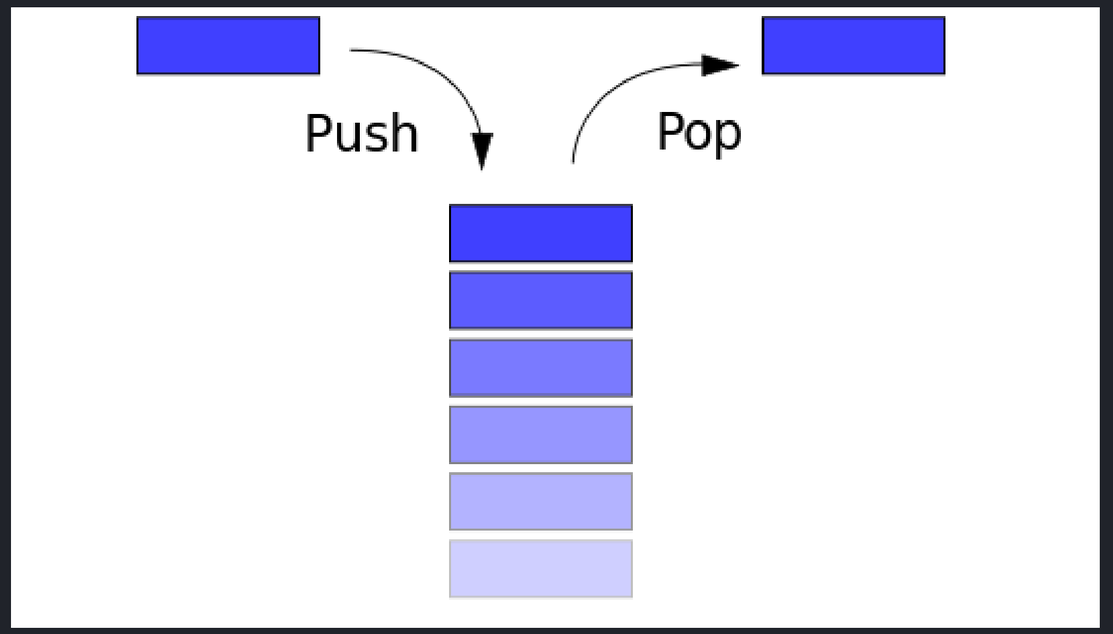
```
class Gato : public Animal
{
    protected:
        string color;
    public:
        Gato(string nombre, string color) : Animal(nombre)
        {
            this->color=color;
        }
        void Sonido() // Si se usa este método, sobreescribirá el método de la clase base
        {
            cout<<"El gato con nombre: "<<nombre<<", es de color: "<<color<<" dice: miau miau soy un gato"<<endl;
        }
};
int main()
{
    Perro * p = new Perro("Blu", "Cafe");
    Gato * g = new Gato("Gordo", "Naranjo");

    Animal* Arreglo[2]={p,g};

    for(int i=0;i<2;i++) // objeto p va a llamar a sonido y despues g a sonido
    {
        Arreglo[i]->Sonido();
    }
}
```

Vector

Los vectores en C++ funcionan como un arreglo dinámico, es decir, un arreglo pero sin un tamaño definido. Funciona en base a la metodología LIFO, y su tamaño va variando dependiendo de los elementos que añadimos o eliminamos.

LIFO?..

LIFO quiere decir Last In First Out, es decir, último entrar, primero en salir. Intenten verlo como si fuera almacenar una pila de cajas, nosotros ponemos cajas una encima de otra, y si sacamos una caja será la última que hemos puesto.



Funciones

El vector en C++ se debe incluir de esta manera en el código

```
#include <vector>
```

Se declara de esta forma

```
vector<tipoDeDato> nombre;
```

```
vector<int> numeros;
vector<string> palabras;
vector<Objeto*> objetos;
```

push_back(item): Inserta un ítem

pop_back(): Elimina un ítem.

clear(): Elimina todo.

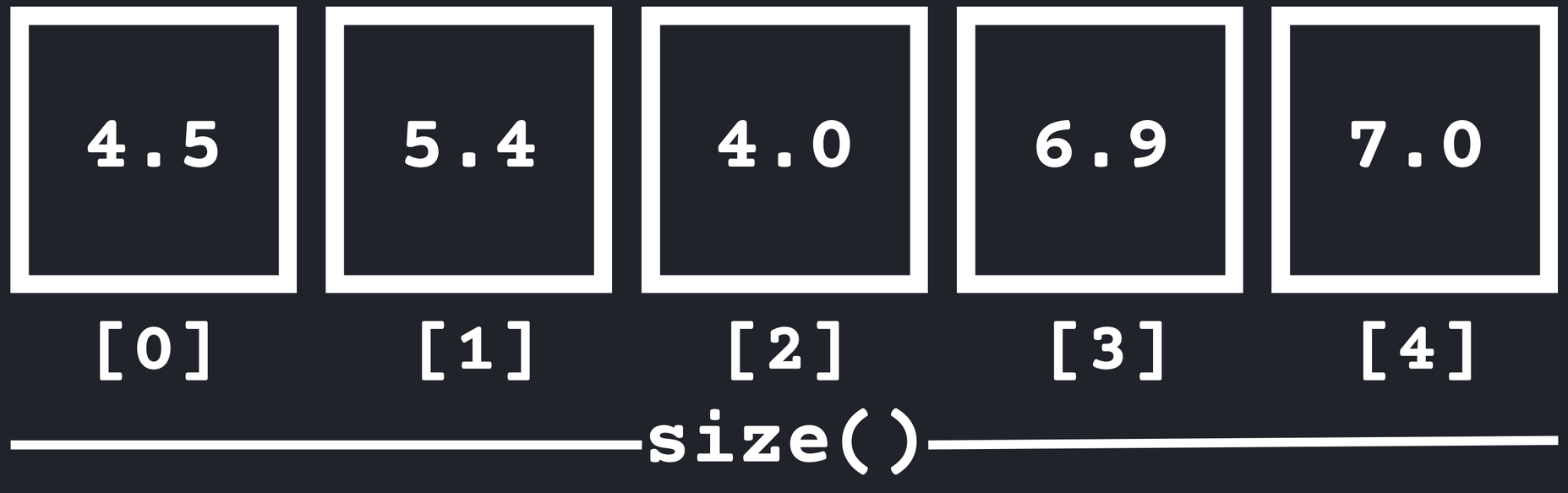
size(): Retorna el número de elementos del vector.

at(i): Retorna el valor del vector en una posición dada.

empty() : Retorna true si el vector está vacío.

Funciones de vector

```
vector<float> notas;
```




```
notas.size() -> 5
```

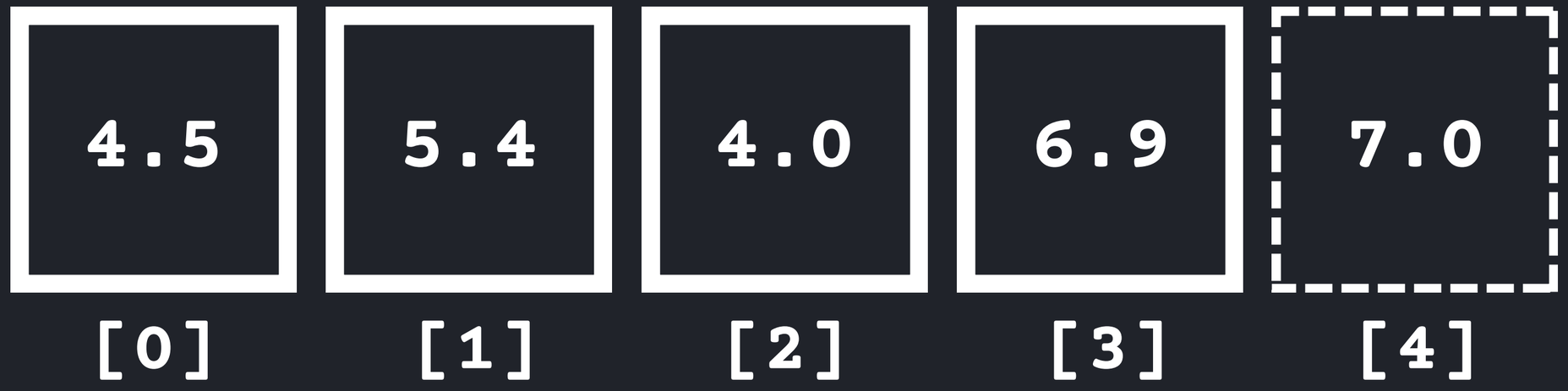
Funciones de vector

```
vector<float> notas;
```


```
notas.at(i) = notas[i]
```

Funciones de vector

```
vector<float> notas;
```




```
notas.pop_back();
```

Funciones de vector

```
vector<float> notas;
```

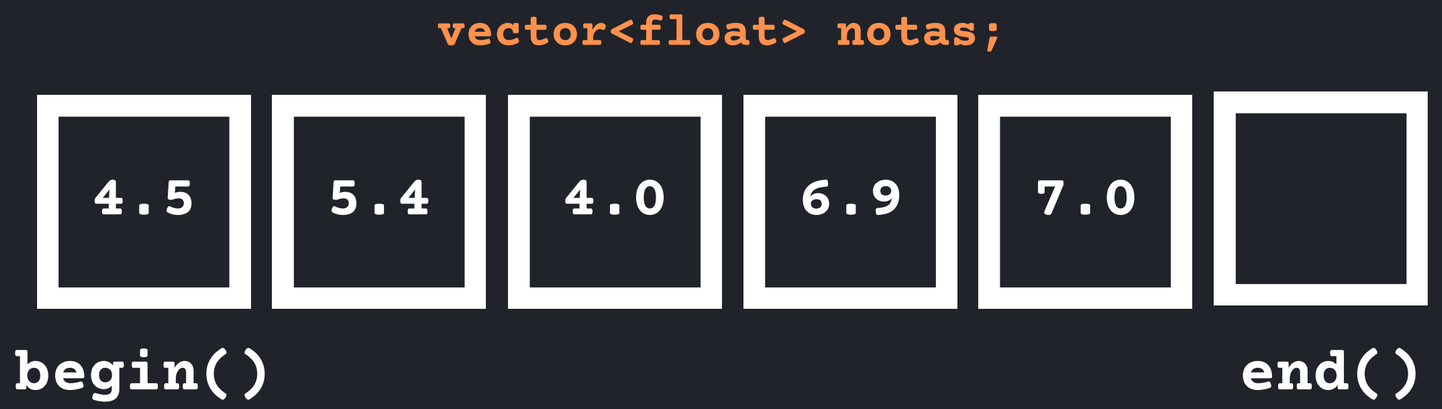

```
notas.push_back(1.0);
```

Funciones

Vector tiene más funciones como lo son

begin(): Apunta al primer elemento, se usa para comenzar un recorrido.

end(): Apunta al elemento después del último



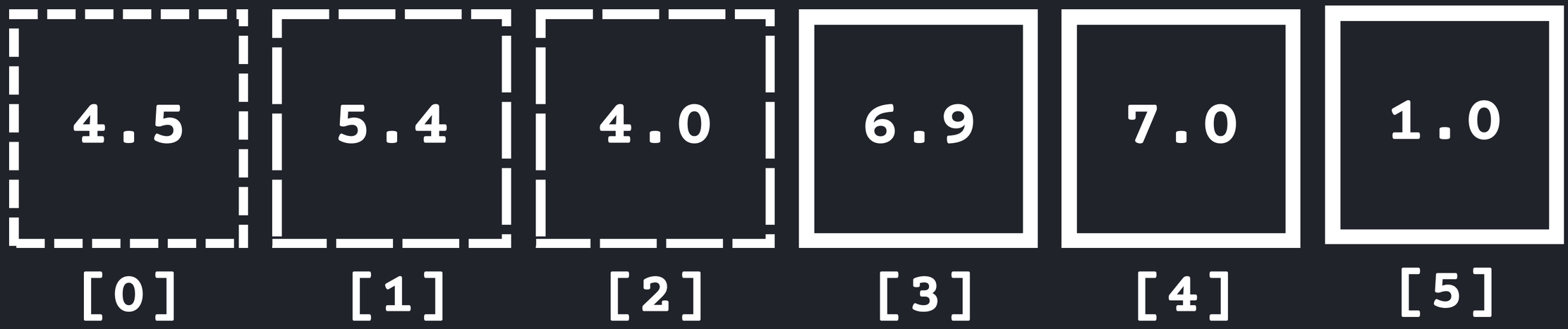
erase()

Esta función sirve para eliminar elementos dentro de un rango dado


```
notas.erase(notas.begin());
```

erase()

Esta función sirve para eliminar elementos dentro de un rango dado




```
notas.erase(notas.begin(),notas.begin()+3);
```

```cpp
#include <iostream>
#include <vector>

using namespace std;

int main() {
    vector<float> notas = {1.0, 3.0, 2.5, 4.0, 5.0};

    cout << "Notas almacenadas en el vector:" << endl;
    for (int i = 0; i < notas.size(); i++) {
        cout << "Nota " << i + 1 << ": " << notas[i] << endl;
    }

    notas.erase(notas.begin(), notas.begin() + 3);

    cout << "Notas después de eliminar las tres primeras:" << endl;
    for (int i = 0; i < notas.size(); i++) {
        cout << "Nota " << i + 1 << ": " << notas[i] << endl;
    }

    return 0;

}
```

```
Notas almacenadas en el vector:
Nota 1: 1
Nota 2: 3
Nota 3: 2.5
Nota 4: 4
Nota 5: 5
Notas después de eliminar las tres primeras:
Nota 1: 4
Nota 2: 5
```

Ejercicio

Los supermercados Ekono están teniendo un problema para implementar su sistema de carritos con espacio infinito, por lo cual se le pide que realice un programa en C++ donde un cliente tenga nombre, rut y un vector de productos (Los productos tienen solo su precio).
Este programa debe ser capaz de mostrar la información del cliente por pantalla, ingresar el precio de un producto, ingresar los precios de un vector de productos, mostrar la lista de precio de los productos y mostrar el precio total de todos los productos

Map

Es una estructura de dato que almacena pares Llave valor. Los elementos en un map están ordenados automáticamente según las llaves, y cada llave es única, lo que significa que no puede haber duplicados.

Declaración: map<llave,valor> nombre;

Se debe incluir con #include <map>

```cpp
#include <map>
using namespace std;

int main() {
    map<string, int> edad;

    edad["Vicente"] = 25;
    edad["Diego"] = 30;
    edad["Javier"] = 20;
}
```

```
map<string,float> notas;
```

Llave		Valor
21.000.000-0	→	6.3
21.000.001-0	→	2.4
21.000.002-0	→	1.0
21.000.003-0	→	6.0
21.000.004-0	→	4.3
21.000.005-0	→	7.0

Funciones

Para agregar un valor al mapa: map[llave] = valor;

begin(): Retorna el iterador de la primera posición.

end(): Retorna el iterador después de la última posición.

empty(): Devuelve true si el map está vacío.

size(): Devuelve el tamaño del map.

swap(map): Intercambia valores de un map a otro.

count(llave): Retorna true si la llave existe en el map, false en caso contrario.

find(llave): Retorna un iterador al elemento con la clave especificada

Si queremos recorrer el map...

```cpp
int main() {
    map<string, int> edad;

    edad["Vicente"] = 25;
    edad["Diego"] = 30;
    edad["Javier"] = 20;


    map<string, int>::iterator i;
    for (i = edad.begin(); i != edad.end(); i++) {
        cout << "Nombre: " << i->first << ", Edad: " << i->second << endl;
    }

    return 0;
}
```

Ejercicio

El día de hoy el profe John Duhart anda de muy buenos ánimos, por lo cual, se seleccionan 3 alumnos para tener 5 décimas extra en su nota, aunque solo un alumno será ganador de este gran premio. Para esto, haga un programa en C++ que tenga 3 map con el nombre y nota de los 3 alumnos, debe mostrar la lista de alumnos con su nombre y nota. Se le preguntará el nombre del alumno que quiere darle las décimas, después se mostrará la nota final, si el alumno supera la nota 7.0 con las décimas debe indicar cuántas décimas tiene para la siguiente evaluación.

```
if (notas.find(nombre) != notas.end())
```

Pista: Use este if para verificar el alumno y sumarle las décimas

Contacto

+569 86031881

benjamin.aceituno@mail.udp.cl

benja.mp4

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