C++对象切片探秘：派生类对象如何被‘切割’?

作者：微信公众号：【架构师老卢】 12-11 15:16 250

概述：C++中的对象切片指通过将派生类对象赋值给基类对象，导致派生部分被“切掉”，只保留基类部分。这可能发生在值传递、赋值等操作中。对象切片的基础功能示例展示了派生类对象赋值给基类对象时的现象，而高级功能示例则展示了通过基类指针实现派生类对象的访问和多态。

对象切片（Object Slicing）是指通过将派生类对象赋值给基类对象，导致派生类对象的派生部分被“切掉”，只保留基类部分的现象。这通常发生在使用值传递或赋值操作时。

基础功能：

示例源代码：

```cpp
#include <iostream>
#include <string>

class Base {
public:
    Base(int baseData) : baseData_(baseData) {}
    virtual void Print() const {
        std::cout << "基类数据：" << baseData_ << std::endl;
    }
private:
    int baseData_;
};

class Derived : public Base {
public:
    Derived(int baseData, const std::string& derivedData)
        : Base(baseData), derivedData_(derivedData) {}
    void Print() const override {
        std::cout << "基类数据：" << GetBaseData() << ", 派生类数据：" << derivedData_ << std::endl;
    }
private:
    std::string derivedData_;
};

void DisplayBaseObject(const Base& obj) {
    obj.Print();
}

int main() {
    Derived derivedObject(42, "派生数据");

    // 对象切片发生，只保留基类部分
    Base baseObject = derivedObject;
    DisplayBaseObject(baseObject);

    return 0;
}
```

在这个示例中，`Derived` 类公有继承自 `Base` 类，当派生类对象 `derivedObject` 被赋值给基类对象 `baseObject` 时，发生了对象切片。虽然 `derivedObject` 包含派生类的数据成员，但只有基类部分被保留。

高级功能：

示例源代码：

```cpp
#include <iostream>
#include <memory>

class Base {
public:
    virtual void Print() const {
        std::cout << "基类数据：" << baseData_ << std::endl;
    }
protected:
    int baseData_ = 0;
};

class Derived : public Base {
public:
    void Print() const override {
        std::cout << "基类数据：" << baseData_ << ", 派生类数据：" << derivedData_ << std::endl;
    }
private:
    std::string derivedData_ = "派生数据";
};

int main() {
    std::unique_ptr<Base> basePtr = std::make_unique<Derived>(); // 指向派生类对象的基类指针

    // 通过基类指针调用虚函数，实现多态
    basePtr->Print();

    return 0;
}
```

在这个示例中，通过使用智能指针 `std::unique_ptr`，可以实现指向派生类对象的基类指针。通过基类指针调用虚函数 `Print`，实现了多态性。这也是对象切片的一种应用，通过基类指针可以访问派生类的虚函数。