CURSOS INTERSEMESTRALES

PROTECO

C++ intermedio

Programación Orientada a Objetos: Polimorfismo

Polimorfismo

Polimorfismo significa “muchas formas”.

En programación orientada a objetos, se refiere a la propiedad de los nombres de tener diferentes comportamientos, dependiendo del tipo de objeto con el cuál se esté interactuando.

PROTECO

Sobreescritura y sobrecarga

Sobrecarga

Se refiere a la capacidad de tener distintos constructores o funciones con el mismo nombre. Se diferencia entre ellos mediante el número y tipo de parámetros usados en las distintas definiciones.

"**Mismo nombre, diferentes parámetros**"

PROTECO

Sobreescritura y sobrecarga

Sobreescritura

Se refiere a la capacidad de poder redefinir un atributo, constructor o función de una superclase en alguna de sus clases subhijas. Las funciones deberán conservar el mismo prototipo, pero su definición (implementación, código) será distinta.

"**Mismo nombre, mismos parámetros, diferente comportamiento**"

PROTECO

Sobrecarga de constructores

Utilizado cuando queremos tener más de una manera de crear objetos.

```
class Computadora {
      Computadora(string marca);
      Computadora(string marca, string modelo);
      Computadora(string marca, string modelo;
            float precio);
//...
};
```

PROTECO

Sobrecarga de métodos

Utilizado cuando se define una método con el mismo nombre pero distintos parámetros.

```
class Perro {
      void jugar (Juguete j);
      void jugar (Perro p);
      void jugar (Humano h);
//...
};
```

PROTECO

Sobreescritura de atributos

Utilizado cuando tenemos constantes o variables con valores predefinidos en nuestras clases.

```
class Figura {
public:
      int lados;
};

class Cuadrado : public Figura {
public:
      int lados = 4; //Atributo sobreescrito
};
```

PROTECO

Sobreescritura de métodos

Utilizado cuando queremos definir un comportamiento distinto en alguna subclase.

```
class Ave {
     void volar() { cout << “Volando”;  }
};

class Gallina : public Ave {
     void volar() { cout << “No puedo volar”; }
};
```

Al sobreescribir un método sobrecargado, se tiene que sobreescribir cada definición del mismo.

PROTECO

Sobrecarga de operadores

En C++, nosotros podemos definir como queremos que se compartan la mayoria operadores para nuestras clases.

Por ejemplo, al sumar dos objetos, podemos definir que se sumen sus atributos, u algún otro comportamiento que deseemos.

```
<n_clase> operator<op>(const n_clase& id);
Complejo operator+(const Complejo& comp);
Cuerda operator-(const Cuerda& cuerda);
```

Los operadores punto (. , .*), resolución de ámbito (::), condicional(?:) no pueden ser sobrecargados.

dynamic_cast

Al usar el operador **static_cast**, el programador debe estar completamente seguro que la conversión se podrá realizar, ya que no "avisa" si hubo algún error, por lo tanto es inseguro al programar.

Para ello utilizamos el operador **dynamic_cast**, el cuál nos permite conocer si la conversión fue exitosa o no.

```
<apuntador> = dynamic_cast<clase*>(ap_objeto);
Cuadrado *ap_cuad = NULL;
ap_cuad = dynamic_cast<Figura*>(&fig);
```

PROTECO

Clases polimórficas y funciones virtuales

Una **función o método virtual** es aquel que nos permite llamar siempre a un método en la jerarquía (subclase) más baja.

Esto ayuda a que nosotros podamos llamar a un método de una subclase mientras lo manejamos como uno de su superclase (downcasting).

Al incluir métodos virtuales en una clase, estamos creando una **clase polimórfica**.

PROTECO

Upcasting y downcasting

Upcasting

Un objeto se puede tratar como si fuera un objeto de su superclase (clase padre).

Downcasting

Un objeto se puede tratar como si fuera un objeto de una de sus subclases (clase hijo).