

第10讲 const修饰符



目录 CONTENTS



1 const修饰符概述

2 const修饰变量、指针及引用

3 const修饰参数、返回值

4 const修饰成员及对象

1. const修饰符概述

- ④ const关键字是一种修饰符，是在编译过程中，给编译器的一些“要求”或“提示”，但它本身并不产生任何实际代码。
- ④ const的含义是“不变的”、“只读”，作用是限制某些内容不被改变，并利用编译器帮助检查正确性。
- ④ const修饰符可以用来修饰变量、指针、引用、成员变量、成员函数和对象等。

2. const修饰变量

- const修饰变量表示该变量的值只读，不能被改变，因此必须在定义时初始化。

const 数据类型 变量名 = 初始值

数据类型 const 变量名 = 初始值

```
const int STUDENT_NUMBER = 50;  
STUDENT_NUMBER = 60; ❌
```

3. const修饰指针

- ◎ const用来修饰指针可以有两种效果，分别为：常量指针和指针常量。
- ◎ 常量指针
 - ▣ 也称为常指针，是指不可以通过指针间接来修改指针所指向内容，常量指针定义时可以不初始化。

const 数据类型* 指针名 [=地址]

```
double rate = 1.2;  
const double* p;  
p = &rate;
```

```
*p = 1.3;  
rate = 1.3;
```




3. const修饰指针


◎ 指针常量

- 指针**内容为常量**，指针初始化后不能更改指向，指针常量定义时**必须**初始化。

数据类型* const 指针名 = 地址

```
double rate = 1.2;  
double* const p = &rate;
```

```
double other = 1.5;  
p = &other; 
```

```
*p = 1.3; 
```


3. const修饰指针


◎ 常量指针与指针常量

- 可以同时用const修饰指针与指针指向内容。

const 数据类型* const 指针名 = 地址

```
double rate = 1.2;  
const double* const p = &rate;
```

```
double other = 1.5;  
p = &other; 
```

```
*p = 1.3; 
```

4. const修饰引用

- const修饰引用与常量指针类似，是指不能通过引用间接修改变量的内容，但并不影响变量原有的使用性质。

const 数据类型& 引用名 = 变量

```
double rate = 1.2;  
const double& r = rate;
```

```
r = 1.3;  
rate = 1.3;
```



5. const修饰函数参数

- 在函数调用的过程中，函数的参数是建立在栈上的变量，因此也可以通过 const 进行修饰。
- 当用const修饰函数参数时，表示函数内部不能够改变这个参数的内容。

```
void show(const int value)
{
    cout<<"数值为: "<<value<<endl;
}
int main()
{
    int v = 100;
    show(v);
    return 0;
}
```

```
类名(const 类名 &ob){
    .....
}
```

6. const修饰函数返回值

- ④ const在修饰函数返回值时，返回值可以是**指针**，也可以是**引用**。
- ④ 如果函数返回值是指针，添加 const 修饰后，返回指针指向的内容不能被修改，而且只能**赋值给使用const 修饰的同类型指针**。

6. const修饰函数返回值

- const在修饰函数返回值时，返回值可以是指针，也可以是引用。

- 如果函数返回的内容不是常量，那么就不能使用const修饰。类型指针。

```
const int* getPtr()
{
    int* p = new int;
    return p;
}

int main()
{
    int v = 100;
    const int* q = getPtr();
    q = &v;
    *q = 5;
    return 0;
}
```



返回指针指
const 修饰的同

6. const修饰函数返回值

- 如果函数返回值是引用，添加 const 修饰后，**不能**将函数调用表达式作为**左值使用**。

6. const修饰函数返回值

- 如果函数返回值
调用表达式作

能将函数

```
int& setValueNonConst(int& value)
{
    value = 10;
    return value;
}
const int& setValueConst(int& value)
{
    value = 10;
    return value;
}
int main()
{
    int v = 100;
    setValueNonConst(v) = 20;
    setValueConst(v) = 30;
    return 0;
}
```



7. const修饰成员变量

- ⊙ const修饰成员变量时，成员变量为只读，初始化后不能够再被修改。
- ⊙ const成员变量必须通过初始化列表完成初始化，不能在构造函数中初始化。

```
class Data
{
private:
    const int m_nValue;
public:
    Data(int v): m_nValue(v)
    {
    }
};
```

8. const修饰成员函数

- const 修饰成员函数时，函数内不能修改任何成员变量，除了被mutable修饰的成员变量。

```
class Data
{
private:
    const int m_nValue;
public:
    Data(int v): m_nValue(v)
    {
    }
    int getData() const
    {
        return m_nValue;
    }
};
```

```
int main()
{
    Data d(100);
    int v = d.getData();
    return 0;
}
```

9. const修饰对象

- const 也可以修饰对象，对象创建后，不能做任何修改，而且只能调用const成员函数。

9. const修饰对象

◎ C++
而

```
class Data
{
private:
    const int m_nValue;
public:
    Data(int v): m_nValue(v)
    {
    }
    int getData() const
    {
        return m_nValue;
    }
    void show()
    {
        cout<<m_nValue<<endl;
    }
};
```

```
int main()
{
    const Data d(100);
    int v = d.getData();
    d.show();
    return 0;
}
```



9. const修饰对象

◎ const 对象不能调用非const成员函数的原因是什么？

□ 每个非const成员函数都包含一个隐含参数this，定义为：

类名* const this

□ 每个const成员函数也包含一个隐含参数this，定义为：

const 类名* const this

□ 当const对象调用非const成员函数时，需要将const对象的地址作为参数传递给参数（**类名* const this**），const对象要求不能修改任何成员变量，而这个this可以修改成员变量，所以参数不一致，出现错误。

□ 而当const对象调用const成员函数时，参数匹配，正确执行。