第二章 C++简单程序设计（1）

清华大学　郑 莉

教材：C++语言程序设计（第4版） 郑莉　清华大学出版社
答疑：每周一17:30—19:00，东主楼8区310

问题与解决

- 计算机的最基本功能是数据处理，因此：
  - 程序要有能力表示与存储各种类型的数据
  - 程序要能够进行各种运算
- C++支持的基本数据类型：整数、实数、字符、逻辑数据
- C++支持的基本运算：算术运算、逻辑运算
- 程序的执行流程不总是顺序的，因此
  - 程序要能够对执行流程进行选择（选择语句、开关语句）
  - 程序要能够用反复同一算法依次处理大批量数据（循环语句）
- 基本数据类型能够表示的数据种类很有限，因此
  - 需要能够在程序中自定义类型（第4章讲类）
  - 枚举类型就是通过列出所有可取值，来定义一种新类型

目录

C++语言概述

基本数据类型和表达式

数据的输入与输出

C++的产生和发展

- C++从C语言发展演变而来，最初的C++被称为“带类的C”；
- 1983年正式取名为C++；
- 于1998年11月被国际标准化组织（ISO）批准为国际标准；
- 2003年10月15日发布第2版C++标准ISO/IEC 14882:2003；
- 2011年8月12日ISO公布了第三版C++标准C++11，C++11标准包含核心语言的新机能，而且扩展C++标准程序库。
- 2014年8月18日ISO公布了C++14，其正式名称为"International Standard ISO/IEC 14882:2014(E) Programming Language C++"。C++14旨在作为C++11的一个小扩展，主要提供漏洞修复和小的改进。

C++的特点

- 兼容C
  - 它保持了C的简洁、高效和接近汇编语言等特点
  - 对C的类型系统进行了改革和扩充
  - C++也支持面向过程的程序设计，不是一个纯正的面向对象的语言
- 支持面向对象的方法
- 支持泛型程序设计方法

例2-1C++程序实例

```
//2_1.cpp
#include <iostream>
using namespace std;
int main() {
        cout << "Hello!" << endl;
        cout << "Welcome to c++!" << endl;
        return 0;
}
```

运行结果：

```
Hello!
Welcome to c++ !
```

C++字符集

- 大小写的英文字母：A～Z，a～z
- 数字字符：0～9
- 特殊字符：

!	#	%	^	&	*	_	+	=	-	
~	<	>	/	\	'	"	;	.	,	:
?	(	)	[	]	{	}	|			

词法记号

- 关键字　　C++预定义的单词
- 标识符　　程序员声明的单词，它命名程序正文中的一些实体
- 文字　　在程序中直接使用符号表示的数据
- 操作符　　用于实现各种运算的符号
- 分隔符 ()　{}　,　:　;
  用于分隔各个词法记号或程序正文
- 空白符　　空格、制表符（TAB键产生的字符）、垂直制表符、换行符、回车符和注释的总称

关键字

```
alignas    alignof    asm    auto    bool    break    case
catch    char    char16_t    char32_t    class    const    constexpr
const_cast    continue    decltype    default    delete    do    double
dynamic_cast    else    enum    explicit    export    extern    false
float    for    friend    goto    if    inline    int
long    mutable    namespace    new    noexcept    nullptr    operator
private    protected    public    register    reinterpret_cast    return
short    signed    sizeof    static    static_cast    struct    switch
template    this    thread_local    throw    true    try    typedef
typeid    typename    union    unsigned    using    virtual    void
volatile    wchar_t    while
```

标识符的构成规则

- 以大写字母、小写字母或下划线(_)开始。
- 可以由以大写字母、小写字母、下划线(_)或数字0～9组成。
- 大写字母和小写字母代表不同的标识符。
- 不能是C++关键字或操作符。

补充2_1：计算商品总价（整数、实数、运算）

知识点：整数与实数、常量、变量、初始化、C风格字符串常量、算数运算、赋值运算、表达式、标准输入/输出

```
#include<iostream>
using namespace std;
int main()
{
const double price1 = 25.5, price2 = 10.3, price3 = 12.5;
double total=0;
int number1=0, number2=0, number3=0;
cout << "三种商品价格为:"<< price1 << ','<< price2<< ','<<price3<< '\n';
cout << "请问每样买几件？" << endl;
cin >> number1 >> number2 >> number3;
total = number1*price1 + number2*price2 + number3*price3;
cout << "应付款总额：" << total << endl;
return 0;
}
```

基本数据类型

- C++能够处理的基本数据类型
  - 整数类型
  - 浮点数类型
  - 字符类型
  - 布尔类型
- 程序中的数据
  - 常量
    - 在源程序中直接写明的数据，其值在整个程序运行期间不可改变，这样的数据称为常量。
  - 变量
    - 在程序运行过程中允许改变的数据，称为变量。

文字常量

- 所谓常量是指在程序运行的整个过程中其值始终不可改变的量，文字常量是直接使用符号（文字）表示的值。例如：12，3.5，’A’都是常量。

变量

- 变量定义：
  - 数据类型　变量名1, 变量名2, ..., 变量名n;
- 在定义一个变量的同时，也可以对它初始化。C++语言中提供了多种初始化方式，例如：
  - int a = 0;
  - int a(0);
  - int a = {0};
  - int a{0};
  - 其中使用大括号的初始化方式称为列表初始化，列表初始化时不允许信息的丢失。例如用double值初始化int变量，就会造成数据丢失。
- C++基本数据类型没有字符串类型，C++标准库中有string类。

符号常量

- 常量定义语句的形式为：
  - const 数据类型说明符 常量名=常量值；
  - 或：
  - 数据类型说明符 const 常量名=常量值；
- 例如，我们可以定义一个代表圆周率的符号常量：
  - const float PI = 3.1415926；
- 注意，符号常量在定义时一定要赋初值，而在程序中间不能改变其值。

补充2_2：不同类型整数的最值

```
#include <iostream>
#include <limits>
using namespace std;
int main()
{
cout << "Min of short is : " << SHRT_MIN << endl;
cout << "Max of short is : " << SHRT_MAX << endl;
cout << "Min of int is : " << INT_MIN << endl;
cout << "Max of int is : " << INT_MAX << endl;
cout << "Min of long is : " << LONG_MIN << endl;
cout << "Max of long is : " << LONG_MAX << endl;
cout << "Max of unsigned short is : " << USHRT_MAX << endl;
cout << "Max of unsigned int is : " << UINT_MAX << endl;
cout << "Max of unsigned long is : " << ULONG_MAX << endl;
return 0;
}
```

补充2_2：不同类型整数的最值

运行结果：

```
Min of short is: -32768
Max of short is: 32767
Min of int is: -2147483648
Max of int is: 2147483647
Min of long is: -2147483648
Max of long is: 2147483647
Max of unsigned short is: 65535
Max of unsigned int is: 4294967295
Max of unsigned long is: 4294967295
```

基本数据类型

- 整数类型
  - 基本的整数类型
    - int
  - 按符号分
    - 符号的（signed）和无符号的（unsigned）
  - 按照数据范围分
    - 短整数（short）和长整数（long）、长长整数（ long long ）
  - char类型
    - 字符型，实质上存储的也是整数（详见字符类型）

整数文字常量

- 以文字形式出现的整数；
- 十进制
  - 若干个0~9的数字，但数字部分不能以0开头，正数前边的正号可以省略。
- 八进制
  - 前导0+若干个0~7的数字。
- 十六进制
  - 前导0x+若干个0~9的数字及A~F的字母（大小写均可）。
- 后缀
  - 后缀L（或l）表示类型至少是long，后缀LL（或ll）表示类型是long long，后缀U（或u）表示unsigned类型。

补充2_3：不同类型浮点数的应用

```
#include <iostream>
#include <limits>
using namespace std;
const float PI_FLOAT = 3.1415926f;
const double PI_DOUBLE = 3.1415926;
const long double PI_LDOUBLE = 3.1415926;
int main()
{
    float nRadiusFloat = 5.5f, nAreaFloat;
    double nRadiusDouble = 5.5, nAreaDouble;
    long double nRadiusLDouble = 5.5, nAreaLDouble;
    nAreaFloat = PI_FLOAT*nRadiusFloat*nRadiusFloat;
    nAreaDouble = PI_DOUBLE*nRadiusDouble*nRadiusDouble;
    nAreaLDouble = PI_DOUBLE*nRadiusDouble*nRadiusDouble;
    cout<<"nAreaFloat ="<< nAreaFloat<<", sizeof(nAreaFloat) = "<<sizeof(nAreaFloat)<<endl;
    cout<<"nAreaDouble ="<<nAreaDouble<<", sizeof(nAreaDouble) ="<<sizeof(nAreaDouble)<<endl;
    cout<<"nAreaLDouble ="<< nAreaLDouble<<", sizeof(nAreaLDouble) ="<<sizeof(nAreaLDouble)<<endl;
    return 0;
}
```

补充2_3：不同类型浮点数的应用

运行结果：

```
nAreaFloat =  95.0332 , sizeof(nAreaFloat) = 4
nAreaDouble = 95.0332 , sizeof(nAreaDouble) = 8
nAreaLDouble = 95.0332 , sizeof(nAreaLDouble) = 8
```

基本数据类型

- 浮点数类型
  - 单精度
    - float
  - 双精度
    - double
  - 扩展精度
    - long double

浮点数文字常量

- 以文字形式出现的实数。
- 一般形式：
  - 例如，12.5，-12.5等。
- 指数形式：
  - 例如，0.345E+2，-34.4E-3
  - 整数部分和小数部分可以省略其一
- 浮点常量默认为double型，如果后缀F（或f）可以使其成为float型，例如：12.3f。

补充2_4：字符数据的应用

```
#include <iostream>
using namespace std;
int main()
{
    cout << 'A'<< ' '<<'a'<<endl;       //输出普通字符
    cout << "one\ttwo\tthree\n";      //使用水平制表符
    cout << "123\b\b45\n";              //使用退格符
    cout << "Alert\a\n";                    //使用响铃键
    return 0;
}
```

补充2_4：字符数据的应用

运行结果：

```
A a
one     two     three
145
Alert
```

补充2_5：字符串变量的输入和输出

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

int main()
{
  string str;
  cout << "请输入字符串：";
  cin >> str;
  cout << "输入的字符串为：" << str << endl;
  return 0;
}
```

补充2_6：bool类型的变量

```
#include <iostream>
using namespace std;
int main()
{
  bool bV1 = true, bV2 = false;
  cout << "bool value bV1 = " << bV1 << endl;
  cout << "bool value bV2 = " << bV2 << endl;
  int nV1 = bV1, nV2 = 0;
  bV1 = nV2;
  cout << "int value nV1 = " << nV1 << endl;
  cout << "bool value bV1 = " << bV1 << endl;
  return 0;
}
```

补充2_6：bool类型的变量

运行结果：

```
bool value bV1 = 1
bool value bV2 = 0
int value nV1 = 1
bool value bV1 = 0
```

基本数据类型

- 字符类型
  - char类型
  - 容纳单个字符的编码
- 字符串类型（详见第6章）
  - C风格的字符串
    - 采用字符数组
  - C++风格的字符串
    - 采用标准C++类库中的string类
- 布尔类型
  - bool类型，只有两个值：true（真）、false（假）
  - 常用来表示关系比较、相等比较或逻辑运算的结果

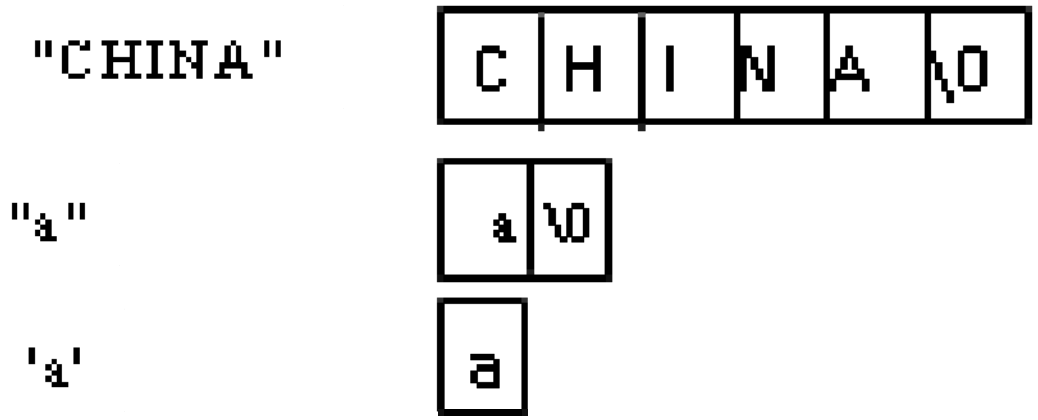
字符文字常量

- 单引号括起来的一个字符，如：'a', 'D', '?', '$ ‘
- C++转义字符列表（用于在程序中表示不可显示字符）

字符常量形式	ASCII码（十六进制）	含义
\a	07	响铃
\n	0A	换行
\t	09	水平制表符
\v	0B	垂直制表符
\b	08	退格
\r	0D	回车
\f	0C	换页
\\	5C	字符“\”
\”	22	双引号
\’	27	单引号
\?	3F	问号

C风格字符串常量

- 一对双引号括起来的字符序列
- 字符串与字符是不同的，它在内存中的存放形式是：按串中字符的排列次序顺序存放，每个字符占一个字节，并在末尾添加 ‘\0’ 作为结尾标记。



C风格字符串常量

- 通过添加前缀可以改变字符常量或者字符串常量的类型，前缀及其含义如下表所示：

前缀	含义	类型
u	Unicode 16 字符	char16_t
U	Unicode 32字符	char32_t
L	宽字符	wchar_t
u8	UTF-8（仅用于字符串字面常量）	char

基本数据类型

类型名	长度（字节）	取值范围
bool	1	false，true
char	1	-128~127
signed char	1	-128~127
unsigned char	1	0~255
short (signed short)	2	-32768~32767
unsigned short	2	0~65535
int (signed int)	4	-2^{31}~2^{31}-1
unsigned int	4	0~2^{32}-1
long (signed long)	4	-2^{31}~2^{31}-1
unsigned long	4	0~2^{32}-1
long long	8	-2^{63}~2^{63}-1
unsigned long long	8	0~2^{64}-1
float	4	绝对值范围3.4×10^{-38}~3.4×10^{38}
double	8	绝对值范围1.7×10^{-308}~1.7×10^{308}
long double	8	绝对值范围1.7×10^{-308}~1.7×10^{308}

注：表中各类型的长度和取值范围，以面向IA32处理器的msvc12和2 gcc4.8为准

基本数据类型

- ISO C++标准并没有明确规定每种数据类型的字节数和取值范围，它只是规定它们之间的字节数大小顺序满足：
  - (signed/unsigned)signed char ≤ (unsigned) short int ≤ (unsigned) int ≤ (unsigned) long int ≤ long long int

补充2_7：sizeof运算

```
#include <iostream>
#include <string>
using namespace std;
int main()
{
    cout << "sizeof(short) = " << sizeof(short) << endl;
    cout << "sizeof(unsigned short)= " << sizeof(unsigned short) << endl;
    cout << "sizeof(int) = " << sizeof(int) << endl;
    cout << "sizeof(unsigned int)= " << sizeof(unsigned int) << endl;
    cout << "sizeof(long) = " << sizeof(long) << endl;
    cout << "sizeof(unsigned long)= " << sizeof(unsigned long) << endl;
    cout << "sizeof(float)= " << sizeof(float) << endl;
    cout << "sizeof(double) = " << sizeof(double) << endl;
    cout << "sizeof(long double)= " << sizeof(long double)<< endl;
    cout << "sizeof(char)= " << sizeof(char) << endl;
    return 0;
}
```

sizeof 运算符

- 语法形式
  sizeof (类型名)
  或 sizeof 表达式
- 结果值：
  “类型名”所指定的类型，或“表达式”的结果类型所占的字节数。
- 例：

```
sizeof(short)
sizeof  x
```

类型别名

- 可以为一个已有的数据类型另外命名
  - typedef 已有类型名 新类型名表
    - typedef double Area, Volume;
    - typedef int Natural;
    - Natural i1,i2;
    - Area a;
    - Volume v;
  - using 新类型名 = 已有类型名;
    - using Area = double;
    - using Volume = double;

auto类型与decltype类型

- auto让编译器通过初始值自动推断变量的类型。
  - 例如：

    auto val = val1 + val2;

    val的类型取决于表达式val1+val2的类型，如果val1+val2是int类型，那么val将是int类型；如果val1+val2是double类型，那么val将是double类型。
- 定义一个变量与某一表达式的类型相同，但并不想用该表达式初始化这个变量，这时我们需要decltype变量
  - 例如：

    decltype(i) j = 2;

    表示 j 以 2 作为初始值，类型与 i 一致。

算术运算符与算术表达式

- 基本算术运算符

  + - * /(若整数相除，结果取整)

  %（取余，操作数为整数）

- 优先级与结合性

  先乘除，后加减，同级自左至右

- ++, --（自增、自减）

  例：i++; --j;

赋值运算符和赋值表达式
——简单的赋值运算符"="

- 举例
  n = n + 5
- 表达式的类型
  赋值运算符左边对象的类型
- 表达式的值
  赋值运算符左边对象被赋值后的值

补充2_1回顾：计算商品总价（整数、实数、运算）

知识点：整数与实数、常量、变量、初始化、C风格字符串常量、算数运算、赋值运算、表达式、标准输入/输出

```
#include<iostream>
using namespace std;
int main()
{
const double price1 = 25.5, price2 = 10.3, price3 = 12.5;
double total=0;
int number1=0, number2=0, number3=0;
cout << "三种商品价格为:"<< price1 << ','<< price2<< ','<<price3<< '\n';
cout << "请问每样买几件？" << endl;
cin >> number1 >> number2 >> number3;
total = number1*price1 + number2*price2 + number3*price3;
cout << "应付款总额：" << total << endl;
return 0;
}
```

逗号运算和逗号表达式

- 格式

  表达式1，表达式2
- 求解顺序及结果
  - 先求解表达式1，再求解表达式2
  - 最终结果为表达式2的值
- 例

  a = 3 * 5，a * 4　最终结果为60

补充2_9：逻辑运算及短路现象

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

int main()
{
	int a, b;
	cin>>a>>b;
	cout << ((a > 2) && (++b)) << endl;
	cout << "b = " << b << endl;
	cout << ((a > 2) || (++b)) << endl;
	cout << "b = " << b << endl;
	return 0;
}
```

关系运算与关系表达式

- 关系运算是比较简单的一种逻辑运算，优先次序为：

  < <= > >= == !=

  优先级相同（高） 优先级相同（低）

- 关系表达式是一种最简单的逻辑表达式

  其结果类型为 bool，值只能为 true 或 false。

- 例如：a > b，c <= a + b，x + y == 3

逻辑运算与逻辑表达式

- 逻辑运算符

  !(非)　　&&(与)　　||(或)

  优先次序：　高　　→　　低

- 逻辑表达式

  例如：(a > b) && (x > y)

  其结果类型为 bool，值只能为 true 或 false

逻辑运算与逻辑表达式(续)

- “&&” 的“短路特性”

  表达式1 && 表达式2
  - 先求解表达式1
  - 若表达式1的值为false，则最终结果为false，**不再求解表达式2**
  - 若表达式1的结果为true，则求解表达式2，以表达式2的结果作为最终结果
- “||”也具有类似的特性

补充2_8：比较数据并输出结果

```
#include <iostream>
using namespace std;
int main()
{
    int ival1 = 1, ival2 = 2, ival3 = 3, ival4 = 4;
    bool nFlag;
    nFlag = ival1 == ival2 ? true : false;
    cout << "Is 1 equals 2 ? : " << nFlag << endl;
    nFlag = ival1<ival2 ? true : false;
    cout << "Is 1less than 2 ? : " << nFlag << endl;
    nFlag = (ival1<ival2 && ival3<ival4) ? true : false;
    cout << "Is 1less than 2 and 3 less than 4: " << nFlag << endl;
}
```

补充2_8：比较数据并输出结果

运行结果：

```
Is 1 equals 2?: 0
Is 1less than 2?: 1
Is 1less than 2 and 3 less than 4: 1
```

条件运算符与条件表达式

- 一般形式

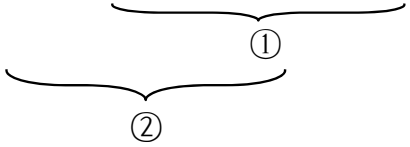
  表达式1？表达式2：表达式3

  表达式1 必须是 bool 类型

- 执行顺序
  - 先求解表达式1，
  - 若表达式1的值为true，则求解表达式2，表达式2的值为最终结果
  - 若表达式1的值为false，则求解表达式3，表达式3的值为最终结果
- 例： x = a > b ? a : b;

条件运算符与条件表达式（续）

- 条件运算符的优先级
  - 条件运算符优先级高于赋值运算符，低于逻辑运算符
  - 表达式2、3的类型可以不同，条件表达式的最终类型为 2 和 3 中较高的类型。
- 例： x = a > b ? a : b;



位运算——按位与（&）

- 运算规则
  - 将两个运算量的每一个位进行逻辑与操作
- 举例：计算 3 & 5

  3：　0 0 0 0 0 0 1 1

  5：　0 0 0 0 0 1 0 1

  3 & 5:　0 0 0 0 0 0 0 1
- 用途：
  - 将某一位置0，其他位不变。例如：
    将char型变量a的最低位置0: a = a & 0xfe;
  - 取指定位。
    例如：有char c; int a;
    取出a的低字节，置于c中：c=a & 0xff;

位运算——按位或（|）

- 运算规则
  - 将两个运算量的每一个位进行逻辑或操作
- 举例：计算 3 | 5

  3：　0 0 0 0 0 0 1 1

  5：　0 0 0 0 0 1 0 1

  3 | 5:　0 0 0 0 0 1 1 1
- 用途：
  - 将某些位置1，其他位不变。
    例如：将 int 型变量 a 的低字节置 1 ：

    a = a | 0xff;

位运算——按位异或（^）

- 运算规则
  - 两个操作数进行异或：
    若对应位相同，则结果该位为 0，
    若对应位不同，则结果该位为 1，
- 举例：计算 071^052

```
    071:    0 0 1 1 1 0 0 1
    052:    0 0 1 0 1 0 1 0
071^052:    0 0 0 1 0 0 1 1
```

位运算——按位异或（^）（续）

- 用途：
  - 使特定位翻转（与0异或保持原值，与1异或取反）

  例如：要使 01111010 低四位翻转：

  ```
      0 1 1 1 1 0 1 0
  (^) 0 0 0 0 1 1 1 1
      0 1 1 1 0 1 0 1
  ```

位运算——取反（~）

单目运算符，对一个二进制数按位取反。

例： 025：0000000000010101

~025：1111111111101010

位运算——移位

- 左移运算（<<）
  左移后，低位补0，高位舍弃。
- 右移运算（>>）
  右移后，
  低位：舍弃
  高位：无符号数：补0
  有符号数：补“符号位”

补充2_10：位运算

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

int main()
{
	unsigned int a = 24, b = 13;
	cout << "a & b =" << (a & b) << endl;
	cout << "a | b = " << (a | b) << endl;
	cout << "a ^ b = " << (a ^ b) << endl;
	cout << "~a = " << (~a) << endl;
	cout << (5 << 1) << endl;
	cout << (5 >> 1) << endl;
	cout << (13 >> 2) << endl;
	cout << (13 << 2) << endl;
	return 0;
}
```

混合运算时数据类型的转换——隐含转换

条件		转换
有一个操作数是long double型。		将另一个操作数转换为long double型。
前述条件不满足，并且有一个操作数是double型。		将另一个操作数转换为double型。
前述条件不满足，并且有一个操作数是float型。		将另一个操作数转换为float型。
前述条件不满足（两个操作数都不是浮点数）。	有一个操作数是unsigned long long型。	将另一个操作数转换为unsigned long long型。
	有一个操作数是long long型，另一个操作数是unsigned long型	两个操作数都转换为unsigned long long型。
	前述条件不满足，并且有一个操作数是unsigned long型。	将另一个操作数转换为unsigned long型。
	前述条件不满足，并且有一个操作数是long型，另一个操作数是unsigned int型。	将两个操作数都转换为unsigned long型。
	前述条件不满足，并且有一个操作数是long型。	将另一个操作数转换为long型。
	前述条件不满足，并且有一个操作数是unsigned int型。	将另一个操作数转换为unsigned int型。
	前述条件都不满足	将两个操作数都转换为int型。

混合运算时数据类型的转换

- 当把一个非布尔类型的算术值赋给布尔类型时，算术值为0则结果为false，否则结果为true。
- 当把一个布尔值赋给非布尔类型时，布尔值为false则结果为0，布尔值为true则结果为1
- 当把一个浮点数赋给整数类型时，结果值将只保留浮点数中的整数部分，小数部分将丢失。
- 当把一个整数值赋给浮点类型时，小数部分记为0。如果整数所占的空间超过了浮点类型的容量，精度可能有损失。

混合运算时数据类型的转换——显式转换

- 语法形式（3种）：
  - 类型说明符(表达式)
  - (类型说明符)表达式
  - 类型转换操作符<类型说明符>(表达式)
    - 类型转换操作符可以是：
      const_cast、dynamic_cast、
      reinterpret_cast、static_cast
- 显式类型转换的作用是将表达式的结果类型转换为类型说明符所指定的类型。
- 例：int(z), (int)z, static_cast<int>(z)
  三种完全等价

I/O流

- 在C++中，将数据从一个对象到另一个对象的流动抽象为“流”。流在使用前要被建立，使用后要被删除。
- 从流中获取数据的操作称为提取操作，向流中添加数据的操作称为插入操作。
- 数据的输入与输出是通过I/O流来实现的，cin和cout是预定义的流类对象。cin用来处理标准输入，即键盘输入。cout用来处理标准输出，即屏幕输出。

预定义的插入符和提取符

- “<<”是预定义的插入符，作用在流类对象cout上便可以实现项标准输出设备输出。
  - cout << 表达式 << 表达式...
- 标准输入是将提取符作用在流类对象cin上。
  - cin >> 表达式 >> 表达式...
- 提取符可以连续写多个，每个后面跟一个表达式，该表达式通常是用于存放输入值的变量。例如：
  - int a, b;
  - cin >> a >> b;

简单的I/O格式控制

常用的I/O流类库操纵符

操纵符名	含　　义
dec	数值数据采用十进制表示
hex	数值数据采用十六进制表示
oct	数值数据采用八进制表示
ws	提取空白符
endl	插入换行符，并刷新流
ends	插入空字符
setsprecision(int)	设置浮点数的小数位数（包括小数点）
setw(int)	设置域宽

例:cout << setw(5) << setprecision(3) << 3.1415;

第2章（1）小结

- 主要内容
  - C++语言概述、基本数据类型和表达式、数据的输入与输出。
- 达到的目标
  - 掌握C++语言的基本概念和基本语句，能够编写简单的程序段。