

## § . 习题课 (Week-04)



### 1、双编译器的基本使用

- ★ 配色方案的选择（工具-选项-环境-常规-颜色主题）
- ★ VS的“解决方案资源管理器”的位置（鼠标拖动的方法）
- ★ VS的“解决方案资源管理器”被关闭（视图-解决方案资源管理器）
- ★ 一个解决方案包含多个项目（后续作业推荐使用此方案）
- ★ 进入到某项目源程序文件所在的目录（方便Dev启动）
- ★ 设定Dev为. c/. cpp的默认打开程序



## §. 习题课 (Week-04)

### 2、运算符优先级与结合性

#### ★ 含有cin/cout的表达式分析

- cout、endl当运算数
- << 当运算符
- cout << 表达式的计算结果为cout

### §. 基础知识题 – cin与cout的基本使用



#### 1、cout的基本理解

B. 观察下列4个程序的运行结果，回答问题并将各程序的运行结果截图贴上(如果有错则贴错误信息截图)

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

int main()
{
    int a=10, b=15, c=20;
    cout << a << b << c;
    return 0;
}
```

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

int main()
{
    int a=10, b=15, c=20;
    cout << a, b, c;
    return 0;
}
```

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

int main()
{
    int a=10, b=15, c=20;
    cout << (a, b, c) << endl;
    return 0;
}
```

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

int main()
{
    int a=10, b=15, c=20;
    cout << a, b, c << endl;
    return 0;
}
```

解释这3个程序输出不同的原因：

解释错误原因：

结论：一个流插入运算符 << 只能输出\_\_\_\_\_个数据。



## §. 习题课 (Week-04)

2、运算符优先级与结合性

★ 含有cin/cout的表达式分析

★ 含有()的表达式分析

### §. 基础知识题

2、仿照课件PDF的P. 65–85，用栈方式给出下列表达式的求解过程

C.  $a + (a - 3 / (b + c) + 5) \% 4$  (假设所有变量均为int型)

(本题提示：将左右小括号分开处理，

1、“进栈前优先级最高，进栈后优先级最低；

2、)”优先级最低，因此要将栈中压在”(“之上的全部运算符都计算完成，随后和”(“成对消除即可

表达式一共有 10 个运算符，因此计算的 6 个步骤分别是（左右括号不算步骤）：

步骤①：



## §. 习题课 (Week-04)

3、IEEE754相关

### ★ 补码的数学解析

#### ● 模与补数：

模： 某种类型的数据能够表示的范围(例：1字节数据，模=256)  
=> 数据超过模的范围则自动取模 (溢出截断)

补数：一个数的补数等于模-自身 (例：数据 补数

$$\begin{array}{ll} 10 & 246 \\ 127 & 129 \end{array}$$

#### ● 减法的两个方法：

- ◆ 正常减法：A - B
- ◆ 补数加法：A + B的补数

例：100 - 10 = 90

$$100 + 246 \bmod 256 = 90 \text{ (溢出截断)}$$

=> 计算机内所有减法都可以表示为加法

#### ● 负数的补码当做无符号数理解，正好就是其补数

(两者绝对值相加为模)

$$\begin{aligned} -10 : \text{ 绝对值 } &\rightarrow 0000\ 1010 \\ \text{ 取反 } &\rightarrow 1111\ 0101 \\ +1 &\rightarrow 1111\ 0110 = 2^7 + 2^6 + 2^5 + 2^4 + 2^2 + 2^1 \\ &\quad = 128 + 64 + 32 + 16 + 4 + 2 = 246 \end{aligned}$$

## §. 习题课

3、IEEE754相关

★ 三个特殊的值：0/inf/nan

0：

● 指数部分全0，尾数部分全0，则表示0

此时不能理解为  $0.0 * 2^{-127}$  (23位位数为0)

$1.0 * 2^{-127}$  (补隐含的1)

```
#include <iostream>
using namespace std;
int main()
{
    float f = 0;
    cout << f << endl;

    unsigned char* p = (unsigned char*)&f;
    cout << hex << (int)(*p) << endl;
    cout << hex << (int)((p+1)) << endl;
    cout << hex << (int)((p+2)) << endl;
    cout << hex << (int)((p+3)) << endl;
```

```
}
```

0000 0000 0000 0000 0000 0000 0000 0000 0000

The screenshot shows the Microsoft Visual Studio IDE. The code editor window displays the file 'demo.cpp' with the following content:

```
1 #include <iostream>
2 using namespace std;
3 int main()
4 {
5     float f = 0;
6     cout << f << endl;
7
8     unsigned char* p = (unsigned char*)&f;
9     cout << hex << (int)(*p) << endl;
10    cout << hex << (int)((p + 1)) << endl;
11    cout << hex << (int)((p + 2)) << endl;
12    cout << hex << (int)((p + 3)) << endl;
13
14    return 0;
15 }
```

To the right of the code editor is the 'Immediate Window' titled 'Microsoft Visual Studio 调试控制台'. It shows the output of the program:

```
0
0
0
0
0
0
0
0
0
```

A yellow arrow points from the bottom of the Immediate Window back up towards the code editor, highlighting the memory dump output.

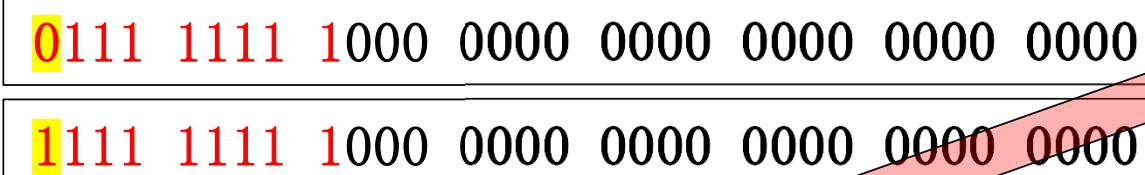
## §. 习题课 (Week-04)

### 3、IEEE754相关

★ 三个特殊的值: 0/inf/nan

inf: 无穷大(infinite)

- 指数部分全1, 尾数部分全0, 则表示无穷大, 可根据符号位分正负
- 浮点数本身范围有限, 虽然可表示无穷大, 但无法表示可增长到
- double型自行测试



```
#include <iostream>
using namespace std;
int main()
{
    float f = 1e40; //超float上限
    cout << f << endl;

    unsigned char* p = (unsigned char*)&f;
    cout << hex << (int)(*p) << endl;
    cout << hex << (int)(*(p+1)) << endl;
    cout << hex << (int)(*(p+2)) << endl;
    cout << hex << (int)(*(p+3)) << endl;

    return 0;
}
```

```
#include <iostream>
using namespace std;
int main()
{
    float f = -1e40; //超float下限
    cout << f << endl;

    unsigned char* p = (unsigned char*)&f;
    cout << hex << (int)(*p) << endl;
    cout << hex << (int)(*(p+1)) << endl;
    cout << hex << (int)(*(p+2)) << endl;
    cout << hex << (int)(*(p+3)) << endl;

    return 0;
}
```

```
demo.cpp demo-cpp (全局范围)
1 #include <iostream>
2 using namespace std;
3 int main()
4 {
5     float f = 1e40;
6     cout << f << endl;
7
8     unsigned char* p = (unsigned char*)&f;
9     cout << hex << (int)(*p) << endl;
10    cout << hex << (int)(*(p+1)) << endl;
11    cout << hex << (int)(*(p+2)) << endl;
12    cout << hex << (int)(*(p+3)) << endl;
13
14 }
15
16 return 0;
```

Microsoft Visual Studio 调试控制台  
inf  
0  
0  
80  
7f

```
demo.cpp demo-cpp (全局范围)
1 #include <iostream>
2 using namespace std;
3 int main()
4 {
5     float f = -1e40;
6     cout << f << endl;
7
8     unsigned char* p = (unsigned char*)&f;
9     cout << hex << (int)(*p) << endl;
10    cout << hex << (int)(*(p+1)) << endl;
11    cout << hex << (int)(*(p+2)) << endl;
12    cout << hex << (int)(*(p+3)) << endl;
13
14 }
15
16 return 0;
```

Microsoft Visual Studio 调试控制台  
-inf  
0  
0  
80  
ff



## §. 习题课 (Week-04)

3、IEEE754相关

★ 三个特殊的值: 0/inf/nan

nan: 非法(not a number)

ind: indeterminate

● 指数部分全1, 尾数部分不全0, 则表示非法, 符号位无意义

1111 1111 1100 0000 0000 0000 0000 0000

```
#include <iostream>
using namespace std;
int main()
{
    float f = sqrt(-2); //负数开根号非法
    cout << f << endl;

    unsigned char* p = (unsigned char*)&f;
    cout << hex << (int)(*p) << endl;
    cout << hex << (int)(*(p+1)) << endl;
    cout << hex << (int)(*(p+2)) << endl;
    cout << hex << (int)(*(p+3)) << endl;

    return 0;
}
```

The screenshot shows the Microsoft Visual Studio IDE with the demo.cpp file open. The code demonstrates the IEEE754 representation of NaN. A yellow arrow points from the line of code `cout << f << endl;` to the corresponding output in the 'Microsoft Visual Studio 调试控制台' (Debug Console). The output shows the binary representation of NaN: -nan(ind).

```
demo.cpp ② x (全局范围)
1 #include <iostream>
2 using namespace std;
3 int main()
4 {
5     float f = sqrt(-2); //负数开根号非法
6     cout << f << endl;
7
8     unsigned char* p = (unsigned char*)&f;
9     cout << hex << (int)(*p) << endl;
10    cout << hex << (int)(*(p+1)) << endl;
11    cout << hex << (int)(*(p+2)) << endl;
12    cout << hex << (int)(*(p+3)) << endl;
13
14 }
15
16
Microsoft Visual Studio 调试控制台
-nan(ind)
0
0
c0
ff
```



## §. 习题课 (Week-04)

3、IEEE754相关

### ★ 有效位数的理解

- 有效位数的数学定义：从左边第一个非零数字起的数位数
- 对float型数据，23位尾数，加隐含的1，则尾数的最大值(二进制)  
 $1111\ 1111\ 1111\ 1111\ 1111 = 16777216 \quad (2^{24})$   
可以表示 $0\sim9999999$ 的所有十进制7位数，但8位数不能超过16777216  
=> float型数的有效位数为7位  
=> 浮点数表示时，一般表示为  $x.\underset{\text{6位}}{\text{xxxxxx}} * 10^{\underset{\text{2位}}{\text{xx}}}$ ，也称小数点后6位  
=> 综上：各种资料一般都称float的有效数字为 $6\sim7$ 位
- 对double型数据，52位位数，加隐含的1，则位数的最大值(二进制)  
 $9,007,199,254,740,992 \quad (2^{53})$  是一个16位的十进制数  
=> double型数的有效位数为15位  
=> 因为接近能将16位整数全部表示完，有时也称有效位数16位  
=> 综上：各种资料一般都称float的有效数字为 $15\sim16$ 位
- **本课程统一：** float的有效位数7位， double的有效位数15位

## §. 习题课 (Week-04)



3、IEEE754相关

### ★ 浮点数的输出

- 超出有效位数的可以输出，但不可信

=> 按前页，float型数的有效位数为前7位100%可信，第8位只有16%可信

=> 按前页，double型数的有效位数为前15位100%可信，第16位只有90%可信

## §. 习题课 (Week-04)



3、IEEE754相关

### ★ 极小的浮点数

- 小于等于  $10^{-46}$  次方则无法表示

0000 0000 0000 0000 0000 0000 0000 0000

```
#include <iostream>
using namespace std;
int main()
{
    float f = 1e-46;
    cout << f << endl;

    unsigned char* p = (unsigned char*)&f;
    cout << hex << (int)(*p) << endl;
    cout << hex << (int)(*(p+1)) << endl;
    cout << hex << (int)(*(p+2)) << endl;
    cout << hex << (int)(*(p+3)) << endl;

    return 0;
}
```

The screenshot shows the Microsoft Visual Studio IDE. On the left, the code editor displays the `demo.cpp` file with the following content:

```
1 #include <iostream>
2 using namespace std;
3 int main()
4 {
5     float f = 1e-46;
6     cout << f << endl;
7
8     unsigned char* p = (unsigned char*)&f;
9     cout << hex << (int)(*p) << endl;
10    cout << hex << (int)(*(p+1)) << endl;
11    cout << hex << (int)(*(p+2)) << endl;
12    cout << hex << (int)(*(p+3)) << endl;
13
14 }
15
16
```

On the right, the Immediate Window shows the output of the program:

```
0
0
0
0
0
0
```

A yellow arrow points from the bottom of the Immediate Window back up towards the code editor.

## §. 习题课 (Week-04)

### 3、IEEE754相关

#### ★ 极小的浮点数

- 小于等于  $10^{-46}$  次方则无法表示

```
0000 0000 0000 0000 0000 0000 0000 0000
```

- 最小值为  $10^{-45}$

```
0000 0000 0000 0000 0000 0000 0001
```

```
1000 0000 0000 0000 0000 0000 0001
```

```
#include <iostream>
using namespace std;
int main()
{
    float f = 1e-45;
    cout << f << endl;

    unsigned char* p = (unsigned char*)&f;
    cout << hex << (int)(*p) << endl;
    cout << hex << (int)((p+1)) << endl;
    cout << hex << (int)((p+2)) << endl;
    cout << hex << (int)((p+3)) << endl;

    return 0;
}
```

```
#include <iostream>
using namespace std;
int main()
{
    float f = -1e-45;
    cout << f << endl;

    unsigned char* p = (unsigned char*)&f;
    cout << hex << (int)(*p) << endl;
    cout << hex << (int)((p+1)) << endl;
    cout << hex << (int)((p+2)) << endl;
    cout << hex << (int)((p+3)) << endl;

    return 0;
}
```

The screenshot shows the Microsoft Visual Studio IDE. The code in demo.cpp prints the value of `f` using `cout << f << endl;`. It then prints the memory representation of `f` at four consecutive addresses: `(*p)`, `(*p + 1)`, `(*p + 2)`, and `(*p + 3)` using `cout << hex << (int)(*)p << endl;`. The output window shows the value `1.4013e-45` followed by four lines of memory dump: `1 0 0 0`.

```
demo.cpp  demo.cpp  (全局范围)
1  #include <iostream>
2  using namespace std;
3  int main()
4  {
5      float f = 1e-45;
6      cout << f << endl;
7
8      unsigned char* p = (unsigned char*)&f;
9      cout << hex << (int)(*p) << endl;
10     cout << hex << (int)((p+1)) << endl;
11     cout << hex << (int)((p+2)) << endl;
12     cout << hex << (int)((p+3)) << endl;
13
14     return 0;
15 }
16
```

Microsoft Visual Studio 调试控制台  
1.4013e-45  
1  
0  
0  
0

The screenshot shows the Microsoft Visual Studio IDE. The code in demo.cpp prints the value of `f` using `cout << f << endl;`. It then prints the memory representation of `f` at four consecutive addresses: `(*p)`, `(*p + 1)`, `(*p + 2)`, and `(*p + 3)` using `cout << hex << (int)(*)p << endl;`. The output window shows the value `-1.4013e-45` followed by four lines of memory dump: `1 0 0 0`.

```
demo.cpp  demo.cpp  (全局范围)
1  #include <iostream>
2  using namespace std;
3  int main()
4  {
5      float f = -1e-45;
6      cout << f << endl;
7
8      unsigned char* p = (unsigned char*)&f;
9      cout << hex << (int)(*p) << endl;
10     cout << hex << (int)((p+1)) << endl;
11     cout << hex << (int)((p+2)) << endl;
12     cout << hex << (int)((p+3)) << endl;
13
14     return 0;
15 }
16
```

Microsoft Visual Studio 调试控制台  
-1.4013e-45  
1  
0  
0  
0



## §. 习题课 (Week-04)

3、IEEE754相关

### ★ 极小的浮点数

- 小于等于  $10^{-46}$  次方则无法表示

0000 0000 0000 0000 0000 0000 0000 0000

- 最小值为  $10^{-45}$

0000 0000 0000 0000 0000 0000 0000 0001

$$000\ 0000\ 0000\ 0000\ 0000\ 0001 = 0.00000011920928955078125$$

$$1.00000011920928955078125 \times 2^{-127} = 1.4013e-45$$

1000 0000 0000 0000 0000 0000 0000 0001

$$000\ 0000\ 0000\ 0000\ 0000\ 0001 = 0.00000011920928955078125$$

$$-1.00000011920928955078125 \times 2^{-127} = -1.4013e-45$$

```
Microsoft Visual Studio 调试控制台
1. 4013e-45
1
0
0
0
```

```
Microsoft Visual Studio 调试控制台
-1. 4013e-45
1
0
0
80
```

## §. 习题课 (Week-04)

### 3、IEEE754相关

#### ★ 浮点数的比较范围

- float型1e-45范围内
- double型自行探究

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

int main()
{
    float f1 = 1.4e-45;
    float f2 = 1.3e-45;
    cout << f1 - f2 << endl;
    cout << (fabs(f1 - f2) < 1e-44) << endl;
    cout << (fabs(f1 - f2) < 1e-45) << endl;
    cout << (fabs(f1 - f2) < 1e-46) << endl;

    return 0;
}
```

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

int main()
{
    float f1 = 1.4e-44;
    float f2 = 1.3e-44;
    cout << f1 - f2 << endl;
    cout << (fabs(f1 - f2) < 1e-44) << endl;
    cout << (fabs(f1 - f2) < 1e-45) << endl;
    cout << (fabs(f1 - f2) < 1e-46) << endl;

    return 0;
}
```

The screenshot shows the Microsoft Visual Studio IDE with two panes. The left pane displays the code for 'demo.cpp'. The right pane shows the 'Microsoft Visual Studio 调试控制台' (Debug Console) with the following output:

```
0
1
1
1
```

A red arrow points from the bottom of the console output to the value '1'.

The screenshot shows the Microsoft Visual Studio IDE with two panes. The left pane displays the code for 'demo.cpp'. The right pane shows the 'Microsoft Visual Studio 调试控制台' (Debug Console) with the following output:

```
1.4013e-45
1
0
0
```

A red arrow points from the bottom of the console output to the value '0'.



## §. 习题课 (Week-04)

### 4、cin相关

#### ★ cin的基本步骤

Step1: 用户输入数据(字符序列)到缓冲区中(采用队列方式)

Step2: 按**合理且最长原则**读缓冲区, 直到回车/空格/非法输入为止

Step3: 将读到的字符序列转为变量类型, 如果转换过程中超过变量的范围, 则置good()为0/fail()为1

## § 3. 结构化程序设计基础

### 3. 4. C++的输入与输出

#### 3. 4. 3. 输入流的基本操作

第3章课件 P.19

格式: `cin >> 变量1 >> 变量2 >> ... >> 变量n;`

★ 输入终止条件为回车、空格、非法输入

★ 系统会自动根据cin后变量的类型按**最长原则**来读取合理数据

★ 变量读取后, 系统会判断输入数据是否超过变量的范围, 若超过则**置内部的错误标记**并返回一个**不可信的值**(不同编译器处理不同)

★ cin输入完成后, 通过`cin.good()/cin.fail()`可判断本次输入是否正确

输入	<code>cin.good()</code> 返回	<code>cin.fail()</code> 返回
正确范围+回车/空格/非法输入	1	0
错误范围+回车/空格/非法输入	0	1
非法输入	0	1



## §. 习题课 (Week-04)

4、cin相关

★ cin的基本步骤

★ 如何理解**合理且最长原则**

	第一个字符	后续字符	非法字符
整型(以十进制为例)	+、-、0~9	0~9	非0~9(包括+、-)
浮点型(以小数形式为例)	+、-、小数点、0~9	0~9及第一个出现的小数点	非0~9(包括+、-、小数点)
字符	缓冲区中第一个可读字符	无	不含空格、回车等不可读字符
字符串(暂不考虑)			

- 按数学常识理解即可，**不需要额外知识**
- 第一个非法则置good()为0/fail()为1并返回

例1:

```
int a;  
cin >> a; //键盘输入1234567890123456.12
```

处理：读到.为止，超上限，good()为0，值**不可信**

例2:

```
char a;  
cin >> a; //键盘输入1234567890123456.12
```

处理：读到2为止，只读到一个1，值为**ASCII值49**



## §. 习题课 (Week-04)

4、cin相关

★ cin的基本步骤

★ 如何理解合理且最长原则

★ 连续多个输入的理解

例1：

### §. 基础知识题 – cin与cout的基本使用

4、cin的基本理解 – 其他情况

E. 程序如下，观察编译及运行结果（贴图在清晰可辨的情况下尽可能小）

```
#include <iostream>
using namespace std;
int main()
{
    char c1, c2;
    int a;
    float b;
    cin >> c1 >> c2 >> a >> b;

    cout << c1 << ' ' << c2 << ' ' << a << ' ' << b << endl;
    return 0;
}
```

注：' '表示空格

1、输入：1234 ' 56.78  
输出：

2、输入：1 ' 2 ' 34 ' 56.78  
输出：

3、分析在以上两种不同输入的情况下，  
为什么输出相同(提示：空格的作用)



## §. 习题课 (Week-04)

4、cin相关

- ★ cin的基本步骤
- ★ 如何理解合理且最长原则
- ★ 连续多个输入的理解

例2：

```
int a;  
char b;  
cin >> a >> b; //键盘输入12.3↙
```

处理：a读到.为止，得值12，缓冲区中剩余.3↙，b读到.，缓冲区中还剩余3↙（程序运行完成后丢弃）

The screenshot shows a Microsoft Visual Studio IDE window. The code editor displays the following C++ code:

```
demo.cpp  demo-cpp  (全局范围)  main()  
1  #include <iostream>  
2  using namespace std;  
3  
4  int main()  
5  {  
6      int a;  
7      char b;  
8      cin >> a >> b; //键盘输入12.3↙  
9      cout << cin.good() << ',' << a << ',' << b << endl;  
10     return 0;  
11 }  
12
```

The output window below the code editor shows the results of the program's execution:

```
12.3  
1 12 .
```



## §. 习题课 (Week-04)

4、cin相关

★ cin的基本步骤

★ 如何理解合理且最长原则

★ 连续多个输入的理解

例3：

```
int a;  
float b;  
cin >> a >> b; //键盘输入12.3↙
```

处理：a读到.为止，得值12，缓冲区中剩余.3↙，b读到.3，得值0.3，缓冲区中还剩余↙(程序运行完成后丢弃)

The screenshot shows a Microsoft Visual Studio IDE window. The title bar says "demo.cpp". The code editor contains the following C++ code:

```
demo.cpp  
demo-cpp (全局范围) main()  
1 #include <iostream>  
2 using namespace std;  
3  
4 int main()  
5 {  
6     int a;  
7     float b;  
8     cin >> a >> b; //键盘输入12.3↙  
9     cout << cin.good() << ' ' << a << ' ' << b << endl;  
10    return 0;  
11 }  
12
```

The output window below the editor shows the results of the program's execution:

```
12.3  
1 12 0.3
```



## §. 习题课 (Week-04)

4、cin相关

★ cin的基本步骤

★ 如何理解合理且最长原则

★ 连续多个输入的理解

例4：

```
int a, b;  
cin >> a >> b; //键盘输入12.3↙
```

处理：a读到.为止，得值12，缓冲区中剩余.3↙

b读到.，第一个即非法，置good()为0，b被赋值但不可信(VS下b为0)，缓冲区中还剩余.3↙

(在good()恢复为1前不再读任何数据)

```
demo.cpp  x  demo-cpp  (全局范围)  main()
```

```
1 #include <iostream>  
2 using namespace std;  
3  
4 int main()  
5 {  
6     int a, b;  
7     cin >> a >> b; //键盘输入12.3↙  
8     cout << cin.good() << ' ' << a << ' ' << b << endl;  
9     return 0;
```

Microsoft Visual Studio 调试控制台  
12.3  
0 12 0



## §. 习题课 (Week-04)

4、cin相关

★ cin的基本步骤

★ 如何理解合理且最长原则

★ 连续多个输入的理解

例5：

```
int a=11, b=22, c=33;  
cin >> a >> b >> c; //键盘输入12.3✓
```

处理：a读到.为止，得值12，缓冲区中剩余.3✓

b读到.，第一个即非法，置good()为0，b被赋值但不可信(VS下b为0)，缓冲区中还剩余.3✓

(在good()恢复为1前不再读任何数据，因此c维持原值33)

The screenshot shows a Microsoft Visual Studio interface with a code editor and a debugger window. The code editor displays a file named 'demo.cpp' with the following content:

```
1 #include <iostream>  
2 using namespace std;  
3  
4 int main()  
5 {  
6     int a = 11, b = 22, c = 33;  
7     cin >> a >> b >> c; //键盘输入12.3✓  
8     cout << cin.good() << ' ' << a << ' ' << b << ' ' << c << endl;  
9  
10 }  
11
```

The debugger window shows the output of the program:

```
12.3  
0 12 0 33
```



## §. 习题课 (Week-04)

4、cin相关

★ cin的基本步骤

★ 如何理解合理且最长原则

★ 连续多个输入的理解

例5：变化

```
int a, b, c;  
cin >> a >> b >> c; //键盘输入12.3✓
```

处理：a读到.为止，得值12，缓冲区中剩余.3✓

b读到.，第一个即非法，置good()为0，b被赋值但不可信(VS下b为0)，缓冲区中还剩余.3✓

(在good()恢复为1前不再读任何数据，因此c维持原值，若未初始化则为不确定的随机值)

The screenshot shows a Microsoft Visual Studio IDE window. The title bar says "demo.cpp". The code editor contains the following C++ code:

```
1 #include <iostream>  
2 using namespace std;  
3  
4 int main()  
5 {  
6     int a, b, c;  
7     cin >> a >> b >> c; //键盘输入12.3✓  
8     cout << cin.good() << ' ' << a << ' ' << b << ' ' << c << endl;  
9     return 0;  
10 }  
11 }
```

The output window at the bottom shows the results of the program execution:

```
12.3  
0 12 0 -858993460
```

## §. 习题课 (Week 1)

4、cin相关

- ★ cin的基本步骤
- ★ 如何理解合理且最长原则
- ★ 连续多个输入的理解
- ★ 不可信值的理解
- 只要good()不为1，值就不可信

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

int main()
{
    short a;
    cin >> a;
    cout << cin.good() << ' ' << a << endl;

    return 0;
}
```

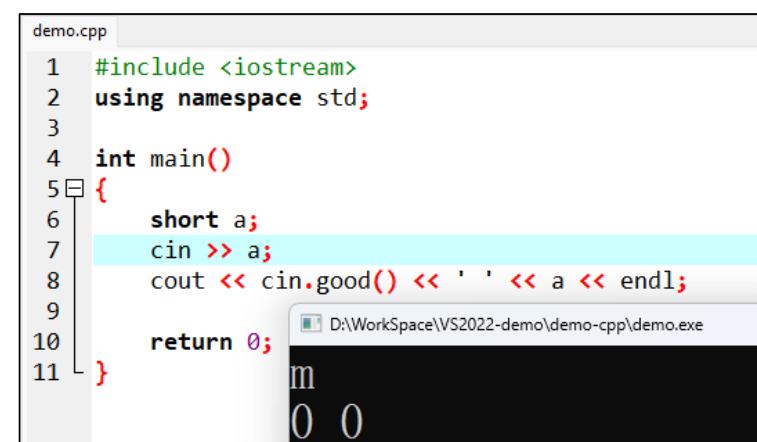


```
1 #include <iostream>
2 using namespace std;
3
4 int main()
5 {
6     short a;
7     cin >> a;
8     cout << cin.good() << ' ' << a << endl;
9
10 }  m
11      0 0
```

```
[root@RH9-DEV root]# cat test.cpp
#include <iostream>
using namespace std;
int main()
{
    short k;
    cin >> k;
    cout << "k=" << k << endl;
    cout << "cin.good()=" << cin.good() << endl;
    cout << "cin.fail()=" << cin.fail() << endl;
    return 0;
}
```

某Linux编译器

```
[root@RH9-DEV root]# ./test
m
k=16385
cin.good()=0
cin.fail()=1
```



```
1 #include <iostream>
2 using namespace std;
3
4 int main()
5 {
6     short a;
7     cin >> a;
8     cout << cin.good() << ' ' << a << endl;
9
10 }  m
11      0 0
```



## §. 习题课 (Week-04)

### 4、cin相关

- ★ cin的基本步骤
- ★ 如何理解合理且最长原则
- ★ 连续多个输入的理解
- ★ 不可信值的理解

- 只要good()不为1，值就不可信
- 只要good()为1则可信（无符号数读入负数问题）

```
#include <iostream>
using namespace std;
int main()
{
    unsigned short a;
    cin >> a;
    cout << cin.good() << ' ' << a << endl;
    return 0;
}
```

The screenshot shows a Microsoft Visual Studio IDE window. The code editor displays the following C++ code:

```
demo.cpp
#include <iostream>
using namespace std;
int main()
{
    unsigned short a;
    cin >> a;
    cout << cin.good() << ' ' << a << endl;
    return 0;
```

The output window shows the results of running the program. It displays two lines of text: "-12345" and "1 53191". A tooltip from the Microsoft Visual Studio debugger indicates "Microsoft Visual Studio 调试控制台".

1100 1111 1100 0111  
有符号理解: -12345  
无符号理解: 53191  
cin读入后按有符号转为二进制,  
赋值给无符号, 不超限即可 ( $-65535 \sim -1$ )



## §. 习题课 (Week-04)

### 5. 其他

#### ★ 除数/模数为0的问题

观察除数为变量/常量(包括符号常量)的不同表现

### §. 基础知识题

4、求复合赋值表达式的值（要求给出计算过程、每步计算结果及数据类型、对应的验证程序及结果截图，具体见下）

假设 `int a = 8, n = 13;`

D. `n %= a %= 4` 本题需要解释，为什么编译不报错，但运行无输出、返回代码为负值、且运行时间比7.ABC长  
(无法理解或说清楚原因的，给出合理猜测也可)

The screenshot shows the Microsoft Visual Studio IDE. The code editor displays the following C++ code:

```
demo.cpp
1 #include <iostream>
2 using namespace std;
3
4 int main()
5 {
6     int a = 8, n = 13;
7     cout << (n %= a %= 4) << endl;
8     return 0;
9 }
10
```

Below the code editor is the "Microsoft Visual Studio 调试控制台" (Debug Console). It shows the output of the program:

```
D:\WorkSpace\VS2022-demo\Debug\demo-cpp.exe (进程 34000)已退出, 代码为 -1073741676 (0xc0000094)。
按任意键关闭此窗口. . .
```

A red arrow points from the question text above to the exit code `-1073741676 (0xc0000094)` in the console window.



## §. 习题课 (Week-04)

### 5、其他

#### ★ 除数/模数为0的问题

观察除数为变量/常量(包括符号常量)的不同表现

为什么编译器处理不同?

The screenshot shows two instances of Microsoft Visual Studio comparing different ways to handle division by zero in C++.

**Left Window (VS2022):**

```
demo.cpp
1 #include <iostream>
2 #include <iomanip>
3 using namespace std;
4
5 int main()
6 {
7     int a = 0;
8
9     cout << 10/a << endl;
10
11    return 0;
12 }
```

**Right Window (VS2022):**

```
demo.cpp
1 #include <iostream>
2 #include <iomanip>
3 using namespace std;
4
5 int main()
6 {
7     const int a = 0;
8
9     cout << 10%a << endl;
10
11    return 0;
12 }
```

Both windows show the same error message in the output pane:

0:26...  
启动生成: 项目: demo-cpp, 配置: Debug Win32  
ace\VS2022-demo\demo-cpp\demo.cpp(9,12): error C2124: 被零除或对零求模  
对项目“demo-cpp.vcxproj”的操作 - 失败。  
生成: 0 成功, 1 失败, 0 最新, 0 已跳过  
生成于 0:26 完成, 耗时 00.535 秒

Red arrows point from the error messages in both panes to the specific lines of code causing the error: `10/a` in the left window and `10%a` in the right window.



## §. 习题课 (Week-04)

### 5. 其他

#### ★ 作业中部分数据的类型问题

问题：按课件说法，不加U/L等后缀的整型，默认应该是int型，然而下面两个例子都不是

### §. 基础知识题

3、求表达式的值（要求给出计算过程、每步计算结果及数据类型、对应的验证程序及结果截图）

F.  $\text{long}(2.8F + 3.3) * 2 + (\text{int})1.9 \% 7U - 'p' * 2UL$

```
demo.cpp
1 #include <iostream>
2 using namespace std;
3
4 int main()
5 {
6     cout << typeid(long(2.8F + 3.3) * 2 + (int)1.9 % 7U - 'p' * 2UL).name() << endl;
7     cout << typeid(long(2.8F + 3.3) * 2 + (int)1.9 % 7U).name() << endl;
8     return 0;
9 }
10
```

Microsoft Visual Studio 调试控制台

unsigned long  
unsigned long

按整型提升规则，第一个是UL  
但为什么第2个也是UL而不是L？

星期三 23:37

2451527-计算机-唐斌韬 LV6

#Q#

```
k5 = -0x90000000; // <-0x80000000
cout << k1 << (unsigned int)2415919104U
cout << k2 << 使用 Copilot 进行描述
cout << k3 <<

k5 = -2952790015; // <-0x80000000
cout << k1 << (long long)2952790015i64
cout << k2 << 使用 Copilot 进行描述
cout << k3 <<

k5 = -0x90000000LL; // <-0x80000000
cout << k1 << (long long)2415919104i64
cout << k2 << 使用 Copilot 进行描述
```

使用负号作用于一个数（这个数在int于uint之间）时，如果是这个数是十六进制，就会被解析为uint型导致负号无法作用。但这个数是十进制就会被解析成longlong，负号可以正确作用，请问这是编译器的规则吗



## §. 习题课 (Week-04)

### 5. 其他

#### ★ 作业中部分数据的类型问题

问题：按课件说法，不加U/L等后缀的整型，默认应该是int型，然而下面两个例子都不是

答案：整型不带后缀的情况下，有一个从小到大的**梯次适应**规则，超过小数据的范围则自动递增一级

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

int main()
{
    cout << typeid(0x80000000).name() << endl;
    cout << typeid(0x100000000).name() << endl;

    return 0;
}
```

demo.cpp

```
1 #include <iostream>
2 using namespace std;
3
4 int main()
5 {
6     cout << typeid(0x80000000).name() << endl;
7     cout << typeid(0x100000000).name() << endl;
8     return 0;
9 }
10
```

Microsoft Visual Studio 调试控制台

```
unsigned int
__int64
```

二进制, 八进制或十六进制底

- \* int
- \* unsigned int ←
- \* long int
- \* unsigned long int
- \* long long int (C++11起) ←
- \* unsigned long long int (C++11起)

字面量的类型		
后缀	十进制底	二进制, 八进制或十六进制底
(无后缀)	<ul style="list-style-type: none"><li>* int</li><li>* unsigned int</li><li>* long int</li><li>* long long int (C++11起)</li></ul>	<ul style="list-style-type: none"><li>* int</li><li>* unsigned int</li><li>* long int</li><li>* unsigned long int</li><li>* long long int (C++11起)</li><li>* unsigned long long int (C++11起)</li></ul>
u 或 U	<ul style="list-style-type: none"><li>* unsigned int</li><li>* unsigned long int</li><li>* unsigned long long int (C++11起)</li></ul>	<ul style="list-style-type: none"><li>* unsigned int</li><li>* unsigned long int</li><li>* unsigned long long int (C++11起)</li></ul>
l 或 L	<ul style="list-style-type: none"><li>* long int</li><li>* unsigned long int (C++11前)</li><li>* long long int (C++11起)</li></ul>	<ul style="list-style-type: none"><li>* long int</li><li>* unsigned long int</li><li>* long long int (C++11起)</li><li>* unsigned long long int (C++11起)</li></ul>
同时有 l/L 和 u/U	<ul style="list-style-type: none"><li>* unsigned long int</li><li>* unsigned long long int (C++11起)</li></ul>	<ul style="list-style-type: none"><li>* unsigned long int</li><li>* unsigned long long int (C++11起)</li></ul>
ll 或 LL	<ul style="list-style-type: none"><li>* long long int (C++11起)</li></ul>	<ul style="list-style-type: none"><li>* long long int (C++11起)</li><li>* unsigned long long int (C++11起)</li></ul>
同时有 ll/LL 和 u/U	<ul style="list-style-type: none"><li>* unsigned long long int (C++11起)</li></ul>	<ul style="list-style-type: none"><li>* unsigned long long int (C++11起)</li></ul>
z 或 Z	<ul style="list-style-type: none"><li>* std::size_t 的有符号版本 (C++23起)</li><li>* std::size_t (C++23起)</li></ul>	<ul style="list-style-type: none"><li>* std::size_t 的有符号版本 (C++23起)</li><li>* std::size_t (C++23起)</li></ul>
同时有 z/Z 和 u/U	<ul style="list-style-type: none"><li>* std::size_t (C++23起)</li></ul>	<ul style="list-style-type: none"><li>* std::size_t (C++23起)</li></ul>

若不具有**大小后缀**的(C++23起)整数字面量的值过大从而无法符合任何后缀/底组合所允许的类型，且编译器支持能表示该字面量的值的扩展整数类型(如 `_int128`)，则字面量可以被授予该扩展整数类型——否则程序非良构。



## §. 习题课 (Week-04)

5、其他

★ 作业中部分数据的类型问题

问题：按课件说法，不加U/L等后缀的整型，默认应该是int型，然而下面两个例子都不是

答案：整型不带后缀的情况下，有一个从小到大的**梯次适应**规则，超过小数据的范围则自动递增一级

再次强调：限于课时，课上所讲的只是一个简单规则，更复杂的表现并未进一步讨论，一旦出现不同，**编译器具体表现为准**