Ayudantía N°4

Vectores

(y polimorfismo)

Ayudante: Natalia Romero

Programación Avanzada S07

Antes de empezar...

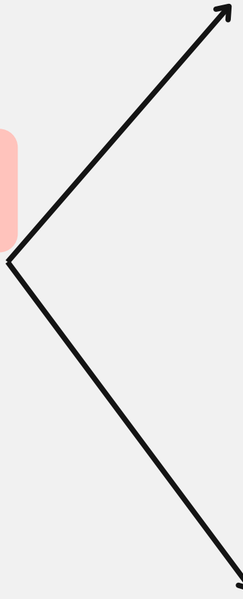
- Ya vimos herencia, pero nos quedó pendiente el polimorfismo.

Polimorfismo

- Es la habilidad de los objetos de diferentes clases que están relacionados mediante la herencia para responder en forma diferente al mismo mensaje.

Polimorfismo

```
class Animal{
  ..
 void sonido(){
   cout<<"..";
  }
};
```



```
class Gato:public Animal{
  ..
 void sonido(){
   cout<<"miau";
  }
};
```

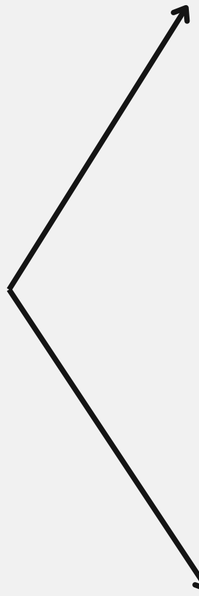
```
class Perro:public Animal{
  ..
 void sonido(){
   cout<<"guau";
  }
};
```

Polimorfismo

- **Funciones virtual:**
  - Permite que las funciones compartidas de la clase padre puedan ser modificadas y de esta manera se puedan distinguir cuando se usan.

Polimorfismo

```
class Animal{
  ..
 virtual void sonido(){
   cout<<"..";
  }
};
```



```
class Gato:public Animal{
  ..
 virtual void sonido(){
   cout<<"miau";
  }
};
```

```
class Perro:public Animal{
  ..
 virtual void sonido(){
   cout<<"guau";
  }
};
```

Polimorfismo

SIN VIRTUAL

```
int main(){
 Gato *g = new Gato();
 Perro *p = new Perro();
 Animal *a[2] = {g,p};

 a[0]->sonido(); "..."
 a[1]->sonido(); "..."
};
```

CON VIRTUAL

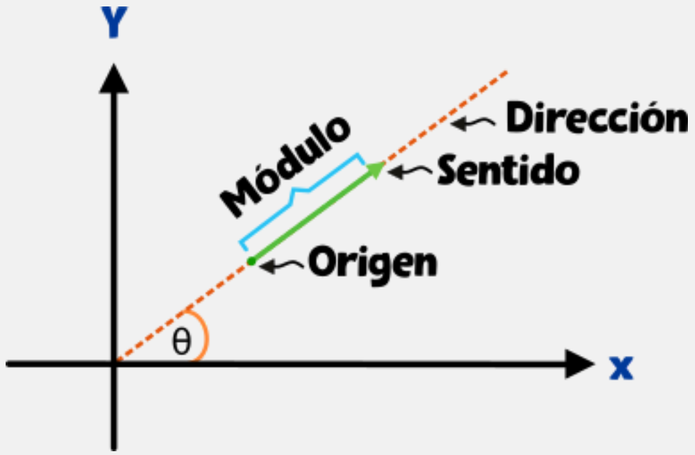
```
int main(){
 Gato *g = new Gato();
 Perro *p = new Perro();
 Animal *a[2] = {g,p};

 a[0]->sonido(); "miau"
 a[1]->sonido(); "guau"
};
```

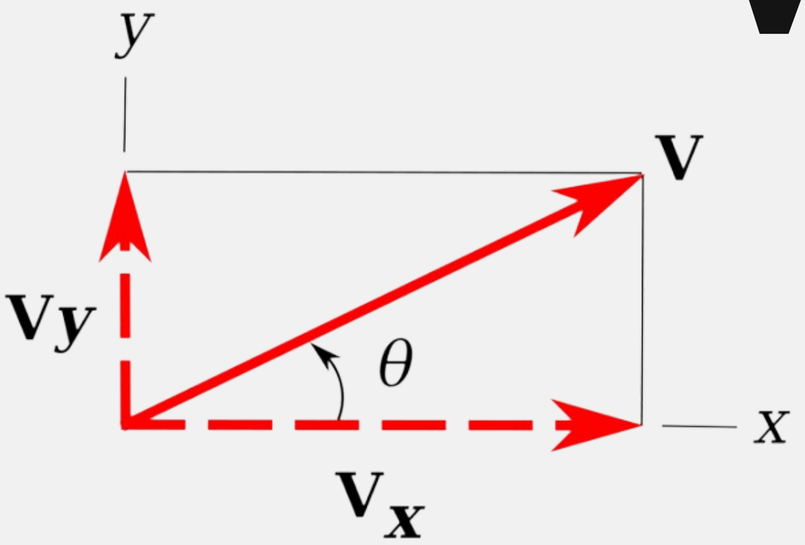
Ejercicio 1

- La empresa Toretto vende autos hace muuuchos años. Con urgencia le piden a usted crear un sistema de venta de sus vehículos. Un vehículo se distingue con un color, ruedas, cantidad (stock de inventario) y precio. Los tipos de vehiculos y sus particularidades son:
  - Auto: cantidad de pasajeros y velocidad.
  - Camion: carga (kg)
- Se pide hacer una función que despliegue la información estándar de cada vehículo, y además la información especifica de cada tipo (auto/camión).
- En el main debe crear un arreglo que contenga la clase camnion y auto, además con el uso del arreglo cree un menú que permita:
  - Vender los vehículos según su tipo, en cada venta se debe descontar del stock (stock inicial es 20 por cada uno), además debe imprimir la información al momento de realizar la venta.
  - Ver stock actual de cada vehiculo.

Ahora si

Vectores





Contenedores

- Actualmente, cuando teníamos que guardar muchos datos usabamos arreglos, pero debíamos asignar un tamaño y eso no nos permitía agregar datos de manera variable.

Contenedores

- Los contenedores nos ayudarán con este problema, ya que permite el ingreso y salida de datos masivos, sin tener un tamaño definido anteriormente. De esta forma se optimiza el uso de memoria.
- Durante este curso trabajaremos con 4 contenedores:

Vector - Map - Queue - Stack

Contenedores

- Existen dos conceptos importantes para poder distinguir cómo funciona la entrada (input) y salida (output) de datos a estos contenedores.
- Estos métodos son:
  - FIFO : “First in first out”
  - LIFO : “Last in first out”

Contenedores

- FIFO : “First in first out”

Contenedores

- FIFO : “First in first out”

¿Qué es un vector?

- Podemos verlos como "arreglos dinámicos" ya que guardarán datos o ítems pero sin un tamaño definido.
- Es de metodología LIFO.
- Tiene la ventaja de permitirnos acceder a todos los datos almacenados.

Vectores

- Para usarlos debemos incluír esta librería:

```
#include <vector>
```

Vectores

- Declaración:

```
vector<tipo_de_dato> nombre;
```

Vectores

- Declaración ejemplos:

```
vector<int> edades;
```

```
vector<string> direcciones;
```

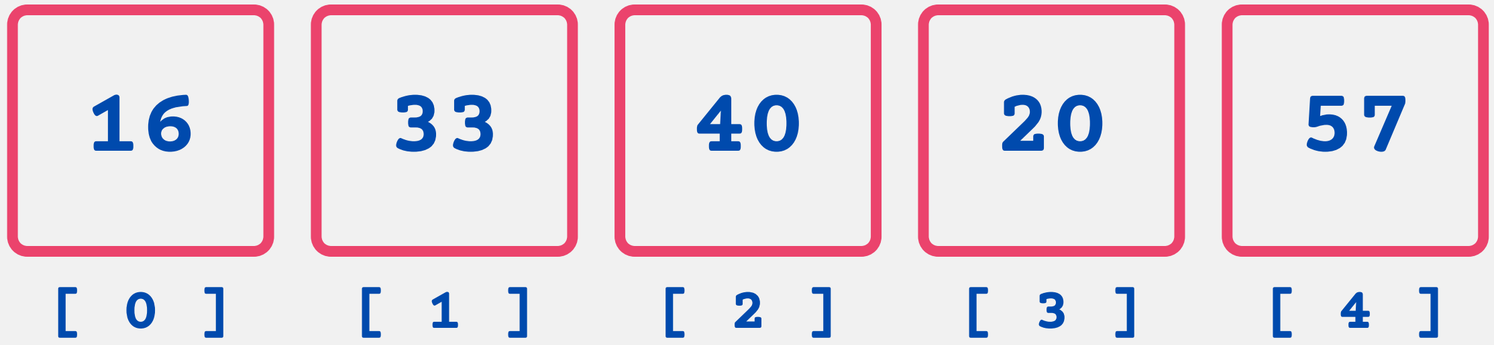
```
vector<Persona*> personas;
```

Funciones

- **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.

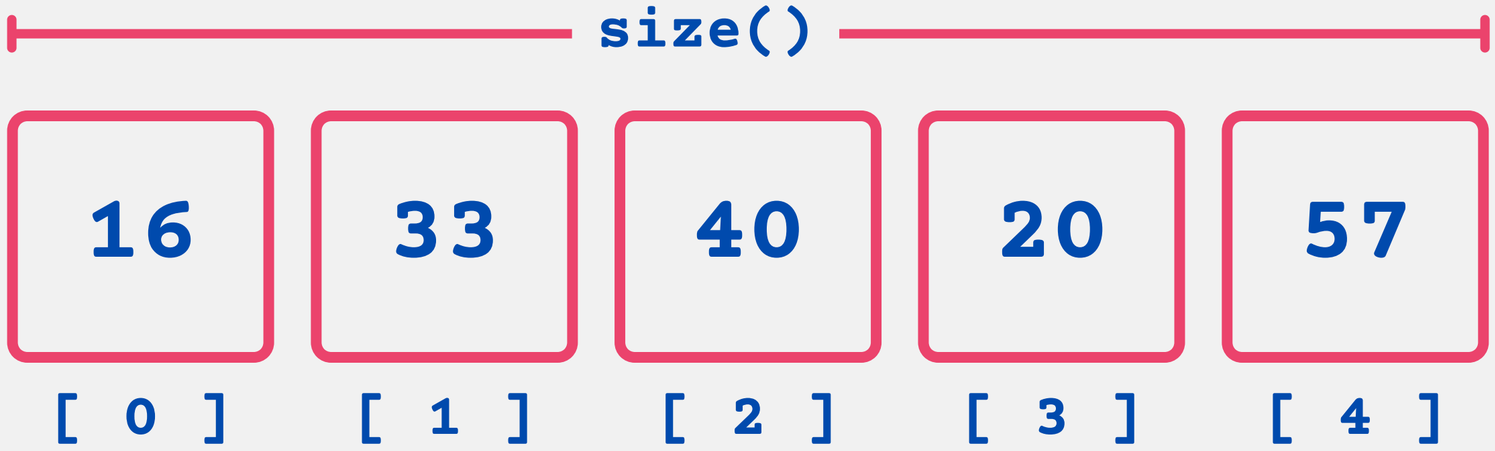
Funciones

```
vector<int> edades;
```



Funciones

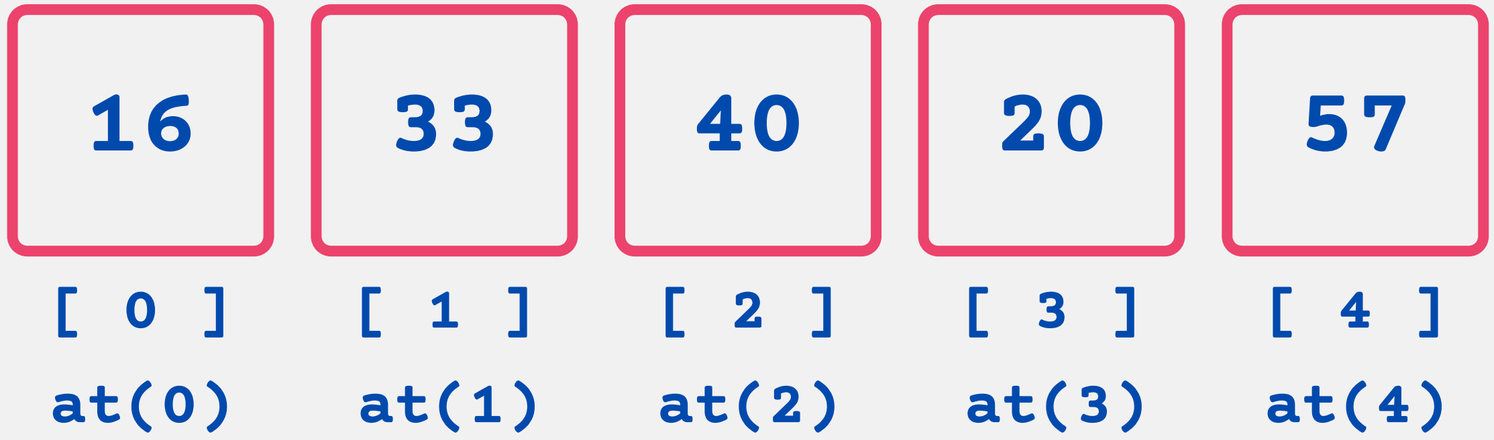
```
vector<int> edades;
```




```
edades.size() → 5
```

Funciones

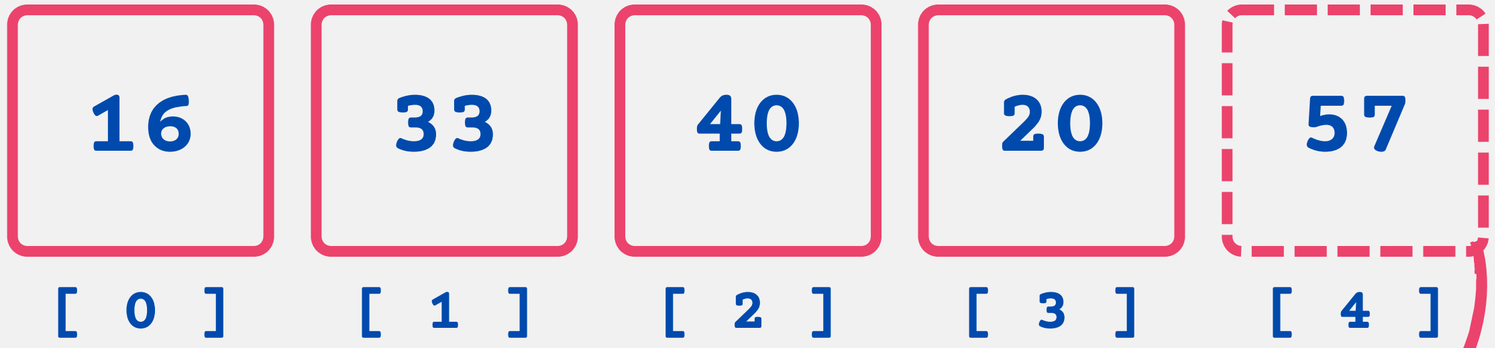
```
vector<int> edades;
```




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

Funciones

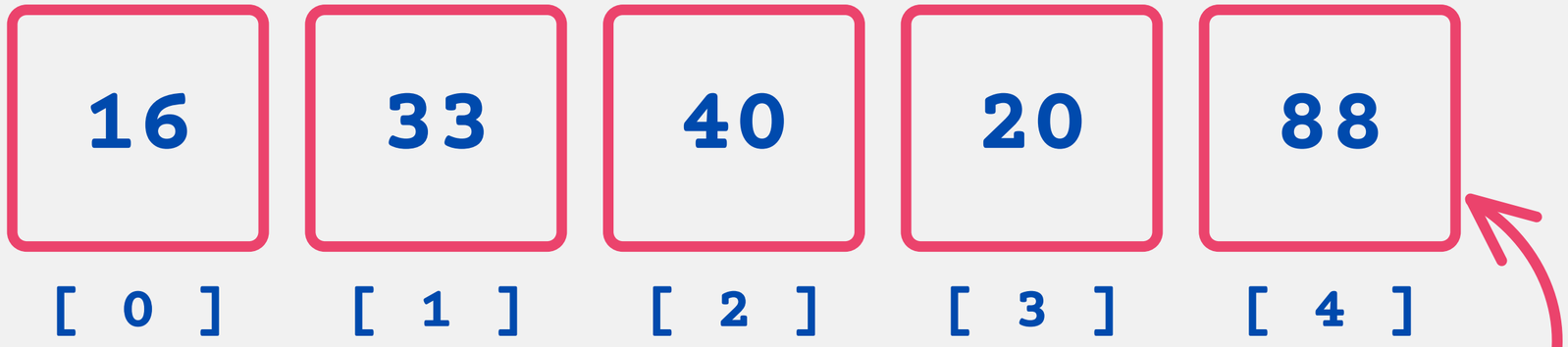
```
vector<int> edades;
```




```
edades.pop_back();
```

Funciones

```
vector<int> edades;
```




```
edades.push_back(88);
```

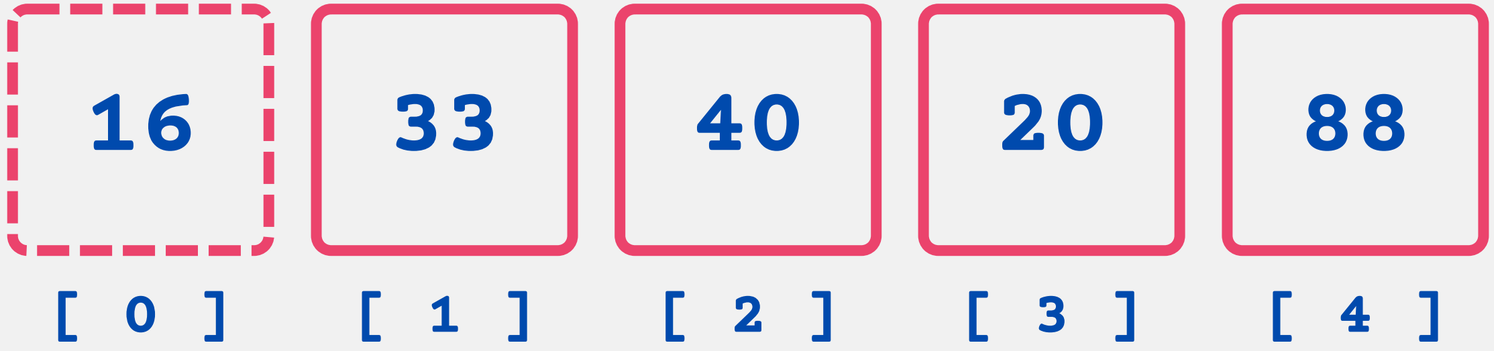
Funciones

- **end():** Retorna un iterador que apunta a lo que va después del último elemento.
- **begin():** Retorna un iterador que apunta al primer elemento.

Funciones

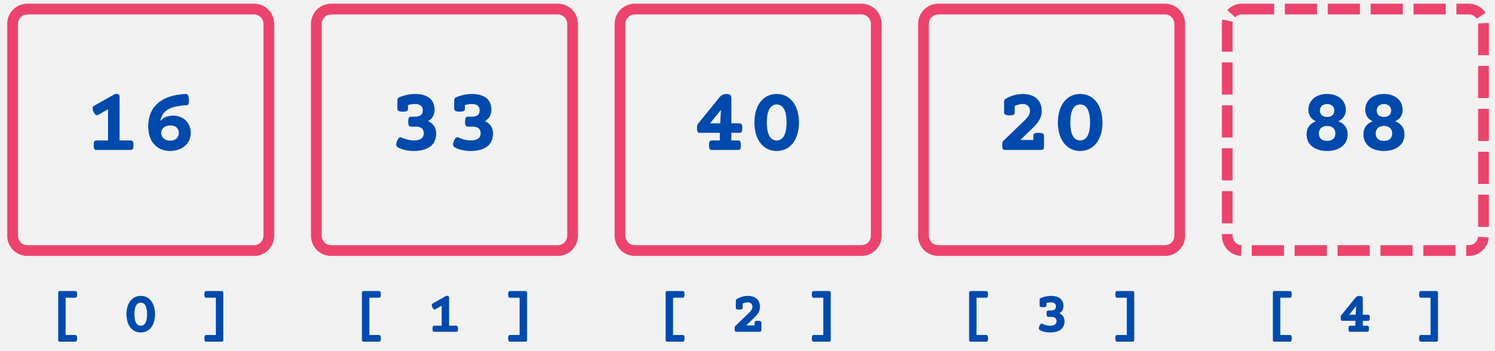
- **erase(*rango*):** Elimina el o los elementos según un rango dado. Este rango se debe indicar con los iteradores vistos anteriormente.

Funciones



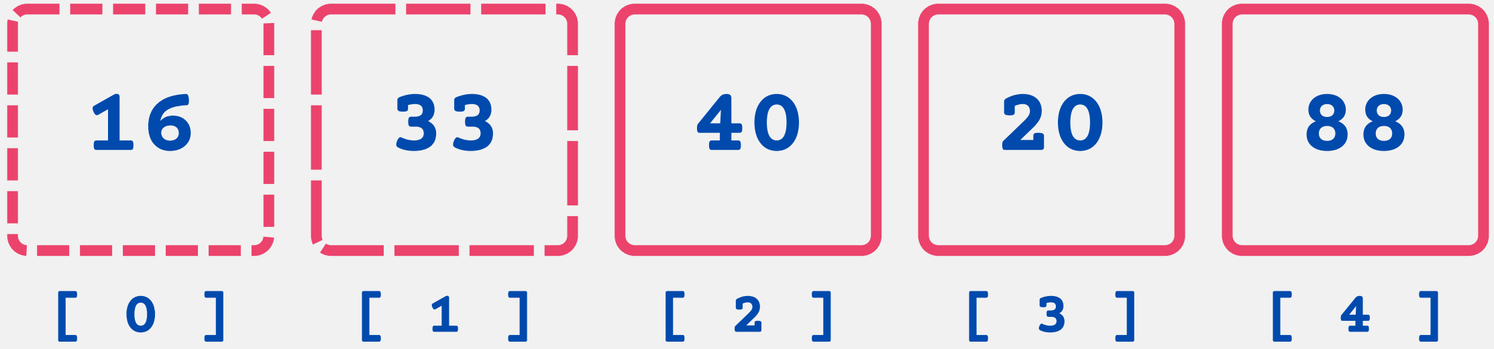

```
edades.erase(edades.begin());
```

Funciones




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

Funciones




```
edades.erase(edades.begin(),edades.begin()+2);
```

Ejercicio 2

- Implemente un programa en c++ que permita al usuario ingresar alumnos a un curso de tamaño ilimitado.
- Los alumno son identificados por su rut y nombre.
- No existe información respecto a la cantidad de evaluaciones que tendrá el curso, todas tienen igual ponderación.
- Implemente métodos que permitan ingresar notas por alumno, calcular el promedio, y finalmente, mostrar la lista de notas de todos los alumnos.

¿Dudas, consultas?

Contacto

+569 53890796
natalia.romero_g@mail.udp.cl

Material

https://github.com/natalia-romero/progra_av2022.git