第2章 C++语言编程基础

学习 C++语言编程，首先需要掌握其基础知识。本章将详细介绍 C++语言基础，包括基本数据类型、变量和常量、运算符和表达式、流程控制语句，最后介绍简单的输入/输出方法。

2.1 C++语言数据类型

具有丰富的数据类型是 C++语言的特点之一，这使得 C++语言可以处理各种不同的数据。C++语言的数据类型可分为基本数据类型与构造数据类型。基本数据类型是指 C++语言已经预定义、不可进一步分割的数据类型，构造数据类型是指由一种或几种数据类型组合而成的数据类型。C++语言的数据类型如图 2-1 所示。

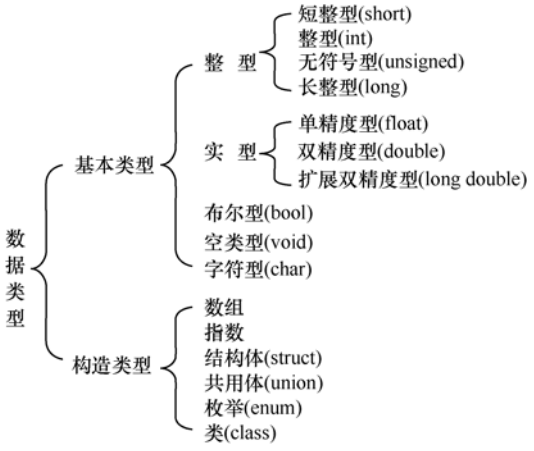

图 2-1 C++语言的数据类型

本章介绍基本数据类型，构造数据类型将在后面章节中分别介绍。

2.1.1 标识符

标识符是 C++语言中用来表示数据、变量、函数等实体名称的有效字符序列，这里的有效字符是指键盘上除“@”“`”“$”外的有可显示字符。C++语言标识符有两种：关键字和自定义标识符。

关键字也称保留字，是程序设计语言中约定已具有某种特定含义的标识符，不可以再作为其

他用途。表 2-1 列出了 C++语言中常用的关键字。

表 2-1　　C++语言中常用的关键字

int	double	if	for
char	float	else	while
void	const	switch	do
long	short	break	return
this	struct	continue	private
inline	case	union	protected
operator	default	enum	public
virtual	auto	class	friend
static	extern	signed	delete
register	typedef	unsigned	new

自定义标识符是指在编程中，用户根据需要为变量、函数等对象起的名称。作为名称的标识符必须由字母、数字、下画线组成，且不能以数字开头。下面列出的均为合法的自定义标识符：

```
Student_003,Float,str4,_345,year,name
```

下面是不合法的自定义标识符：

```
float,003_Student,-345,$Abs,C++,A.B,π
```

命名自定义标识符时要注意如下几点。

（1）通常使用具有某种含义的标识符，尽量做到“见名知义”。

（2）不可以将关键字作为自定义标识符。

（3）C++语言中严格区分大小写。例如，myName 与 myname 是两个不同的标识符。

2.1.2　基本数据类型

C++语言基本数据类型包括布尔型（bool）、整型（int）、单精度实型（float）、双精度实型（double）、字符型（char）、空类型（void）等，常用情况如表 2-2 所示。对于每种数据类型，C++没有统一规定精度、数据范围和在内存空间中所占字节数。

表 2-2　　C++语言常用的基本数据类型

名　称		类　型	长度（字节）	取 值 范 围
布尔型		bool	1	true 或 false
字符型		char	1	-128～127
整型		int	4	-2^{31}～$2^{31}-1$
实型	单精度	float	4	-10^{38}～10^{38}
	双精度	double	8	-10^{308}～10^{308}
空类型		void	0	无值

为了更准确地描述数据类型，C++语言还提供了 4 个关键字 short、long、unsigned 和 signed 用来修饰数据类型，相应的基本数据类型如表 2-3 所示。

表 2-3　　C++语言中经修饰的基本数据类型

名　　称	类　　型	长度（字节）	取 值 范 围
无符号字符型	unsigned char	1	0～255
短整型	short int	2	-32768～32737
长整型	long int	4	-2^{31}～$2^{31}-1$
无符号短整型	unsigned short int	2	0～65535
无符号长整型	unsigned long int	4	0～2^{32}-1
长双精度型	long double	8	-10^{308}～10^{308}

有关数据类型的几点说明如下。

（1）数据的“类型”是对数据的抽象，决定数据所占内存的字节数、数据的取值范围以及其上可进行的操作。程序中使用的所有数据都必定属于某一种数据类型。

（2）布尔型又称逻辑型，有真（true）和假（false）两种状态，用整型值 1（true）和 0（false）参与运算。

（3）整数类型的数据按二进制数形式存储，第一位为符号位，0 表示正数，1 表示负数，其余各位用于表示数值本身。无符号整型数据只能表示非负数，没有符号位。

（4）字符型数据按 ASCII 码存储，可以像整型数据一样参与运算。

（5）实型数据均为有符号数据，不可以用 unsigned 修饰。

（6）空类型又称无值型、空值型，意为“未知”类型，可以用于函数的形参、函数类型、指针等，但不能说明空类型的变量。

2.2　常　　量

常量是指在程序运行中，其值不能被改变的量，包括字面常量和符号常量两种。字面常量又包括整型常量、实型常量、字符型常量、字符串常量，在程序运行时，字面常量直接参与运算，不占用内存存储空间。

2.2.1　整型常量

C++中的整型常量有十进制（D）、八进制（Q）、十六进制（H）3 种形式。默认为十进制常量，八进制常量以 0 开头，十六进制常量以 0x 或 0X 开头。例如：

```
23,+74,-18,0123,-057,0,-0x5a,0X94BC
```

还可以加后缀 L（或 l）和 U（或 u）表示长整型数或无符号型整数常量，如 0X94BCL，23l，0123U，0u 等。

2.2.2　实型常量

实型常量又称浮点小数。在 C++中实型常量只使用十进制表示，有两种表示形式：一般形式和指数形式。

用一般形式表示时，整数和小数间用小数点隔开，加后缀 F（或 f）表示 float 型常量，默认

为 doubl 型常量，也可以用 L（或 l）表示长双精度型实型常量。例如：

```
16.3,-13.5f,.34, 62.,-76.43L,93.88F
```

指数形式表示由尾数和指数两部分组成，中间用 E（或 e）隔开，其中指数部分只能是十进制整数。例如，0.573E + 2 表示 0.573×10^2，-34.9E-3 表示-34.9×10^{-3}，1e4 表示 1×10^4，.3e1 表示 0.3×10^1。当以指数形式表示一个实数时，指数部分不可以省略，且应为整数，尾数的整数部分和小数部分可以省略其一，但不能都省略。例如，.263E-1 和 68.E2 都是正确的，而 E-6、5e、42e5.5 等都是错误的。

2.2.3 字符型常量

字符型常量简称字符常量，是用单引号括起来的一个字符，在内存中用 ASCII 码形式存储，占一字节，分普通字符型常量和转义字符型常量。如'a'、'8'、'?'、'$'、' '等都是普通字符型常量，其中最后一个为空格字符常量。

转义字符型常量用来表示一些不可显示，也无法通过键盘输入的特殊字符，如响铃、换行、制表符等。转义字符以“\”开头，后面跟表示特殊含义的字符序列。表 2-4 列出了常用的转义字符。

表 2-4　　常用的转义字符

字 符 形 式	含　义
'\n'	换行
'\a'	响铃
'\t'	水平制表符（横向跳格，相当于 Tab 键）
'\v'	竖向跳格
'\0'	空字符
'\\'	反斜杠\
'\''	单引号'
'\"'	双引号"
'\ddd'	1～3 位八进制数
'\xdd'	1～2 位十六进制数

表 2-4 中最后两行是所有字符的通用表示方法，即可用八进制数或十六进制数表示，其值转换成十进制数必须在 0～255 之间。例如，'\160'、'\x70'都表示字符'p'，这里的 160 是八进制数，x70 为十六进制数，转换成十进制数为 112，不超过 255。

2.2.4 字符串常量

字符串常量是用双引号括起来的若干字符，简称字符串。例如：

```
"We are studends. ","0234","-546.5UL","re\x70\t67\160\n0\0"
```

字符串中所含有效字符的个数称为字符串的长度，如字符串"I am a student."的长度为 15，字符串"0234"的长度为 4，字符串"-546.5UL"的长度为 8，字符串"re\x70\t67\160\n0\0"的长度为 9。

字符串常量在内存中是按顺序逐个存储串中字符的 ASCII 码，通常占用长度加 1 字节的内存

空间，这是因为 C++语言用特殊字符'\0'作为字符串结束符。'\0'为空字符，ASCII 码值为 0。例如，"A"占 2 字节，而'A'只占 1 字节。

2.2.5 符号常量

符号常量是一个标识符，对应一个存储空间，该空间中保存的数据就是该符号常量的值。例如，保留字中的 true 和 false 就是系统预先定义的两个符号常量，它们的值分别为 1 和 0。C++语言提供了两种声明符号常量的方法。

1. 用#define 声明符号常量

其一般格式如下。

```
#define 标识符 常量值
```

这里的标识符又称为宏名，用来表示常量名，常量值可以是上面介绍的各种类型。例如：

```
#define PI 3.1415926                    //用标识符 PI 表示数 3.1415926
#define SS "We are students."           //用标识符 SS 表示字符串"We are students."
```

2. 用 const 声明符号常量

其一般格式如下。

```
const 数据类型 常量名=常量值;
```

或

```
const 数据类型 常量名(常量值);
```

其中数据类型可以是任何一种 C++数据类型，且位置可以与保留字 const 互换，“=”称为赋值号，完成赋值操作。例如：

```
const float PI=3.1415926;               //定义实型常量 PI，其值为 3.1415926
char const c='\160';                    //定义字符型常量 c，其值为'p'
```

2.3 变　　量

在程序运行过程中值可以改变的量称为变量。一个变量有一个名称，其本质是在内存中分配一块存储空间，用以存储数据。

2.3.1 变量的定义

在 C++语言中，使用变量前必须先定义变量，告知系统需要分配的空间和确定该变量能进行的操作。定义变量的格式如下。

```
数据类型 变量名 1，变量名 2，…，变量名 n;
```

例如：

```
int x, y, z;                    //定义 3 个整型变量 x、y、z
float value;                    //定义一个实型变量 value
char s;                         //定义一个字符型变量 s
```

2.3.2 变量的初始化

在定义变量的同时给变量赋初值称为初始化，可以在定义变量时直接赋初值，其一般格式如下。

数据类型 变量名=常量值;

或

数据类型 变量名(常量值);

也可以定义变量后再用赋值号给变量赋值。例如：

```
int  x=1,y,z(3);        //定义 3 个整型变量 x、y、z，并给 x 和 z 赋初始值
y=4;                    //给变量 y 赋值
z=x;                    //将变量 x 的值赋给变量 z，此时 z 的值与 x 的值相同
```

2.3.3 指针变量

C++语言源程序经过编译系统处理后，每一个变量在程序执行前将分配在内存指定的位置上，程序执行时，按变量名对应的内存地址处理相应的数据，这种按变量的地址直接存取变量的方法称为“直接访问”方式。存储变量的内存空间的首地址称为该变量的地址。

如果将一个变量的地址放在另一个变量中，则存放地址的变量称为指针型变量，简称指针变量。这时存取变量，也可以间接地由指针变量取得该变量的地址进行，称为“间接访问”方式。定义指针变量的一般格式如下。

数据类型 *变量名 1, *变量名 2, …,*变量名 n;

在这里，“*”号是一个标志，表示定义的变量是一个指针变量，以表示与一般变量的区别。例如：

```
int  *p,*q;             //定义两个整型指针变量 p 和 q
float  *s;              //定义一个实型指针变量 s
```

由于指针变量中的值是另一个变量的地址，习惯上形象地称为指针变量指向某变量。指针变量中的值也简称为指针，所以指针就是地址。例如：

```
char  a='5';
char  *r=&a;
```

这里“&”为取地址运算符，表示将字符变量 a 的地址赋给字符指针变量 r，也称 r 指向 a。

指针变量中存放的是某种类型数据的地址，C++语言中任何类型的地址均占 4 字节。需要注意的是，指针变量的类型与其指向的数据的类型必须相同。

如果将一个指针变量的地址再放在另一个变量中，则该变量称为二级指针变量。定义二级指针变量的一般格式如下。

数据类型 **变量名 1, **变量名 2, …, **变量名 n;

在这里，“**”号同样起到标志的作用。例如：

```
char  a='5';
char  *r=&a;            //r 为一级指针
char  **q=&r;           //定义二级指针变量 q,指向指针变量 r
```

2.3.4　引用变量

在程序编写过程中，有时需要为某个变量起多个名称，此时可采用引用型变量，简称引用变量。引用就是给变量起个别名。定义引用变量的一般格式如下。

```
数据类型　&引用名=变量名;
```

其中，“&”号标志是一个引用变量。例如：

```
int  solution;
int  &result=solution;          // 将 result 定义为 solution 的别名
result=5;                       //将 result 赋值为 5，solution 的值也是 5
```

定义引用变量时，必须对其初始化，且变量名必须是已经定义的，并与引用的类型相同，即赋值号以及后面部分不可缺少，以说明是为哪个变量起别名。

2.4　C++语言的基本语句

一个 C++语言的程序可由若干源程序文件组成，一个源程序文件可由若干函数组成，一个函数可由若干条语句组成。语句是 C++语言程序中最小的独立单位，相当于自然语言一篇文章中的一个句子。按照其功能，语句可以分为：用于描述计算机要执行操作运算的语句和控制上述操作运算执行顺序的语句两类，前一类称为操作运算语句，后一类称为流程控制语句。具体地，C++语言的语句有声明语句、表达式语句、空语句、复合语句几种。在 C++语言中使用分号表示一条语句结束。

2.4.1　声明语句

声明语句是指对某种类型的变量、函数原型、结构、类等的说明。例如：

```
int  a=2;                       //声明一个变量
void  fun(int x,float y);       //声明一个函数
```

2.4.2　表达式语句

在表达式后加上分号，就构成了一条表达式语句。它的作用是执行表达式的计算。例如：

```
a=a*2;                          //算术表达式语句
a++;                            //后置自增表达式语句
```

2.4.3　空语句

仅由分号组成的语句称为空语句，它不执行任何动作，通常用在语法上需要语句，但又没有任何操作要做的地方。

2.4.4　复合语句

复合语句又称块语句，是用一对花括号将一条或多条语句括起来组成的。例如：

```
{
  char ch='3';
  cout<<ch<<'\n';
}
```

在复合语句中声明的变量，只在复合语句内部有效。复合语句在结构上被看成是单条语句，用在需要用多条语句描述某个问题，但语法上只能是一条语句的地方。

2.4.5 基本输入/输出语句

数据的输入/输出在程序设计中是必不可少的，在 C++语言中，数据的输入/输出都由预定义的库函数或对象来完成。在 C++语言标准类 iostream 中包含标准输入流对象 cin 和标准输出流对象 cout，分别用来实现从键盘上读取数据，以及将数据在屏幕上输出，当使用 C++语言的标准输入/输出时，程序应包含头文件 iostream。

1. C++的输入

C++的输入是由 cin 配合使用提取操作符“>>”实现的。一般格式如下。

```
cin>>变量 1>>变量 2>>…>>变量 n;
```

例如：

```
int  x;                                  //定义整型变量 x
float  y;                                //定义实型变量 y
char  c;                                 //定义字符型变量 c
cin>>x>>y>>c;      //依次输入一个整型、实型和字符型数据，分别存入变量 x、y、c 中
```

输入数据时用空格、水平制表符（Tab）或回车符作为分隔，最后以回车符确认。若执行上述语句时输入以下数据（其中↙表示输入回车符）：

```
8   12.5   g↙
```

则变量 x 的值为整数 8，变量 y 的值为实数 12.5，变量 c 的值为字符'g'。

输入数据的类型应该与变量的类型一致，否则变量赋值会出现异常。如果执行上述语句时输入以下数据：

```
12.5   8   g↙
```

则变量 x 的值为整数 12，变量 y 的值为实数 0.5，变量 c 的值为字符'8'。

要注意的是空格和回车符也是字符，但用 cin 输入时却不能接收它们，此时可用函数 cin.get 来实现输入。例如：

```
char  ch;          //定义字符型变量 ch
cin.get(ch);       //输入一个字符（可以是空格或回车），存放到变量 ch 中
```

2. C++的输出

C++的输出由 cout 配合使用插入操作符“<<”进行。其一般格式如下。

```
cout<<表达式 1<<表达式 2<<…<<表达式 n;
```

例如：

```
int  x=1, y, z(3);
float  value=3.71;
cout<<"x="<<x<<'\n';                          //输出：x=1
```

```
cout<<"value"<<value<<'\t'<<"z"<<z<<endl;   //输出：value=3.71   z=3
```

双引号中的字符串数据按原样输出，不加引号的 z 是变量，程序根据运行时变量的取值情况输出结果。endl 的作用同'\n'，用来控制输出格式，表示换行。

【例 2-1】 编写一个简单程序，计算键盘输入的两个整数的平均值。

程序设计

计算机解决问题时常用变量代表要计算的数据。在 C++程序设计语言中，变量是需要先定义后使用的。同时，C++程序设计语言中的变量区分类型。例如，整型变量只能代表整型数据而不能代表实型数据。本例设计两个整型变量 a 和 b，存放从键盘输入的两个整数，另外设计实型变量 c 存放它们的和。

源程序代码

```
#include<iostream>
using namespace std;
int main(void) {
     int  a, b;            // 定义整型变量 a 和 b
     float  c;             // 定义整型变量 c
     cout<<"请输入两个整数：";
     cin>>a>>b;
     c=(a+b)/2.0;
     cout<<"c="<<c<<endl;
     return 0;
}
```

程序运行结果（带下画线的数据表示键盘输入，↙表示输入回车符）

请输入两个整数：1　2↙

c=1.5

2.5　运算符与表达式

在 C++语言中，对常量或变量进行运算或处理的符号称为运算符，参与运算的变量或常量对象称为操作数。常用的运算符包括算术运算符、关系运算符、逻辑运算符、赋值运算符等，将变量、常量和运算符按语法规则结合起来就组成了表达式，如算术表达式、关系表达式、逻辑表达式、赋值表达式等。由多种运算符连接起来的式子称为混合表达式。

运算符具有优先级与结合性。当一个表达式包含多个运算符时，先进行优先级高的运算，再进行优先级低的运算。如果表达式中出现了多个相同优先级的运算，运算顺序就要看运算符的结合性了。所谓结合性，是指当一个操作数左右两边的运算符优先级相同时，按什么样的顺序运算，是自左向右，还是自右向左。表达式运算时，应注意每种运算符的优先级和结合性，表 2-5 列出了 C++语言中的运算符及其优先级和结合性。

表 2-5　　C++语言中运算符的优先级和结合性

优　先　级	运　算　符	结　合　性
1	::、( )、[ ]、.、->、&、++、−−	从左向右
2	!、++、−−、−（负）、+（正）、（类型）、*、&、sizeof、new、delete	从右向左

续表

优 先 级	运 算 符	结 合 性
3	*、/、%	从左向右
4	+（加）、−（减）	从左向右
5	<<、>>	从左向右
6	<、 <= 、> 、>=	从左向右
7	==、!=	从左向右
8	&（位运算）	从左向右
9	^（位运算）	从左向右
10	|（位运算）	从左向右
11	&&	从左向右
12	||	从左向右
13	? :	从右向左
14	=、+=、−=、*=、/=、%=	从右向左
15	,	从左向右

其中优先级中数字越小，优先级越高。优先级为 5、8、9、10 中的“<<”“>>”“&”“^”“|”在这里不做介绍。

2.5.1 算术运算符与算术表达式

C++语言的算术运算符包括基本算术运算符：加或正值（+）、减或负值（−）、乘（*）、除（/）、取余（%），以及自增（++）、自减（−−）运算符，除正值、负值及自增、自减运算符外，均为双目运算符。含算术运算符的表达式称算术表达式。C++语言算术运算符的优先级、目数及结合性见表 2-6。

表 2-6 C++语言算术运算符的优先级、目数和结合性

优 先 级	运 算 符	目 数	结 合 性
1	++（自增）、−−（自减）	单目	从右向左
2	−（负）、+（正）	单目	从右向左
3	*（乘）、/（除）、%（取余）	双目	从左向右
4	+（加）、−（减）	双目	从左向右

其中：

（1）除法运算分母不能为零，且当两个操作数均为整数时，除法运算后将舍去小数部分只取整数。例如，5/4 的结果为 1。

（2）取余运算也称求模运算，当两个操作数均为整数，且运算符右边的数不为 0 时，可进行取余运算，结果为两个整数相除后的余数。如果两个整数中有负数，则先用两数绝对值求余，最后结果的符号与被除数相同。例如，6%7 为 6，−7%6 为−1，7%−6 为 1。

（3）自增（++）和自减（−−）运算符是单目运算符，++和−−运算根据运算符的位置不同

分前置和后置两种。无论前置或后置，它们的作用都是将操作数的值增 1（减 1）后，重新写回该操作数在内存中的原有位置。所以，如果变量 i 原来的值是 1，计算表达式 i++后，表达式的结果为 2，并且 i 的值也被改变为 2。但是，当自增、自减运算的结果要用于继续参与其他操作时，前置与后置时的情况就完全不同了。所谓前置自增，是指先将变量值自增后再参与表达式的运算，后置自增是指先参与表达式的运算后，变量值再自增。自减运算的含义类似。例如：

```
int x=3, y=3;
cout<<++x<<endl;              // x 的值为 4，输出：4
cout<<y--<<endl;              // y 的值为 2，输出：3
```

（4）在 C++语言中，算术运算应注意数据溢出问题，即运算结果超出对应数据类型的表示范围。编译程序只会对除法运算时除数为 0 的情况提示出错，而对于特别容易溢出的整数的加、减和乘法运算产生溢出的情况，系统不作为错误处理，程序将继续执行并产生错误的计算结果。因此，必须在程序中尽量避免整数溢出问题。

【例 2-2】 写出下列各语句的输出结果。

```
int  x=6;
cout<<-x<<'\t';                        //A
cout<<(1+'a') <<'\t';                  //B
cout<<(5/3-8) <<'\t';                  //C
cout<<(5%3*5/3) <<'\n';                //D
```

输出结果

```
-6      98      -7      3
```

在例 2-2 中，整型变量 x 的值为 6，A 行输出-x，故输出值为-6。由于字符 a 的 ASCII 码值为 97，故 B 行输出结果是 98。在 C 行中，5/3 的值为 1，故输出结果为-7。在 D 行中，5%3 的值为 2，乘以 5 后值为 10，10/3 的值为整数，故输出结果为 3。

2.5.2　赋值运算符与赋值表达式

C++语言中的赋值通过赋值运算符（=）来完成，赋值运算符为双目运算符。带有赋值运算符的表达式称为赋值表达式。其一般格式如下。

变量名=表达式

赋值表达式的意义是将赋值号右边的值送到左边变量对应的单元中，“=”号左边只能是变量名，而不能是常量或表达式。例如：

```
int a,b;
a=5;                           //将 5 赋给变量 a
b=a=a-2;                       //将 a 的值减 2 后重新赋给 a，再将 a 赋给 b
cout<<b;                       //输出：3
8=b;                           //错误。左值不能是常数
a+b=5;                         //错误。左值不能是表达式
```

在 C++语言中，凡是多于一个符号的运算符均称为复合运算符，复合运算符是一个整体，中间不能用空格隔开。C++语言中的赋值运算符除了“=”外，还有其他一些复合赋值运算符，部分复合赋值运算符见表 2-7。

表 2-7 C++语言中的部分复合赋值运算符

优先级	运算符	复合赋值运算符表达式	一般表达式	结合性
14	+=（加等于）	x+=a	x=x+a	从右向左
14	-=（减等于）	x-=a	x=x-a	从右向左
14	*=（乘等于）	x*=a	x=x*a	从右向左
14	/=（除等于）	x/=a	x=x/a	从右向左
14	%=（模等于）	x%=a	x=x%a	从右向左

复合赋值运算符的功能是将运算符右边的值与左边变量的值进行相应的算术运算后，再将运算结果赋给左边的变量，必须遵循赋值运算与算术运算双重规则的制约。例如：

```
int  a=6, b=1, c=8;
b*=a+2;                          //相当于 b=b*(a+2)，b 的值为 8
a/= -c%3;                        //相当于 a=a/(-c%3)，a 的值为-3
```

而“c-1+=a/4;”是一个错误的表达式，因为复合赋值运算符的左边不是一个变量。

2.5.3 关系运算符和关系表达式

C++语言中的关系运算符都是双目运算符，包括大于（>）、大于等于（>=）、小于（<）、小于等于（<=）、等于（==）和不等于（!=），见表 2-8。

表 2-8 C++语言中部分关系运算符的优先级及其结合性

优 先 级	运 算 符	结 合 性
6	>（大于）、>=（大于等于）、<（小于）、<=（小于等于）	从左向右
7	==（等于）、!=（不等于）	从左向右

用关系运算符将两个表达式连接起来的式子，称为关系表达式。例如，4>5、5!=4、8>2、(5!=4)==(8>=2)都是关系表达式。

关系运算符完成两个操作数大小的比较，结果为逻辑值真（true）或假（false）。因为在 C++中，这两个逻辑值与整数之间有对应关系，真对应 1，假对应 0。所以关系运算的结果可以作为整数参与算术运算、关系运算、逻辑运算及其他运算。例如：

```
int  a=4>5;                      //相当于“int  a=(4>5);”，a 的值为 0
int  b=(5!=4)==(8>=2);           //相当于“int  b=((5!=4)==(8>=2));”，b 的值为 1
```

要特别注意等于运算符（==）与赋值运算符（=）的区别。当比较两个表达式的值时，要用等于运算符（==），而不能用赋值运算符（=）。例如：

```
int  a=3==8, b=4,c;              //相当于“int  a=(3==8), b=4;”，a 的值为 0，b 的值为 4
c=a+4==b;                        //相当于“c=(a+4==b);”，c 的值为 1
```

2.5.4 逻辑运算符和逻辑表达式

逻辑运算符包括逻辑非（!）、逻辑与（&&）和逻辑或（||）。其中!是单目运算符，&&和||是双目运算符。在逻辑运算中，所有非零值都表示逻辑真（true），0 表示逻辑假（false）。逻辑运算

符及其逻辑运算的真值分别见表 2-9 和表 2-10。

表 2-9　　C++逻辑运算符的优先级及其目数、结合性

优　先　级	运　算　符	目　　数	结　合　性
2	!（逻辑非）	单目	从右向左
11	&&（逻辑与）	双目	从左向右
12	||（逻辑或）	双目	从左向右

表 2-10　　C++逻辑运算真值表

a	b	!a	!b	a&&b	a||b
非 0	非 0	0	0	1	1
非 0	0	0	1	0	1
0	非 0	1	0	0	1
0	0	1	1	0	0

逻辑表达式是运算符为逻辑运算符的表达式，其结果只能为 true（1）或 false（0）。例如：

```
5&&'A'                    //值为真
(3<9) && (2==1)           //值为假
(3<9) ||(2==1)            //值为真
(3>9) ||(2==1)            //值为假
```

值得注意的是，C++对逻辑运算进行了优化，即一旦逻辑表达式的值能够确定，运算将不再继续进行。

【例 2-3】 设已定义变量 a 和 b，其值分别为 4 和 7，写出下列各语句的输出结果，并说明语句执行后，变量 a 和 b 的值。

```
cout<<((b=5)||(a=6))<<endl;          //A
cout<<(a=(a-4)&&(b=1))<<endl;        //B
```

在例 2-3 的 A 行，先执行 b=5，b 的值变为 5，为真，此时 a=6 将不执行，因此 a 的值仍为 4。输出结果为 1。在 B 行中，先进行（a-4）&&（b=1）的运算，由于 a-4 的值为 0，进行&&运算时，与操作数（b=1）无关，因而（a-4）&&（b=1）的值为 0，故 a 的值为 0，所以输出结果为 0。此时因为 b=1 同样不执行，所以 b 的值还是 5。

2.5.5　其他运算符及表达式

除了上述的运算符外，C++语言还提供了其他多种运算符，如求字节大小运算符（sizeof）、条件运算符（?:）、地址运算符（&）、指针运算符（*），以及逗号运算符（,）等。由这些运算符与操作数连接起来的式子构成相应的表达式。

1. sizeof 运算符

C++中提供的 sizeof 运算符是单目运算符，优先级较高，用来确定某种数据类型所占的空间大小。其一般格式如下。

```
sizeof(类型名)
```

或

sizeof(表达式)

例如：

```
cout<<sizeof(char);            //输出 1
cout<<sizeof('A'+5);           //输出 4
cout<<sizeof(4.0+2);           //输出 8
```

2. 条件运算符

条件运算符（?:）是 C++语言中唯一的三目运算符，优先级较低，仅高于赋值运算符和逗号运算符。其一般格式如下。

表达式 1? 表达式 2: 表达式 3

当表达式 1 的值为真时，整个表达式的值为表达式 2 的值，表达式 3 不运算；否则运算结果为表达式 3 的值，表达式 2 不运算。例如：

```
int  x=1, y(3), z;
z=x>y?++x:y++;
cout<<x<<'\t'<<y<<'\t'<<z<<endl;            //输出: 1  4  3
```

3. &和*运算符

&和*运算符均为单目运算符，优先级较高。其中&运算符为地址运算符，其作用是返回变量的地址值；*运算符为指针运算符（也称间接访问运算符），其作用是求指针变量所指内存空间的值。例如：

```
char  ch='5',*r;                            //A
r=&ch;                                      //B
cout<<*r;                                   //C
*r='7';                                     //D
```

在 A 行，*表示 r 是一个指针变量；B 行将 ch 的地址赋给 r；C 行与 D 行中的*r 为指针变量 r 所指变量 ch 的值，C 行输出字符'5'，D 行相当于 ch='7'。

注意 A 行与 C 行中*的区别，以及 B 行中&与引用的区别。

4. 逗号运算符

逗号运算符（,）又称顺序求值运算符，逗号表达式的一般格式如下。

表达式 1, 表达式 2, …, 表达式 n

其含义为依次从左到右运算，并将最后一个表达式的值作为整个逗号表达式的值。例如：

```
d=(x=1,3+x,++x);              //d 的值为 2
```

2.5.6 表达式中数据类型的转换

每个表达式都有确定的结果和确定的类型（结果的类型）。计算表达式的值不仅要考虑构成表达式的运算符的目数、优先级和结合性，还要考虑操作数类型的转换。因为当表达式中多种类型的数据进行混合运算时，首先要进行类型转换。C++的类型转换有自动类型转换和强制类型转换两种。

1. 自动类型转换

自动类型转换又称隐式类型转换。在双目运算中，如果两个操作数的类型不一致，则自动进行类型转换。转换的基本原则是将精度较低的类型向精度较高的类型转换。具体的转换顺序如下。

char→short→int→long→float→double

另外，字符型数据参与运算是用它的 ASCII 码进行的，因而会自动转换成整型数据；实型数据参与运算时会自动转换成双精度型数据。

进行赋值运算时，若左右两边的类型不一致，则将右边操作数转换成左边变量的类型。例如：

```
int  a=1;
float  x=3.5;
a=x;                                    //a 的值为 3
cout<<'F'-'B'<<endl;                    //输出整数 4
cout<<x+2<<endl;                        //输出双精度型数 5.5
cout<<(a*6+x/2-'1')<<endl;              //输出双精度型数-29.25
```

在执行上述语句时，变量 x 的值不改变。

2. 强制类型转换

强制类型转换也称显式类型转换，是指将一个表达式强制转换为某个指定类型。其一般格式如下。

（数据类型名）表达式

或

数据类型名（表达式）

例如：

```
cout<<(int)3.5;                         //输出整数 3
cout<<2/(float)3;                       //输出 0.666667
```

2.5.7　表达式的格式

C++语言中表达式的书写格式不同于一般的数学表达式，除了必须用 C++语言的合法运算符外，表达式中所有的符号必须在同一行上，且表达式中只能用圆括号来指定运算次序。C++表达式与数学表达式的对比情况见表 2-11。

表 2-11　　C++语言表达式与数学表达式的对比

数学表达式	C++表达式
$\sqrt{b^2-4ac}$	sqrt(b*b-4*a*c)
$\ln x+10^{-5}$	log(x)+1E-5
$'a'\leqslant x\leqslant'z'$	'a'<=x&&x<='z'
$\frac{x+2}{\lvert y\rvert}$	(x+2)/fabs(y)

2.6　程序的基本控制结构

程序的控制结构是控制程序中语句执行顺序的。任何程序都可以分解成 3 种基本控制结构，分别是顺序结构、选择结构和循环结构。每一种基本结构都由若干模块组成。流程控制语句的分

类如图 2-2 所示。

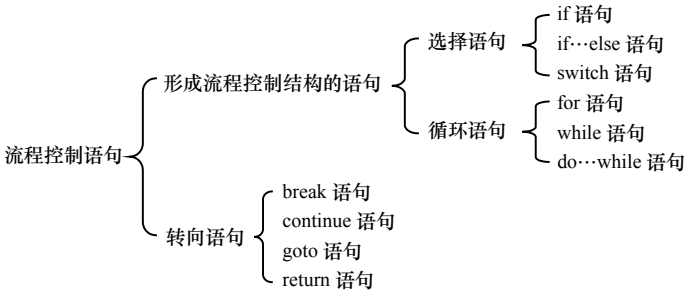

图 2-2　流程控制语句的分类

2.6.1　顺序结构

顺序结构是指程序执行时，按语句块编写顺序从上到下依次执行的结构。例如：

```
int  x=6, y, z;
y=4;
z=x+y;
cout<<z;
```

顺序结构可以用来解决一些简单的问题，其流程如图 2-3 所示。图 2-3 中 a 为程序段的入口，A、B 为实现某种操作的功能块，b 为程序段的出口。

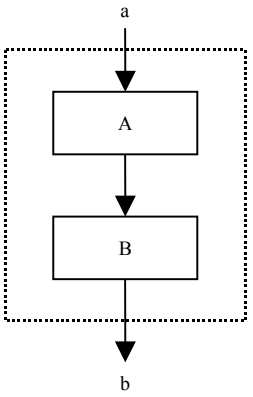

图 2-3　顺序结构流程图

2.6.2　分支结构

现实中很多问题用顺序结构无法解决，如数学中的分段函数，就需要判断后进行选择。分支结构又称选择结构，是指根据给定的条件进行判断，由判断结果再决定执行哪一步操作。C++语言中提供了 3 种设计选择结构的语句，即 if 语句、if…else 语句和 switch 语句。

1. if 语句

if 语句的语法格式如下。

```
if(表达式)语句块;
```

它表示如果表达式为真，则执行语句块，否则跳过此语句块，执行 if 结构下面的其他语句。

其流程如图 2-4 所示。

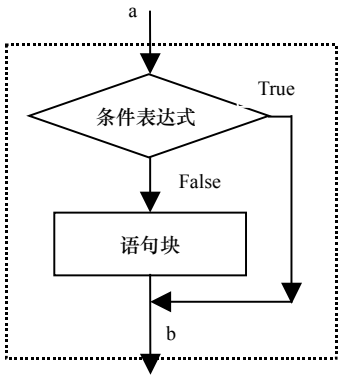

图 2-4　if 语句流程图

【例 2-4】 编程求一个整数的绝对值。

程序设计

定义一个整型变量，从键盘上为该变量输入一个值，若变量的值为负，则绝对值为其相反数。

源程序代码

```
#include<iostream>
using namespace std;
int main() {
    int a;
    cin>>a;
    if(a<0)a=-a;
    cout<<a<<endl;
    return 0;
}
```

2. if…else 语句

if…else 语句又称双分支选择语句，其语法格式如下。

```
if(表达式)语句块 A;
else 语句块 B;
```

它表示如果表达式为真，则执行语句块 A，否则执行语句块 B。其流程如图 2-5 所示。

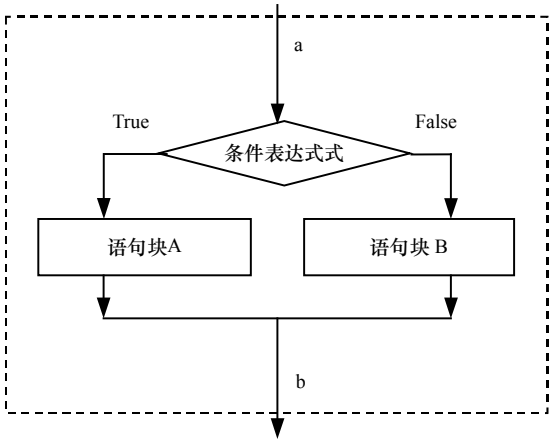

图 2-5　if…else 语句流程图

【例 2-5】 用 if…else 语句编程求一个整数的绝对值。

程序设计

同样的问题，既可以用 if 语句解决，也可以用 if…else 语句完成，此时需要用两个输出语句实现。

源程序代码

```
#include<iostream>
using namespace std;
int  main(){
    int a;
    cin>>a;
    if(a<0)cout<<-a;
    else cout<<a;
    cout<<endl;
    return 0;
}
```

3. switch 语句

switch 语句也称多分支选择语句，或称开关语句。其语法格式如下。

```
switch(表达式){
    case 常量表达式 1: 语句序列 1; break;
    case 常量表达式 2: 语句序列 2; break;
     ……
    case 常量表达式 n: 语句序列 n; break;
    default: 语句序列 n+1;
}
```

它的含义为：先计算表达式的值并与各 case 后面的常量表达式的值比较，如果与第 i（1≤i≤n）个常量表达式的值相等，则执行语句序列 i，i+1，…，n+1 直到遇到 break 语句，跳出 switch 结构，继续向下执行 switch 后面的程序；如果不与任何一个常量表达式相等，则执行语句序列 n+1 后跳出 switch 结构，继续向下执行 switch 后面的程序。其流程如图 2-6 所示。

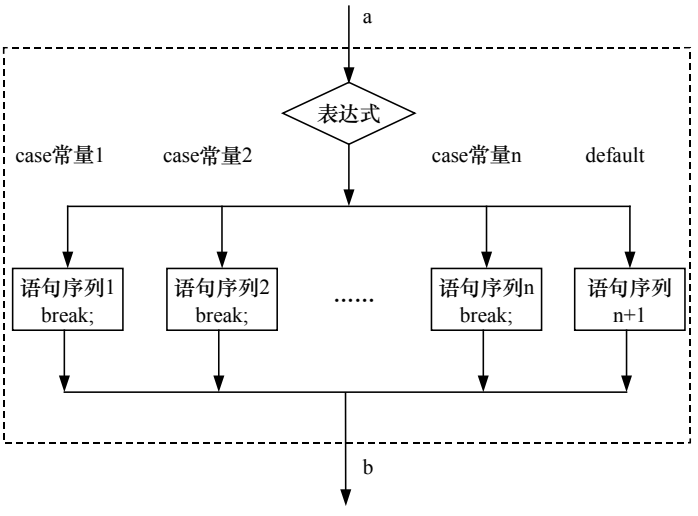

图 2-6　switch 语句流程图

在 switch 语句中的表达式及各常量表达式的值都只能是整型、字符型或枚举类型，且每个常量表达式的值必须互不相同。

【例 2-6】 编写程序，根据输入的学生成绩，给出相应的等级。假设 90 分以上为 A，80～89 分为 B，70～79 分为 C，60～69 分为 D，60 分以下为 E。

程序设计

本题程序可以用 if…else 语句，也可以用 switch 语句编写，但一般情况下对于多种情况分类，用 if…else 语句容易引起逻辑上的错误，而用 switch 语句可更清楚地表示各语句逻辑上的关系，故采用 switch 语句编写。设用 score 变量表示学生成绩，由于 switch 语句不能表示数值的范围，因而需要做一定的处理，将取值范围转换成确定的值，这里利用整数运算的特性，取 score/10。

源程序代码

```
#include<iostream>
using namespace std;
int main() {
     int  score;
     cin>>score;
     switch(score/10){
          case 10:
          case 9:cout<<'A'<<'\n';break;
          case 8:cout<<'B'<<'\n';break;
          case 7:cout<<'C'<<'\n';break;
          case 6:cout<<'D'<<'\n';break;
          default:cout<<'E'<<'\n';
     }
     return 0;
}
```

switch 结构中的 break 语句不是必须的，它的作用是结束 switch 结构。如果某个 case 下的语句中不包括 break 语句，则将继续执行 switch 结构的下一条语句，不需要判断新条件。同时，switch 结构中的 default 可以放在 switch 中的任何位置，也可以省略。当 switch 语句中没有 default 分支，且匹配失败时，将不执行任何分支。

【例 2-7】 设 grade 表示学生成绩，根据输入值分析下列程序的输出结果。

```
#include<iostream>
using namespace std;
int  main() {
     int  grade;
     cin>>grade;
     switch(grade/10){
          case 10:
          case 9:
          case 8:
          case 7:
          case 6:cout<<"通过"<<'\n';break;
          default:cout<<"不通过"<<'\n';
     }
     return 0;
}
```

在例 2-7 程序执行过程中，由于 case 10、case 9、case 8、case 7 后面都是空语句，且没有 break 语句，故若输入大于等于 60 的整数，则输出“通过”，否则输出“不通过”。

4. 选择语句的嵌套

根据求解问题的需要，编程时在 if 语句中可以嵌套 if 语句，还可以嵌套 switch 语句。同样在 switch 语句中也可以嵌套 if 语句，称为选择语句的嵌套。

【例 2-8】 从键盘输入一个字符，判断其类型。假设字符分为控制字符（ASCII 码小于 32 的字符）、大写字母、小写字母、数字字符和其他字符 5 类。

程序设计

为了在输入时能包括如空格、换行符等字符，这里用 cin.get()函数输入一个字符变量。由于字符在内存中是以 ASCII 码的形式存储的，故对于输入的字符，若其 ASCII 码值小于 32，则是控制字符，否则看其是否为大写字母，若其 ASCII 码值大于等于字符'A'的 ASCII 码值，且小于等于字符'Z'的 ASCII 码值，则该字符为大写字母。小写字母及数字字符类似。

源程序代码

```
#include<iostream>
using namespace std;
int  main(){
     char  c;
     cin.get(c);
     if(c<32)cout<<"这是一个控制字符。"<<endl;
     else if(c>='A'&&c<='Z')cout<<"这是大写字母。"<<endl;
        else if(c>='a'&&c<='z')cout<<"这是小写字母。"<<endl;
          else if(c>='0'&&c<='9')cout<<"这是一个数字字符。"<<endl;
            else cout<<"这是一个其他字符。"<<endl;
     return 0;
}
```

需要注意的是，在 if…else 语句中，只能在 if 后面加条件，切不可加到 else 后面，并且每个 else 必须跟唯一一个 if 配对，配对的方法是与在它上方同一个块中离它最近，且没有配对过的 if 配对。

【例 2-9】 编写程序完成两个数的四则运算。

程序设计

因为本题需要按输入的运算符确定具体的四则运算，是多选择问题，所以采用 switch 语句实现。但对于除法运算分母不能为 0，故需要用选择语句判断。

源程序代码

```
#include<iostream>
using namespace std;
int  main(){
    float a,b;
    char ch;
    cout<<"请输入表达式(如 a+b 的形式):";
    cin>>a>>ch>>b;
    switch(ch){
        case  '+':cout<<a<<'+'<<b<<'='<<a+b<<'\n';break;
        case  '-':cout<<a<<'-'<<b<<'='<<a-b<<'\n';break;
        case  '*':cout<<a<<'*'<<b<<'='<<a*b<<'\n';break;
        case  '/':
           if(b==0)cout<<"分母不能为零!"<<'\n';
           else cout<<a<<'/'<<b<<'='<<a/b<<'\n';
```

```
            break;
        default:  cout<<"表达式错误!"<<'\n';
    }
    return 0;
}
```

2.6.3　循环结构

在程序设计中，常常需要根据条件重复执行一些操作，这种重复执行的过程称为循环。C++中有 3 种循环语句，分别是 while 语句、do…while 语句和 for 语句。

1. while 语句

while 语句属于当型循环，语法格式如下。

```
while(条件表达式)
    循环体;
```

其含义为当条件表达式的值为真时，执行循环体，直至条件表达式的值是假为止。其中，条件表达式可以是任何合法的表达式，称为循环控制条件；循环体可以是单语句、复合语句，也可以是空语句。其流程如图 2-7 所示。

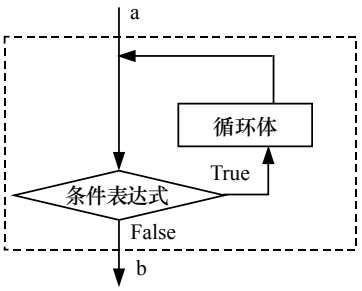

图 2-7　while 语句流程图

【例 2-10】 编写程序，求 s=1+2+…+100 的值。

程序设计

设置变量 s 存放和，其初始值为 0。本题是将一些数重复地加到和 s 上。这里可设置变量 i，初始值为 1，使其不断增加来控制重复次数，该变量称为循环变量，用 i<=100 作为循环条件。

源程序代码

```
#include<iostream>
using namespace std;
int  main(){
     int  i=1,s=0;
     while(i<=100){                    //A
          s+=i;
          i++;                         //B，改变循环变量
     }
     cout<<s<<endl;
     return 0;
}
```

在执行循环过程中，若循环无法终止，将形成死循环或称无限循环。因此在程序设计时应避

免出现死循环。例如，在上述程序中，将 A 行中的条件表达式改为 1，或将 B 行去掉，由于条件永远成立，因此是死循环。

2. do…while 语句

do…while 语句属于直到型循环，语法格式如下。

```
do{
  循环体;
}while(条件表达式);
```

其含义为首先执行循环体，然后计算条件表达式的值，当条件表达式的值为真时，继续执行循环体，直至表达式的值是假为止。其流程如图 2-8 所示。

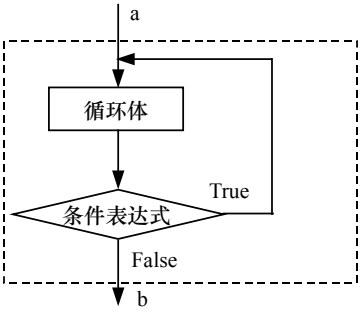

图 2-8　do…while 语句流程图

【例 2-11】 用 do…while 语句编写程序，求 s=1+2+…+100 的值。

程序设计　改用 do…while 语句，即先执行循环体，再判断循环条件，即循环体至少执行一次。

源程序代码

```
#include<iostream>
using namespace std;
int  main(){
     int  i=1,s=0;
     do{
        s+=i;
        i++;
     }while(i<=100);
     cout<<s<<endl;
     return 0;
}
```

在本例程序中，循环体是一个由两条语句组成的复合语句，也可以改为一条语句实现。

```
do{
     s+=i++;
}while(i<=100);
```

使用循环语句时，需仔细考虑循环的边界条件。例如，将本例程序中的自增语句改成前置，或放到循环条件上，程序又应该如何写呢?

3. for 语句

for 语句的语法格式如下。

```
for(表达式 1；表达式 2；表达式 3)
    循环体;
```

其执行过程如下。

步骤 1：执行表达式 1。

步骤 2：判断表达式 2 的值，若为真，则执行循环体，转步骤 3；否则循环结束。

步骤 3：执行表达式 3，转步骤 2。

在 for 语句中，3 个表达式都可以是任何合法的表达式，也可以是空表达式，表达式 1 和表达式 3 为空表示不做任何操作，表达式 2 为空表示条件恒成立。其流程如图 2-9 所示。

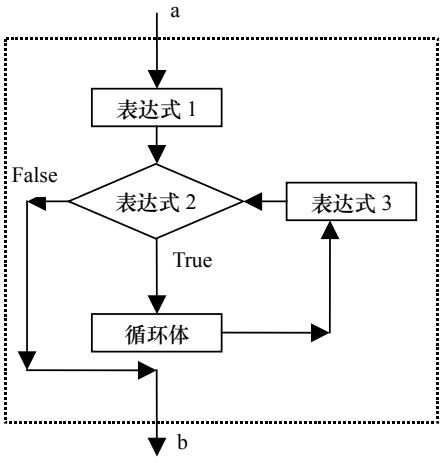

图 2-9　for 循环流程图

实际上，3 种循环语句的使用是可以相互转换的，在使用时要注意各语句的执行过程，变量初始值、循环结束条件，以及哪些语句应该参与循环。无论使用哪种循环语句，都必须严格按照语法格式来写。

【例 2-12】 用 for 语句编写程序，求 s=1+2+…+100 的值。

程序设计　根据 for 语句的执行过程，表达式 1 可以初始化变量，表达式 2 用来作循环条件，表达式 3 用于修改循环变量。

源程序代码

```
#include<iostream>
using namespace std;
int  main(){
     int i,s=0;
     for(i=1;i<=100;i++)
         s+=i;
     cout<<s<<endl;
     return 0;
}
```

本题中同样可以将自增语句放到循环体中，此时 for 语句中的表达 3 为空语句。

```
for(i=1;i<=100;){
     s+=i;
     i++;
}
```

一般说来，当已知循环次数时，选用 for 语句和 while 语句较多，而在不知道循环次数的情况下，多选用 do…while 语句。

4. 循环语句的嵌套

一个循环语句的循环体中可以包含另一个循环语句，称为循环语句的嵌套，也称为多重循环。另外，循环语句与选择语句还可以相互嵌套。

【例 2-13】 编程计算 s=1!+2!+…+10!的值。

程序设计

本题首先需要用循环语句求 10 个值的和，其中每个值是一个阶乘，也就是要对每个循环变量 i（1≤i≤10）计算 i!。为此又需要用循环语句，使循环变量 j 从 1 到 I 做乘法运算，故需要双重循环。

源程序代码

```
#include<iostream>
using namespace std;
int  main(){
     int  i,j,t,s=0;
     for(i=1;i<=10;i++){                    //外层循环开始
          t=1;                              //A
          for(j=1;j<=i;j++)                 //内层循环开始
               t=t*j;                       //内层循环结束
          s+=t;
     }                                      //外层循环结束
     cout<<s<<endl;
     return 0;
}
```

程序中外循环用来求 10 个数的和，其中 A 行语句不可放到外层循环之前，这是因为对于每个循环变量 i，i!都应该从初始值 1 开始计算。

本例也可以用单循环来实现，源程序代码如下。

```
#include<iostream>
using namespace std;
int  main(){
     int i,j,t=1,s=0;
     for(i=1;i<=10;i++){
          t=t*i;
          s+=t;
     }
     cout<<s<<endl;
     return 0;
}
```

2.6.4 转向语句

转向语句是用来改变原来执行顺序的语句，通常与循环语句一起使用，包括 break、continue 和 goto 语句。

1. break 语句

break 语句的语法格式如下。

```
break;
```

除了前面在 switch 语句中用于跳出 switch 结构外，break 还可以用在循环语句中，表示跳出循环结构，执行循环语句后面的语句。

【例 2-14】 编写程序，判断一个整数是否为素数。

程序设计

判断整数 n 是否为素数的方法为：用 n 分别除以数 2～n-1（或用 n 分别除以数 2～n/2，或用 n 分别除以数 2～$\sqrt{n}$ ），若都不能整除，则 n 是素数，否则 n 不是素数。本例用 n 分别除以数 2～n-1 来判断。

源程序代码

```
#include<iostream>
using namespace std;
int  main(){
     int n,k=1;
     cout<<"请输入一个整数:";
     cin>>n;
     for(int i=2;i<=n-1;i++)
          if(n%i==0){
              k=0;
              break;
     }
     if(k)cout<<n<<"是素数!"<<'\n';
     else cout<<n<<"不是素数!"<<'\n';
     return 0;
}
```

在本程序中，k 起到标示的作用，编程时也可以不使用变量 k，直接判断 n 是否为素数。代码如下。

```
for(int i=2;i<=n-1;i++)                    //A
    if(n%i==0)break;
if(i>n-1)cout<<n<<"是素数!"<<'\n';
else cout<<n<<"不是素数!"<<'\n';
```

A 行循环语句或在某个 i（i<=n-1）时，由于 i 是 n 的因子，通过 break 语句结束，此时 n 不是素数，或在循环条件 i<=n-1 不满足（即 i>n-1）时结束，此时 n 是素数。

2. continue 语句

continue 语句的语法格式如下。

```
continue;
```

它的作用是跳过循环体中 continue 后面的语句，即结束本次循环，开始下一次循环。continue 语句只能用在循环语句中。

【例 2-15】 编程求 2～100 间的非素数。

程序设计

对 2～100 间的每一个数，逐个判断是否为素数，若是则跳过该数判断下一个数，否则输出。这里分别用 i 除以数 2～i/2 来判断 i 是否为素数。

源程序代码

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

```
int  main(){
int i,j,k=0;                              //变量 k 用来存放素数的个数
for(i=2;i<=100;i++){
     for(j=2;j<=i/2;j++)                  //判断 i 是否为素数
          if(i%j==0)  break;
     if(j>i/2)                            //i 是素数
          continue;                       //结束这一次循环
     k++;
     cout<<i<<'\t';
}
cout<<'\n';
cout<<"共有"<<k<<"个非素数."<<'\n';
return 0;
}
```

当 j>i/2 时，说明所有的 i%j 均不为零，即 i 是素数，此时语句“k++;”与“cout<<i<<'\t';”不执行，故用 continue 语句将它们跳过，进入下一次循环，判断下一个 i。

3. goto 语句

goto 语句又称无条件转向语句，其语法格式如下。

```
goto  label;
……
label:
```

它的作用是将程序控制转移到 label 标号指定的语句处继续执行。标号是由用户自定义的一个标识符。在这里，goto 语句与标号 label 必须在同一个函数中。

由于 goto 语句会破坏程序的结构，使得程序层次不清且不易阅读，故一般不主张使用。

2.7 程 序 举 例

【例 2-16】 编写程序，求一个三角形的面积。

程序设计

定义 3 个变量 a、b、c 分别存放三角形的 3 条边，首先判断 3 条边是否构成一个三角形。若能构成三角形，则用求面积公式计算并输出面积，否则输出“不能构成三角形!”。

源程序代码

```
#include<iostream>
#include<cmath>
using namespace std;
int  main(){
     double a,b,c;
     cin>>a>>b>>c;
     if(a+b>c&&b+c>a&&c+a>b){
          double s,area;
          s=(a+b+c)/2;
          area=sqrt(s*(s-a)*(s-b)*(s-c));
          cout<<"三角形的面积为: "<<area<<'\n';
     }
     else cout<<"不能构成三角形! "<<'\n';
```

```
    return 0;
}
```

【例 2-17】 编程求出所有的“水仙花数”。

程序设计

所谓“水仙花数”是指一个 3 位数，其各位数字的立方和恰好等于该数本身。例如 $153=1^3+5^3+3^3$，所以 153 是“水仙花数”。本题可由多种方法实现。

方法 1：穷举出所有 3 位数，对每一个 3 位数，先分别求出其百、十、个位上的数字，再求出各个数字的立方和。最后判断其和与这个 3 位数是否相等。

方法 2：对方法 1 进行改进，其中求各位数字的立方和用循环语句实现，即先将原数 i 用变量 n 保存下来，求出 n 的最后一位数（用取余运算），同时将最后一位数的立方加到和 s 上，并用 n/10 取代 n（去掉这个数的最后一位），重复此过程，直到 n 是 0 为止。最后判断和 s 与 i 是否相等。此方法适用于任何位数的情况。

方法 3：用 3 个变量分别表示 3 位数的百位、十位和个位，利用三重循环嵌套，组合出所有 3 位数，判断由这 3 个数组成的 3 位数与其数字立方和是否相等。

源程序代码

方法 1：

```
#include<iostream>
using namespace std;
int  main(){
     int  i,a,b,c;
     for(i=100;i<=999;i++){
          a=i/100;                    //a 是数 i 的百位数
          b=i/10-a*10;                //b 是数 i 的十位数
          c=i-b*10-a*100;             //c 是数 i 的个位数
          if(i==a*a*a+b*b*b+c*c*c)
                cout<<a<<b<<c<<endl;
       }
     return 0;
}
```

方法 2：

```
#include<iostream>
using namespace std;
int  main(){
     int  i,n,k,s;
     for(i=100;i<=999;i++){
          s=0;n=i;
          while(n){
              k=n%10;               //取出最后一位数
              n/=10;                //去掉最后一位
              s+=k*k*k;             //将取出数的立方加到和中
          }
          if(i==s)cout<<i<<endl;
     }
     return 0;
}
```

方法 3：

```
#include<iostream>
using namespace std;
int  main(){
    int  i,j,k;
    for(i=1;i<=9;i++)                    //百位数
        for(j=0;j<=9;j++)                //十位数

            for(k=0;k<=9;k++)            //个位数
                if(i*100+j*10+k==i*i*i+j*j*j+k*k*k)
                     cout<< i*100+j*10+k <<endl;
     return 0;
}
```

【例 2-18】 设计一个程序，求 Fibonacci 数列的前 20 项。要求每行输出 4 项。

程序设计

Fibonacci 数列是指满足下列条件的数列。

$$f_n=\begin{cases}1 & n=1\\ 1 & n=2\\ f_{n-1}+f_{n-2} & n\geqslant 3\end{cases}$$

本题可重复使用 3 个变量 f1、f2 和 f3，迭代出第 3 项至第 20 项，即由 f3=f1+f2，f1=f2，f2=f3 逐项计算出每一项。

源程序代码

```
#include<iostream>
#include <iomanip>                                  //A
using namespace std;
int  main(){
    long  int f1=1, f2=1, f3;
    cout<<setw(12)<<f1<<setw(12) <<f2 ;             //B，输出前 2 项
    for(int n=3; n<=20;n++){                        //求 3～20 项
           f3=f1+f2;
           cout<<setw(12)<<f3;                      //C，输出新值
           f1=f2;  f2=f3;                           //更新 f1 和 f2，注意赋值次序
           if(n%4==0) cout<<'\n';                   //每行输出 4 项
      }
    cout <<endl;
    return 0;
}
```

例 2-18 程序中的 B 行及 C 行用函数 setw()设置输出宽度，setw()函数在头文件 iomanip 中，故 A 行不可少。setw(12)表示其后的输出项占 12 字节，右对齐。

【例 2-19】 利用迭代法求 $\sqrt{a}$ 的近似值，要求前后两次求出的根的近似值之差的绝对值小于 10^{-5}。迭代公式为：$x_{n+1}=(x_n+a/x_n)/2$。

程序设计

本程序设计的基本思想为指定一个初始值 x0，依据迭代公式计算出 x1。|x1-x0|< ε（ε=10^{-5}）停止，否则，将 x1 作为 x0，依据迭代公式重新计算出 x1，再比较 x1 与 x0 之差的绝对值。如此

继续，直到满足|x1-x0|< ε 为止。

源程序代码

```
#include<iostream>
#include <cmath>
using namespace std;
int  main(){
    float x0,x1,a;
    cout<<"输入一个正数：";
    cin>>a;
    if(a<0) cout<<a<<"不能开平方! \n";
    else{
         x1=a/2;                          //初始值
         do {
            x0=x1;
           x1=(x0+a/x0)/2;
        }while(fabs(x1-x0)>1e-5);
        cout<<a<<"的平方根等于："<<x1<<'\n';
    }
    return 0;
}
```

【例 2-20】 用公式：$\frac{\pi}{4}\approx 1-\frac{1}{3}+\frac{1}{5}-\frac{1}{7}+$　求 π 的近似值，要求最后一项的绝对值不大于 10^{-6}。

程序设计

根据公式用循环语句求出每一项，并将其加到和中，循环条件为求出的项大于给定的条件。公式中每项符号与前一项符号相反，循环中通过 k*=-1 实现每项符号的变换。每项分母比前一项分母大 2，每次循环分母加 2。最后计算 π 的值。

源程序代码

```
#include<iostream>
#include <cmath>
using namespace std;
int  main(){
     double pi=0,fac=1,den=1;       //pi 表示和，fac 表示某一项，den 表示分母
     int k=1;
     while(fabs(fac)>1e-6){
          pi+=fac;
          den+=2;
          k*=-1;
          fac=k/den;
     }
     pi*=4;
     cout<<"π的值为:"<<pi<<endl;
     return 0;
}
```

2.8　习　　题

1. 设有变量 x，y，z，写出下列数学表达式在 C++中的相应形式。

（1）$|x|$　　（2）$2x$　　（3）'a'≤x≤'z'　　（4）$\frac{4}{\sqrt{x^3y^3}}$　　（5）tan$x$

2. 写出判断某年 x 为闰年的表达式。

3. 下列字符串的长度分别是多少？

（1）"abc"　　（2）"abc\0xy"　　（3）"a\134\n\\bc\t"

4. 设有变量定义：int x=2,y=4,z=7；写出下列表达式的值以及计算表达式后 x、y、z 的值。

（1）z%=x　　（2）z=（++x，y--）　　（3）x+y>++z　　（4）x>（y>z?y：z）?x：（y>z?y：z）

（5）x=y=z　　（6）y==z　　（7）（x--<y）&&（++x<z）

（8）（x--<y）||（++x<z）

5. 编写程序，求从键盘输入的 3 个数中的最小数。

6. 编程求方程 $ax^2+bx+c=0$ 的解。

7. 任意给定一个月份数，编程输出它属于哪个季节（12 月、1 月、2 月是冬季；3 月、4 月、5 月是春季；6 月、7 月、8 月是夏季；9 月、10 月、11 月是秋季）。

8. 从键盘上输入 10 个整数，编程求它们的平均值。

9. 从键盘上输入若干人学生的成绩，统计并输出其中的最高成绩和最低成绩，当输入负数时结束输入。

10. 编写程序，找出 1 000 以内的所有完数。一个数如果恰好等于它的因子之和，这个数就称为“完数”。例如，6 的因子为 1、2、3，而 6=1+2+3，因此 6 是完数。