友元

小组成员：

邵东林 :202307885

谭棵 :202306630

苏怡力 :202305962

目录

Contents

思想与好处

Background significance of the topic

01.

好处

- 好处
  - 提高程序效率：友元函数可以直接访问类的私有和保护成员，避免了通过公有接口的间接访问，从而减少了不必要的函数调用和数据拷贝，提高了程序的运行效率。
  - 简化代码结构：友元函数可以简化代码结构，因为它可以直接访问私有和保护成员，不需要额外的接口函数来间接访问这些成员。
  - 增强表达能力：在某些情况下，友元函数可以提供更直观和简洁的方式来实现功能，例如在运算符重载中，友元函数可以让类对象之间直接参与运算，而不是通过函数调用的方式。
  - 实现信息隐藏的：友元函数和友元类可以控制其他类对自己的访问权限，从而实现了信息隐藏，保护了类的封装性和安全性。
  - 支持特定设计模式：在某些设计模式中，如观察者模式或策略模式，友元可以用来实现更灵活和高效的设计。
- 思想
  - 面向对象编程：友元函数是面向对象编程中的一部分，它体现了类之间的合作关系，允许一个类访问另一个类的内部细节，以实现更复杂的功能。
  - 信息共享：友元函数允许类之间共享数据，特别是在需要多个类协作完成某项任务时，友元函数可以简化这种协作，提高开发效率。
  - 初始化的集中管理：友元函数提供了一个集中管理对象初始化的地方，使得初始化逻辑更加清晰和易于维护。
  - 运算符重载：友元函数在运算符重载中起到重要作用，它允许类对象之间直接参与运算，而不是通过函数调用的方式，从而提供更直观的使用体验。

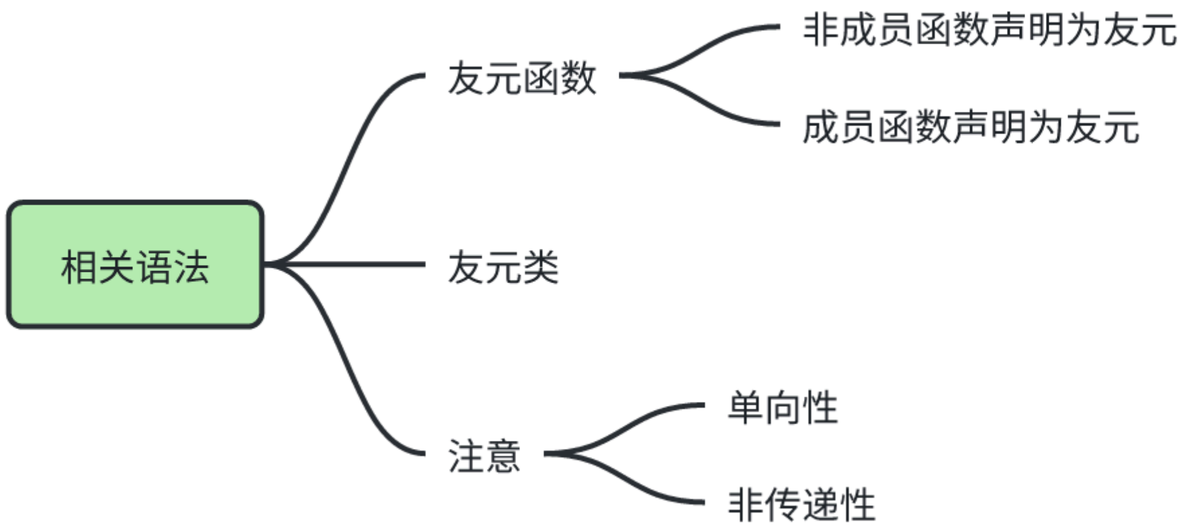
注意

友元函数虽然在某些情况下提供了便利，但使用时需谨慎，因为它可能降低代码的可维护性和可读性。因此，友元的使用应仅限于必要的情况，以确保代码的合理性和有效性。

相关语法

Relevant Syntax

02.



在 C++ 中，一个类中可以有 public、protected、private 三种属性的成员，通过对象可以访问 public 成员，只有本类中的函数可以访问本类的 private 成员。

友元（ friend）可以使得其他类中的成员函数以及全局范围内的函数访问当前类的 private 成员。friend 的意思是朋友，或者说是好友，与好友的关系显然要比一般人亲密一些。我们会对好朋友敞开心扉，倾诉自己的秘密，而对一般人会谨言慎行，潜意识里就自我保护。在 C++ 中，这种友好关系可以用friend关键字指明，借助友元可以访问与其有好友关系的类中的私有成员。

友元函数--非成员函数声明为友元

```
class ClassName {
public:
    // 默认构造函数
    ClassName() {
        // 初始化代码
    }

    // 带参数的构造函数
    ClassName(Type1 param1, Type2 param2) {
        // 初始化代码，使用参数
    }

    // 成员初始化列表
    ClassName(Type1 param1, Type2 param2) : member1(param1), member2(param2) {
        // 其他初始化代码
    }
};
```

解释：

Person类有一个私有成员age。

printAge是一个非成员函数，但它被声明为Person的友元函数。

因此，printAge可以直接访问Person的私有成员age。

友元函数--成员函数声明为友元

```cpp
#include <iostream>

class Person
{
private:
    int age;

public:
    Person(int a) : age(a) {}

    // 声明非成员函数为友元    identifier "Type1" is undefined
    friend void printAge(const Person &p);
};

// 定义友元函数
void printAge(const Person &p)
{    identifier "Type1" is undefined
    std::cout << "Age: " << p.age << std::endl; // 可以直接访问Person的私有成员
}

int main()
{
    Person p(25);
    printAge(p); // 输出: Age: 25
    return 0;
}
```

- Student类有一个私有成员score。
- School类的成员函数checkStudent被声明为Student的友元。
- 因此，checkStudent可以直接访问Student的私有成员score。

```cpp
#include <iostream>

class Student;

class School
{
public:
    void checkStudent(const Student &s);
};

class Student
{
private:
    int score;

public:
    Student(int s) : score(s) {}

    // 声明School类的成员函数为友元
    friend void School::checkStudent(const Student &s);
};

void School::checkStudent(const Student &s)
{
    if (s.score >= 60)
    {
        std::cout << "Pass!" << std::endl;
    }
    else
    {
        std::cout << "Fail!" << std::endl;
    }
}

int main()
{
    Student s(75);
    School school;
    school.checkStudent(s); // 输出: Pass!
    return 0;
}
```

解释：

- Person类有一个私有成员age。
- Doctor类被声明为Person的友元类。
- 因此，Doctor类的所有成员函数（如checkAge）都可以直接访问Person的私有成员age。

示例分析

Example Analysis

03.

1. 定义了三个类：Servant、NoblePhantasm 和 Battle。它们之间的关系如下：
2. Servant 类：
3. 用于模拟英灵的属性，包括名称、职阶和强度。
4. 提供了构造函数和成员函数 show，用于展示英灵及其宝具的信息。
5. 声明了非成员函数 compareStrength 和 Battle 类的成员函数 startBattle 为友元，允许它们访问 Servant 的私有成员。
6. NoblePhantasm 类：
7. 用于模拟宝具的属性，包括名称、等级和类型。
8. 声明了 Servant 类为友元类，允许 Servant 的成员函数访问 NoblePhantasm 的私有成员。
9. Battle 类：
10. 提供了静态成员函数 startBattle，用于模拟两个英灵之间的战斗。
11. 通过友元机制访问 Servant 的私有成员。
12. 主函数：
13. 创建了多个 Servant 和 NoblePhantasm 对象。
14. 调用了非成员函数 compareStrength 和 Battle 类的成员函数 startBattle，展示了友元机制的实际应用。

https://xcnx25vdviba.feishu.cn/wiki/QqRJwrhvtio7pSkuUPPciLGDnWg?fromScene=spaceOverview

```
(yolov8) D:\code\Algorithm_beginner_learning_notes>cd "d:\code\Algorithm_beginner_learning_notes\" && g++
 1.cpp -o 1 && "d:\code\Algorithm_beginner_learning_notes\"1
英灵：阿尔托莉雅，职阶：Saber，强度：90
宝具：誓约胜利之剑，等级：EX，类型：对军宝具
英灵：卫宫，职阶：Archer，强度：85
宝具：无限剑制，等级：E~A++，类型：现实宝具
英灵：美杜莎，职阶：Rider，强度：88
宝具：波塞冬之子，等级：A++，类型：对城宝具
英灵：迪卢木多，职阶：Lancer，强度：87
宝具：丛云Swift，等级：A+++，类型：对人宝具
阿尔托莉雅 的强度 (90) 高于 卫宫 的强度 (85)
美杜莎 的强度 (88) 高于 迪卢木多 的强度 (87)
战斗开始：阿尔托莉雅 vs 美杜莎
阿尔托莉雅 获胜！
战斗开始：卫宫 vs 迪卢木多
迪卢木多 获胜！
```

知识点总结

Key Points summary

04.

总结

- 友元机制虽然强大，但会破坏类的封装性，因此应谨慎使用。
- 友元关系是单向的，即如果A是B的友元，B不一定也是A的友元。
- 友元关系不能继承，即如果A是B的友元，A的子类不一定是B的友元。

- 除非有必要，一般不建议把整个类声明为友元类，而只将某些成员函数声明为友元函数，这样更安全一些。

THANK YOU FOR WATCHING

恳请大家批评指正

小组成员：邵东林 谭棵 苏怡力