Polimorfismo y clases abstractas

Conceptos

- En la vida real, el poliformismo consiste en aplicar una acción con el mismo nombre a diferentes cosas, realizando tareas internas diferentes
- Por ejemplo:
  - tocar el piano vs. tocar la trompeta
- Estas actividades son similares en su concepto, pero diferentes en su ejecución: empleamos un solo verbo

Conceptos

- De igual forma, en POO podemos tener familias de clases, y emplear un único nombre para realizar actividades similares en diferentes clases
- Por ejemplo, podemos usar un único método llamado “print()” para imprimir valores de cualquier objeto de la familia, en lugar de una función con un nombre distinto en cada clase

Ejemplo

```
class Person
{
    char name[80];
    int age;
public:
    Person(const char n[], const int a);
    void display() { cout << name << " | " << age; }
};

Person::Person(const char n[], const int a)
{
    strncpy(name, n, strlen(n) + 1);
    age = a;
}
```

Ejemplo

```
class Student : public Person {
    char id[20];
public:
    Student(const char i[], const char n[], const int a);
    void display();
};
Student::Student(const char i[], const char n[], const int a):
   Person(n, a) {
    strncpy(id, i, strlen(i) + 1);
}
void Student::display() {
    Person::display();
    cout << " | " << id;
}
```

Ejemplo

```
class Teacher : public Person
{
    double salary;
public:
    Teacher(const double s, const char n[], const int a):
            Person(n, a), salary(s) {}
};
```

Ejemplo

```
int main() {
    Person p1("George Smith", 25);
    Student s1("8NB5", "Jane Marlin", 18);
    Teacher t1(32000.00, "Mr. Frazier", 38);
    Person* ptr[3];
    ptr[0] = &p1;
    ptr[1] = &s1;
    ptr[2] = &t1;
    for (int i = 0; i < 3; i++) {
        ptr[i]->display();
        cout << endl;
    }
    return 0;
};
```

Funciones `virtual`

- Utilizando “virtual” como calificativo de un método indicamos al compilador que no enlace el código correspondiente en tiempo de compilación (*early binding*), sino que el código a ejecutar se decida en tiempo de ejecución dependiendo de la clase concreta que invoque el método (*late binding*)
- Por ejemplo, en la clase base:

```
virtual void display() { cout << name << " | " << age; }
```

Clases abstractas

- Una clase abstracta es aquella clase (base) que contiene al menos un método “pure virtual”
- Un método “pure virtual” no contiene código, solo actúa como modelo para clases derivadas, las cuales están forzadas a implementarlo
- Por ejemplo:
  - `virtual void display() = 0;`
- No se pueden instanciar objetos de clases que contengan algún método “pure virtual”