static成员存在的意义是让类的全体对象来访问一个全局变量。

非static成员存在于每一个类对象中，而static成员变量独立于该类的任意对象而存在，每一个static成员只与类相关联，而与类对象无关。

static成员函数可以访问类的static成员变量，但不能直接访问类的非static成员变量。（这里指的是本类的非static成员，可以访问其他类的非static成员，参考程序class\_test47）

例：程序class\_test4

#ifndef STUDENT\_H

#define STUDENT\_H

class Student

{

public:

Student(char\* name\_ptr = "no name");

~Student();

static int number(); // 静态成员函数

protected:

static int number\_students; // 静态成员变量

char name[40];

};

#endif

源文件：

#include <iostream>

#include <string>

#include "student.h"

int Student::number\_students = 0; // 初始化静态数据成员

//静态数据成员在类声明外分配空间和初始化

//常量对象和引用用冒号运算符初始化（在类的初始化列表中初始化）

Student::Student(char\* name\_ptr)

{

std::cout << "Create one student\n";

strcpy(name, name\_ptr);

++number\_students; // 静态成员：每创建一个对象，学生人数增1

std::cout << number\_students << std::endl;

}

Student::~Student()

{

std::cout << "destruct one student\n";

--number\_students;//每析构一个对象，学生人数减1

}

int Student:: number()

{

return number\_students;

}

Student s1;

Student s2;

Student::number();

s1.number(); // 这两种访问静态成员函数的方法都可行。

s2.number();

一个静态成员函数不与任何对象相联系，故不能对非静态成员进行默认访问，也最好不要对非静态成员进行访问。

静态成员函数与非静态成员函数的根本区别在于静态成员函数没有this指针，而非静态成员函数有一个指向当前对象的this指针

class Sc

{

public:

void nsfn(int a);//像声明Sc::nsfn(Sc\* this,int a);

static void sfn(int a);//无this指针

}

void f(Sc& s)

{

s.nsfn(10);//转换为Sc::nsfn(&s,10)

s.sfn(10);//转换为Sc::sfn(10)

}

例：程序class\_test5

// 类中的静态变量

#include <iostream>

using namespace std;

class P

{

public:

static void init()

{

// 静态成员函数只能调用静态成员函数和静态成员变量，原因是静态成员函数在类对象之前就已经存在代码区，

// 这个时间非静态成员还没有分配内存空间，非静态成员在类对象构造时才能分配内存空间。

cout << "静态成员函数" << endl;

}

void output()

{

init(); // 非静态成员函数可以调用静态成员函数和成员变量

x = 1;

y = 1;

cout << x << endl;

cout << y << endl;

cout <<" 非静态成员函数" << endl;

}

static double z;

private:

static int x;

static int y;

};

// 如果要访问静态成员变量时，静态成员变量必须进行初始化，

// 并且在类定义之外进行，如果不访问静态成员变量，静态成员变量不需要进行初始化

int P::x = 0;

int P::y = 0;

double P::z = 0.0;

int main(int argc, char\* argv[])

{

P p1;

p1.init();//可以这样访问静态成员函数

p1.output();

P::init();//也可以这样访问静态成员函数

P::z = 1.0;//z为公共静态成员变量，可以这样访问

return 0;

}

const static成员变量

例：程序class\_test6

class Account

{

public:

const static int number = 30; // 可以在类定义体中初始化，

double array[number];

};

static成员不是类对象的组成部分，所以可以有下面几种用法：

class Bar

{

public:

Bar(int number = i) // 静态变量作为形参的默认值

{

}

private:

static int i;

static Bar bar1; // ok

// Bar bar2; // error，使用正在定义的Bar类型

};

int Bar::i = 9; // 静态变量i必须初始化，因为在构造函数中其作为形参的默认值

Bar Bar::bar1;

友元：

允许一个类将其非公有成员的访问权授予指定的函数或类。

通常，将友元声明放在类的开头或结尾。

例：程序class\_test7

class A; // 前向声明，类名声明

class C

{

public:

void SetNumber(A& a); // 此处只能有函数声明，不能有函数的实现。

void output(A& a);

};

class A

{

friend class B; // 类B的所有成员函数均可访问类A的private成员。

friend void C::output(A& a); // 类C的成员函数output()可以访问类A的private成员，其余成员函数均不可访问类A的private成员。

private:

int number;

};

class B

{

public:

void SetNumber(A& a)

{

a.number = 5;

}

void output(A& a)

{

std::cout << a.number << std::endl;

}

};

void C::output(A& a)

{

std::cout << a.number << std::endl;

}

void C::SetNumber(A& a)

{

// a.number = 5; // error，无法访问A的private成员

}