C++程序设计

"<<"：插入运算符 / ">>"：提取运算符.

cout：标准输出流对象 / cin：标准输入流对象

```
cout << " a=" << a << endl;
cin >> a >> b;
using namespace std;    使用命名空间
```

```
#include <iosstream>
int main
{
    std::cout << "Enter two number:" << std:endl;
    int v1, v2;
    std::cin >> v1 >> v2;
    std::cout << "The sum of" << v1 << "and" << v2
              << "is" << v1+v2 << std::endl;
    return 0;
}
```

:: 域操作符/域运算符、作用域运算符

引用 `int &b = a;` b为a别名

字符串 string.

函数重载

函数名称必须相同.

参数列表必须不同（个数不同，类型不同，参数排列顺序不同）

返回类型可以相同/不同.

重载模板

```
template   <typename T>
void       my_swap_2 (T &a, T &b)
{          T  c;
           c = a;
           a = b;
           b = c;
}
```

默认参数

```
double volumn( double r=1, double h=1)
{   return  3.14*r*h*r; }
```

若未给出参数实参值，则默认值

可以 volumn(double r, double h=1)

不可以 volumn(double r=1, double h)

避免与函数重载混合/予值

new 和 delete

```
int *p = new int;          int *p = new int(10);      int *p = new int[10];
         申请                       申请并赋值                  申请十个空间,
delete p;                  delete p;                  delete []p;
                                   p=NULL;
```

类和对象

类是对象的抽象，对象是类的实例。

面向对象

抽象 封装 继承 多态

class 类名
{ private:
私有的数据和成员函数；
public:
公用的数据和成员函数；
}
未指定，默认私有

struct 类名
{
.
.
.
}
未指定，默认公用

支持在类内声明函数，在类外定义函数，但定义时需加作用域限定符
例 void Student::display() { }

内置函数不保留返回地址等处理，而是将函数嵌入程序调用点。
可大大减少调用函数时间开销。

CPP对类内定义的成员函数，可省略inline，已表示隐含指定为内置函数。

若在类外定义成员函数，系统不默认为内置函数，需用inline作显式声明。
或在成员函数定义时inline，或在成员函数声明时inline

类外定义的成员函数需将类的声明和成员函数的定义放在同一个文件中，
只有将类外定义的成员函数规模小且调用频率高，才指定为内置函数。

类和对象的使用

构造函数　用以处理对象的初始化

不需要用户调用，建立对象时自动执行，无类型，无返回值

构造函数只执行一次

```
Class time
{ public:
    time( );
}
time::time()
{ hour=0;
  minute=0;
  second=0;
}
```

```
class time
{ public:
    time(int, int, int);
};
类外定义
time::time(int h, int m, int s)
{ hour=h;
  minute=m;
  second=s;
}
```

调用时

```
int main()
{ time time1(12,25,30);
}
```

参数初始化表

```
time::time(int h,int m,int s):hour(h),minute(m),second(s){}
```

调用 time time1(12,10,5);

构造函数亦可定义为参数重载

无 Box box1(); 不为定义Box类的对象box1，而是定义返回类型为Box型的函

构造函数也可以设置默认参数.

析构函数.

用于在域或对象生命周期结束时,由自动执行析构函数,只能有一个析构函数

~类名()
{ }

对象数组

student::student(int=1001, int=18, int=60);
student stud[] = {1005, 60, 70}. 分别作为3个元素第一个实参
student stud[3] = {
student(~,~,~), student(~,~,~), student(~,~,~)}
分别调用构造函数初始化

对象指针

对象成员函数指针

Void (Time::*p2)(); //定义p2为指向Time类的指针变量
p2 = &Time::get_time;

this指针 指向当前对象的this指针

常对象 Time const t1(12,34,46);
常成员函数 void get_time() const; 表明get_time是一个const型函数,即常成员函数。
只能通过调用常对象调用其常成员函数,不能调用其普通成员函数
且常成员函数不能改变对象数据成员的值.
声明 mutable 常对象也可改变.

指向常对象的指针对象

const 类型名 *指针变量名;

Time t1(10,12,15);

const Time *p = &t1;

指向对象的常指针

类型名 *const 指针变量名;

对象的常引用

不希望在函数中修改t1的值

void fun(const Time &t);

对象的动态建立与释放

~~new box;~~

Box *pt;

Pt = new Box;

使用完成后 delete pt; pt=[];

对象的赋值

对象名1=对象名2;

复制构造函数

Box box2(box1); //将box1复制给box2

或 Box box2=box1;

静态数据成员 ∈类

静态数据成员可以初始化，只能在类体外初始化.

数据类型 :: 静态数据成员名=初值;

不能使用参数文初始化表对静态数据成员初始化

可以使用对象名引用静态数据成员，也可用类名引用

静态成员函数 ∈类

不能访问本类中非静态成员

只用静态成员函数引用静态数据成员，不引用非静态数据成员

友元

友元函数可以访问类中的私有成员，可以不是成员函数 一般不是成员函数

也可以是另一个类中成元函数

友元类

可将一个类(B)声明为类(A)的友元类. 单向

在类中(B)声明：

friend 类名(另一个A); 说明 A可以访问B.

类模板

template < class 虚拟名称(numtype)>

启用该模板构造类

用类模板定义对象

类模板名<实际类型名> 对象名(参数表);

该模板外定义成员函数

```
template <class numtype>
numtype Compare <numtype>::max()
    { return(x>y) ?x:y;}
```

template < class 虚拟类型参数>

函数类型 类模板名 < 虚拟类型参数 >:: 成员函数名(形参){}

若有多个类型参数

```
template < class T1, class T2>
class someclass { };
```

定义对象 someclass<int, float> obj;

继承与派生

派生类无法直接访问基类

派生类增加函数访问派生增加的成员

无法选择性继承成员或成员函数

不继承构造函数和析构函数