

# C++程序设计

主讲：王老师



尚德机构

学习是一种信仰

## 教材介绍

# C++程序设计

(2008年版)

编著：刘振安

机械工业出版社



## 考试题型

单选题  $1\text{分} \times 20\text{题} = 20\text{分}$

填空题  $1\text{分} \times 20\text{题} = 20\text{分}$

程序改错题  $4\text{分} \times 5\text{题} = 20\text{分}$

完成程序题  $4\text{分} \times 5\text{题} = 20\text{分}$

程序分析题  $5\text{分} \times 2\text{题} = 10\text{分}$

程序设计题  $10\text{分} \times 1\text{题} = 10\text{分}$

# 第一章 认识C++的对象



## 本章主要内容



- 认识C++的函数和对象
- 认识C++语言面向过程编程的特点
- 程序的编辑、编译和运行的基本概念

# §1.1 初识C++的函数和对象

第1章 认识C++的对象

1.1 初识C++的函数和对象

1.2 认识C++语言面向过程编程的特点

1.3 程序的编辑、编译和运行的基本概念

## 1、混合型语言

C++以**.cpp**为文件扩展名，**有且只有一个名为main的主函数**，因保留了这个面向过程的主函数，所以被称为混合型语言。

## 2、注释方式

①从 `"/**"` 开始，到 `"*/"` 结束，如： `/* ..... */`

②从 `"//"` 开始到**本行结束**，如： `//.....`

# §1.1 初识C++的函数和对象

### 3、输入输出对象

①提取操作：用提取操作符“>>”从**cin**输入流中提取字符，如：cin >> a.x ;

②插入操作：用插入操作符“<<”向**cout**输出流中插入字符，如：cout << “we” ; cout << endl;

③使用标准输入（**键盘输入**）cin及标准输出（**屏幕输出**）cout前，要在主函数前使用 **#include <iostream>** 将C++标准输入输出库头文件iostream将其包括。

④换行操作：用语句**cout<<endl;** 或**cout<< “\n”** ; 实现，其中endl可以插在流的中间。

如：cout<<a.x<<endl<<a.y<<endl;



# 使用命名空间

## 4、使用命名空间

C++相比C而言，**可以省略“.h”标识头文件，但必须使用语句using namespace std;**来让命名空间中的对象名称曝光。因此一般的程序都要具有下面的两条语句：

```
#include <iostream>           //包含头文件  
using namespace std;          //使用命名空间
```

注意C++库中代替C库中头文件的正确名称，例如下面两个语句效果一样：

```
① #include <math.h>  
② #include <cmath>  
using namespace std;
```



# 对象的定义和初始化

定义对象包括为它命名并赋予它数据类型，一般即使初值只用来表示该对象尚未具有真正意义的值，也应将每个对象初始化。

C++中利用构造函数语法实现初始化，如：

```
int z(0);    //等同于int z = 0;
```

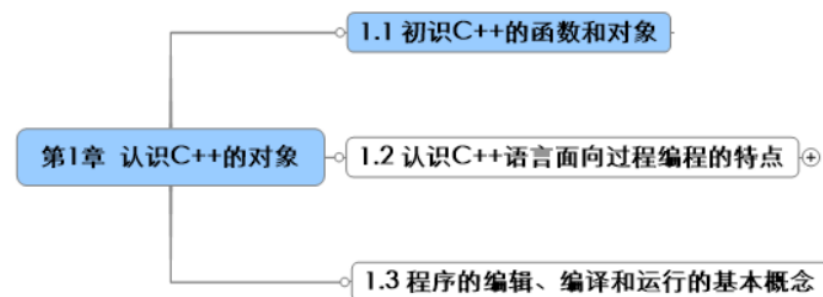
# 函数原型及其返回值

①C++使用变量和函数的基本规则都是：**先声明，后使用**。变量有时也可边声明边使用，但必须声明，否则出错。

比如对函数的调用，要在主函数之前先对调用的函数进行原型声明，如：`int result (int, int)`；它向编译系统声明，后面有一个result函数，该函数有两个整型类型的参数，函数返回整型值。

函数声明时，除了默认参数（需给出默认参数的默认值）和内联函数（需给出函数体及其内语句）外，不需给出参数的变量名称，如果给出，效果也一样，如：`int result (int a, int b)`；和上面的声明效果一样。

# 函数原型及其返回值



②除构造函数与析构函数外，函数都需要有类型声明。

- 如int main ( ) ，指出main是整数类型，**返回值由return后面的表达式决定**，且表达式的值必须与声明函数的类型一致。
- 如果函数确实不需要返回值，还可用**void**标识，一旦使用void标识，函数体内就不再需要使用return语句，否则会编译出错，但可使用return; 语句。

③C++函数有库函数（标准函数，引用时函数名外加< >）和自定义函数（引用时函数名外加“ ”）两类。



# const (常量) 修饰符及预处理程序

第1章 认识C++的对象

1.1 初识C++的函数和对象

1.2 认识C++语言面向过程编程的特点

1.3 程序的编辑、编译和运行的基本概念

①const修饰符：用于定义符号常量。

C中一般使用宏定义“# define”定义常量，而C++中除此外，建议使用const代替宏定义，用关键字**const**修饰的标识符称为**符号常量**。

因const是放在语句定义之前的，因此可以进行类型判别，这比用宏定义更安全一些。如下面两个语句是等同的，但是后者可以比前者避免一些很难发现的错误。

```
# define BUFSIZE 100
```

```
const int BUFSIZE=100;
```



# const (常量) 修饰符及预处理程序

第1章 认识C++的对象

1.1 初识C++的函数和对象

1.2 认识C++语言面向过程编程的特点

1.3 程序的编辑、编译和运行的基本概念

- 常量定义也可使用构造函数的初始化方法，如：  
`const int k (2) ;` //等同于`const int k = 2;`
- 因被const修饰的变量的值在程序中不能被改变，所以在声明符号常量时，必须对符号常量进行初始化，除非这个变量是用extern修饰的外部变量，如：  
`const int d;` ×     `const int d=2;` √     `extern const int d;` √
- const的用处不仅是在常量表达式中代替宏定义，如果一个变量在生存期内的值不会改变，就应该用const来修饰这个变量，以提高程序安全性。



# const (常量) 修饰符及预处理程序

第1章 认识C++的对象

1.1 初识C++的函数和对象

1.2 认识C++语言面向过程编程的特点

1.3 程序的编辑、编译和运行的基本概念

## ②预处理程序

C++的预处理程序不是C++编译程序的一部分，它负责在编译程序的其他部分之前分析处理预处理语句，为与一般的C++语句区别，所有预处理语句都以位于行首的符号“#”开始，作用是把所有出现的、被定义的名字全部替换成对应的“字符序列”。

预处理语句有三种：**宏定义、文件包含（也成嵌入指令）和条件编译。**



# const (常量) 修饰符及预处理程序

第1章 认识C++的对象

1.1 初识C++的函数和对象

1.2 认识C++语言面向过程编程的特点

1.3 程序的编辑、编译和运行的基本概念

文件包含是指一个程序把另一个指定文件的内容包含进来，书写时可以使用引号也可以使用尖括号，前者引用自己定义的包含文件，如：`#include "E:\prog\myfile.h"`，后者引用系统提供的包含文件，如标准输入输出是定义在标准库*iostream*中的，引用时要包括以下两条语句：

```
#include <iostream>           //包含头文件
using namespace std;         //使用命名空间
```



```
#include<iostream.h>
```

```
#define PRICE 30 //常量，在程序中保持不变
```

```
void main(void)
```

```
{ int num, total; //定义变量,在内存中开辟区间
```

```
    num=10;    //变量赋值,10为常量
```

```
    total=num*PRICE;
```

```
    cout<<"total="<<total; //输出结果
```

```
}
```

其中: num=10

total=num\*PRICE

是赋值号，不同于数学意义上的等号。

num	total
10	300

PRICE
30



## 课堂练习

对C++中主函数描述正确的是( )

A:名称为main, 可为多个

B:名称不限, 可为多个

C:名称为main, 必须有且只能有一个

D:名称不限, 必须有且只能有一个



## 课堂练习

对C++中主函数描述正确的是( )

A:名称为main, 可为多个

B:名称不限, 可为多个

C:名称为main, 必须有且只能有一个

D:名称不限, 必须有且只能有一个

答案: C

## 课堂练习

C++源程序文件扩展名为(     )

A:.cpp

B:.h

C:.lib

D:.obj



## 课堂练习

C++源程序文件扩展名为( )

A:.cpp

B:.h

C:.lib

D:.obj

答案：A



## 课堂练习

在C++中使用流进行输入输出，其中专用于从键盘进行输入的流是( )

A:cerr

B:cin

C:cout

D:cfile

## 课堂练习

在C++中使用流进行输入输出，其中专用于从键盘进行输入的流是( )

A:cerr

B:cin

C:cout

D:cfile

答案： B





## 课堂练习

对C++语言和C语言的兼容性，描述正确的是（ ）

A:C++兼容C

B:C++部分兼容C

C:C++不兼容C

D:C兼容C++



## 课堂练习

对C++语言和C语言的兼容性，描述正确的是（ ）

A:C++兼容C

B:C++部分兼容C

C:C++不兼容C

D:C兼容C++

答案：A



## 课堂练习

以下说法中正确的是 ( )

A:C++总是从第一个定义的函数开始执行

B:C++程序总是从main函数开始执行

C:C++中函数必须有返回值

D:C++中函数名必须唯一



## 课堂练习

以下说法中正确的是（ ）

A:C++总是从第一个定义的函数开始执行

B:C++程序总是从main函数开始执行

C:C++中函数必须有返回值

D:C++中函数名必须唯一

答案：B

# 程序书写格式

C++的格式和C一样，都很自由，一行可以写几条语句，但也要注意以下规则，增加可读性：

- ①括号紧跟函数名后面，但在for和while后面，应用一个空格与左括号隔开；
- ②数学运算符左右各留一个空格，以与表达式区别；
- ③在表示参数时，逗号后面留一个空格；
- ④在for、do...while和while语句中，合理使用缩进、一对花括号和空行；
- ⑤适当增加空行和程序注释以增加可读性；
- ⑥太长的程序分为两行或几行，并注意选取合适的分行和缩进位置。

# 一个简单的C++程序

```
#include<iostream.h>
```

包含文件

函数体  
开始

主函数

```
void main(void )
```

分号，一条完整  
语句的结束符

```
{ cout<<"I am a student.\n"; //输出字符串
```

```
}
```

函数体  
结束

输出流，在屏幕上打  
印引号内的字符串

注释或说明

本程序编译执行后，在屏幕上显示：

**I am a student.**

另一个例子

```
#include <iostream.h>
```

```
void main(void)
```

```
{
```

```
    cout << "i="; //显示提示符
```

```
    int i;        //说明变量i
```

```
    cin >>i;      //从键盘上输入变量i的值
```

```
    cout << "i的值为: " <<i<<'\n'; // 输出变量i的值
```

```
}
```



# 标识符

这些标识符合法吗？  
1A、M.D.John、¥123、#33、  
Tatol、int、max

## ★标识符

- ❖定义：标识变量名、符号常量名、函数名、数组名、文件名的字符串序列——**名字**。
- ❖命名规则：
  - 只能由**字母**、**数字**、**下划线**组成，且**第一个字符必须是不能是数字**
  - 大小写字母含义不同，一般用**小写**
  - 不能使用**关键字**
- ❖使用：**先定义、后使用**



标识符应该“**见名知意**”，如 total，max  
标识符应该“**不宜混淆**”，如 1与l，o与0



变量名的命名方法：

变量名、数组名、函数名...称为**标识符**。

标识符只能由**字母、数字、下划线**这三种字符组成，且第一个字符必须为字母或下划线，长度不大于**247**个字符，**大小写不通用**。（关键字不能作为标识符）。

**关键字**即是**VC++**的语法要求中使用的字。

如 **int if while** 等。

正确的标识符：**INT, sum, de12, SUM**等。变量必须使用前定义，以分配空间。

表 2-2 C 语言关键字的含义

关 键 字	含 义	关 键 字	含 义	关 键 字	含 义
auto	自动	extern	外部	sizeof	计算字节数
break	中止	float	浮点	static	静态
case	情况	for	对于	struct	结构
char	字符	goto	转向	switch	开关
continue	继续	int	整	typedef	类型定义
const	常量	if	如果	union	共用
default	默认	long	长	unsigned	无符号
do	做	register	寄存器	void	空
double	双	return	返回	volatile	可变的
else	否则	short	短	while	当
enum	枚举	signed	有符号		

## 附录 按字母表顺序排列的 C 和 C++ 保留字

#	catch	if	struct
##	char	inline	switch
#define	clase	int	template
#elif	const	long	this
#else	const _ cast	mutable	throw
#endif	continue	namespace	true
#error	default	new	try
#if	defined	operator	typedef
#ifdef	delete	private	typeid
#ifndef	do	protected	union
#include	double	public	unsigned
#line	else	register	using
#pragma	enum	reinterpret _ cat	virtual
#undef	extern	return	void
asm	false	short	volatile
auto	float	signed	while
bool	for	sizeof	
break	friend	static	
case	goto	static _ cast	

## 课堂练习

可用作C++语言用户标识符的一组标识符是（ ）。

A:void, define, +WORD

B:a3\_b3, \_123, YN

C:for, -abc, Case

D:2a, DO, sizeof



## 课堂练习

可用作C++语言用户标识符的一组标识符是（ ）。

A:void, define, +WORD

B:a3\_b3, \_123, YN

C:for, -abc, Case

D:2a, DO, sizeof

答案：B

## 课堂练习

下列不属于C++关键字的是（ ）。

A:extern

B:goto

C:free

D:default



## 课堂练习

下列不属于C++关键字的是（ ）。

A:extern

B:goto

C:free

D:default

答案：C

## 运算符

算术运算符：（+ - \* / % +（取正） -（取负））

自增、自减运算符 ++ --

关系运算符：（< <= == > >= !=）

逻辑运算符：（! && ||）

位运算符：（<< >> ~ | ^ &）

赋值运算符：（= += -= \*= /=等）

条件运算符：（?:）

逗号运算符：（,）

数据长度运算符：（sizeof）

强制类型转换：（类型）

成员运算符：（. ->）

下标运算符：（[]）

指针运算符：（\* &）

## ★学习运算符应注意：

❖运算符功能

❖与运算量关系

●要求运算量个数

●要求运算量类型

❖运算符优先级别

❖结合方向

❖结果的类型

## 结合性

结合性是C语言的独有概念。除单目运算符、赋值运算符和条件运算符是右结合之外，其他的运算符都是左结合。

表 3-4 C 语言运算符的优先级和结合性

优 先 级	运 算 符	结 合 性
1	() [] -> .	从左至右
2	! ~ ++ -- * (取内容) & (取地址) + (取正) - (取负) sizeof	从右至左
3	* (乘) / %	从左至右
4	+ (加) - (减)	
5	<< >>	
6	< <= > >=	
7	== !=	
8	& (位逻辑与)	从右至左
9	^	
10		
11	&&	
12		
13	?:	从右至左
14	= += -= *= /= %= &= ^=  = >>= <<=	
15	,	从左至右

注：优先级的数字越小，级别越高。

表 3-12 复合赋值运算符的说明

名称	运算符	运算对象个数	运算规则	运算对象类型	结果类型	结合性
加赋值	+=	双目 中缀	$x+=y$ 等价于 $x=x+(y)$	数值型	数值型	从右至左
减赋值	-=		$x-=y$ 等价于 $x=x-(y)$			
乘赋值	*=		$x*=y$ 等价于 $x=x*(y)$			
除赋值	/=		$x/=y$ 等价于 $x=x/(y)$			
模赋值	%=		$x\%=y$ 等价于 $x=x\%(y)$	整型	整型	
位与赋值	&=		$a\&=b$ 等价于 $a=a\&(b)$			
位或赋值	=		$a =b$ 等价于 $a=a (b)$			
位异或赋值	^=		$a^=b$ 等价于 $a=a^(b)$			
位左移赋值	<<=		$a<<=b$ 等价于 $a=a<<(b)$			
位右移赋值	>>=		$a>>=b$ 等价于 $a=a>>(b)$			

$a+=3 \iff a=a+3$

$x*=y+8 \iff x=x*(y+8)$

$x\%=3 \iff x=x\%3$



## 赋值表达式

由赋值运算符或复合赋值运算符将一个变量和一个表达式连接起来的表达式，称为赋值表达式。

赋值表达式的构成规则：**变量=表达式**

变量 复合赋值运算符 表达式

其中，表达式可以是算术表达式、关系表达式、逻辑表达式、条件表达式、赋值表达式和逗号表达式等。由于逗号运算符的优先级低于赋值运算符，所以应注意加圆括号。

例：

a=b=c=5	// 表达式值为5, a, b, c值为5
a=(b=5)	// b=5; a=5
a=5+(c=6)	// 表达式值11, c=6, a=11
a=(b=4)+(c=6)	// 表达式值10, a=10, b=4, c=6
a=(b=10)/(c=2)	// 表达式值5, a=5, b=10, c=2

# 逗号运算符和逗号表达式

- ❖形式: 表达式1,表达式2,.....表达式n
- ❖结合性:从左向右
- ❖优先级: 15, 级别最低
- ❖逗号表达式的值: 等于表达式n的值
- ❖用途: 常用于循环for语句中

例: 逗号表达式使用

main()

```
{ int x,y=7;  
  float z=4;  
  x=(y=y+6,y/z);  
  printf("x=%d\n",x);  
}
```

运行结果:

X=3

例 a=3\*5,a\*4 //a=15,表达式值60

a=3\*5,a\*4,a+5 //a=15,表达式值20

例 x=(a=3,6\*3) //赋值表达式, 表达式值18, x=18

x=a=3,6\*a //逗号表达式,表达式值18,x=3

例 a=1;b=2;c=3;

cout<<a<<','<<b<<','<<c); //1,2,3

cout<<(a,b,c)<<','<<b<<','<<c; //3,2,3

# 数据类型的转换

自动类型转换：

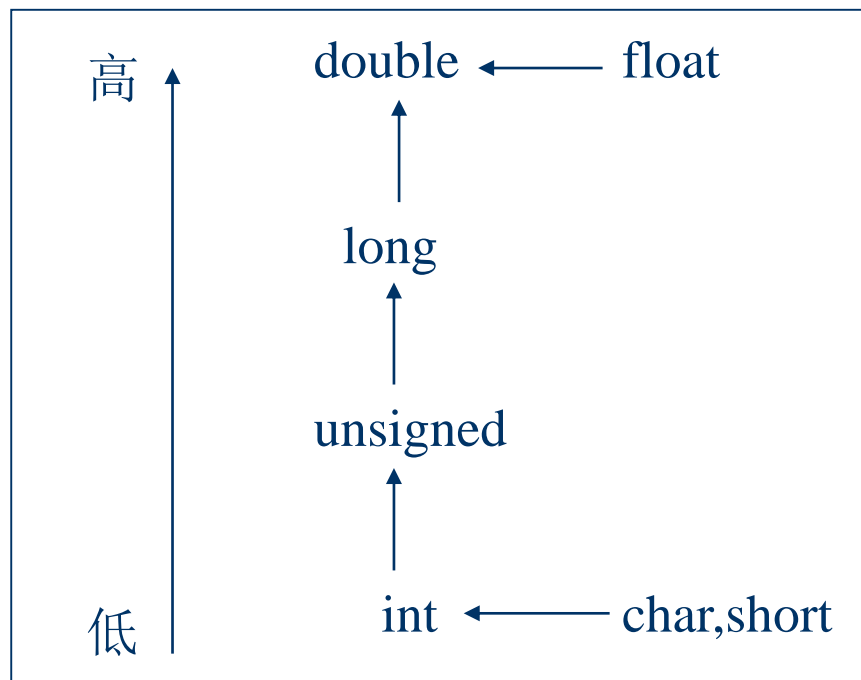
参与运算的各个数据都转换成数据较长的数据类型，然后计算，结果的类型就是数据长度较长的数据类型。本质上是较低类型的转换，即“就高不就低”或“就长不就短”的原则。

## ❖ 什么情况下发生

- **运算转换**-----不同类型数据混合运算时
- **赋值转换**-----把一个值赋给与其类型不同的变量时
- **输出转换**-----输出时转换成指定的输出格式
- **函数调用转换**-----实参与形参类型不一致时转换

❖ **运算转换规则**：不同类型数据运算时先**自动**转换成同一类型





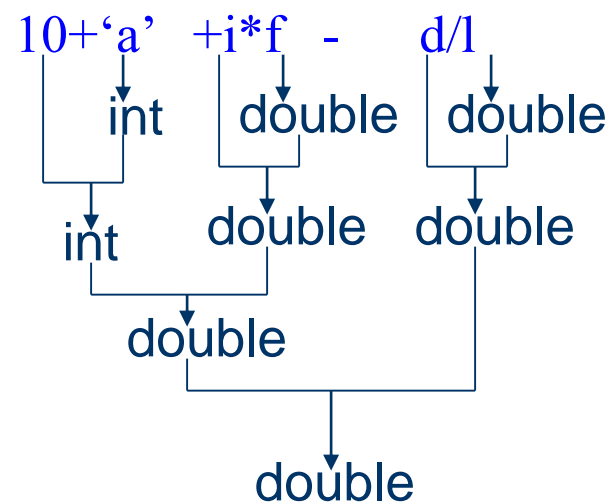
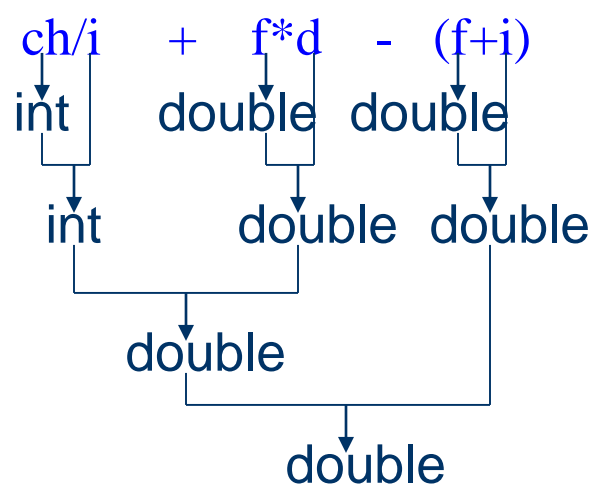
说明:

← 必定的转换

↑ 运算对象类型不同时转换

例 int i;  
float f;  
double d;  
long l;

例 char ch;  
int i;  
float f;  
double d;





## 课堂练习

若有以下类型标识符定义：int x=2;char w= 'a' ;float y=23.45f;double z=45.6789;则表达式w\*x+y-z的结果类型是 ( )

A:float

B:char

C:int

D:double

## 课堂练习

若有以下类型标识符定义：int x=2;char w= 'a' ;float y=23.45f;double z=45.6789;则表达式w\*x+y-z的结果类型是 ( )

A:float

B:char

C:int

D:double

答案：D



## §1.2 认识C++ 面向过程编译的特点

1.2 认识C++语言面向过程编程的特点

1.2.1 使用函数重载
1.2.2 新的基本数据类型及其注意事项 L(l) F(f) O(0) x(x)
1.2.3 动态分配内存 new delete
1.2.4 引用
1.2.5 对指针使用const限定符
1.2.6 泛型算法应用于普通数组
1.2.7 数据的简单输入输出格式

### 一、使用函数重载

C++ 允许为同一个函数定义几个版本，从而使一个函数名具有多种功能，这称为**函数重载**。

假设有一个函数max，分别具有以下函数原型：

```
int max (int, int) ;    //2个整型参数的函数原型
```

```
int max (int, int, int) ; //3个整型参数的函数原型
```

只要分别为不同参数的max编制相应的函数体，就可以实现各自的功能。



# 新的基本数据类型及其注意事项

1.2 认识C++语言面向过程编程的特点

1.2.1 使用函数重载

1.2.2 新的基本数据类型及其注意事项 L(l) F(f) 0 0x

1.2.3 动态分配内存 new delete

1.2.4 引用

1.2.5 对指针使用const限定符

1.2.6 泛型算法应用于普通数组

1.2.7 数据的简单输入输出格式

- 1、void是无类型标识符，只能声明函数的返回值类型，不能声明变量。
- 2、C++还比C多了**bool**（布尔）型。
- 3、C++只限定int和short至少要有16位，而long至少32位，short不得长于int，int不能长于long，VC++6.0规定int使用4字节，这与C使用2字节不同。



# 新的基本数据类型及其注意事项

1.2 认识C++语言面向过程编程的特点

- 1.2.1 使用函数重载
- 1.2.2 新的基本数据类型及其注意事项 L(I) F(I) 0 0x
- 1.2.3 动态分配内存 new delete
- 1.2.4 引用
- 1.2.5 对指针使用const限定符
- 1.2.6 泛型算法应用于普通数组
- 1.2.7 数据的简单输入输出格式

4、地址运算符 “&” 用来取对象存储的**首地址**，对于数组，则数组名就是数组的首地址。

如：int x = 56; 定义x，VC++ 6.0使用4个字节存储对象56，假设存放的内存首地址用十六进制表示为006AFDEC，则语句cout <<&x; 自动使用十六进制输出存储的首地址006AFDEC。

# 新的基本数据类型及其注意事项

1.2 认识C++语言面向过程编程的特点

1.2.1 使用函数重载

1.2.2 新的基本数据类型及其注意事项 L(l) F(f) 0 0x

1.2.3 动态分配内存 new delete

1.2.4 引用

1.2.5 对指针使用const限定符

1.2.6 泛型算法应用于普通数组

1.2.7 数据的简单输入输出格式

## 5、C++中的常量分三种：

- 第一种为符号常量；
- 第二种为整数常量，有4种类型，分别为十进制、长整型（后缀L（l））、八进制（前缀0）、十六进制（前缀0x），并用前缀和后缀进行分类标识；
- 第三种为浮点常量，有三种类型，分别为float型、long float型、double型，并用后缀进行分类识别。



# 新的基本数据类型及其注意事项

1.2 认识C++语言面向过程编程的特点

1.2.1 使用函数重载

1.2.2 新的基本数据类型及其注意事项 L(l) F(f) 0 0x

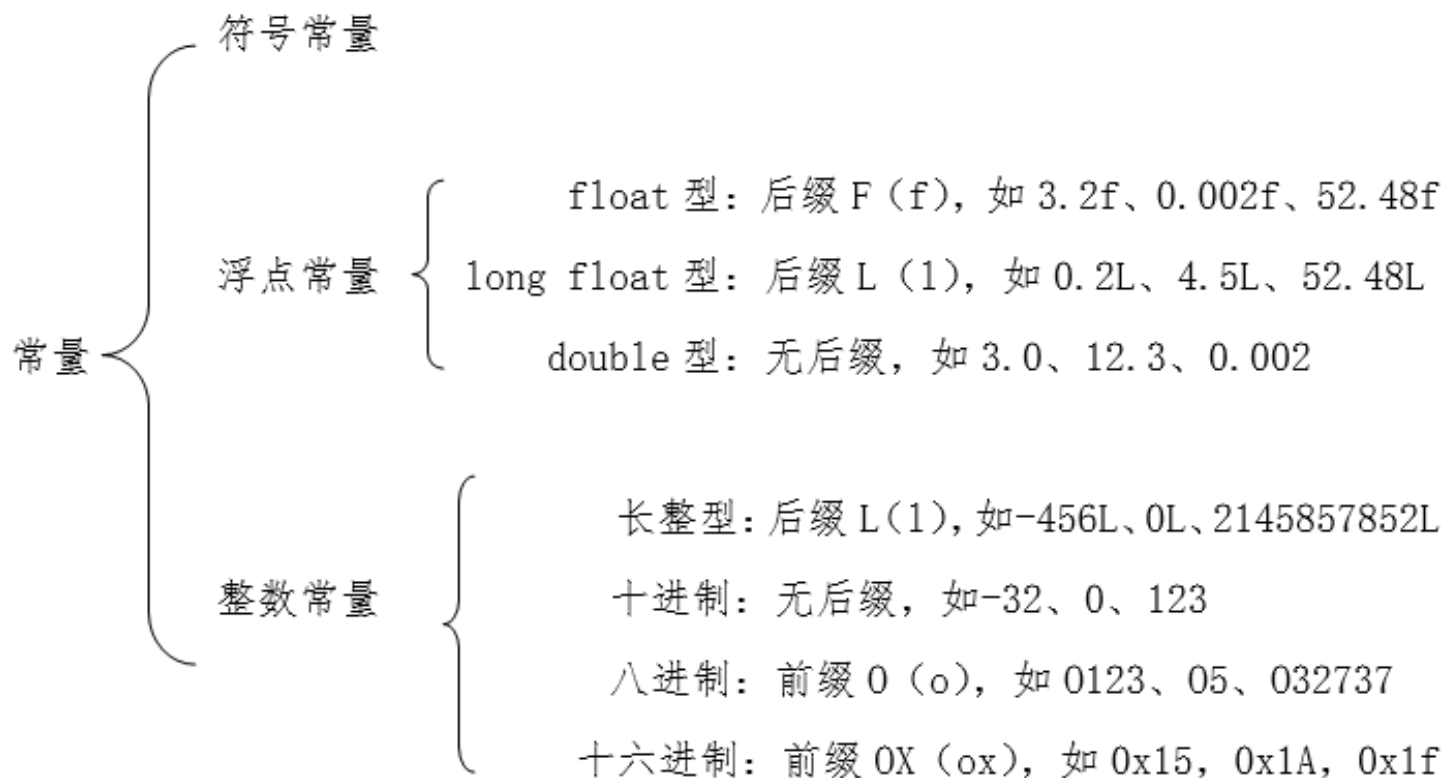
1.2.3 动态分配内存 new delete

1.2.4 引用

1.2.5 对指针使用const限定符

1.2.6 泛型算法应用于普通数组

1.2.7 数据的简单输入输出格式



C++ 与 C 一样，也使用转义序列。如：' \0' 表示 ASCII 码值为零的空字符 (NULL)，' \101' 表示字符 A。

## 动态分配内存

1.2 认识C++语言面向过程编程的特点

1.2.1 使用函数重载

1.2.2 新的基本数据类型及其注意事项 L(l) F(f) 0 0x

1.2.3 动态分配内存 new delete

1.2.4 引用

1.2.5 对指针使用const限定符

1.2.6 泛型算法应用于普通数组

1.2.7 数据的简单输入输出格式

1、在使用指针时，如果不使用对象地址初始化指针，可以自己给它分配地址。

对于只存储一个基本类型数据的指针，申请方式如下：

`new 类型名[size]`      //申请可以存储size个该数据类型的对象

不再使用时，必须使用`delete 指针名`；来释放已经申请的存储空间。

如： .....

`double *p;`              //声明double型指针

`p = new double[3]`      //分配3个double型数据的存储空间

.....

`delete p;`              //释放已申请的存储空间



## 动态分配内存

1.2 认识C++语言面向过程编程的特点

1.2.1 使用函数重载

1.2.2 新的基本数据类型及其注意事项 L(l) F(f) O(0) x

1.2.3 动态分配内存 new delete

1.2.4 引用

1.2.5 对指针使用const限定符

1.2.6 泛型算法应用于普通数组

1.2.7 数据的简单输入输出格式

2、C必须在可执行语句之前集中声明变量，而C++可以在使用对象时再声明或定义。

3、C++为结构动态分配内存一般格式为：

指针名 = new 结构名; //分配

delete 指针名; //释放

例如给书中例1.1的Point结构指针分配内存：

p = new Point;

当不再使用这个空间时，必须使用delete p; 释放空间。.....

## 引用

1、引用简单的说，就是为现有的对象起个**别名**，别名的地址与引用对象的地址是一样的。

引用的声明方式为：

**数据类型 & 别名 = 对象名；**

注意对象在引用前必须先初始化，另外声明中符号“&”的位置无关紧要，比如

`int& a = x; 、 int & a = x; 和int &a = x; 等效。`

例： .....

```
int x = 56;    //定义并初始化x
```

```
int & a = x;    //声明a是x的引用，二者地址相同
```

```
int &r = a;     //声明r是a的引用，二者地址相同
```

.....

```
r = 25;        //改变r，则a和x都同步变化
```

## 引用

2、所谓“引用”，就是将一个新标识符和一块已经存在的存储区域相关联。因此，使用引用时没有分配新的存储区域，它本身不是新的数据类型。

可以通过修改引用来修改原对象，但是不能有空引用，在程序中必须确保引用是和一块正确的存储区域关联。

引用通常用于函数的参数表中或作为函数的返回值。前者因为使用引用作为函数参数不产生临时对象，可提高程序执行效率和安全性（§4.4.3），后者则是因为引用作为函数返回值可用于赋值运算符的左边。

## 引用

3、引用实际上就是变量的别名，使用引用就如同直接使用变量一样，引用与变量名在使用的形式上完全一样，引用只是作为一种标识对象的手段。

但要注意，可以声明指向变量或引用的指针，

如：int \*p=&x; √

int &a = x; int \* p = &a; √;

也可以声明指针对指针的引用，如：int \* & p2 = p1; √

（式中p1、p2是指针，\* 声明p2是指针，&声明p2是p1的引用）；

但不能声明指针对变量的引用，如：int \* &P = &x; ×；

不能声明引用的引用，如：int & & r = x; ×；

也不能直接声明对数组的引用。

## 引用

1.2 认识C++语言面向过程编程的特点

1.2.1 使用函数重载
1.2.2 新的基本数据类型及其注意事项 L(i) F(f) 0 0x
1.2.3 动态分配内存 new delete
1.2.4 引用
1.2.5 对指针使用const限定符
1.2.6 泛型算法应用于普通数组
1.2.7 数据的简单输入输出格式

4、引用的作用与指针有相似之处，它会对内存地址上存在的变量进行修改，但它不占用新的地址，从而节省开销。二者除使用形式不同，本质也不同：**指针是低级的直接操作内存地址的机制**，可由整型数强制类型转换得到，功能强大但易出错；**引用则是较高级的封装了指针的特性**，不直接操作内存地址，不可由强制类型转换而得，安全性较高。

## 引用

5、虽然不能直接定义对数组的引用，但可以通过typedef来间接的建立对数组的引用。如：

.....

```
typedef int array[10]; //定义int型数组类型array
```

.....

```
array a = {12, 34, .....}; //定义array型数组a
```

```
array & b = a; //定义数组a的引用b
```

.....

# 对指针使用const限定符

1.2 认识C++语言面向过程编程的特点

1.2.1 使用函数重载

1.2.2 新的基本数据类型及其注意事项 L(l) F(f) 0 0x

1.2.3 动态分配内存 new delete

1.2.4 引用

1.2.5 对指针使用const限定符

1.2.6 泛型算法应用于普通数组

1.2.7 数据的简单输入输出格式

### 1、变量的指针、指向变量的指针变量、指针变量指向的变量

变量的指针就是变量的地址，存放变量地址的变量是指针变量，为了表示指针变量和它所指向的变量之间的联系，在程序中用“\*”符号表示“指向”。例如用p代表指针变量，来存放变量a所在的内存地址，则\*p代表指针变量指向的变量，也就是变量a，且下面等式成立：

$p = \&a$      $*p = * \&a = a$      $\&*p = \&a$      $(*p)++ = a++$

# 左值和右值

1.2 认识C++语言面向过程编程的特点

1.2.1 使用函数重载

1.2.2 新的基本数据类型及其注意事项 L(l) F(f) O(0) x

1.2.3 动态分配内存 new delete

1.2.4 引用

1.2.5 对指针使用const限定符

1.2.6 泛型算法应用于普通数组

1.2.7 数据的简单输入输出格式

左值是指某个对象的表达式，必须是可变的。

左值表达式在赋值语句中即可作为左操作数，也可作为右操作数，如： $x = 56$ ；  
和 $y = x$ ；

而右值56就只能作为右操作数，不能作为左操作数。

某些运算符如指针运算符“\*”和取首地址运算符“&”也可产生左值，例如p是一个指针类型的表达式，则“\*p”是左值表达式，代表由p指向的对象，且可通过“\*p = ”改变这个对象的值；

“&p”也是左值表达式，代表由p指向的对象的首地址，且可通过“&p = ”改变这个指针的指向。



## 指向常量的指针

(`const int *p = &x;` “`*p =`” 的操作不成立)

1.2 认识C++语言面向过程编程的特点

1.2.1 使用函数重载

1.2.2 新的基本数据类型及其注意事项 L(l) F(f) O 0x

1.2.3 动态分配内存 new delete

1.2.4 引用

1.2.5 对指针使用const限定符

1.2.6 泛型算法应用于普通数组

1.2.7 数据的简单输入输出格式

指向常量的指针是在非常量指针声明前面使用const，如：`const int *p;`，它告诉编译器，`*p`是常量，不能将`*p`作为左值进行操作，即限定了“`*p =`”的操作，所以称为指向常量的指针。如：

```
const int y = 58;
```

```
const int *p1 = &y; //指向常量的指针指向常量y，y不能作为左值
```

```
int x = 45;
```

```
const int *p = &x; //只能通过左值x间接改变*p的值
```

上式中`*p`不能作为左值，但可以通过“`x =`”改变`x`的值，间接改变`*p`的值，即const仅是限制使用`*p`的方式，`*p`仍然可以作为右值使用，还可以通过运算符`&`改变指针所指向的地址，但不能改变指针所指向的内存地址中的内容。



## 常量指针

(`int * const p = &x;` “`p =`” 的操作不成立)

1.2 认识C++语言面向过程编程的特点

1.2.1 使用函数重载

1.2.2 新的基本数据类型及其注意事项 `L(l) F(f) O(0) x`

1.2.3 动态分配内存 `new delete`

1.2.4 引用

1.2.5 对指针使用const限定符

1.2.6 泛型算法应用于普通数组

1.2.7 数据的简单输入输出格式

把const限定符放在\*号的右边，就可使指针本身成为一个const，即常量指针。

如：

```
int x = 5;
```

```
int * const p = &x;
```

式中的指针本身是常量，编译器要求给它一个初始化值，这个值在指针的整个生存期中都不会改变，编译器把p看作常量地址，所以不能作为左值，即“p =”不成立，也就是说不能改变指针p所指向的地址。但这个内存地址里的内容可以使用间接引用运算符\*改变其值，例如语句 `*p = 56;` 将上面的x的值改变为56。

# 指向常量的常量指针

1.2 认识C++语言面向过程编程的特点

1.2.1 使用函数重载

1.2.2 新的基本数据类型及其注意事项 L(l) F(f) O(0) x

1.2.3 动态分配内存 new delete

1.2.4 引用

1.2.5 对指针使用const限定符

1.2.6 泛型算法应用于普通数组

1.2.7 数据的简单输入输出格式

也可以声明指针本身和所指向的对象都不能改变的“指向常量的常量指针”，这时必须要初始化指针。如：

```
int x = 2;
```

```
const int * const p = &x;
```

语句告诉编译器，\*p和p都是常量，都不能作为左值，即“\*p = ”和“p = ”两操作均不成立，这种指针限制“&”和“\*”运算符，在实际中很少用。

## 泛型算法应用于普通数组

### 1、数组中元素及位置的关系

如`int a[] = {5, 6, 7, 8};`

则数组中各元素分别为： $a[0] = 5$ ， $a[1] = 6$ ， $a[2] = 7$ ， $a[3] = 8$ 。 $a$ 为数组的起始地址，各元素的位置分别是： $a + 1$ 位置为5， $a + 2$ 位置为6， $a + 3$ 位置为7， $a + 4$ 位置为8。对数组按元素位置进行操作时，不包括起始位置，如：语句`sort (a + 1, a + 4)`；，只对从 $a + 1$ 位置（不含 $a + 1$ 位置的元素）起到 $a + 4$ 位置（含 $a + 4$ 位置的元素）为止的各元素进行操作，即 $a + 2$ ， $a + 3$ ， $a + 4$ 三个位置上的三个元素，而不是 $a + 1 \sim a + 4$ 四个位置上的所有4个元素。

注意式中的 $a + 1$ 并不是地址 $a$ 加上一个字节后的地址，而是 $a + 1 \times d$ 得到的地址，其中 $d$ 是元素类型占用的字节数，比如C++中整型数占用4个字节，则 $a + 1$ 位置上元素的地址就是地址 $a$ 加上4个字节后得到的地址。

# 泛型算法应用于普通数组

1.2 认识C++语言面向过程编程的特点

- 1.2.1 使用函数重载
- 1.2.2 新的基本数据类型及其注意事项 L(i) F(i) O Ox
- 1.2.3 动态分配内存 new delete
- 1.2.4 引用
- 1.2.5 对指针使用const限定符
- 1.2.6 泛型算法应用于普通数组
- 1.2.7 数据的简单输入输出格式

2、数组不能作为整体输出，C++引入STL库提供的泛型算法，大大简化数组操作。所谓泛型算法，就是提供的操作与元素的类型无关。

3、对数组内容进行升幂、输出、反转和复制等操作需要包含头文件 `<algorithm>`；对数组内容进行降幂和检索等操作需要包含头文件 `<functional>`。

## 泛型算法应用于普通数组

1.2 认识C++语言面向过程编程的特点

1.2.1 使用函数重载
1.2.2 新的基本数据类型及其注意事项 L(l) F(f) O(0) x
1.2.3 动态分配内存 new delete
1.2.4 引用
1.2.5 对指针使用const限定符
1.2.6 泛型算法应用于普通数组
1.2.7 数据的简单输入输出格式

4、假设一维数组a和b的长度均为Len，数据类型为Type，则对数组内容的相关操作和语句如下：

①数据内容反转：

```
reverse (a, a + Len) ;    //数组元素反转排列
```

②复制数组内容：

```
copy (a, a + Len, b) ;    //将数组a的内容原样复制到数组b
```

```
reverse_copy (a, a + Len, b) ; //逆向复制数组a中内容到数组b
```

③数组升幂排序：

```
sort (a, a + Len) ;    //默认排序方式是升幂
```

## 泛型算法应用于普通数组

④数组降幂排序：

```
sort (b, b + Len, greater<Type> () ) ; //数组降幂排序
```

⑤检索查找数组内容：

```
find (a, a + Len, value) ; //查找数组a中是否存在值为value的元素
```

find函数返回的是位置指针，一般使用判别语句输出查找的内容，如：

```
Type *x = find (a, a + Len, value) ; //x是类型为type的指针
```

```
if (x == a + Len) cout<< “没有值为value的数组元素” ;
```

```
else cout<< “有值为value的数组元素” ;
```



## 泛型算法应用于普通数组

### ⑥输出数组的内容

`copy` (a, a + Len, ostream\_iterator<Type> (cout, "字符串" ) ) ;

可将ostream\_iterator简单理解为输出流操作符，<Type>表示数组元素的数据类型，本语句将数组内容按正向送往屏幕，输出方式是将每个元素与“字符串”的内容组合在一起连续输出。如果使用空格“ ”或换行符“\n”，可以按格式输出。也可将数组内容按逆向方式送往屏幕，语句为：

`reverse_copy` (a,a + Len,ostream\_iterator<Type> (cout, "字符串" ) ) ;



# 泛型算法应用于普通数组

1.2.1 使用函数重载
1.2.2 新的基本数据类型及其注意事项 L(l) F(f) O(0) x
1.2.3 动态分配内存 new delete
1.2.4 引用
1.2.5 对指针使用const限定符
1.2.6 泛型算法应用于普通数组
1.2.7 数据的简单输入输出格式

1.2 认识C++语言面向过程编程的特点

关键字

反转: reverse

复制: copy, reverse\_copy (逆向复制)

排序: sort (默认升幂, 尾部加greater<Type> () 为降幂)

检索: find

输出: copy (尾部必须加ostream\_iterator<Type> (cout, "字符串"))

# 数据的简单输入输出格式

1.2 认识C++语言面向过程编程的特点

- 1.2.1 使用函数重载
- 1.2.2 新的基本数据类型及其注意事项 L(l) F(f) O(0) x
- 1.2.3 动态分配内存 new delete
- 1.2.4 引用
- 1.2.5 对指针使用const限定符
- 1.2.6 泛型算法应用于普通数组
- 1.2.7 数据的简单输入输出格式

1、C++提供了两种格式控制方式，一种是使用iso\_base类提供的接口，另一种是使用一种称为操控符的特殊函数，操控符的特点是可直接包含在输入和输出表达式中，因此更为方便，不带形式参数的操控符定义在头文件<iostream>中，带形式参数的操控符定义在头文件<iomanip>中。

在使用操控符时，一是要正确包含它们，二是只有与符号“<<”或“>>”连接时才起作用，三是无参数的操控符函数不能带有“( )”号。



# 常用操控符及其作用

1.2 认识C++语言面向过程编程的特点

- 1.2.1 使用函数重载
- 1.2.2 新的基本数据类型及其注意事项 L(i) F(f) O Ox
- 1.2.3 动态分配内存 new delete
- 1.2.4 引用
- 1.2.5 对指针使用const限定符
- 1.2.6 泛型算法应用于普通数组
- 1.2.7 数据的简单输入输出格式

格式	含义	作用
dec	设置转换基数为十进制	输入/输出
oct	设置转换基数为八进制	输入/输出
hex	设置转换技术为十六进制	输入/输出
endl	输出一个换行符并刷新流	输出
setprecision (n)	设置浮点数输出精度n	输出
setw (n)	设置输出数据字段宽度	输出
setfill ('字符')	设置ch为填充字符	输出
setiosflags (flag)	设置flag指定的标志位	输出
resetiosflags (flag)	清除flag指定的标志位	输出

上表中操控符使用时，后四个操控符必须包含头文件<iomanip>，其中后两个操控符的参数flag是引用C++的类ios\_base里定义的枚举常量，要使用限定符“::”，下面的表中是几个常用的ios\_base定义的枚举常量，另外flag可由多个常量“或”起来使用，如：setiosflags (ios\_base::showpoint | ios\_base::fixed)。



# 参数flag常引用的枚举常量及其含义

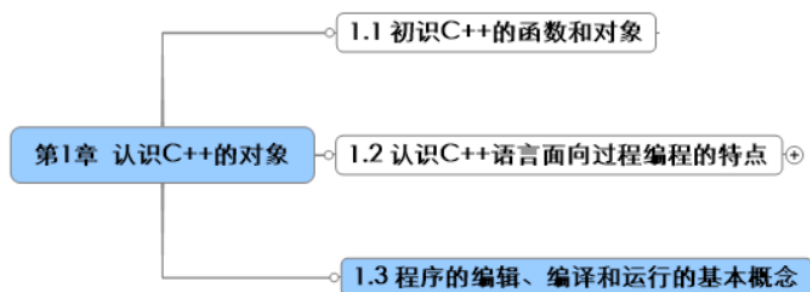
1.2 认识C++语言面向过程编程的特点

- 1.2.1 使用函数重载
- 1.2.2 新的基本数据类型及其注意事项 L(l) F(f) O 0x
- 1.2.3 动态分配内存 new delete
- 1.2.4 引用
- 1.2.5 对指针使用const限定符
- 1.2.6 泛型算法应用于普通数组
- 1.2.7 数据的简单输入输出格式

常量名	含义
ios_base::left	输出数据按输出域左边对齐输出
ios_base::right	输出数据按输出域右边对齐输出
ios_base::showpos	在正数前添加一个 “+” 号
ios_base::showpoint	浮点输出时必须带有一个小数点
ios_base::scientific	使用科学计数法表示浮点数
ios_base::fixed	使用定点形式表示浮点数



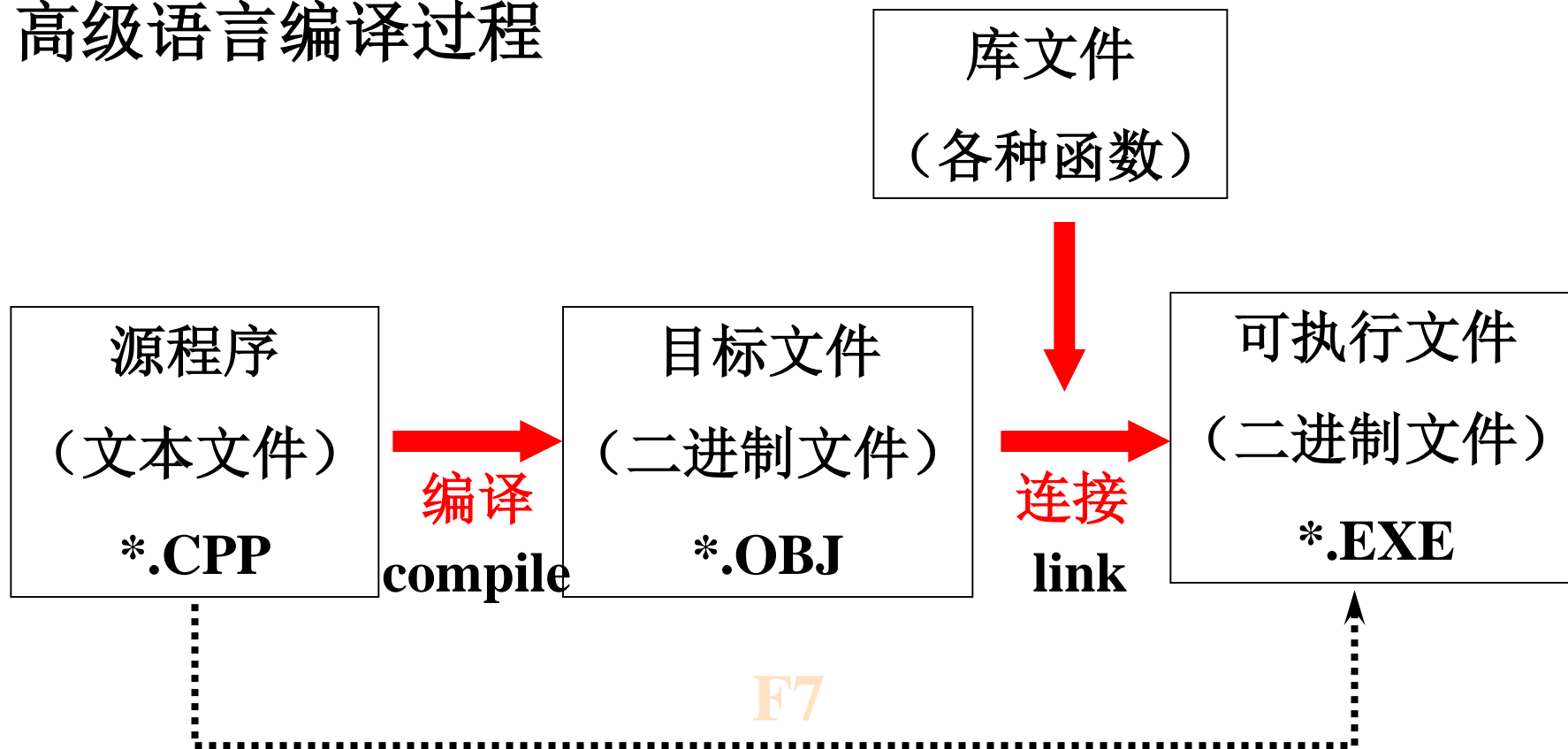
# §1.3 程序的编辑、编译和运行的基本概念



## 1、C + + 程序编制过程

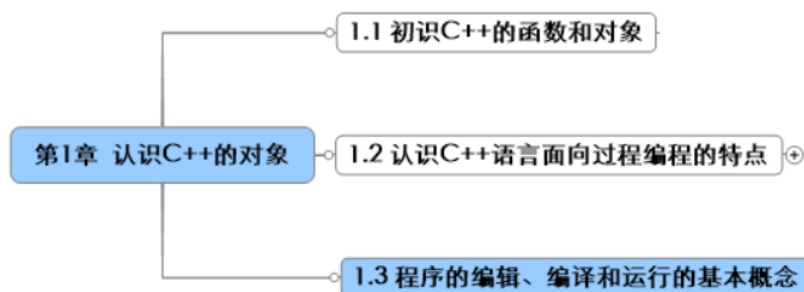
- ①先使用编辑器编辑一个C + + 程序mycpp.cpp, 又称其为C + + 的源程序;
- ②然后使用C + + 编译器对这个C + + 程序进行编译, 产生文件mycpp.obj;
- ③再使用连接程序 (又称Link) , 将mycpp.obj变成mycpp.exe文件。

# 高级语言编译过程



在Vitual C++系统中，可直接从源程序编译连接至可执行程序，但依然要生成\*.OBJ及\*.EXE这两个文件。

# ➡➡➡ C + + 程序编制环境及使用方法



现在C + + 的编制一般都使用集成环境，如Visual C + + 6.0等，所谓集成环境，就是将C + + 语言的编辑、编译、连接和运行程序都集成到一个综合环境中。

利用VC编制C + + 程序源文件的步骤如下：

①启动VC6.0;

②File菜单 - New对话框 - Project选项卡 - Win32 Console Application选项，在右边的Project name输入框中输入项目名称myfile，在右边的Location输入框中输入存储目录，然后单击OK按键，进入Win32 Console Application制作向导的第一步，编辑C + + 程序文件是选择An empty project选项，单击Finish按钮，完成设置；

# ➤➤➤ C + + 程序编制环境及使用方法

第1章 认识C++的对象

1.1 初识C++的函数和对象

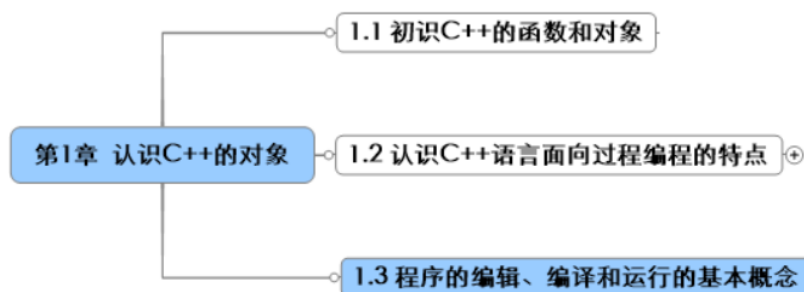
1.2 认识C++语言面向过程编程的特点

1.3 程序的编辑、编译和运行的基本概念

③选中File View选项卡，进入空项目，单击它展开树形结构，选中myfile files节点；选中Source File标记，再从File菜单中选new命令，弹出new对话框；选中C + + Source File选项，在右方的File输入框中输入C + + 程序文件名（mycpp），系统默认的文件扩展名为.cpp，单击OK按钮，返回集成环境，并在Source File项下面产生空文件mycpp.cpp；在右边的源代码编辑框中输入源文件。



# 部分Build菜单项描述



菜单项	描 述
<b>Compile Build</b>	编译源代码窗口中的活动源文件 查看工程中所有文件，并对最近修改过的文件进行编译和链接
<b>Rebuild All</b>	对工程中的所有文件全部进行重新编译和连接
<b>Clean</b>	删除项目的中间文件和输出文件
<b>Start Debug Execute</b>	弹出级联菜单，主要包括有关程序调试的选项 运行应用程序



# 本章总结





祝大家顺利通过考试!