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分×20题 = 20分

填空题　　1分×20题 = 20分

程序改错题　4分×5题 = 20分

完成程序题　4分×5题 = 20分

程序分析题　5分×2题 = 10分

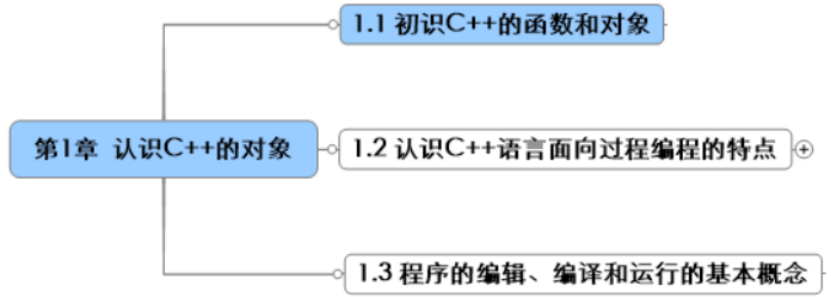
程序设计题 10分×1题 = 10分

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



1、混合型语言

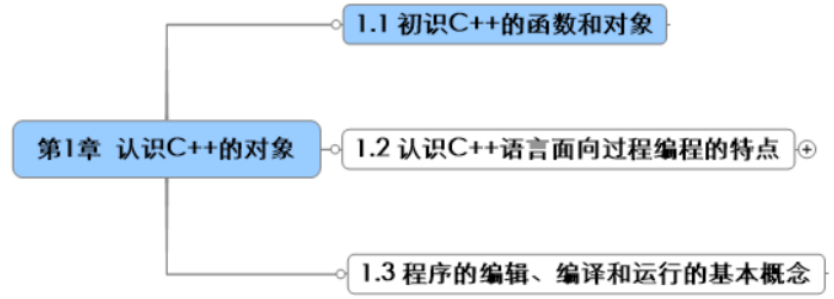
C++以**.cpp**为文件扩展名，有且只有一个名为main的主函数，因保留了这个面向过程的主函数，所以被称为混合型语言。

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

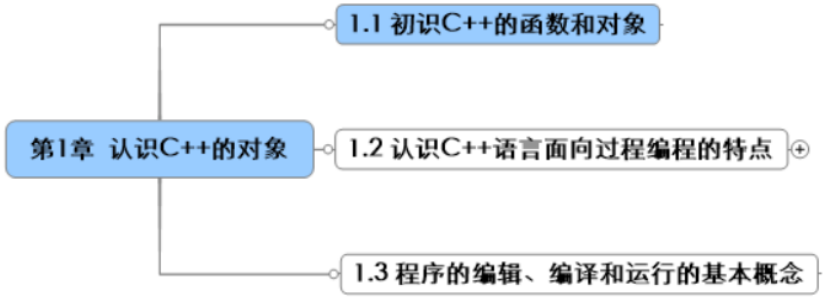
①提取操作：用提取操作符“>>”从**cin**输入流中提取字符，如：cin >> a.x ;

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

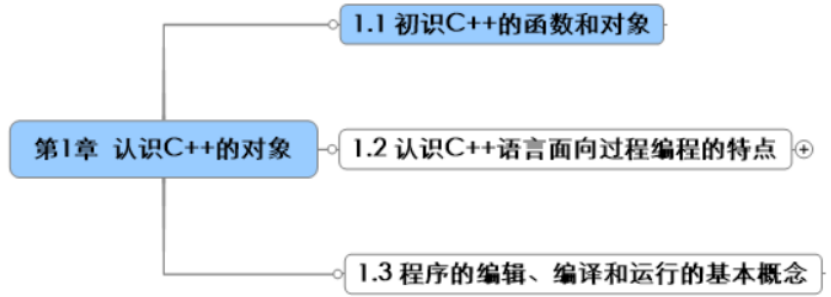
C＋＋相比C而言，**可以省略“ .h”标识头文件，但必须使用语句using namespace std;**来让命名空间中的对象名称曝光。因此一般的程序都要具有下面的两条语句：

```
#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;
```



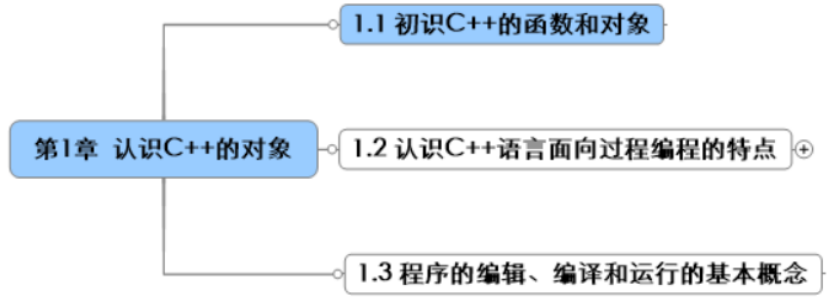
对象的定义和初始化

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

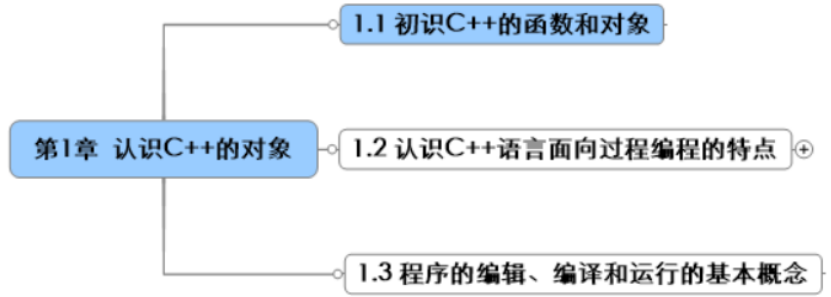


①C + + 使用变量和函数的基本规则都是：先声明，后使用。变量有时也可边声明边使用，但必须声明，否则出错。

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

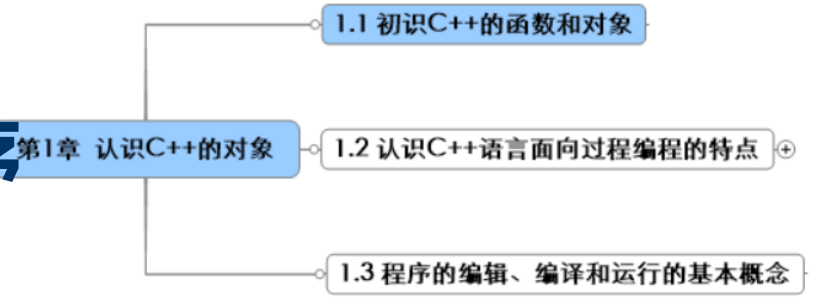


②除构造函数与析构函数外，函数都需要有类型声明。

- 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



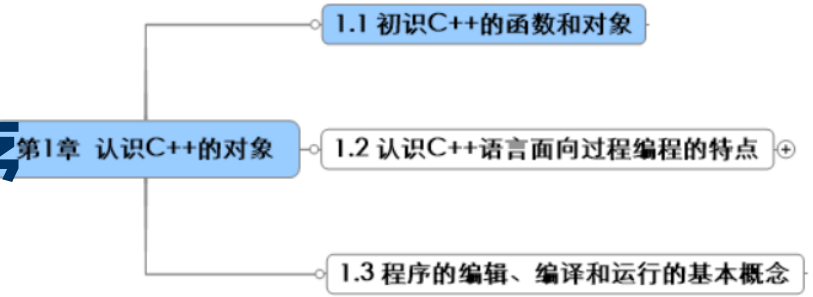
①const修饰符：用于定义符号常量。

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



- 常量定义也可使用构造函数的初始化方法，如：

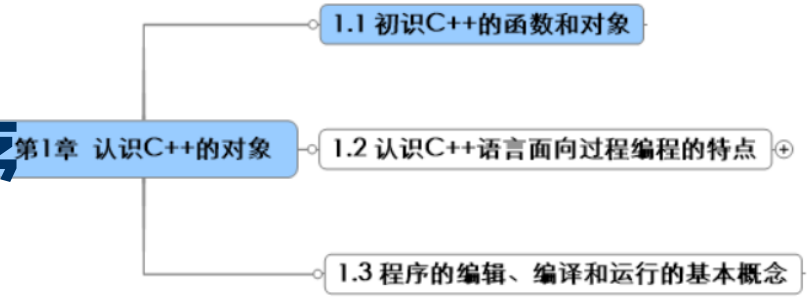
  const int k（2）；　　//等同于const int k＝2；

- 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

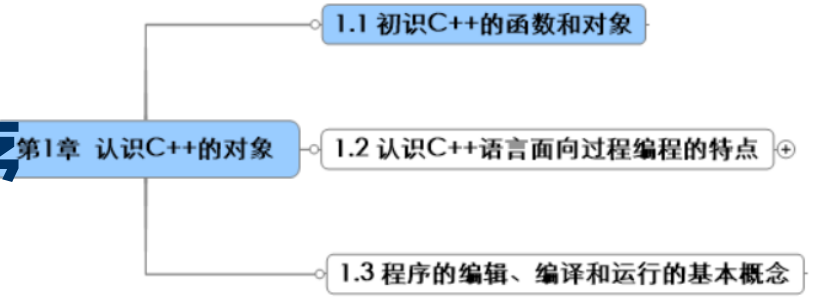


②预处理程序

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。**

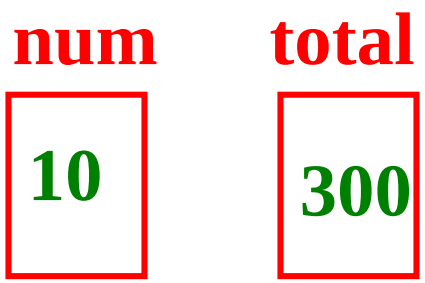
const（常量）修饰符及预处理程序



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
```

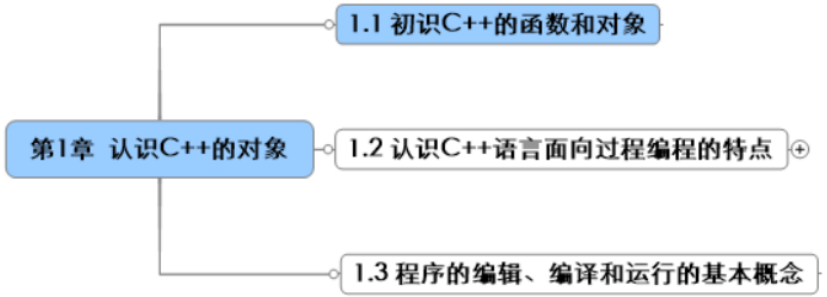
```
#include<iostream.h>
#define  PRICE   30  //常量，在程序中保持不变
void main(void)
{    int  num, total;  //定义变量,在内存中开辟区间
     num=10;        //变量赋值,10为常量
    total=num*PRICE;
    cout<<“total=“<<total; //输出结果
}
```



其中：num=10

total=num*PRICE

是赋值号，不同于数学意义上的等号。



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>

void  main(void )

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

}
```

包含文件

函数体开始

主函数

分号，一条完整语句的结束符

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
}
```

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。

§1.2 认识C++面向过程编译的特点

一、使用函数重载

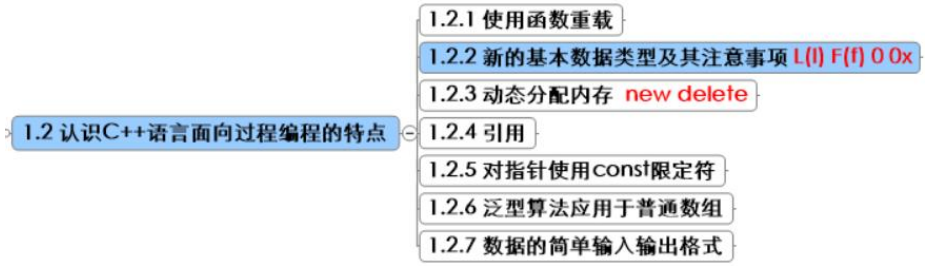
C++允许为同一个函数定义几个版本，从而使一个函数名具有多种功能，这称为**函数重载**。

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、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

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

5、C + + 中的常量分三种：

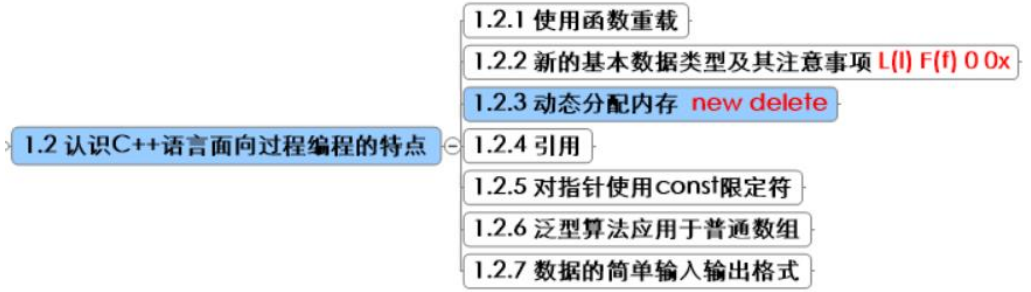
- 第一种为符号常量；
- 第二种为整数常量，有4种类型，分别为十进制、长整型（后缀L（l））、八进制(前缀0)、十六进制(前缀0x)，并用前缀和后缀进行分类标识；
- 第三种为浮点常量，有三种类型，分别为float型、long float型、double型，并用后缀进行分类识别。

新的基本数据类型及其注意事项

- 常量
  - 符号常量
  - 浮点常量
    - float 型：后缀 F（f），如 3.2f、0.002f、52.48f
    - long float 型：后缀 L（1），如 0.2L、4.5L、52.48L
    - double 型：无后缀，如 3.0、12.3、0.002
  - 整数常量
    - 长整型：后缀 L(1)，如-456L、0L、2145857852L
    - 十进制：无后缀，如-32、0、123
    - 八进制：前缀 0（o），如 0123、05、032737
    - 十六进制：前缀 0X（ox），如 0x15，0x1A，0x1f

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、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：

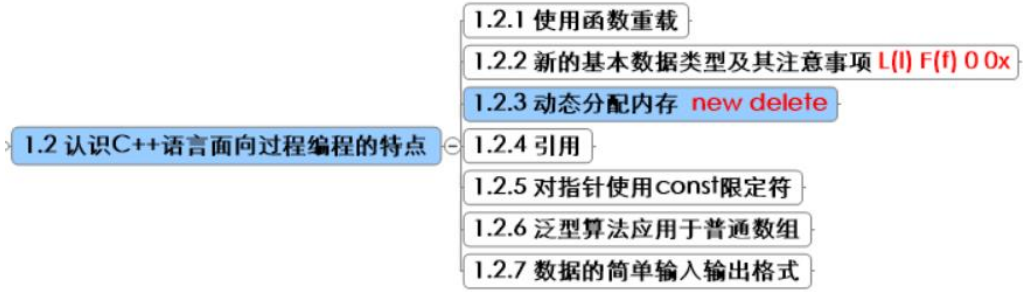
new 类型名[size]　　//申请可以存储size个该数据类型的对象

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



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

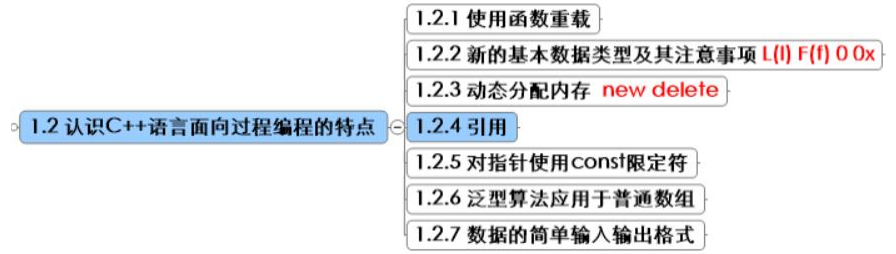
delete 指针名；　　//释放

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：

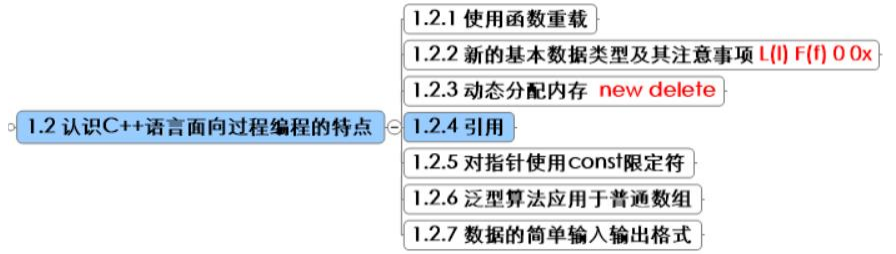
数据类型 & 别名 = 对象名;

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; √

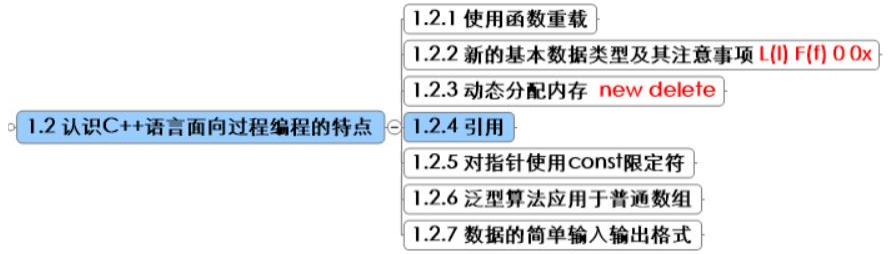
（式中p1、p2是指针, * 声明p2是指针，&声明p2是p1的引用）；

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; ×;

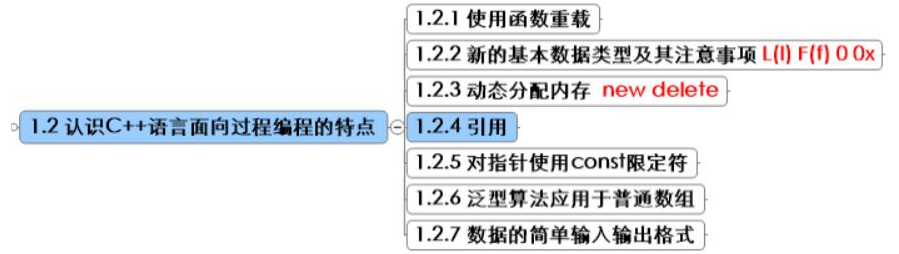
也不能直接声明对数组的引用。

引用



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、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

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)

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)

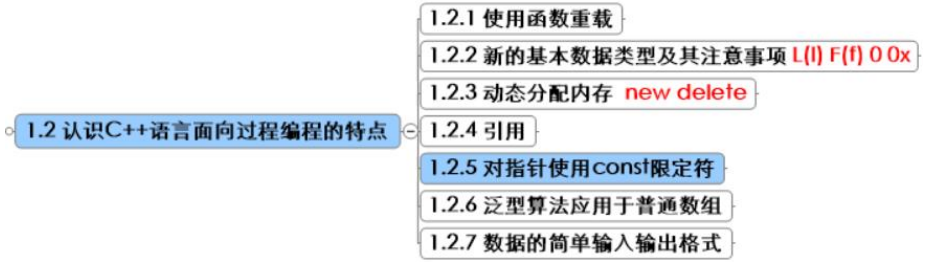
把const限定符放在*号的右边，就可使指针本身成为一个const，即常量指针。

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



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

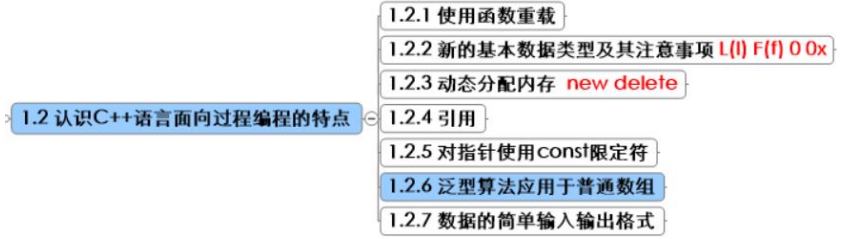
1、数组中元素及位置的关系

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 ~ 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×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。

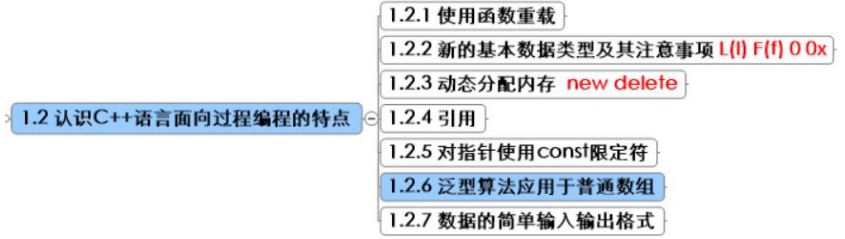
泛型算法应用于普通数组



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



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

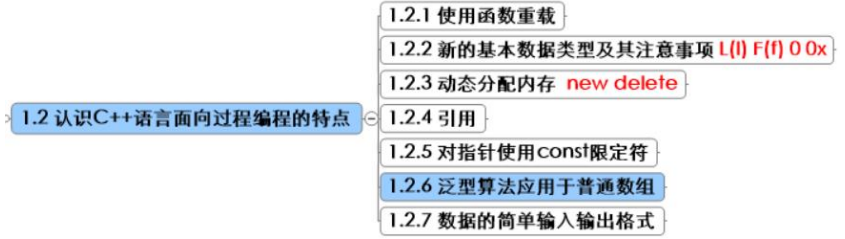
⑥输出数组的内容

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”））；

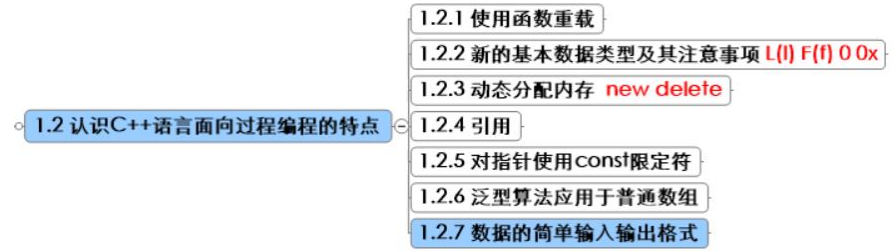
泛型算法应用于普通数组



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、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

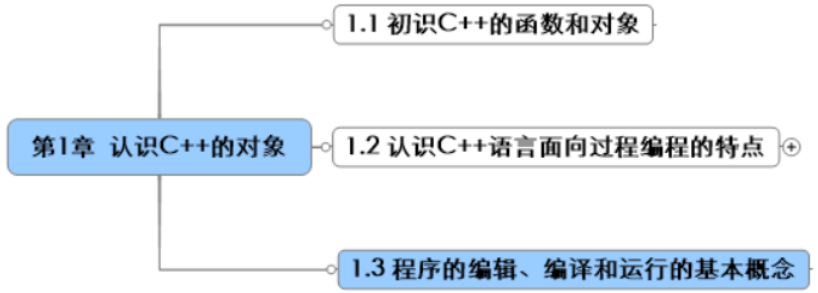
格式	含义	作用
dec	设置转换基数为十进制	输入/输出
oct	设置转换基数为八进制	输入/输出
hex	设置转换技术为十六进制	输入/输出
endl	输出一个换行符并刷新流	输出
setprecision（n）	设置浮点数输出精度n	输出
setw（n）	设置输出数据字段宽度	输出
setfill（'字符'）	设置ch为填充字符	输出
setiosflags（flag）	设置flag指定的标志位	输出
resetiosflags（flag）	清除flag指定的标志位	输出

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

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



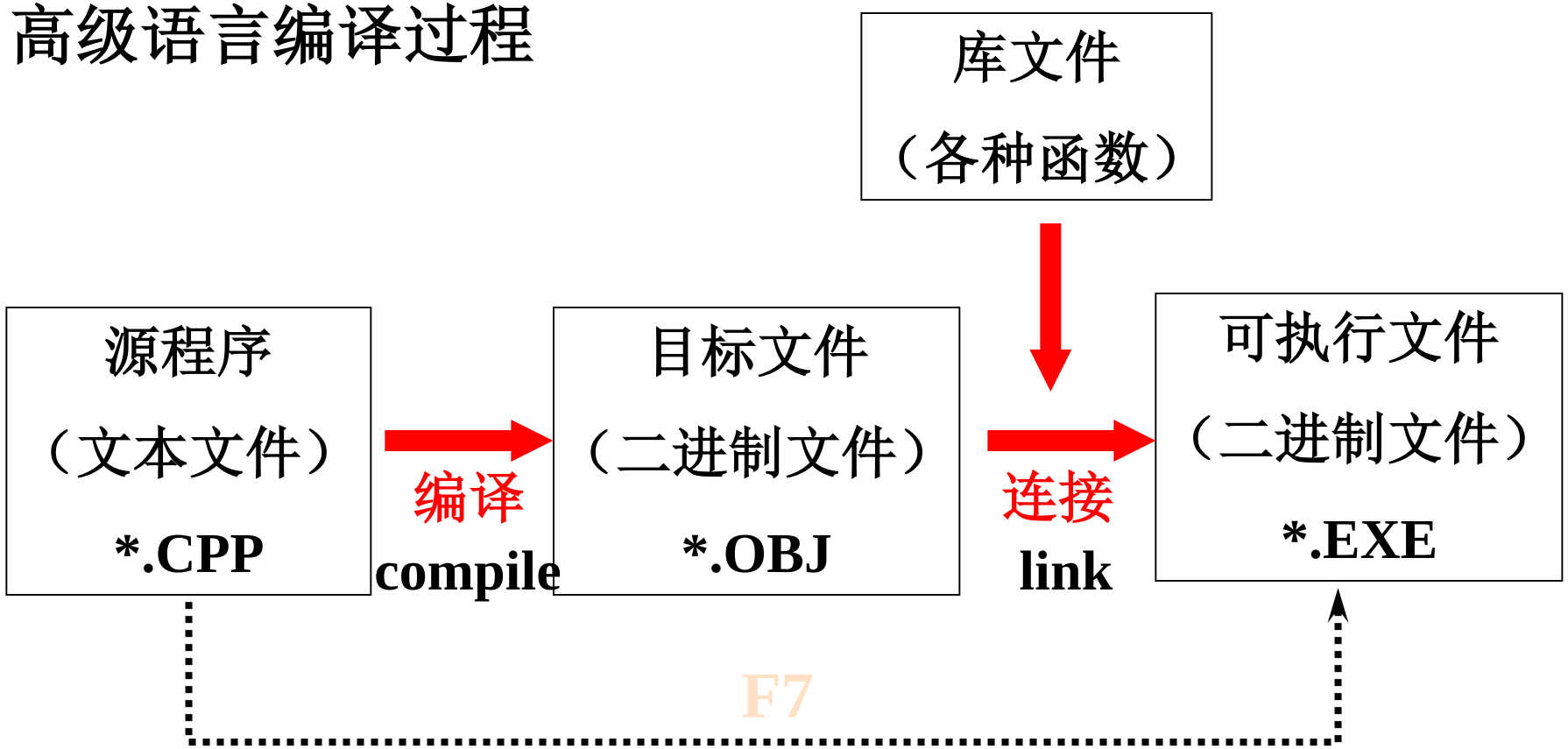
1、C + + 程序编制过程

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；

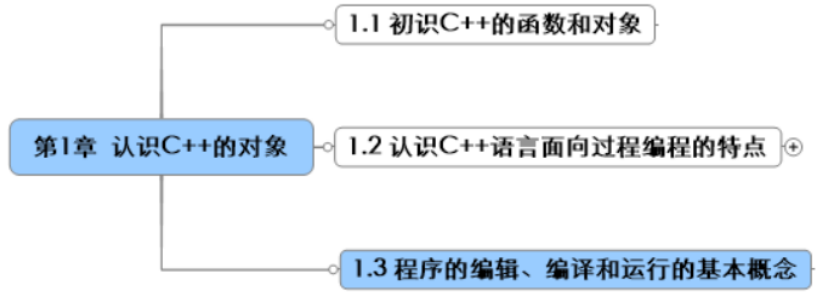
②然后使用C + + 编译器对这个C + + 程序进行编译，产生文件mycpp.obj；

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。



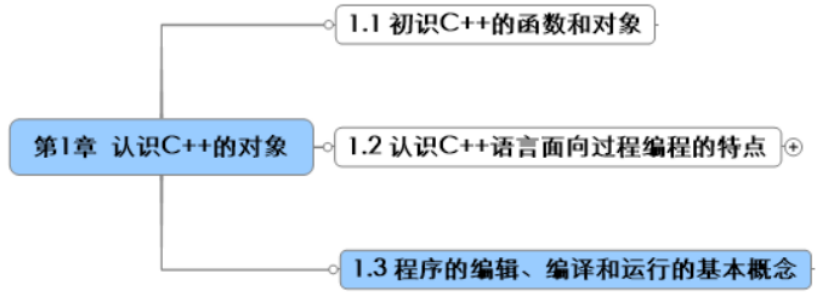
C++程序编制环境及使用方法

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：

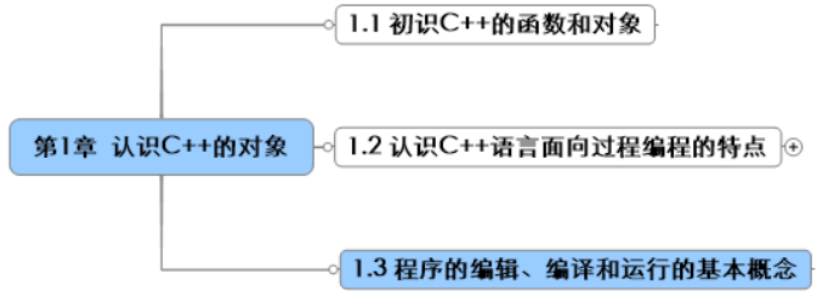
①启动VC6.0；

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

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
Compile	编译源代码窗口中的活动源文件
Build	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
Rebuild All	对工程中的所有文件全部进行重新编译和连接
Clean	删除项目的中间文件和输出文件
Start Debug	弹出级联菜单，主要包括有关程序调试的选项
Execute	运行应用程序

本章总结


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
1.2.1 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
1.3 程序的编辑、编译和运行的基本概念

祝大家顺利通过考试！