# 入门

## 数据类型

### 指针

指向类的数据成员指针：

指向类，所指数据成员必须是共有的，用途不大

指向类成员函数的指针：

类并不是对象，有时候可将其视为对象使用，可以声明并使用指向对象成员函数的指针。

例如：void(A::\*pafn)(void)：表示他指向无参数和无返回值的类A的成员。

使用：

pafn=A::fa;

A x;

A \*px=&x;

x.fa();

对象调用：(x.\*pafn)();

指针调用：(px->\*pafn)();

一般说明方法： type(A::\*pointer)(list);

pointer=A::fun;

函数名表示地址，在数组中，数组名表示首地址。

Test \*one{new Test(3),new Test(4);},

one[0]表示第一个地址，要把它当指针用。\*one[0]可表所指内容。

## 流

输入输出格式控制：

自动识别浮点数，并用最短格式输出，例如：

int a;double b;

cin>>a>>b; //23.23时...

单个字符：舍去空格,直到对字符 char a,b,c;cin>>a>>b>>c;

字符串：没有结束位，加"\0"，防止后面的乱码

char \*p=new char[5];

for(int i=0;i<4;i++) cin>>\*(p+i);

p[4]=‘\0’;cout<<p;

字符数组：用数组名整体读入 char a[30];cin>>a;

对于字符指针，尽量动态分配地址，也能逐个赋值。例如：

char \*a;

st=new char[strlen（n）+1];

## C到C++的演变

C++允许机构中定义函数

struct Point {

private:

int a,b;

public:

Point(){};

Point(int x,int y)

{a=x;b=y};

void show(){..};

};

1.struct成员函数默认public，类的成员都是private

2.struct定义了构造函数。

## C++语言特性

**变量**

变量初始化：int a(5) 等同a=5;

### const常量

const修饰符和预处理程序。C++建议用const，如

#define BUFSIZE 100。在置换过程很多隐藏错误，还有类型问题。BUFSIZE究竟整形还是浮点数？等效const int BUFSIZE=100;常量不能改变，必须初始化

### 函数

1.函数要有类型说明。否则void。函数要先声明或写在前面才能用。

int add(int ,int)不要参数名（函数原型）。声明只告诉编译器类型

2.参数传递：传值和传引用。可使用对象（传值，传的是备份，不影响原来的），对象指针，对象引用（C++推荐）

如：void swap(string,string)不改变原来的

3.返回值：

引用 int& max(int, int)

指针　int　＊　ｍａｘ（ｉｎｔ，ｉｎｔ）

对象 string max（int，int）

4.定义成员函数

除了在类中，也可在类外定义，但返回值，参数类表都要一样

int Pont::show(int a=1,int b=2)

{

x=a; y=b;

}

等价于int Point::show:x(1),y(2){..}

**this指针**

C++在类外定义成员函数可以访问类的私有成员。实际隐藏this指针

void Point::say(string str,(Point)\*this)

{

this->name=str;

}

说明：1.this指向调用它的对象

2.作用域运算符“：：”，之后内容表示属于这个类

构造函数

编译器给类默认生成，但自己定义了就不产生

**复制构造函数**

编译器默认产生的原型Point::Point(Point& t)；

若定义Point::Point(const Point& t) //推荐加const

{

x=t.x;y=t.y;

}

Point p1(1,2);

Point p2(p1); //就可以复制

**析构函数**

编译器默认产生，自动调用。形如：

~Point(){};

delete释放动态数组:delete [] p数组也要掉析构

说明：构造函数，复制构造函数，析构函数是类的基本成员

**内联函数**

加快程序执行速度，例如

inline int isnumber(char c){…..}

**友元函数**

两个类概念相近，一个类可以无限制访问另一个。

1.类的友元函数

class Point{

friend int distances(Point&, Point&);

}

int distances(Point& a, Point& b)

{

}

友元不是Point类的成员，没有this指针（this可访问private）。

作用域：friend声明到类的结束点

好处：按需增加接口，但破坏封装性。

2.成员函数做友元

2.一个类做另一个类的友元

多态性

函数重载

int max(int,int)

int max(int,int,int)

**new**

1.new用于建立生存期可控的动态对象，返回对象指针。

Point \* p=new Point;Point \* p2=new Point(1,2);

delete p;delete p2;

2.指针名 =new 结构名

3.动态变量

.double \*p;p=new double[3]；delete p;

继承

单一继承.多重继承

class Rectangel:public Point，private Oval{

void test(){show();}

}

继承方式 基类 派生类

public private 不可访问

protected protected

public public

private private和protected变成private，public变成protected

在派生类中可以void show(){Point::show}方式来访问

**二义性**

类Rectangel的基类Point，Oval都有show（）。c.show()产生二义性。可以

r.A::show()来区分，Point属于Rectangel的被支配者。

Rec若和基类重名则自动覆盖了。

**多态性**

**string对象**

要#include<string>

初始化：1.string str=”ab”或str[0]=’a’,str[1]=’b’同样cout<<str[0]来单个输出

2.string str(“how are you!”);

**对象**

实例化方式

Point a,b;

Point \*p=&a

Point &r=b; //定义引用

1.对象和引用使用“.”访问对象成员.指针用“->”,例如this

2.Point a; 程序自动调用构造器来分配内存。

3.区别于java，C++私有成员在类外就不能访问，即便是对象，但可以a=b这样赋值

Point a[2]对象数组。可以Point a[2]={ Point(),Point(1,2 ) }

**不完全类声明**

## 编译指令

编译指令在编译程序之前进行，分析处理C++的不是程序一部分。

作用：易读，易改，易移植和便于调试

### 文件包含（嵌入指令）

#include “c:\hello.h” ；“ ”一般自己定义的头文件。<xxx.h>系统的；C++不需要<.h>了。

说明：编译指令不是C++一部分，所以这里是“\”。若在程序中，则

char fname[]=”\\user\\test.h”

<cmath>代替<math.h>，#include <iostream> 标准库函数。但必须

using namespace std;如cout和cin<<x<<y;输入2 4<CR>2和4之间有空格

### 宏定义#define

#define MAX(a,b) ((a)>(b)? \

(a)(b) )

说明：结束不能加分号”;”如换行用“\”。不想影响其他部分用#undef删除

**条件编译。**

#if,#else,#elseif,#endif

形式:#if true/false

….

#else

…..#error(#error常在else中使编译器显示错误信息)

#endif

**defined操作符**

判断标识符是否被#define定义。若用于调试。下如

#define DEBUG //调试结束删除即可

void main()

{

#if defined(DEBUG)

…..

#endif

}

#ifdef,#infndef等同

#if!defined(HEAD\_H)---------------#ifndef HEAD\_H

#define HEAD\_H #define HEAD\_H

…….. ……

#endif #endif

编译指令一个占一行。一个头文件可能在多个文件中使用，造成变量和和类的重复定义。为了避免多个文件重复编译一个头文件，可在头文件Point.h中定义

#if!defined(POINT\_H)

#define PONT\_H

class Point{

…..

}

#endif

**编译器：**编译成.obj，链接程序（Link）处理成.exe

test.cpp----test.obj---test.exe

## 书写规范

1.C++以.cpp扩展名；/\* \*/或//注释；endl和“\n”都是换行；

2.类声明放在头文件中。较大的类采用一个头文件一个实现文件

3.

//Point.h

#if!defined(POINT\_H)

#define PONT\_H

class Point

{

private:

int x,y;

public:

Point();

Point(int ,int);

~Point(){…….};

}

#endif

//main\_test.c

#include <Point.h>

void main()

{

}

## 语法错误

class Point

{

int x=2;(类体内，只有对象才有意义)

}

void main()

{

cout<<Point.x; //x是类的私有成员。可使用public的函数Point.show();

}

2.显示调用构造器：Point a.Point(1,2);

3.Point d,\*p;p=&d;p.d->x;p只能指向对象而不是对象成员

4. 不能直接给类数据成员赋值,因为类对象不能直接访问类私有成员。区别java

## 实例化

1. **class** Coordinate
2. {
3. **public**:
4. **int** m\_iX;
5. **int** m\_iY;
6. };
7. **void** main(){
8. Coordinate coord[3]; //栈上
9. coord[1].m\_iX=10;
10. Coordinate \*p=**new** Coordinate[3]; //堆中
11. p[0].m\_iX=10;
12. p->m\_iX=10;
13. delete []p;
14. p=NULL;
15. }