纯虚函数和抽象类是C++中用于实现抽象数据类型和多态性的重要概念。

纯虚函数和抽象类：

1.纯虚函数： 一个纯虚函数是在基类中声明的虚函数，但没有提供具体的实现。它的声明以 virtual 关键字和 = 0 进行标记，如下所示：

class AbstractBase {

public:

virtual void pureVirtualFunction() const = 0;

};

没有提供纯虚函数的具体实现，是因为这个责任被派生类来完成。

2.抽象类： 包含至少一个纯虚函数的类被称为抽象类。抽象类不能被实例化，即不能创建抽象类的对象。通常，抽象类用于定义接口，而具体的实现则由派生类完成。

虚函数表和虚指针：

3.虚函数表： 在涉及多态性的情况下，每个类都有一个虚函数表（vtable），其中包含指向虚函数的指针。虚函数表使得在运行时能够动态调用正确的函数。

4.虚指针： 每个对象的内存布局中可能包含一个指向其类的虚函数表的指针，称为虚指针。这个指针允许程序在运行时找到正确的虚函数表。

示例：

#include &lt;iostream&gt;

// 抽象基类 AbstractBase

class AbstractBase {

public:

// 纯虚函数

virtual void pureVirtualFunction() const = 0;

};

// 派生类 DerivedClass

class DerivedClass : public AbstractBase {

public:

// 实现纯虚函数

void pureVirtualFunction() const override {

std::cout &lt;&lt; "DerivedClass::pureVirtualFunction()" &lt;&lt; std::endl;

}

};

int main() {

// 尝试实例化抽象基类，会导致编译错误

// AbstractBase obj; // Error: cannot instantiate abstract class

// 创建派生类对象

DerivedClass derivedObj;

// 调用虚函数

derivedObj.pureVirtualFunction();

return 0;

}

在这个示例中，AbstractBase 是一个抽象基类，包含一个纯虚函数。由于包含了纯虚函数，无法创建 AbstractBase 的实例。然后，我们创建了一个派生类 DerivedClass，实现了基类的纯虚函数。通过虚函数表和虚指针的机制，程序在运行时能够正确调用派生类的实现。这种设计允许我们通过基类的指针或引用来操作派生类的对象，实现了多态性。