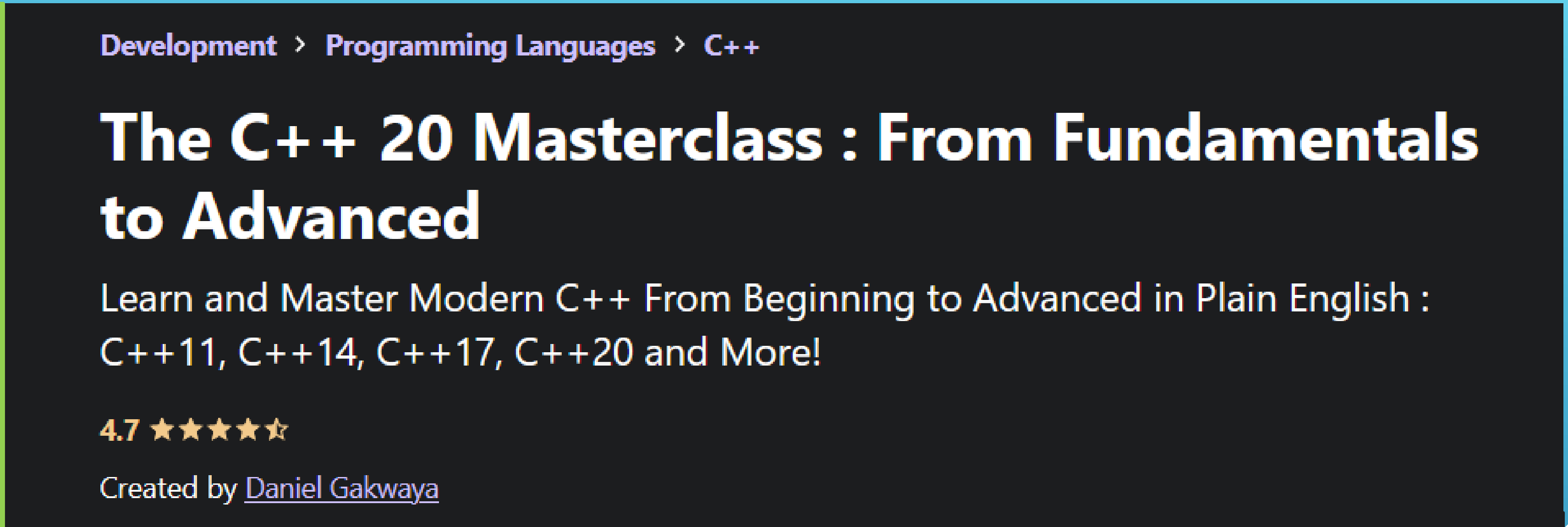
Slides



Development > Programming Languages > C++

The C++ 20 Masterclass : From Fundamentals to Advanced

Learn and Master Modern C++ From Beginning to Advanced in Plain English : C++11, C++14, C++17, C++20 and More!

4.7 ★★★★☆

Created by Daniel Gakwaya

Section : Friends

Slide intentionally left empty

Friend functions

External functions with access to private members of a class

```
class Dog {
public :

     Dog(std::string dog_name, int dog_age){
         this->dog_name = dog_name;
         this->dog_age = dog_age;
     }
private :
     std::string dog_name;
     int dog_age;
 };
```

Potential friend functions

```
void debug_dog_info(const Dog & dog)
{
    std::cout << "Dog name : " << dog.dog_name << ", dog age : "
    << dog.dog_age << std::endl;
}

void debug_dog_info(){
    Dog internal_dog("Pinch",4);
    std::cout << "Dog name : " << internal_dog.dog_name
    << ", dog age : " << internal_dog.dog_age << std::endl;

}
```

Establishing friendship

```
class Dog {
public :

     Dog(std::string dog_name, int dog_age){
         this->dog_name = dog_name;
         this->dog_age = dog_age;
     }

    friend void debug_dog_info(const Dog & dog) ;
    friend void debug_dog_info();

private :
     std::string dog_name;
     int dog_age;
 };
```

- The friend function is first declared and alive somewhere
- The implementation can live in any translation unit. Just be sure that it will be found at link stage
- The declaration has to show up before you call the function
- The class determines who is its friend (through friend declaration)
- The friend declaration can appear in public or private section of the class, either works the same
- We can have multiple friend functions in the same class
- Friend functions can be overloaded
- We have to use the object_name.member_var_name syntax in the friend function
- Friend functions are not member functions
- Friend functions don’t have access to the this pointer

Slide intentionally left empty

Friend classes

Friend classes

```
class Dog {
public :
     Dog(std::string dog_name, int dog_age);
     friend class Cat;

private :
     std::string dog_name;
     int dog_age;
};


class Cat {
public :
     Cat (std::string cat_name);

    void show_info_about_dog( const Dog & dog) const{
        std::cout << "dog name : " << dog.dog_name << std::endl;
    }

private :
    std::string cat_name;
};
```

Friends : Summary

Giving access to our own member variables to external entities. External entities can be either functions or classes

Friend functions

```
class Dog {
public :

     Dog(std::string dog_name, int dog_age){
         this->dog_name = dog_name;
         this->dog_age = dog_age;
     }

    friend void debug_dog_info(const Dog & dog) ;
    friend void debug_dog_info();

private :
     std::string dog_name;
     int dog_age;
 };
```

Friend classes

```
class Dog {
public :
     Dog(std::string dog_name, int dog_age);
     friend class Cat;

private :
     std::string dog_name;
     int dog_age;
};


class Cat {
public :
     Cat (std::string cat_name);

    void show_info_about_dog( const Dog & dog) const{
        std::cout << "dog name : " << dog.dog_name << std::endl;
    }

private :
    std::string cat_name;
};
```

As a general guideline, don’t set up friendship relationships easily, as your friends have the power to do the most damage to your internal parts. Just like in real life! Do this after some careful consideration.

Slide intentionally left empty