code|cademy

Object-Oriented Programming in C++

Classes and Objects

A C++ *class* is a user-defined data type that encapsulates information and behavior about an object.

A class can have two types of *class members*:

Attributes, also known as member data, consist of information about an instance of the class.

Methods, also known as member functions, are functions that can be used with an instance of the class.

An *object* is an instance of a class and can be created by specifying the class name.

```cpp
#include <iostream>

class Dog {
public:
  int age;

  void sound() {
    std::cout << "woof\n";
  }
};

int main() {
  Dog buddy;

  buddy.age = 5;

  buddy.sound();    // Outputs: woof
}
```

Access Specifiers

Access specifiers are C++ keywords that determine the scope of class components:

> `public` : Class members are accessible from anywhere in the program.
>
> `private` : Class members are only accessible from inside the class.

Encapsulation is achieved by declaring class attributes as `private` :

> Accessor functions: return the value of `private` member variables.
>
> Mutator functions: change the value of `private` member variables.

```cpp
#include <iostream>

class Computer {
private:
  int password;

public:
  int getPassword() {
    return password;
  }

  void setPassword(int new_password) {
    password = new_password;
  }
};

int main()
{
  Computer dell;

  dell.setPassword(12345);
  std::cout << dell.getPassword();

  return 0;
}
```

Constructors

For a C++ class, a *constructor* is a special kind of method that enables control regarding how the objects of a class should be created. Different class constructors can be specified for the same class, but each constructor signature must be unique.

A constructor can have multiple parameters as well as default parameter values.

In order to initialize `const` or reference type attributes, use *member initializer lists* instead of normal constructors.

```cpp
#include <iostream>

using namespace std;

class House {
private:
  std::string location;
  int rooms;

public:
  // Constructor with default parameters
  House(std::string loc = "New York", int num = 5) {
    location = loc;
    rooms = num;
  }

  // Destructor
  ~House() {
    std::cout << "Moved away from " <<
location << "\n";
  }
};

int main()
{
  House default_house;   // Calls
House("New York", 5)
  House texas_house("Texas");    // Calls
House("Texas", 5)
  House big_florida_house("Florida",
10);    // Calls House("Florida", 10)

  return 0;
}
```

Inheritance

In C++, a class can inherit attributes and methods from another class. In an inheritance relationship, there are two categories of classes:

> *Base class*: The class being inherited from.

> *Derived class*: The class that inherits from the base class.

It's possible to have multi-level inheritance where classes are constructed in order from the "most base" class to the "most derived" class.

```cpp
#include <iostream>

class Base {
public:
  int base_id;

  Base(int new_base) : base_id(new_base) {}
};

class Derived: public Base {
public:
  int derived_id;

  Derived(int new_base, int new_derived)
    : Base(new_base),
derived_id(new_derived) {}

  void show() {
    std::cout << base_id << " " <<
derived_id;
  }
};

int main() {
  Derived temp(1, 2);

  temp.show(); // Outputs: 1 2

  return 0;
}
```

Polymorphism

In C++, *polymorphism* occurs when a derived class overrides a method inherited from its base class with the same function signature.

Polymorphism gives a method many "forms". Which form is executed depends on the type of the caller object.

```cpp
#include <iostream>

class Employee {
public:
  void salary() {
    std::cout << "Normal salary.\n";
  }
};

class Manager: public Employee {
public:
  void salary() {
    std::cout << "Normal salary and bonus.\n";
  }
};

int main() {
  Employee newbie;
  Manager boss;

  newbie.salary(); // Outputs: Normal salary.
  boss.salary(); // Outputs: Normal salary and bonus.

  return 0;
}
```

Class Members

A class is comprised of class members:

> *Attributes*, also known as member data, consist of information about an instance of the class.

> *Methods*, also known as member functions, are functions that can be used with an instance of the class.

```cpp
class City {

  // Attribute
  int population;

public:
  // Method
  void add_resident() {
    population++;
  }

};
```

Constructor

For a C++ class, a *constructor* is a special kind of method that enables control regarding how the objects of a class should be created. Different class constructors can be specified for the same class, but each constructor signature must be unique.

```cpp
#include "city.hpp"

class City {

  std::string name;
  int population;

public:
  City(std::string new_name, int new_pop);

};
```

Objects

In C++, an *object* is an instance of a class that encapsulates data and functionality pertaining to that data.

```cpp
City nyc;
```

Access Control Operators

C++ classes have access control operators that designate the scope of class members:

```
public

private
```

`public` members are accessible everywhere; `private` members can only be accessed from within the same instance of the class or from friends classes.

```cpp
class City {

  int population;

public:
  void add_resident() {
    population++;
  }

private:
  bool is_capital;

};
```

Class

A C++ class is a user-defined data type that encapsulates information and behavior about an object. It serves as a blueprint for future inherited classes.

```cpp
class Person {

};
```