# 编程语言

## C/C++

### 1.1 const

#### 1.1.1作用

1. 修饰变量，说明该变量不可以被改变；
2. 修饰指针，分为指向常量的指针和指针常量；
3. 常量引用，经常用于形参类型，即避免了拷贝，又避免了函数对值的修改；
4. 修饰成员函数，说明该成员函数内不能修改成员变量。

#### 1.1.2 const 使用

// 类

class A

{

private:

const int a; // 常对象成员，只能在初始化列表赋值

public:

// 构造函数

A() : a(0) { };

A(int x) : a(x) { }; // 初始化列表

// const可用于对重载函数的区分

int getValue(); // 普通成员函数

int getValue() const; // 常成员函数，不得修改类中的任何数据成员的值

};

void function()

{

// 对象

A b; // 普通对象，可以调用全部成员函数、更新常成员变量

const A a; // 常对象，只能调用常成员函数

const A \*p = &a; // 常指针

const A &q = a; // 常引用

// 指针

char greeting[] = "Hello";

char\* p1 = greeting; // 指针变量，指向字符数组变量

const char\* p2 = greeting; // 指针变量，指向字符数组常量

char\* const p3 = greeting; // 常指针，指向字符数组变量

const char\* const p4 = greeting; // 常指针，指向字符数组常量

}

// 函数

void function1(const int Var); // 传递过来的参数在函数内不可变

void function2(const char\* Var); // 参数指针所指内容为常量

void function3(char\* const Var); // 参数指针为常指针

void function4(const int& Var); // 引用参数在函数内为常量

// 函数返回值

const int function5(); // 返回一个常数

const int\* function6(); // 返回一个指向常量的指针变量，使用：const int \*p = function6();

int\* const function7(); // 返回一个指向变量的常指针，使用：int\* const p = function7();

### **1.2 static**

#### 1.2.1作用

1. 修饰普通变量，修改变量的存储区域和生命周期，使变量存储在静态区，在 main 函数运行前就分配了空间，如果有初始值就用初始值初始化它，如果没有初始值系统用默认值初始化它。
2. 修饰普通函数，表明函数的作用范围，仅在定义该函数的文件内才能使用。在多人开发项目时，为了防止与他人命名空间里的函数重名，可以将函数定位为 static。
3. 修饰成员变量，修饰成员变量使所有的对象只保存一个该变量，而且不需要生成对象就可以访问该成员。
4. 修饰成员函数，修饰成员函数使得不需要生成对象就可以访问该函数，但是在 static 函数内不能访问非静态成员。Static函数只能访问static成员。

### 1.3 this指针

#### 1.3.1 作用

1. this 指针是一个隐含于每一个非静态成员函数中的特殊指针。它指向调用该成员函数的那个对象。
2. 当对一个对象调用成员函数时，编译程序先将对象的地址赋给 this 指针，然后调用成员函数，每次成员函数存取数据成员时，都隐含使用 this 指针。
3. 当一个成员函数被调用时，自动向它传递一个隐含的参数，该参数是一个指向这个成员函数所在的对象的指针。
4. this 指针被隐含地声明为: ClassName \*const this，这意味着不能给 this 指针赋值；在 ClassName 类的 const 成员函数中，this 指针的类型为：const ClassName\* const，这说明不能对 this 指针所指向的这种对象是不可修改的（即不能对这种对象的数据成员进行赋值操作）；
5. this 并不是一个常规变量，而是个右值，所以不能取得 this 的地址（不能 &this）。
6. 在以下场景中，经常需要显式引用 this 指针：
7. 为实现对象的链式引用；
8. 为避免对同一对象进行赋值操作；
9. 在实现一些数据结构时，如 list。