Ayudantía 1

Clases y Objetos

Benjamín Aceituno

Programación Avanzada S09

Clases

Una clase se puede interpretar como un modelo que define un conjunto de propiedades (variables y funciones).

Sirve para definir las características y comportamientos de una entidad, representando un conjunto de datos como tal.

Las clases en C++ tienen atributos y métodos

Atributos de una clase

Estos son las caracteristicas de la clase y van decretar las variables asignadas a una entidad.

Métodos de una clase

Estas son las funciones que cumplirá la clase como tal y estas serán las “acciones” de una entidad

Ejemplo de clases

```
class Gato {
}
```

Atributos

- Nombre -> “Gordo”
- Peso -> “8”
- Raza -> “Naranjo”
- Dueño -> “BenjamínAceituno”

Métodos

- comer()
- dormir()
- maullar()
- irAlBaño()

Ejemplo de clases

```
class Persona {
}
```

Atributos

- Nombre -> “Vicente”
- Peso -> “70”
- Estatura -> “1.67”
- Ocupación -> “Estudiante”

Métodos

- estudiar()
- dormir()
- comer()
- mostrarInformación()

Sintaxis

Para la sintaxis de clases en C++ tendremos accesos privados y públicos.

Acceso Privado

Solo puede ser accedido desde la misma clase, se usa para los atributos de esta misma.

Acceso Público

Son accesibles fuera y dentro de la clase, se usa para los metodos de esta misma.

```cpp
class Gato {
private:
    // Atributos
    string nombre = "Gordo";
    float peso = 8.0;
    string raza = "Naranjo";
    string dueno = "Benjamín";

public:
    // Métodos
    void comer() {
        cout << nombre << " toy comiendo..." << endl;
    }

    void dormir() {
        cout << nombre << " toy durmiendo..." << endl;
    }

    void maullar() {
        cout << nombre << " miau" << endl;
    }

    void irAlBano() {
        cout << nombre << " yendo al arenero..." << endl;
    }
};
```

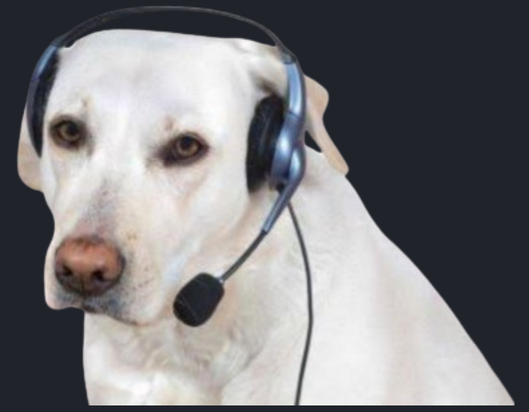
Objetos

Los objetos son una instancia específica de una clase.
Mientras que la clase actúa como un plantilla que define atributos y métodos, el objeto es una entidad que deriva de la clase y sus caracteristicas.

Ejemplo: clase “Mascotas”

nombre: Diego
color: Blanco
tipo: Gato

nombre: Miguel
color: Blanco
tipo: Perro



nombre: Mailo
color: Café
tipo: Perro

Objetos y Constructor

Para inicializar un objeto se hace desde la función int main(), donde se debe crear un puntero que apunta a un objeto de la clase descrita, esto es para la declaración de un solo objeto.

También usaremos el constructor, que es un metodo para inicializar los atributos de una clase al mismo tiempo que se declara

Inicialización de objetos perro y gato

```
int main() {
    // Creación de objetos Perro y Gato
    Mascota *gato = new Mascota("Diego", "Blanco", "Gato");
    Mascota *perro1 = new Mascota("Mailo", "Cafe", "Perro");
    Mascota *perro2 = new Mascota("Miguel", "Blanco", "Perro");
```

Constructor para mascota

```
public:
    // Constructor
    Mascota(string n, string c, string t) : nombre(n), color(c), tipo(t) {}

    // Método de ejemplo
    void mostrarInformacion() {
        cout << "Nombre: " << nombre << ", Color: " << color << ", Tipo: " << tipo << endl;
    }
};
```

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

// Clase Mascota
class Mascota {
private:
    string nombre;
    string color;
    string tipo;

public:
    // Constructor
    Mascota(string n, string c, string t) : nombre(n), color(c), tipo(t) {}

    // Método de ejemplo
    void mostrarInformacion() {
        cout << "Nombre: " << nombre << ", Color: " << color << ", Tipo: " << tipo << endl;
    }
};

int main() {
    // Creación de objetos Perro y Gato
    Mascota *gato = new Mascota("Diego", "Blanco", "Gato");
    Mascota *perro1 = new Mascota("Mailo", "Cafe", "Perro");
    Mascota *perro2 = new Mascota("Miguel", "Blanco", "Perro");

    // Mostrar la información de las mascotas
    gato->mostrarInformacion();
    perro1->mostrarInformacion();
    perro2->mostrarInformacion();

    return 0;
}
```

Definimos los atributos de la clase mascota dentro de private

Se ocupa un constructor y método dentro de public

Se crean los objetos con sus respectivos datos, en el mismo orden que el constructor

Se muestra la información de cada mascota, usando objeto->método()

Declaración y manejo de múltiples objetos

Podemos declarar múltiples objetos para que nuestro programa pueda manejar varias instancias de una clase simultáneamente, hay distintas formas, como por ejemplo, declaración individual y múltiple.

Ingreso individual

```
//Se ingresan perros de manera individual, con su nombre, raza y peso
Perro* perro1 = new Perro("Aldo", "Grande", 45.0);
Perro* perro2 = new Perro("Pepe", "Mediana", 20.0);
Perro* perro3 = new Perro("Blu", "Pequena", 10.0);
```

Ingreso múltiple con arreglos

```
//Se ingresan objetos perros a la clase Perro mediante arreglos
Perro* perros[3];

// Inicializar el arreglo con instancias de Perro

perros[0] = new Perro("Aldo", "Grande", 45.0);
perros[1] = new Perro("Pepe", "Mediana", 20.0);
perros[2] = new Perro("Blu", "Pequeña", 10.0);
```

SET Y GET

Ahora conoceremos dos métodos muy comunes para especificar y obtener atributos fuera de una clase

SET

Como su nombre lo indica, este método sirve para “colocar” un valor a un atributo fuera de la clase

GET

Como su nombre lo indica, este método sirve para “obtener” el valor de un atributo fuera de la clase

```
// Métodos set
void setNombre(string n) {
    nombre = n;
}


// Métodos get
string getNombre() {
    return nombre;
}
```

THIS

En ciertos escenarios es necesario ocupar this, que es un puntero que toma un parámetro y se lo asigna a un atributo, evitando ambigüedades.

```
this->atributo = atributoNuevo;
```

En métodos set y constructor se ocupa cuando los nombres de los parámetros son iguales a los nombres de los atributos de la clase

Para setters:

```
void setNombre(string nombre) {
    this->nombre=nombre;
}

void setColor(string color) {
    color = color;
}

void setTipo(string t) {
    tipo = tipo;
}
```

Para constructor:

```
public:

    Mascota(string nombre, string color, string tipo){
        this->nombre = nombre;
        this->color = color;
        this->tipo = tipo;
    }
```

Ejercicio

Una veterinaria quiere implementar un sistema donde se puedan ingresar distintos perros y ver si estos están bajo peso, peso normal o en sobrepeso, fijandose en su respectiva raza. Implemente un programa en C++ que cumpla los requisitos de la veterinaria

Tenga en cuenta los siguientes datos de peso ideal

Razas pequeñas: 5 a 25 kilos
Razas medianas: 14 a 27 kilos
Razas grandes: 21 a 39 kilos

Contacto

+569 86031881

benjamin.aceituno@mail.udp.cl

benja.mp4

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