COMP 322 Lecture 6 - Classes in C++

Junji Duan

2024/2/9

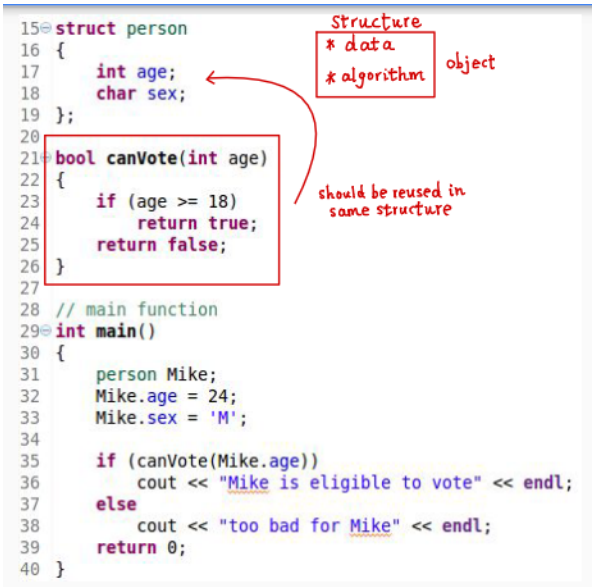
Today's Outline

- Review of OO concept
- C structs
- C++ Classes
- Constructors
- Destructors

OO programming: Quick review

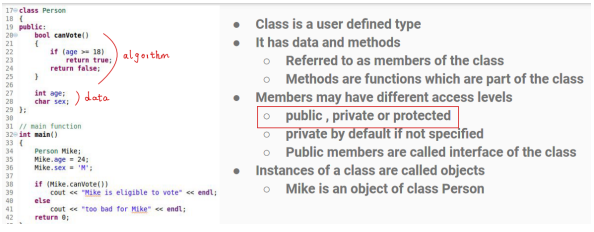
- Approach to design modular and reusable systems
- Programming is about manipulating data through code (methods or algorithms)
  - Object: is a coupling of both data and methods
- Extension to the concept of structures
  - Not only we group multiple data elements but we also attach the intelligence needed to manipulate the data (also called encapsulation)

C structs: data and code are separated

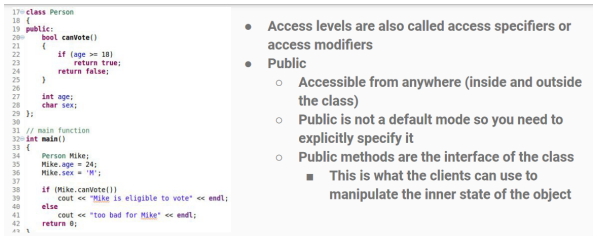



```
15 struct person
16 {
17     int age;
18     char sex;
19 };
20
21 bool canVote(int age)
22 {
23     if (age >= 18)
24         return true;
25     return false;
26 }
27
28 // main function
29 int main()
30 {
31     person Mike;
32     Mike.age = 24;
33     Mike.sex = 'M';
34
35     if (canVote(Mike.age))
36         cout << "Mike is eligible to vote" << endl;
37     else
38         cout << "too bad for Mike" << endl;
39     return 0;
40 }
```

C++ Classes: introduction

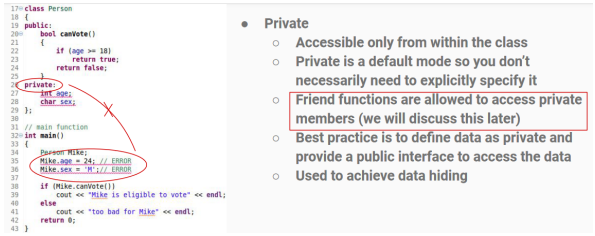


C++ Classes: access levels (public)



- Access levels are also called access specifiers or access modifiers
- Public
  - Accessible from anywhere (inside and outside the class)
  - Public is not a default mode so you need to explicitly specify it
  - Public methods are the interface of the class
    - This is what the clients can use to manipulate the inner state of the object

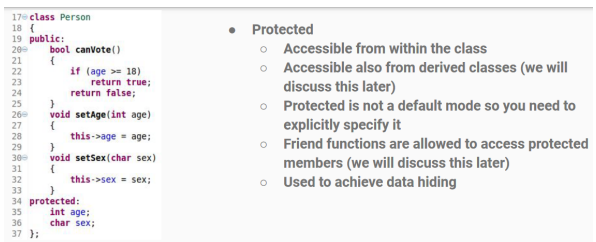
C++ Classes: access levels (private)



- Private
  - Accessible only from within the class
  - Private is a default mode so you don't necessarily need to explicitly specify it
  - Friend functions are allowed to access private members (we will discuss this later)
  - Best practice is to define data as private and provide a public interface to access the data
  - Used to achieve data hiding

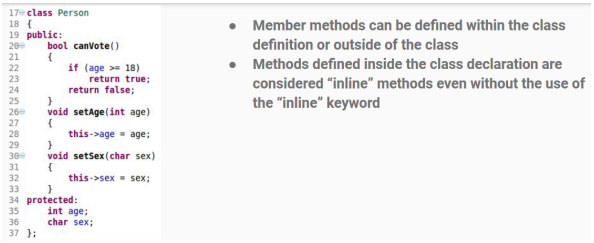
C++ Classes: access levels (private)

C++ Classes: access levels (protected)

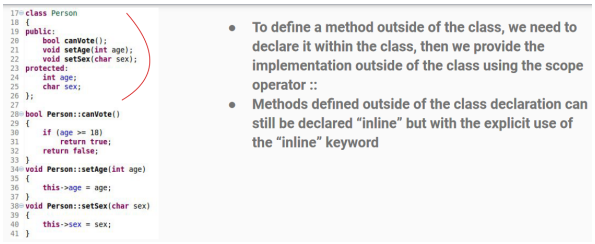


- Protected
  - Accessible from within the class
  - Accessible also from derived classes (we will discuss this later)
  - Protected is not a default mode so you need to explicitly specify it
  - Friend functions are allowed to access protected members (we will discuss this later)
  - Used to achieve data hiding

C++ Classes: method definition



- Member methods can be defined within the class definition or outside of the class
- Methods defined inside the class declaration are considered "inline" methods even without the use of the "inline" keyword



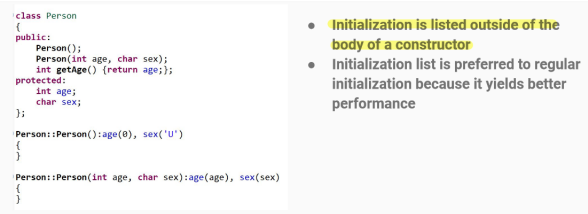
- To define a method outside of the class, we need to declare it within the class, then we provide the implementation outside of the class using the scope operator ::
- Methods defined outside of the class declaration can still be declared "inline" but with the explicit use of the "inline" keyword

C++ classes: constructors

- When instantiating a class, a special method is implicitly called first
  - This method is the constructor
- Every class has a constructor (at least one)
- If a constructor is not provided by the programmer, the compiler will provide a default implicit constructor (that does basically nothing)
  - This is how the construction of the Person class from the previous example was possible
- The Constructor method:
  - is used to initialize the data members of a class
  - must have the same name as the class
  - must be declared public in general
    - There are some exceptions when implementing advanced design patterns
  - does not have a return type
    - Constructors dont return values

- Constructor can be personalized using parameters
- User can define as many different constructors as needed

Constructor: initialization list



```
class Person
{
public:
    Person();
    Person(int age, char sex);
    int getAge() {return age;};
protected:
    int age;
    char sex;
};

Person::Person():age(0), sex('U')
{
}

Person::Person(int age, char sex):age(age), sex(sex)
{
}
```

- Initialization is listed outside of the body of a constructor
- Initialization list is preferred to regular initialization because it yields better performance

Public, Private and Protected

In C++, public, private, and protected are three types of access specifiers used to set the access level for class members. Below is an explanation of each access specifier and its usage through an example.

- public
  - Public members can be accessed anywhere, including outside the class. This means that if a class member is declared as public, any external code can access it directly.
- private
  - Private members can only be accessed by member functions of the same class. This means that if you attempt to access a private member from outside the class, the compiler will throw an error.
- protected
  - Protected members are similar to private members in that they restrict external access. However, unlike private, protected members are accessible in derived classes (subclasses).

Suppose we have a class named Person with three members: name, age, and height. We want name to be accessible by anyone, height to be accessible only within the class or its subclasses, and age to be accessible only within the class.

```
class Person {
public:
    string name;  // 公有成员，任何地方都能访问

protected:
    int height;  // 保护成员，只能在类内部及其子类中访问

private:
    int age;  // 私有成员，只能在类内部访问

public:
    Person(string n, int h, int a) : name(n), height(h), age(a) {}

    void printInfo() {
        cout << "姓名：" << name << ", 身高：" << height << ", 年龄：" << age << endl;
    }
};

class Student : public Person {
public:
    Student(string n, int h, int a) : Person(n, h, a) {}

    void printHeight() {
        cout << "学生身高：" << height << endl;  // 可以访问protected成员
    }
};
```

```
int main() {
    Person p("张三", 170, 30);
    p.printInfo();
    // cout << p.age;  // 错误：age是私有成员，不能在类外访问
    cout << "姓名：" << p.name << endl;  // 正确：name是公有成员

    Student s("李四", 160, 20);
    s.printHeight();  // 正确：派生类可以访问protected成员
    // cout << s.height;  // 错误：height在派生类外部不可访问
}
```

In this example, name is a public member, so it can be accessed directly in the main function. height is a protected member, so it can be accessed in the member function printHeight of the Student class (derived from Person), but not directly in the main function. Finally, age is a private member, so it cannot be accessed directly in the main function or in the Student class.

Struct, Class

In C++, struct and class are both composite data types used for data encapsulation, with the main difference lying in their default access level and default inheritance mode. In C++, types defined using struct or class can contain data members (member variables) and function members (member functions or methods), thereby encapsulating data and behavior.

struct

- Default access level: The members of a struct are public by default, meaning that by default, members of a struct can be accessed from outside the struct.
- Usage: Typically used for smaller data structures, more focused on storing data rather than emphasizing behavior.

```cpp
struct Person {
    std::string name;
    int age;

    void printInfo() const {
        std::cout << "Name: " << name << ", Age: " << age << std::endl;
    }
};

int main() {
    Person p1;
    p1.name = "张三";
    p1.age = 30;
    p1.printInfo(); // 直接访问并使用Person的数据成员和成员函数

    return 0;
}
```

class

- Default access level: The members of a class are private by default, meaning that by default, members of a class cannot be accessed from outside the class, ensuring encapsulation and data safety.
- Usage: Usually used to define complex data types containing both data and function members, more focused on the behavior and interface of objects.

```cpp
class Animal {
private:
    std::string name;
    int age;

public:
    Animal(const std::string& n, int a) : name(n), age(a) {}

    void printInfo() const {
        std::cout << "Name: " << name << ", Age: " << age << std::endl;
    }
};

int main() {
    Animal a1("小白", 5);
    a1.printInfo(); // 使用Animal的公有成员函数

    return 0;
}
```

Summary

- The essential difference between struct and class lies in their default access level (struct is public, class is private).
- In C++, both struct and class can be used to define types containing data members and member functions.
- The choice between struct and class often depends on your requirements for the default access level of the type and how you intend to use it (more emphasis on data or behavior).

P.S. Constructor and Destructor notes are in Lecture 7.