面向对象程序设计

Object Oriented Programming

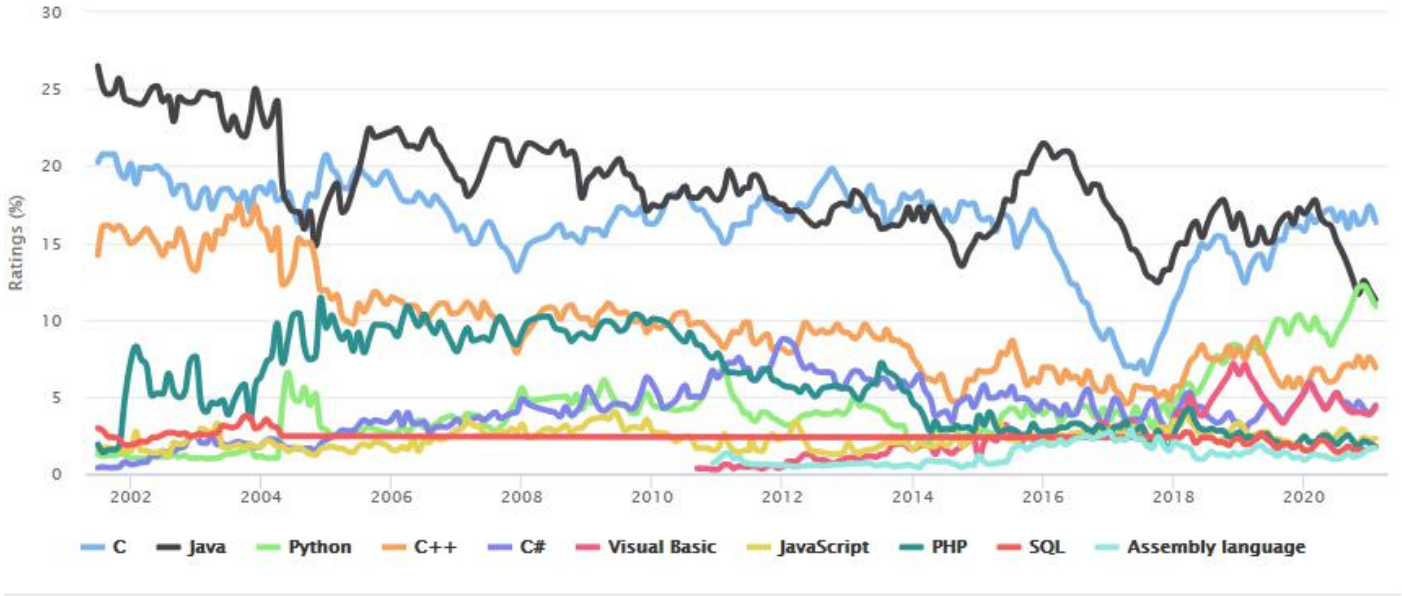
主 讲 教 师：陈洪刚
开 课 时 间：2021年
honggang_chen@yeah. net

CHAPTER 1

CONTENTS

1.1 从C到C++

- C语言是结构化和模块化的语言，最初是为了编写UNIX操作系统而设计的，经过多次改进，成为最流行的编程语言之一。



https://www.tiobe.com/tiobe-index/

1.1 从C到C++

C语言是一种结构化的程序设计语言，语言本身简洁、使用灵活方便。
既适用于设计和编写大的系统程序，又适用于编写小的控制程序，也适用科学计算。

程序的可移植性好。C语言在某一种型号的计算机上开发的程序，基本上可以不作修改，直接移植到其它型号和不同档次的计算机上运行。

C语言特点

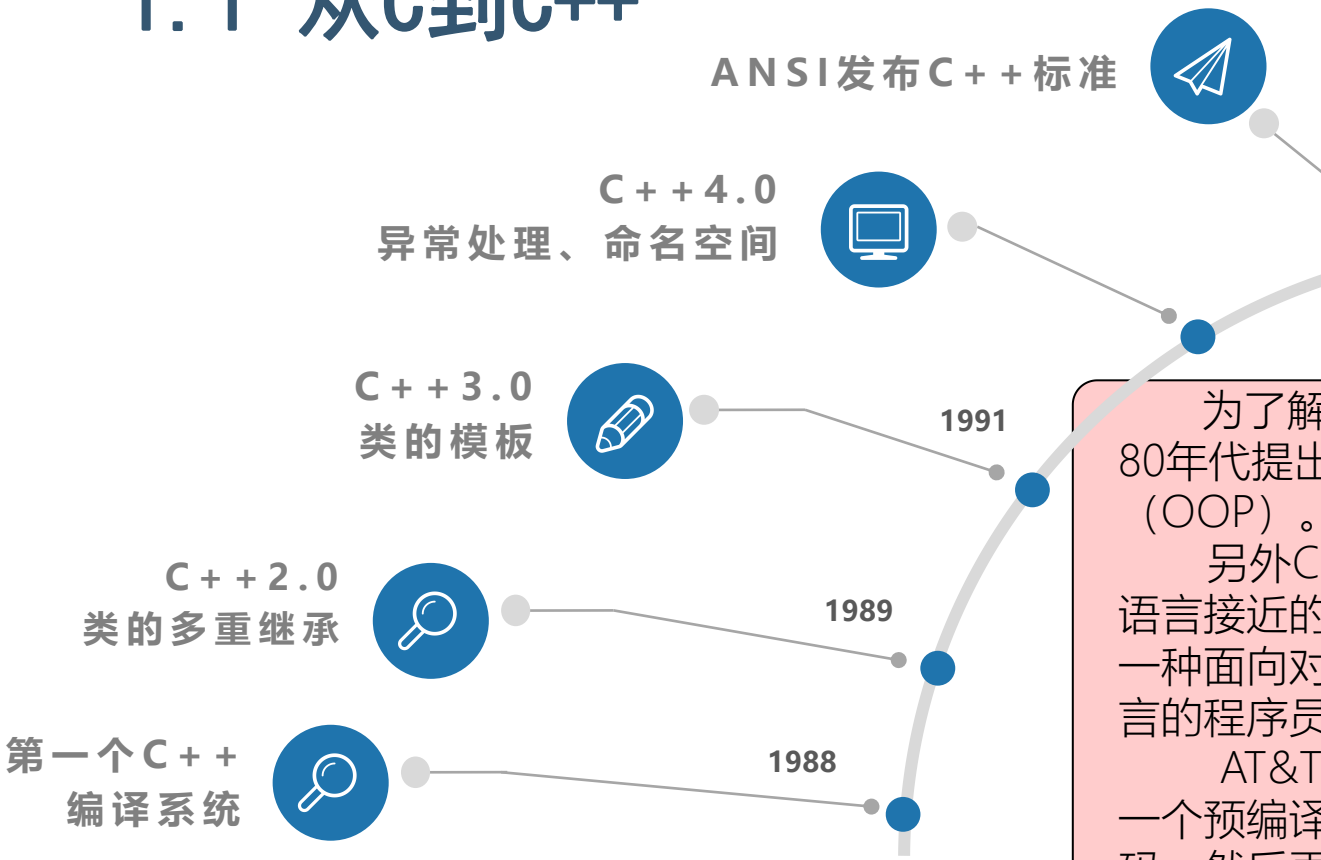
既有高级语言的特点，又具有汇编语言的特点。运算符丰富，除了提供对数据的算术逻辑运算外，还提供了二进制的位运算。并且也提供了灵活的数据结构。

用C语言编写的程序表述灵活方便，功能强大。用C语言开发的程序，其结构性好，目标程序质量高，程序执行效率高。

程序的语法结构不够严密，程序设计的自由度大。这对于比较精通C语言的程序设计者来说，可以设计出高质量的非常通用的程序。
但对于初学者来说，要能比较熟练运用C语言来编写程序，并不是一件容易的事情。与其它高级语言相比而言，调试程序比较困难。往往是编好程序输入计算机后，编译时容易通过，而在执行时还会出错。但只要对C语言的语法规则真正领会，编写程序及调试程序还是比较容易掌握的。

随着C语言应用的推广，C语言存在的一些缺陷或不足也开始流露出来，并受到大家的关注。
如：C语言对数据类型检查的机制比较弱；缺少支持代码重用的结构；随着软件工程规模的扩大，难以适应开发特大型的程序等等。

1.1 从C到C++

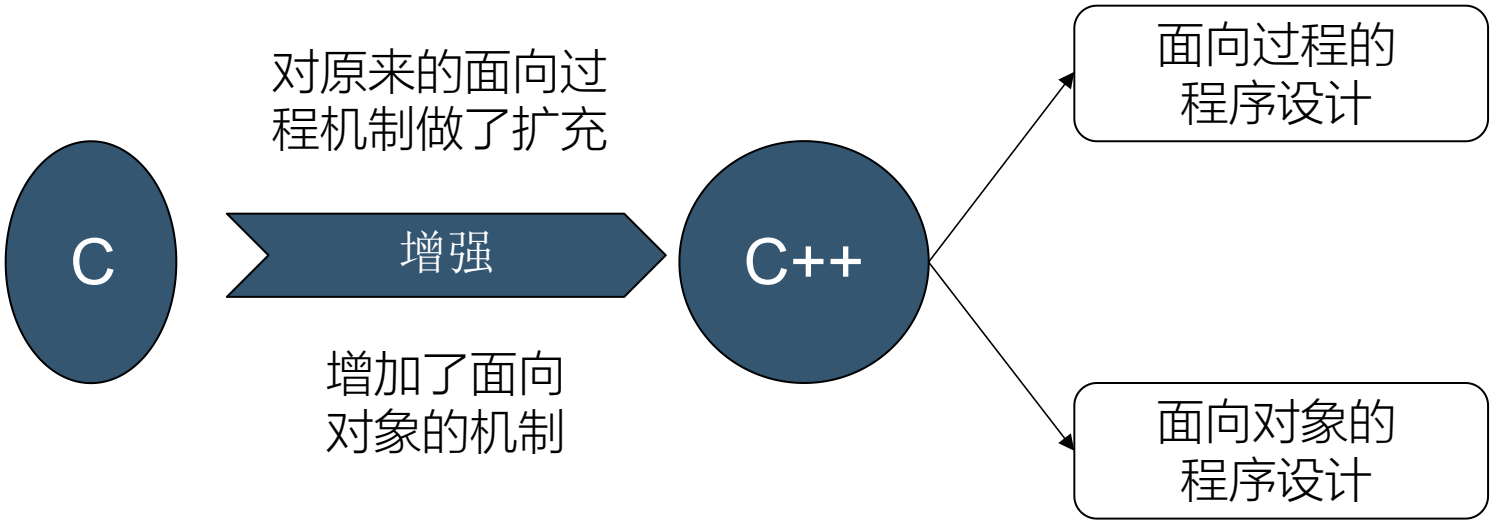


为了解决软件设计危机，在20世纪80年代提出了面向对象的程序设计思想（OOP）。

另外C语言又有简洁、高效，与汇编语言接近的特点，如果在它的基础上开发一种面向对象的语言，很容易受熟悉C语言的程序员欢迎，于是产生了C++。

AT&T发布的第一个C++编译系统是一个预编译器，它把C++代码转换成C代码，然后再用C编译系统生成目标代码。

1.1 从C到C++



1.2 最简单的C++程序

例1.1　输出一行字符“This is a C++ program.”。

```
#include <iostream>        //   用cout输出时需要用此头文件
using namespace std;     //   使用命名空间std

int main()
{
    cout<<"This is a C++ program.\n";   //     上面用C++的方法输出一行
    return 0;
}
```

1.2 最简单的C++程序

- 标准C++规定main函数必须声明为int类型，如果程序正常运行，向操作系统返回一个零值，否则返回非零值，通常是-1。
- C++程序中可以用/*...*/做注释，可以用//做注释。前者可以做多行注释，后者只做单行注释。
- C++程序中常用cout、cin进行输出输入：cout是C++定义的输出流对象，<<是流插入运算符；cin是输入流对象，>>是流提取运算符。
- 使用cout、cin需要用头文件iostream，在程序开始要用#include声明包含的头文件。C++标准中，系统提供的头文件不带.h。
- using namespace std; 意思是使用命名空间。C++标准库中的类和函数是在命名空间std中声明的，因此程序中如用C++标准库中的有关内容（此时需要用#include命令行），就要用using namespace std; 语句声明。

1.2 最简单的C++程序

例1.2 求a和b 两个数之和

```
// 求两数之和
#include <iostream>        // 预处理命令
using namespace std;    // 使用命名空间std
int main()                    // 主函数首部
{                                // 函数体开始
  int a,b,sum;                // 定义变量
  cin>>a>>b;                  // 输入语句
  sum=a+b;                    // 赋值语句
  cout<<"a+b="<<sum<<endl;        // 输出语句
  return 0;    // 如程序正常结束，返回一个零值
}
```

- cin 是C++定义的输入流对象。">>"是输入运算符，与cin配合，其作用是从输入设备中提取数据送到输入流cin中。在程序执行时，键盘输入的第一个数据赋予a，第二个数据赋予b 。
- cout语句中的endl（end line）是C++控制符常数，作用是让光标换行。
- 如果在本程序运行时输入：123 456 <回车> ，输出?

1.2 最简单的C++程序

例1.3 求两个数中的大数

```
#include <iostream>
using namespace std;
int main()
{ int max(int x,int y) ;          //对max函数作声明
  int a,b,c;
  cin>>a>>b;
  c=max(a,b);                     //调用max函数
  cout<<"max="<<c<<endl;
  return 0; }

int max(int x,int y)              //定义max函数
{ int z;
  if(x>y) z=x;
  else z=y;
  return(z); }
```

- 本程序包含两个函数，主函数main和被调用的函数max。max函数的作用是将两个整数中的大数赋予变量z。return语句将z的值返回给主函数main。返回值是通过函数名max带回到main函数的调用处。
- 程序运行情况：
  18　25　　(输入18和25)，输出?
- 如何改写程序可以不用声明?

1.2 最简单的C++程序

➢ 怎么用类来编写程序?

例1.3 求两个数中的大数

```
#include <iostream>
using namespace std;
int main()
{ int max(int x,int y) ;          //对max函数作声明
  int a,b,c;
  cin>>a>>b;
  c=max(a,b);                     //调用max函数
  cout<<"max="<<c<<endl;
  return 0; }

int max(int x,int y)              //定义max函数
{ int z;
  if(x>y) z=x;
  else z=y;
  return(z); }
```

例1.3 求两个数中的大数

```
#include <iostream>
using namespace std;
int max(int x,int y)              //定义max函数
{ int z;
  if(x>y) z=x;
  else z=y;
  return(z);
}
int main()
{ int a,b,c;
  cin>>a>>b;
  c=max(a,b);                     //调用max函数
  cout<<"max="<<c<<endl;
  return 0;
}
```

1.2 最简单的C++程序

- 类有什么特点?

使用类改写前面的程序

```
#include <iostream>        //  预处理命令
using namespace std;     //  使用命名空间std

class AB                          // class声明一个类，类名AB
{ private:                         //   以下为类中的私有部分
    int a, b;                      // 私有数据成员a, b
  public:                          //   以下为类中公用部分
    void setdata()                 //   定义公有成员函数setdata
        {  cin>>a>>b;  }           //   输入a, b的值
    void displaysum()              //   定义公有成员函数display
        { cout << "a+b=" << a + b << endl; }
         // 输出a, b相加结果
    int max()                      //   定义公有成员函数max
        { return a > b ? a : b; } // 返回a, b的最大值
};
```

```
int main()                    //  主函数首部
{                             //  函数体开始
 AB ab;
      //  定义类的变量，叫对象
 ab.setdata();
      //  调用对象ab的setdata函数
 ab.displaysum();
      //  调用对象ab的displaysum函数
 cout<<"max="<<ab.max()<< endl;
      // 显示最大值
 return 0;
      //  程序正常结束，返回一个零值
}
```

1.2 最简单的C++程序

- 类有什么特点?

例1.4 包含类的C++程序

```
#include <iostream>
using namespace std;
class Student              //  声明一个类，类名为Student
{  private:                //  以下为类中的私有部分
     int num;              //  私有变量num
     int score;            //  私有变量score
   public:                 //  以下为类中公用部分
   void setdata()          //  定义公用函数setdata
    {cin>>num;             //  输入num的值
     cin>>score; }         //  输入score的值
   void display()          //  定义公用函数display
    {cout<<"num="<<num<<endl;       //  输出num
   cout<<"score="<<score<<endl; };  //  输出score
};                         //  类的声明结束
```

```
Student stud1,stud2;
  //定义stud1和stud2为Student类的变量，称为对象

int main()                 // 主函数首部
{stud1.setdata();
  // 调用对象stud1的setdata函数
 stud2.setdata();
  // 调用对象stud2的setdata函数
 stud1.display();
  // 调用对象stud1的display函数
 stud2.display();
  // 调用对象stud1的display函数
 return 0;
}
```

1.2 最简单的C++程序

- 程序中声明一个被称为类的类型Student。声明时要用关键字class。C++类中可以包含两种成员即数据(如变量num、 score)和函数(如setdata函数和display函数)。分别称为数据成员和成员函数。

- 在C++ 中将一组数据和访问这组数据的函数封装在一起，组成类。一个类是由一组数据，一组对其访问的若干函数，以及数据和函数的访问属性组成的。在前面程序中看到的private （私有） 、public（公有）保留字代表数据和函数的访问属性。

- 凡是指定为公有的数据和函数，既可由本类的函数访问和调用，也可由其他函数或语句访问和调用；凡是指定为私有的数据和函数，通常只能由本类的函数访问和调用。

- 程序中"Student stud1,stud2; "是一个定义语句，定义两个Student 类型变量stud1和stud2， Student 类与int一样是C++的合法类型。

1.2 最简单的C++程序

- 具有类类型的变量称为对象。 Student的对象stud1，stud2具有同样的结构和特征。

- 在类外调用成员函数时必须在函数名前冠以类的名称。

表 1.1　引用对象的成员

对象名	num(学号)	score(成绩)	setdata 函数	display 函数
stud1	stud1. num(如 1001)	stud1. score(如 98. 5)	stud1. setdata( )	stud1. display( )
stud2	stud2. num(如 1002)	stud2. score(如 76. 5)	stud2. setdata( )	stud2. display( )

- 主函数中第一条语句输入学生1的学号和成绩，第二条语句输入学生2的学号和成绩，第三条语句输出学生1的学号和成绩，第四条语句输出学生2的学号和成绩。
- 程序运行情况：1001 98.5 <回车>，输出?

1.3 C++对C的扩充

1.3.1 C++的输入输出

- C的输入输出
  printf和scanf　头文件：stdio.h

- C++的输入输出
  cin和cout　　头文件：iostream

表1.2　C++预定义的标准流

流名	含义	隐含设备	流名	含义	隐含设备
cin	标准输入	键盘	cerr	标准出错输出	屏幕
cout	标准输出	屏幕	clog	cerr 的缓冲形式	屏幕

1.3.1 C++的输入输出-cout输出

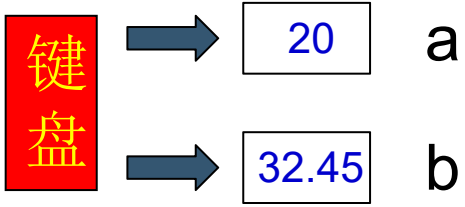
- 格式： cout <<表达式1[<<表达式2……]
- 功能：由左向右逐个计算表达式的值，将其插入到输出流cout中。
- cout 与输出运算符<<一起使用，每个<<后跟一个表达式，<<运算符的结合方向是从左向右，所以各个表达式的值按从左到右的顺序插入到输出流中。

```
for  (int i = 1; i < =3; i++)
     cout << "count=" << i << endl;
```

- endl是C++输出流的常数，在头文件iostream中定义，代表让光标换行。在C++中也可以用"\n"控制光标换行。所以输出语句也可写成：
- cout << "count=" << i << "\n";
- 注意不能写成：cout << "count", i, "\n";

1.3.1 C++的输入输出-cin输入

- 格式：cin>>变量1 [>>变量2 ……]
- >>是C++的输入运算符，表示从标准输入设备取得数据，赋予其后的变量。
- 从键盘输入数值数据时，两个数据之间用空格分隔或用回车分隔。
- 注意输入之间不能用逗号、分号等
- int a; float b; cin>>a>>b;
- 从键盘输入：20 32.45 <回车> 或
  20 <回车> 32.45 <回车>



1.3.1 C++的输入输出-cin和cout使用

例1.5 cin和cout的使用

```
#include <iostream>
using namespace std;
int main( )
{
 cout<<"please enter your name and age:"<<endl;
 char name[10];
 int age;
 cin>>name;
 cin>>age;
 cout<<"your name is "<<name<<endl;
 cout<<"your age is "<< age<<endl;
 return 0;
}
```

- 输出格式：
  cout <<表达式1[<<表达式2……]
- 功能：由左向右逐个计算表达式的值，将其插入到输出流cout中。
- 输入格式：
  cin>>变量1 [>>变量2 ……]
- 功能：从标准输入设备取得数据，赋予其后的变量。

1.3.1 C++的输入输出-文件

- 文件输入流类 ifstream
- 文件输出流类 ofstream
- 定义在fstream或fstream.h中
- ifstream infile(“input.txt”);
  //输入方式打开
- ofstream outfile(“output.txt”);
  //输出方式打开
- 如果input.txt的内容是10，
  那么output.txt的内容?

文件输入输出

```
#include <fstream>
using namespace std;
int main()
{
  ifstream infile("input.txt");
  ofstream outfile("output.txt");
  int inch = 0;
  infile >> inch;
  outfile << inch << "inches=" << inch*2.54 << "cm";
  return 0;
}
```

1.3.2 const说明符

- const用途：　　说明符号常量；
  const说明指针变量；
  说明函数的参数和返回值等

- C语言用#define定义符号常量：　#define PI 3.14159
  PI不是变量，它没有类型，没有存储空间。只是在程序编译时进行字符置换。

- C++提供一个更灵活的方式　const float PI = 3.14159
  它有类型，有存储空间,而且可以有指针指向这个值，但是不能被修改。

- 注意：const float PI = 3.14159;　float const PI = 3.14159; // 等价
  当定义一个const常量时，必须立即给它赋值, 否则产生编译错误。
  const int i;　　//错误　　const int i = 10; //正确

1.3.2 const说明符–说明指针变量

- **指向常量的指针**
- const出现在*的前面，是使指针指向的对象是常量，而不是指针为常量。
- 格式： const int *p;　　int const *p;
- 指向常量的指针不一定指向真正的常量，只是从指针角度来看，它指向的对象是常量，通过该指针不能修改对象。
- 无约束的指针可以赋值给指向常量的指针，但是常量的地址不能赋值给无约束的指针。

```
int a = 0, b = 1;
const int c = 3;
const int *p;  // 定义指向常量的指针
p = &a;    // 正确，首先使p指向a
*p = 10;    // 错误，不能修改指向的对象
p = &b;   // 正确，使p指向b
b = 20;    // 正确
```

```
const char *pc = "atst";   //指向数组
pc[3]='a'; //错误，不能修改指向的对象
pc = "ghik"; //正确，指向新的常量
```

```
int a = 100;
int *const p = &a; //正确
char const cc = 'c';
char *pc = &cc;    //错误
char const *pc = &cc;    //正确
```

1.3.2 const说明符–说明指针变量

- 常指针
- const出现在*后面，使指针成为常量。
- 格式：int *const p;

```
int a = 0, b = 1;
int *const p = &a;
*p = 10;
    //正确，只是指针为常量

p = &b;
    //错误，p本身为常量，地址不能改变
```

- 指向常量的常指针
- 要使两个目标都为常量，必须二者都声明为const。
- 格式：int const * const p;
  const int * const p;

```
int a = 0, b = 1;
const int *const p = &a;
*p = 10;
      //错误，不能修改指针指向的对象
a = 10;
p = &b;
      //错误，不能指向其他对象
```

1.3.2 const说明符–说明参数和返回值

- 传递const值
- 表明函数的参数不能在函数内修改，只能在函数内使用。

```
void f(const int x)
{
        x++; //会产生编译错误
}
```

- 传递const地址
- 如果传递或者返回一个指针，用户可以通过指针改变指针所指向的对象。
- 为了防止这种事情的发生，我们可以使传递的指针成为指向常量的指针。

```
 void f(const int *p)
{
  *p = 2;        //错误
  int i = *p;   //正确
  int *p2 = p;//错误
}
```

1.3.3 函数原型声明

- C++规定，如果函数调用在函数定义之前，要求在调用之前声明该函数的原型。
- 格式：函数类型 函数名（形参表）；形参表中形参可以只写类型。例：

  int max (int first , int second);

  或 int max(int , int);
- 其实C++编译时，只检查参数的数据类型。虽然名字first和second对编译器没有意义，但如果取名恰当的话，这些名字有着指示参数用途的作用，可以帮助程序员正确掌握函数的使用方法。

例1.3 求两个数中的大数

```
#include <iostream>
using namespace std;
int main()
{ int max(int x,int y) ;
  int a,b,c;
  cin>>a>>b;
  c=max(a,b);
cout<<"max="<<c<<endl;
  return 0; }

int max(int x,int y)
{ int z;
  if(x>y) z=x;
  else z=y;
  return(z); }
```

1.3.3 函数原型声明

- 函数原型作用：
  (1) 确定函数的返回类型，使编译器产生函数返回数据类型的正确代码
  (2) 确定了函数使用的参数类型和个数

- 在C++中 函数原型和函数定义在返回类型、函数名、参数表上必须完全一致。

- 函数原型使得C++能够提供更强的类型检查，允许编译器对如下的情况提示出错信息：
  1) 被调用函数的参数类型定义和类型转换非法
  2) 函数参数个数不相符
  3) 函数不能返回预定的数据类型全一致

- 重要作用：使得函数重载称为可能！

如:函数原型(声明)是
```
void f(int a, char *pChar);

调用时
int c = 6; int b = 10; int *pInt = &b;
f(c, pInt); //错误, 类型转换非法, int*到char*
f(c, b);    //错误, 类型转换非法, int到char*
```

1.3.4 函数重载

- 很多编程语言要求我们为函数设定唯一的标识符。例：要实现打印三种不同的数据类型的功能，在C语言中通常不得不用三个不同的函数名：Print_int(int a)；Print_char(char a)；Print_float(float a);
- 缺点：增加编程工作量，不方便阅读程序
- 解决方法：函数重载 —— 一词多意
- 使名字方便使用，是程序设计语言的一个重要特征。
- C++允许在同一个域中用一个函数名定义多个函数，这些函数的参数个数、参数类型不相同。
- 用一个函数名实现不同的功能，就是函数的重载，称为重载函数。

1.3.4 函数重载-函数签名

- C++要求重载的函数具有不同的签名。
- 函数签名包括：
  函数名；
  参数的个数、数据类型和顺序。
- 为保证函数的唯一性，函数必须拥有独一无二的签名。
- 返回值类型不是函数签名的一部分。
- 右侧程序输出？如果是print((double)x); print((int)y); 呢?

重载函数

```
#include <iostream>
using namespace std;
void print(int a);
void print(double a);
int main ()
{
  int x = 8;
  double y = 8;
  print (x);
  print (y);
  return 0;
}
```

```
void print (int a)
{
  cout << a << '\n';
}


void print (double a)
{
  cout << showpoint
       << a << '\n';
}
```

1.3.4 函数重载-函数签名

- 能不能用返回值实现重载，例如
  void f( );
  int f( );

- 答案是否定的。

- 因为我们调用函数时可能忽视它的返回值。这样编译器如何知道调用那个函数呢？而且程序员又怎么知道调用那个函数呢？

- 不允许函数参数个数、参数类型都相同，只是函数返回值不同。因为系统无法从调用形式上判断调用与哪个函数相匹配。

重载函数——做一做

例1：	void f();	void g();
例2：	void f();	void f(int);
例3：	void p(double);	void p(unsigned);
例4：	void m(double, int);	void m(int, double);
例5：	int s(int);	double s(int);

1.3.4 函数重载-例子

例1.6 用函数重载设计参数个数不同的函数，用一个函数名求两个整数或三个整数中的最大数。

```
#include <iostream>
using namespace std;

int max(int a,int b,int c)     //求3个整数中的最大者
{if (b>a) a=b;
 if (c>a) a=c;
 return a;
}
int max(int a, int b)          //求两个整数中的最大者
{if (a>b) return a;
 else return b;
}
int main( )
{
 int a=7,b=-4,c=9;
 cout<<max(a,b,c)<<endl;
   //输出3个整数中的最大者
 cout<<max(a,b)<<endl;
   //输出两个整数中的最大者
 return 0;
}
```

1.3.4 函数重载-例子

- 这个代码有什么问题?

例1.7 设计程序计算三个数中的大数

```
#include <iostream>
using namespace std;
//求3个整数中的最大者
int max(int a,int b,int c)
{ if (b>a) a=b;
if (c>a) a=c;
return a;
}
//求3个实数中的最大者
float max(float a,float b, float c)
{if (b>a) a=b;
 if (c>a) a=c;
 return a;
}
 //求3个长整数中的最大者
long max(long a,long b,long c)
{if (b>a) a=b;
 if (c>a) a=c;
 return a;
}
int main( )
{int a,b,c;  float d,e,f;  long g,h,i;
cin>>a>>b>>c;  cin>>d>>e>>f; cin>>g>>h>>i;
int m; m= max(a,b,c);                    //函数值为整型
cout <<"max_i="<<m<<endl;
float n; n=max(d,e,f);                   //函数值为实型
cout<<"max_f="<<n<<endl;
long p; p=max(g,h,i);                    //函数值为长整型
cout<<"max_l="<<p<<endl;
return 0;}
```

1.3.5 函数模板

- 如果两个函数的参数个数相同，函数的行为相同（做同样的事），只是函数和参数的数据类型不同，用函数重载实现的话，编写的函数代码是相同的。为了节省时间，C++提供了函数模板功能。

- 格式：template <typename 标识符[, typename 标识符, … …]>
  函数定义（函数的类型和参数的类型用声明的标识符表示）

- template 是关键字，含义是模板；typename 是关键字，表示其后的标识符代表类型参数，调用时根据实参的类型确定形参的类型。

- 所谓函数模板，是建立一个通用函数，不指定函数类型和参数类型，而用一个虚拟的类型表示。在调用函数时，用实参的类型取代模板中的虚拟类型。

- **函数模板**是模板的定义，定义中用到通用类型参数。**模板函数**是实际的函数，由函数模板实例化得到，由编译系统在碰见具体的函数调用时生成。

1.3.5 函数模板-例子

例1.8 设计程序计算三个数中的大数-函数模板

```
#include <iostream>
using namespace std;

template <typename T>
T max(T a,T b,T c)  //用虚拟类型T表示类型
{if(b>a) a=b;
 if(c>a) a=c;
 return a;
}
int main()
{int i1=8,i2=5,i3=6,i;
 double d1=56.9,d2=90.765,d3=43.1,d;
 long g1=67843,g2=-456,g3=78123,g;
 i=max(i1,i2,i3);
 d=max(d1,d2,d3);
 g=max(g1,g2,g3);
 cout<<"i_max="<<i<<endl;
 cout<<"d_max="<<d<<endl;
cout<<"g_max="<<g<<endl;
 return 0;
}
```

1.3.5 函数模板-例子

- 当编译程序发现max(i1,i2,i3)调用时（i=max(i1,i2,i3);），生成如下函数：

```
int max(int a, int b, int c) {
    if(b>a) a=b; if(c>a) a=c; return a;
 }
```

- 同样，当发现max(d1,d2,d3)时（d=max(d1,d2,d3);），也生成如下函数：

```
double max(double a, double b, double c) {
    if(b>a) a=b; if(c>a) a=c; return a;
 }
```

- 从程序中看到，此问题用函数模板比用函数重载更方便。模板保证了数据类型的一致，又避免了相同操作的重载函数定义。

- 注意，函数模板只适用于函数参数的个数相同而类型不同，并且函数体相同的情况，如果函数的参数个数不同，则不能用函数模板。

1.3.6 有默认参数的函数

- 如果我们经常用相同的参数调用同一函数，怎么做?
- 如果当函数有长长的参数列表，而且大多数参数在调用时都一样，怎么做?

- 例如：我们去打球。(在那儿打球呢？)

- 解决方法：用默认(缺省)参数
- C++允许为函数的参数设置默认值，这时调用函数时，如果没有实参，就以默认值作为实参值。

- 格式：形参类型 形参变量名 = 常数　　如 int f(int a, int b, int c=100)
- 功能：调用函数时，如果没有实参，就以常数作为该形参的值;如果有实参，仍以实参的值作为该形参的值。

- 注意：有默认值的形参必须放在形参表的右边，不允许无默认参数值和有默认参数值的形参交错排列。

1. 3. 6 有默认参数的函数

- 例：编写计算圆柱体体积函数　float　volume (float h, float r = 12.5)
- 调用可以采用以下任何一种形式：volume(45.6)；　或　volume(32.5, 10.5);
- 函数参数结合从左到右。用第一种方式调用时，只有一个实参，圆半径的值取默认值12.5；用第二种方式调用时，有两个实参，圆半径的值取实参的值10.5。
- 如果用函数原型声明，只要在函数原型声明中定义形参的默认值即可。
- 谨慎：将一个函数名同时用于重载函数和带默认形参值的函数。当调用函数时，如少写一个参数，系统可能无法判断是利用重载函数还是利用带默认参数值的函数，导致出现二义性。
- 例如将例1.6中的第三行改为 int max (int a, int b, int c = 100); 此时 max是重载函数，又带默认参数值，如果出现max(5, 23)形式的调用，编译系统无法断定调用哪个函数，于是发出编译出错的信息。

1.3.7 变量的引用

- C++提供了为变量取别名的功能，这就是变量的引用。
- 格式： 类型 &变量1 = 变量2 如 int & a =b;
- 变量2是在此之前已经定义过的变量，且与变量1的类型相同。这里为变量2定义一个别名变量1，在程序里变量1和变量2 就是同一个变量。
- 注意：两个变量不能用同一个别名。

引用

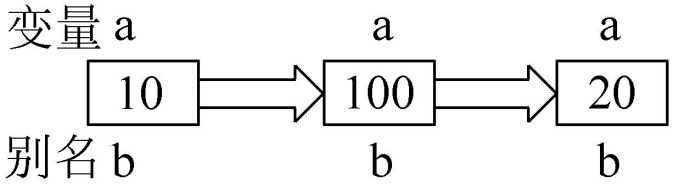
```
// 两个变量不能用同一个别名
int a = 3 ,b =4;
int &c = a;  //  c是a 的别名
int &c = b;  //  错误的用法


// 一个变量可以有多个别名
int a = 3;
int & b= a; //  b是a 的别名
int & c= a; //  c是a 的别名
```

1.3.7 变量的引用

```
#include <iostream>
using namespace std;

int main( )
{
 int a=10;
 int &b=a;   //声明b是a的引用
 a=a*a;      //a的值变化了，b的值也应一起变化
 cout<<a<<"  "<<b<<endl;
 b=b/5;     //b的值变化了，a的值也应一起变化
 cout<<b<<"  "<<a<<endl;
 return 0;
}
```



程序运行结果如下：

100　　100

20　　20

1.3.7 变量的引用-与指针的区别

- 引用本身不是数据类型，而指针是数据类型。
- 引用必须进行初始化，而指针不是必须的。
- 引用一旦初始化，就维系在一定的目标上，再不分开；不能再将引用维系到另一个目标上。而指针可以随时改变指向的目标。

```
int a;
int& ra = a;
int& *p = &ra; // 错误，企图定义一个引用的指针

int* pInt = &ra;
int* & p = pInt; //正确， 对指针建立引用
```

```
int *pInt; //正确
int &rInt; //错误
int a;
int &rInt = a;
```

```
int a = 3; int b = 10;
int &ra = a;
ra = b; //把b的值赋值给(a, ra), 不是把b维系在ra上
int* pInt = &a; //pInt 指向a
pInt = &b; //pInt 指向b
```

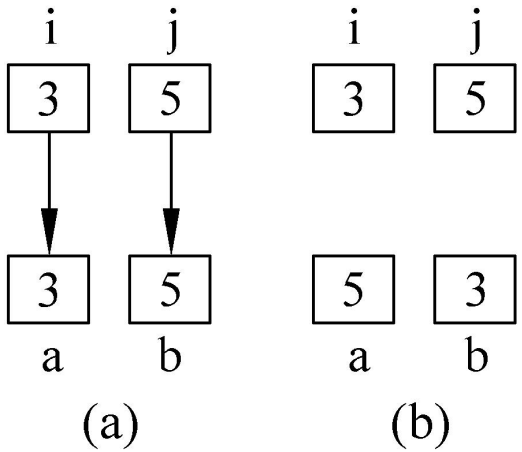
1.3.7 变量的引用–作为函数参数

- C++除了可以用普通变量、指针变量做形参外，还可以用引用变量做形参。

- **用普通变量做形参**：这时传递的是实参的值，在函数内形参与实参是两个不同的内存单元，对形参的修改不会影响实参的值。

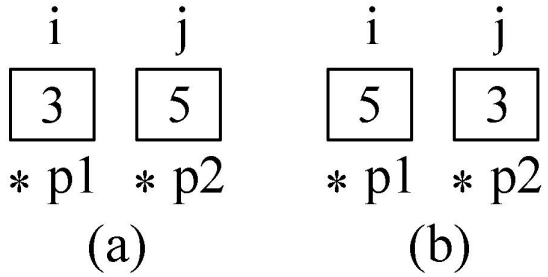
例1.10 无法实现两个变量的值互换的程序

```
#include <iostream>
using namespace std;
void swap(int a,int b)
{ int temp;
 temp=a;
 a=b;
 b=temp;     //  实现a和b的值互换
}
int main( )
{int i=3,j=5;
 swap(i,j);
 cout<<i<<","<<j<<endl;
//  i和j的值未互换
 return 0;
}
```



1.3.7 变量的引用–作为函数参数

- **用指针变量做形参**：这时传递的是实参变量的地址（指针），在函数内利用这个指针访问实参变量。



例1.11 用指针变量做形参，实现两个变量值的交换。

```
#include <iostream>
using namespace std;
void swap(int *p1,int *p2)
{int temp;
 temp=*p1;
 *p1= *p2;
 *p2=temp;
}
int main( )
{int i=3,j=5;
 swap(&i,&j);
 cout<<i<<","<<j<<endl;
//  i和j的值互换成功
 return 0;
}
```

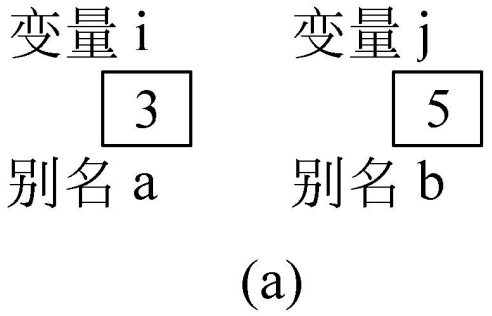
- 在函数体内需要使用(*p)，使程序难以阅读。
- swap(&a, &b)的形式会造成交换两个地址的错觉。
- 传递指针给使用函数的用户增添了编程和理解上的负担。
- 解决方法：使用引用传递参数

1.3.7 变量的引用–作为函数参数

- **用引用变量做形参**：引用变量是变量的别名，在调用函数时，用引用变量做的形参就成了实参变量的别名，在函数中用的形参名就是实参的别名，这样比用指针变量更直观、更方便。

例1.12利用引用变量实现两个变量值的交换

```
#include <iostream>
using namespace std;
void swap(int &a,int &b)
{int temp;
 temp=a;
 a=b;
 b=temp;
}
int main( )
{int i=3,j=5;
 swap(i,j);
 cout<<"i="<<i<<"
"<<"j="<<j<<endl;
//  i和j的值互换成功
 return 0;
}
```



1.3.7 变量的引用-进一步说明

- 引用变量都具有非void类型 void &a = 10; //错误
- 不能建立引用的数组 char c[6]="hello"; char &rc[6] = c; //错误
- 可以建立常引用变量，不允许修改常引用变量的值

  例： int i ;

  const int &a = i;

  a = 3; // 错误的用法

  i = 8; // i 不是常变量，可以修改

- 可以建立指针变量的引用变量

  例：int i;

  int *p = & i;

  int * &pt = p; // pt是p的别名变量，同时也是指针变量

1.3.8 内置/内联函数

- C++ 提供了一种机制，在编译时，将所调用的函数的代码嵌入到调用函数代码中，在执行函数时省去了调用环节，提高了函数的执行速度。这种机制称为内置函数，有的书称内联函数。
- 格式：inline 函数类型 函数名(形参表) { 函数体 }
- 调用格式： 函数名（实参表）
- inline 是C++的关键字，在编译时，编译程序会把这个函数嵌入到调用函数的函数体中。
- 使用内置函数可以节省程序的运行时间，但增加了目标程序的长度。所以在使用时要衡量时间和空间的得失。

1.3.8 内置/内联函数

例1.13 计算三个整数中的大数

```
#include <iostream>
using namespace std;
// 这是一个内置函数，
inline int max(int a,int b,int c)
{if (b>a) a=b;
 if (c>a) a=c;
 return a;
}

int main( )
{int i=7,j=10,k=25,m;
 m=max(i,j,k);
 cout<<"max="<<m<<endl;
 return 0;
}
```

- 由于在定义函数时指定它是内置函数，因此编译系统在遇到函数调用max(i,j,k) 时就用max函数体的代码代替max(i,j,k) ，同时用实参代替形参。

- 调用语句m= max(i,j,k) 就被置换成：

```
{
    a=i ; b = j ; c= k;
    if ( b>a) a=b;
    if ( c>a) a=c;
    m=a;
}
```

1.3.8 内置/内联函数

例1.15 用内置函数计算平方

```
#include <iostream>
using namespace std;
 //定义内置函数
inline int power(int x)
{return x*x;}

int main()
{cout<<power(2)<<endl;
 cout<<power(1+1)<<endl;
 return 0;
}
```

- 编译程序遇见内置函数power时，先求出函数的实参值（1+1=2），然后用power函数体代替函数调用，调用语句变成：

```
{ cout<<2*2<<endl;
  cout<<2*2<<endl;}
```

- 对于函数太复杂以及函数中包含循环的函数，编译器将放弃inline方式。（因为在此情况下内联不能为我们提供任何效率）

- 将函数指定为inline，只是对编译器提出的一种请求，编译器是否执行这项请求要看编译器而定。

1.3.9 作用域运算符

- 并不是所有的变量在程序运行的时时刻刻都是可见的。有的变量在整个程序运行期间都是可见的，称它们为全局变量；有的变量只能在一个函数中可知，被称为局部变量。
- 每个变量都有其有效的作用域，程序只能在变量的有效的作用域内使用变量，不能直接使用其他域中的变量。

程序的内存空间

代码区	程序中各个函数的代码
全局数据区	程序中全局数据和静态数据
堆区	程序中的动态数据
栈区	程序中各函数内的数据

1.3.9 作用域运算符

例1.16 局部变量和全局变量同名

```
#include <iostream>
using namespace std;

float a=13.5;

int main( )
{
    int a=5;
    cout<<a<<endl;
    return 0;
}
```

- 程序中有两个变量a，一个是全局变量，另一个是main函数的局部变量，根据局部变量会屏蔽同名的全局变量规则，在函数中出现的变量a是局部变量，因此输出的值是5，而不是13.5

- 为了在函数中访问全局变量C++提供了作用域运算符 :: ，可以用来指定要访问的作用域，

```
int main( )
  {int a=5;
  cout<<a<<endl;
  cout<<::a<<endl;
  //::a表示全局变量a。注意不能用::访问局部变量。
  return 0;
  }
```

1.3.10 字符串变量

- C++提供了字符串类类型string，实际上它不是C++的基本类型，它是在C++标准库中声明的一个字符串类，程序可以用它定义对象。
- 定义字符串变量格式：　string 变量名表;
- 可在定义变量时用字符串常量为变量赋初值：string　变量名 = 字符串常量;
- 注意：如用字符串变量，在程序开始要用包含语句把C++标准库的string头文件包含进来。

例1.16 局部变量和全局变量同名

```
#include <iostream>
#include <string>
using namespace std;


int main( )
{
    string str1;
    string str2 = "china";
    return 0;
}
```

1.3.10 字符串变量—操作

- **对字符串变量赋值**：字符串变量= 字符串表达式

- **访问字符串中的字符**：C++允许把字符串作为字符数组，第一个字符的下标是0，第二个字符的下标是1，以此类推。

```
string  w = "then";
        w[2] = 'a';
```

- **输入输出字符串**：cin >> 字符串变量
  cout << 字符串变量

字符串操作

```
#include <iostream>
#include <string>
using namespace std;


int main( )
{
    string str1;
    string str2 = “chen”;
    str1 = str2;
    str1[1] = 'x';
    return 0;
}
```

1.3.10 字符串变量—操作

- **字符串连接运算**：字符串1 + 字符串2
- 把连接运算符两端的字符连接成一个字符串。表达式中可以用字符串常量也可以用字符串变量。
- **字符串的比较运算**: 可以用关系运算符>、>=、==、!=、<、<=对两个字符串同一位置的字符进行比较，根据ASCII码值判定字符的大小。

  例："china" > "chinese"
  运算结果是假。

字符串操作

```
#include <iostream>
#include <string>
using namespace std;


int main( )
{
string  st1="C++";
string  st2="Language";
st1 = st1 + st2 ;
return 0;
}
```

1.3.11 动态分配/撤销内存的运算符new和delete

- C的动态存储分配和释放采用函数实现
- C++的动态存储分配和释放采用new和delete运算符来实现。

```
#include <iostream>
using namespace std;
int main( )
{  int *p;
  p = new int;
  // 分配一个存放整型数的存储空间
  *p = 10;
  cout << *p;
  delete p; // 释放这个存储空间
  return 0;
}
```

```
#include <iostream>
using namespace std;
int main( ) {
  int *pArray;
  pArray = new int[10]; // 分配数组，存放10个整型数
  for (int i = 0; i < 10; i++)
  { pArray[i] = i;
    cout<<pArray[i]<<endl;
  }
  delete [ ] pArray;   // 释放数组
  return 0;
}
```

1.3.11 动态分配/撤销内存的运算符new和delete

- new运算符一般格式： type *p = new type [数组维数n]或(初值v)

- 类型是决定分配空间尺寸的关键元素，数组维数n是分配n个类型大小的空间，初值v用来初始化分配的内存.如果运算结果正确，返回值是分配内存空间的起始地址，否则返回NULL。

- 例：

```
int *a=new int ;              // 分配一个整数的空间
int *b=new int(100);          // 分配一个整数空间初值100
char *ch=new char[10];        // 分配10个字符空间
int *q=new int [5][4];        // 分配5x4二维整数数组空间
float *p=new float(3.14159);
```

1.3.11 动态分配/撤销内存的运算符new和delete

- delete运算符一般格式： delete[] 指针变量
- [] 代表数组，如果不是数组可以省略[]。
- 运算功能：撤销指针变量所指的动态内存空间，指针变量的数据类型决定了空间尺寸的大小。
- 例：char *p=new char[10];

  … …

  delete [ ] p;
- 注意不要混合使用new、delete、malloc、free。要正确搭配，不要用new分配内存后，又用free释放内存。

1.3.12 C++对C功能扩展的小结

- 允许使用//开头的注释
- 变量的定义可以出现在程序中的任何行
- 提供标准输入输出流对象cin和cout，不用指定输入输出格式符
- 可以用const定义常变量
- 可以利用函数重载实现同一函数名代表不同的功能类似的函数，方便使用，可读性高
- 可以利用函数模板，简化同一类的函数的编程工作
- 可以使用带默认值的参数的函数，使函数的调用更加灵活

1.3.12 C++对C功能扩展的小结

- 提供变量的引用类型，即为变量提供一个别名，将“引用”作为函数参数，可以实现通过函数的调用来改变实参变量的值
- 增加了内置函数，以提高程序的执行效率
- 增加了单目的作用域运算符，这样在局部变量作用域内也能引用全局变量
- 可以用string类定义字符串变量，使得对字符串的运算更加方便
- 用new和delete运算符代替malloc和free函数，使得分配动态空间更加方便。

1.4 C++程序的编写和实现

- 用C++语言编写程序(编写.cpp文件)：源程序。
- 对源程序进行编译(获得.obj文件)：词法和语法检查；源程序翻译成计算机能识别执行的二进制形式。
- 将目标文件连接(连接obj和lib文件, 生成.exe文件)：“连接程序”将一个程序的所有目标程序和系统的库文件以及系统的其他信息连接起来，形成可执行的二进制文件。
- 运行程序(执行.exe文件，获得结果)
- 分析运行结果(分析结果,排错,重新生成)

面向对象程序设计

Object Oriented Programming

2021

谢谢大家