



# C++ Programming

# C++初步知识 Preliminaries of C++

2025年2月24日

学而不厌 诲人不倦

- ➡ 1.1 从C到C++
- ➡ 1.2 最简单的C++程序
- ➡ **1.3 C++对C的扩充**
- ➡ 1.4 C++程序的编写和实现
- ➡ 1.5 关于C++上机实践



## 1.3 C++对C的扩充

### ➤ 1.3.1 C++的输入输出

#### 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;
}
```

**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。

**注:**

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。

指定输出所占的列数, 可以用控制符setw进行设置 (包含头文件iomanip)

**格式:** `cin>>变量1 [>>变量2 .....`

**功能:** >>是C++的提取运算符, 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。



## 1.3 C++对C的扩充

### ➤ 1.3.2 用const定义常量

#### const 关键字修饰的只读变量

```
const int Max=100;          int const a[5]={1, 2, 3, 4, 5};  
int Array[Max];             const int a[5]={1, 2, 3, 4, 5};
```

```
const int *p; // p 可变, p 指向的对象不可变  
int const *p; // p 可变, p 指向的对象不可变  
int *const p; // p 不可变, p 指向的对象可变  
const int *const p; // 指针 p 和 p 指向的对象都不可变
```

```
#define M 3 //宏常量  
const int N=5; //此时并未将 N 放入内存中  
int i=N; //此时为 N 分配内存, 以后不再分配!  
int l=M; //预编译期间进行宏替换, 分配内存  
int j=N; //没有内存分配 注: 取第一次分配时的地址给 j  
int J=M; //再进行宏替换, 又一次分配内存!
```

```
#include <iostream>  
using namespace std;  
int main( )  
{  
    const int *p1;  
    int const *p2;  
    int a =5;  
    p1 = &a;  
    int b = 3;  
    p1 = &b;  
    b =5;  
    /*p1 = 10;  
    cout<<*p1<<endl;  
    a = 22;  
    cout<<a<<endl;  
    return 0;  
}
```

## 1.3 C++对C的扩充

### ➤ 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。**

回忆：例1.3 求 a 和 b 两个数中的大数。

```
//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);
}
```



## 1.3 C++对C的扩充

### 例1.6 设计程序计算三个数中的大数

#### ➤ 1.3.4 函数的重载

```
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 int p;
    p=max(g,h,i); //函数值为长整型
    cout<<"max_l="<<p<<endl;
    return 0;
}
```

```
#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;
}
float max(float a,float b, float c)
{if (b>a) a=b;
 if (c>a) a=c;
 return a;
}
long max(long a,long b,long c)
{if (b>a) a=b;
 if (c>a) a=c;
 return a;
}
```

**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。**

## 1.3 C++对C的扩充

### ➤ 1.3.4 函数的重载

**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。**

```
#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;
}
//求两个整数中的最大者
int max(int a, int b)
{
    if (a>b) return a;
    else return b;
}
```

### 例1.7 函数重载

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



## 1.3 C++对C的扩充

### ➤ 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**，如果用函数重载的话，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：

↓ 关键字： `template typename`

`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 <class T> 或者 template <typename T>`

`template <class T1, class T2>`



## 1.3 C++对C的扩充

### ➤ 1.3.5 函数模板

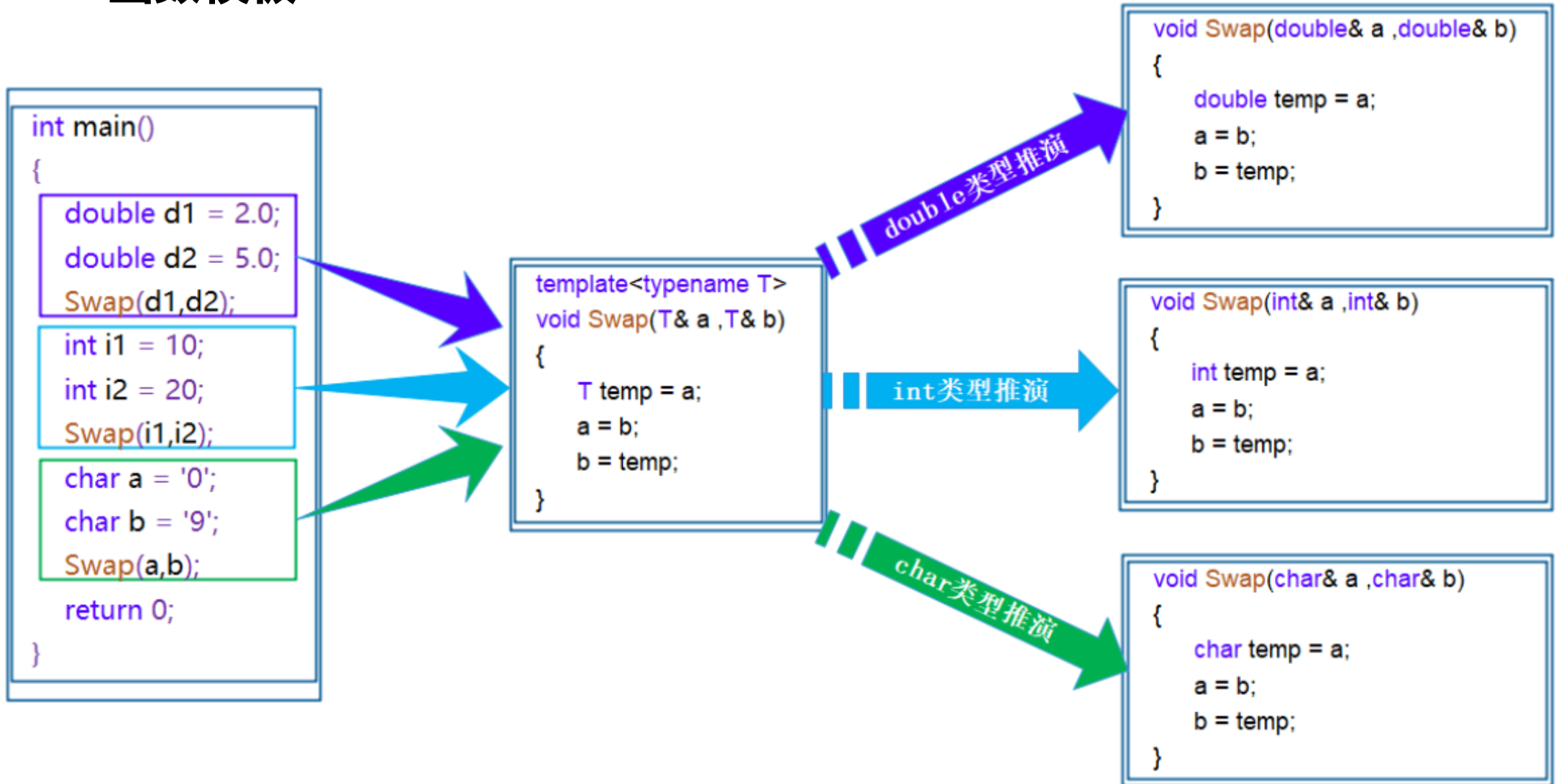
#### 例1.8 函数模板

```
#include <iostream>
using namespace std;
template <typename T>
//用虚拟类型T表示类型
T max(T a,T b,T c)
{
    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 C++对C的扩充

### ➤ 1.3.5 函数模板 泛型编程就是编写与类型无关的通用代码，模板是泛型编程的基础。



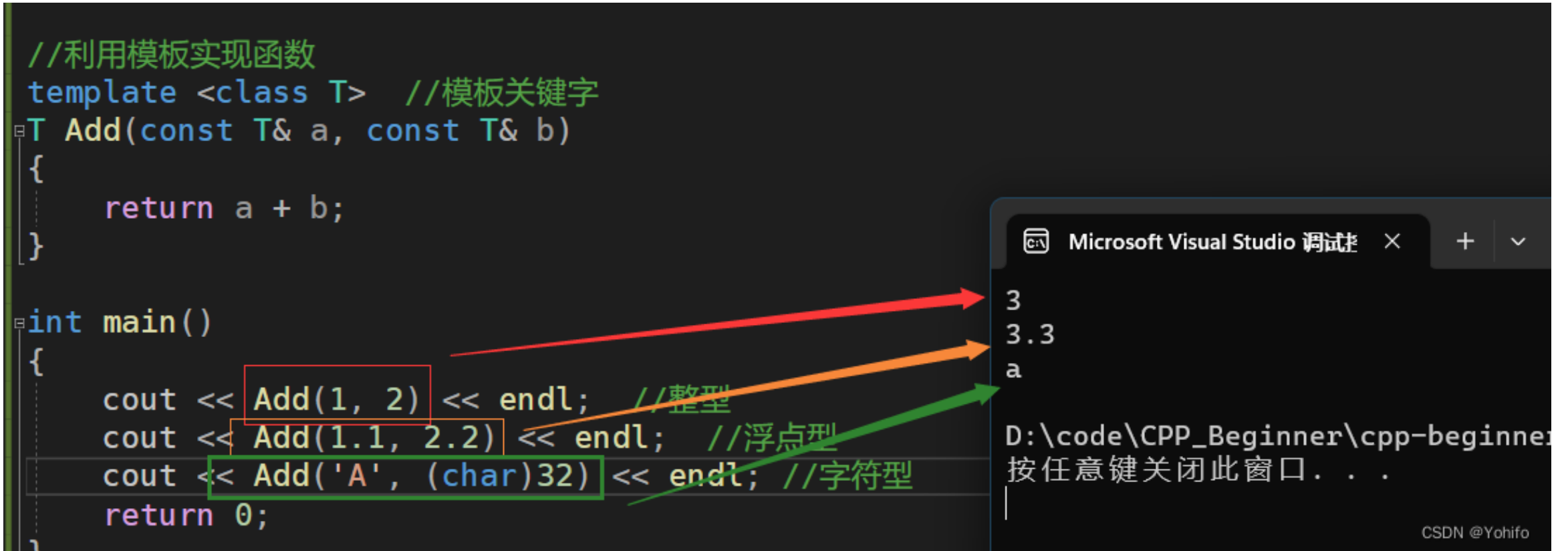
## 1.3 C++对C的扩充

### ➤ 1.3.5 函数模板

泛型编程就是编写与类型无关的通用代码，模板是泛型编程的基础。

```
//利用模板实现函数
template <class T> //模板关键字
T Add(const T& a, const T& b)
{
    return a + b;
}

int main()
{
    cout << Add(1, 2) << endl; //整型
    cout << Add(1.1, 2.2) << endl; //浮点型
    cout << Add('A', (char)32) << endl; //字符型
    return 0;
}
```



Microsoft Visual Studio 调试

3  
3.3  
a

D:\code\CPP\_Beginner\cpp-beginner  
按任意键关闭此窗口...

CSDN @Yohifo



## 1.3 C++对C的扩充

### ➤ 1.3.6 有默认参数的函数

**C++允许为函数的参数设置默认值，调用函数时，如果没有实参，就以默认值作为实参值。**

**例：编写计算圆柱体体积函数**

```
float volume ( float h, float r = 12.5)
```

**注意：一个函数名不能同时用于重载函数和带默认形参值的函数。**

**例：将例1.7中的第三行改为**

```
int max (int a, int b, int c = 100);
```

## 1.3 C++对C的扩充

### ➤ 1.3.7 变量的引用

**C++提供了为变量取别名的功能。**

**格式： 类型 &变量1 = 变量2**

**注意：一个变量可以有多个别名，但两个变量不能用同一个别名。  
引用并不是一种独立的数据类型，声明引用时必须初始化。**

例1.9:

```
int a = 3 ,b =4;  
int &c = a; // c是a 的别名  
int &c = b; // 错误的用法
```

```
int a = 3;  
int & b= a;  
int & c= b; // c也是a的别名
```

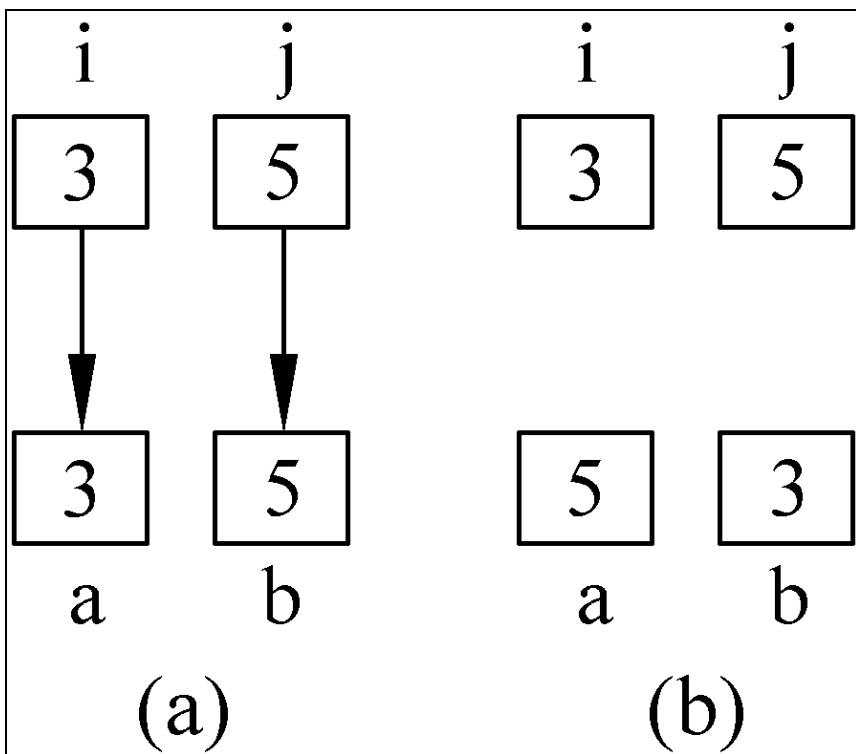
```
int &b=a; //正确  
int &b;    //错误  
float a;  
int &b=a; //错误，必须同类型
```

## 1.3 C++对C的扩充

### ➤ 1.3.7 变量的引用---将引用作为函数参数

#### 例1.10 普通变量做形参

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。



```
#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
}

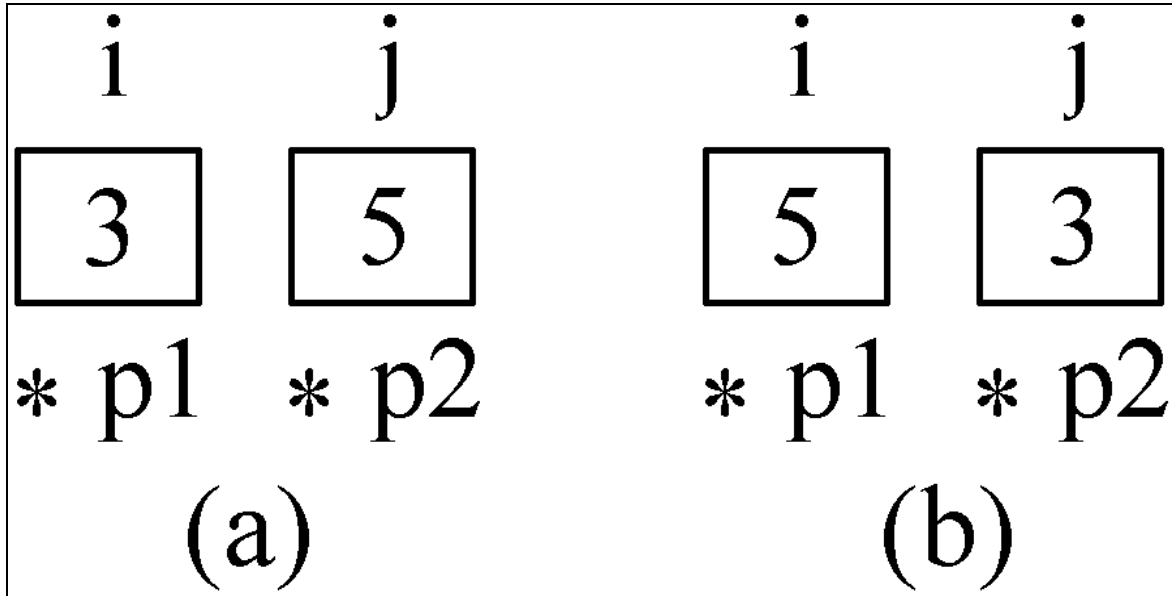
int main( )
{
    int i=3,j=5;
    swap(i,j);
    //i和j的值未互换
    cout<<i<<","<<j<<endl;
    return 0;
}
```

## 1.3 C++对C的扩充

### ➤ 1.3.7 变量的引用---将引用作为函数参数

### 例1.11 指针变量做形参

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。



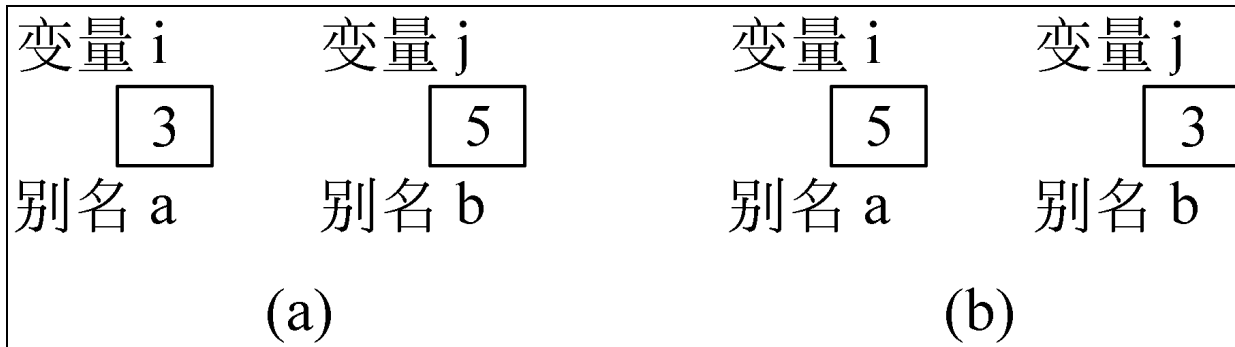
```
#include <iostream>
using namespace std;
void swap(int *p1,int *p2)
{
    int temp;
    temp=*p1;
    *p1= *p2;
    *p2=temp;
}
int main( )
{
    int i=3,j=5;
    swap(&i,&j);
    cout<<i<<","<<j<<endl;
    return 0;
}
```

## 1.3 C++对C的扩充

### ➤ 1.3.7 变量的引用---将引用作为函数参数

### 例1.12 引用变量做形参

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。



```
#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<<","<<j<<endl;
    return 0;
}
```



## 1.3 C++对C的扩充

### ➤ 1.3.7 变量的引用---引用的进一步说明

```
void &a=9; //错误: 不能建立void类型的引用  
char c[6]="hello";  
char &rc[6]=c; //错误: 不能建立引用的数组
```

```
int a=3;  
int &b=a;  
int *p=&b; //等价 int *p=&a  
int &*p=&a //错误 不能建立指向引用类型的指针变量
```

```
int i=5;  
int *p = & i;  
int * &pt = p; // pt是p的别名变量, 同时也是指针变量  
//可以建立指针变量的引用变量
```

## 1.3 C++对C的扩充

### ➤ 1.3.7 变量的引用---引用的进一步说明

//可以建立常引用变量，不允许修改常引用变量的值

```
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
```

//可以用常量或表达式对引用进行初始化，但此时必须用const作声明

```
int i=5;
```

```
const int &a=i+3;
```

```
int temp=i+3;
```

```
const int &a=temp
```

## 1.3 C++对C的扩充

### ➤ 1.3.8 inline内置函数---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，也称内联函数。

**格式：**

```
inline 函数类型 函数名(形参表)
{  函数体  }
```

**inline 是C++的关键字**

## 1.3 C++对C的扩充

### ➤ 1.3.8 inline内置函数---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，求3个整数中的最大者
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;
}
```

### 例1.13/1.14 内联函数

```
//例1.14用内置函数计算平方根
#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;
}
```



## 1.3 C++对C的扩充

### ➤ 1.3.9 作用域运算符

为了在函数中访问全局变量C++提供了**作用域运算符 ::**

**::a表示全局变量a。注意不能用::访问局部变量。**

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

```
//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;
}
```

```
#include <iostream>
using namespace std;
float a=13.5;
int main( )
{
    int a=5;
    cout<<a<<endl;
    cout<<::a<<endl;
    return 0;
}
```



## 1.3 C++对C的扩充

### ➤ 1.3.10 字符串变量

C++提供了字符串类类型string，不是C++的基本类型，是在C++标准库中声明的一个字符串类

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

例：

```
string st1="C++";  
string st2="Language";  
st1 = st1 + st2 ;
```

结果是： C++Language

#### //字符串数组

string name[5]; //包含5个字符串元素

```
string name[5]={"zhang", "Li", "Fan", "Wang", "Tan",}  
//name[0]="zhang"  
//name[1]="Li"  
//name[2]="Fan"  
//name[3]="Wang"  
//name[4]="Tan"
```



## 1.3 C++对C的扩充

### ➤ 1.3.10 字符串变量

C++提供了字符串类类型string，不是C++的基本类型，是在C++标准库中声明的一个字符串类

```
#include <iostream>
#include <string.h>
using namespace std;
int main()
{
    string str1 = "Hello world!";
    cout<<str1<<endl;
    string str2 = "world";
    int j = str1.find(str2);
    cout<<j<<endl;
    return 0;
}
```

//例题：字符串查找

```
#include <iostream>
#include <string.h>
using namespace std;
int main()
{
    string str1 = "Hello world";
    int position=0;
    int i=1;
    //查找某一给定位置后的子串的位置
    while((position=str1.find(str2,position))!=string::npos)
    {
        cout<<"position "<<i<<" : "<<position<<endl;
        position++;
        i++;
    }
    return 0;
}
```

## 1.3 C++对C的扩充

### ➤ 1.3.11 动态分配、撤销内存: new delete

C++提供了字符串类类型string，不是C++的基本类型，是在C++标准库中声明的一个字符串类

例：

```
int *a = new int ;  
int *b = new int( 100);  
char *ch = new char[10];  
int * q = new int [5][4];  
float * p = new float(3.14159);
```

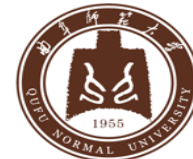
归还动态内存运算

**delete[] 指针变量**

**[] 代表数组，如果不是数组可以省略[]。**



# 小结



对变量的定义可以出现在程序中的任何行  
(但必须在引用该变量前)

cin/cout

const

函数重载

函数模板

默认值参数的函数

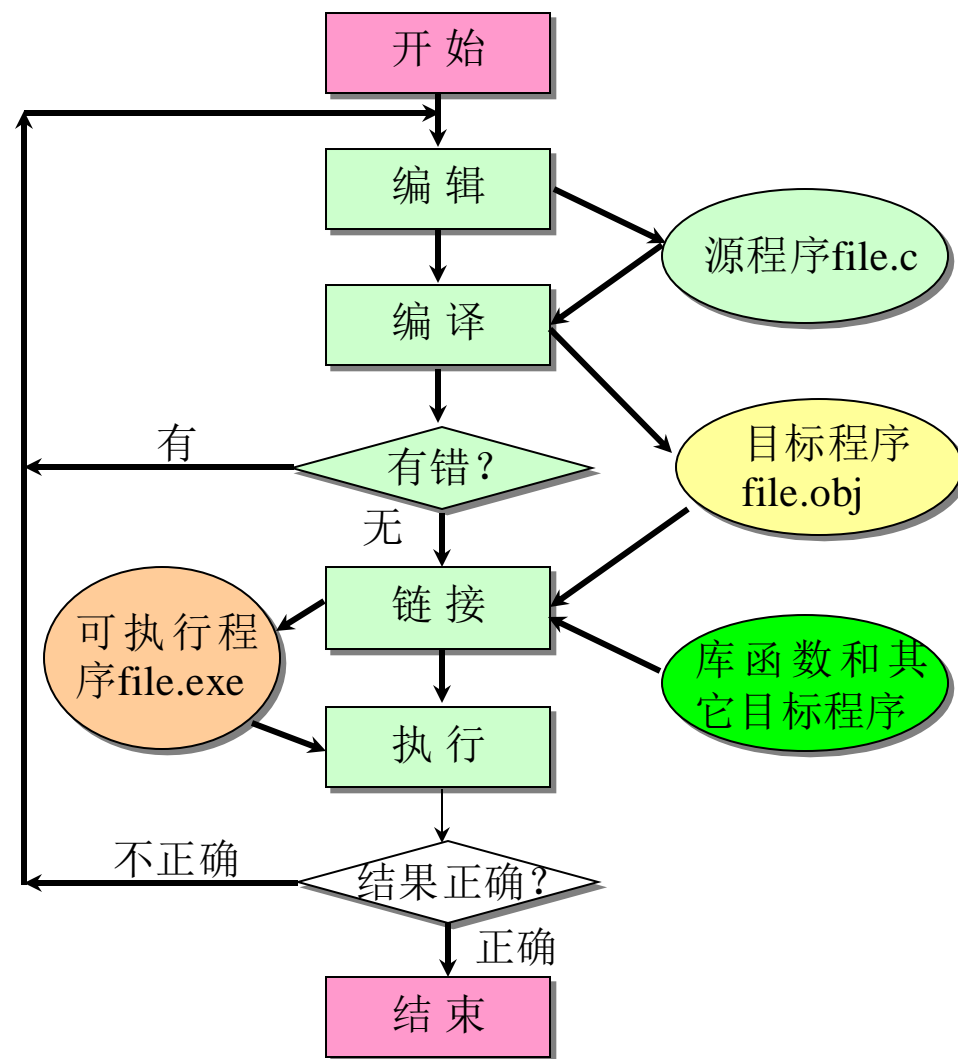
变量引用

内置函数

变量作用域

string

new / delete



调试C/C++程序的流程

## 实验作业二 近10年大学英语四六级考试中单词频率统计

搜集近10年大学英语四六级考试试题（每年分6月和12月两套）共40套（txt文件），完成数据集收集和清洗。用C++语言编程，实现任意单词查询功能，输出单词逐年考查频率，同时输出试题中的完整原题。

**要求：**1. 用C++语言设计类实现基本功能，功能正确，界面友好。

2. 代码注释完整、清晰。

3. 按要求的格式给出实验报告和总结。

4. 在知新平台上传pdf版实验报告，并准备汇报PPT。

5. 2025-04-7课上随机抽取同学进行PPT讲解和演示。

6. 一旦发现雷同或抄袭痕迹，本次实验作业判零分。

2025-02-24



*Thank You !*

*Q & A*