



## §. 基础知识题 – 进制转换

### 1. 十进制转二进制

A. 2023

$$\begin{array}{r}
 2 | 2023 \\
 2 | 1011 \quad 1 \\
 2 | 505 \quad 1 \\
 2 | 252 \quad 1 \\
 2 | 126 \quad 0 \\
 2 | 63 \quad 0 \\
 2 | 31 \quad 1 \\
 2 | 15 \quad 1 \\
 2 | 7 \quad 1 \\
 2 | 3 \quad 1 \\
 2 | 1 \quad 1 \\
 0
 \end{array}$$

$$(2023)_{10} = (11111100111)_2$$

B. 50758

$$\begin{array}{r}
 2 | 50758 \\
 2 | 25379 \quad 0 \\
 2 | 12689 \quad 1 \\
 2 | 6344 \quad 1 \\
 2 | 3172 \quad 0 \\
 2 | 1586 \quad 0 \\
 2 | 793 \quad 0 \\
 2 | 396 \quad 1 \\
 2 | 198 \quad 0 \\
 2 | 99 \quad 0 \\
 2 | 49 \quad 1 \\
 2 | 24 \quad 1 \\
 2 | 12 \quad 0 \\
 2 | 6 \quad 0 \\
 2 | 3 \quad 0 \\
 2 | 1 \quad 1 \\
 0
 \end{array}$$

$$(50758)_{10} = (1100011001000110)_2$$

C. 0.375

$$\begin{array}{r}
 0.375 \\
 \times 2 \\
 \hline
 0.750 \dots 0
 \end{array}$$

$$\begin{array}{r}
 0.75 \\
 \times 2 \\
 \hline
 1.50 \dots 1
 \end{array}$$

$$\begin{array}{r}
 0.5 \\
 \times 2 \\
 \hline
 1.0 \dots 1
 \end{array}$$

$$(0.375)_{10} = (0.011)_2$$

D. 8.0625

$$\begin{array}{r}
 2 | 8 \\
 2 | 4 \quad 0 \\
 2 | 2 \quad 0 \\
 2 | 1 \quad 0 \\
 0 \quad 1
 \end{array}$$

$$\begin{array}{r}
 0.0625 \\
 \times 2 \\
 \hline
 0.1250 \dots 0
 \end{array}$$

$$\begin{array}{r}
 0.125 \\
 \times 2 \\
 \hline
 0.250 \dots 0
 \end{array}$$

$$\begin{array}{r}
 0.25 \\
 \times 2 \\
 \hline
 0.50 \dots 0
 \end{array}$$

$$\begin{array}{r}
 0.5 \\
 \times 2 \\
 \hline
 1.0 \dots 1
 \end{array}$$

$$(8.0625)_{10} = (1000.0001)_2$$

整数部分:

小数部分:



## §. 基础知识题 – 进制转换

### 2. 二进制转十进制

A.  $10101110110$

$$\begin{aligned} &= 1 \times 2^{10} + 0 \times 2^9 + 1 \times 2^8 + 0 \times 2^7 + 1 \times 2^6 + 1 \times 2^5 \\ &\quad + 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 \\ &= 1024 + 0 + 256 + 0 + 64 + 32 + 16 + 0 + 4 + 2 + 0 \\ &= 1398 \end{aligned}$$

$$(10101110110)_2 = (1398)_{10}$$

B.  $1100011001000110$

$$\begin{aligned} &= 1 \times 2^{15} + 1 \times 2^{14} + 0 \times 2^{13} + 0 \times 2^{12} + 0 \times 2^{11} + 1 \times 2^{10} + 1 \times 2^9 + 0 \times 2^8 \\ &\quad + 0 \times 2^7 + 1 \times 2^6 + 0 \times 2^5 + 0 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 \\ &= 32768 + 16384 + 0 + 0 + 0 + 1024 + 512 + 0 + 0 + 64 + 0 + 0 + 0 + 4 + 2 + 0 \\ &= 50758 \end{aligned}$$

$$(1100011001000110)_2 = (50758)_{10}$$

C.  $1011.1101$

$$\begin{aligned} &= 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 + 1 \times 2^{-1} + 1 \times 2^{-2} + 0 \times 2^{-3} + 1 \times 2^{-4} \\ &= 8 + 0 + 2 + 1 + 0.5 + 0.25 + 0 + 0.0625 \\ &= 11.8125 \end{aligned}$$

$$(1011.1101)_2 = (11.8125)_{10}$$

D.  $0.10111101$

$$\begin{aligned} &= 0 \times 2^0 + 1 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3} + 1 \times 2^{-4} + 1 \times 2^{-5} + 1 \times 2^{-6} + 0 \times 2^{-7} + 1 \times 2^{-8} \\ &= 0 + 0.5 + 0 + 0.125 + 0.0625 + 0.03125 + 0.015625 + 0 + 0.00390625 \\ &= 0.73828125 \end{aligned}$$

$$(0.10111101)_2 = (0.73828125)_{10}$$



## §. 基础知识题 – 进制转换

### 3. 十进制与八进制的相互转换

(1) 十进制转八进制

A. 3905

$$\begin{array}{r} 3905 \\ 8 \boxed{488} \quad 1 \\ 8 \boxed{61} \quad 0 \\ 8 \boxed{7} \quad 5 \\ 0 \end{array}$$

$$(3905)_{10} = (7501)_8$$

(2) 八进制转十进制

A. 617032

$$\begin{aligned} 617032 &= 6 \times 8^5 + 1 \times 8^4 + 7 \times 8^3 + 0 \times 8^2 + 3 \times 8^1 + 2 \times 8^0 \\ &= 196608 + 4096 + 3584 + 0 + 24 + 2 \\ &= 204314 \end{aligned}$$

$$(617032)_8 = (204314)_{10}$$

B. 50758

$$\begin{array}{r} 50758 \\ 8 \boxed{6344} \quad 6 \\ 8 \boxed{793} \quad 0 \\ 8 \boxed{99} \quad 1 \\ 8 \boxed{12} \quad 3 \\ 8 \boxed{1} \quad 4 \\ 0 \quad 1 \end{array}$$

$$(50758)_{10} = (143106)_8$$

B. 143106

$$\begin{aligned} 143106 &= 1 \times 8^5 + 4 \times 8^4 + 3 \times 8^3 + 1 \times 8^2 + 0 \times 8^1 + 6 \times 8^0 \\ &= 32768 + 16384 + 1536 + 64 + 0 + 6 \\ &= 50758 \end{aligned}$$

$$(143106)_8 = (50758)_{10}$$



## §. 基础知识题 – 进制转换

### 4. 十进制与十六进制的相互转换

#### (1) 十进制转十六进制

A. 3905

$$\begin{array}{r} 16 \mid 3905 \\ 16 \quad \boxed{244} \quad 1 \\ 16 \quad \boxed{15} \quad 4 \\ \hline 0 \end{array}$$

$$(3905)_{10} = (F41)_{16}$$

#### (2) 十六进制转十进制

A. C35B8

$$\begin{aligned} C35B8 &= 12 \times 16^4 + 3 \times 16^3 + 5 \times 16^2 + 11 \times 16^1 + 8 \times 16^0 \\ &= 786432 + 12288 + 1280 + 176 + 8 \\ &= 800184 \end{aligned}$$

$$(C35B8)_{16} = (800184)_{10}$$

B. 50758

$$\begin{array}{r} 16 \mid 50758 \\ 16 \quad \boxed{31} \quad 2 \quad 6 \\ 16 \quad \boxed{19} \quad 4 \\ 16 \quad \boxed{12} \quad 6 \\ \hline 0 \quad 1 \end{array}$$

$$(50758)_{10} = (C646)_{16}$$

B. C646

$$\begin{aligned} C646 &= 12 \times 16^3 + 6 \times 16^2 + 4 \times 16^1 + 6 \times 16^0 \\ &= 49152 + 1536 + 64 + 6 \\ &= 50758 \end{aligned}$$

$$(C646)_{16} = (50758)_{10}$$



## §. 基础知识题 – 进制转换

### 5. 其他进制的相互转换

#### (1) 二进制转八进制

A. 10101110110

$$(10101110110)_2 = 10\ 101\ 110\ 110 = (2566)_8$$

B. 1100011001000110

$$(1100011001000110)_2 = 1\ 100\ 011\ 001\ 000\ 110 = (143106)_8$$

#### (2) 八进制转二进制

A. 617032

$$(617032)_8 = 110\ 001\ 111\ 000\ 011\ 010 = (11000111000011010)_2$$

B. 143106

$$(143106)_8 = 001\ 100\ 011\ 001\ 000\ 110 = (1100011001000110)_2$$

#### (3) 二进制转十六进制

A. 10101110110

$$(10101110110)_2 = 101\ 0111\ 0110 = (576)_{16}$$

B. 1100011001000110

$$(1100011001000110)_2 = 1100\ 0110\ 0100\ 0110 = (C646)_{16}$$



## §. 基础知识题 – 进制转换

### 5. 其他进制的相互转换

#### (4) 十六进制转二进制

A. C35B8

$$(C35B8)_{16} = 1100\ 0011\ 0101\ 1011\ 1000 = (11000011010110111000)_2$$

B. C646

$$(C646)_{16} = 1100\ 0110\ 0100\ 0110 = (1100011001000110)_2$$

#### (5) 八进制转十六进制

A. 617032

$$(617032)_8 = 110\ 001\ 111\ 000\ 011\ 010 = 0011\ 0001\ 1110\ 0001\ 1010 = (31E1A)_{16}$$

B. 143106

$$(143106)_8 = 001\ 100\ 011\ 001\ 000\ 110 = 1100\ 0110\ 0100\ 0110 = (C646)_{16}$$

#### (6) 十六进制转八进制

A. C35B8

$$(C35B8)_{16} = 1100\ 0011\ 0101\ 1011\ 1000 = 011\ 000\ 011\ 010\ 110\ 111\ 000 = (3032670)_8$$

B. C646

$$(C646)_{16} = 1100\ 0110\ 0100\ 0110 = 001\ 100\ 011\ 001\ 000\ 110 = (143106)_8$$



## §. 基础知识题 - 二进制补码

1. 十进制整数转二进制补码（仿照课件PDF的P. 19，写出具体步骤，包括绝对值、取反、+1）

格式要求：多字节时，每8bit中间加一个空格或-（例：“11010100 00110001”或“11010100-00110001”）

A. -122 （假设为1字节整数）

数值	-122
绝对值的二进制表示	1111010
原码	01111010
取反+1，得到补码	$\begin{array}{r} 10000101 \\ +) \quad 1 \\ \hline 10000110 \end{array}$



## §. 基础知识题 - 二进制补码

1. 十进制整数转二进制补码（仿照课件PDF的P. 19，写出具体步骤，包括绝对值、取反、+1）

格式要求：多字节时，每8bit中间加一个空格或-（例：“11010100 00110001”或“11010100-00110001”）

B. -244 （假设为2字节整数）

数值	-244
绝对值的二进制表示	11110100
原码	00000000 11110100
取反+1，得到补码	$\begin{array}{r} 11111111 \ 00001011 \\ +) \quad \quad \quad 1 \\ \hline 11111111 \ 00001100 \end{array}$



## §. 基础知识题 - 二进制补码

1. 十进制整数转二进制补码（仿照课件PDF的P. 19，写出具体步骤，包括绝对值、取反、+1）

格式要求：多字节时，每8bit中间加一个空格或-（例：“11010100 00110001”或“11010100-00110001”）

C. -244 （假设为4字节整数）

数值	-244
绝对值的二进制表示	11110100
原码	00000000 00000000 00000000 11110100
取反+1，得到补码	$\begin{array}{r} 11111111 & 11111111 & 11111111 & 00001011 \\ +) & & & \\ \hline & 11111111 & 11111111 & 11111111 & 00001100 \end{array}$

## §. 基础知识题 - 二进制补码



## 1. 十进制整数转二进制补码（仿照课件PDF的P. 19，写出具体步骤，包括绝对值、取反、+1）

格式要求：多字节时，每8bit中间加一个空格或-（例：“11010100 00110001” 或 “11010100-00110001”）

D. 本人学号逆序后取最多五位对应的int型十进制负数

数值	-85705
绝对值的二进制表示	10100111011001001
原码	00000000 00000001 01001110 11001001
取反+1，得到补码	$  \begin{array}{r}  11111111 11111110 10110001 00110110 \\  +) \\  \hline  11111111 11111110 10110001 00110111  \end{array}  $



## §. 基础知识题 - 二进制补码

2. 二进制补码转十进制整数（只考虑有符号数，写出具体步骤，包括-1、取反、绝对值、加负号）

格式要求：多字节时，每8bit中间加一个空格或-（例：“11010100 00110001”或“11010100-00110001”）

A. 1011 0101

补码-1	10110101 -) 1 ----- 10110100
取反，得到原码	01001011
绝对值的二进制表示	1001011
绝对值的十进制表示	75
加负号，得到数值	-75



## §. 基础知识题 - 二进制补码

2. 二进制补码转十进制整数（只考虑有符号数，写出具体步骤，包括-1、取反、绝对值、加负号）

格式要求：多字节时，每8bit中间加一个空格或-（例：“11010100 00110001”或“11010100-00110001”）

B. 1011 1001 1101 1010

补码-1	10111001 11011010 -) ----- 10111001 11011001
取反，得到原码	01000110 00100110
绝对值的二进制表示	100011000100110
绝对值的十进制表示	17958
加负号，得到数值	-17958



## §. 基础知识题 - 二进制补码

2. 二进制补码转十进制整数（只考虑有符号数，写出具体步骤，包括-1、取反、绝对值、加负号）

格式要求：多字节时，每8bit中间加一个空格或-（例：“11010100 00110001”或“11010100-00110001”）

C. 1101 1101 0110 0000 0110 1011 1001 0000

补码-1	11011101 01100000 01101011 10010000 -) -----
取反，得到原码	11011101 01100000 01101011 10001111
绝对值的二进制表示	00100010 10011111 10010100 01110000
绝对值的十进制表示	10001010011111001010001110000 580883568
加负号，得到数值	-580883568



## §. 基础知识题 - 二进制补码

2. 二进制补码转十进制整数（只考虑有符号数，写出具体步骤，包括-1、取反、绝对值、加负号）

格式要求：多字节时，每8bit中间加一个空格或-（例：“11010100 00110001”或“11010100-00110001”）

D. 本人学号逆序后取最多五位对应的int型十进制负数的二进制补码形式

补码-1	11111111 11111110 10110001 00110111 -) ----- 11111111 11111110 10110001 00110110
取反，得到原码	00000000 00000001 01001110 11001001
绝对值的二进制表示	10100111011001001
绝对值的十进制表示	85705
加负号，得到数值	-85705

## § . 基础知识题 - 字符串长度



1. 求出下列字符串的长度

A. "\bvt\tnc\4921\x3fr\2a'\r\v\a\f" =21

B. "\19\x2f\43\x8x\383\x65\012\xd5\231\xe3\1325\x6a" =17



## §. 基础知识题 - 字符串长度

1. 求出下列字符串的长度

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

int main()
{
    cout << "2250758" << endl;
    cout << strlen("\23456f") << endl;
    cout << strlen("\43456f") << endl;
    return 0;
}
```

The screenshot shows a code editor window titled "My Project.cpp". The code contains three calls to `strlen()` with octal escape sequences: "\23456f", "\43456f", and "\3456f". The third call triggers a compilation error: "error C2022: “284”：对字符来说太大" (error C2022: "284": too big for character). A tooltip also indicates a warning: "[Warning] octal escape sequence out of range".

```
My Project.cpp + x
My Project
1 #include <iostream>
2 using namespace std;
3
4 int main() error C2022: “284”：对字符来说太大
5 {
6     cout << "2250758" << endl;
7     cout << strlen("\23456f") << endl;
8     cout << strlen("\43456f") << endl;
9     return 0;
10 }
```

C. 运行上面的程序，贴含本人学号的源程序+编译器的错误信息截图

观察编译信息，得到结论如下：

- 1、转义符\后的合法8进制数>3个，则转义符\与其之后的3个合法8进制数组成1个转义字符。
- 2、转义符\后的合法8进制数≤3个但超出上限377，则出现error C2022的提示，转义符\与其之后的3个合法8进制数组成的转义字符不能表示出某一个字符。

编译提示中的那个数字是怎么来的？

$$(434)_8 = (284)_{10}$$



## §. 基础知识题 - 字符串长度

1. 求出下列字符串的长度

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

int main()
{
    cout << "2250758" << endl;
    cout << strlen("\x23") << endl;
    cout << strlen("\x234") << endl;
    return 0;
}
```

```
My Project.cpp
```

```
1 #include <iostream>
2 using namespace std;
3
4 int main() error C2022: “564”：对字符来说太大
5 {
6     cout << "2250758" << endl;
7     cout << strlen("\x23") << endl;
8     cout << strlen("\x234") << endl;
9     return 0;
10 }
```

[Warning] hex escape sequence out of range

D. 运行上面的程序，贴含本人学号的源程序+编译器的错误信息截图

观察编译信息，得到结论如下：

- 1、转义符\x后的合法16进制数>2个，则转义符\x与其之后的合法16进制数组成1个转义字符。  
编译提示中的那个数值是怎么来的?  
 $(234)_{16} = (564)_{10}$
- 2、综合CD，在用转义符表示8/16进制时，超过限定的长度的错误处理是不一致的。



## §. 基础知识题 - 字符串长度

1. 求出下列字符串的长度

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

int main()
{
    cout << "2250758" << endl;
    cout << strlen("\9876") << endl;
    cout << strlen("\*321") << endl;
    return 0;
}
```

```
My Project.cpp + x
My Project
1 #include <iostream>
2 using namespace std;
3
4 int main() warning C4129: “\9”：不可识别的字符转义序列
5 { warning C4129: “\*”：不可识别的字符转义序列
6     cout << "2250758" << endl;
7     cout << strlen("\9876") << endl;
8     cout << strlen("\*321") << endl;
9 }
10
[Warning] unknown escape sequence: '\9'
[Warning] unknown escape sequence: '\*'
```

E. 运行上面的程序，贴含本人学号的源程序+编译器的错误信息截图

观察编译信息，得到结论如下：

- 1、转义符\后直接跟非法的8进制，则’\非法的8进制’是不可识别的字符转义序列，转义符\被忽略，其含义等价于’非法的8进制’。
- 2、对两个strlen的输出结果进行分析（合理猜测）  
’\9’是不可识别的字符转义序列，转义符\被忽略，其含义等价于’9’，故字符串”\9876”的长度为4；  
’\\*’是不可识别的字符转义序列，转义符\被忽略，其含义等价于’\*’，故字符串”\\*321”的长度为4。



## §. 基础知识题 - 字符串长度

1. 求出下列字符串的长度

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

int main()
{
    cout << "2250758" << endl;
    cout << strlen("\xg231") << endl;
    cout << strlen("\x*231") << endl;
    return 0;
}
```

The screenshot shows a code editor window titled "My Project.cpp". The code is as follows:

```
My Project.cpp + x
My Project
1 #include <iostream>
2 using namespace std;
3
4 int main()
5 {
6     cout << "2250758" << endl;
7     cout << strlen("\xg231") << endl;
8     cout << strlen("\x*231") << endl;
9     return 0;
10 }
```

Errors are highlighted in red and shown in a tooltip:

- Line 4: `error C2153: 整数文本必须至少具有一位数`
- Line 7: `error C2153: 整数文本必须至少具有一位数`
- Line 8: `[Error] \x used with no following hex digits`
- Line 8: `[Error] \x used with no following hex digits`

F. 运行上面的程序，贴含本人学号的源程序+编译器的错误信息截图

观察编译信息，得到结论如下：

- 1、转义符\x后直接跟非法的16进制，则出现error C2153的提示，十六进制常量必须至少有一个十六进制数字，\x后面必须至少有一个十六进制数字。
- 2、综合EF，在用转义符表示8/16进制时，直接跟非法字符的错误处理是不一致的。



## §. 基础知识题 - 变量及表达式求值

1. 给出下列程序段中变量b的值

```
A. short a=32750;  
short b=a+24;
```

Step1:  $b = a + 24$ , 得b二进制补码形式

$$\begin{array}{r} a = \textcolor{red}{00000000} \ 00000000 \ 01111111 \ 11101110 \rightarrow a \ (\text{红色表示整型提升的填充位}) \\ +) \ 24 = 00000000 \ 00000000 \ 00000000 \ 00011000 \rightarrow 24 \\ \hline \end{array}$$

$$\begin{array}{r} 00000000 \ 00000000 \ 10000000 \ 00000110 \rightarrow a+24 \ (\text{int型}) \\ b = \textcolor{red}{00000000} \textcolor{red}{-} \textcolor{red}{00000000} \ 10000000 \ 00000110 \rightarrow b=a+24 \ (\text{二进制补码形式, 删除线表示丢弃的位数}) \end{array}$$

Step2: 求b的十进制表示

(1) 减一  $10000000 \ 00000110$   
     $-) \ 00000000 \ 00000001$

$$\hline \ 10000000 \ 00000101$$

- (2) 取反  $01111111 \ 11111010$   
(3) 绝对值 32762 (十进制表示形式)  
(4) 加负号 -32762 (十进制表示形式)



## §. 基础知识题 - 变量及表达式求值

1. 给出下列程序段中变量b的值

```
B. unsigned short a=65520;  
short b=a;
```

Step1: b=a, 得b二进制补码形式

a = **00000000 00000000** 11111111 11110000 -> a (红色表示整型提升的填充位)

a = 00000000 00000000 11111111 11110000 -> a (int型)

b = **00000000-00000000** 11111111 11110000 -> b=a (二进制补码形式, 删除线表示丢弃的位数)

Step2: 求b的十进制表示

$$\begin{array}{r} (1) \text{ 减一} \quad 11111111 \ 11110000 \\ -) \quad 00000000 \ 00000001 \\ \hline \end{array}$$

11111111 11101111

- (2) 取反 00000000 00010000
- (3) 绝对值 16 (十进制表示形式)
- (4) 加负号 -16 (十进制表示形式)

## §. 基础知识题 - 变量及表达式求值



1. 给出下列程序段中变量b的值

```
C. short a=-4095;  
    int b=a;
```

Step1: b=a, 得b二进制补码形式

a = 11111111 11111111 11110000 00000001 → a (红色表示整型提升的填充位)

a = 11111111 11111111 11110000 00000001 → a (int型)

b = 11111111 11111111 11110000 00000001 → b=a (二进制补码形式)

Step2: 求b的十进制表示

(1) 减一    11111111 11111111 11110000 00000001  
        -) 00000000 00000000 00000000 00000001

---

              11111111 11111111 11110000 00000000

(2) 取反    00000000 00000000 00001111 11111111

(3) 绝对值 4095 (十进制表示形式)

(4) 加负号 -4095 (十进制表示形式)



## §. 基础知识题 - 变量及表达式求值

1. 给出下列程序段中变量b的值

```
D. unsigned short a=65520;  
long long int b=a;
```

Step1: b=a, 得b二进制补码形式

a = 00000000 00000000 11111111 11110000 → a  
(红色表示整型提升的填充位)

a = 00000000 00000000 00000000 00000000 00000000 00000000 11111111 11110000 → a  
(long long型)

b = 00000000 00000000 00000000 00000000 00000000 00000000 11111111 11110000 → b=a  
(二进制补码形式)

Step2: 求b的十进制表示

$$(111111111110000)_2 = (65520)_{10}$$



## §. 基础知识题 - 变量及表达式求值

1. 给出下列程序段中变量b的值

```
E. long long int a=4207654321;  
    int b=a;
```

Step1: b=a, 得b二进制补码形式

a = 00000000 00000000 00000000 00000000 1111010 11001011 10110101 10110001 -> a  
a = ~~00000000 00000000 00000000 00000000~~ 1111010 11001011 10110101 10110001 -> a  
 (int型, 删除线表示丢弃的位数)  
b = 1111010 11001011 10110101 10110001 -> b=a  
 (二进制补码形式)

Step2: 求b的十进制表示

(1) 减一 1111010 11001011 10110101 10110001  
 -) 00000000 00000000 00000000 00000001

-----  
1111010 11001011 10110101 10110000

(2) 取反 00000101 00110100 01001010 01001111

(3) 绝对值 87312975 (十进制表示形式)

(4) 加负号 -87312975 (十进制表示形式)

## §. 基础知识题 – 变量及表达式求值



1. 给出下列程序段中变量b的值

```
F. long a=-4207654321;  
     unsigned short b=a;
```

Step1：确定-4207654321的类型

-4207654321为有符号数，long型的数值范围为-2147483648~+2147483647，long long型的数值范围为-9223372036854775808~+9223372036854775807，-4207654321超出了long型数值范围的下限-2147483648，故-4207654321的类型为long long型。通过“`cout << typeid(-4207654321).name() << endl;`”的编译结果为“int64”对-4207654321的类型为long long型进行了验证。

Step2: 确定a的值

```

-4207654321 = 11111111 11111111 11111111 11111111 00000101 00110100 01001010 01001111
    a = 11111111 11111111 11111111 11111111 00000101 00110100 01001010 01001111
                                (long型, 删除线表示丢弃的位数)
    a =                               00000101 00110100 01001010 01001111
                                (二进制补码形式)

```

Step3:  $b=a$ , 得 $b$ 二进制补码形式

a = 00000101 00110100 01001010 01001111 -> a  
a = ~~00000101~~ ~~00110100~~ 01001010 01001111 -> a (unsigned short型, 删除线表示丢弃的位数)  
b = 01001010 01001111 -> b (二进制补码形式)

Step4: 求b的十进制表示

$$(100101001001111)_2 = (19023)_{10}$$



## §. 基础知识题 - 变量及表达式求值

2. 用栈方式给出下列表达式的求解过程

A.  $21 / 2 + 47 \% 3 - 1.3 + 3.5 * 2$

表达式一共有6个运算符，因此计算的6个步骤分别是（本页不需要画栈）：

步骤①： $21 / 2 \Rightarrow$  式1

步骤②： $47 \% 3 \Rightarrow$  式2

步骤③： $式1 + 式2 \Rightarrow$  式3

步骤④： $式3 - 1.3 \Rightarrow$  式4

步骤⑤： $3.5 * 2 \Rightarrow$  式5

步骤⑥： $式4 + 式5 \Rightarrow$  结果



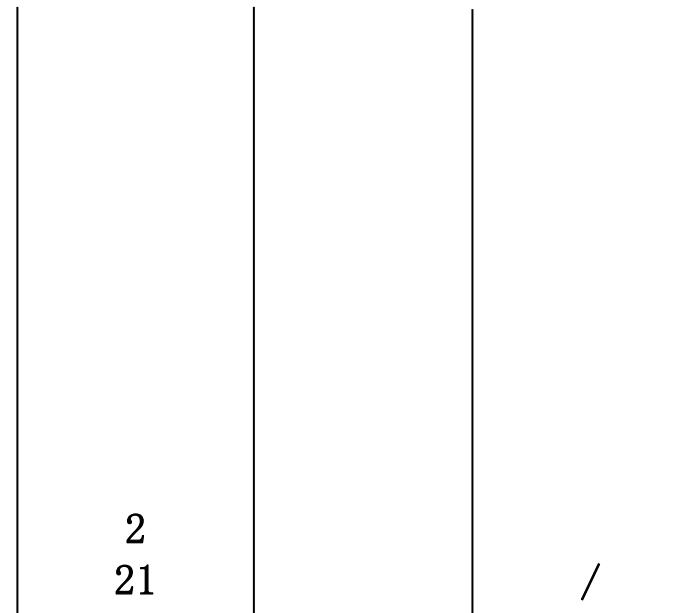
## §. 基础知识题 – 变量及表达式求值

2. 用栈方式给出下列表达式的求解过程

A.  $21 / 2 + 47 \% 3 - 1.3 + 3.5 * 2$



目前准备进栈的运算符如箭头所示，画出当前运算数栈和运算符栈的状态（本页需要画栈）



要进栈的(+)低于栈顶的(/)，先计算



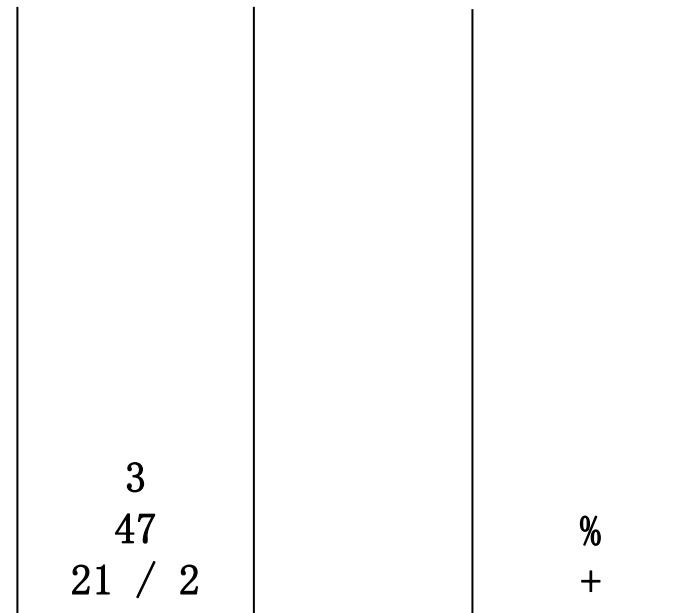
## §. 基础知识题 – 变量及表达式求值

2. 用栈方式给出下列表达式的求解过程

A.  $21 / 2 + 47 \% 3 - 1.3 + 3.5 * 2$



目前准备进栈的运算符如箭头所示，画出当前运算数栈和运算符栈的状态（本页需要画栈）



要进栈的(-)低于栈顶的(%), 先计算



## §. 基础知识题 - 变量及表达式求值

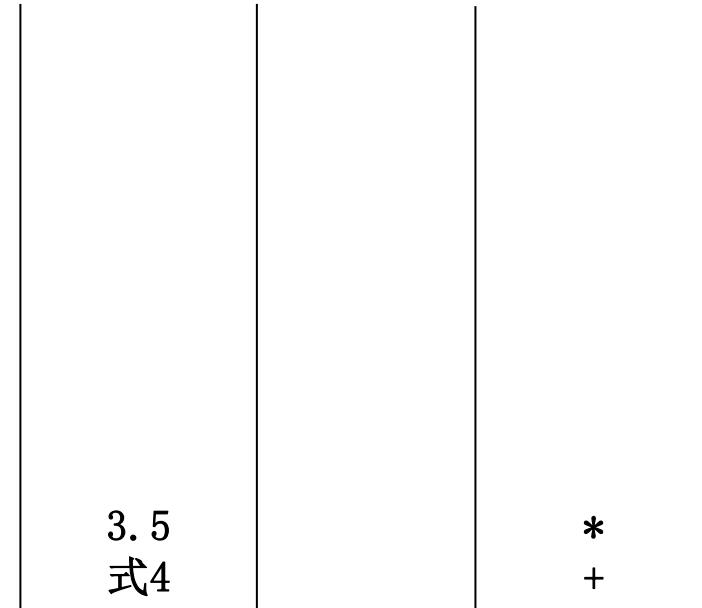
2. 用栈方式给出下列表达式的求解过程

A.  $21 / 2 + 47 \% 3 - 1.3 + 3.5 * 2$



目前准备进栈的运算符如箭头所示，画出当前运算数栈和运算符栈的状态（本页需要画栈）

式1:  $21 / 2$   
式2:  $47 \% 3$   
式3: 式1 + 式2  
式4: 式3 - 1.3



\*进栈 要进栈的(\*)高于栈顶的(+)



## §. 基础知识题 - 变量及表达式求值

2. 用栈方式给出下列表达式的求解过程

B.  $a = 3 * 5, a = b = 6 * 4$  (假设所有变量均为int型)

表达式一共有6个运算符，因此计算的6个步骤分别是：

步骤①:  $3 * 5 \Rightarrow$  式1

步骤②:  $a =$  式1  $\Rightarrow$  式2

步骤③:  $6 * 4 \Rightarrow$  式3

步骤④:  $b =$  式3  $\Rightarrow$  式4

步骤⑤:  $a =$  式4  $\Rightarrow$  式5

步骤⑥: 式2, 式5



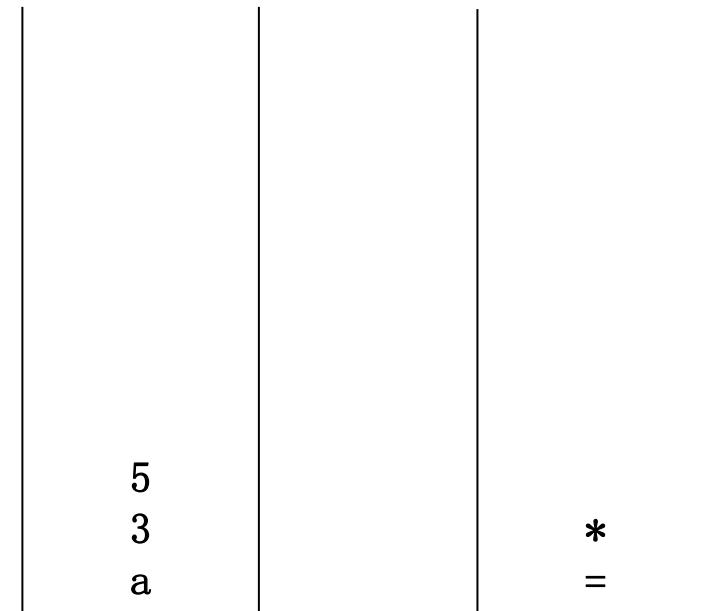
## §. 基础知识题 - 变量及表达式求值

2. 用栈方式给出下列表达式的求解过程

B.  $a = 3 * 5, a = b = 6 * 4$  (假设所有变量均为int型)



目前准备进栈的运算符如箭头所示，画出当前运算数栈和运算符栈的状态（本页需要画栈）



要进栈的(,)低于栈顶的(\*), 先计算



## §. 基础知识题 - 变量及表达式求值

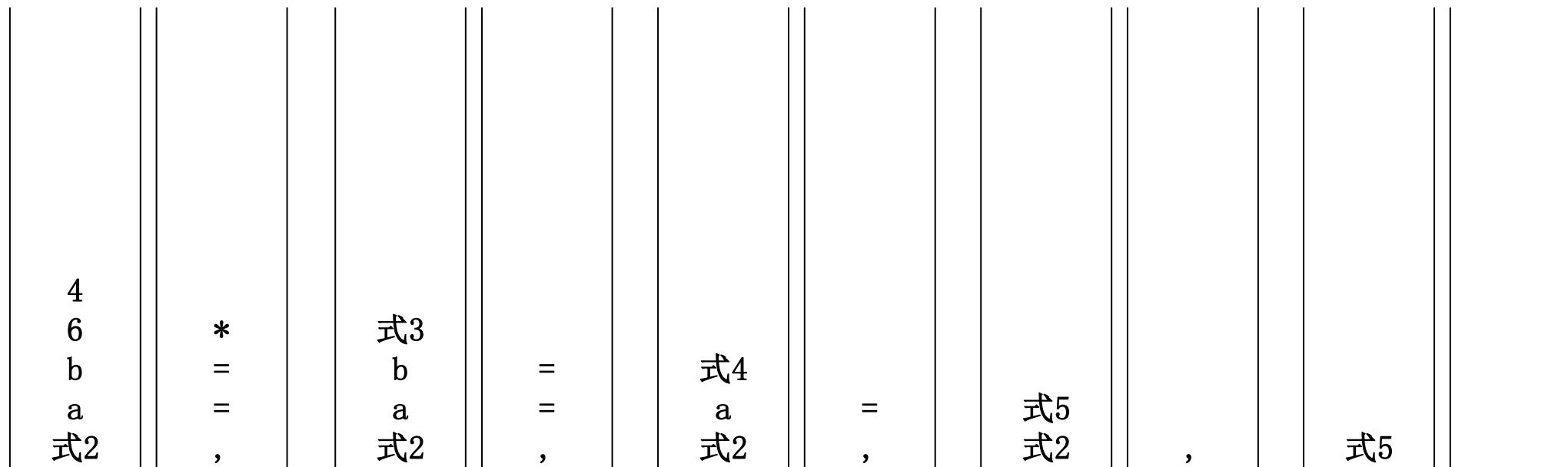
2. 用栈方式给出下列表达式的求解过程

B.  $a = 3 * 5, a = b = 6 * 4$  (假设所有变量均为int型)



目前已分析到整个表达式的尾部，画出从当前栈的状态到整个表达式分析完成的整个过程

- 步骤①:  $3 * 5 \Rightarrow$  式1
- 步骤②:  $a =$  式1  $\Rightarrow$  式2
- 步骤③:  $6 * 4 \Rightarrow$  式3
- 步骤④:  $b =$  式3  $\Rightarrow$  式4
- 步骤⑤:  $a =$  式4  $\Rightarrow$  式5
- 步骤⑥: 式2, 式5



计算  $6 * 4$

计算  $b =$  式3

计算  $a =$  式4

计算 式2, 式5

求值完成



## §. 基础知识题 - 变量及表达式求值

2. 用栈方式给出下列表达式的求解过程

C.  $a + (b - 3 * (a + c) - 2) \% 3$  (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:

步骤①:  $a + c \Rightarrow$  式1

步骤②:  $3 * \text{式1} \Rightarrow$  式2

步骤③:  $b - \text{式2} \Rightarrow$  式3

步骤④:  $\text{式3} - 2 \Rightarrow$  式4

步骤⑤:  $\text{式4 \% 3} \Rightarrow$  式5

步骤⑥:  $a + \text{式5}$



## §. 基础知识题 - 变量及表达式求值

2. 用栈方式给出下列表达式的求解过程

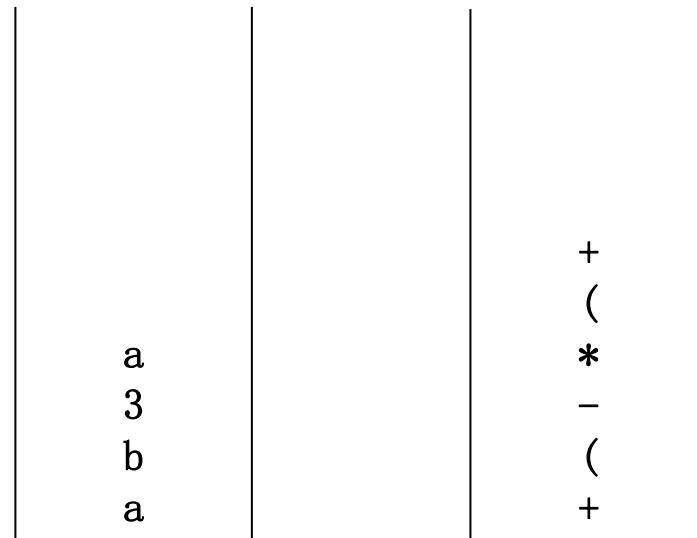
C.  $a + (b - 3 * (a + c) - 2) \% 3$  (假设所有变量均为int型)

(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)

目前准备进栈的运算符如箭头所示, 画出当前运算数栈和运算符栈的状态 (本页需要画栈)



+进栈 要进栈的(+)高于栈顶的()



## §. 基础知识题 - 变量及表达式求值

2. 用栈方式给出下列表达式的求解过程



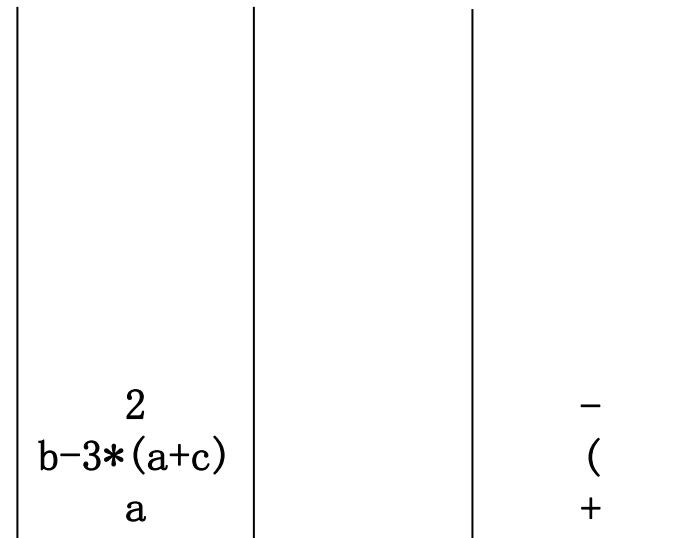
C.  $a + (b - 3 * (a + c) - 2) \% 3$  (假设所有变量均为int型)

(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)

目前准备进栈的运算符如箭头所示, 画出当前运算数栈和运算符栈的状态 (本页需要画栈)



要进栈的())低于栈顶的(-), 先计算



## §. 基础知识题 - 变量及表达式求值

2. 用栈方式给出下列表达式的求解过程

C.  $a + (b - 3 * (a + c) - 2) \% 3$  (假设所有变量均为int型)

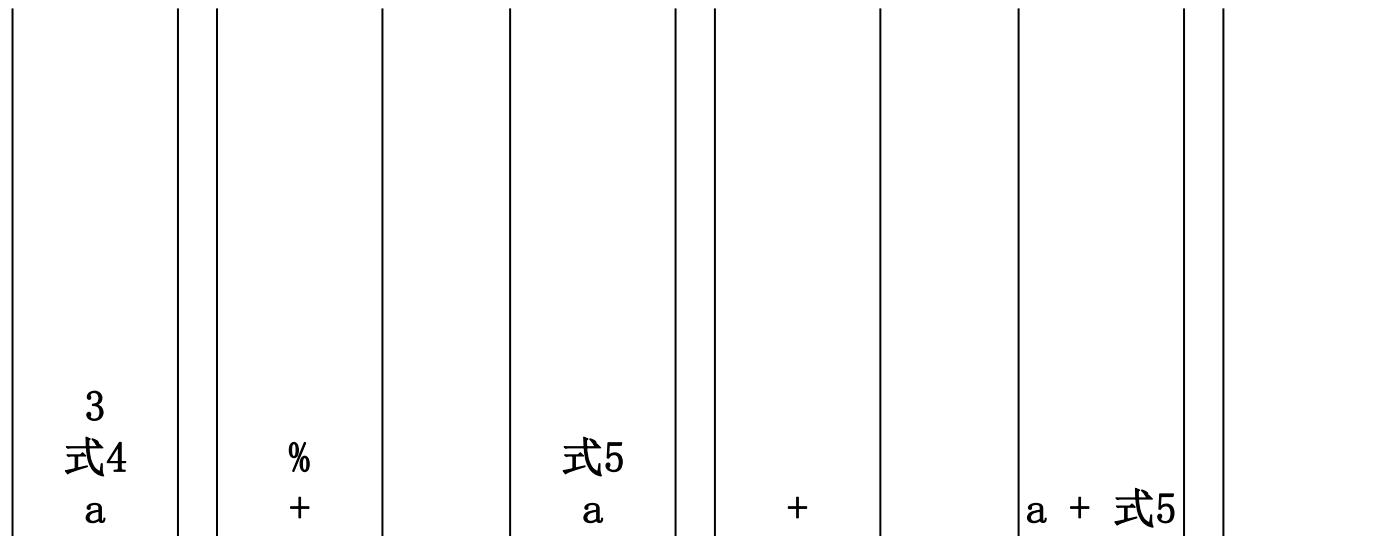
(本题提示: 将左右小括号分开处理,

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)

目前已分析到整个表达式的尾部, 画出从当前栈的状态到整个表达式分析完成的整个过程

步骤①:  $a + c \Rightarrow$  式1  
步骤②:  $3 * \text{式1} \Rightarrow$  式2  
步骤③:  $b - \text{式2} \Rightarrow$  式3  
步骤④:  $\text{式3} - 2 \Rightarrow$  式4  
步骤⑤:  $\text{式4 \% 3} \Rightarrow$  式5  
步骤⑥:  $a + \text{式5}$



计算 式4 \% 3

计算 a + 式5

求值完成

## § . 基础知识题 - 变量及表达式求值



### 3. 求表达式的值

A.  $a = 2 * 3, a = b = 5 * 7$  (写验证程序时, 假设所有变量均为int型)

- (1)  $2 * 3 \Rightarrow 6 \text{ int型}$
- (2)  $a = 2 * 3 \Rightarrow 6 \text{ int型}$
- (3)  $5 * 7 \Rightarrow 35 \text{ int型}$
- (4)  $b = 5 * 7 \Rightarrow 35 \text{ int型}$
- (5)  $a = b = 5 * 7 \Rightarrow 35 \text{ int型}$
- (6)  $a = 2 * 3, a = b = 5 * 7 \Rightarrow 35 \text{ int型}$

The screenshot shows a Microsoft Visual Studio interface. The code editor window is titled "My Project.cpp" and contains the following code:

```
#include <iostream>
using namespace std;
int main()
{
    int a, b;
    cout << (a = 2 * 3, a = b = 5 * 7) << endl;
    cout << typeid(a = 2 * 3, a = b = 5 * 7).name() << endl;
    return 0;
```

The Microsoft Visual Studio 调试控制台 (Debug Output) window at the bottom shows the output:

```
35
int
```

## § . 基础知识题 - 变量及表达式求值



### 3. 求表达式的值

B.  $a - (b + 4 * (b + c) / 3) \% 5$  (写验证程序时, 假设所有变量均为int型, int a=1, b=2, c=3; )

(1) $b + c$	=> 5	int型
(2) $(b + c)$	=> 5	int型
(3) $4 * (b + c)$	=> 20	int型
(4) $4 * (b + c) / 3$	=> 6	int型
(5) $b + 4 * (b + c) / 3$	=> 8	int型
(6) $(b + 4 * (b + c) / 3)$	=> 8	int型
(7) $(b + 4 * (b + c) / 3) \% 5$	=> 3	int型
(8) $a - (b + 4 * (b + c) / 3) \% 5$	=> -2	int型

```
My Project.cpp  x
My Project  (全局范围)

1 #include <iostream>
2 using namespace std;
3
4 int main()
5 {
6     int a = 1, b = 2, c = 3;
7     cout << (a - (b + 4 * (b + c) / 3) % 5) << endl;
8     cout << typeid(a - (b + 4 * (b + c) / 3) % 5).name() << endl;
9     return 0;
10 }
11

Microsoft Visual Studio 调试控制台
-2
int
```

## § . 基础知识题 - 变量及表达式求值



### 3. 求表达式的值

C.  $2.5 * 4\text{UL} + 7\text{U} * 5\text{ULL} - 'x'$

- (1)  $2.5 * 4\text{UL}$  => 10 double型
- (2)  $7\text{U} * 5\text{ULL}$  => 35 unsigned long long型
- (3)  $2.5 * 4\text{UL} + 7\text{U} * 5\text{ULL}$  => 45 double型
- (4)  $2.5 * 4\text{UL} + 7\text{U} * 5\text{ULL} - 'x'$  => -75 double型

The screenshot shows a Microsoft Visual Studio interface. The code editor window is titled "My Project.cpp" and contains the following C++ code:

```
1 #include <iostream>
2 using namespace std;
3
4 int main()
5 {
6     cout << (2.5 * 4UL + 7U * 5ULL - 'x') << endl;
7     cout << typeid(2.5 * 4UL + 7U * 5ULL - 'x').name() << endl;
8     return 0;
9 }
10
```

The code uses decltype to determine the type of the expression  $2.5 * 4\text{UL} + 7\text{U} * 5\text{ULL} - 'x'$ . The output window, titled "Microsoft Visual Studio 调试控制台" (Debug Console), displays the results:

```
-75
double
```

## §. 基础知识题 - 变量及表达式求值



### 3. 求表达式的值

D.  $3\text{LU} \% 7 + 13\text{LL} \% 3 + 3.5\text{F}$

- (1)  $3\text{LU} \% 7$   $\Rightarrow 3$  unsigned long型
- (2)  $13\text{LL} \% 3$   $\Rightarrow 1$  long long型
- (3)  $3\text{LU} \% 7 + 13\text{LL} \% 3$   $\Rightarrow 4$  long long型
- (4)  $3\text{LU} \% 7 + 13\text{LL} \% 3 + 3.5\text{F}$   $\Rightarrow 7.5$  float型

The screenshot shows a Microsoft Visual Studio interface. The code editor window is titled "My Project.cpp" and contains the following code:

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

int main()
{
    cout << (3LU % 7 + 13LL % 3 + 3.5F) << endl;
    cout << typeid(3LU % 7 + 13LL % 3 + 3.5F).name() << endl;
    return 0;
}
```

The code uses `3LU` and `13LL` to demonstrate integer promotion rules. The first two terms result in `unsigned long` and `long long` types respectively, which are then promoted to a common type for addition. The result is a `float` type, as indicated by the question.

The Microsoft Visual Studio debug console window at the bottom shows the output:

```
7.5
float
```

# § . 基础知识题 - 变量及表达式求值



## 3. 求表达式的值

E.  $3.2 + 11 \% 5 * \text{static\_cast}<\text{unsigned long}>(1.8F + 2LL) \% 2 * 3.2F$

- |                                                                                         |        |                |
|-----------------------------------------------------------------------------------------|--------|----------------|
| (1) $11 \% 5$                                                                           | => 1   | int型           |
| (2) $1.8F + 2LL$                                                                        | => 3.8 | float型         |
| (3) $\text{static\_cast}<\text{unsigned long}>(1.8F + 2LL)$                             | => 3   | unsigned long型 |
| (4) $11 \% 5 * \text{static\_cast}<\text{unsigned long}>(1.8F + 2LL)$                   | => 3   | unsigned long型 |
| (5) $11 \% 5 * \text{static\_cast}<\text{unsigned long}>(1.8F + 2LL) \% 2$              | => 1   | unsigned long型 |
| (6) $11 \% 5 * \text{static\_cast}<\text{unsigned long}>(1.8F + 2LL) \% 2 * 3.2F$       | => 3.2 | float型         |
| (7) $3.2 + 11 \% 5 * \text{static\_cast}<\text{unsigned long}>(1.8F + 2LL) \% 2 * 3.2F$ | => 6.4 | double型        |

The screenshot shows a Microsoft Visual Studio IDE window. The title bar says "My Project.cpp" and the tab bar has "My Project". The code editor contains the following C++ code:

```
1 #include <iostream>
2 using namespace std;
3
4 int main()
5 {
6     cout << (3.2 + 11 % 5 * static_cast<unsigned long>(1.8F + 2LL) % 2 * 3.2F) << endl;
7     cout << typeid(3.2 + 11 % 5 * static_cast<unsigned long>(1.8F + 2LL) % 2 * 3.2F).name() << endl;
8     return 0;
9 }
10
```

The output window at the bottom shows the results:

```
Microsoft Visual Studio 调试控制台
6.4
double
```



## § . 基础知识题 - 变量及表达式求值

### 3. 求表达式的值

F.  $\text{long}(3.8 + 1.3) / 2 + (\text{int})3.9 \% 7\text{LU} - 'G' * 3\text{L}$

(1) $3.8 + 1.3$	=> 5.1	double型
(2) $\text{long}(3.8 + 1.3)$	=> 5	long型
(3) $\text{long}(3.8 + 1.3) / 2$	=> 2	long型
(4) $(\text{int})3.9$	=> 3	int型
(5) $(\text{int})3.9 \% 7\text{LU}$	=> 3	unsigned long型
(6) $\text{long}(3.8 + 1.3) / 2 + (\text{int})3.9 \% 7\text{LU}$	=> 5	unsigned long型
(7) $'G' * 3\text{L}$	=> 213	long型
(8) $\text{long}(3.8 + 1.3) / 2 + (\text{int})3.9 \% 7\text{LU} - 'G' * 3\text{L}$	=> 4294967088	unsigned long型
$-208(\text{long}) = 11111111 \ 11111111 \ 11111111 \ 00110000$ (二进制补码形式)		= 4294967088 (unsigned long型)

The screenshot shows the Microsoft Visual Studio IDE interface. The code editor window displays a file named "My Project.cpp" with the following content:

```
1 #include <iostream>
2 using namespace std;
3
4 int main()
5 {
6     cout << (long(3.8 + 1.3) / 2 + (int)3.9 % 7LU - 'G' * 3L) << endl;
7     cout << typeid(long(3.8 + 1.3) / 2 + (int)3.9 % 7LU - 'G' * 3L).name() << endl;
8     return 0;
9 }
10
```

An error message is visible in the status bar: "C26454 算术溢出: “-”操作在编译时生成负的无符号结果(io.5.)" (Arithmetic overflow: the subtraction operator generates a negative unsigned result during compilation (io.5.)).

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

```
4294967088
unsigned long
```



## §. 基础知识题 - 变量及表达式求值

### 4. 求复合赋值表达式的值

假设 `int a = 3, n = 7;`

A. `a += a - n`

(1) `a - n` (记作①)

`a = 3 n = 7`

$\Rightarrow a = a + \textcircled{1}$

(2) `a + ①`

`a = 3 n = 7`

和-1存放在中间变量中

(3) `a = 和`

`a = -1 n = 7`

The screenshot shows a Microsoft Visual Studio interface. The code editor window is titled "My Project.cpp" and contains the following code:

```
1 #include <iostream>
2 using namespace std;
3
4 int main()
5 {
6     int a = 3, n = 7;
7     a += a - n;
8     cout << a << " " << n << endl;
9
10 }
```

To the right of the code editor is a "Microsoft Visual Studio 调试控制台" (Debug Console) window. The console output shows the results of the program execution:

```
-1 7
```



## §. 基础知识题 - 变量及表达式求值

### 4. 求复合赋值表达式的值

假设 `int a = 3, n = 7;`

B. `n += a += 5`

`a += 5` (记作①)

$\Rightarrow a = a + 5$

(1)  $a + 5$

(2)  $a =$  第一个和

`n += ①`

$\Rightarrow n = n + ①$

(3)  $n + ①$

(4)  $n =$  第二个和

$a = 3 \quad n = 7$

$a = 8 \quad n = 7$

第一个和8存放在中间变量中

$a = 8 \quad n = 7$

$a = 8 \quad n = 15$

第二个和15存放在中间变量中

```
My Project.cpp
My Project
1 #include <iostream>
2 using namespace std;
3
4 int main()
5 {
6     int a = 3, n = 7;
7     n += a += 5;
8     cout << a << " " << n << endl;
9
10 }
```

Microsoft Visual Studio 调试控制台  
8 15



## §. 基础知识题 - 变量及表达式求值

### 4. 求复合赋值表达式的值

假设 int a = 3;

C.  $a += a \% a -= a$

$a -= a$  (记作①)

$\Rightarrow a = a - a$

(1)  $a - a$

(2)  $a = \text{差}$

$a \% = ①$

$\Rightarrow a = a \% ①$

(3)  $a \% ①$

为什么编译不报错，但运行无输出、返回代码为负值、且运行时间更长？

程序运行至除0取余步骤时，无法进行除0取余运算，C++的标准规定整型运算中除以0是“未定义行为（Undefined Behavior）”，程序异常退出，无法运行至return 0语句，返回代码-1073741676为负值代表整数除0的异常情况。所以编译不报错，但运行无输出、返回代码为负值、且运行时间更长。

$a = 3$  差0存放在中间变量中

$a = 0$

$a = 0$  (程序运行至此，无法进行除0取余运算，程序无法运行至return 0语句)

```
My Project.cpp
1 #include <iostream>
2 using namespace std;
3
4 int main()
5 {
6     int a = 3;
7     a += a \% a -= a;
8     cout << a << endl;
9     return 0;
10}
11
12 D:\Learning\高级语言程序设计\My Project\My Project\My Project.exe
13
14
15 Process exited after 0.7677 seconds with return value 3221225620
16 请按任意键继续... 
```

```
My Project.cpp
1 #include <iostream>
2 using namespace std;
3
4 int main()
5 {
6     int a = 3;
7     a += a \% a -= a;
8     cout << a << endl;
9     return 0;
10}
11
12 Microsoft Visual Studio 调试控制台
13 D:\Learning\高级语言程序设计\My Project\Debug\My Project.exe <进程 3728>已退出，代码为 -1073741676。
14 按任意键关闭此窗口... 
```



## §. 基础知识题 - 变量及表达式求值

### 4. 求复合赋值表达式的值

假设 `int a = 3, n = 7;`

`D. n %= a %= 3`

`a %= 3` (记作①)

$\Rightarrow a = a \% 3$

(1)  $a \% 3$

(2)  $a = \text{模}$

$n \% = ①$

$\Rightarrow n = n \% ①$

(3)  $n \% ①$

为什么编译不报错，但运行无输出、返回代码为负值、且运行时间更长？

程序运行至除0取余步骤时，无法进行除0取余运算，C++的标准规定整型运算中除以0是“未定义行为（Undefined Behavior）”，程序异常退出，无法运行至`return 0`语句，返回代码-1073741676为负值代表整数除0的异常情况。所以编译不报错，但运行无输出、返回代码为负值、且运行时间更长。

$a = 3 \quad n = 7 \quad \text{模0存放在中间变量中}$

$a = 0 \quad n = 7$

$a = 0 \quad n = 7$

(程序运行至此，无法进行除0取余运算，程序无法运行至`return 0`语句)

```
My Project.cpp
1 #include <iostream>
2 using namespace std;
3
4 int main()
5 {
6     int a = 3, n = 7;
7     n %= a %= 3;
8     cout << a << " " << n << endl;
9     return 0;
10}
11
12 D:\Learning\高级语言程序设计\My Project\My Project.exe
13
14 Process exited after 0.7434 seconds with return value 3221225620
15 请按任意键继续...
16
```

```
My Project.cpp
1 #include <iostream>
2 using namespace std;
3
4 int main()
5 {
6     int a = 3, n = 7;
7     n %= a %= 3;
8     cout << a << " " << n << endl;
9     return 0;
10}
11
12 Microsoft Visual Studio 调试控制台
13 D:\Learning\高级语言程序设计\My Project\Debug\My Project.exe (进程 21784)已退出, 代码为 -1073741676。
14 按任意键关闭此窗口...
```



## §. 基础知识题 – IEEE754的理解

float型数据的内部存储格式

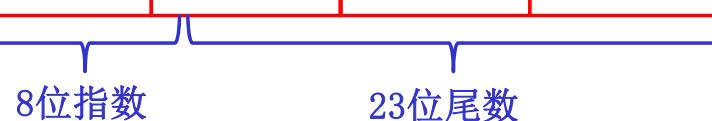
```
#include <iostream>
using namespace std;
int main()
{
    float f = 123.456f;
    unsigned char* p = (unsigned char*)&f;
    cout << hex << (int)(*p)      << endl;
    cout << hex << (int)(*(p+1)) << endl;
    cout << hex << (int)(*(p+2)) << endl;
    cout << hex << (int)(*(p+3)) << endl;
    return 0;
}
```

Microsoft  
79  
e9  
f6  
42

d: 低位在前存放	
2000	0x79
2001	0xe9
2002	0xf6
2003	0x42

上例解读：单精度浮点数123.456，在内存中占四个字节，四个字节的值依次为0x42 0xf6 0xe9 0x79（按打印顺序逆向取）

转换为32bit则为：0100 0010 1111 0110 1110 1001 0111 1001





## §. 基础知识题 – IEEE754的理解

double型数据的内部存储格式

```
#include <iostream>
using namespace std;
int main()
{
    double d = 1.23e4;
    unsigned char* p = (unsigned char*)&d;
    cout << hex << (int)(*p)      << endl;
    cout << hex << (int)(*(p+1)) << endl;
    cout << hex << (int)(*(p+2)) << endl;
    cout << hex << (int)(*(p+3)) << endl;
    cout << hex << (int)(*(p+4)) << endl;
    cout << hex << (int)(*(p+5)) << endl;
    cout << hex << (int)(*(p+6)) << endl;
    cout << hex << (int)(*(p+7)) << endl;
    return 0;
}
```

d: 低位在前存放	
2000	0x00
2001	0x00
2002	0x00
2003	0x00
2004	0x00
2005	0x06
2006	0xc8
2007	0x40

上例解读：双精度浮点数1.23e4，在内存中占八个字节，八个字节的值依次为0x40 0xc8 0x06 0x00 0x00 0x00 0x00 0x00 (逆向)

转换为64bit则为： 0100 0000 | 1100 1000 | 0000 0100 | 0000 0000 | 0000 0000 | 0000 0000 | 0000 0000 | 0000 0000





## §. 基础知识题 – IEEE754的理解

### 1. float型数的机内表示

格式要求：多字节时，每4bit中间加一个空格或-（例：“1101 0100 0011 0001” 或 “1101-0100-0011-0001”）

A. 2250758.8570522

注：尾数为正、指数为正

(1) 得到的32bit的机内表示是：0100 1010 0000 1001 0110 0000 0001 1011

(2) 其中：符号位是0

指数是1001 0100（填32bit中的原始形式）

指数转换为十进制形式是148（32bit中的原始形式按二进制原码形式转换）

指数表示的十进制形式是21（32bit中的原始形式按IEEE754的规则转换）

尾数是000 1001 0110 0000 0001 1011（填32bit中的原始形式）

尾数转换为十进制小数形式是0.07324540615081787109375（32bit中的原始形式按二进制原码形式转换）

尾数表示的十进制小数形式是1.07324540615081787109375（加整数部分的1）



## §. 基础知识题 – IEEE754的理解

### 1. float型数的机内表示

格式要求：多字节时，每4bit中间加一个空格或-（例：“1101 0100 0011 0001” 或 “1101-0100-0011-0001”）

B. -8570522.2250758

注：尾数为负、指数为正

(1) 得到的32bit的机内表示是：1100 1011 0000 0010 1100 0110 1001 1010

(2) 其中：符号位是1

指数是1001 0110（填32bit中的原始形式）

指数转换为十进制形式是150（32bit中的原始形式按二进制原码形式转换）

指数表示的十进制形式是23（32bit中的原始形式按IEEE754的规则转换）

尾数是000 0010 1100 0110 1001 1010（填32bit中的原始形式）

尾数转换为十进制小数形式是0.0216858386993408203125（32bit中的原始形式按二进制原码形式转换）

尾数表示的十进制小数形式是1.0216858386993408203125（加整数部分的1）



## §. 基础知识题 – IEEE754的理解

### 1. float型数的机内表示

格式要求：多字节时，每4bit中间加一个空格或-（例：“1101 0100 0011 0001” 或 “1101-0100-0011-0001”）

C. 0.002250758

注：尾数为正、指数为负

(1) 得到的32bit的机内表示是：0011 1011 0001 0011 1000 0001 0111 0100

(2) 其中：符号位是0

指数是0111 0110（填32bit中的原始形式）

指数转换为十进制形式是118（32bit中的原始形式按二进制原码形式转换）

指数表示的十进制形式是-9（32bit中的原始形式按IEEE754的规则转换）

尾数是001 0011 1000 0001 0111 0100（填32bit中的原始形式）

尾数转换为十进制小数形式是0.152388095855712890625（32bit中的原始形式按二进制原码形式转换）

尾数表示的十进制小数形式是1.152388095855712890625（加整数部分的1）



## §. 基础知识题 – IEEE754的理解

### 1. float型数的机内表示

格式要求：多字节时，每4bit中间加一个空格或-（例：“1101 0100 0011 0001” 或 “1101-0100-0011-0001”）

D. -0.008570522

注：尾数为负、指数为负

(1) 得到的32bit的机内表示是：1011 1100 0000 1100 0110 1011 0110 0000

(2) 其中：符号位是1

指数是0111 1000（填32bit中的原始形式）

指数转换为十进制形式是120（32bit中的原始形式按二进制原码形式转换）

指数表示的十进制形式是-7（32bit中的原始形式按IEEE754的规则转换）

尾数是000 1100 0110 1011 0110 0000（填32bit中的原始形式）

尾数转换为十进制小数形式是0.097026824951171875（32bit中的原始形式按二进制原码形式转换）

尾数表示的十进制小数形式是1.097026824951171875（加整数部分的1）



## §. 基础知识题 – IEEE754的理解

### 2. double型数的机内表示

格式要求：多字节时，每4bit中间加一个空格或-（例：“1101 0100 0011 0001” 或 “1101-0100-0011-0001”）

A. 2250758.8570522

注：尾数为正、指数为正

(1) 得到的64bit的机内表示是：0100 0001 0100 0001 0010 1100 0000 0011 0110 1101 1011 0011 1110 0010 1111 0001

(2) 其中：符号位是0

指数是1000 0010 100（填64bit中的原始形式）

指数转换为十进制形式是1044（64bit中的原始形式按二进制原码形式转换）

指数表示的十进制形式是21（64bit中的原始形式按IEEE754的规则转换）

尾数是0001 0010 1100 0000 0011 0110 1101 1011 0011 1110 0010 1111 0001（填64bit中的原始形式）

尾数转换为十进制小数形式是0.0732454571972847（64bit中的原始形式按二进制原码形式转换）

尾数表示的十进制小数形式是1.0732454571972847（加整数部分的1）



## §. 基础知识题 – IEEE754的理解

### 2. double型数的机内表示

格式要求：多字节时，每4bit中间加一个空格或-（例：“1101 0100 0011 0001” 或 “1101-0100-0011-0001”）

B. -8570522.2250758

注：尾数为负、指数为正

(1) 得到的64bit的机内表示是：1100 0001 0110 0000 0101 1000 1101 0011 0100 0111 0011 0011 1101 0010 0010 1010

(2) 其中：符号位是1

指数是1000 0010 110（填64bit中的原始形式）

指数转换为十进制形式是1046（64bit中的原始形式按二进制原码形式转换）

指数表示的十进制形式是23（64bit中的原始形式按IEEE754的规则转换）

尾数是0000 0101 1000 1101 0011 0100 0111 0011 0011 1101 0010 0010 1010（填64bit中的原始形式）

尾数转换为十进制小数形式是0.02168586553046703（64bit中的原始形式按二进制原码形式转换）

尾数表示的十进制小数形式是1.02168586553046703（加整数部分的1）



## §. 基础知识题 – IEEE754的理解

### 2. double型数的机内表示

格式要求：多字节时，每4bit中间加一个空格或-（例：“1101 0100 0011 0001” 或 “1101-0100-0011-0001”）

C. 0.002250758

注：尾数为正、指数为负

(1) 得到的64bit的机内表示是：0011 1111 0110 0010 0111 0000 0010 1110 1000 0000 0000 1001 1110 1010 0101 0011

(2) 其中：符号位是0

指数是0111 1110 110（填64bit中的原始形式）

指数转换为十进制形式是1014（64bit中的原始形式按二进制原码形式转换）

指数表示的十进制形式是-9（64bit中的原始形式按IEEE754的规则转换）

尾数是0010 0111 0000 0010 1110 1000 0000 0000 1001 1110 1010 0101 0011（填64bit中的原始形式）

尾数转换为十进制小数形式是0.15238809599999992（64bit中的原始形式按二进制原码形式转换）

尾数表示的十进制小数形式是1.1523880959999992（加整数部分的1）



## §. 基础知识题 – IEEE754的理解

### 2. double型数的机内表示

格式要求：多字节时，每4bit中间加一个空格或-（例：“1101 0100 0011 0001” 或 “1101-0100-0011-0001”）

D. -0.008570522

注：尾数为负、指数为负

(1) 得到的64bit的机内表示是：1011 1111 1000 0001 1000 1101 0110 1011 1111 1101 1001 1000 1110 0001 0101 0010

(2) 其中：符号位是1

指数是0111 1111 000（填64bit中的原始形式）

指数转换为十进制形式是1016（64bit中的原始形式按二进制原码形式转换）

指数表示的十进制形式是-7（64bit中的原始形式按IEEE754的规则转换）

尾数是0001 1000 1101 0110 1011 1111 1101 1001 1000 1110 0001 0101 0010（填64bit中的原始形式）

尾数转换为十进制小数形式是0.09702681600000007（64bit中的原始形式按二进制原码形式转换）

尾数表示的十进制小数形式是1.0970268160000007（加整数部分的1）

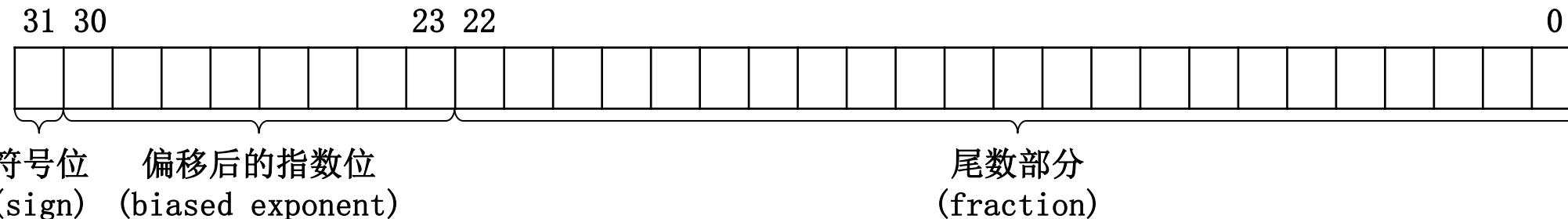
## §. 基础知识题 – IEEE754的理解



### 3. 总结

(1) ①float型数据的32bit是如何分段来表示一个单精度的浮点数的？给出bit位的分段解释。

float型数据在内存中的存储分为三部分，分别是符号位、指数部分和尾数部分。



②尾数的正负如何表示?

尾数的正负由符号位表示，符号位为0表示该double型数据为正数，为1表示为负数。

### ③尾数如何表示?

把十进制浮点数表示为二进制形式，把这个二进制数转换为以2为底的指数形式，把小数点放在左起第一位和第二位之间，且第一位需要是个非0数。尾数部分的最高位始终为1，所以隐藏高位的整数部分1，低位补0，补齐23位，如果小数部分无限循环，则四舍五入为23位，这23位二进制数即为尾数在32bit中的原始形式。

#### ④指数的正负如何表示？指数如何表示？

8位二进制数是指数在32bit中的原始形式，按二进制原码形式转换，可以得到十进制形式的256个非负整数(0~255)，每个非负整数按IEEE754的规则转换，减去偏移量(127)，就得到了指数，范围是-127~128，于是指数的正负便表示出来了。同时，以2为底的指数通过这种规则也表示出来了。



## §. 基础知识题 – IEEE754的理解

### 3. 总结

(2) ①为什么float型数据只有7位十进制有效数字？为什么最大只能是 $3.4 \times 10^{38}$ ？

float型数据在内存中的存储分为三部分，分别是1位符号位、8位指数位和23位尾数位（实际上尾数位为24位，因为在尾数位前还隐藏了一个整数部分1或0）。符号位用于控制正负号，指数位用于控制小数点的移动，尾数位用于控制精度（或者说记录状态）。在24位尾数中能精确记录 $2^{24}=16777216$ 种不同的状态，即float型数据最多只能存储16777216种十进制状态。16777216是8位数，所以float型数据的精度最多是7位十进制有效数字（0~9999999），共10000000种状态。如果float型数据的精度是8位十进制（0~99999999），则共100000000种状态，大于了float型数据能存储的状态上限16777216。**所以说float型数据的精度不能达到8位十进制有效数字，只有7位十进制有效数字。**

根据IEEE754，按照尾数位隐藏的整数部分是1还是0，可以将浮点数划分为normal number和subnormal number，当指数位全部被1填充时，这个浮点数为non-number，即这个数可能是±infinity或NaN。

当float型数据的指数位为00000000时，对应的数为subnormal number；当float型数据的指数位在00000001到11111110之间（转换为十进制为1~254，减去偏移量127为-126~127）时，对应的数为normal number；当float型数据的指数位为11111111时，对应的数为non-number。讨论浮点数的取值范围实际上讨论的是normal number的取值范围。float型数据的指数部分无法取到-127和128，因为用于表示-127的00000000被用来表示subnormal number了，用于表示128的11111111被用来表示non-number了。所以float型数据的指数部分的取值范围为[-126, 127]。

尾数（含隐藏的整数部分）所表示的值的范围其实是 $[1.00\dots00, 1.11\dots11]$ ，即十进制形式的 $[1, 2)$ 。

float型数据的取值范围=±尾数 $\times 2^{\text{指数}}$ =± $[1, 2) \times 2^{[-126, 127]}$ ，把以2为底替换为以10为底，得到float型数据的取值范围为 $(-3.402823669\dots \times 10^{38}, -1.17549435\dots \times 10^{-38}] \cup [1.17549435\dots \times 10^{-38}, 3.402823669\dots \times 10^{38})$ ，从上面这个集合中取一个更容易表示的子集 $[-3.4 \times 10^{38}, -1.18 \times 10^{-38}] \cup [1.18 \times 10^{-38}, 3.4 \times 10^{38}]$ 就是float型数据的取值范围。之所以能写成闭区间的形式是因为它是真正取值范围的一个子集。**所以说float型数据最大只能是 $3.4 \times 10^{38}$ 。**



## §. 基础知识题 – IEEE754的理解

### 3. 总结

(2) ②有些资料上说有效位数是6~7位，能找出6位/7位不同的例子吗？

float型数据的尾数部分是23bit,  $2^{23}=8388608$ , 故float型数据能表示的最大数为8388608, 而大于8388608的7位整数, 如9999999, float型数据便不能表示, 这就决定了float型数据的有效位数不能是7位, 但是绝对能保证精度的有效位数是6位, 所以说float型数据的有效位数是6~7位。不超过8388608的float型数据是精度保证的, 大于8388608而小于9999999的float型数据则是精度不保证的。

6位/7位不同的例子如下：

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

```
1 #include <iostream>
2 #include <iomanip>
3 using namespace std;
4
5 int main()
6 {
7     float f = 9.111111;
8     cout << setprecision(8) << f << endl;
9     return 0;
10 }
```

The debug console window at the bottom right is titled "Microsoft Visual Studio 调试控制台" and displays the output: "9.1111107".

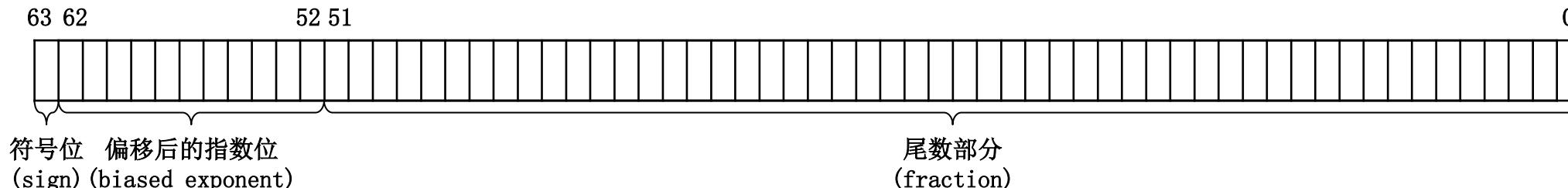


## §. 基础知识题 – IEEE754的理解

### 3. 总结

(3) ① double型数据的64bit是如何分段来表示一个双精度的浮点数的？给出bit位的分段解释。

`double`型数据在内存中的存储分为三部分，分别是符号位、指数部分和尾数部分。



②尾数的正负如何表示?

尾数的正负由符号位表示，符号位为0表示该double型数据为正数，为1表示为负数。

### ③尾数如何表示?

把十进制浮点数表示为二进制形式，把这个二进制数转换为以2为底的指数形式，把小数点放在左起第一位和第二位之间，且第一位需要是个非0数。尾数部分的最高位始终为1，所以隐藏高位的整数部分1，低位补0，补齐52位，如果小数部分无限循环，则四舍五入为52位，这52位二进制数即为尾数在64bit中的原始形式。

④指数的正负如何表示？指数如何表示？

11位二进制数是指数在64bit中的原始形式，按二进制原码形式转换，可以得到十进制形式的2048个非负整数(0~2047)，每个非负整数按IEEE754的规则转换，减去偏移量(1023)，就得到了指数，范围是-1023~1024，于是指数的正负便表示出来了。同时，以2为底的指数通过这种规则也表示出来了。



## §. 基础知识题 – IEEE754的理解

### 3. 总结

(4) ①为什么double型数据只有15位十进制有效数字？为什么最大只能是 $1.7 \times 10^{308}$ ？

double型数据在内存中的存储分为三部分，分别是1位符号位、11位指数位和52位尾数位（实际上尾数位为53位，因为在尾数位前还隐藏了一个整数部分1或0）。符号位用于控制正负号，指数位用于控制小数点的移动，尾数位用于控制精度（或者说记录状态）。在53位尾数中能精确记录 $2^{53}=9007199254740992$ 种不同的状态，即double型数据最多只能存储9007199254740992种十进制状态。9007199254740992是16位数，所以double型数据的精度最多是15位十进制有效数字（ $0^{\sim}999999999999999$ ），共1000000000000000种状态。如果double型数据的精度是16位十进制（ $0^{\sim}999999999999999$ ），则共1000000000000000种状态，大于了double型数据能存储的状态上限9007199254740992。**所以说double型数据的精度不能达到16位十进制有效数字，只有15位十进制有效数字。**

根据IEEE754，按照尾数位隐藏的整数部分是1还是0，可以将浮点数划分为normal number和subnormal number，当指数位全部被1填充时，这个浮点数为non-number，即这个数可能是 $\pm\infty$ 或NaN。

当double型数据的指数位为00000000000时，对应的数为subnormal number；当double型数据的指数位在00000000001到11111111110之间（转换为十进制为 $1^{\sim}2046$ ，减去偏移量1023为 $-1022^{\sim}1023$ ）时，对应的数为normal number；当double型数据的指数位为11111111111时，对应的数为non-number。讨论浮点数的取值范围实际上讨论的是normal number的取值范围。double型数据的指数部分无法取到-1023和1024，因为用于表示-1023的0000000000被用来表示subnormal number了，用于表示1024的1111111111被用来表示non-number了。所以double型数据的指数部分的取值范围为 $[-1022, 1023]$ 。

尾数（含隐藏的整数部分）所表示的值的范围其实是 $[1.00\dots00, 1.11\dots11]$ ，即十进制形式的 $[1, 2)$ 。

double型数据的取值范围=±尾数 $\times 2^{\text{指数}}=\pm [1, 2) \times 2^{[-1022, 1023]}$ ，把以2为底替换为以10为底，得到double型数据的取值范围为 $(-1.797693134\dots \times 10^{308}, -4.940656458\dots \times 10^{-324}] \cup [4.940656458\dots \times 10^{-324}, 1.797693134\dots \times 10^{308})$ ，从上面这个集合中取一个更容易表示的子集 $[-1.7 \times 10^{308}, -5.0 \times 10^{-324}] \cup [5.0 \times 10^{-324}, 1.7 \times 10^{308}]$ 就是double型数据的取值范围。之所以能写成闭区间的形式是因为它是真正取值范围的一个子集。**所以double型数据最大只能是 $1.7 \times 10^{308}$ 。**



## §. 基础知识题 – IEEE754的理解

### 3. 总结

(4) ②有些资料上说有效位数是15~16位，能找出15位/16位不同的例子吗？

double型数据的尾数部分是52bit,  $2^{52}=4503599627370496$ , 故double型数据能表示的最大数为4503599627370496, 而大于4503599627370496的16位整数, 如9999999999999999, double型数据便不能表示, 这就决定了double型数据的有效位数不能是16位, 但是绝对能保证精度的有效位数是15位, 所以说double型数据的有效位数是15~16位。不超过4503599627370496的double型数据是精度保证的, 大于4503599627370496而小于9999999999999999的double型数据则是精度不保证的。

15位/16位不同的例子如下：

```
My Project.cpp  x
My Project

1  #include <iostream>
2  #include <iomanip>
3  using namespace std;
4
5  int main()
6  {
7      double f = 9.11111111111111;
8      cout << setprecision(17) << f << endl;
9      return 0;
10 }
```

Microsoft Visual Studio 调试控制台  
9.111111111111107



## §. 基础知识题 – IEEE754的理解

### 4. 思考

(1) 8/11bit的指数的表示形式是2进制补码吗？如果不是，一般称为什么方式表示？

不是。二进制原码。

(2) double赋值给float时，下面两个程序，double型常量不加F的情况下，左侧有warning，右侧无warning，为什么？总结一下规律。

```
#include <iostream>
using namespace std;
int main()
{
    float f = 1.2;
    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;
}
```

warning C4305: “初始化”：从“double”到“float”截断

```
#include <iostream>
using namespace std;
int main()
{
    float f = 100.25;
    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;
}
```

左侧有warning的原因：double型数据1.2在赋值给float进行机内存储时，小数部分0.2在转换为二进制形式时，小数部分无限循环，需要四舍五入为23位，此处体现出误差，所以编译器显示warning C4305：“初始化”：从“double”到“float”截断。

右侧无warning的原因：double型数据100.25在赋值给float进行机内存储时，小数部分0.25在转换为二进制形式时，小数部分需要低位补0，补齐23位，此处未体现出误差，所以无warning。

规律总结：double型数据在赋值给float时，若数据被截断，体现出误差，则有warning；若数据未被截断，未体现出误差，则无warning。



## §. 基础知识题 – 关系运算、逻辑运算与选择结构

### 1. 关系运算符的求值顺序

A. 观察下列程序的运行结果，回答问题并将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=1, b=2, c=3, d;

    d = a > b > c;
    cout << d << endl;

    d = a > b < c;
    cout << d << endl;

    d = a > b < c;
    cout << d << endl;

    return 0;
}
```

#### 1、贴运行结果

```
Microsoft Visual Studio 调试控制台
0
warning C4804: ">" : 在操作中使用类型“bool”不安全
1
warning C4804: "<" : 在操作中使用类型“bool”不安全
warning C4804: "<" : 在操作中使用类型“bool”不安全
1
```

#### 2、VS下为什么会有三个warning？说说你的理解

根据运算符优先级，>和<为优先级第8组，左结合，=为优先级第16组，右结合， $a > b$ 会返回bool类型，其值为0/1，再与c进行比较，因此连续是比较是不安全的。所以会显示warning C4804: “>”：在操作中使用类型“bool”不安全。



## §. 基础知识题 – 关系运算、逻辑运算与选择结构

### 1. 关系运算符的求值顺序

B. 观察下列程序的运行结果，回答问题并将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=3, b=2, c=1, d;

    d = a > b > c;
    cout << d << endl;

    d = a < b < c;
    cout << d << endl;

    d = b > a < c;
    cout << d << endl;

    return 0;
}
```

#### 1、贴运行结果

```
Microsoft Visual Studio 调试控制台
0
warning C4804: ">" : 在操作中使用类型“bool”不安全
1
warning C4804: "<" : 在操作中使用类型“bool”不安全
warning C4804: "<" : 在操作中使用类型“bool”不安全
1
```

2、(1)  $a > b > c$ 这个式子，按常规理解， $3 > 2 > 1$ 是正确的，为什么结果是0？

$a > b$ 为真，返回bool类型，其值为1， $1 > c$ 为假，返回bool类型，其值为0，0赋值给整型变量d，所以结果是0。

(2)  $a < b < c$ 这个式子，按常规理解， $3 < 2 < 1$ 是错误的，为什么结果是1？

$a < b$ 为假，返回bool类型，其值为0， $0 < c$ 为真，返回bool类型，其值为1，1赋值给整型变量d，所以结果是1。

(3)  $b > a < c$ 这个式子，按常规理解， $2 > 3 < 1$ 是错误的，为什么结果是1？

$b > a$ 为假，返回bool类型，其值为0， $0 < c$ 为真，返回bool类型，其值为1，1赋值给整型变量d，所以结果是1。



## §. 基础知识题 – 关系运算、逻辑运算与选择结构

### 2. 关系运算符与实数

A. 观察下列程序的运行结果，回答问题并将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>
#include <cmath>
using namespace std;

int main()
{
    float f1 = 100.25;
    cout << (f1 - 100.25) << endl;
    cout << (f1 == 100.25) << endl;
    cout << (fabs(f1-100.25) < 1e-6) << endl;

    float f2 = 1.2;
    cout << (f2 - 1.2) << endl;
    cout << (f2 == 1.2) << endl;
    cout << (fabs(f2-1.2) < 1e-6) << endl;

    return 0;
}
```

1、贴VS+Dev下的运行结果

```
0
1
1
4.76837e-08
0
```

warning C4305: “初始化”：从“double”到“float”截断

2、删除第2行的#include<cmath>后，再次贴VS+Dev的运行结果

```
0
1
1
4.76837e-08
0
```

[Error] 'fabs' was not declared in this scope; did you mean 'labs'?

3、由本例得出的结论，实数进行相等比较时的通用方法是当两数相减的绝对值小于某个值则认为相等。



## §. 基础知识题 – 关系运算、逻辑运算与选择结构

### 2. 关系运算符与实数

B. 观察下列程序的运行结果，回答问题并将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>
#include <cmath> //VS可不加
using namespace std;

int main()
{
    double d1=123.456789012345678;
    double d2=123.456789123456789;
    cout << (d1==d2) << endl;
    cout << (fabs(d1-d2)<1e-6) << endl;
    cout << (fabs(d1-d2)<1e-7) << endl;

    float f1=123.456789012345678;
    float f2=123.456789123456789;
    cout << (f1==f2) << endl;
    cout << (fabs(f1-f2)<1e-6) << endl;
    cout << (fabs(f1-f2)<1e-7) << endl;

    return 0;
} //VS有两个warning
```

#### 1、贴运行结果

```
Microsoft Visual Studio 调试控制台
0
1
0
1
1
1
```

warning C4305: “初始化”：从“double”到“float”截断  
warning C4305: “初始化”：从“double”到“float”截断

2、观察  $\text{fabs}(**)<1e-6$  和  $\text{fabs}(**)<1e-7$  在 float 和 double 下的表现，哪个相同？哪个不同？为什么？

$\text{fabs}(**)<1e-6$  和  $\text{fabs}(**)<1e-7$  在 float 下的表现相同；  
 $\text{fabs}(**)<1e-6$  和  $\text{fabs}(**)<1e-7$  在 double 下的表现不同；

从“double”到“float”截断，超出 float 型有效位数的数据根据 IEEE754 在机内存储，小数点后第 7 位在机内的存储是相同的，所以  $\text{fabs}(**)<1e-6$  和  $\text{fabs}(**)<1e-7$  在 float 下的表现相同；double 型的有效位数比 float 型多，小数点后第 7 位在机内的存储是不同的，所以  $\text{fabs}(**)<1e-6$  和  $\text{fabs}(**)<1e-7$  在 double 下的表现不同。



## §. 基础知识题 – 关系运算、逻辑运算与选择结构

### 3. 逻辑常量与逻辑变量

A. 观察下列程序的运行结果，回答问题并将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()
{
    cout << true    << endl;
    cout << "true"  << endl;

    cout << endl;

    cout << false   << endl;
    cout << "false" << endl;

    return 0;
}
```

1、贴运行结果

Microsoft Visual Studio 调试控制台

```
1
true
0
false
```

2、解释 true 和 "true" 的区别 (false和"false")

true是逻辑常量，数据类型为逻辑变量bool类型；"true"是字符串常量；false是逻辑常量，数据类型为逻辑变量bool类型；"false"是字符串常量。

3、进阶思考：目前直接输出逻辑常量true和false，在屏幕上的输出是1/0，如果想输出为true/false，应该怎么做？

使用前导格式控制符boolalpha。



## §. 基础知识题 – 关系运算、逻辑运算与选择结构

### 3. 逻辑常量与逻辑变量

B. 观察下列程序的运行结果，回答问题并将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()
{
    bool k1 = true;

    cout << sizeof(true) << endl;
    cout << sizeof(k1) << endl;
    cout << k1 << ',' << int(k1) << endl;

    cout << endl;

    bool k2 = false;
    cout << sizeof(false) << endl;
    cout << sizeof(k1) << endl;
    cout << k2 << ',' << int(k2) << endl;

    return 0;
}
```

1、贴运行结果

```
Microsoft Visual Studio 调试控制台
1
1
1 1
1
1
0 0
```

2、bool型常量/变量在内存中占用1字节，  
值是0(false)/1(true)

总结bool型常量/变量在输出时的规则  
(限制：在无3. A的前导格式控制符的前提下)

bool型常量/变量在输出时按整型量进行处理，屏幕上的输出是1/0，而不是true/false。



## §. 基础知识题 – 关系运算、逻辑运算与选择结构

### 3. 逻辑常量与逻辑变量

C. 观察下列程序的运行结果，回答问题并将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()
{
    bool k;

    cin >> k;
    cout << k << ',' << int(k) << endl;

    return 0;
}
```

1、输入0，输出是：

Microsoft Visual Studio 调试控制台  
0  
0 0

2、输入1，输出是：

Microsoft Visual Studio 调试控制台  
1  
1 1

3、输入123，输出是：

Microsoft Visual Studio 调试控制台  
123  
1 1

4、输入true，输出是：

Microsoft Visual Studio 调试控制台  
true  
0 0

5、输入false，输出是：

Microsoft Visual Studio 调试控制台  
false  
0 0

总结bool型变量在输入时的规则：  
cin时只能输入0/1，否则得到不可信值。



## §. 基础知识题 – 关系运算、逻辑运算与选择结构

### 3. 逻辑常量与逻辑变量

D. 观察下列程序的运行结果，回答问题并将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()
{
    bool k;

    k='A';
    cout << k << ',' << (int)k << endl;

    k=0;
    cout << k << ',' << (int)k << endl;

    k=256;
    cout << k << ',' << (int)k << endl;

    return 0;
}
```

1、贴运行结果

Microsoft Visual Studio 调试控制台

```
1 1
warning C4305: “=”：从“char”到“bool”截断 0 0
warning C4305: “=”：从“int”到“bool”截断 1 1
```

2、解释VS下waring的意思

从多字节变量赋值给取值只有0/1的bool类型少字节变量时会发出此警告，数据被截断，这可能将导致信息丢失。

3、`k='A'`是1字节赋值给1字节，为什么还有warning?

A的ASCII码为65，`k='A'`代表将65赋值给值只有0/1的bool类型，数据被截断，所以会显示warning。

4、`k=256`如果按整型的4字节赋给1字节，`k`应该是多少？

现在实际是多少？为什么？

`k=256`如果按整型的4字节赋给1字节，`k`应该是0，现在实际是1，因为bool赋值时不是多字节赋少字节的规则，而是“非0为真0为假”。

5、“非0为真0为假”这句话如何解释？

非0值在赋值给bool类型时值为真(true)，取值为1，0值在赋值给bool类型时值为假(false)，取值为0



## §. 基础知识题 – 关系运算、逻辑运算与选择结构

### 3. 逻辑常量与逻辑变量

E. 观察下列程序的运行结果，回答问题并将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()
{
    bool f=true;
    int a=10;

    a=a+f;
    cout << a << endl;

    return 0;
}
```

1、贴运行结果

Microsoft Visual Studio 调试控制台  
11

2、当bool参与表达式计算时，当做整型值，即0/1。



## §. 基础知识题 – 关系运算、逻辑运算与选择结构

### 4. 逻辑运算符与逻辑运算

A. 完成下列两个表格的填写

a	b	$!a$	$!b$	$a \&\& b$	$a \mid\mid b$
1	1	0	0	1	1
1	0	0	1	0	1
0	1	1	0	0	1
0	0	1	1	0	0

a	b	$!a$	$!b$	$a \&\& b$	$a \mid\mid b$
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



## §. 基础知识题 – 关系运算、逻辑运算与选择结构

### 4. 逻辑运算符与逻辑运算

B. 观察下列程序的运行结果，回答问题并将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=1, b=2, c=3, d=4, m=1, n=1;

    cout << "m=" << m << " n=" << n << endl;
    (m=a>b)&&(n=c>d);
    cout << "m=" << m << " n=" << n << endl;

    return 0;
}
```

#### 1、贴运行结果

```
m=1 n=1
m=0 n=1
```

2、解释 $(m=a>b) \&\& (n=c>d)$ 的求值过程（标出步骤顺序）

- ①  $a>b$
- ②  $m=a>b$
- ③  $(m=a>b) \&\& (n=c>d)$

3、短路运算的意思是：仅当必须执行下一个逻辑运算符才能求出解时，才执行该运算符，否则不执行



## §. 基础知识题 – 关系运算、逻辑运算与选择结构

### 4. 逻辑运算符与逻辑运算

C. 观察下列程序的运行结果，回答问题并将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;

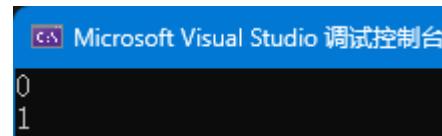
int main()
{
    int m = 1;
    cout << (8 < 4 - !0) << endl;
    5 > 3 && 2 || (m = 8 < 4 - !0);
    cout << m << endl;

    return 0;
}
```

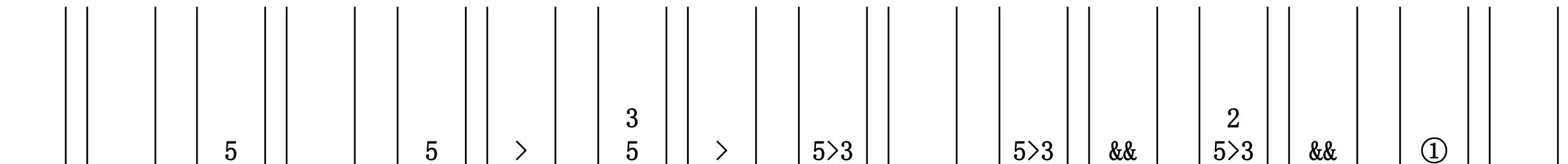
有以下逻辑表达式

$5 > 3 \&\& 2 \mid\mid 8 < 4 - !0$

1、构造一个测试程序，在不改变该表达式目前求值顺序的情况下（允许插入新的运算，但目前这几个运算符的顺序不要变），证明  $8 < 4 - !0$  存在短路运算



2、用栈方式画包含短路运算的表达式，则从分析到短路运算符进栈开始（本例中为  $\mid\mid$ ），忽略比  $\mid\mid$  优先级高的运算符。



初始：两栈均  
为空

5进栈

>进栈

3进栈

要进栈的( $\&\&$ )低于  
栈顶的( $>$ )，先计算

&&进栈

2进栈

要进栈的( $\mid\mid$ )低于栈  
顶的( $\&\&$ )，先计算  
( $5 > 3 \&\& 2$  记作①)

①的值为1，要进栈的运算符为  $\mid\mid$ ，此时该表达式的值必为1，后面的运算符不再求解



## §. 基础知识题 – 关系运算、逻辑运算与选择结构

### 5. if语句 – 基本使用

A. 观察下列程序的运行结果，回答问题并将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;

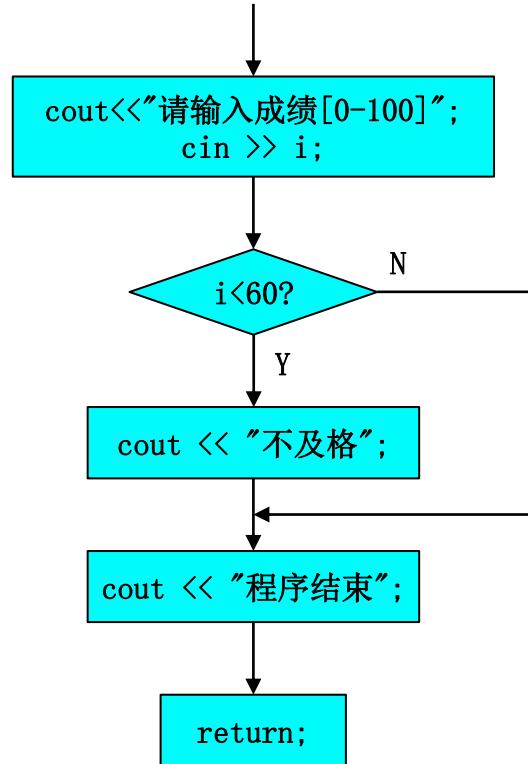
int main()
{
    int i;

    cout<<"请输入成绩[0-100]"<<endl;
    cin >> i;

    if (i<60) {
        cout << "不及格" << endl;
    }
    cout << "程序结束" << endl;

    return 0;
}
```

- 1、输入34，贴运行结果
- 2、输入74，贴运行结果
- 3、画出程序对应的流程框图



The screenshot shows two separate runs of the program in the Microsoft Visual Studio debugger's Immediate window. In the first run, the user inputs '34' and the output is '不及格' (Fail). In the second run, the user inputs '74' and the output is '程序结束' (Program ends).



## §. 基础知识题 – 关系运算、逻辑运算与选择结构

### 5. if语句 - 基本使用

B. 观察下列程序的运行结果，回答问题并将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 i;

    cout<<"请输入成绩[0-100]"<<endl;
    cin >> i;

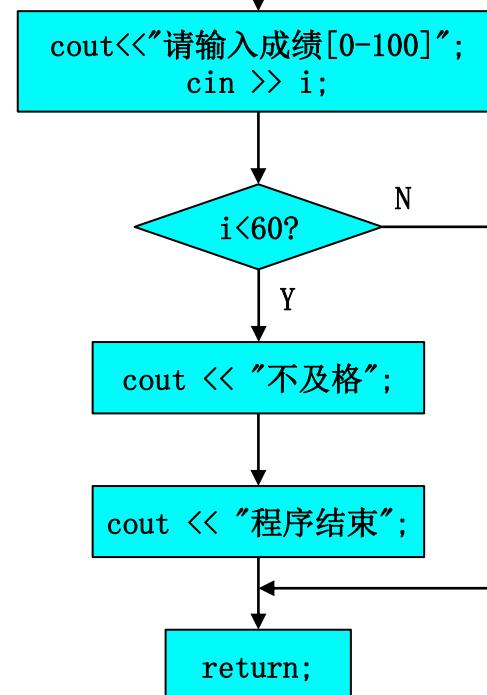
    if (i<60) {
        cout << "不及格" << endl;
        cout << "程序结束" << endl; //未缩进
    }

    return 0;
}
```

1、输入34，贴运行结果

2、输入74，贴运行结果

3、画出程序对应的流程框图



Microsoft Visual Studio 调试控制台

请输入成绩[0-100]

34

不及格

程序结束

Microsoft Visual Studio 调试控制台

请输入成绩[0-100]

74

4、程序标注“未缩进”的行，应该缩进



## §. 基础知识题 – 关系运算、逻辑运算与选择结构

### 5. if语句 – 基本使用

C. 观察下列程序的运行结果，回答问题并将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 i;

    cout<<"请输入成绩[0-100]"<<endl;
    cin >> i;

    if (i<60;)
        cout << "不及格" << endl;
    cout << "程序结束" << endl; //未缩进
}

return 0;
}
```

错误信息截图:

```
warning C4552: “<”: 未使用表达式结果
error C2429: 语言功能 “if/switch 中的 init-statement” 需要编译器标志 “/std:c++17”
error C2059: 语法错误: “;”
error C2143: 语法错误: 缺少“;”(在“{”的前面)
```

错误原因:

if语句条件表达式后无分号，若在条件表达式后加分号，则不符合语法规则。整个if语句可以作为一个语句来看待，若在条件表达式后加分号，编译器会认为该语句已结束，所以会出现warning C4552、error C2429、error C2059和error C2143的提示。

# §. 基础知识题 - 关系运算、逻辑运算与选择结构

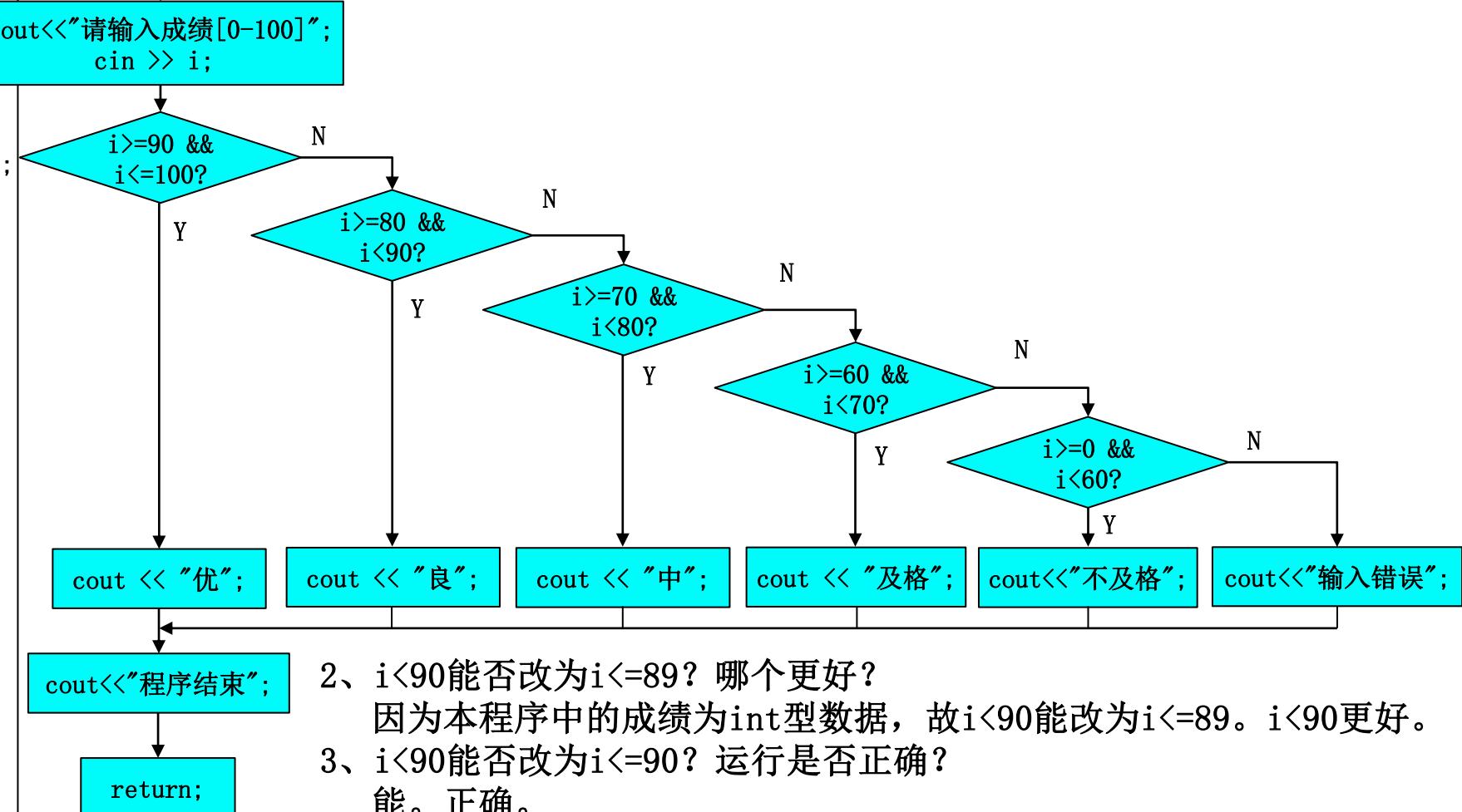
## 5. if语句 - 基本使用

D. 观察下列程序的运行结果，回答问题

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

int main()
{
    int i;
    cout << "请输入成绩[0-100]" << endl;
    cin >> i;
    if (i>=90 && i<=100)
        cout << "优" << endl;
    else if (i>=80 && i<90)
        cout << "良" << endl;
    else if (i>=70 && i<80)
        cout << "中" << endl;
    else if (i>=60 && i<70)
        cout << "及格" << endl;
    else if (i>=0 && i<60)
        cout << "不及格" << endl;
    else
        cout << "输入错误" << endl;
    cout << "程序结束" << endl;
    return 0;
}
```

1、给出程序的流程框图



2、 $i < 90$ 能否改为 $i \leq 89$ ? 哪个更好?

因为本程序中的成绩为int型数据，故 $i < 90$ 能改为 $i \leq 89$ 。 $i < 90$ 更好。

3、 $i < 90$ 能否改为 $i \leq 90$ ? 运行是否正确?  
能。正确。



## §. 基础知识题 – 关系运算、逻辑运算与选择结构

### 6. if语句 – 多重嵌套

A. 一个有10行代码的if语句嵌套，回答问题

```
0: if (表达式) {  
1: if (表达式) {  
2: }  
3: else {  
4: }  
5: }  
6: else {  
7: if (表达式) {  
8: }  
9: }
```

第0行的“{”和第5行的“}”配对

第1行的“{”和第2行的“}”配对

第3行的“{”和第4行的“}”配对

第6行的“{”和第9行的“}”配对

第7行的“{”和第8行的“}”配对

总结：给出大括号配对的基本准则

自上而下，忽略 {，以 } 为准向上匹配未配对的 {。{} 的匹配可用栈理解，遇 { 进栈，遇 } 则栈顶 { 出栈并匹配为一对，若到最后仍有 { 或 } 未配对则语法错。



## §. 基础知识题 – 关系运算、逻辑运算与选择结构

### 6. if语句 - 多重嵌套

B. 一个if语句嵌套如下，回答问题

```
if (表达式1) {  
    if (表达式2) {  
        A;  
    }  
    B;  
}
```

- 1、当表达式1 真，表达式2 真 时，执行语句A
- 2、当表达式1 真，表达式2 任意 时，执行语句B



## §. 基础知识题 – 关系运算、逻辑运算与选择结构

### 6. if语句 - 多重嵌套

C. 一个if语句嵌套如下，回答问题

```
if (表达式1) {  
    if (表达式2) {  
        A;  
    }  
    else {  
        B;  
    }  
    C;  
}  
else {  
    if (表达式3) {  
        D;  
    }  
    E;  
}
```

- 1、当表达式1真，表达式2真时，执行语句A
- 2、当表达式1真，表达式2假时，执行语句B
- 3、当表达式1真，表达式2任意时，执行语句C
- 4、当表达式1假，表达式3真时，执行语句D
- 5、当表达式1假，表达式3任意时，执行语句E



## §. 基础知识题 – 关系运算、逻辑运算与选择结构

### 6. if语句 - 多重嵌套

D. 一个if语句嵌套如下，回答问题

```
if (表达式1) {  
    if (表达式2) {  
        A;  
    }  
    else {  
        B;  
    }  
    C;  
}  
→ F;  
else {  
    if (表达式3) {  
        D;  
    }  
    E;  
}
```

在6.C的基础上，在箭头位置插入语句F

1、请构造一个符合此要求的测试程序，并给出该程序的程序及编译错误截图

2、请说明错误原因

整个双支语句可以作为一个语句来看待，中间不允许插入任何的其他语句。因为从编译器的思维角度，从上到下看，单if、并列的cout，因此else非法，会显示 error C2181: 没有匹配if的非法else。

```
My Project.cpp x  
My Project  
1 #include<iostream>  
2 using namespace std;  
3  
4 int main()  
5 {  
6     int n;  
7     cin >> n;  
8  
9     if (n > 0) {  
10         if (n > 10) {  
11             cout << "A" << endl;  
12         }  
13         else {  
14             cout << "B" << endl;  
15         }  
16         cout << "C" << endl;  
17     }  
18     cout << "F" << endl;  
19     else {  
20         if (n < -10) {  
21             cout << "D" << endl;  
22         }  
23         cout << "E" << endl;  
24     }  
25  
26 }  
27  
return 0;
```



## §. 基础知识题 – 关系运算、逻辑运算与选择结构

### 6. if语句 - 多重嵌套

E. 一个if语句嵌套如下，回答问题

<pre>if (表达式1) { if (表达式2) { A; } B; } else { C; }</pre>	<pre>左侧代码按缩进格式排版 if (表达式1) {     if (表达式2) {         A;     }     B; } else {     C; }</pre>	<ol style="list-style-type: none"><li>1、当表达式1<u>真</u>，表达式2<u>真</u>时，执行语句A</li><li>2、当表达式1<u>真</u>，表达式2<u>任意</u>时，执行语句B</li><li>3、当表达式1<u>假</u>，表达式2<u>任意</u>时，执行语句C</li></ol>
<pre>if (表达式1) { if (表达式2) { A; } else { B; } C; }</pre>	<pre>左侧代码按缩进格式排版 if (表达式1) {     if (表达式2) {         A;     }     else {         B;     }     C; }</pre>	<ol style="list-style-type: none"><li>1、当表达式1<u>真</u>，表达式2<u>真</u>时，执行语句A</li><li>2、当表达式1<u>真</u>，表达式2<u>假</u>时，执行语句B</li><li>3、当表达式1<u>真</u>，表达式2<u>任意</u>时，执行语句C</li></ol>



# §. 基础知识题 – 关系运算、逻辑运算与选择结构

## 7. 条件运算符与条件表达式

A. 观察下列程序的运行结果，回答问题并将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, b;

    cin >> a >> b;

    if (a>b)
        cout << "max=" << a << endl;
    else
        cout << "max=" << b << endl;

    a > b ? cout << "max=" << a << endl : cout << "max=" << b << endl; //1

    cout << "max=" << (a>b?a:b) << endl; //2

    printf("max=%d", a>b?a:b); //3

    return 0;
}
```

1、输入12 34，给出运行截图

```
Microsoft Visual Studio 调试控制台
12 34
max=34
max=34
max=34
max=34
```

2、输入34 12，给出运行截图

```
Microsoft Visual Studio 调试控制台
34 12
max=34
max=34
max=34
max=34
```

3、//1 //2 //3这三种条件运算符的使用，  
按你的喜欢程度排序为//2、//1、//3



## §. 基础知识题 – 关系运算、逻辑运算与选择结构

### 7. 条件运算符与条件表达式

B. 观察下列程序的运行结果，回答问题并将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=1, b=2;

    a==1 ? "Hello" : 123;           //编译报错

    a>b ? cout << a : printf("%d", b); //编译报错

    a==1 ? 'A' : 123;             //编译正确

    return 0;
}
```

#### 1、给出编译报错的截图

```
error C2446: “:”：没有从“int”到“const char [6]”的转换
message : 从整型类型转换为指针类型需要 reinterpret_cast、C 样式转换或带圆括号的函数样式强制转换
error C2678: 二进制“?”：没有找到接受“std::basic_ostream<char, std::char_traits<char>>”类型的左操作数的运算符(或没有可接受的转换)
message : 可以是“内置 C++ operator?(int, int)”
message : 尝试匹配参数列表“(std::basic_ostream<char, std::char_traits<char>>, int)”时
```

#### 2、条件表达式使用的三句中，前两句报错，最后一句正确，总结下条件表达式使用时的限制规则 (提示：注意表达式2和表达式3的类型)

表达式1、2、3的类型可以不同，但2、3的类型必须相容。



# §. 基础知识题 – 关系运算、逻辑运算与选择结构

## 8. switch-case语句

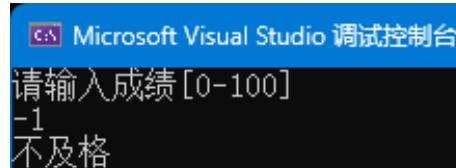
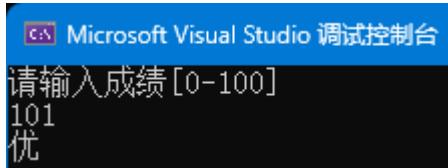
A. 观察下列程序的运行结果，回答问题并将程序的运行结果截图贴上(如果有错则贴错误信息截图)

```
#include <iostream>
using namespace std;
int main()
{
    int score;
    cout<<"请输入成绩[0-100]"<<endl;
    cin >> score;
    switch(score/10) {
        case 10:
        case 9:
            cout<<"优"<<endl;
            break;
        case 8:
            cout<<"良"<<endl;
            break;
        case 7:
            cout<<"中"<<endl;
            break;
        case 6:
            cout<<"及格"<<endl;
            break;
        case 5:
        case 4:
        case 3:
        case 2:
        case 1:
        case 0:
            cout<<"不及格"<<endl;
            break;
        default:
            cout<<"输入错误"<<endl;
            break;
    }
    return 0;
}
```

程序的期望，是当输入的score在[0.. 100]时，分段输出“优/良/中/及格/不及格”，否则输出“输入错误”

1、程序不完全正确，找出不符合期望的两个数据区间并给出运行截图  
(不需要改对)

[101, 109] 和 [-9, -1]





# §. 基础知识题 – 关系运算、逻辑运算与选择结构

## 8. switch-case语句

B. 观察下列程序的运行结果，回答问题并将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()
{
    const int k=5;
    int score;
    cout<<"请输入成绩[0-100]"<<endl;
    cin >> score;
    switch(score/10) {
        case 10:
        case 9:
            cout<<"优"<<endl;
            break;
        case 6:
            cout<<"及格"<<endl;
            break;
        default:
            cout<<"输入错误"<<endl;
            break;
        case k+2:
            cout<<"中"<<endl;
            break;
        case 8:
            cout<<"良"<<endl;
            break;
        case 5:
        case 4:
        case 3:
        case 2:
        case 1:
        case 0:
            cout<<"不及格"<<endl;
            break;
    }
    return 0;
}
```

在8. A的基础上

- 1、将6、8、default的位置进行了交换
- 2、将7写为常变量+常量形式

验证此程序与8. A的功能是否完全一致

(即：8. A中正确的，此程序中同样正确；8. A错误的，此程序中同样错误)

结论：8. A和8. B完全一致



# §. 基础知识题 – 关系运算、逻辑运算与选择结构

## 8. switch-case语句

C. 观察下列程序的运行结果，回答问题并将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 k=5;
    int score;
    cout<<"请输入成绩[0-100]"<<endl;
    cin >> score;
    switch(score/10) {
        case 10:
        case 9:
            cout<<"优"<<endl;
            break;
        case 6:
            cout<<"及格"<<endl;
            break;
        default:
            cout<<"输入错误"<<endl;
            break;
        case k+2:
            cout<<"中"<<endl;
            break;
        case 8:
            cout<<"良"<<endl;
            break;
        case 5:
        case 4:
        case 3:
        case 2:
        case 1:
        case 0:
            cout<<"不及格"<<endl;
            break;
    }

    return 0;
}
```

在8. B的基础上，将k从const int改为int

1、给出编译错误的截图

```
error C2131: 表达式的计算结果不是常数
message : 因读取超过生命周期的变量而失败
message : 请参见“k”的用法
error C2051: case 表达式不是常量
```

2、解释错误原因

case后面应该加整型常量表达式，不能加变量表达式，否则不符合语法规则。



# §. 基础知识题 – 关系运算、逻辑运算与选择结构

## 8. switch-case语句

D. 观察下列程序的运行结果，回答问题并将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 score;
    cout<<"请输入成绩[0-100]"<<endl;
    cin >> score;
    switch(score/10) {
        case 10:
        case 9:
            cout<<"优"<<endl;
            break;
        case 8:
            cout<<"良"<<endl;
            break;
        case 7:
            cout<<"中"<<endl;
            break;
        case 6:
        case 4+2:
            cout<<"及格"<<endl;
            break;
        case 5:
        case 4:
        case 3:
        case 2:
        case 1:
        case 0:
            cout<<"不及格"<<endl;
            break;
        default:
            cout<<"输入错误"<<endl;
            break;
    }
    return 0;
}
```

在8.A的基础上，多了一个case 4+2

1、给出编译错误的截图

error C2196: case 值“6”已使用

2、解释错误原因

各整型常量表达式的值应各不相同。4+2的值为6，与6重复。



## §. 基础知识题 – 关系运算、逻辑运算与选择结构

### 8. switch-case语句

E. 观察下列程序的运行结果，回答问题并将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()
{
    float score;
    cout<<"请输入成绩[0-100]"<<endl;
    cin >> score;
    switch(score/10) {
        case 10:
        case 9:
            cout<<"优"<<endl;
            break;
        case 8:
            cout<<"良"<<endl;
            break;
        case 7:
            cout<<"中"<<endl;
            break;
        case 6:
            cout<<"及格"<<endl;
            break;
        case 5:
        case 4:
        case 3:
        case 2:
        case 1:
        case 0:
            cout<<"不及格"<<endl;
            break;
        default:
            cout<<"输入错误"<<endl;
            break;
    }
    return 0;
}
```

在8.A的基础上，将score从int改为float

1、给出编译错误的截图

error C2450: 类型为“float”的 switch 表达式无效  
message : 要求整型表达式

2、解释错误原因

表达式可以是任何类型，但最终取值应该为整型。score为float型数据，  
score/10的结果也为float型数据，最终取值不是整型。



# §. 基础知识题 – 关系运算、逻辑运算与选择结构

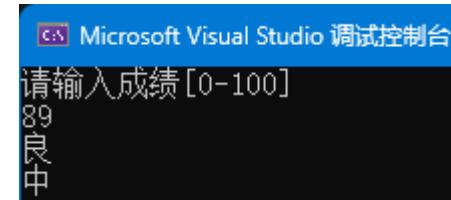
## 8. switch-case语句

F. 观察下列程序的运行结果，回答问题并将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 score;
    cout<<"请输入成绩[0-100]"<<endl;
    cin >> score;
    switch(score/10) {
        case 10:
        case 9:
            cout<<"优"<<endl;
            break;
        case 8:
            cout<<"良"<<endl;
            break;
        case 7:
            cout<<"中"<<endl;
            break;
        case 6:
            cout<<"及格"<<endl;
            break;
        case 5:
        case 4:
        case 3:
        case 2:
        case 1:
        case 0:
            cout<<"不及格"<<endl;
            break;
        default:
            cout<<"输入错误"<<endl;
            break;
    }
    return 0;
}
```

在8. A的基础上，删除case 8后面的break

1、给出与8. A运行结果不一致的测试数据即截图



2、解释break的作用  
提前结束循环体的循环。



# §. 基础知识题 – 关系运算、逻辑运算与选择结构

## 8. switch-case语句

G. 观察下列程序的运行结果，回答问题并将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 score;
    cout<<"请输入成绩[0-100]"<<endl;
    cin >> score;
    switch(score/10) {
        case 10:
        case 9:
            cout<<"优"<<endl;
            break;
        case 8:
            cout<<"良"<<endl;
            break;
        case 7:
            cout<<"中"<<endl;
            break;
        case 6:
            cout<<"及格"<<endl;
            break;
        case 5:
        case 4:
        case 3:
        case 2:
        case 1:
        case 0:
            cout<<"不及格"<<endl;
            break;
        default:
            cout<<"输入错误"<<endl;
            break;
    }
    return 0;
}
```

程序同8.A，将其改正确，即所有[0.. 100]之外的数据均给出“输入错误”即可

```
#include <iostream>
using namespace std;
int main()
{
    int score;
    cout << "请输入成绩[0-100]" << endl;
    cin >> score;
    if (score >= 0 && score <= 100) {
        switch (score / 10) {
            case 10:
            case 9:
                cout << "优" << endl;
                break;
            case 8:
                cout << "良" << endl;
                break;
            case 7:
                cout << "中" << endl;
                break;
            case 6:
                cout << "及格" << endl;
                break;
            default:
                cout << "不及格" << endl;
                break;
        }
    } else
        cout << "输入错误" << endl;
    cout << "程序结束" << endl;
    return 0;
}
```



# §. 基础知识题 – 关系运算、逻辑运算与选择结构

## 8. switch-case语句

### H. 思考

如果将成绩区间对应为：

[85-100] - 优

[70-85) - 良

[60-70) - 及格

[0-60) - 不及格

1、用if-else语句完成该程序  
并贴图

2、如果用switch语句，  
该如何实现？

The image shows two side-by-side code editors, both titled "My Project.cpp".

**Left Editor (Original if-else Solution):**

```
1 #include <iostream>
2 using namespace std;
3
4 int main()
5 {
6     int score;
7     cout << "请输入成绩[0-100]" << endl;
8     cin >> score;
9     if (score >= 85 && score <= 100)
10    {
11        cout << "优" << endl;
12    }
13    else if (score >= 70 && score < 85)
14    {
15        cout << "良" << endl;
16    }
17    else if (score >= 60 && score < 70)
18    {
19        cout << "及格" << endl;
20    }
21    else if (score >= 0 && score < 60)
22    {
23        cout << "不及格" << endl;
24    }
25    else
26    {
27        cout << "输入错误" << endl;
28    }
29    cout << "程序结束" << endl;
30    return 0;
31 }
```

**Right Editor (Switch Statement Solution):**

```
1 #include <iostream>
2 using namespace std;
3
4 int main()
5 {
6     int score;
7     cout << "请输入成绩[0-100]" << endl;
8     cin >> score;
9     if (score >= 0 && score <= 100) {
10         switch (score / 5) {
11             case 20:
12             case 19:
13             case 18:
14             case 17:
15                 cout << "优" << endl;
16                 break;
17             case 16:
18             case 15:
19             case 14:
20                 cout << "良" << endl;
21                 break;
22             case 13:
23             case 12:
24                 cout << "及格" << endl;
25                 break;
26             default:
27                 cout << "不及格" << endl;
28                 break;
29         }
30     }
31     else {
32         cout << "输入错误" << endl;
33     }
34 }
35 }
```



## §. 基础知识题 – 关系运算、逻辑运算与选择结构

### 8. switch-case语句

#### H. 思考

3、如果学生成绩带小数点，即“xx. 5”形式，能用if语句吗？能用switch语句吗？请解释原因

能用if语句，不能用switch语句，因为switch语句表达式可以是任何类型，但最终取值应该为整型，“xx. 5”形式为浮点型。

4、总结switch语句使用时的注意事项

- (1) 表达式可以是任何类型，最终取值为整型即可
- (2) 当整型表达式的取值与整型常量表达式1-n中的任意一个相等时，执行对应的语句序列；否则执行default后的语句序列（不能是实数，因为无法直接判断相等）
- (3) 各整型常量表达式的值应各不相同，但顺序无要求
- (4) 各语句序列的最后一句应是break，否则连续执行下一case语句，最后一个可省
- (5) 语句序列不必加{}
- (6) 多个case可以共用一组语句
- (7) 不能完全替代多重if语句的嵌套

5、switch-case语句能完全取代if-else吗？

不能。



## §. 基础知识题 - 循环结构

### 1. 循环的嵌套

A. 观察下列程序的运行结果，回答问题并将程序的运行结果截图贴上

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

int main()
{
    int i, j, k;
    int count1 = 0, count2 = 0, count3 = 0;

    for(i=1; i<=100; i++) {
        ++count1;
        for(j=1; j<=100; j++) {
            ++count2;
            for(k=1; k<=100; k++)
                ++count3;
        }
    }

    cout << "count1=" << count1 << endl;
    cout << "count2=" << count2 << endl;
    cout << "count3=" << count3 << endl;
    return 0;
}
```

#### 1、贴运行结果

```
Microsoft Visual Studio 调试控制台
count1=100
count2=10000
count3=1000000
```

2、当循环嵌套时，内层循环的执行次数和外层循环是什么关系？

外层循环每执行一次，内层循环都要执行一遍。因此嵌套循环的总执行次数等于内层循环的执行次数与外层循环的执行次数的乘积。



## §. 基础知识题 - 循环结构

### 1. 循环的嵌套

B. 观察下列程序的运行结果，回答问题并将程序的运行结果截图贴上

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

int main()
{
    int i, j, k;
    int count1 = 0, count2 = 0, count3 = 0;

    for(i=1; i<=100; i++) {
        ++count1;
        for(j=i; j<=100; j++) {
            ++count2;
            for(k=j; k<=100; k++)
                ++count3;
        }
    }

    cout << "count1=" << count1 << endl;
    cout << "count2=" << count2 << endl;
    cout << "count3=" << count3 << endl;
    return 0;
}
```

1、贴运行结果

2、当循环嵌套时，内层循环的执行次数和外层循环是什么关系？

Microsoft Visual Studio 调试控制台  
count1=100  
count2=5050  
count3=171700

第一层循环 第i次执行	第二层循环 的执行次数	第三层循环的执行次数		
1	100	5050	= 1+2+...+100	= (1+100)*100/2
2	99	4950 (比上次的执行次数少100)	= 1+2+...+99	= (1+99)*99/2
3	98	4851 (比上次的执行次数少99)	= 1+2+...+98	= (1+98)*98/2
4	97	4753 (比上次的执行次数少98)	= 1+2+...+97	= (1+97)*97/2
5	96	4656 (比上次的执行次数少97)	= 1+2+...+96	= (1+96)*96/2
...	...	...	= ...	= ...
i	101-i	(比上次的执行次数少100-i)	= 1+2+...+(101-i)	= (1+101-i)*(101-i)/2
...	...	...	= ...	= ...
96	5	15 (比上次的执行次数少6)	= 1+2+3+4+5	= (1+5)*5/2
97	4	10 (比上次的执行次数少5)	= 1+2+3+4	= (1+4)*4/2
98	3	6 (比上次的执行次数少4)	= 1+2+3	= (1+3)*3/2
99	2	3 (比上次的执行次数少3)	= 1+2	= 1+2
100	1	1 (比上次的执行次数少2)	= 1	= 1

第一层循环 总执行次数	第二层循环 总执行次数	第三层循环总执行次数
100	$(1+100)*100/2$ =5050	$\sum_{i=1}^{100} \frac{(1+101-i)(101-i)}{2} = 171700$

关系：当第一层循环第i次执行时，第二层循环的执行次数为 $100-i$ ，第三层循环的执行次数为 $(1+101-i)*(101-i)/2$ ；当第一层循环第i次执行，第二层循环第j次执行时，第三层循环的执行次数为 $(101-i)-(j-1)$ 。



## §. 基础知识题 - 循环结构

### 1. 循环的嵌套

C. 观察下列程序的运行结果，回答问题并将程序的运行结果截图贴上

```
#include <iostream>
#include <conio.h>
using namespace std;

int main()
{
    int i, j, count = 0;
    for(i=1; i<=100; i++) {
        for(j=1; i<=100; j++) {
            ++count;
            if (count % 1000 == 0) {
                cout << "*";
                _getch();
            }
        }
    }

    cout << "count = " << count << endl;
    return 0;
}

//注意：这个程序无法通过按CTRL+C终止，要关窗口
```

#### 1、贴运行结果



D:\Learning\高级语言程序设计\My Project\Debug\My Project.exe

```
*****
```

2、按内外for循环的执行步骤依次分析，为什么会得到这个结果？

第 1 步 - 外循环表达式1 - i=1  
第 2 步 - 外循环表达式2 - i<=100为真  
第 3 步 - 内循环表达式1 - j=1  
第 4 步 - 内循环表达式2 - i<=100为真  
第 5 步 - ++count - count值为1  
第 6 步 - 条件表达式 - count%1000==0为假  
第 7 步 - j++ - j的值为2  
第 8 步 - 内循环表达式2 - i<=100为真  
第 9 步 - ++count - count值为2  
第 10步 - 条件表达式 - count%1000==0为假  
...  
第 n 步 - ++count - count值为1000  
第n+1步 - 条件表达式 - count%1000==0为真  
第n+2步 - cout << "\*";  
第n+3步 - \_getch();  
第n+3步 - j++ - j的值为1001  
第n+4步 - 内循环表达式2 - i<=100为真  
...  
第 m 步 - ++count - count值为2000  
第m+1步 - 条件表达式 - count%1000==0为真  
第m+2步 - cout << "\*";  
第m+3步 - \_getch();  
...

#### 原因分析：

内层for循环的表达式2为*i<=100*，在内层for循环中没有能改变表达式2取值的语句，因此内层for循环为永真循环。当*count%1000==0*为真时，执行if单支语句序列，输出一个\*，并使用`_getch()`函数输入一个字符。但是由于`_getch()`函数无回显，不需要回车键且`_getch()`函数的返回值被忽略，因此造成了每次在键盘上输入一个字符，程序都会输出一个\*，并且暂停等待输入下一个字符的效果。



## §. 基础知识题 - 循环结构

### 2. break与continue

A. 已知代码如下，回答问题

```
while(1) {  
    ①  
    ②  
    if (X)  
        continue;  
    ③  
    ④  
}
```

当X为真时，重复执行①②

当X为假时，重复执行①②③④

```
for(1; 1; ④) {  
    ①  
    ②  
    if (X)  
        continue;  
    ③  
}
```

当X为真时，重复执行①②④

当X为假时，重复执行①②③④

## §. 基础知识题 - 循环结构



### 2. break与continue

B. 观察下列程序的运行结果，回答问题并将程序的运行结果截图贴上

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

int main()
{
    int i=0, sum=0;

    while(i<1000) {
        i++;
        break;
        sum=sum+i;
    }

    cout << "i=" << i << endl;
    cout << "sum=" << sum << endl;

    return 0;
}

//问题1：循环执行了多少次？1次
//问题2：sum=sum+i执行了多少次？0次
```

Microsoft Visual Studio 调试控制台  
i=1  
sum=0

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

```
int main()
{
    int i=0, sum=0;
```

```
while(i<1000) {
    i++;
    continue;
    sum=sum+i;
}
```

```
cout << "i=" << i << endl;
cout << "sum=" << sum << endl;
```

```
return 0;
}
```

//问题1：循环执行了多少次？1000次  
//问题2：sum=sum+i执行了多少次？0次

Microsoft Visual Studio 调试控制台  
i=1000  
sum=0



## §. 基础知识题 - 循环结构

### 3. 观察程序运行结果

A. 观察下列程序的运行结果，回答问题并将程序的运行结果截图贴上

```
#include <iostream>
#include <iomanip> //格式输出
#include <cmath> //fabs
#include <windows.h> //取系统时间
using namespace std;

int main()
{
    int s=1;
    double n=1, t=1, pi=0;

    LARGE_INTEGER tick, begin, end;
    QueryPerformanceFrequency(&tick); //取计数器频率
    QueryPerformanceCounter(&begin); //取初始硬件定时器计数

    while(fabs(t)>1e-6) {
        pi=pi+t;
        n=n+2;
        s=-s;
        t=s/n;
    }

    QueryPerformanceCounter(&end); //获得终止硬件定时器计数

    pi=pi*4;
    cout << "n=" << setprecision(10) << n << endl;
    cout << "pi=" << setiosflags(ios::fixed) << setprecision(9) << pi << endl;

    cout << "计数器频率：" << tick.QuadPart << "Hz" << endl;
    cout << "时钟计数：" << end.QuadPart - begin.QuadPart << endl;
    cout << setprecision(6) << (end.QuadPart - begin.QuadPart)/double(tick.QuadPart) << "秒" << endl;

    return 0;
}
```

用下面的迭代公式求Pi的值

$$1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \dots$$

(1) n, t, pi为double型

精度为1e-6: n=1000001	pi=3.141590654	时间=0.002594(秒)
1e-7: n=10000001	pi=3.141592454	时间=0.020412(秒)
1e-8: n=100000001	pi=3.141592634	时间=0.203840(秒)
1e-9: n=1000000001	pi=3.141592652	时间=1.934466(秒)

(2) n, t, pi为float型

精度为1e-6: n=1000001	pi=3.141593933	时间=0.018857(秒)
1e-7: n=10000001	pi=3.141596556	时间=0.183082(秒)
1e-8: 无结果		

问：1、7项中哪个没结果？为什么？

n, t, pi为float型，精度为1e-8时没结果。因为精度1e-8已经超出了float型数据的精度，fabs(t)>1e-8永真，循环为永真循环，故无结果。

2、float和double同精度下哪个时间快？

float和double同精度下double时间快。



## §. 基础知识题 - 循环结构

### 3. 观察程序运行结果

B. 观察下列程序的运行结果，回答问题并将程序的运行结果截图贴上

```
My Project.cpp  □ ×
My Project      (全局范围)

1 #include <iostream>
2 #include <iomanip>
3
4 using namespace std;
5
6 int main()
7 {
8     int n = 0, i, m, k;
9     bool prime;
10
11    for (m = 101; m <= 200; m += 2) { //偶数没必要判断
12        prime = true;           //对每个数，先认为是素数
13        k = int(sqrt(m));
14
15        for (i = 2; i <= k; i++)
16            if (m % i == 0) {
17                prime = false;
18                break;
19            }
20
21        if (prime) {
22            cout << setw(5) << m;
23            n = n + 1; //计数器，只为了加输出换行
24
25            if (n % 10 == 0) //每10个数输出一行
26                cout << endl;
27        }
28    } //end of for
29
30    return 0;
31 }
```

(1) 目前输出结果：一共21个，每10个一行

Microsoft Visual Studio 调试控制台

101	103	107	109	113	127	131	137	139	149
151	157	163	167	173	179	181	191	193	197
199									

(2) 将m的初值从101改为103，应该是20个，共2行。  
实际呢？为什么？

Microsoft Visual Studio 调试控制台

103	107	109	113	127	131	137	139	149	151
157	163	167	173	179	181	191	193	197	199

实际是20个，  
共4行。

因为：  
当m的值为151时， prime的值为true， n的值为  
10， n%10==0为真，换行；  
当m的值为153时， prime的值为false， n的值仍  
为10， n%10==0为真，换行；  
当m的值为155时， prime的值为false， n的值仍  
为10， n%10==0为真，换行；  
当m的值为157时， prime的值为true， n的值为  
11， n%10==0为假，不换行。  
所以最终显示20个数据，共4行。

(3) 将左侧程序改正确



## §. 基础知识题 - 函数基础

### 1. C和C++的不写函数返回类型时的差异

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

```
#include <stdio.h>

fun()
{
    /* 注意：输出必须改为自己学号-姓名 */
    printf("2250758-林继申\n");
    return 0;
}

int main()
{
    fun();
    return 0;
}
```

1、将左侧程序贴到. c后缀的源程序中

2、将左侧程序贴到. cpp后缀的源程序中

3、结论:

如果C程序不写函数的返回类型，则编译器不报错，函数可以被正常调用，C程序可以正常运行。

如果C++程序不写函数的返回类型，则编译器报错，函数不能被正常调用，C++程序不能正常运行。



## §. 基础知识题 - 函数基础

1. C和C++的不写函数返回类型时的差异

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

```
#include <stdio.h>
int main()
{
    printf("%lf\n", sqrt(2));
    return 0;
}
```

Microsoft Visual Studio 调试控制台  
0.000000

warning C4013: “sqrt”未定义；假设外部返回 int  
warning C4477: “printf”：格式字符串“%lf”需要类型“double”的参数，但可变参数 1 拥有了类型“int”

将左侧的两个程序贴到.c后缀的源程序中编译运行，分析为什么结果不同

sqrt() 函数是math.h头文件中的函数，前者程序中未加math.h头文件，故sqrt()函数未定义，假设外部返回int型，并返回值为0，所以输出0.000000；后者程序中添加math.h头文件，故sqrt(2)返回值为2的算术平方根，所以输出1.414214。

```
#include <stdio.h>
#include <math.h>

int main()
{
    printf("%lf\n", sqrt(2));
    return 0;
}
```

Microsoft Visual Studio 调试控制台  
1.414214



## §. 基础知识题 - 函数基础

### 2. main函数的返回值差异

A. 观察下列程序的运行结果，回答问题并将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;

main()
{
    /* 注意：输出必须改为自己学号-姓名 */
    cout << "2250758-林继申" << endl;

    return 0;
}
```

1、VS下编译

error C4430: 缺少类型说明符 - 假定为 int。注意: C++ 不支持默认 int

2、在Dev下编译





## §. 基础知识题 - 函数基础

### 2. main函数的返回值差异

B. 观察下列程序的运行结果，回答问题并将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;

void main()
{
    /* 注意：输出必须改为自己学号-姓名 */
    cout << "2250758-林继申" << endl;

    return;
}
```

#### 1、VS下编译

```
Microsoft Visual Studio 调试控制台
2250758-林继申
warning C4326: “main”的返回类型应为“int”而非“void”
```

#### 2、在Dev下编译

```
[Error] '::main' must return 'int'
In function 'int main()':
[Error] return-statement with no value, in function returning 'int' [-fpermissive]
```



## §. 基础知识题 - 函数基础

### 2. main函数的返回值差异

C. 观察下列程序的运行结果，回答问题并将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;

long main()
{
    /* 注意：输出必须改为自己学号-姓名 */
    cout << "2250758-林继申" << endl;

    return 0L;
}
```

1、VS下编译

Microsoft Visual Studio 调试控制台  
2250758-林继申  
warning C4326: “main”的返回类型应为“int”而非“long”

2、在Dev下编译

[Error] '::main' must return 'int'

3、综合2.A/2.B/2.C三题的结论，main函数的返回类型应定义为int最合适。



## §. 基础知识题 - 函数基础

### 3. 函数的单向传值

A. 观察下列程序的运行结果，回答问题并将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;

void fun(int x)
{
    cout << "x1=" << x << endl;
    x=5;
    cout << "x2=" << x << endl;
}

int main()
{
    int k=15;
    cout << "k1=" << k << endl;
    fun(k);
    cout << "k2=" << k << endl;

    /* 注意：输出必须改为自己学号-姓名 */
    cout << "2250758-林继申" << endl;
    return 0;
}
```

#### 1、运行结果

```
Microsoft Visual Studio 调试控制台
k1=15
x1=15
x2=5
k2=15
2250758-林继申
```

#### 2、为什么x的改变不会影响到k？

因为参数的传递方式是“单向传值”，即将实参的值复制一份到形参中，可以理解为“形参 = 实参”的形式，所以执行后形参的变化不影响实参值，形参在使用时分配空间，函数运行结束后释放空间。



## §. 基础知识题 - 函数基础

### 4. 函数的返回值

A. 观察下列程序的运行结果，回答问题并将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 fun(short x)
{
    cout << "x=" << x << endl;
    return 0;
}

int main()
{
    long k=70000;
    fun(k);
    cout << "k=" << k << endl;

    /* 注意：输出必须改为自己学号-姓名 */
    cout << "2250758-林继申" << endl;
    return 0;
}
```

#### 1、运行结果 (含warning)

```
Microsoft Visual Studio 调试控制台
x=4464
k=70000
2250758-林继申
```

warning C4244: “参数”：从“long”转换到“short”，可能丢失数据

#### 2、用第2章的知识分析并解释运行结果

参数的传递方式是“单向传值”，即将实参的值复制一份到形参中，可以理解为“形参 = 实参”的形式，即 short x = k, 70000的二进制补码为10001000101110000, 赋值给short型时按照不同长度的整型数据相互赋值的规则进行赋值，高位丢弃得到二进制补码为1000101110000, 转换为十进制数为4464，即x的值。

## §. 基础知识题 – 函数基础



## 4. 函数的返回值

B. 观察下列程序的运行结果，回答问题并将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;

short fun3()
{
    long a = 70000;
    return a;
}

int main()
{
    long d;
    d = fun3();
    cout << d << endl;
}

/* 注意：输出必须改为自己学号-姓名 */
cout << "2250758-林继申" << endl;
return 0;
}
```

## 1、运行结果（含warning）

Microsoft Visual Studio 调试控制台  
4464  
2250758-林继申

warning C4244: “return”：从“long”转换到“short”，可能丢失数据

## 2、用第2章的知识分析并解释运行结果

调用函数fun3()时，long型变量a的值为70000，函数fun3()的返回值为short型的a，因此进行数据类型强制转换，70000的二进制补码为10001000101110000，赋值给short型时按照不同长度的整型数据相互赋值的规则进行赋值，高位丢弃得到二进制补码为1000101110000，转换为十进制数为4464，即fun3()的返回值，之后再赋值给long型变量d，进行整型提升操作后得到二进制补码为00000000000000001000101110000，转换为十进制数为4464，即d的值。



## §. 基础知识题 - 函数基础

### 4. 函数的返回值

C. 观察下列程序的运行结果，回答问题并将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;

void fun()
{
    /* 注意：输出必须改为自己学号-姓名 */
    cout << "2250758-林继申" << endl;
}

int main()
{
    fun();
}

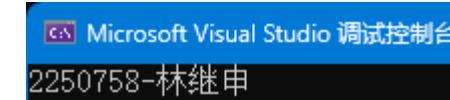
//main函数，返回类型为int，无return
```

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

int fun()
{
    /* 注意：输出必须改为自己学号-姓名 */
    cout << "2250758-林继申" << endl;
}

int main()
{
    fun();
    return 0;
}

//非main函数，返回类型为int，无return
```



error C4716: “fun”：必须返回一个值



## §. 基础知识题 - 函数基础

### 4. 函数的返回值

D. 观察下列程序的运行结果，回答问题并将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;

void f()
{
    int x=10;
}

int main()
{
    int k=10;

    k = k + f();
    k, f();
    cout << (k, f()) << endl;
    cout << (k, f(), k+2) << endl;

    return 0;
}
```

#### 1、编译结果

```
error C2186: “+”：“void”类型的操作数非法
error C2679: 二元“<<”：没有找到接受“void”类型的右操作数的运算符(或没有可接受的转换)
```

#### 2、解释报error的两行为什么错

void表示不需要返回类型，f()无返回类型，不能参与运算，所以显示error C2186：“+”：“void”类型的操作数非法；

逗号表达式(k, f())的值为f()，但f()无返回类型，不能参与运算，所以显示error C2679：二元“<<”：没有找到接受“void”类型的右操作数的运算符(或没有可接受的转换)。

#### 3、结论：

- ① 返回类型为void的函数不能出现在除逗号表达式外的任何表达式中
- ② 若逗号表达式要参与其它运算/输出，则返回类型为void的函数不能做为第n个表达式出现  
(假设逗号表达式有n个表达式组成，此处填1~n)



## §. 基础知识题 - 函数基础

### 4. 函数的返回值

E. 观察下列程序的运行结果，回答问题并将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 fun(int x)
{
    if (x>10) {
        if (x>20)
            return 1;
    }
    else
        return 0;
}

int main()
{
    /* 注意：输出必须改为自己学号-姓名 */
    cout << "2250758-林继申" << endl;
    cout << fun(5) << endl;
    cout << fun(15) << endl;
    cout << fun(25) << endl;
    return 0;
}
```

#### 1、运行结果

```
Microsoft Visual Studio 调试控制台
2250758-林继申
0
7139431
1
```

warning C4715: “fun” : 不是所有的控件路径都返回值

#### 2、解释warning的含义

由于函数中的分支语句条件覆盖不全面，return未覆盖全部分支/出口，指定的函数可能无法返回值。如x的值为15时，x>10为真，x>20为假，无return语句，故这时函数的返回值不可信。

#### 3、后三行输出中哪行不可信？

后三行输出中第二行的输出不可信。



## §. 基础知识题 - 函数基础

### 4. 函数的返回值

F. 观察下列程序的运行结果，回答问题并将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 fun(int x)
{
    if (x>1)
        return 1;
    else if (x<=1)
        return 0;
} //if+else if已覆盖int型的全部表示范围

int main()
{
    /* 注意：输出必须改为自己学号-姓名 */
    cout << "2250758-林继申" << endl;

    cout << fun(0) << endl;
    cout << fun(2) << endl;
    return 0;
}
```

#### 1、运行结果

```
Microsoft Visual Studio 调试控制台
2250758-林继申
0
1
```

warning C4715: “fun” : 不是所有的控件路径都返回值

2、解释warning的含义（编译器是如何理解fun的逻辑的）  
编译器不会理解到if已覆盖int型的全部表示范围，  
编译器只会理解函数中的分支语句条件覆盖不全面，  
return未覆盖全部分支/出口，所以指定的函数可能无法  
返回值。



## §. 基础知识题 - 函数基础

### 5. 函数的调用

A. 观察下列程序的运行结果，回答问题并将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;

long fun(void)
{
    /* 注意：输出必须改为自己学号-姓名 */
    cout << "2250758-林继申" << endl;
    return 0L;
}

int main()
{
    long k;
    k = fun();
    return 0;
}
```

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

long fun(void)
{
    /* 注意：输出必须改为自己学号-姓名 */
    cout << "2250758-林继申" << endl;
    return 0L;
}

int main()
{
    long k;
    k = long fun();
    return 0;
}
```



error C2062: 意外的类型“long”

结论：函数调用时，不能写返回类型



## §. 基础知识题 - 函数基础

### 5. 函数的调用

B. 观察下列程序的运行结果，回答问题并将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 fun(void)
{
    /* 注意：输出必须改为自己学号-姓名 */
    cout << "2250758-林继申" << endl;
    return 0;
}

int main()
{
    int k;
    k = fun();
    return 0;
}
```



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

int fun(void)
{
    /* 注意：输出必须改为自己学号-姓名 */
    cout << "2250758-林继申" << endl;
    return 0;
}

int main()
{
    int k;
    k = fun(void);
    return 0;
}
```

```
error C2144: 语法错误：“void” 的前面应有“)”
error C2144: 语法错误：“void” 的前面应有“:”
error C2059: 语法错误：“)” 
warning C4091: “”: 没有声明变量时忽略“void”的左侧
```

结论：无参函数调用时，参数位置不能写void



## §. 基础知识题 - 函数基础

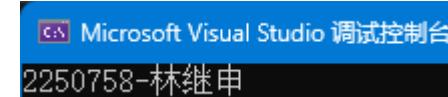
### 5. 函数的调用

B. 观察下列程序的运行结果，回答问题并将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 fun(int x, int y)
{
    /* 注意：输出必须改为自己学号-姓名 */
    cout << "2250758-林继申" << endl;
    return 0;
}

int main()
{
    int k, x=10, y=15;
    k = fun(x, y);
    return 0;
}
```



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

int fun(int x, int y)
{
    /* 注意：输出必须改为自己学号-姓名 */
    cout << "2250758-林继申" << endl;
    return 0;
}

int main()
{
    int k, x=10, y=15;
    k = fun(int x, int y);
    return 0;
}
```

```
(13,13): error C2144: 语法错误: “int”的前面应有“ ”
(13,9): error C2660: “fun”: 函数不接受 0 个参数
(4,5): message : 参见“fun”的声明
(13,9): message : 尝试匹配参数列表“()”时
(13,13): error C2144: 语法错误: “int”的前面应有“:”
(13,17): error C2086: “int x”: 重定义
(12,12): message : 参见“x”的声明
(13,20): error C2062: 意外的类型“int”
(13,25): error C2059: 语法错误: “)”
```

结论：有参函数调用时，实参不能写类型



## §. 基础知识题 - 函数基础

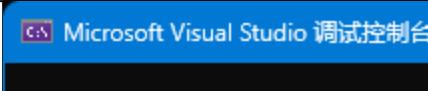
### 5. 函数的调用

C. 观察下列程序的运行结果，回答问题并将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 fun(int x, int y)
{
    /* 注意：输出必须改为自己学号-姓名 */
    cout << "2250758-林继申" << endl;
    cout << "x=" << x << endl;
    cout << "y=" << y << endl;
    return 0;
}

int main()
{
    int x=10, y=15;
    int fun(int x, int y);
    return 0;
}
```



为什么本程序不报错但也无输出？

int fun(int x, int y); 是对fun()函数的调用说明，在调用说明后没有对fun()函数的调用，程序直接运行至语句return 0;，所以本程序不报错但也无输出。



## §. 基础知识题 - 函数基础

### 5. 函数的调用

D. 观察下列程序的运行结果，回答问题并将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 x = 10, y = 15;
    fun(x, y);
    return 0;
}
int fun(int x, int y);
void f()
{
    fun(10, 15);
}
int fun(int x, int y)
{
    /* 注意：输出必须改为自己学号-姓名 */
    cout << "2250758-林继申" << endl;
    cout << "x=" << x << endl;
    cout << "y=" << y << endl;
    return 0;
}
```

error C3861: “fun” : 找不到标识符

```
#include <iostream>
using namespace std;
int main()
{
    int fun(int x, int y);
    int x = 10, y = 15;
    fun(x, y);
    return 0;
}
void f()
{
    fun(10, 15);
}
int fun(int x, int y)
{
    /* 注意：输出必须改为自己学号-姓名 */
    cout << "2250758-林继申" << endl;
    cout << "x=" << x << endl;
    cout << "y=" << y << endl;
    return 0;
}
```

error C3861: “fun” : 找不到标识符

结论：1、函数声明如果放在函数外，则对哪些函数有效？

针对后面所有函数都有效，函数声明后面的所有被声明函数都能被调用。

2、函数声明如果放在函数内，则对哪个函数有效？

只对本函数有效，被声明函数只能在本函数内被调用。



## §. 基础知识题 - 变量类型

### 1. 自动变量及形参的分配与释放

A. 观察下列程序的运行结果，回答问题并将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;

void f1(int x)
{
    int y;
    cout << &x << ',' << &y << endl;//打印地址
}
void f2(long p)
{
    float q;
    cout << &p << ',' << &q << endl;//打印地址
}

int main()
{
    f1(10);
    f2(15L);
    return 0;
}
```

#### 1、运行结果截图及结论

截图：

```
Microsoft Visual Studio 调试控制台
010FFA50 010FFA3C
010FFA50 010FFA3C
```

结论：

1. 1 int型变量x 和 long型变量p 共用了从 010FFA50 (每次运行该地址不确定) 开始的4个字节空间。

1. 2 int型变量y 和 float型变量q 共用了从 010FFA3C (每次运行该地址不确定) 开始的4个字节空间。

#### 2、把f2中float q改为short q，运行结果截图及结论 截图：

```
Microsoft Visual Studio 调试控制台
008FFA98 008FFA84
008FFA98 008FFA84
```

结论：

2. 1 q和y 共用了从 008FFA84 (每次运行该地址不确定) 开始的 2 个字节空间。



## §. 基础知识题 - 变量类型

### 1. 自动变量及形参的分配与释放

B. 观察下列程序的运行结果，回答问题并将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;

void f1(int x)
{
    int y;
    cout << &x << ',' << &y << endl;
}

int main()
{
    f1(10);
    cout << "..." << endl;

    f1(10);
    cout << "..." << endl;

    f1(10);
    cout << "..." << endl;
    return 0;
}
```

#### 1、运行结果截图

Microsoft Visual Studio 调试控制台  
0094FD44 0094FD30  
..  
0094FD44 0094FD30  
..  
0094FD44 0094FD30  
...

#### 2、结论：

2.1 本示例中，三次调用时分配的x占用相同空间，  
三次调用时分配的y占用相同空间。

2.2 总结形参x和自动变量y的分配和释放规则  
形参在使用时分配空间，函数运行结束后释放空  
间；自动变量在函数进入后分配空间，函数运行结束  
后释放空间（重复进行）。



## §. 基础知识题 - 变量类型

### 1. 自动变量及形参的分配与释放

C. 观察下列程序的运行结果，回答问题并将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;

void f1()
{
    int a = 15;
    cout << &a << ', ' << a << endl;
}

void f2()
{
    long a = 70000;
    cout << &a << ', ' << a << endl;
}

void f3()
{
    short a = 23;
    cout << &a << ', ' << a << endl;
}

int main()
{
    f1();
    f2();
    f3();
    return 0;
}
```

#### 1、运行结果截图

```
Microsoft Visual Studio 调试控制台
0073FC44 15
0073FC44 70000
0073FC44 23
```

#### 2、结论：

2. 1 f1/f2/f3中的三个a占用相同空间。

2. 2 如果当前正在执行f2函数，则f1中的a已释放，  
f3中的a未分配。



## §. 基础知识题 - 变量类型

### 1. 自动变量及形参的分配与释放

D. 观察下列程序的运行结果，回答问题并将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;

void f3()
{
    short a = 23;
    cout << "f3" << &a << ',' << a << endl;
}
void f2()
{
    long a = 70000;
    cout << "f2-1" << &a << ',' << a << endl;
    f3();
    cout << "f2-2" << endl;
}
void f1()
{
    int a = 15;
    cout << "f1-1" << &a << ',' << a << endl;
    f2();
    cout << "f1-2" << endl;
}
int main()
{
    f1();
    return 0;
}
```

#### 1、运行结果截图

```
f1-1 012FFC70 15
f2-1 012FFB8C 70000
f3 012FFAA8 23
f2-2
f1-2
```

#### 2、结论：

2. 1 f1/f2/f3中的三个a占用不同空间
2. 2 如果当前正在执行f1函数的cout-1语句，则f2中的a未分配，f3中的a未分配；
2. 3 如果当前正在执行f1函数的cout-2语句，则f2中的a已释放，f3中的a已释放；
2. 4 如果当前正在执行f2函数的cout-1语句，则f1中的a已分配，f3中的a未分配；
2. 5 如果当前正在执行f2函数的cout-2语句，则f1中的a已分配，f3中的a已释放；
2. 6 如果当前正在执行f3函数的cout语句，则f1中的a已分配，f2中的a已分配。
2. 7 上述2.2~2.6问题中如果某个a是已分配状态，则此时这个a在何处?  
a在“现场栈”中。



## §. 基础知识题 - 变量类型

### 2. 局部变量的作用范围

A. 观察下列程序的运行结果，回答问题并将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;
void fun()
{
    int i, a;
    a=15;
    for(i=0;i<10;i++) {
        int y;
        y=11;
        a=16;
    }
    y=12;
    a=17;
}
int main()
{
    /* 注意：输出必须改为自己学号-姓名 */
    cout << "2250758-林继申" << endl;
    fun();
    return 0;
}
```

注：如果是error，贴error截图  
如果是warning，贴warning截图+运行结果  
如果正常，贴运行结果

#### 1、截图

error C2065: “y” : 未声明的标识符

#### 2、解释出现的error/warning的原因

循环语句内的变量，只在循环语句中有效，自动变量在进入循环后分配空间，退出循环后释放空间，所以“y=12”处会显示error C2065: “y” : 未声明的标识符，因为在进入循环后分配空间的int型变量y在退出循环后已经释放空间。



## §. 基础知识题 - 变量类型

### 2. 局部变量的作用范围

B. 观察下列程序的运行结果，回答问题并将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;

void fun()
{
    int i, a=15;
    {
        int y;
        y=11;
        a=16;
        {
            int w=10;
            y=12;
            a=13;
            w=14;
        }
        w=15;
    }
    y=12;
    a=17;
}
int main()
{
    /* 注意：输出必须改为自己学号-姓名 */
    cout << "2250758-林继申" << endl;
    fun();
    return 0;
}
```

注：如果是error，贴error截图  
如果是warning，贴warning截图+运行结果  
如果正常，贴运行结果

#### 1、截图

```
error C2065: “w” : 未声明的标识符
error C2065: “y” : 未声明的标识符
```

#### 2、解释出现的error/warning的原因

复合语句内的变量，只在复合语句中有效，自动变量在进入复合语句后分配空间，退出复合语句后释放空间，所以“w=12”处会显示error C2065：“w”：未声明的标识符，“y=12”处会显示error C2065：“y”：未声明的标识符，因为在进入复合语句后分配空间的int型变量w和int型变量y在退出复合语句后已经释放空间。



## §. 基础知识题 - 变量类型

### 2. 局部变量的作用范围

C. 观察下列程序的运行结果，回答问题并将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;

void fun()
{
    a=14;
}

int main()
{
    /* 注意：输出必须改为自己学号-姓名 */
    cout << "2250758-林继申" << endl;

    int a;
    a=15;
    fun();
    a=16;
    return 0;
}
```

注：如果是error，贴error截图  
如果是warning，贴warning截图+运行结果  
如果正常，贴运行结果

#### 1、截图

error C2065: “a” : 未声明的标识符

2、结论：在某个函数(main)中定义的自动变量，在它的调用函数(fun)中不允许访问。



## §. 基础知识题 - 变量类型

### 3. 全局变量的作用范围

A. 观察下列程序的运行结果，回答问题并将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 f1()
{
    a=15;
}

int a;
int main()
{
    /* 注意：输出必须改为自己学号-姓名 */
    cout << "2250758-林继申" << endl;
    a=16;
    return 0;
}

int f2()
{
    a=17;
}
```

注：如果是error，贴error截图  
如果是warning，贴warning截图+运行结果  
如果正常，贴运行结果

#### 1、截图

error C2065: “a” : 未声明的标识符

#### 2、解释出现的error/warning的原因

全局变量从定义点到源文件结束之间的所有函数均可使用。“a=15”处在全局变量定义点之前，不可用，所以“a=15”处显示error C2065: “a” : 未声明的标识符。



## §. 基础知识题 - 变量类型

### 3. 全局变量的作用范围

B. 观察下列程序的运行结果，回答问题并将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 a;
void f1()
{
    a=15;
    cout << "fa=" << a << ',' << &a << endl;
}
int main()
{
    /* 注意：输出必须改为自己学号-姓名 */
    cout << "2250758-林继申" << endl;

    a=10;
    cout << "ma1=" << a << ',' << &a << endl;
    f1();
    cout << "ma2=" << a << ',' << &a << endl;
    return 0;
}
```

注：如果是error，贴error截图  
如果是warning，贴warning截图+运行结果  
如果正常，贴运行结果

#### 1、截图

Microsoft Visual Studio 调试控制台

```
2250758-林继申
ma1=10 0067C13C
fa=15 0067C13C
ma2=15 0067C13C
```

2、由运行结果中的地址可以证明，f1和main中访问的变量a是相同的a。



## §. 基础知识题 - 变量类型

### 3. 全局变量的作用范围

C. 观察下列程序的运行结果，回答问题并将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 f1(int a)
{
    a=15;
    cout << "fa=" << a << ',' << &a << endl;
    return a;
}

int main()
{
    /* 注意：输出必须改为自己学号-姓名 */
    cout << "2250758-林继申" << endl;

    int a =10;
    cout << "ma1=" << a << ',' << &a << endl;
    a = f1(a);
    cout << "ma2=" << a << ',' << &a << endl;
    return 0;
}
```

注：如果是error，贴error截图  
如果是warning，贴warning截图+运行结果  
如果正常，贴运行结果

#### 1、截图

Microsoft Visual Studio 调试控制台  
2250758-林继申  
ma1=10 001CF798  
fa=15 001CF6C4  
ma2=15 001CF798

2、由运行结果中的地址可以证明，f1和main中访问的变量a是不同的a。

3、a不是全局变量，解释为什么ma1和ma2两句cout输出的a值不相同？a是如何被改变的？

因为main()和f1()访问的是不同的a，所以ma1和ma2两句cout输出的a值不相同。实参a是因为赋值语句而改变的：实参a在main()函数中定义并初始化，输出“ma1=10”，之后实参向形参单向传值，输出“fa=15”并用函数返回值改变实参的值，输出“ma2=15”。



## §. 基础知识题 - 变量类型

### 4. 变量同名

A. 观察下列程序的运行结果，回答问题并将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 a=10, b;
void f1()
{
    int a=5, b;
    cout << "a1=" << a << ',' << &a << endl;
    cout << "b1=" << b << ',' << &b << endl;
}
void f2()
{
    cout << "a2=" << a << ',' << &a << endl;
    cout << "b2=" << b << ',' << &b << endl;
}
int main()
{
    f1();
    f2();
    return 0;
}
```

注：如果是error，贴error截图  
如果是warning，贴warning截图+运行结果  
如果正常，贴运行结果

#### 1、截图

Microsoft Visual Studio 调试控制台

a1=5 008FFB74
b1=-858993460 008FFB68
a2=10 0053C008
b2=0 0053C13C

! C6001 使用未初始化的内存“b”。

2、由b可知，局部变量不初始化，初值为-858993460(不确定值)；全局变量不初始化，初值为0。

3、由截图可知，全局变量a/b的起始地址差308个字节；  
局部变量a/b之间差12个字节；全局和局部之间差约2610KB，说明这是两个不同的存储区，全局变量在静态存储区，局部变量在动态存储区。



## §. 基础知识题 - 变量类型

### 4. 变量同名

B. 观察下列程序的运行结果，回答问题并将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 a=10;
short a;
void f1()
{
    int x=5;
    double x=1.2;

    short p=1, p=2;

}
int main()
{
    f1();
    return 0;
}
```

注：如果是error，贴error截图  
如果是warning，贴warning截图+运行结果  
如果正常，贴运行结果

#### 1、截图

```
(5, 7): error C2371: “a” : 重定义；不同的基类型
(4, 5): message : 参见“a”的声明
(9, 12): error C2371: “x” : 重定义；不同的基类型
(8, 9): message : 参见“x”的声明
(11, 18): error C2374: “p” : 重定义；多次初始化
(11, 11): message : 参见“p”的声明
```

2、结合4.A/4.B可以得知：不同级别的变量允许同名；  
相同级别的变量不允许同名；变量同名是的使用规则是  
“低层屏蔽高层”的规则。“低层屏蔽高层”的规则适用  
于全局变量与局部变量同名、局部变量和复合语句内的局  
部变量同名、在多层次嵌套的情况下不同层次的变量同名  
的情况。



## §. 基础知识题 - 变量类型

### 5. 自动变量与静态局部变量

A. 观察下列程序的运行结果，回答问题并将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;

void f1()
{
    int a=1;
    a++;
    cout << "a=" << a << ',' << &a << endl;

    static int b=1;
    b++;
    cout << "b=" << b << ',' << &b << endl;
}

int main()
{
    f1();
    f1();
    f1();
    return 0;
}
```

注：如果是error，贴error截图  
如果是warning，贴warning截图+运行结果  
如果正常，贴运行结果

#### 1、截图

```
Microsoft Visual Studio 调试控制台
a=2 004FFDF8
b=2 006CC008
a=2 004FFDF8
b=3 006CC008
a=2 004FFDF8
b=4 006CC008
```

2、结合a/b各自的地址和值，得到结论为：

自动变量a多次调用，则每次进行初始化，函数运行结束后会释放空间，下次进入时再次分配；

静态局部变量a多次调用，则仅第一次进行初始化，函数运行结束后不会释放空间，下次进入时继续使用上次的空间；

根据上面的分析结果，自动变量应该放在动态数据存储区，静态局部变量应该放在静态数据存储区。



## §. 基础知识题 – cin与cout的基本使用

### 1. cout的基本理解

A. 观察下列程序的运行结果，回答问题并将程序的运行结果截图贴上

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

int main()
{
    /* 第1组 */
    cout << "This is a C++ program." << endl;

    /* 第2组 */
    cout << "This is " << "a C++ " << "program." << endl;

    /* 第3组 */
    cout << "This is "
        << "a C++ "
        << "program."
        << endl;

    /* 第4组 */
    cout << "This is ";
    cout << "a C++ ";
    cout << "program.";
    cout << endl;

    return 0;
}
```

Microsoft Visual Studio 调试控制台  
This is a C++ program.  
This is a C++ program.  
This is a C++ program.  
This is a C++ program.

第3组和第4组在语句上的区别是：  
第3组是1个cout语句分4行，前三行无分号；  
第4组是4个cout语句，每行有分号。



# §. 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

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

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

Microsoft Visual Studio 调试控制台  
101520

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

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

Microsoft Visual Studio 调试控制台  
10

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

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

Microsoft Visual Studio 调试控制台  
20

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

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

error C2563: 在形参表中不匹配  
error C2568: “<<”：无法解析函数重载  
message : 可能是“std::basic\_ostream<\_Elem, \_Traits>::endl(std::basic\_ostream<\_Elem, \_Traits> &)”

解释这3个程序输出不同的原因：

程序1：<<运算符为左结合，优先级相同时先进行输出，所以依次输出a、b、c的值

程序2：<<运算符的优先级高于逗号运算符，先进行输出，所以输出a的值

程序3：()运算符的优先级高于<<运算符，先计算(a, b, c)，得到的值为c，所以输出c的值

结论：一个流插入运算符 << 只能输出1个数据。

解释错误原因：

一个流插入运算符 << 只能输出1个数据。另外，逗号运算符的优先级低于流插入运算符<<，所以后面相当于c<<endl，无法解析函数重载。



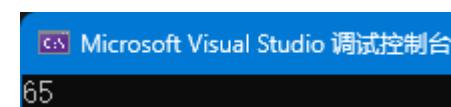
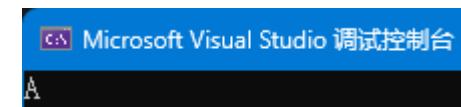
## §. 基础知识题 – cin与cout的基本使用

### 1. cout的基本理解

C. 观察下列2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

```
#include <iostream>
using namespace std;
int main()
{
    char ch = 65;
    cout << ch << endl;
    return 0;
}
```

```
#include <iostream>
using namespace std;
int main()
{
    int ch = 65;
    cout << ch << endl;
    return 0;
}
```



解释这两个程序输出不同的原因：

程序1：ch为字符型变量，65代表字符型变量ch的ASCII码，输出结果为ASCII码为65的字符，即输出结果为A

程序2：ch为整型变量，65代表整型变量ch的值，即输出结果为65



## §. 基础知识题 - cin与cout的基本使用

### 1. cout的基本理解

D. 程序同C，将修改后符合要求的程序及运行结果贴上

```
#include <iostream>
using namespace std;
int main()
{
    char ch = 65;
    cout << ch << endl;
    return 0;
}
```

```
My Project.cpp + x
My Project
1 #include <iostream>
2 using namespace std;
3 int main()
4 {
5     char ch = 65;
6     cout << int(ch) << endl;
7     return 0;
8 }
```

Microsoft Visual Studio 调试控制台

65

在char类型不变的情况下，要求输出为65  
(不允许添加其它变量)

```
#include <iostream>
using namespace std;
int main()
{
    int ch = 65;
    cout << ch << endl;
    return 0;
}
```

```
My Project.cpp + x
My Project
1 #include <iostream>
2 using namespace std;
3 int main()
4 {
5     int ch = 65;
6     cout << char(ch) << endl;
7     return 0;
8 }
```

Microsoft Visual Studio 调试控制台

65

在int类型不变的情况下，要求输出为A  
(不允许添加其它变量)

# §. 基础知识题 - cin与cout的基本使用



## 1. cout的基本理解

E. 程序同C，将修改后符合要求的程序及运行结果贴上

```
#include <iostream>
using namespace std;
int main()
{
    char ch = 65;
    cout << ch << endl;
    return 0;
}
```

在char类型不变的情况下，要求输出为65  
(不允许添加其它变量，  
不允许使用任何方式的强制类型转换)

The screenshot shows the Microsoft Visual Studio interface. The code editor window is titled "My Project.cpp" and contains the following C++ code:

```
1 #include <iostream>
2 using namespace std;
3 int main()
4 {
5     char ch = 65;
6     cout << ch + 0 << endl;
7     return 0;
8 }
```

The output window at the bottom is titled "Microsoft Visual Studio 调试控制台" and displays the result: "65".



# §. 基础知识题 - cin与cout的基本使用

## 2. cin的基本理解 - 单数据情况

A. 运行下面的程序，观察不同输入下的运行结果

```
#include <iostream>
using namespace std;
int main()
{
    short k;
    cin >> k;
    cout << cin.good();
    cout << " k=" << k << endl;

    return 0;
}
```

基础知识：

short的最小值是：-32768

short的最大值是：32767

- 1、输入：123↙
- 2、输入：123 456↙ (一个空格)
- 3、输入：123 456↙ (多个空格)
- 4、输入：123m↙
- 5、输入：m↙
- 6、输入： 123↙ (持续多个空格后，再输入123，按回车)
- 7、输入： ↵ (持续多个空格后，按回车)  
 123↙ (再输入123，按回车)
- 8、输入： ↵  
...  
 ↵  
 123↙ (持续多个空回车后，输入123)

分析结果：

- 1、在前面有正确输入的情况下，回车、空格、(对int型而言是非法的字符)m的作用是？  
中止输入
- 2、直接输入若干空格和回车后，再输入正确，变量是否能得到正确的值？  
是
- 3、直接输入(对int型而言是)非法的数据m，输出是？  
0

The screenshot shows a vertical stack of Microsoft Visual Studio Debug Console windows, each displaying a different input scenario and its output. The inputs include single digits, two-digit numbers, three-digit numbers, and even a four-digit number. The outputs show the digit(s) followed by the variable 'k' and its value (123). The console windows have a blue header bar with the Microsoft Visual Studio logo and the text 'Microsoft Visual Studio 调试控制台'.



# §. 基础知识题 - cin与cout的基本使用

## 2. cin的基本理解 - 单数据情况

B. 运行下面的程序，观察不同输入下的运行结果

```
#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;
}
```

结论：

多个输入中，编号4、5、6 输入的k值是不可信的

1、输入： 123↙ (正确+回车)

2、输入： 123↙456↙ (正确+空格)

3、输入： -123m↙ (正确+非法字符)

4、输入： m↙ (直接非法字符)

5、输入： 54321↙ (超上限)

6、输入： -40000↙ (超下限)

VS与Dev编译结果一致

The screenshot shows six separate instances of the Microsoft Visual Studio Debug Console (调试控制台). Each instance displays the input value followed by the variable 'k' and the results of the 'cin.good()' and 'cin.fail()' functions.

- Input: 123, k=123, cin.good()=1, cin.fail()=0
- Input: 123 456, k=123, cin.good()=1, cin.fail()=0
- Input: -123m, k=-123, cin.good()=1, cin.fail()=0
- Input: m, k=0, cin.good()=0, cin.fail()=1
- Input: 54321, k=32767, cin.good()=0, cin.fail()=1
- Input: -40000, k=-32768, cin.good()=0, cin.fail()=1



## §. 基础知识题 - cin与cout的基本使用

### 2. cin的基本理解 - 单数据情况

B-Compare. 运行下面的**对比**程序（cin输入与赋值），观察运行结果并与B的输出结果进行对比分析

```
#include <iostream>
using namespace std;
int main()
{
    short k1, k2, k3, k4, k5;

    k1 = 12345;
    k2 = 54321;
    k3 = 70000;
    k4 = -12345;
    k5 = -54321;

    cout << k1 << endl;
    cout << k2 << endl;
    cout << k3 << endl;
    cout << k4 << endl;
    cout << k5 << endl;

    return 0;
}
```

B的输入:

- 1、输入: 12345↙ (合理范围)  
对应本例的k1=
- 2、输入: 54321↙ (超上限但未超同类型的u\_short上限)  
对应本例的k2=
- 3、输入: 70000↙ (超上限且超过同类型的u\_short上限)  
对应本例的k3=
- 4、输入: -12345↙ (合理范围)  
对应本例的k4=
- 5、输入: -54321↙ (超下限)  
对应本例的k5=

```
Microsoft Visual Studio 调试控制台
12345
-11215
4464
-12345
11215
```

对比分析: 输入的值在合理范围内时，输出与输入一致。输入的值超出short型数据的上下限时，输出与输入不一致，输出的值是short型数据的上下限，赋值时得到的结果是原数据的二进制补码进行高位丢弃和低位赋值操作之后对应的十进制数。



## §. 基础知识题 - cin与cout的基本使用

### 2. cin的基本理解 - 单数据情况

C. 仿B, 自行构造不同测试数据, 观察不同输入下的运行结果

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

int main()
{
    int k;
    cin >> k;
    cout << "k=" << k << endl;
    cout << "cin.good()=" << cin.good() << endl;
    cout << "cin.fail()=" << cin.fail() << endl;
    return 0;
}
```

结论:

多个输入中, 编号2、3、5 输入的k值是不可信的

1、输入: 2000000000↙  
(合理范围)

2、输入: 4000000000↙  
(超上限但未超同类型的u\_int上限)

3、输入: 9000000000↙  
(超上限且超过同类型的u\_int上限)

4、输入: -2000000000↙  
(合理范围)

5、输入: -4000000000↙  
(超下限)

Microsoft Visual Studio 调试控制台  
2000000000  
k=2000000000  
cin.good()=1  
cin.fail()=0

Microsoft Visual Studio 调试控制台  
4000000000  
k=2147483647  
cin.good()=0  
cin.fail()=1

Microsoft Visual Studio 调试控制台  
9000000000  
k=2147483647  
cin.good()=0  
cin.fail()=1

Microsoft Visual Studio 调试控制台  
-2000000000  
k=-2000000000  
cin.good()=1  
cin.fail()=0

Microsoft Visual Studio 调试控制台  
-4000000000  
k=-2147483648  
cin.good()=0  
cin.fail()=1

VS与Dev编译结果一致



## §. 基础知识题 – cin与cout的基本使用

### 2. cin的基本理解 – 单数据情况

C-Compare. 仿B-Compare, 构造**对比程序** (cin输入与赋值, int型), 观察运行结果并与C的输出结果进行对比分析

需要回答下列问题:

1、输入/赋值超int上限但未超同类型的u\_int上限, 两者是否一致? 如果有区别, 区别是?

不一致。

输入超int上限但未超同类型的u\_int上限, 得到的结果是int型变量的上限2147483647;

赋值超int上限但未超同类型的u\_int上限, 按照同长度的signed与unsigned相互赋值的规制进行赋值, 得到的结果的二进制补码是原数据的二进制形式。

2、输入/赋值超int上限且超同类型的u\_int上限, 两者是否一致? 如果有区别, 区别是?

不一致。

输入超int上限且超同类型的u\_int上限, 得到的结果是int型变量的上限2147483647;

赋值超int上限且超同类型的u\_int上限, 按照不同长度的整型数据相互赋值的规则进行赋值, 得到的结果是原数据的二进制补码进行高位丢弃和低位赋值操作之后对应的十进制数。

3、输入/赋值超int下限, 两者是否一致? 如果有区别, 区别是?

不一致。

输入超int下限, 得到的结果是int型变量的下限-2147483648;

赋值超int下限, 得到的结果是原数据的二进制补码进行高位丢弃操作和低位赋值之后对应的十进制数。



## §. 基础知识题 – cin与cout的基本使用

### 2. cin的基本理解 – 单数据情况

D. 运行下面的程序，观察不同输入下的运行结果

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

int main()
{
    unsigned short k;
    cin >> k;
    cout << "k=" << k;
    cout << " good=" << cin.good();
    cout << " fail=" << cin.fail() << endl;
    return 0;
}
```

结论：

多个输入中，编号2、6 输入的k值是不可信的

VS与Dev编译结果一致

- 1、输入：12345↙  
(合理范围)
- 2、输入：70000↙  
(超上限)
- 3、输入：-12345↙  
(负数但未超过short下限)
- 4、输入：-1↙  
(负数且未超过short下限)
- 5、输入：-65535↙  
(负数且未超过u\_short上限加负号后的下限)
- 6、输入：-65536↙  
(负数且超过u\_short上限加负号后的下限)

```
Microsoft Visual Studio 调试控制台
12345
k=12345 good=1 fail=0

Microsoft Visual Studio 调试控制台
70000
k=65535 good=0 fail=1

Microsoft Visual Studio 调试控制台
-12345
k=53191 good=1 fail=0

Microsoft Visual Studio 调试控制台
-1
k=65535 good=1 fail=0

Microsoft Visual Studio 调试控制台
-65535
k=1 good=1 fail=0

Microsoft Visual Studio 调试控制台
-65536
k=65535 good=0 fail=1
```



# §. 基础知识题 – cin与cout的基本使用

## 2. cin的基本理解 – 单数据情况

D-Compare. 仿B-Compare构造的**对比程序** (cin输入与赋值, u\_short型) , 观察运行结果并与D的输出结果进行对比分析

```
#include <iostream>
using namespace std;
int main()
{
    unsigned short k1, k2, k3, k4, k5, k6;

    k1 = 12345;
    k2 = 70000;
    k3 = -12345;
    k4 = -1;
    k5 = -65535;
    k6 = -65536;

    cout << k1 << endl;
    cout << k2 << endl;
    cout << k3 << endl;
    cout << k4 << endl;
    cout << k5 << endl;
    cout << k6 << endl;
    return 0;
}
```

- 1、输入: 12345↙ (合理范围)  
对应本例的k1=
- 2、输入: 70000↙ (超上限)  
对应本例的k2=
- 3、输入: -12345↙ (负数但未超过short下限)  
对应本例的k3=
- 4、输入: -1↙ (负数且未超过short下限)  
对应本例的k4=
- 5、输入: -65535↙ (负数且未超过u\_short上限加负号后的下限)  
对应本例的k5=
- 6、输入: -65536↙ (负数且超过u\_short上限加负号后的下限)  
对应本例的k6=

```
Microsoft Visual Studio 调试控制台
12345
4464
53191
65535
1
0 warning C4305: “=” : 从“int”到“unsigned short”截断常量值
warning C4309: “=” : 截断常量值
warning C4309: “=” : 截断常量值
```

```
D:\Learning\高级语言程序设计\My Project\My Project\My Project.exe
12345
4464
53191
65535
[Warning] unsigned conversion from 'int' to 'short unsigned int' changes value from '70000' to '4464' [-Woverflow]
[Warning] unsigned conversion from 'int' to 'short unsigned int' changes value from '-65535' to '1' [-Woverflow]
[Warning] unsigned conversion from 'int' to 'short unsigned int' changes value from '-65536' to '0' [-Woverflow]
```



## §. 基础知识题 – cin与cout的基本使用

### 2. cin的基本理解 – 单数据情况

E. 仿D, 自行构造不同测试数据, 观察不同输入下的运行结果

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

int main()
{
    unsigned int k;
    cin >> k;
    cout << "k=" << k;
    cout << " good()=" << cin.good();
    cout << " fail()=" << cin.fail() << endl;
    return 0;
}
```

结论:

多个输入中, 编号2、5 输入的k值是不可信的

- 1、输入: 4000000000 ✓  
(合理范围)
- 2、输入: 5000000000 ✓  
(超上限)
- 3、输入: -2000000000 ✓  
(负数但未超int下限)
- 4、输入: -4000000000 ✓  
(负数且未超过u\_int  
上限加负号后的下限)
- 5、输入: -5000000000 ✓  
(负数且超过u\_int上限加负号后的下限)

```
4000000000
k=4000000000 good()=1 fail()=0

5000000000
k=4294967295 good()=0 fail()=1

-2000000000
k=2294967296 good()=1 fail()=0

-4000000000
k=294967296 good()=1 fail()=0

-5000000000
k=4294967295 good()=0 fail()=1
```

VS与Dev编译结果一致



## §. 基础知识题 – cin与cout的基本使用

### 2. cin的基本理解 – 单数据情况

E-Compare. 仿B-Compare, 构造**对比程序** (cin输入与赋值, u\_int型), 观察运行结果并与E的输出结果进行对比分析

需要回答下列问题:

1、输入/赋值超u\_int上限, 两者是否一致? 如果有区别, 区别是?

不一致。

输入超u\_int上限, 得到的结果是u\_int型变量的上限4294967295;

赋值超u\_int上限, 按照不同长度的整型数据相互赋值的规则进行赋值, 得到的结果是原数据的二进制补码进行高位丢弃和低位赋值操作之后对应的十进制数。

2、输入/赋值为负数但未超int下限, 两者是否一致? 如果有区别, 区别是?

一致。

3、输入/赋值为负数且未超过u\_int上限加负号后的下限, 两者是否一致? 如果有区别, 区别是?

一致。

4、输入/赋值为负数且超过u\_int上限加负号后的下限? 如果有区别, 区别是?

不一致。

输入超int下限, 得到的结果是u\_int型变量的上限4294967295;

赋值超int下限, 得到的结果是原数据的二进制补码进行高位丢弃和低位赋值操作之后对应的十进制数。



# §. 基础知识题 - cin与cout的基本使用

## 2. cin的基本理解 - 单数据情况

### B-E. 总结

- 1、signed数据在输入正确且范围合理的情况下  
cin输入与赋值的输出一致。
- 2、signed数据在输入正确但超上限（未超同类型unsigned上限）的情况下  
cin输入与赋值的输出不一致，cin输入得到的结果是该signed数据取值的上限，赋值得到的结果是原数据的二进制补码进行高位丢弃和低位赋值操作之后对应的十进制数。
- 3、signed数据在输入正确且超上限（超过同类型unsigned上限）的情况下  
cin输入与赋值的输出不一致，cin输入得到的结果是该signed数据取值的上限，赋值得到的结果是原数据的二进制补码进行高位丢弃和低位赋值操作之后对应的十进制数。
- 4、signed数据在输入正确但超下限范围的情况下  
cin输入与赋值的输出不一致，cin输入得到的结果是该signed数据取值的下限，赋值得到的结果是原数据的二进制补码进行高位丢弃和低位赋值操作之后对应的十进制数。
- 5、unsigned数据在输入正确且范围合理的情况下  
cin输入与赋值的输出一致。
- 6、unsigned数据在输入正确且超上限的情况下  
cin输入与赋值的输出不一致，cin输入得到的结果是该unsigned数据取值的上限，赋值得到的结果是原数据的二进制补码进行高位丢弃和低位赋值操作之后对应的十进制数。
- 7、unsigned数据在输入正确但为负数（未超同类型signed下限）的情况下  
cin输入与赋值的输出一致。
- 8、unsigned数据在输入正确且为负数（超过同类型signed下限）的情况下  
cin输入与赋值的输出一致。
- 9、unsigned数据在输入正确且为负数（超过同类型unsigned上限加负号后的下限）的情况下  
cin输入与赋值的输出不一致，cin输入得到的结果是该unsigned数据取值的上限，赋值得到的结果是原数据的二进制补码进行高位丢弃和低位赋值操作之后对应的十进制数。

**对比：cin输入与变量赋值，在输入/右值超范围的情况下，表现是否相同？总结规律**

表现不相同。cin输入在输入超范围的情况下，得到的结果是该数据类型取值的上限或下限；变量赋值在右值超范围的情况下，得到的结果是原数据的二进制补码进行高位丢弃和低位赋值操作之后对应的十进制数。

**cin输入与变量赋值，在输入/右值合理范围的情况下，表现是否相同？总结规律**

表现相同。



## §. 基础知识题 – cin与cout的基本使用

### 2. cin的基本理解 – 单数据情况

F. 运行下面的程序，观察不同输入下的运行结果

```
#include <iostream>
using namespace std;
int main()
{
    char ch;
    cin >> ch;

    cout << "ch=" << int(ch) << endl;
    cout << "ch=" << ch << endl;

    return 0;
}
```

- 1、键盘输入A (单个图形字符)
- 2、键盘输入\b (退格键的转义符)
- 3、键盘输入\101 (A的ASCII码的8进制转义表示)
- 4、键盘输入\x41 (A的ASCII码的16进制转义表示)
- 5、键盘输入65 (A的ASCII码的十进制整数形式表示)
- 6、键盘输入CtrlL+C (注意：是Ctrl+C组合键，注意不要有输入法栏)
- 7、键盘输入CtrlL+z (注意：是Ctrl+z组合键，注意不要有输入法栏)

The screenshot shows seven consecutive runs of the program in the Microsoft Visual Studio Debug Console. Each run displays the character input and its corresponding ASCII value and octal/hex representation.

- Run 1: Input 'A', Output: ch=65, ch=A
- Run 2: Input '\b', Output: ch=92, ch=\b
- Run 3: Input '\101', Output: ch=92, ch=\101
- Run 4: Input '\x41', Output: ch=92, ch=\x41
- Run 5: Input '65', Output: ch=54, ch=65
- Run 6: Input Ctrl+C (simulated by ^C), Output: ch=-52
- Run 7: Input Ctrl+Z (simulated by ^Z), Output: ch=-52, ch=

# §. 基础知识题 - cin与cout的基本使用

## 2. cin的基本理解 - 单数据情况

### G. 运行下面的程序，观察不同输入下的运行结果

```
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
    float f;
    cin >> f;

    cout << f << endl;
    cout << setprecision(20) << f << endl;
}
```

//注: setprecision(20)表示输出时保留小数点后20位  
// (已超float和double的有效位数)

- 1、键盘输入123.456 (合理范围正数, 小数形式)
- 2、键盘输入1.23456e2 (合理范围正数, 指数形式)
- 3、键盘输入-123.456 (合理范围负数, 小数形式)
- 4、键盘输入-1.23456e2 (合理范围负数, 指数形式)
- 5、键盘输入123.456789 (合理范围, 但超有效位数)
- 6、键盘输入6.7e38 (尾数超上限但数量级未超, 仍是10<sup>38</sup>)
- 7、键盘输入1.7e39 (超上限且数量级已超10<sup>38</sup>)
- 8、键盘输入-2.3e39 (超上限且数量级已超10<sup>38</sup>)
- 9、键盘输入1.23e-30 (合理范围整数但指数很小)
- 10、键盘输入-1.23e-30 (合理范围负数但指数很小)

123.456  
123.456  
123.45600128173828125

1.23456e2  
123.456  
123.45600128173828125

-123.456  
-123.456  
-123.45600128173828125

-1.23456e2  
-123.456  
-123.45600128173828125

123.456789  
123.457  
123.456787109375

6.7e38  
0  
0

1.7e39  
0  
0

-2.3e39  
0  
0

1.23e-30  
1.23e-30  
1.229999549998595325e-30

-1.23e-30  
-1.23e-30  
-1.229999549998595325e-30



## §. 基础知识题 – cin与cout的基本使用

### 3. cin的基本理解 – 多个同类型数据的情况

A. 观察下列3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

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

int main()
{
    int a, b, c, d;
    cin >> a >> b >> c >> d;

    cout << a << endl;
    cout << b << endl;
    cout << c << endl;
    cout << d << endl;

    return 0;
}
```

```
1 2 3 4
```

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

int main()
{
    int a, b, c, d;
    cin >> a
        >> b
        >> c
        >> d;
    cout << a << endl;
    cout << b << endl;
    cout << c << endl;
    cout << d << endl;
}
```

```
1 2 3 4
```

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

int main()
{
    int a, b, c, d;
    cin >> a;
    cin >> b;
    cin >> c;
    cin >> d;
    cout << a << endl;
    cout << b << endl;
    cout << c << endl;
    cout << d << endl;
}
```

```
1 2 3 4
```

1、程序运行后，输入：1 2 3 4↙，观察输出结果

2、解释第2个和第3个程序的cin语句的使用区别：

第2个程序是1个cin语句分4行，前三行无分号；第3个程序是4个cin语句，每行有分号。



## §. 基础知识题 – cin与cout的基本使用

3. cin的基本理解 – 多个同类型数据的情况

B. 程序同A，观察不同输入下的运行结果

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

int main()
{
    int a, b, c, d;
    cin >> a >> b >> c >> d;

    cout << a << endl;
    cout << b << endl;
    cout << c << endl;
    cout << d << endl;

    return 0;
}
```

1、输入: 1 2 3 4↙

2、输入: 1 2 3 4↙ (每个数字间多于一个空格)

3、输入: 1↙  
2↙  
3↙  
4↙ (每个数字后立即加回车)

4、输入: 1↙  
    ↙  
    2↙  
    ↙  
    3↙  
    ↙  
    4↙ (每个数字后立即加回车 + 多个空回车)

结论: 在输入正确的情况下, 回车和空格的作用?  
中止输入

Microsoft Visual Studio 调试控制台

```
1 2 3 4  
1  
2  
3  
4
```

Microsoft Visual Studio 调试控制台

```
1 2 3 4  
1  
1  
2  
3  
4
```

Microsoft Visual Studio 调试控制台

```
1  
2  
3  
4  
1  
2  
3  
4  
1  
2  
3  
4
```

Microsoft Visual Studio 调试控制台

```
1  
2  
3  
4  
1  
2  
3  
4
```



## §. 基础知识题 - cin与cout的基本使用

3. cin的基本理解 - 多个同类型数据的情况

C. 程序同A，观察不同输入下的运行结果

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

int main()
{
    int a, b, c, d;
    cin >> a >> b >> c >> d;

    cout << a << endl;
    cout << b << endl;
    cout << c << endl;
    cout << d << endl;

    return 0;
}
```

1、输入: 1 2 3 4↙

2、输入: 1 2 3m 4↙

3、输入: 1 2m 3 4↙

4、输入: 1m 2 3 4↙

5、输入: 1 2 3 m↙

6、输入: 1 2 m 4↙

7、输入: 1 m 3 4↙

8、输入: m 2 3 4↙

**总结:** 多个cin输入时，错误输入出现在不同位置对输入正确性的影响

**要求:** 综合观察运行结果，加上自己的思考，给出总结性的结论，这个结论要能对多个输入情况下不同位置的错误情况有普遍适应性，而不仅仅是简单的根据结论说错在1/2/3/4位置  
当错误输入出现时，cin语句会中止输入，这时错误输入位置之前已输入的值是可信的，错误输入位置之后的值是不可信的。

The screenshot shows five separate Microsoft Visual Studio debug windows, each displaying a different input scenario for the program. The inputs are:  
1、输入: 1 2 3 4↙  
2、输入: 1 2 3m 4↙  
3、输入: 1 2m 3 4↙  
4、输入: 1m 2 3 4↙  
5、输入: 1 2 3 m↙  
In each window, the first few integers (1, 2, 3) are correctly read, followed by '-858993460' which is the default value for an integer in Visual Studio's debugger output.

```
XYZ
a=88
b=89
c=90
```

```
X YZ
a=88
b=89
c=90
```

```
a=-5
```

```
Xa=-52
```

```
XYa=-52
```

```
XYZa=-52
```

```
^Z
a=-52
b=-52
c=-52
```

```
^ZXZY
a=-52
b=-52
c=-52
```

# §. 基础知识题 - cin与cout的基本使用

## 3. cin的基本理解 - 多个同类型数据的情况

### D. 观察不同输入下的运行结果

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

int main()
{
    char a, b, c;
    cin >> a >> b >> c;

    cout << "a=" << int(a) << endl;
    cout << "b=" << int(b) << endl;
    cout << "c=" << int(c) << endl;

    return 0;
}
```

1、输入: XYZ↙

2、输入: X YZ↙

3、输入: Ctrl+C↙

4、输入: XCtrl+C↙

5、输入: XYCtrl+C↙

6、输入: XYZCtrl+C↙

7、输入: Ctrl+z↙

(若未出结果则继续输入, 可以按回车后多行输入, 打印后观察结果)

8、输入: Ctrl+zXYZ↙

(若未出结果则继续输入, 可以按回车后多行输入, 打印后观察结果)

总结: 多个cin输入时char型数据时

1、能否输入空格  
能

2、Ctrl+C在输入中表示什么? (可自行查阅资料, 若资料与表现不符, 信哪个?)  
Ctrl+C在输入中表示强制中断程序的执行, 进程被完全关闭。(信表现)

3、Ctrl+z在输入中表示什么? (可自行查阅资料, 若资料与表现不符, 信哪个?)  
Ctrl+z在输入中表示中断任务, 挂起一个进程, 进程未被关闭, 仍然存在。(信表现)

4、Ctrl+z后不按回车而继续输入的其它字符, 能否被读入?  
不能

```
1. 7e39 123.456 1.23456e2
a=0
0
b=-107374176
-107374176
c=-107374176
-107374176
```

```
-4e38 123.456 1.23456e2
a=0
0
b=-107374176
-107374176
c=-107374176
-107374176
```

```
123.456 1.7e39 1.23456e2
a=123.456
123.45600128173828125
b=0
0
c=-107374176
-107374176
```

```
123.456 -4e38 1.23456e2
a=123.456
123.45600128173828125
b=0
0
c=-107374176
-107374176
```

```
123.456 1.23456e2 1.7e39
a=123.456
123.45600128173828125
b=123.45600128173828125
123.45600128173828125
c=0
0
```

```
123.456 1.23456e2 -4e38
a=123.456
123.45600128173828125
b=123.45600128173828125
123.45600128173828125
c=0
0
```

# §. 基础知识题 – cin与cout的基本使用

## 3. cin的基本理解 – 多个同类型数据的情况

### E. 自行构造测试数据, 观察不同输入下的运行结果

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

int main()
{
    float a, b, c;
    cin >> a >> b >> c;

    cout << "a=" << a << endl;
    cout << setprecision(20) << a << endl;

    cout << "b=" << b << endl;
    cout << setprecision(20) << b << endl;

    cout << "c=" << c << endl;
    cout << setprecision(20) << c << endl;

    return 0;
}
```

1、输入: 1. 7e39 123.456 1.23456e2 ↵ (第1个超上限, 2/3正常)

2、输入: -4e38 123.456 1.23456e2 ↵ (第1个超下限, 2/3正常)

3、输入: 123.456 1.7e39 1.23456e2 ↵ (1/3正常, 第2个超上限)

4、输入: 123.456 -4e38 1.23456e2 ↵ (1/3正常, 第2个超下限)

5、输入: 123.456 1.23456e2 1.7e39 ↵ (1/2正常, 第3个超上限)

6、输入: 123.456 1.23456e2 -4e38 ↵ (1/2正常, 第3个超下限)

总结:

1、多个cin输入时, 错误输入出现在不同位置对输入正确性的影响

**要求: 综合观察运行结果, 加上自己的思考, 给出总结性的结论, 这个结论要能对多个输入情况下不同位置的错误情况有普遍适应性, 而不仅仅是简单的根据结论说错在1/2/3位置**

错误输入位置之前已输入的值是可信的, 错误输入位置之后的值是不可信的。

2、将float替换为double, 上述结论是否仍然成立?  
仍然成立



## §. 基础知识题 – cin与cout的基本使用

### 4. cin的基本理解 – 其他情况

A. 程序如下，观察编译及运行结果

```
#include <iostream>
using namespace std;
int main()
{
    int a, b, c;
    cin >> a, b, c;

    cout << a << endl;
    cout << b << endl;
    cout << c << endl;
    return 0;
}
```

1、如果编译有error或warning，则贴相应信息的截图

```
error C4700: 使用了未初始化的局部变量“b”
error C4700: 使用了未初始化的局部变量“c”
```

2、如果能运行(包括有warning)，则输入三个正确的int型数据  
(例：1 2 3↙)，观察输出

```
D:\Learning\高级语言程序设计\My Project\My Project\My Project.exe
1 2 3
1
4242432
7929704
```

3、分析为什么只有某个变量的结果是正确的

一个流提取运算符>>只能输入1个数据，因此只输入了a的值，未输入b和c的值，所以会出现error C4700: 使用了未初始化的局部变量“b” 和 error C4700: 使用了未初始化的局部变量“c”。

VS与Dev编译结果不一致  
VS有error，不能运行；Dev无error，能运行



## §. 基础知识题 – cin与cout的基本使用

### 4. cin的基本理解 – 其他情况

B. 程序如下，观察编译及运行结果

```
#include <iostream>
using namespace std;
int main()
{
    int a=66, b=67, c=68;
    cin >> a, b, c;

    cout << a << endl;
    cout << b << endl;
    cout << c << endl;
    return 0;
}
```

1、运行后，输入三个正确的int型数据(例：1 2 3↙，注意不要是预置值)，观察输出

```
Microsoft Visual Studio 调试控制台
1 2 3
1
67
68
```

2、通过观察三个变量的输出，你得到了什么结论？

a, b, c均被初始化赋了初值，一个流提取运算符>>只能输入1个数据，因此只输入了a的值，未输入b和c的值，所以输出a的值被赋了新值，输出b和c的值仍为初值。



# §. 基础知识题 - cin与cout的基本使用

## 4. cin的基本理解 - 其他情况

C. 程序如下，观察编译及运行结果

```
#include <iostream>
using namespace std;
int main()
{
    int a;
    cin >> 5;
    cin >> a+10;

    cout << a << endl;
    return 0;
}
```

```
error C2678: 二进制“>>”：没有找到接受“std::istream”类型的左操作数的运算符(或没有可接受的转换)
message : 可能是“std::basic_istream<char, std::char_traits<char>> &std::basic_istream<char, std::char_traits<char>>::operator >>(std::basic_streambuf<char, std::char_traits<char>> *)”
message : 或   “std::basic_istream<char, std::char_traits<char>> &std::basic_istream<char, std::char_traits<char>>::operator >>(void *& )”
message : 或   “std::basic_istream<char, std::char_traits<char>> &std::basic_istream<char, std::char_traits<char>>::operator >>(long double & )”
message : 或   “std::basic_istream<char, std::char_traits<char>> &std::basic_istream<char, std::char_traits<char>>::operator >>(double & )”

[Error] no match for 'operator>>' (operand types are 'std::istream' [aka 'std::basic_istream<char>'] and 'int')
In file included from C:/Program Files (x86)/Dev-Cpp/MinGW64/lib/gcc/x86_64-w64-mingw32/9.2.0/include/c++/iostream
      from D:\Learning\高级语言程序设计\My Project\My Project\My Project.cpp
[Note] candidate: 'std::basic_istream<_CharT, _Traits>::__istream_type& std::basic_istream<_CharT, _Traits>::operator >>(__istream_type& (*)(std::basic_istream<_CharT, _Traits>::__istream_type&)) [with _CharT = char; _Trait
```

1、如果编译有error或warning，则贴相应信息的截图(信息太多则前五行)

2、分析为什么编译有错

“`cin >> 5;`”：

`cin`语句无法给5赋值，因为5是常量，常量不能被赋值；

“`cin >> a+10;`”：

`cin`语句无法给`a+10`赋值，因为`a+10`是表达式，表达式不能被赋值。

3、结论：流提取运算符后面必须跟变量，不能是常量或表达式。



## §. 基础知识题 – cin与cout的基本使用

### 4. cin的基本理解 – 其他情况

D. 程序如下，观察编译及运行结果

```
#include <iostream>
using namespace std;
int main()
{
    int a=66, b=67, c=68;
    cin >> (a, b, c);

    cout << a << endl;
    cout << b << endl;
    cout << c << endl;
    return 0;
}
```

1、运行后，输入三个正确的int型数据(例：1 2 3↙，注意不要是预置值)，观察输出

```
Microsoft Visual Studio 调试控制台
1 2 3
66
67
1
```

2、通过观察三个变量的输出，你得到了什么结论？  
流提取运算符>>的优先级低于后续运算符时，会先进行后续的计算，之后再对计算出的结果进行输入。

3、和B进行比较，分析为什么结果有差异？  
因为流提取运算符>>的优先级低于()运算符，故先计算(a, b, c)的值为c，这时为cin >> c，可以输入c的值，而B是输入a的值，所以结果有差异。

4、和C进行比较，与C得出的结论矛盾吗？  
不矛盾，因为(a, b, c)的值为c，c是变量，满足“流提取运算符后面必须跟变量”的条件。



## §. 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

```
#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:

```
Microsoft Visual Studio 调试控制台
1234 56.78
1 2 34 56.78
```

2、输入: 1↙2↙34↙56.78↙

输出:

```
Microsoft Visual Studio 调试控制台
1 2 34 56.78
1 2 34 56.78
```

3、分析在以上两种不同输入的情况下，为什么输出相同。

cin是按格式读入，到空格、回车、非法为止，系统会自动根据in后变量的类型按最长原则来读取合理数据。

在输入1234↙56.78时，按照最长原则，首先读取1赋值给char型数据c1，读取2赋值给char型数据c2，之后读取34（因为空格输入中止）赋值给int型数据a，最后读取56.78赋值给float型数据b；

在输入1↙2↙34↙56.78时，读取到空格输入中止，1、2、34、56.78分别给赋值给char型数据c1、char型数据c2、int型数据a、float型数据b。

所以在以上两种不同输入的情况下输出相同，最后的输出为1↙2↙34↙56.78。



# §. 基础知识题 – cin与cout的基本使用

## 4. cin的基本理解 – 其他情况

F. 程序如下，观察编译及运行结果

```
#include <iostream>
using namespace std;
int main()
{
    int a;
    cin >> a >> endl;

    return 0;
}
```

1、如果编译有error或warning，则贴相应信息的截图(信息太多则前五行)

```
error C2679: 二元“>>”：没有找到接受“overloaded-function”类型的右操作数的运算符(或没有可接受的转换)
message : 可能是“std::basic_istream<char, std::char_traits<char>> &std::basic_istream<char, std::char_traits<char>>::operator >>(std::basic_streambuf<char, std::char_traits<char>> *)”
message : 或 “std::basic_istream<char, std::char_traits<char>> &std::basic_istream<char, std::char_traits<char>>::operator >>(void *)”
message : 或 “std::basic_istream<char, std::char_traits<char>> &std::basic_istream<char, std::char_traits<char>>::operator >>(long double *)”
message : 或 “std::basic_istream<char, std::char_traits<char>> &std::basic_istream<char, std::char_traits<char>>::operator >>(double *)”

[Error] no match for 'operator>>' (operand types are 'std::basic_istream<char>::__istream_type' {aka 'std::basic_istream<char>'} and '<unresolved overloaded function type>')
In file included from C:/Program Files (x86)/Dev-Cpp/MinGW64/lib/gcc/x86_64-w64-mingw32/9.2.0/include/c++/iostream
    from D:\Learning\高级语言程序设计\My Project\My Project.cpp
[Note] candidate: 'std::basic_istream<_CharT, _Traits>::__istream_type& std::basic_istream<_CharT, _Traits>::operator >>(std::basic_istream<_CharT, _Traits>::__istream_type& (*) (std::basic_istream<_CharT, _Traits>::__istream_type&)) [with _CharT = char, _Trait
[Note] no known conversion for argument 1 from '<unresolved overloaded function type>' to 'std::basic_istream<char>::__istream_type& (*) (std::basic_istream<char>::__istream_type&)' {aka 'std::basic_istream<char> & (*) (std::basic_istream<char> &)')}
```

2、结论：在cin中不能跟endl



# §. 基础知识题 – C++方式输入输出的格式化控制

## 1. 在cout中使用格式化控制符

### A. 进制前导符的使用：回答问题并将程序的运行结果截图贴上

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

int main()
{
    short a1 = 1234, a2 = 0x1234, a3 = 01234, a4 = 0b1101001; //常量为各进制表示正数
    cout << "dec:" << dec << a1 << ',' << a2 << ',' << a3 << ',' << a4 << endl;
    cout << "hex:" << hex << a1 << ',' << a2 << ',' << a3 << ',' << a4 << endl;
    cout << "oct:" << oct << a1 << ',' << a2 << ',' << a3 << ',' << a4 << endl;
    cout << endl;

    short b1 = -1234, b2 = -0x1234, b3 = -01234, b4 = -0b1101001; //常量为各进制表示负数
    cout << "dec:" << dec << b1 << ',' << b2 << ',' << b3 << ',' << b4 << endl;
    cout << "hex:" << hex << b1 << ',' << b2 << ',' << b3 << ',' << b4 << endl;
    cout << "oct:" << oct << b1 << ',' << b2 << ',' << b3 << ',' << b4 << endl;
    cout << endl;

    short c1 = 40000, c2 = 0x9876, c3 = 0171234, c4 = 0b1101010100111100; //赋值后最高位均为1，有warning
    cout << "dec:" << dec << c1 << ',' << c2 << ',' << c3 << ',' << c4 << endl;
    cout << "hex:" << hex << c1 << ',' << c2 << ',' << c3 << ',' << c4 << endl;
    cout << "oct:" << oct << c1 << ',' << c2 << ',' << c3 << ',' << c4 << endl;
    cout << endl;

    return 0;
}
```

```
Microsoft Visual Studio 调试控制台
dec:1234 4660 668 105
hex:4d2 1234 29c 69
oct:2322 11064 1234 151

dec:-1234 -4660 -668 -105
hex:fb2e edcc fd64 ff97
oct:175456 166714 176544 177627

dec:-25536 -26506 -3428 -10948
hex:9c40 9876 f29c d53c
oct:116100 114166 171234 152474

warning C4309: “初始化”：截断常量值
warning C4309: “初始化”：截断常量值
warning C4309: “初始化”：截断常量值
warning C4309: “初始化”：截断常量值
```



## §. 基础知识题 – C++方式输入输出的格式化控制

### 1. 在cout中使用格式化控制符

#### A. 总结及结论:

- 1、源程序中的整数，有3种不同进制的表示形式
- 2、无论源程序中整型常量表示为何种进制，它的机内存储均为二进制补码形式
- 3、如果想使数据输出时使用不同进制，要加dec、hex、oct等进制前导符
- 4、输出无二进制前导符
- 5、只有10进制有负数形式输出：  
16进制输出负数时，特征是输出的16进制数是原负数的二进制补码直接转换的16进制数  
8进制输出负数时，特征是输出的8进制数是原负数的二进制补码直接转换的8进制数



## §. 基础知识题 – C++方式输入输出的格式化控制

1. 在cout中使用格式化控制符

B. 进制前导符的连续使用：回答问题并将程序的运行结果截图贴上

```
#include <iostream>
#include <iomanip>

using namespace std;
int main()
{
    int a = 10;

    cout << a << ',' << a+1 << ',' << a+2 << endl;
    cout << hex;
    cout << a << ',' << a+1 << ',' << a+2 << endl;
    cout << oct;
    cout << a << ',' << a+1 << ',' << a+2 << endl;
    cout << dec;
    cout << a << ',' << a+1 << ',' << a+2 << endl;

    return 0;
}
```

Microsoft Visual Studio 调试控制台

```
10 11 12
a b c
12 13 14
10 11 12
```

结论：

dec/hex/oct等进制前导符设置后，对后面的所有数据有效，直到用另一个控制符去改变为止



# §. 基础知识题 – C++方式输入输出的格式化控制

## 1. 在cout中使用格式化控制符

C. setbase的使用：同1. A的形式，按要求自行构造测试程序，回答问题并将程序的运行结果截图贴上

```
My Project.cpp ② X
My Project (全局范围)

1 #include <iostream>
2 #include <iomanip>
3
4 using namespace std;
5 int main()
6 {
7     int a = 10;
8
9     cout << setbase(8);
10    cout << a << ',' << a + 1 << ',' << a + 2 << endl;
11    cout << setbase(10);
12    cout << a << ',' << a + 1 << ',' << a + 2 << endl;
13    cout << setbase(16);
14    cout << a << ',' << a + 1 << ',' << a + 2 << endl;
15    cout << setbase(2);
16    cout << a << ',' << a + 1 << ',' << a + 2 << endl;
17
18    return 0;
19 }
```

自行构造若干组测试数据，运行并截图

结论：

- 1、setbase中允许的合法值有8、10、16
- 2、当setbase中出现非法值时，处理方法是以默认十进制形式输出数据
- 3、setbase设置后，对后面的所有数据有效，直到用另一个setbase去改变为止

```
Microsoft Visual Studio 调试控制台
12 13 14
10 11 12
a b c
10 11 12
```



# §. 基础知识题 – C++方式输入输出的格式化控制

## 1. 在cout中使用格式化控制符

D. ios::uppercase的使用：按要求自行构造测试程序，能对比看出用和不用的差别即可

```
My Project.cpp 1 (全局范围)
1 #include <iostream>
2 #include <iomanip>
3 using namespace std;
4
5 int main()
6 {
7     int a = 10;
8
9     cout << a << ',' << a + 1 << ',' << a + 2 << endl;
10    cout << hex;
11    cout << a << ',' << a + 1 << ',' << a + 2 << endl;
12    cout << oct;
13    cout << a << ',' << a + 1 << ',' << a + 2 << endl;
14    cout << dec;
15    cout << a << ',' << a + 1 << ',' << a + 2 << endl;
16
17    cout << setiosflags(ios::uppercase);
18    cout << a << ',' << a + 1 << ',' << a + 2 << endl;
19    cout << hex;
20    cout << a << ',' << a + 1 << ',' << a + 2 << endl;
21    cout << oct;
22    cout << a << ',' << a + 1 << ',' << a + 2 << endl;
23    cout << dec;
24    cout << a << ',' << a + 1 << ',' << a + 2 << endl;
25
26    cout << resetiosflags(ios::uppercase);
27    cout << a << ',' << a + 1 << ',' << a + 2 << endl;
28    cout << hex;
29    cout << a << ',' << a + 1 << ',' << a + 2 << endl;
30    cout << oct;
31    cout << a << ',' << a + 1 << ',' << a + 2 << endl;
32    cout << dec;
33    cout << a << ',' << a + 1 << ',' << a + 2 << endl;
34
35    return 0;
36 }
```

测试程序中的数据类型为int，自行构造若干组测试数据，运行并截图

结论：

- 1、 uppercase和16进制一起使用才能看出效果
- 2、 uppercase设置后，对后面的所有数据有效
- 3、 同一个程序中，设置完uppercase，如果想恢复小写，具体的做法是使用resetiosflags(ios::uppercase)；终止已设置的输出格式状态

```
Microsoft Visual Studio 调试控制台
10 11 12
a b c
12 13 14
10 11 12
10 11 12
A B C
12 13 14
10 11 12
10 11 12
a b c
12 13 14
10 11 12
```



# §. 基础知识题 – C++方式输入输出的格式化控制

## 1. 在cout中使用格式化控制符

E. ios::showpos的使用：按要求自行构造测试程序，能对比看出用和不用的差别即可

```
My Project.cpp
My Project (全局范围)

1 #include <iostream>
2 #include <iomanip>
3 using namespace std;
4
5 int main()
6 {
7     int a = 10;
8
9     cout << a << ',' << a + 1 << ',' << a + 2 << endl;
10    cout << hex;
11    cout << a << ',' << a + 1 << ',' << a + 2 << endl;
12    cout << oct;
13    cout << a << ',' << a + 1 << ',' << a + 2 << endl;
14    cout << dec;
15    cout << a << ',' << a + 1 << ',' << a + 2 << endl;
16
17    cout << setiosflags(ios::showpos);
18    cout << a << ',' << a + 1 << ',' << a + 2 << endl;
19    cout << hex;
20    cout << a << ',' << a + 1 << ',' << a + 2 << endl;
21    cout << oct;
22    cout << a << ',' << a + 1 << ',' << a + 2 << endl;
23    cout << dec;
24    cout << a << ',' << a + 1 << ',' << a + 2 << endl;
25
26    cout << resetiosflags(ios::showpos);
27    cout << a << ',' << a + 1 << ',' << a + 2 << endl;
28    cout << hex;
29    cout << a << ',' << a + 1 << ',' << a + 2 << endl;
30    cout << oct;
31    cout << a << ',' << a + 1 << ',' << a + 2 << endl;
32    cout << dec;
33    cout << a << ',' << a + 1 << ',' << a + 2 << endl;
34
35    return 0;
36 }
```

测试程序中的数据类型为int，自行构造若干组测试数据，运行并截图

结论：

- 1、 showpos和10进制一起使用才能看出效果
- 2、 showpos设置后，对后面的所有数据有效
- 3、 同一个程序中，设置完showpos，如果想取消，具体的做法是使用resetiosflags(ios::showpos)；终止已设置的输出格式状态

```
Microsoft Visual Studio 调试控制台
10 11 12
a b c
12 13 14
10 11 12
+10 +11 +12
a b c
12 13 14
+10 +11 +12
10 11 12
a b c
12 13 14
10 11 12
```



# §. 基础知识题 – C++方式输入输出的格式化控制

## 1. 在cout中使用格式化控制符

### F. setprecision的使用 – 单独使用 – (1)

```
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
    float f1 = 1234.5678F;
    float f2 = 8765.4321F;

    /* 第1组：不设或非法 */
    cout << f1 << ',' << f2 << endl;
    cout << setprecision(0) << f1 << ',' << f2 << endl;

    /* 第2组：小于等于整数位数 */
    cout << endl;
    cout << setprecision(1) << f1 << ',' << f2 << endl;
    cout << setprecision(2) << f1 << ',' << f2 << endl;
    cout << setprecision(3) << f1 << ',' << f2 << endl;
    cout << setprecision(4) << f1 << ',' << f2 << endl;

    /* 第3组：大于整数位数，但小与等于float型有效数字 */
    cout << endl;
    cout << setprecision(5) << f1 << ',' << f2 << endl;
    cout << setprecision(6) << f1 << ',' << f2 << endl;
    cout << setprecision(7) << f1 << ',' << f2 << endl;

    /* 第4组：大于float型有效数字 */
    cout << endl;
    cout << setprecision(8) << f1 << ',' << f2 << endl;
    cout << setprecision(9) << f1 << ',' << f2 << endl;
    cout << setprecision(10) << f1 << ',' << f2 << endl;
    cout << setprecision(25) << f1 << ',' << f2 << endl;

    return 0;
}
```

```
Microsoft Visual Studio 调试控制台
1234.57 8765.43
1e+03 9e+03
1e+03 9e+03
1.2e+03 8.8e+03
1.23e+03 8.77e+03
1235 8765
1234.6 8765.4
1234.57 8765.43
1234.568 8765.432
1234.5677 8765.4316
1234.56775 8765.43164
1234.567749 8765.431641
1234.5677490234375 8765.431640625
```



# §. 基础知识题 - C++方式输入输出的格式化控制

## 1. 在cout中使用格式化控制符

### F. setprecision的使用 - 单独使用 - (2)

```
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
    float f1 = 1234567890123456789.0F;
    float f2 = 9876543210987654321.0F;

    /* 第1组: 不设或非法 */
    cout << f1 << ',' << f2 << endl;
    cout << setprecision(0) << f1 << ',' << f2 << endl;

    /* 第2组: 小于等于整数位数 并且 小与等于float型有效数字 */
    cout << endl;
    cout << setprecision(1) << f1 << ',' << f2 << endl;
    cout << setprecision(2) << f1 << ',' << f2 << endl;
    cout << setprecision(3) << f1 << ',' << f2 << endl;
    cout << setprecision(4) << f1 << ',' << f2 << endl;
    cout << setprecision(5) << f1 << ',' << f2 << endl;
    cout << setprecision(6) << f1 << ',' << f2 << endl;
    cout << setprecision(7) << f1 << ',' << f2 << endl;

    /* 第3组: 大于float型有效数字 */
    cout << endl;
    cout << setprecision(8) << f1 << ',' << f2 << endl;
    cout << setprecision(9) << f1 << ',' << f2 << endl;
    cout << setprecision(10) << f1 << ',' << f2 << endl; //为什么f1比f2少一位?
    cout << setprecision(11) << f1 << ',' << f2 << endl; 最后一位是0被省略
    cout << setprecision(25) << f1 << ',' << f2 << endl;

    return 0;
}
```

```
Microsoft Visual Studio 调试控制台
1. 23457e+18 9.87654e+18
1e+18 1e+19

1e+18 1e+19
1.2e+18 9.9e+18
1.23e+18 9.88e+18
1.235e+18 9.877e+18
1.2346e+18 9.8765e+18
1.23457e+18 9.87654e+18
1.234568e+18 9.876544e+18

1. 2345679e+18 9.8765435e+18
1.23456794e+18 9.87654352e+18
1.23456794e+18 9.876543516e+18
1.2345679396e+18 9.8765435164e+18
1234567939550609408 9876543516404875264
```



## §. 基础知识题 - C++方式输入输出的格式化控制

### 1. 在cout中使用格式化控制符

#### F. setprecision的使用 - 单独使用 - (3)

```
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
    float f1 = 0.12345678F;
    float f2 = 0.87654321F;

    /* 第1组: 不设或非法 */
    cout << f1 << ',' << f2 << endl;
    cout << setprecision(0) << f1 << ',' << f2 << endl;

    /* 第2组: 小于等于float型有效数字 */
    cout << endl;
    cout << setprecision(1) << f1 << ',' << f2 << endl;
    cout << setprecision(2) << f1 << ',' << f2 << endl;
    cout << setprecision(3) << f1 << ',' << f2 << endl;
    cout << setprecision(4) << f1 << ',' << f2 << endl;
    cout << setprecision(5) << f1 << ',' << f2 << endl;
    cout << setprecision(6) << f1 << ',' << f2 << endl;
    cout << setprecision(7) << f1 << ',' << f2 << endl;

    /* 第3组: 大于float型有效数字 */
    cout << endl;
    cout << setprecision(8) << f1 << ',' << f2 << endl;
    cout << setprecision(9) << f1 << ',' << f2 << endl;
    cout << setprecision(10) << f1 << ',' << f2 << endl;
    cout << setprecision(25) << f1 << ',' << f2 << endl;

    return 0;
}
```

```
Microsoft Visual Studio 调试控制台
0.123457 0.876543
0.1 0.9
0.1 0.9
0.12 0.88
0.123 0.877
0.1235 0.8765
0.12346 0.87654
0.123457 0.876543
0.1234568 0.8765432
0.12345678 0.87654322
0.123456784 0.876543224
0.1234567836 0.8765432239
0.1234567835927009582519531 0.876543223857879638671875
```



## §. 基础知识题 – C++方式输入输出的格式化控制

### 1. 在cout中使用格式化控制符

#### F. setprecision的使用 – 单独使用 – 总结

**重要结论:** setprecision指定输出位数后, 系统会按指定位数输出, 即使指定位数超过数据的有效位数  
(即: 输出数据的某位开始是不可信的, 但依然会输出)

1、给出setprecision单独使用时的显示规律总结 (如果数据不够, 可以再自己构造测试数据)

- (1) 当不设置setprecision时, 以默认形式输出数据, 即保留6位有效数字, 超出有效位数的部分四舍五入;
- (2) 当非法设置setprecision时, 输出数据保留1位有效数字, 超出有效位数的部分四舍五入;
- (3) 当设置setprecision小于等于float型有效数字时, 输出数据保留设定位数的有效数字, 超出有效位数的部分四舍五入;
- (4) 当设置setprecision大于float型有效数字时, 将按照IEEE754存储的数据四舍五入至对应有效位数, 当指定位数超过数据的有效位数时, 输出数据从这一位开始是不可信的, 但依然会输出。

2、将1. F-(1)~(3)中的数据类型换为double型 (有效位数为15位), 自行构造测试数据, 验证总结出的float型数据的显示规律是否同样适用于double型

适用

## §. 基础知识题 – C++方式输入输出的格式化控制



## 1. 在cout中使用格式化控制符

### G. `setprecision`的使用 - 和`ios::fixed`一起 - (1)

```
#include <iostream>
#include <iomanip>

using namespace std;
int main()
{
    float f1 = 1234.5678F;
    float f2 = 8765.4321F;

    /* 第1组：不设precision */
    cout << f1 << ',' << f2 << endl;
    cout << setiosflags(ios::fixed) << f1 << ',' << f2 << endl;

    /* 第2组：设置precision */
    cout << endl;
    cout << setprecision(1) << f1 << ',' << f2 << endl;
    cout << setprecision(4) << f1 << ',' << f2 << endl;
    cout << setprecision(7) << f1 << ',' << f2 << endl;
    cout << setprecision(10) << f1 << ',' << f2 << endl;
    cout << setprecision(25) << f1 << ',' << f2 << endl;

    return 0;
}
```

```
1234.57 8765.43  
1234.567749 8765.431641  
  
1234.6 8765.4  
1234.5677 8765.4316  
1234.5677490 8765.4316406  
1234.5677490234 8765.4316406250  
1234.56774902343750000000000000 8765.43164062500000000000000000
```



## §. 基础知识题 – C++方式输入输出的格式化控制

### 1. 在cout中使用格式化控制符

#### G. setprecision的使用 – 和ios::fixed一起 – (2)

```
#include <iostream>
#include <iomanip>

using namespace std;
int main()
{
    float f1 = 1234567890123456789.0F;
    float f2 = 9876543210987654321.0F;

    /* 第1组: 不设precision */
    cout << f1 << ',' << f2 << endl;
    cout << setiosflags(ios::fixed) << f1 << ',' << f2 << endl;

    /* 第2组: 设置precision */
    cout << endl;
    cout << setprecision(1) << f1 << ',' << f2 << endl;
    cout << setprecision(4) << f1 << ',' << f2 << endl;
    cout << setprecision(7) << f1 << ',' << f2 << endl;
    cout << setprecision(10) << f1 << ',' << f2 << endl;
    cout << setprecision(25) << f1 << ',' << f2 << endl;

    return 0;
}
```

Microsoft Visual Studio 调试控制台

```
1. 23457e+18 9.87654e+18
1234567939550609408.000000 9876543516404875264.000000
1234567939550609408.0 9876543516404875264.0
1234567939550609408.0000 9876543516404875264.0000
1234567939550609408.000000 9876543516404875264.000000
1234567939550609408.0000000000 9876543516404875264.0000000000
1234567939550609408.0000000000000000 9876543516404875264.0000000000000000
```



## §. 基础知识题 – C++方式输入输出的格式化控制

### 1. 在cout中使用格式化控制符

#### G. setprecision的使用 – 和ios::fixed一起 – (3)

```
#include <iostream>
#include <iomanip>

using namespace std;
int main()
{
    float f1 = 0.12345678F;
    float f2 = 0.87654321F;

    /* 第1组: 不设precision */
    cout << f1 << ',' << f2 << endl;
    cout << setiosflags(ios::fixed) << f1 << ',' << f2 << endl;

    /* 第2组: 设置precision */
    cout << endl;
    cout << setprecision(1) << f1 << ',' << f2 << endl;
    cout << setprecision(4) << f1 << ',' << f2 << endl;
    cout << setprecision(7) << f1 << ',' << f2 << endl;
    cout << setprecision(10) << f1 << ',' << f2 << endl;
    cout << setprecision(25) << f1 << ',' << f2 << endl;

    return 0;
}
```

```
Microsoft Visual Studio 调试控制台
0.123457 0.876543
0.123457 0.876543
0.1 0.9
0.1235 0.8765
0.1234568 0.8765432
0.1234567836 0.8765432239
0.1234567835927009582519531 0.8765432238578796386718750
```



## §. 基础知识题 – C++方式输入输出的格式化控制

### 1. 在cout中使用格式化控制符

#### G. setprecision的使用 – 和ios::fixed一起 – 总结

1、给出setprecision+ios::fixed使用时的显示规律总结（如果数据不够，可以再自己构造测试数据）

(1) 不设置setprecision，只设置ios::fixed时，浮点数保留6位小数位数输出，超出保留小数位数的部分四舍五入；

(2) 先设置ios::fixed，再设置setprecision(n)时，浮点数保留n位小数位数输出，将按照IEEE754存储的数据四舍五入至对应保留小数位数。

2、将1.G-(1)~(3)中的数据类型换为double型（有效位数为15位），自行构造测试数据，验证总结出的float型数据的显示规律是否同样适用于double型

适用



## §. 基础知识题 – C++方式输入输出的格式化控制

### 1. 在cout中使用格式化控制符

#### H. setprecision的使用 – 和ios::scientific一起 – (1)

```
#include <iostream>
#include <iomanip>

using namespace std;
int main()
{
    float f1 = 1234.5678F;
    float f2 = 8765.4321F;

    /* 第1组: 不设precision */
    cout << f1 << ',' << f2 << endl;
    cout << setiosflags(ios::scientific) << f1 << ',' << f2 << endl;

    /* 第2组: 设置precision */
    cout << endl;
    cout << setprecision(1) << f1 << ',' << f2 << endl;
    cout << setprecision(4) << f1 << ',' << f2 << endl;
    cout << setprecision(7) << f1 << ',' << f2 << endl;
    cout << setprecision(10) << f1 << ',' << f2 << endl;
    cout << setprecision(25) << f1 << ',' << f2 << endl;

    return 0;
}
```

```
Microsoft Visual Studio 调试控制台
1234.57 8765.43
1.234568e+03 8.765432e+03

1.2e+03 8.8e+03
1.2346e+03 8.7654e+03
1.2345677e+03 8.7654316e+03
1.2345677490e+03 8.7654316406e+03
1.23456774902343750000000000e+03 8.765431640625000000000000e+03
```



## §. 基础知识题 – C++方式输入输出的格式化控制

### 1. 在cout中使用格式化控制符

#### H. setprecision的使用 – 和ios::scientific一起 – (2)

```
#include <iostream>
#include <iomanip>

using namespace std;
int main()
{
    float f1 = 1234567890123456789.0F;
    float f2 = 9876543210987654321.0F;

    /* 第1组: 不设precision */
    cout << f1 << ',' << f2 << endl;
    cout << setiosflags(ios::scientific) << f1 << ',' << f2 << endl;

    /* 第2组: 设置precision */
    cout << endl;
    cout << setprecision(1) << f1 << ',' << f2 << endl;
    cout << setprecision(4) << f1 << ',' << f2 << endl;
    cout << setprecision(7) << f1 << ',' << f2 << endl;
    cout << setprecision(10) << f1 << ',' << f2 << endl;
    cout << setprecision(25) << f1 << ',' << f2 << endl;

    return 0;
}
```

Microsoft Visual Studio 调试控制台

```
1. 23457e+18 9.87654e+18
1. 234568e+18 9.876544e+18

1. 2e+18 9.9e+18
1. 2346e+18 9.8765e+18
1. 2345679e+18 9.8765435e+18
1. 2345679396e+18 9.8765435164e+18
1. 2345679395506094080000000e+18 9.8765435164048752640000000e+18
```



## §. 基础知识题 – C++方式输入输出的格式化控制

### 1. 在cout中使用格式化控制符

#### H. setprecision的使用 – 和ios::scientific一起 – (3)

```
#include <iostream>
#include <iomanip>

using namespace std;
int main()
{
    float f1 = 0.12345678F;
    float f2 = 0.87654321F;

    /* 第1组: 不设precision */
    cout << f1 << ',' << f2 << endl;
    cout << setiosflags(ios::scientific) << f1 << ',' << f2 << endl;

    /* 第2组: 设置precision */
    cout << endl;
    cout << setprecision(1) << f1 << ',' << f2 << endl;
    cout << setprecision(4) << f1 << ',' << f2 << endl;
    cout << setprecision(7) << f1 << ',' << f2 << endl;
    cout << setprecision(10) << f1 << ',' << f2 << endl;
    cout << setprecision(25) << f1 << ',' << f2 << endl;

    return 0;
}
```

```
Microsoft Visual Studio 调试控制台
0.123457 0.876543
1.234568e-01 8.765432e-01
1.2e-01 8.8e-01
1.2346e-01 8.7654e-01
1.2345678e-01 8.7654322e-01
1.2345678359e-01 8.7654322386e-01
1.2345678359270095825195312e-01 8.7654322385787963867187500e-01
```



## §. 基础知识题 – C++方式输入输出的格式化控制

### 1. 在cout中使用格式化控制符

#### H. setprecision的使用 – 和ios::scientific一起 – 总结

1、给出setprecision+ios::scientific使用时的显示规律总结（如果数据不够，可以再自己构造测试数据）

(1) 不设置setprecision，只设置ios::scientific时，浮点数保留6位小数位数，以科学计数法（即指数形式）输出，超出保留小数位数的部分四舍五入；

(2) 先设置ios::scientific，再设置setprecision(n)时，浮点数保留n位小数位数，以科学计数法（即指数形式）输出，将按照IEEE754存储的数据四舍五入至对应保留小数位数。

2、将1. H-(1)~(3)中的数据类型换为double型（有效位数为15位），自行构造测试数据，验证总结出的float型数据的显示规律是否同样适用于double型

适用



# §. 基础知识题 – C++方式输入输出的格式化控制

## 1. 在cout中使用格式化控制符

### I. ios::fixed和ios::scientific的混合使用 – 错误用法

```
#include <iostream>
#include <iomanip>

using namespace std;
int main()
{
    float f1 = 1234.5678F, f2 = 8765.4321F;

    /* 第1组 */
    cout << f1 << ',' << f2 << endl;
    cout << setiosflags(ios::fixed) << f1 << ',' << f2 << endl;

    /* 第2组 */
    cout << endl;
    cout << setiosflags(ios::scientific) << f1 << ',' << f2 << endl;

    return 0;
}
```

```
#include <iostream>
#include <iomanip>

using namespace std;
int main()
{
    float f1 = 1234.5678F, f2 = 8765.4321F;

    /* 第1组 */
    cout << f1 << ',' << f2 << endl;
    cout << setiosflags(ios::scientific) << f1 << ',' << f2 << endl;

    /* 第2组 */
    cout << endl;
    cout << setiosflags(ios::fixed) << f1 << ',' << f2 << endl;

    return 0;
}
```

运行截图:

```
Microsoft Visual Studio 调试控制台
1234.57 8765.43
1234.567749 8765.431641
0x1.34a4560000000p+10 0x1.11eb740000000p+13
```

运行截图:

```
Microsoft Visual Studio 调试控制台
1234.57 8765.43
1.234568e+03 8.765432e+03
0x1.34a4560000000p+10 0x1.11eb740000000p+13
```



# §. 基础知识题 – C++方式输入输出的格式化控制

## 1. 在cout中使用格式化控制符

### I. ios::fixed和ios::scientific的混合使用 – 在上一页的基础上将程序改正确，并给出截图

```
#include <iostream>
#include <iomanip>

using namespace std;
int main()
{
    float f1 = 1234.5678F, f2 = 8765.4321F;

    /* 第1组 */
    cout << f1 << ',' << f2 << endl;
    cout << setiosflags(ios::fixed) << f1 << ',' << f2 << endl;

    /* 第2组 */
    cout << endl;
    cout << setiosflags(ios::scientific) << f1 << ',' << f2 << endl;

    return 0;
}
```

```
#include <iostream>
#include <iomanip>

using namespace std;
int main()
{
    float f1 = 1234.5678F, f2 = 8765.4321F;

    /* 第1组 */
    cout << f1 << ',' << f2 << endl;
    cout << setiosflags(ios::scientific) << f1 << ',' << f2 << endl;

    /* 第2组 */
    cout << endl;
    cout << setiosflags(ios::fixed) << f1 << ',' << f2 << endl;

    return 0;
}
```

运行截图:

Microsoft Visual Studio 调试控制台

```
1234.57 8765.43
1234.567749 8765.431641
0x1.34a4560000000p+10 0x1.11eb740000000p+13
```

运行截图:

Microsoft Visual Studio 调试控制台

```
1234.57 8765.43
1.234568e+03 8.765432e+03
0x1.34a4560000000p+10 0x1.11eb740000000p+13
```

结论:

如果想要在一个程序中同时显示fixed和scientific形式，需要在两者之间加入一句:

cout << setiosflags(ios::fixed); 或 cout << setiosflags(ios::scientific);



## §. 基础知识题 – C++方式输入输出的格式化控制

### 1. 在cout中使用格式化控制符

#### J. setw的基本使用 – (1)

```
#include <iostream>
#include <iomanip>

using namespace std;
int main()
{
    int a = 12345;

    cout << "0      1      2      3" << endl;
    cout << "0123456789012345678901234567890123456789" << endl;

    cout << setw(3) << a << '#' << a + 1 << '*' << endl;
    cout << setw(6) << a << '#' << a + 1 << '*' << endl;
    cout << setw(10) << a << '#' << a + 1 << '*' << endl;
    cout << setw(15) << a << '#' << a + 1 << '*' << endl;

    return 0;
}
```

```
Microsoft Visual Studio 调试控制台
0      1      2      3
0123456789012345678901234567890123456789
12345#12346*
12345#12346*
12345#12346*
12345#12346*
```

#### 结论：

- 1、setw指定的宽度是总宽度，当总宽度大于数据宽度时，显示规律为输出宽度被设置为n，缺省为右对齐，并用空格填充；当总宽度小于数据宽度时，显示规律为setw设置的宽度将被忽略，直接以默认形式输出数据。
- 2、setw的设置后，对后面的仅一个数据有效
- 3、程序最前面两行的输出，目的是什么？易于阅读，方便确定输出宽度，显示每个数字对应的位置。
- 4、每行输出的最后一个\*，目的是什么？a+1已输出完毕，#与\*之间输出的内容即为a+1输出的内容，用于显示末尾的位置，也是为了看清是否有隐含空格，便于确定输出宽度。



## §. 基础知识题 – C++方式输入输出的格式化控制

### 1. 在cout中使用格式化控制符

#### J. setw的基本使用 – (2)

```
#include <iostream>
#include <iomanip>

using namespace std;
int main()
{
    double a = 0.123456789012345;

    cout << "0      1      2      3" << endl;
    cout << "0123456789012345678901234567890123456789" << endl;

    cout << setw(6) << a << '*' << endl;
    cout << setw(9) << a << '*' << endl;
    cout << setw(15) << a << '*' << endl;
    cout << setw(30) << a << '*' << endl;

    return 0;
}
```

```
Microsoft Visual Studio 调试控制台
0      1      2      3
0123456789012345678901234567890123456789
0.123457*
0.123457*
0.123457*
0.123457*
```

结论：

1、setw指定的宽度是总宽度，对于实型数据，包含小数点



## §. 基础知识题 – C++方式输入输出的格式化控制

### 1. 在cout中使用格式化控制符

#### K. setw+setfill的使用

```
#include <iostream>
#include <iomanip>

using namespace std;
int main()
{
    int a = 12345;

    cout << "0      1      2      3" << endl;
    cout << "0123456789012345678901234567890123456789" << endl;

    cout << setfill('=') << setw(10) << a << '#' << setw(10) << a + 1 << '*' << endl;
    cout << setw(15) << setfill('-') << a << '#' << a + 1 << '*' << endl;

    return 0;
}
```

```
Microsoft Visual Studio 调试控制台
0      1      2      3
0123456789012345678901234567890123456789
=====12345#=====12346*
-----12345#12346*
```

结论：

- 1、setfill的作用是setfill(c)可以设置填充字符，c可以是字符常量或字符变量
- 2、setfill的设置后，对后面的 所有 数据有效
- 3、解释为什么第4行的第2个数(12346)前面没有-：因为setw的设置后，对后面的仅一个数据有效，所以数据12346以默认形式输出，无空格填充，故无字符填充，12346前面没有-。



## §. 基础知识题 – C++方式输入输出的格式化控制

### 1. 在cout中使用格式化控制符

#### L. setw/setfill与ios::left/ios::right的混合使用 – (1)

```
#include <iostream>
#include <iomanip>

using namespace std;
int main()
{
    int a = 12345;
    cout << "0      1      2      3" << endl;
    cout << "0123456789012345678901234567890123456789" << endl;
    cout << setw(10) << a << '#' << setw(10) << a + 1 << '*' << endl;
    cout << setiosflags(ios::left);
    cout << setw(10) << a << '#' << setw(10) << a + 1 << '*' << endl;
    return 0;
}
```

```
Microsoft Visual Studio 调试控制台
0      1      2      3
01234567890123456789012345678901234567890123456789
12345#    12346*
12345      #12346      *
```

```
#include <iostream>
#include <iomanip>

using namespace std;
int main()
{
    int a = 12345;
    cout << "0      1      2      3" << endl;
    cout << "0123456789012345678901234567890123456789" << endl;
    cout << setfill('=') << setw(10) << a << '#' << setw(10) << a + 1 << '*' << endl;
    cout << setiosflags(ios::left);
    cout << setfill('=') << setw(10) << a << '#' << setw(10) << a + 1 << '*' << endl;
    return 0;
}
```

结论：

- 1、ios::left的作用是使输出数据左对齐
- 2、如果不设置，缺省是右对齐

```
Microsoft Visual Studio 调试控制台
0      1      2      3
01234567890123456789012345678901234567890123456789
=====12345#=====12346*
12345=====#12346=====*
```



## §. 基础知识题 – C++方式输入输出的格式化控制

### 1. 在cout中使用格式化控制符

#### L. setw/setfill与ios::left/ios::right的混合使用 – (2) – 同时使用(错误)

```
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
    int a = 12345;
    cout << "0      1      2      3" << endl;
    cout << "0123456789012345678901234567890123456789" << endl;
    /* 左对齐 */
    cout << setiosflags(ios::left) << setw(10) << a << '#' << setw(10) << a + 1 << '*' << endl;
    /* 右对齐 */
    cout << setiosflags(ios::right) << setw(10) << a << '#' << setw(10) << a + 1 << '*' << endl;
    /* 左对齐 */
    cout << setiosflags(ios::left) << setw(10) << a << '#' << setw(10) << a + 1 << '*' << endl;
    return 0;
}
```

```
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
    int a = 12345;
    cout << "0      1      2      3" << endl;
    cout << "0123456789012345678901234567890123456789" << endl;
    /* 右对齐 */
    cout << setiosflags(ios::right) << setw(10) << a << '#' << setw(10) << a + 1 << '*' << endl;
    /* 左对齐 */
    cout << setiosflags(ios::left) << setw(10) << a << '#' << setw(10) << a + 1 << '*' << endl;
    return 0;
}
```

Microsoft Visual Studio 调试控制台

0	1	2	3
0123456789012345678901234567890123456789	12345	#12346	*
	12345#	12346*	
	12345#	12346*	

Microsoft Visual Studio 调试控制台

0	1	2	3
0123456789012345678901234567890123456789	12345#	12346*	
	12345#	12346*	



# §. 基础知识题 – C++方式输入输出的格式化控制

## 1. 在cout中使用格式化控制符

L. setw/setfill与ios::left/ios::right的混合使用 – 在上一页的基础上将程序改正确

```
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
    int a = 12345;
    cout << "0      1      2      3" << endl;
    cout << "0123456789012345678901234567890123456789" << endl;
    /* 左对齐 */
    cout << setiosflags(ios::left) << setw(10) << a << '#' << setw(10) << a + 1 << '*' << endl;
    /* 右对齐 */
    cout << setiosflags(ios::right) << setw(10) << a << '#' << setw(10) << a + 1 << '*' << endl;
    /* 左对齐 */
    cout << resetiosflags(ios::right);
    cout << setiosflags(ios::left) << setw(10) << a << '#' << setw(10) << a + 1 << '*' << endl;
    return 0;
}
```

```
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
    int a = 12345;
    cout << "0      1      2      3" << endl;
    cout << "0123456789012345678901234567890123456789" << endl;
    /* 右对齐 */
    cout << setiosflags(ios::right) << setw(10) << a << '#' << setw(10) << a + 1 << '*' << endl;
    /* 左对齐 */
    cout << resetiosflags(ios::right);
    cout << setiosflags(ios::left) << setw(10) << a << '#' << setw(10) << a + 1 << '*' << endl;
    return 0;
}
```

Microsoft Visual Studio 调试控制台

0	1	2	3
0123456789012345678901234567890123456789	12345	#12346	*
	12345#	12346*	
12345	#12346	*	

Microsoft Visual Studio 调试控制台

0	1	2	3
0123456789012345678901234567890123456789	12345#	12346*	
	12345	#12346	*



# §. 基础知识题 – C++方式输入输出的格式化控制

## 2. 在cin中使用格式化控制符

### A. 基本要求：从键盘输入16进制数

```
#include <iostream>
#include <iomanip>

using namespace std;
int main()
{
    short a;
    cin >> hex >> a;

    cout << "dec:" << dec << a << endl;
    cout << "hex:" << hex << a << endl;
    cout << "oct:" << oct << a << endl;

    return 0;
}
```

1、输入: 1a2b↙  
(合理正数)

Microsoft Visual Studio 调试控制台  
1a2b  
dec:6699  
hex:1a2b  
oct:15053

2、输入: a1b2↙  
(超上限但未超同类型的unsigned上限)

Microsoft Visual Studio 调试控制台  
a1b2  
dec:32767  
hex:7fff  
oct:77777

3、输入: ffffff↙  
(超上限且超过同类型的unsigned上限)

Microsoft Visual Studio 调试控制台  
ffffff  
dec:32767  
hex:7fff  
oct:77777

4、输入: -1a2b↙  
(合理负数)

Microsoft Visual Studio 调试控制台  
-1a2b  
dec:-6699  
hex:e5d5  
oct:162725

5、输入: -fffff↙  
(超下限)

Microsoft Visual Studio 调试控制台  
-fffff  
dec:-32768  
hex:8000  
oct:100000



## §. 基础知识题 – C++方式输入输出的格式化控制

### 2. 在cin中使用格式化控制符

#### B. 基本要求：从键盘输入8进制数（自行构造测试数据）

```
#include <iostream>
#include <iomanip>

using namespace std;
int main()
{
    int a;
    cin >> setbase(8) >> a;

    cout << "dec:" << dec << a << endl;
    cout << "hex:" << hex << a << endl;
    cout << "oct:" << oct << a << endl;

    return 0;
}
```

1、输入: 7346545000↙  
(合理正数)

Microsoft Visual Studio 调试控制台  
7346545000  
dec:1000000000  
hex:3b9aca00  
oct:7346545000

2、输入: 26264057000↙  
(超上限但未超同类型的unsigned上限)

Microsoft Visual Studio 调试控制台  
26264057000  
dec:2147483647  
hex:7fffffff  
oct:177777777777

3、输入: 45201371000↙  
(超上限且超过同类型的unsigned上限)

Microsoft Visual Studio 调试控制台  
45201371000  
dec:2147483647  
hex:7fffffff  
oct:177777777777

4、输入: -7346545000↙  
(合理负数)

Microsoft Visual Studio 调试控制台  
-7346545000  
dec:-1000000000  
hex:c4653600  
oct:30431233000

5、输入: -26264057000↙  
(超下限)

Microsoft Visual Studio 调试控制台  
-26264057000  
dec:-2147483648  
hex:80000000  
oct:200000000000



## §. 基础知识题 – C++方式输入输出的格式化控制

### 2. 在cin中使用格式化控制符

#### C. 格式控制符setiosflags(ios::skipws)的使用

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

int main()
{
    int a, b;
    cin >> a >> b;
    cout << a << endl;
    cout << b << endl;
    return 0;
}
```

```
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
    int a, b;
    cin >> setiosflags(ios::skipws);
    cin >> a >> b;
    cout << a << endl;
    cout << b << endl;
    return 0;
}
```

```
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
    int a, b;
    cin.unsetf(ios::skipws);
    cin >> a >> b;
    cout << a << endl;
    cout << b << endl;
    return 0;
}
```

假设键盘输入为: 12 34↙

则输出为: 12



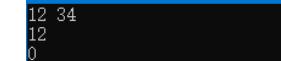
假设键盘输入为: 12 34↙

则输出为: 12



假设键盘输入为: 12 34↙

则输出为: 12



综合以上三个例子可以得到如下结论:

- 1、"忽略前导空格"的意思, 是空格不作为终止输入条件, 而是做为字符 (因此导致第3个例子b未取得34)
- 2、setiosflags(ios::skipws)在缺省情况下是有效的, 即不设置也生效
- 3、如果想取消"忽略前导空格"的设置, 应使用cout << resetiosflags(ios::skipws);



## §. 基础知识题 – C方式输入输出的格式化控制

### 1. 格式化输出函数printf的基本理解

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

```
#include <stdio.h>

int main()
{
    int a=10, b=5;
    printf("a=%d, b=%d\n", a, b);

    printf("Hello, Welcome!\n");
    printf("Hello, Welcome\x21\n");
    return 0;
}
```

运行结果：

```
a=10, b=5
Hello, Welcome!
Hello, Welcome!
```

\x21是哪个ASCII字符的16进制转义表示?  
!

转义符在格式控制表列中的输出形式是：  
字符

//写出与左侧程序输出完全一致的，用C++方式的cout实现的代码

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

int main()
{
    int a = 10, b = 5;
    cout << "a=" << a << ", b=" << b << endl;
    cout << "Hello, Welcome!" << endl;
    cout << "Hello, Welcome\x21" << endl;

    return 0;
}
```



## §. 基础知识题 – C方式输入输出的格式化控制

### 1. 格式化输出函数printf的基本理解

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

```
#include <stdio.h>

int main()
{
    int a=10, b=5;
    printf("a=%d\n", a, b);

    printf("Hello, Welcome!\n");
    return 0;
}
```

运行结果：

```
a=10
Hello, Welcome!
warning C4474: printf: 格式字符串中传递的参数太多
message : 占位符和其参数预计 1 可变参数, 但提供的却是 2 参数
```

结论：如果%d(格式符的数量) 小于 后面输出表列的数量，则只输出格式符数量个数的变量，超过格式符数量的变量将不会被输出。

```
#include <stdio.h>

int main()
{
    int a=10, b=5;
    printf("a=%d %d %d\n", a, b);

    printf("Hello, Welcome!\n");
    return 0;
}
```

运行结果：

```
a=10 5 13111331
Hello, Welcome!
warning C4473: “printf”：没有为格式字符串传递足够的参数
message : 占位符和其参数预计 3 可变参数, 但提供的却是 2 参数
message : 缺失的可变参数 3 为格式字符串 “%d” 所需
```

结论：如果%d(格式符的数量) 大于 后面输出表列的数量，则前输出表列数量个数的变量输出是正确的，超过输出表列数量的变量输出是不可信的。



## §. 基础知识题 – C方式输入输出的格式化控制

### 1. 格式化输出函数printf的基本理解

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

```
#include <stdio.h>

int main()
{
    int a=10, b=5;
    int ret1, ret2, ret3, ret4, ret5;

    ret1 = printf("a=%d, b=%d\n", a, b);
    ret2 = printf("a=%d b=%d\n", a, b); //跟上面比，少一个逗号

    ret3 = printf("a=%d\n", a*1000);

    ret4 = printf("Hello\n");
    ret5 = printf("Hello"); //跟上面比，少一个\n
    printf("\n");

    printf("%d %d %d %d\n", ret1, ret2, ret3, ret4, ret5);

    return 0;
}
```

运行结果：

```
Microsoft Visual Studio 调试控制台
a=10, b=5
a=10 b=5
a=10000
Hello
Hello
10 9 8 6 5
```

printf的返回值的含义是：

printf的返回值的数据类型为int型，其含义是打印的所有字符的总数（包括转义字符，比如回车\n）



# §. 基础知识题 – C方式输入输出的格式化控制

## 1. 格式化输出函数printf的基本理解

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

```
#include <stdio.h>

int main()
{
    short a = -2;
    printf("a=%hi %hd %hu %o %hx %hX\n", a, a, a, a, a, a);
    printf("a=%i %d %u %o %x %X\n", a, a, a, a, a, a);
    printf("a=%li %ld %lu %lo %lx %lX\n", a, a, a, a, a, a);

    unsigned short b = 40000;
    printf("b=%hi %hd %hu %o %hx %hX\n", b, b, b, b, b, b);
    printf("b=%i %d %u %o %x %X\n", b, b, b, b, b, b);
    printf("b=%li %ld %lu %lo %lx %lX\n", b, b, b, b, b, b);

    int c = 70000;
    printf("c=%hi %hd %hu %o %hx %hX\n", c, c, c, c, c, c);
    printf("c=%i %d %u %o %x %X\n", c, c, c, c, c, c);
    printf("c=%li %ld %lu %lo %lx %lX\n", c, c, c, c, c, c);

    return 0;
}
```

运行结果：

```
Microsoft Visual Studio 调试控制台
a=-2 -2 65534 177776 fffe FFFE
a=-2 -2 4294967294 37777777776 ffffffe FFFFFFFE
a=-2 -2 4294967294 37777777776 ffffffe FFFFFFFE
b=-25536 -25536 40000 116100 9c40 9C40
b=40000 40000 40000 116100 9c40 9C40
b=40000 40000 40000 116100 9c40 9C40
c=4464 4464 4464 10560 1170 1170
c=70000 70000 70000 210560 11170 11170
c=70000 70000 70000 210560 11170 11170
```

参考printf的格式控制符和附加格式控制符，给出解释：

附加控制符l的作用：  
表示长整型整数

附加控制符h的作用：  
表示短整型整数

★ 在C方式中，如果要输出的数据类型与格式控制符的  
类型不一致，则以格式控制符为准



# §. 基础知识题 – C方式输入输出的格式化控制

## 1. 格式化输出函数printf的基本理解

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

```
#include <stdio.h>

int main()
{
    int a = 70000;
    printf("a=%ld*\n", a);
    printf("a=%10ld*\n", a);
    printf("a=%-10ld*\n\n", a);

    printf("a=%d*\n", a);
    printf("a=%10d*\n", a);
    printf("a=%10d*\n", -a);
    printf("a=%-10d*\n\n", a);
    printf("a=%-10d*\n", -a);

    printf("a=%hd*\n", a);
    printf("a=%10hd*\n", a);
    printf("a=%-10hd*\n\n", a);

    return 0;
} //注：最后加*的目的，是为了看清是否有隐含空格
```

运行结果：

```
a=70000*
a=    70000*
a=70000    *
a=70000*
a=    70000*
a=   -70000*
a=70000    *
a=-70000    *
a=4464*
a=    4464*
a=4464    *
```

参考printf的格式控制符和附加格式控制符，给出解释：

%ld : 以长整型类型的数据类型输出  
%10ld : 以长整型类型输出，总宽度10，右对齐  
%-10ld: 以长整型类型输出，总宽度10，左对齐  
%d : 以整型类型的数据类型输出  
%10d : 以整型类型输出，总宽度10，右对齐  
%-10d : 以整型类型输出，总宽度10，左对齐  
%hd : 以短整型类型的数据类型输出  
%10hd : 以短整型类型输出，总宽度10，右对齐  
%-10hd: 以短整型类型输出，总宽度10，左对齐

如果输出负数且指定宽度，负号 占 总宽度



# §. 基础知识题 – C方式输入输出的格式化控制

## 1. 格式化输出函数printf的基本理解

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

```
#include <stdio.h>

int main()
{
    float f = 123.456f;
    printf("f=%f\n", f);
    printf("f=%e\n", f);
    printf("f=%E\n", f);
    printf("f=%g\n", f);
    printf("f=%G\n\n", f);

    f = 0.123456789f;
    printf("f=%f\n", f);
    printf("f=%e\n", f);
    printf("f=%E\n", f);
    printf("f=%g\n", f);
    printf("f=%G\n\n", f);

    f = 123456789.0f;
    printf("f=%f\n", f);
    printf("f=%e\n", f);
    printf("f=%E\n", f);
    printf("f=%g\n", f);
    printf("f=%G\n\n", f);

    return 0;
}
```

运行结果：

参考printf的格式控制符和附加格式控制符，给出解释：

%f：将浮点数以十进制的小数形式输出

%e：将浮点数以十进制的指数形式输出

%E：将浮点数以十进制的指数形式输出

%e和%E的区别是：以十进制的指数形式输出时，%e输出小写e，%E输出大写E

%g/%G：输出形式为从f, e中选择宽度较短的形式输出浮点数

★ 仔细观察并叙述清楚，如果觉得左例还不足以理解，可以自己再构造测试数据

%g/%G：输出形式的差别为若以小数形式输出则无差别，若以指数形式输出%g输出小写e，%G输出大写E

Microsoft Visual Studio 调试控制台

```
f=123.456001
f=1.234560e+02
f=1.234560E+02
f=123.456
f=123.456

f=0.123457
f=1.234568e-01
f=1.234568E-01
f=0.123457
f=0.123457

f=123456792.000000
f=1.234568e+08
f=1.234568E+08
f=1.23457e+08
f=1.23457E+08
```



# §. 基础知识题 – C方式输入输出的格式化控制

## 1. 格式化输出函数printf的基本理解

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

```
#include <stdio.h>
int main()
{
    double f = 123.456;
    printf("f=%f\n", f);
    printf("f=%lf\n", f);
    printf("f=%e\n", f);
    printf("f=%le\n", f);
    printf("f=%g\n", f);
    printf("f=%lg\n\n", f);

    f = 0.123456789;
    printf("f=%f\n", f);
    printf("f=%lf\n", f);
    printf("f=%e\n", f);
    printf("f=%le\n", f);
    printf("f=%g\n", f);
    printf("f=%lg\n\n", f);

    f = 123456789.0;
    printf("f=%f\n", f);
    printf("f=%lf\n", f);
    printf("f=%e\n", f);
    printf("f=%le\n", f);
    printf("f=%g\n", f);
    printf("f=%lg\n\n", f);
    return 0;
}
```

运行结果：

参考printf的格式控制符和附加格式控制符，给出解释：

- (1) 附加控制符f表示以小数形式输出
- (2) 附加控制符e表示以指数形式输出
- (3) 附加控制符g表示从f和e中选择宽度较短的形式输出

对于double数据：

1、格式符%f和%lf是否有区别？  
无区别

2、如何证明你给出的1的结论？  
三组数据均可证明该结论

Microsoft Visual Studio 调试控制台

```
f=123. 456000
f=123. 456000
f=1. 234560e+02
f=1. 234560e+02
f=123. 456
f=123. 456

f=0. 123457
f=0. 123457
f=1. 234568e-01
f=1. 234568e-01
f=0. 123457
f=0. 123457

f=123456789. 000000
f=123456789. 000000
f=1. 234568e+08
f=1. 234568e+08
f=1. 23457e+08
f=1. 23457e+08
```



# §. 基础知识题 – C方式输入输出的格式化控制

## 1. 格式化输出函数printf的基本理解

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

```
#include <stdio.h>

int main()
{
    double f = 123456.789;

    printf("f=%f\n", f);
    printf("f=%.2f\n", f);
    printf("f=%10.2f\n", f);
    printf("f=-10.2f\n", f);

    printf("f=%e\n", f);
    printf("f=%.2e\n", f);
    printf("f=%10.2e\n", f);
    printf("f=-10.2e\n", f);

    printf("f=%g\n", f);
    printf("f=%.2g\n", f);
    printf("f=%.3g\n", f);
    printf("f=%10.2g\n", -f);
    printf("f=%10.3g\n", f);
    printf("f=-10.2g\n", -f);
    printf("f=-10.3g\n", f);

    return 0;
}

//注：最后加*的目的，是为了看清是否有隐含空格
```

运行结果：

```
Microsoft Visual Studio 调试控制台
f=123456. 789000*
f=123456. 79*
f= 123456. 79*
f=123456. 79 *

f=1. 234568e+05*
f=1. 23e+05*
f= 1. 23e+05*
f=1. 23e+05  *

f=123457*
f=1. 2e+05*
f=1. 23e+05*
f= -1. 2e+05*
f= 1. 23e+05*
f=-1. 2e+05  *
f=1. 23e+05  *
```

参考printf的格式控制符和附加格式控制符，给出解释：

%10.2f : 以小数类型输出，总宽度10，小数点后2位，右对齐  
%-10.2f: 以小数类型输出，总宽度10，小数点后2位，左对齐  
%10.2e : 以指数类型输出，总宽度10，小数点后2位，右对齐  
%-10.2e: 以指数类型输出，总宽度10，小数点后2位，左对齐

对%f和%e而言，指定的总宽度包含小数点

对%g而言，%m.n中n代表的位数是指保留小数的位数

如果输出负数且指定宽度，负号占总宽度



# §. 基础知识题 – C方式输入输出的格式化控制

## 1. 格式化输出函数printf的基本理解

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

```
#include <stdio.h>

int main()
{
    float f = 123456789.123;

    printf("f=%f*\n", f);
    printf("f=%10.2f*\n", f);
    printf("f=%-10.2f*\n", f);
    printf("f=%.2f*\n\n", f);

    double d = 12345678901234567.6789;

    printf("d=%f*\n", d);
    printf("d=%10.2f*\n", d);
    printf("d=%-10.2f*\n", d);
    printf("d=%.2f*\n\n", d);

    return 0;
}

//注：最后加*的目的，是为了看清是否有隐含空格
```

运行结果：

```
Microsoft Visual Studio 调试控制台
f=123456792.000000*
f=123456792.00*
f=123456792.00*
f=123456792.00*

d=12345678901234568.000000*
d=12345678901234568.00*
d=12345678901234568.00*
d=12345678901234568.00*
```

给出下面两个概念的结论：

- 1、在数据的有效位数超过精度时：  
按照精度输出对应位数的数据，输出的数据在精度之前的位数可信，在精度之后的位数不可信
- 2、如果指定的总宽度小于有效位数的宽度，则：  
按照有效位数的宽度输出



# §. 基础知识题 – C方式输入输出的格式化控制

## 1. 格式化输出函数printf的基本理解

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

```
#include <stdio.h>

#define str "abcdefghijklmnopqrstuvwxyz"

int main()
{
    printf("str=%s*\n", str);
    printf("str=%30s*\n", str);
    printf("str=%-30s*\n", str);
    printf("str=%5s*\n", str);
    printf("str=%-5s*\n", str);
    printf("str=%.5s*\n", str);
    printf("str=%-.5s*\n", str);
    printf("str=%10.5s*\n", str);
    printf("str=%-10.5s*\n", str);

    return 0;
}

//注：最后加*的目的，是为了看清是否有隐含空格
```

运行结果：

```
Microsoft Visual Studio 调试控制台
str=abcdefghijklmnopqrstuvwxyz*
str=      abcdefghijklmnopqrstuvwxyz*
str=abcdefghi jklmnopqrstuvwxyz   *
str=abcdefghi jklmnopqrstuvwxyz*
str=abcdefghi jklmnopqrstuvwxyz*
str=abcdefghi jklmnopqrstuvwxyz*
str=abcde*
str=abcde*
str=      abcde*
str=abcde   *
```

参考printf的格式控制符和附加格式控制符，给出解释：

%s : 输出字符串类型的数据

%30s : 输出字符串类型的数据，总宽度30，右对齐

%-30s: 输出字符串类型的数据，总宽度30，左对齐

如果指定的总宽度小于字符串的长度，则：按照实际字符串的长度输出字符串类型的数据，不受宽度限制

对%s而言，%m. n中n代表的位数是指前n个字符



# §. 基础知识题 – C方式输入输出的格式化控制

## 1. 格式化输出函数printf的基本理解

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

```
#include <stdio.h>

#define str "Student"
int main()
{
    int a = 65;
    printf("a=%o\n", a);
    printf("a=%x\n", a);
    printf("ch=%c\n", a);
    printf("s=%s\n\n", str);

    printf("a=0%o\n", a);
    printf("a=0x%x\n", a);
    printf("ch=\''%c\'\n", a);
    printf("s=\"%s\"\n\n", str);

    double d = 0.783;
    printf("百分比=%.2f%%\n", d * 100);

    return 0;
}
```

运行结果：

```
a=101
a=41
ch=A
s=Student

a=0101
a=0x41
ch='A'
s="Student"

百分比=78.30%
```

1、对比第1组和第2组输出，得出的结论是：

    格式控制符/附加格式控制符，只负责给出最原始数据的输出，若需要前导字符、单双引号等，需要自行加入

2、输出字符'%'的方法是：在格式控制表列中在对应位置添加%



# §. 基础知识题 – C方式输入输出的格式化控制

## 2. 格式化输入函数scanf的基本理解

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

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

int main()
{
    int a;
    scanf("%d", a);
    printf("a=%d\n", a);
    return 0;
}
```

在VS中编译：

```
warning C4477: “scanf”：格式字符串“%d”需要类型“int *”的参数，但可变参数 1 拥有了类型“int”
error C4700: 使用了未初始化的局部变量“a”
```

在Dev中编译：

假设键盘输入为：10  
则输出为： a=11739468

```
D:\Learning\高级语言程序设计\C-demo\C-demo\C-demo.exe
10
a=11739468
```

(经多次测试，发现每次输出a的值是一个随机数)

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

int main()
{
    int a = 0;
    scanf("%d", a);
    printf("a=%d\n", a);
    return 0;
}
```

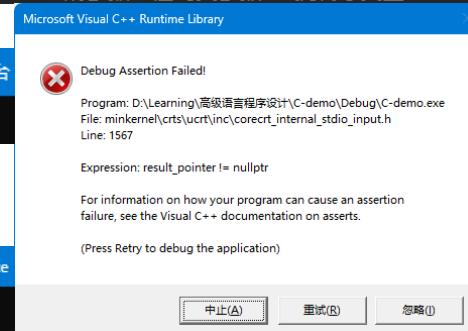
在VS中编译：

```
warning C4477: “scanf”：格式字符串“%d”需要类型“int *”的参数，但可变参数 1 拥有了类型“int”
```

假设键盘输入为：10  
则输出为： 无

在Dev中编译：  
假设键盘输入为：10  
则输出为： 无

```
D:\Learning\高级语言程序设计\C-demo\C-demo\C-demo.exe
10
```



结论：用scanf输入时，如果地址表列中直接跟变量名，则错误，其中VS的表现是显示warning C4477，或者弹出错误窗口，程序终止运行，Dev的表现是得到一个不可信的随机值，或者程序终止运行。



## §. 基础知识题 – C方式输入输出的格式化控制

### 2. 格式化输入函数scanf的基本理解

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

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

int main()
{
    int a, b;
    scanf("%d %d", &a, &b);
    printf("a=%d, b=%d\n", a, b);
    return 0;
}
```

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

int main()
{
    int a, b;
    scanf("%d%d", &a, &b); // %d间无空格
    printf("a=%d, b=%d\n", a, b);
    return 0;
}
```

假设键盘输入为: 10 15 ↵

则输出为:

假设键盘输入为: 10 ↵  
15 ↵

则输出为:

假设键盘输入为: 10 15 ↵

则输出为:

假设键盘输入为: 10 ↵  
15 ↵

则输出为:

结论:

多个输入时，格式控制符间是否有空格不影响正确性。



## §. 基础知识题 – C方式输入输出的格式化控制

### 2. 格式化输入函数scanf的基本理解

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

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

int main()
{
    int a=0, b=0;
    scanf("%d", &a, &b); //地址表列多
    printf("a=%d, b=%d\n", a, b);
    return 0;
}
```

warning C4474: scanf: 格式字符串中传递的参数太多  
message : 占位符和其参数预计 1 可变参数, 但提供的却是 2 参数

假设键盘输入为: 10 15↙

则输出为:

假设键盘输入为: 10↙

则输出为:

结论: 当地址表列的个数多于格式控制符时输入的值只能赋值给格式控制符数量的变量, 不能赋值给其余变量

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

int main()
{
    int a;
    scanf("%d %d", &a); //格式符多
    printf("a=%d\n", a);
    return 0;
}
```

VS:

warning C4473: “scanf”: 没有为格式字符串传递足够的参数  
message : 占位符和其参数预计 2 可变参数, 但提供的却是 1 参数  
message : 缺失的可变参数 2 为格式字符串 “%d” 所需

假设键盘输入为: 10 15↙

则输出为: 无输出, 程序终止运行

假设键盘输入为: 10↙  
15↙

则输出为: 无输出, 程序终止运行

结论:

当格式控制符的个数多于地址表列时VS会显示warning C4473, Dev会将输入的值赋值给地址表列数量的变量, 其余的格式控制符会被忽略

Dev:

假设键盘输入为: 10 15↙  
则输出为:

假设键盘输入为: 10↙  
15↙  
则输出为:



## §. 基础知识题 – C方式输入输出的格式化控制

### 2. 格式化输入函数scanf的基本理解

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

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

int main()
{
    int a, ret;
    ret = scanf("%d", &a);
    printf("a=%d, ret=%d\n", a, ret);
    return 0;
}
```

假设键盘输入为: 10 ↵

则输出为:

```
Microsoft Visual Studio 调试控制台
10
a=10, ret=1
```

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

int main()
{
    int a, b, ret;
    ret = scanf("%d %d", &a, &b);
    printf("a=%d, b=%d ret=%d\n", a, b, ret);
    return 0;
}
```

假设键盘输入为: 10 15 ↵

则输出为:

```
Microsoft Visual Studio 调试控制台
10 15
a=10, b=15 ret=2
```

结论: 在输入正确时, scanf的返回值是正确按指定格式输入变量的个数, 即能正确接收到值的变量个数



# §. 基础知识题 – C方式输入输出的格式化控制

## 2. 格式化输入函数scanf的基本理解

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

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

int main()
{
    int a, b;
    scanf("%d, %d", &a, &b);
    printf("a=%d, b=%d\n", a, b);
    return 0;
}
```

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

int main()
{
    int a, b;
    scanf("a=%d, b=%d", &a, &b);
    printf("a=%d, b=%d\n", a, b);
    return 0;
}
```

假设键盘输入为: 10 15

则输出为:

```
Microsoft Visual Studio 调试控制台
10 15
a=10, b=-858993460
```

假设键盘输入为: 10, 15

则输出为:

```
Microsoft Visual Studio 调试控制台
10, 15
a=10, b=15
```

假设键盘输入为: 10 15

则输出为:

Microsoft Visual Studio 调试控制台

```
10 15
a=-858993460, b=-858993460
```

Microsoft Visual Studio 调试控制台

```
10, 15
a=-858993460, b=-858993460
```

Microsoft Visual Studio 调试控制台

```
a=10, b=15
a=10, b=15
```

假设键盘输入为: a=10, b=15

则输出为:

结论: 当格式控制符中有其它字符(逗号, a=等)时, 对这些字符的输入方法是在格式控制符的位置输入内容, 其他对应位置依旧输入原来的字符



# §. 基础知识题 - C方式输入输出的格式化控制

## 2. 格式化输入函数scanf的基本理解

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

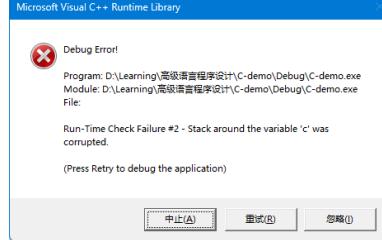
```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

int main()
{
    short c;

    scanf("%d", &c);
    printf("c=%hd\n", c);

    return 0;
}
```

假设键盘输入为: 10  
则输出为: (弹出错误窗口) —



```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

int main()
{
    0;
} int c;

scanf("%hd", &c);
printf("c=%d\n", c);

return
}
```

假设键盘输入为: 10  
则输出为:



warning C4477: "scanf": 格式字符串 "%hd" 需要类型 "short \*" 的参数, 但可变参数 1 拥有了类型 "int \*"  
message : 请考虑在格式字符串中使用 "%d"  
message : 请考虑在格式字符串中使用 "%I32d"  
message : 请考虑在格式字符串中使用 "%I32d"

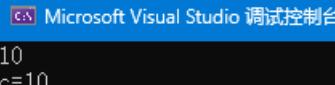
```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

int main()
{
    short c;

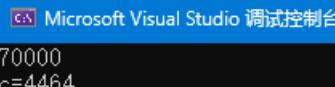
    scanf("%hd", &c);
    printf("c=%hd\n", c);

    return 0;
}
```

假设键盘输入为: 10  
则输出为:



假设键盘输入为: 70000  
则输出为:



结论:

1、附加格式控制符h的作用是输入短整型数

2、如果格式控制符的数据类型和要读取的变量类型的字节大小不一致(例: 4/2字节), 则显示warning C4477, 输出一个不可信的值, 或弹出错误窗口。



## §. 基础知识题 – C方式输入输出的格式化控制

### 2. 格式化输入函数scanf的基本理解

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

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

int main()
{
    int a, b, c;

    scanf("%d %x %o", &a, &b, &c);
    printf("a=%d, b=%d, c=%d\n", a, b, c);

    return 0;
}
```

假设键盘输入为: 10 11 12

则输出为:

```
Microsoft Visual Studio 调试控制台
10 11 12
a=10, b=17, c=10
```

假设键盘输入为: 12 ab 76

则输出为:

```
Microsoft Visual Studio 调试控制台
12 ab 76
a=12, b=171, c=62
```

假设键盘输入为: 10 -11 +12

则输出为:

```
Microsoft Visual Studio 调试控制台
10 -11 +12
a=10, b=-17, c=10
```

假设键盘输入为: 12 -ab +76

则输出为:

```
Microsoft Visual Studio 调试控制台
12 -ab +76
a=12, b=-171, c=62
```



## §. 基础知识题 – C方式输入输出的格式化控制

### 2. 格式化输入函数scanf的基本理解

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

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

int main()
{
    short a, b, c;

    scanf("%hd %hx %ho", &a, &b, &c);
    printf("a=%hd, b=%hd, c=%hd\n", a, b, c);

    return 0;
}
```

假设键盘输入为: 10 11 12 ↴

则输出为:

```
Microsoft Visual Studio 调试控制台
10 11 12
a=10, b=17, c=10
```

假设键盘输入为: 12 ab 76 ↴

则输出为:

```
Microsoft Visual Studio 调试控制台
12 ab 76
a=12, b=171, c=62
```

假设键盘输入为: 10 -11 +12 ↴

则输出为:

```
Microsoft Visual Studio 调试控制台
10 -11 +12
a=10, b=-17, c=10
```

假设键盘输入为: 12 -ab +76 ↴

则输出为:

```
Microsoft Visual Studio 调试控制台
12 -ab +76
a=12, b=-171, c=62
```



## §. 基础知识题 – C方式输入输出的格式化控制

### 2. 格式化输入函数scanf的基本理解

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

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

int main()
{
    int a;

    scanf("%3d", &a);
    printf("a=%d\n", a);

    return 0;
}
```

假设键盘输入为: 12345678↙

则输出为:

Microsoft Visual Studio 调试控制台  
12345678  
a=123

结论: %md中的m表示: 指定输入数据所占的宽度, 即输入到达该指定宽度时, 超过该指定宽度的内容将不再作为输入内容, 在该指定宽度内的内容为有效输入内容。

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

int main()
{
    int a, b;

    scanf("%3d %*2d %3d", &a, &b);
    printf("a=%d b=%d\n", a, b);

    return 0;
}
```

假设键盘输入为: 12345678↙

则输出为:

Microsoft Visual Studio 调试控制台  
12345678  
a=123 b=678

结论: \*md的\*m表示: 本输入项不赋给相应的变量



# §. 基础知识题 – C方式输入输出的格式化控制

## 2. 格式化输入函数scanf的基本理解

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

```
#define _CRT_SECURE_NO_WARNINGS  
#include <stdio.h>  
int main()  
{  
    int a;  
    scanf("%d", &a);  
    printf("%d\n", a);  
    return 0;  
}
```

假设键盘输入为:  
123↙  
则输出为:  
123  
123

假设键盘输入为:  
123 456↙  
则输出为:  
123 456  
123

假设键盘输入为:  
123a\*\*↙  
则输出为:  
123a\*\*  
123

```
#define _CRT_SECURE_NO_WARNINGS  
#include <stdio.h>  
int main()  
{  
    int a;  
    scanf("%x", &a);  
    printf("%d\n", a);  
    return 0;  
}
```

假设键盘输入为:  
123↙  
则输出为:  
123  
291

假设键盘输入为:  
123 456↙  
则输出为:  
123 456  
291

假设键盘输入为:  
123a\*\*↙  
则输出为:  
123a\*\*  
4666

```
#define _CRT_SECURE_NO_WARNINGS  
#include <stdio.h>  
int main()  
{  
    int a;  
    scanf("%3d", &a);  
    printf("%d\n", a);  
    return 0;  
}
```

假设键盘输入为:  
123↙  
则输出为:  
123  
123

假设键盘输入为:  
123a\*\*↙  
则输出为:  
123a\*\*  
123

假设键盘输入为:  
12a\*\*↙  
则输出为:  
12a\*\*  
12

结论:

scanf输入的终止条件是回车、空格、非法输入和达到指定输入数据所占的宽度



# §. 基础知识题 – C方式输入输出的格式化控制

## 2. 格式化输入函数scanf的基本理解

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

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
int main()
{
    int a, b;
    scanf("%3d%3d", &a, &b);
    printf("%d %d\n", a, b);
    return 0;
}
```

输入: 12↙ 345↙ , 输出:

输入: 12↙ 3456↙ , 输出:

输入: 123↙ 456↙ , 输出:

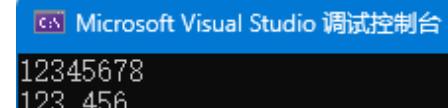
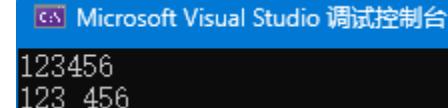
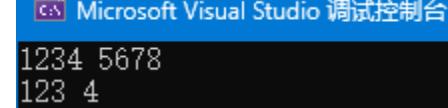
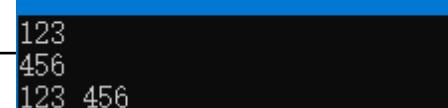
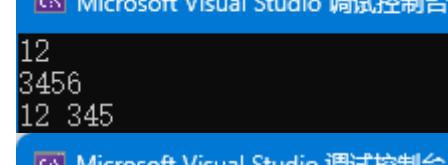
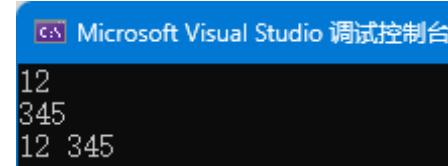
输入: 1234↙ 5678↙ , 输出:

输入: 123456↙ , 输出:

输入: 12345678↙ , 输出:

输出第4项的结果的原因:

首先读取123，达到指定输入数据所占的宽度3，终止输入，将123赋值给变量a，之后读取4，遇到空格，终止输入，将4赋值给变量b。



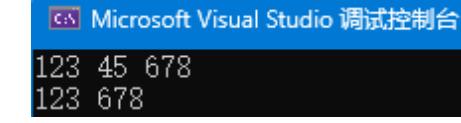
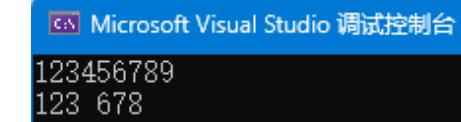
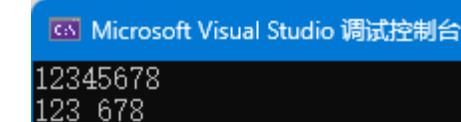
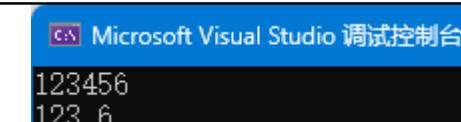
```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
int main()
{
    int a, b;
    scanf("%3d%*2d%3d", &a, &b);
    printf("%d %d\n", a, b);
    return 0;
}
```

输入: 123456↙ , 输出:

输入: 12345678↙ , 输出:

输入: 123456789↙ , 输出:

输入: 123 45 678↙ , 输出:





# §. 基础知识题 – C方式输入输出的格式化控制

## 2. 格式化输入函数scanf的基本理解

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

#define _CRT_SECURE_NO_WARNINGS #include <stdio.h>  int main() { float f;  scanf("%f", &f); printf("f=%f\n", f);  return 0; }	#define _CRT_SECURE_NO_WARNINGS #include <stdio.h>  int main() { float f;  scanf("%lf", &f); printf("f=%f\n", f);  return 0; }	#define _CRT_SECURE_NO_WARNINGS #include <stdio.h>  int main() { double f;  scanf("%lf", &f); printf("f=%f\n", f);  return 0; }	#define _CRT_SECURE_NO_WARNINGS #include <stdio.h>  int main() { double f;  scanf("%f", &f); printf("f=%f\n", f);  return 0; }
----------------------------------------------------------------------------------------------------------------------------------------------------------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------

假设键盘输入为:

123.45

则输出为:

```
Microsoft Visual Studio 调试控制台
123.45
f=123.449997
```

warning C4477: “scanf”：格式字符串“%lf”需要类型“double \*”的参数，但可变参数 1 拥有了类型“float \*”  
message : 请考虑在格式字符串中使用“%f”

假设键盘输入为:

123.45

则输出为: (弹出错误窗口)

```
D:\Learning\高级语言程序设计\C-demo\Debug\C-demo.exe
123.45
f=-107374184.000000
```

假设键盘输入为:

123.45

则输出为:

```
Microsoft Visual Studio 调试控制台
123.45
f=123.450000
```

warning C4477: “scanf”：格式字符串“%f”需要类型“float \*”的参数，但可变参数 1 拥有了类型“double \*”  
message : 请考虑在格式字符串中使用“%lf”

结论:

第二组程序的错误信息截图

第四组程序的错误信息截图

1、附加格式控制符1的作用是输入double型数

2、如果格式控制符的数据类型和要读取的变量类型的字节大小不一致（例：4/8字节），则赋值后输出的值不可信

3、printf中，输出double型数据时，%f 和 %lf 无差别  
scanf中， 输入double型数据时，%f 和 %lf 有差别



# §. 基础知识题 – C方式输入输出的格式化控制

## 2. 格式化输入函数scanf的基本理解

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

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
int main()
{
    float f;
    scanf("%7.2f", &f);
    printf("%f\n", f);
    return 0;
}
```

warning C4476: "scanf": 格式说明符中的类型字段字符“.”未知  
warning C4474: scanf: 格式字符串中传递的参数太多  
message : 占位符和其参数预计 0 可变参数, 但提供的却是 1 参数

假设键盘输入为: 1234. 56↙  
则输出为:

Microsoft Visual Studio 调试控制台  
1234.56  
-107374176.000000

假设键盘输入为: 12. 3456↙  
则输出为:

Microsoft Visual Studio 调试控制台  
12.3456  
-107374176.000000

假设键盘输入为: 123↙  
则输出为:

Microsoft Visual Studio 调试控制台  
123  
-107374176.000000

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
int main()
{
    float f;
    scanf("%7f", &f);
    printf("%f\n", f);
    return 0;
}
```

假设键盘输入为: 1234. 5678↙  
则输出为:

Microsoft Visual Studio 调试控制台  
1234.5678  
1234.560059

假设键盘输入为: 12. 345678↙  
则输出为:

Microsoft Visual Studio 调试控制台  
12.345678  
12.345600

假设键盘输入为: 12345678↙  
则输出为:

Microsoft Visual Studio 调试控制台  
12345678  
1234567.000000

结论:

1、%mf/%mlf如果指定了宽度m，则scanf在读取时只读取宽度为m的数据

2、%m. nf/%m. nlf如果指定了精度（小数点后的位数），则scanf的%f/%lf不支持.n形式的附加格式控制符，会得到不可信值



# §. 基础知识题 – C方式输入输出的格式化控制

## 2. 格式化输入函数scanf的基本理解

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

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
int main()
{
    char c1, c2;
    scanf("%c %c", &c1, &c2);
    printf("c1=%c c2=%c\n", c1, c2);
    return 0;
}
```

假设键盘输入为: ABCD↙  
则输出为:

假设键盘输入为: A BCD↙  
则输出为:

假设键盘输入为: 'A' BCD↙  
则输出为:

假设键盘输入为: \n↙  
则输出为:

结论: 1、%c只读1个字符  
2、%c在输入转义符/单引号等特殊字符时，得到的是特殊字符自身的ASCII码  
3、空格是scanf中%c方式的有效输入，但必须注意在格式控制表列中%c间无空格时，空格才会被视为有效输入，在格式控制表列中%c间有空格时，空格不会被视为有效输入

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
int main()
{
    char c1, c2;
    scanf("%c%c", &c1, &c2); //两个%c间无空格
    printf("c1=%d c2=%d\n", c1, c2);
    return 0;
}
```

假设键盘输入为: ABCD↙  
则输出为:

假设键盘输入为: A BCD↙  
则输出为:

假设键盘输入为: 'A' BCD↙  
则输出为:

假设键盘输入为: \n↙  
则输出为:



# §. 基础知识题 – C方式输入输出的格式化控制

## 2. 格式化输入函数scanf的基本理解

### 0. 观察下列程序的运行结果，回答问题并将程序的运行结果截图贴上(如果有错则贴错误信息截图)

#define _CRT_SECURE_NO_WARNINGS #include <stdio.h>  int main() { short ch;  scanf("%c", &ch); printf("ch=%hd\n", ch);  return 0; }	#define _CRT_SECURE_NO_WARNINGS #include <stdio.h>  int main() { int ch;  scanf("%c", &ch); printf("ch=%d\n", ch);  return 0; }	#define _CRT_SECURE_NO_WARNINGS #include <stdio.h>  int main() { long ch;  scanf("%c", &ch); printf("ch=%ld\n", ch);  return 0; }	#define _CRT_SECURE_NO_WARNINGS #include <stdio.h>  int main() { float ch;  scanf("%c", &ch); printf("ch=%f\n", ch);  return 0; }
假设键盘输入为: A↙ 则输出为: 	假设键盘输入为: A↙ 则输出为: 	假设键盘输入为: A↙ 则输出为: 	假设键盘输入为: A↙ 则输出为: 
第一组程序错误信息截图 	warning C4477: "scanf": 格式字符串 "%c" 需要类型 "char *" 的参数, 但可变参数 1 拥有了类型 "short *" message : 请考虑在格式字符串中使用 "%lc" message : 请考虑在格式字符串中使用 "%llc" message : 请考虑在格式字符串中使用 "%Lc" message : 请考虑在格式字符串中使用 "%wc"	warning C4477: "scanf": 格式字符串 "%c" 需要类型 "char *" 的参数, 但可变参数 1 拥有了类型 "int *" 	warning C4477: "scanf": 格式字符串 "%c" 需要类型 "char *" 的参数, 但可变参数 1 拥有了类型 "long *" 
第二组程序错误信息截图 	warning C4477: "scanf": 格式字符串 "%c" 需要类型 "char *" 的参数, 但可变参数 1 拥有了类型 "float *" 		结论: %c方式读入时, 地址表列 中的变量不能是非字符类型。



# §. 基础知识题 - C方式输入输出的格式化控制

## 2. 格式化输入函数scanf的基本理解

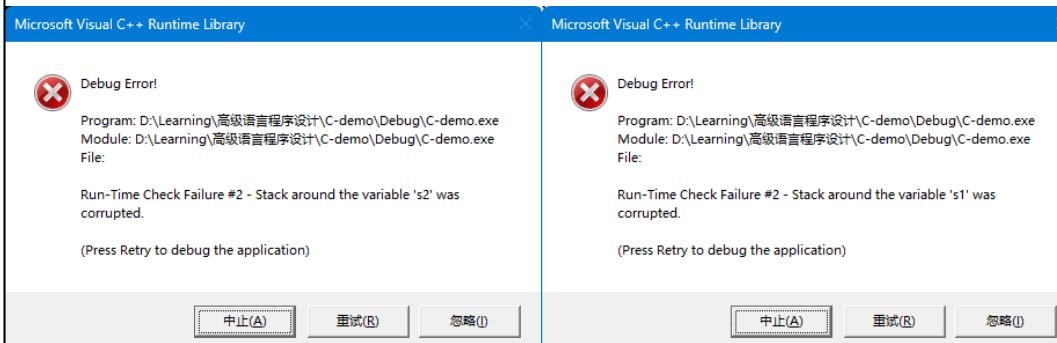
P. 观察下列程序的运行结果，回答问题并将程序的运行结果截图贴上(如果有错则贴错误信息截图)

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

int main()
{
    char s1[10], s2[10]; //s1/s2是数组(后续内容)

    scanf("%s %s", s1, s2);
    printf("s1=%s\ns2=%s\n", s1, s2);

    return 0;
}
```



假设键盘输入为: tong ji  
则输出为:

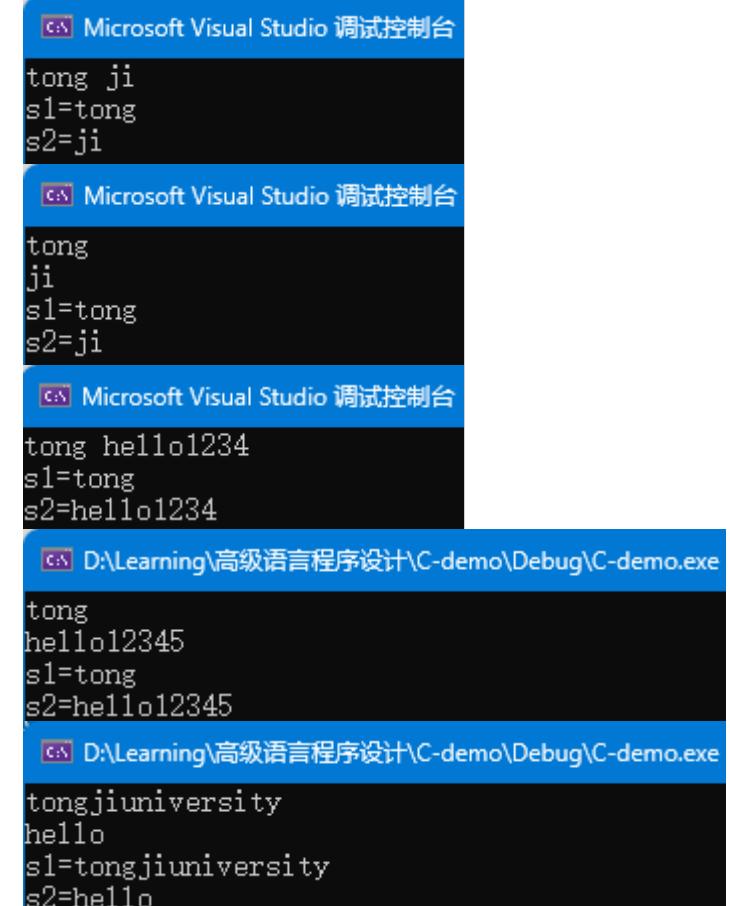
假设键盘输入为: tong  
ji  
则输出为:

假设键盘输入为: tong  
hello1234  
(9个字符)  
则输出为:

假设键盘输入为: tong  
hello12345  
(10个字符)  
则输出为:

假设键盘输入为: tongjiuniversity  
(超过10个)  
hello  
则输出为:

结论:  
1、%s不能读入含空格的字符串  
2、%s输入时，如果数组的大小为n，则最多输入n-1个字符



## §. 基础知识题 – C方式输入输出的格式化控制



## 2. 格式化输入函数scanf的基本理解

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

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
int main()
{
    char s[80];
    scanf("%s", s);
    printf("%s\n", s);
    return 0;
}
```

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
int main()
{
    char s[80], t[80];
    scanf("%s, %s", s, t);
    printf("s=%s\n", s);
    printf("t=%s\n", t);
    return 0;
}
```

假设键盘输入为：“\r\n\tabc”

则输出为： Microsoft Visual Studio 调试控制台

"\r\n\tabc"  
"\r\n\tabc"

该字符串真正的内存存储为10个字节，这些字节的值分别是92、114、92、110、92、116、97、98、99、0

假设键盘输入为： abc, def ↴

则输出为：

 Microsoft Visual Studio 调试控制台

abc, def

s=abc, def

t=烫烫烫烫

52-  
E

→ E

有效字

14

10 of 10

10 of 10

10 of 10

THEORY OF THE STATE

1

与2-E不同， "%s, %s"之间的逗号是当做第一个字符串的有效字符



## §. 基础知识题 – C方式输入输出的格式化控制

### 2. 格式化输入函数scanf的基本理解

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

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
int main()
{
    int a, ret;
    ret = scanf("%d", &a);
    printf("a=%d ret=%d\n", a, ret);
    return 0;
}
```

假设键盘输入为: 10  
则输出为:

假设键盘输入为: 10a  
则输出为:

假设键盘输入为: abc  
则输出为:

结论: scanf返回值是正确按指定格式输入变量的个数，即能正确接收到值的变量个数

```
Microsoft Visual Studio 调试控制台
10
a=10 ret=1
```

```
Microsoft Visual Studio 调试控制台
10 20
a=10 b=20 ret=2
```

```
Microsoft Visual Studio 调试控制台
10a20
a=10 b=-858993460 ret=1
```

```
Microsoft Visual Studio 调试控制台
abc
a=-858993460 b=-858993460 ret=0
```

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
int main()
{
    int a, b, ret;
    ret = scanf("%d %d", &a, &b);
    printf("a=%d b=%d ret=%d\n", a, b, ret);
    return 0;
}
```

假设键盘输入为: 10 20  
则输出为:

假设键盘输入为: 10 20a  
则输出为:

假设键盘输入为: 10a20  
则输出为:

假设键盘输入为: abc  
则输出为:



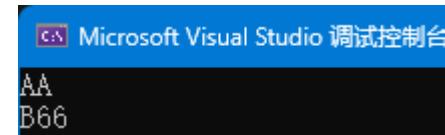
## §. 基础知识题 – 字符的输入与输出

### 1. putchar的基本使用

A. 程序如下，观察编译及运行结果

```
#include <iostream>
#include <cstdio>
using namespace std;
int main()
{
    char ret1;
    cout << (ret1 = putchar('A')) << endl;
    int ret2;
    cout << (ret2 = putchar('B')) << endl;
    return 0;
}
```

1、观察运行结果



2、分析运行结果中各输出是哪个语句/函数造成的第一行：

第一个A: putchar  
第二个A: cout

第二行：

B: putchar  
66: cout

3、这个例子能确认上个Page的基本知识中的说法：  
“返回值是int型，是输出字符的ASCII码”

完全正确/部分正确吗？

不能。putchar的返回值被赋值给char型变量ret1或int型变量ret2，cout根据数据类型决定输出形式，所以我们不能确认“返回值是int型，是输出字符的ASCII码”是完全正确/部分正确的。



## §. 基础知识题 – 字符的输入与输出

### 1. putchar的基本使用

B. 自行构造测试程序，证明putchar的返回值是int型而不是char型

//方法一

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

int main()
{
    cout << typeid(putchar('A')).name()
        << endl;
}
```

//方法2

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

int main()
{
    cout << sizeof(putchar('A')) << endl;
}
```

Microsoft Visual Studio 调试控制台

int

Microsoft Visual Studio 调试控制台

4



## §. 基础知识题 – 字符的输入与输出

### 2. getchar的基本使用

A. 程序如下，观察编译及运行结果（可手填）

```
#include <iostream>
#include <cstdio>
using namespace std;
int main()
{
    char ch;
    ch = getchar();
    cout << ch << endl;

    return 0;
}
```

```
#include <iostream>
#include <cstdio>
using namespace std;
int main()
{
    char ch;
    cout << (ch = getchar()) << endl;

    return 0;
}
```

```
#include <iostream>
#include <cstdio>
using namespace std;
int main()
{
    int ch;
    ch = getchar();
    cout << ch << endl;

    return 0;
}
```

输入: a↙

输出: a

输出的是: ch的值

输入: a↙

输出: a

输出的是: 赋值表达式值

输入: a↙

输出: 97



## §. 基础知识题 – 字符的输入与输出

### 2. getchar的基本使用

B. 自行构造测试程序，证明getchar的返回值是int型而不是char型

//方法一

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

int main()
{
    cout << typeid(getchar()).name()
        << endl;
}
```

//方法2

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

int main()
{
    cout << sizeof(getchar()) << endl;
}
```

Microsoft Visual Studio 调试控制台

int

Microsoft Visual Studio 调试控制台

4



## §. 基础知识题 – 字符的输入与输出

### 2. getchar的基本使用

C. 程序如下，观察编译及运行结果

```
#include <iostream>
#include <cstdio>
using namespace std;
int main()
{
    char ch;
    ch = getchar();
    cout << int(ch) << endl;

    return 0;
}
```

- 1、键盘输入: Hello↙ (5个字母+回车)
- 2、键盘输入: ↵ (空回车)
- 3、键盘输入: ↵ (空格+回车)
- 4、键盘输入: \n↙ (2个字符+回车)
- 5、键盘输入: \101↙ (4个字符+回车)

The screenshot shows five separate instances of the Microsoft Visual Studio Debug Console. Each instance displays the input entered and its corresponding ASCII value.

- Input: Hello↙, Output: Hello 72
- Input: ↵, Output: 10
- Input: ↵, Output: 32
- Input: \n↙, Output: \n 92
- Input: \101↙, Output: \101 92

结论:

可以输入空格、回车等cin无法处理的非图形字符，但仍不能处理转义符。



# §. 基础知识题 - 字符的输入与输出

## 2. getchar的基本使用

### D. 程序如下，观察编译及运行结果

```
#include <iostream>
#include <cstdio>
using namespace std;
int main()
{
    cout << "--Step1--" << endl;
    cout << getchar() << endl;

    cout << "--Step2--" << endl;
    cout << getchar() << endl;

    cout << "--Step3--" << endl;
    cout << getchar() << endl;

    cout << "--Step4--" << endl;
    cout << getchar() << endl;
    return 0;
}
```

#### 1、每次输入一个回车

程序从开始执行到结束，共停顿了4次来等待输入  
第1次停顿时，屏幕上输出的最后一行是Step1  
第2次停顿时，屏幕上输出的最后一行是Step2  
第3次停顿时，屏幕上输出的最后一行是Step3  
第4次停顿时，屏幕上输出的最后一行是Step4

#### 2、第一次输入一个字母+回车，以后每次停顿，均输入一个字母+回车

程序从开始执行到结束，共停顿了2次来等待输入  
第1次停顿时，屏幕上输出的最后一行是Step1  
第2次停顿时，屏幕上输出的最后一行是Step3  
第3次停顿时，屏幕上输出的最后一行是Step(没有停顿)  
第4次停顿时，屏幕上输出的最后一行是Step(没有停顿)

#### 3、第一次即输入4个以上的字母+回车

程序从开始执行到结束，共停顿了1次来等待输入  
第1次停顿时，屏幕上输出的最后一行是Step1  
第2次停顿时，屏幕上输出的最后一行是Step(没有停顿)  
第3次停顿时，屏幕上输出的最后一行是Step(没有停顿)  
第4次停顿时，屏幕上输出的最后一行是Step(没有停顿)

结论：getchar每次仅从输入缓冲区中取需要的字节，  
多余的字节仍保留在输入缓冲区中供下次读取

思考：结合“cin与cout的基本使用”中3.c的例子，考虑一下3.c中非法m对int的影响（错在第几个数）与输入缓冲区的关系，为什么？  
在输入缓冲区中，非法m之前的内容可以正常读取，其值是可信的，非法m之后的内容读取后，其值是不可信的。

```
--Step1--  
10  
--Step2--  
10  
--Step3--  
10  
--Step4--  
10  
  
--Step1--  
a  
97  
--Step2--  
10  
--Step3--  
b  
98  
--Step4--  
10  
  
--Step1--  
abcde  
97  
--Step2--  
98  
--Step3--  
99  
--Step4--  
100
```



## §. 基础知识题 – 字符的输入与输出

### 2. getchar的基本使用

#### E. 自行构造证明D结论的使用cin读入的测试程序

```
#include <iostream>
using namespace std;
int main()
{
    char a, b, c, d;

    cout << "--Step1--" << endl;
    cin >> a;
    cout << a << endl;
    cout << "--Step2--" << endl;
    cin >> b;
    cout << b << endl;
    cout << "--Step3--" << endl;
    cin >> c;
    cout << c << endl;
    cout << "--Step4--" << endl;
    cin >> d;
    cout << d << endl;

    return 0;
}
```

因为cin不能读取空格、回车（有特殊方法可读，先忽略），因此测试有所不同

1、第一次输入两个字母+回车，以后每次停顿，均输入两个字母+回车

程序从开始执行到结束，共停顿了2 次来等待输入

第1次停顿时，屏幕上输出的最后一行是Step1

第2次停顿时，屏幕上输出的最后一行是Step3

第3次停顿时，屏幕上输出的最后一行是Step(没有停顿)

第4次停顿时，屏幕上输出的最后一行是Step(没有停顿)

2、第一次即输入4个以上的字母+回车

程序从开始执行到结束，共停顿了1 次来等待输入

第1次停顿时，屏幕上输出的最后一行是Step1

第2次停顿时，屏幕上输出的最后一行是Step(没有停顿)

第3次停顿时，屏幕上输出的最后一行是Step(没有停顿)

第4次停顿时，屏幕上输出的最后一行是Step(没有停顿)

结论：cin每次仅从输入缓冲区中取需要的字节，多余的字节仍保留在输入缓冲区中供下次读取



## §. 基础知识题 – 字符的输入与输出

### 3. getchar、\_getch与\_getche的基本使用

A. 程序如下，观察编译及运行结果

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

int main()
{
    char ch;
    ch = getchar();
    cout << (int)ch << endl;

    return 0;
}
```

1、输入: a↙

输出: 97

Microsoft Visual Studio 调试控制台

a  
97

输入回显: 有

按回车生效: 是

2、输入: ↴

输出: 10

Microsoft Visual Studio 调试控制台

10

```
#include <iostream>
#include <conio.h>
using namespace std;

int main()
{
    char ch;
    ch = _getch();
    cout << (int)ch << endl;

    return 0;
}
```

1、输入: a↙

输出: 97

Microsoft Visual Studio 调试控制台

97

输入回显: 无

按回车生效: 否

2、输入: ↴

输出: 13

Microsoft Visual Studio 调试控制台

13

- 1、测试时cmd窗口下面不能是中文输入法
- 2、<conio.h>是\_getch()/\_getche()需要的头文件

```
#include <iostream>
#include <conio.h>
using namespace std;

int main()
{
    char ch;
    ch = _getche();
    cout << (int)ch << endl;

    return 0;
}
```

1、输入: a↙

输出: a97

Microsoft Visual Studio 调试控制台

a97

输入回显: 有

按回车生效: 否

2、输入: ↴

输出: 13

Microsoft Visual Studio 调试控制台

13

VS与Dev  
编译结  
果一致



## §. 基础知识题 – 字符的输入与输出

### 3. getchar、\_getch与\_getche的基本使用

B. 程序如下，观察编译及运行结果

- 1、测试时cmd窗口下面不能是中文输入法  
2、<conio.h>是\_getch()/\_getche()需要的头文件

哪个编译器报错？VS

```
#include <iostream>
#include <conio.h>
using namespace std;

int main()
{
    char ch;
    ch = getch();
    cout << (int)ch << endl;

    return 0;
}
```

```
#include <iostream>
#include <conio.h>
using namespace std;

int main()
{
    char ch;
    ch = getche();
    cout << (int)ch << endl;

    return 0;
}
```

error C4996: 'getch': The POSIX name for this item is deprecated. Instead, use the ISO C and C++ conformant name: \_getch. See online help for details.  
error C4996: 'getche': The POSIX name for this item is deprecated. Instead, use the ISO C and C++ conformant name: \_getche. See online help for details.

哪个编译器下结果同A？Dev

1、输入：a↙  
输出：97  
输入回显：无  
按回车生效：否

D:\Learning\高级语言程序设计\My Project\My Project\My Project.exe

97

2、输入：↙  
输出：13

D:\Learning\高级语言程序设计\My Project\My Project\My Project.exe

13

1、输入：a↙  
输出：a97  
输入回显：有  
按回车生效：否

D:\Learning\高级语言程序设计\My Project\My Project\My Project.exe

a97

2、输入：↙  
输出：13

D:\Learning\高级语言程序设计\My Project\My Project\My Project.exe

13

13

## §. 基础知识题 – 字符数组的输入与输出



## 1. 输入

逐个输入： `scanf("%c", &数组元素)` C方式  
`cin >> 数组元素` C++方式

## 例1：C方式输入单个字符

```
#define _CRT_SECURE_NO_WARNINGS //VS需要
#include <iostream>
using namespace std;

int main()
{
    char a[10];
    int i;

    for(i=0; i<10; i++)
        cout << int(a[i]) << endl;

    scanf("%c%c", &a[3], &a[7]);

    for(i=0; i<10; i++)
        cout << int(a[i]) << endl;

    return 0;
}
```

数组下标表示前有  
取地址符号&  
因为scanf规定后面  
必须是变量的地址

scanf前首先输出10行，内容是：  
-52 (随机)  
-52 (随机)

scanf时，输入AB并回车，输出是：  
-52 (随机)  
-52 (随机)  
-52 (随机)  
**65**  
-52 (随机)  
-52 (随机)  
-52 (随机)  
**66**  
-52 (随机)  
-52 (随机)

//用不同颜色标注出有变化的内容

# §. 基础知识题 - 字符数组的输入与输出



## 1. 输入

逐个输入: `scanf("%c", &数组元素)`    C方式  
`cin >> 数组元素`    C++方式

例2: C++方式输入单个字符

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

int main()
{
    char a[10];
    int i;

    for(i=0; i<10; i++)
        cout << int(a[i]) << endl;

    cin >> a[3] >> a[7];

    for(i=0; i<10; i++)
        cout << int(a[i]) << endl;

    return 0;
}
```

数组下标表示前  
无取地址符号&

cin前首先输出10行, 内容是

-52 (随机)	-52

cin时, 输入AB并回车, 输出是:

-52 (随机)	-52
-52 (随机)	-52
-52 (随机)	-52
<b>65</b>	AB
-52 (随机)	-52
-52 (随机)	-52
-52 (随机)	65
<b>66</b>	-52
-52 (随机)	-52
-52 (随机)	66

//用不同颜色标注出有变化的内容



## §. 基础知识题 - 字符数组的输入与输出

### 1. 输入

逐个输入: `scanf("%c", &数组元素)`    C方式  
`cin >> 数组元素`    C++方式

例3: C方式多次逐个输入时回车的处理

```
#define _CRT_SECURE_NO_WARNINGS //VS需要
#include <iostream>
using namespace std;

int main()
{
    char a[10];
    int i;

    for(i=0; i<10; i++)
        cout << int(a[i]) << endl;

    scanf("%c%c", &a[3], &a[7]);
    scanf("%c", &a[0]);

    for(i=0; i<10; i++)
        cout << int(a[i]) << endl;

    return 0;
}
```

scanf前首先输出10行, 内容是

-52 (随机)	-52

scanf时, 输入AB并回车, 输出是:

10	-52
-52 (随机)	-52
-52 (随机)	-52
65	AB
-52 (随机)	10
-52 (随机)	-52
-52 (随机)	-52
66	65
-52 (随机)	-52
-52 (随机)	-52

//用不同颜色标注出有变化的内容

## §. 基础知识题 – 字符数组的输入与输出



## 1. 输入

逐个输入： `scanf("%c", &数组元素)` C方式  
`cin >> 数组元素` C++方式

例4：C++方式多次逐个输入时回车的处理

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

int main()
{
    char a[10];
    int i;

    for(i=0; i<10; i++)
        cout << int(a[i]) << endl;

    cin >> a[3] >> a[7];
    cin >> a[0];

    for(i=0; i<10; i++)
        cout << int(a[i]) << endl;

    return 0;
}
```

cin前首先输出10行，内容是

cin时，输入AB并回车，表现如何？

光标闪烁，正在等待读入字符。

多按几次回车，表现如何？

光标下移并闪烁，正在等待读入字符。

最后再输入C并回车，则输出是：

67  
-52 (随机)  
-52 (随机)  
**65**  
-52 (随机)  
-52 (随机)  
-52 (随机)  
**66**  
-52 (随机)  
-52 (随机)

//用不同颜色标注出有变化的内容

综合例3/4得到结论：当多次逐个输入时，

C方式处理回车的方式是：

在`scanf("%c", &char)`情况下将回车符作为合法的输入字符在输入缓冲区中被读取

C++方式处理回车的方式是：

回车符作为cin输入的终止条件（不作为合法的输入字符）不会在输入缓冲区中被读取

## §. 基础知识题 – 字符数组的输入与输出



## 1. 输入

字符串形式: `scanf("%s", 数组名)` C方式  
`cin >> 数组名` C++方式

例5：C方式输入字符串（正确）

```
#define _CRT_SECURE_NO_WARNINGS //VS需要
#include <iostream>
using namespace std;

int main()
{
    char a[10];
    int i;
    for(i=0; i<10; i++)
        cout << int(a[i]) << endl;
    scanf("%s", a);
    for(i=0; i<10; i++)
        cout << int(a[i]) << endl;
    return 0;
}
```

直接数组名，  
也不加&  
因为C/C++规定  
代表数组的起始地址

直接数组名，无下标，  
也不加&  
因为C/C++规定，数组名  
代表数组的起始地址

scanf前首先输出10行， 内容是

等待键盘输入，输入Hello并回车，  
输出为

72  
101  
108  
108  
111  
2

-52 (随机)  
-52 (随机)  
-52 (随机)  
-52 (随机)

//用不同颜色标注出有变化的内容

He111  
72  
101  
108  
108  
111  
0  
-52  
-52  
-52  
-52

问：

1、回车是否在数组中？

回车不在数组中。

2、Hello后面的一个字符是什么？

ASCII码为0的字符，即尾零(' \0')。

# §. 基础知识题 - 字符数组的输入与输出



## 1. 输入

字符串形式: `scanf("%s", 数组名)` C方式  
`cin >> 数组名` C++方式

### 例6: C方式输入字符串(错误)

```
#define _CRT_SECURE_NO_WARNINGS //VS需要
#include <iostream>
using namespace std;

int main()
{
    char a[10];
    int i;

    for(i=0; i<10; i++)
        cout << int(a[i]) << endl;

    scanf("%s", a);

    for(i=0; i<10; i++)
        cout << int(a[i]) << endl;

    return 0;
}
```

直接数组名, 无下标,  
也不加&  
因为C/C++规定, 数组名  
代表数组的起始地址

scanf前首先输出10行, 内容是  
-52 (随机)  
等待键盘输入:  
测试1: 输入9个及以内字  
符并回车, 输出? (以输  
入12345678为例)  
49  
50  
51  
52  
53  
54  
55  
56  
57  
12345678  
49  
50  
51  
52  
53  
54  
55  
56  
0  
-52 (随机)

测试2: 输入10个及以上字符并回车, 输出?  
(以输入0123456789为例)

Microsoft Visual C++ Runtime Library  
Debug Error  
Program: D:\Learning\高级语言程序设计\高级语言程序设计\Debug\C++  
Demo.exe  
Module: D:\Learning\高级语言程序设计\高级语言程序设计\Debug\C++  
Demo.exe  
File:  
Run-Time Check Failure #2 - Stack around the variable 'a' was  
corrupted.  
(Press Retry to debug the application)  
中止(A) 重试(B) 忽略(I)  
0123456789  
48  
49  
50  
51  
52  
53  
54  
55  
56  
57  
输入10个及以上字符并  
回车, 出现弹窗报错  
问: 如果要保证输入正确, 输入  
的字符个数要小于定义的  
字符数组的长度

## §. 基础知识题 – 字符数组的输入与输出



## 1. 输入

字符串形式: `scanf("%s", 数组名)` C方式  
`cin >> 数组名` C++方式

例7：C++方式输入字符串（正确）

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

int main()
{
    char a[10];
    int i;
    cout << int(a[i]) << endl;
    cin >> a;
    for(i=0; i<10; i++)
        cout << int(a[i]) << endl;
    return 0;
}
```

直接数组名, 无下标  
也不加&

直接数组名, 无下标  
也不加&

cin前首先输出10行， 内容是

等待键盘输入，输入Hello并回车，  
输出为

72  
101  
108  
108  
111  
0

-52 (随机)  
-52 (随机)  
-52 (随机)  
-52 (随机)  
//用不同颜色标注出有变化的内容

Hello  
72  
101  
108  
108  
111  
0  
-52  
-52  
-52  
-52

问：

1、回车是否在数组中？

回车不在数组中。

2、Hello后面的一个字符是什么？

ASCII码为0的字符，即尾零(' \0')。



# §. 基础知识题 - 字符数组的输入与输出

## 1. 输入

字符串形式: `scanf("%s", 数组名)`      C方式  
`cin >> 数组名`      C++方式

例8: C++方式输入字符串(错误)

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

int main()
{
    char a[10];
    int i;

    for(i=0; i<10; i++)
        cout << int(a[i]) << endl;

    cin >> a;

    for(i=0; i<10; i++)
        cout << int(a[i]) << endl;

    return 0;
}
```

直接数组名, 无下标,  
也不加&

cin前首先输出10行, 内容是

-52 (随机)  
-52 (随机)

等待键盘输入:

测试1: 输入9个及以内字符并回车, 输出? (以输入12345678为例)

49  
50  
51  
52  
53  
54  
55  
56  
57  
12345678  
49  
50  
51  
52  
53  
54  
55  
56  
0  
-52 (随机)

输入9个及以内字符并回车, 依次输出字符的ASCII码和一个尾零('0'), 若有未初始化的元素则输出随机值

测试2: 输入10个及以上字符并回车, 输出?  
(以输入0123456789为例)

48  
49  
50  
51  
52  
53  
54  
55  
56  
57  
0123456789  
48  
49  
50  
51  
52  
53  
54  
55  
56  
57

输入10个及以上字符并回车, 出现弹窗报错

问: 如果要保证输入正确, 输入的字符个数要小于定义的字符数组的长度

# §. 基础知识题 - 字符数组的输入与输出



## 2. 输出

逐个: `printf("%c", 数组元素)`      C方式  
`cout << 数组元素`      C++方式

例9: C/C++方式输出单个字符

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

int main()
{
    char a[]="Student"; //长度缺省为8

    cout << sizeof(a) << endl;

    printf("%c*\n", a[5]);

    cout << a[3] << '*' << endl;

    return 0;
} //输出加*是为了确认只输出了一个字符
```

输出为:

Microsoft Visual Studio 调试控制台  
8  
n\*  
d\*



## §. 基础知识题 - 字符数组的输入与输出

### 2. 输出

逐个: `printf("%c", 数组元素)`      C方式  
`cout << 数组元素`      C++方式

例10: C/C++方式以单个字符+循环形式输出整个数组

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

int main()
{
    int i;
    char a[]="Student";

    for(i=0; i<7; i++)
        printf("%c", a[i]);
    cout << endl; //换行

    for(i=0; i<7; i++)
        cout << a[i];
    cout << endl; //换行

    return 0;
}
```

数组 a 缺省长度为8  
输出[0]-[6], 尾零不输出

输出为:

```
Microsoft Visual Studio 调试控制台
Student
Student
Student
```



## §. 基础知识题 - 字符数组的输入与输出

### 2. 输出

逐个: `printf("%c", 数组元素)`      C方式  
`cout << 数组元素`      C++方式

例11: C/C++方式以单个字符+循环形式输出整个数组

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

int main()
{
    int i;
    char a[]="Student";

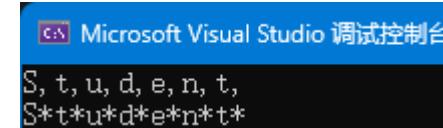
    for(i=0; i<7; i++)
        printf("%c, ", a[i]);
    cout << endl; //换行

    for(i=0; i<7; i++)
        cout << a[i] << '*';
    cout << endl; //换行

    return 0;
}
```

%c后面多一个,  
cout方式每个字符  
后面多一个\*

输出为:  
S, t, u, d, e, n, t,  
S\*t\*u\*d\*e\*n\*t\*



# §. 基础知识题 - 字符数组的输入与输出



## 2. 输出

字符串形式: `printf("%s", 数组名)` C方式  
`cout << 数组名` C++方式

例12: C/C++以字符串方式输出字符数组

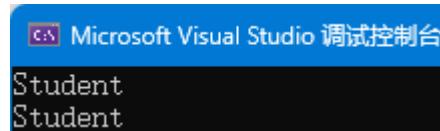
```
#include <iostream>
using namespace std;

int main()
{
    char a[] = "Student";
    printf("%s\n", a);
    cout << a << endl;

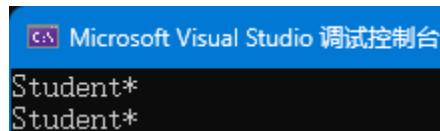
    return 0;
}
```

跟数组名  
不是数组元素名

输出为:  
Student  
Student



问:  
尾零输出了吗? 如何证明?  
  
尾零没有输出, 可以通过右侧程序的输出结果来证明尾零没有输出。右侧程序的输出结果为:



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

int main()
{
    char a[] = "Student";
    printf("%s*\n", a);
    cout << a << "*" << endl;

    return 0;
}
```

# §. 基础知识题 - 字符数组的输入与输出



## 2. 输出

字符串形式: `printf("%s", 数组名)` C方式  
`cout << 数组名` C++方式

例13: C/C++以字符串方式输出字符数组

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

int main()
{
    char a[]="Student\0china";

    cout << sizeof(a) << endl;

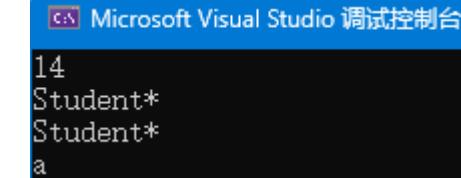
    printf("%s*\n", a);
    cout << a << '*' << endl;

    cout << a[12] << endl;

    return 0;
}
```

输出为:

14  
Student\*  
Student\*  
a



问1: 从本例的结果可知, 数组a的长度是14。  
最后是否还有隐含的\0? 是。  
a中的字符串的长度是7。

问2: 字符串形式输出字符数组, 如果数组中包含显式'\0',  
则输出到第一个显式'\0'(不包括'\0')为止。

# §. 基础知识题 - 字符数组的输入与输出



## 2. 输出

字符串形式: `printf("%s", 数组名)` C方式  
`cout << 数组名` C++方式

例14: C/C++以字符串方式输出字符数组(不含尾零)

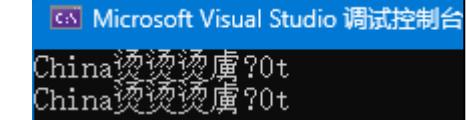
```
#include <iostream>
using namespace std;

int main()
{
    //注意: 不能以字符串方式初始化
    char a[5]={'C', 'h', 'i', 'n', 'a'};

    printf("%s\n", a);
    cout << a << endl;

    return 0;
}
```

输出为: (每次输出表现随机)  
China烫烫烫膚?0t  
China烫烫烫膚?0t



问1: 为什么会有乱字符?

输出跟数组名时字符串方式输出(从数组的起始地址开始依次输出各字符的值, 到\0为止), 数组a是不含\0的数组, 当字符串方式输出不含\0的数组时, 会持续越界输出直到碰见\0为止, 错误的具体表现为正常/乱码等多种形式。

问2: 如果%s方式换成下面形式, 还会看到乱字符吗? 为什么?

```
int i;
for (i=0; i<5; i++)
    printf("%c", a[i]);
```

不会看到乱字符, 因为依次输出字符数组中的每一个字符不是以字符串方式输出, 不会出现越界的情况。

# §. 基础知识题 - 字符数组的输入与输出



## 2. 输出

字符串形式: `printf("%s", 数组名)` C方式  
`cout << 数组名` C++方式

例15: C/C++以字符串方式输出字符数组(不含尾零)

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

int main()
{
    char a[5]; //不初始化

    printf("%s\n", a);
    cout << a << endl;

    return 0;
}
```

输出为: (每次输出表现随机)  
烫烫烫烫烫烫烫 IX, 鴨  
烫烫烫烫烫烫烫 IX, 鴨



问1: 为什么会有乱字符?

char型数组a未进行初始化, 当数组为自动变量时, 数组元素的值不确定, 输出跟数组名时字符串方式输出, 即从数组的起始地址开始依次输出各字符的值, 到\0为止, 由于数组元素的值不确定, 所以会持续随机输出直到碰见\0为止, 错误的具体表现为正常/乱码等多种形式。

问2: 乱字符出现几行是正常的? 一行? 多行? 或者都正常?  
都正常。

结论: 不能字符串形式输出不含'\0'的字符数组, 否则可能会得到不正确的结果。

# §. 基础知识题 – 字符数组的输入与输出



## 3. 从任一元素开始以字符串形式输入/输出

### 例16：从任一元素开始以字符串形式输出

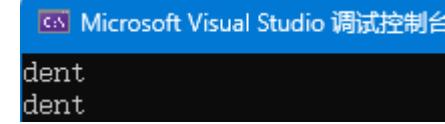
```
#include <iostream>
using namespace std;

int main()
{
    char a[]="Student";
    printf("%s\n", &a[3]);
    cout << &a[3] << endl;
    return 0;
}
```

%s形式

&数组元素名形式

输出为：  
dent  
dent



Microsoft Visual Studio 调试控制台

dent  
dent



## §. 基础知识题 - 字符数组的输入与输出

### 3. 从任一元素开始以字符串形式输入/输出

例17：C方式从任一元素开始以字符串形式输入

```
#define _CRT_SECURE_NO_WARNINGS //VS需要
#include <iostream>
using namespace std;

int main()
{
    int i;
    char a[10];

    for(i=0; i<10; i++)
        cout << int(a[i]) << endl;

    scanf("%s", &a[3]);

    for(i=0; i<10; i++)
        cout << int(a[i]) << endl;

    return 0;
}
```

&数组元素名形式

scanf先输出10行，内容是

-52 (随机)	-52	-52
Hello	72	101
-52	108	108
-52	111	111
72	0	0
101	-52	-52
108	72	72
108	101	101
111	108	108
0	111	111
-52 (随机)	0	0

等待键盘输入，输入Hello并回车，  
输出为

//用不同颜色标注出有变化的内容

# §. 基础知识题 - 字符数组的输入与输出



## 3. 从任一元素开始以字符串形式输入/输出

### 例18：C++方式从任一元素开始以字符串形式输入

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

int main()
{
    int i;
    char a[10];

    for(i=0; i<10; i++)
        cout << int(a[i]) << endl;

    cin >> &a[3];

    for(i=0; i<10; i++)
        cout << int(a[i]) << endl;

    return 0;
}
```

&数组元素名形式

cin先输出10行，内容是

-52 (随机)	-52	-52

等待键盘输入，输入Hello并回车，  
输出为

-52 (随机)	72	101
-52 (随机)	108	108
-52 (随机)	111	111
72	0	-52
101		
108		
108		
111		
0		
-52 (随机)		

//用不同颜色标注出有变化的内容

综合例16-18的结果，得出的结论是：

C/C++方式从任一元素开始以字符串形式输入输出时，表示形式都是  
&数组元素名的形式

# §. 基础知识题 - 字符数组的输入与输出



## 1-3. 总结

	C方式	C++方式
输入单个字符	scanf("%c", &元素名)	cin >> 元素名
输入字符串	scanf("%s", 数组名)	cin >> 数组名
输出单个字符	printf("%c", 元素名)	cout << 元素名
输出字符串	printf("%s", 数组名)	cout << 数组名
任一元素开始输入串	scanf("%s", &元素名)	cin >> &元素名
任一元素开始输出串	printf("%s", &元素名)	cout << &元素名



## §. 基础知识题 - 字符数组的输入与输出

### 4. 多个字符串的输入

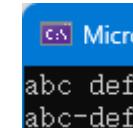
例19：C方式多个字符串的输入

```
#define _CRT_SECURE_NO_WARNINGS //VS需要
#include <iostream>
#include <cstdio>
using namespace std;

int main()
{
    char a[10], b[20];
    scanf("%s%s", a, b);
    printf("%s-%s\n", a, b);
    return 0;
}
```

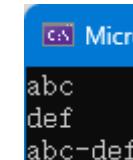
1、假设输入为abc空格def并回车

则输出为：  
abc-def



2、假设输入为abc回车  
def回车

则输出为：  
abc-def



结论：空格是输入分隔符。



## §. 基础知识题 – 字符数组的输入与输出

### 4. 多个字符串的输入

例20：C++方式多个字符串的输入

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

int main()
{
    char a[10], b[20];

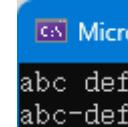
    cin >> a >> b;

    cout << a << '-' << b << endl;

    return 0;
}
```

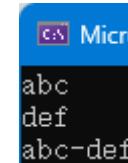
1、假设输入为abc空格def并回车

则输出为：  
abc-def



2、假设输入为abc回车  
def回车

则输出为：  
abc-def



结论：空格是输入分隔符。

综合例19-20可知：  
scanf/cin从键盘上输入的字符串  
不能包含空格和回车等非图形字符。



## §. 基础知识题 – 字符数组的输入与输出

### 4. 多个字符串的输入

★ 从键盘输入含空格字符串的方法(不同编译器不同)

例21：VS下用gets\_s输入含空格的字符串（VS2022：有gets\_s，无gets，有fgets）

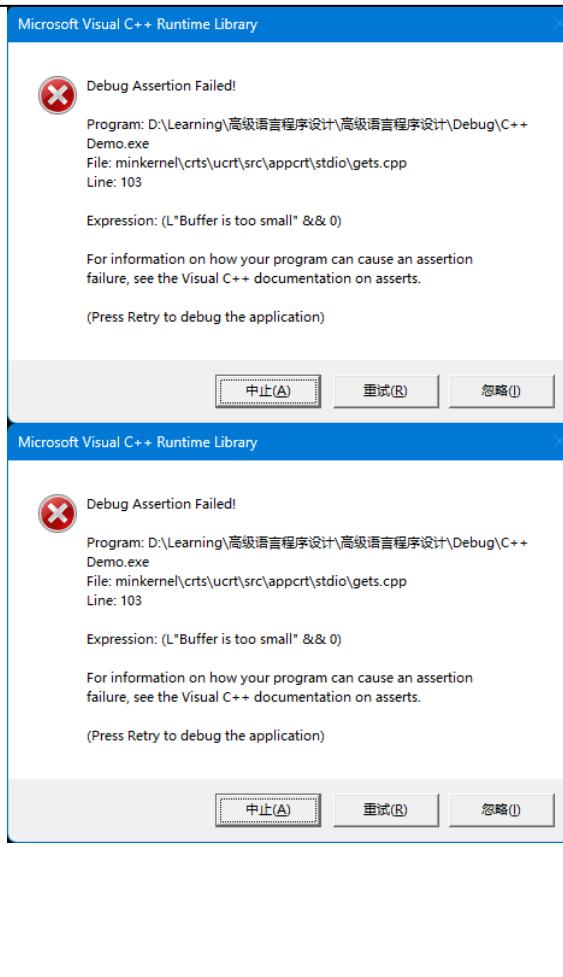
```
#include <iostream>
using namespace std;

int main()
{
    char a[10], b[20];

    gets_s(a);
    gets_s(b);

    cout << a << endl;
    cout << b << endl;

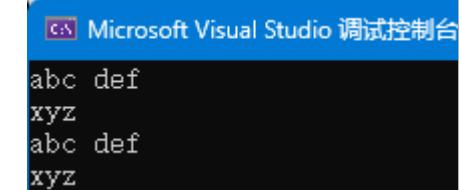
    return 0;
}
```



1、键盘输入abc空格def并回车，会继续等待输入，再输入xyz并回车，则输出为：

abc def

xyz



2、键盘输入超过9个字符，观察弹窗报错。

3、键盘先输入Hello并回车，再输入超过19个字符，观察弹窗报错。

问：为什么a最长输入只能是9？

字符串方式输出下\0表示结束，所以要保证输入正确，输入的字符个数要小于定义的字符数组的长度，所以a最长输入只能是9。

为什么b最长输入只能是19？

字符串方式输出下\0表示结束，所以要保证输入正确，输入的字符个数要小于定义的字符数组的长度，所以a最长输入只能是19。



## §. 基础知识题 – 字符数组的输入与输出

### 4. 多个字符串的输入

★ 从键盘输入含空格字符串的方法(不同编译器不同)

例22: Dev C++下用gets输入含空格的字符串 (Dev C++: 有gets, 无gets\_s, 有fgets)

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

int main()
{
    char a[10], b[20];

    gets(a);
    gets(b);

    cout << a << endl;
    cout << b << endl;

    return 0;
}
```

1、键盘输入abc空格def并回车，会继续等待输入，再输入xyz并回车，则输出为: abc def  
xyz

2、键盘输入超过9个字符，观察:

return value 3221225477

3、键盘先输入Hello并回车，再输入超过19个字符，观察:

C:\Us... abc def xyz abc def xyz	C:\Users\lenovo\... 1abcdefghijklmn 2abcdefghijklmn 1abcdefghijklmn 2abcdefghijklmn	C:\Users\lenovo\Desktop\demo.exe Hello abcdefghijklmnopqrstuvwxyz uvwxyz abcdefghijklmnopqrstuvwxyz
----------------------------------------------	-------------------------------------------------------------------------------------------------	-----------------------------------------------------------------------------------------------------------------

问：为什么a最长输入只能是9？

字符串方式输出下\0表示结束，所以要保证输入正确，输入的字符个数要小于定义的字符数组的长度，所以a最长输入只能是9。

为什么b最长输入只能是19？

字符串方式输出下\0表示结束，所以要保证输入正确，输入的字符个数要小于定义的字符数组的长度，所以a最长输入只能是19。



## §. 基础知识题 – 字符数组的输入与输出

### 4. 多个字符串的输入

★ 不同编译器从键盘输入含空格字符串的方法不同

例23：VS和Dev C++均可用fgets输入含空格的字符串（fgets函数的原型定义为：fgets(字符数组名，最大长度，stdin);）

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

int main()
{
    char a[10], b[20];

    fgets(a, 10, stdin);
    fgets(b, 20, stdin);

    cout << a << endl;
    cout << b << endl;
    int i;
    for(i=0; a[i]!='\0'; i++)
        cout << int(a[i]) << ',';
    cout << endl;

    for(i=0; b[i]!='\0'; i++)
        cout << int(b[i]) << ',';
    cout << endl;

    return 0;
}
```

1、键盘输入abc空格def并回车，会继续等待输入，再输入xyz并回车，则输出为：

xyz

97 98 99 32 100 101 102 10  
120 121 122 10

问1：和例21-22的输出区别在哪里？  
例21-22没有输出回车符，例23输出回车符。

问2：后面两段红色代码的目的是什么？  
输出读入的每个字符的ASCII码。

Microsoft Visual Studio 调试控制台

abc def  
xyz  
abc def  
  
xyz

97 98 99 32 100 101 102 10  
120 121 122 10

2、键盘输入9个字符并回车，则输出为：

abcdefghijklm

97 98 99 100 101 102 103 104 105  
10

Microsoft Visual Studio 调试控制台

abcdefghijklm  
abcdefghijklm

97 98 99 100 101 102 103 104 105  
10

3、如果输入28个字符并回车，则输出为：

abcdefghijklm

jklnopqrstuvwxyzab  
97 98 99 100 101 102 103 104 105  
106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 97 98

4、如果输入超过28个字符并回车，则输出为：

abcdefghijklm

jklnopqrstuvwxyzab  
97 98 99 100 101 102 103 104 105  
106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 97 98



## §. 基础知识题 - 字符数组的输入与输出

### 5. 二维字符数组的输入/输出

★ 数组名加双下标表示元素，单下标表示一维数组

例24：二维字符数组以双下标形式输出单个字符/单下标形式输出字符串

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

int main()
{
    char a[3][30]={"ABCDEFGHIJKLMNOPQRSTUVWXYZ",
                   "abcdefghijklmnopqrstuvwxyz",
                   "0123456789" };
    // 单个字符输出(数组名+双下标)
    printf("a[0][2]=%c\n", a[0][2]);
    cout << "a[1][20]=" << a[1][20] << endl;

    // 字符串输出(数组名+单下标)
    printf("a[0]=%s\n", a[0]);
    cout << "a[2]=" << a[2] << endl;

    return 0;
}
```

输出为：

a[0][2]=C  
a[1][20]=u  
a[0]=ABCDEFGHIJKLMNOPQRSTUVWXYZ  
a[2]=0123456789

Microsoft Visual Studio 调试控制台  
a[0][2]=C  
a[1][20]=u  
a[0]=ABCDEFGHIJKLMNOPQRSTUVWXYZ  
a[2]=0123456789



## §. 基础知识题 - 字符数组的输入与输出

### 5. 二维字符数组的输入/输出

★ 数组名加双下标表示元素，单下标表示一维数组

例25：二维字符数组以双下标形式输入单个字符

```
#define _CRT_SECURE_NO_WARNINGS //VS需要
#include <iostream>
using namespace std;

int main()
{
    char a[3][30]={"ABCDEFGHIJKLMNOPQRSTUVWXYZ",
                   "abcdefghijklmnopqrstuvwxyz",
                   "0123456789" };
    // 单字符输入(数组名+双下标)
    scanf("%c\n", &a[0][2]); //格式符为%c
    cin >> a[1][20]; //无&

    // 字符串输出(数组名+单下标)
    printf("a[0]=%s\n", a[0]);
    cout << "a[1]=" << a[1] << endl;

    return 0;
}
```

1、键盘输入#@并回车，输出为：

a[0]=AB#ABCDEFGHIJKLMNOPQRSTUVWXYZ  
a[1]=abcdefghijklmnopqrstuvwxyz@vwxyz

```
Microsoft Visual Studio 调试控制台
#@  
a[0]=AB#ABCDEFGHIJKLMNOPQRSTUVWXYZ  
a[1]=abcdefghijklmnopqrstuvwxyz@vwxyz
```

2、键盘输入#并回车，输入@并回车，输出为：

a[0]=AB#ABCDEFGHIJKLMNOPQRSTUVWXYZ  
a[1]=abcdefghijklmnopqrstuvwxyz@vwxyz

```
Microsoft Visual Studio 调试控制台
#  
@  
a[0]=AB#ABCDEFGHIJKLMNOPQRSTUVWXYZ  
a[1]=abcdefghijklmnopqrstuvwxyz@vwxyz
```



# §. 基础知识题 - 字符数组的输入与输出

## 5. 二维字符数组的输入/输出

★ 数组名加双下标表示元素，单下标表示一维数组

例26：二维字符数组以单下标形式输入字符串

```
#define _CRT_SECURE_NO_WARNINGS //VS需要
#include <iostream>
using namespace std;

int main()
{
    char a[3][30]={"ABCDEFGHIJKLMNOPQRSTUVWXYZ",
                   "abcdefghijklmnopqrstuvwxyz",
                   "0123456789" };

    scanf("%s", a[1]); //a[1]是一维数组名, 无&

    cout << "a[0]=" << a[0] << endl;
    cout << "a[1]=" << a[1] << endl;
    cout << "a[2]=" << a[2] << endl;

    return 0;
}
```

The screenshot shows three separate Microsoft Visual Studio debug console windows. Each window displays the following:  
1. The first window shows the declaration of a 3x30 character array 'a'.  
2. The second window shows the initial values assigned to each row: Row 0 contains 'ABCDEFGHIJKLMNOPQRSTUVWXYZ', Row 1 contains 'abcdefghijklmnopqrstuvwxyz', and Row 2 contains '0123456789'.  
3. The third window shows the state of the array after the first input operation. Row 0 remains the same. Row 1 has been modified to contain 10 '#' characters. Row 2 has been modified to contain 35 '#' characters.

1、输入≤29个字符，输出为：（以连续输入10个#为例）

```
a[0]=ABCDEFGHIJKLMNOPQRSTUVWXYZ
a[1]=#####
a[2]=0123456789
```

2、输入30~59个字符，输出为：（以连续输入35个#为例）

```
a[0]=ABCDEFGHIJKLMNOPQRSTUVWXYZ
a[1]=#####
a[2]=#####
```

3、输入60个以上字符，输出为：（以连续输入70个#为例）（弹窗报错）

```
a[0]=ABCDEFGHIJKLMNOPQRSTUVWXYZ
a[1]=#####
a[2]=#####
```

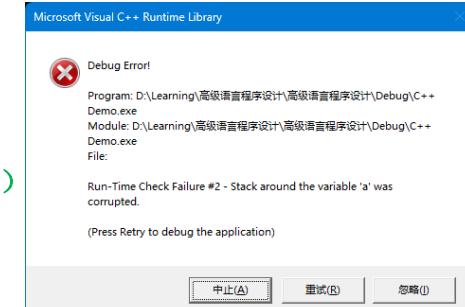
将scanf换为cin>>a[1];再重复1、2、3，观察结果：结果相同

问1：输入30~59个字符为什么不出现错误？a[2]中是什么？

第31~59个字符赋值给a[2]数组，未超过整个二维字符数组的合法空间，并未越界，所以不出现错误。a[2]中是第31~59个字符。

问2：简述你是怎么理解二维数组越界的？

可以将二维数组看成一维数组，二维数组在内存中也是连续存储的。数组的下标是有范围限制的（二维数组的下标范围为0~行数\*列数-1），所以二维数组的下标小于0或者大于行数\*列数-1时，都是越界访问，超出了数组的合法空间。





# §. 基础知识题 - 字符数组的输入与输出

## 5. 二维字符数组的输入/输出

★ 数组名加双下标表示元素，单下标表示一维数组

例27：二维字符数组从任一位置开始输出字符串

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

int main()
{
    char a[3][30]={"ABCDEFGHIJKLMNOPQRSTUVWXYZ",
                   "abcdefghijklmnopqrstuvwxyz",
                   "0123456789" };

    // (第1组) 单字符输出(数组名+双下标)
    printf("a[0][2]=%c\n", a[0][2]);
    cout << "a[1][20]=" << a[1][20] << endl;

    // (第2组) 字符串输出(&+数组名+双下标)
    printf("a[0][2]=%s\n", &a[0][2]);
    cout << "a[1][20]=" << &a[1][20] << endl;

    // (第3组) 字符串输出(数组名+单下标)
    printf("a[0]=%s\n", a[0]);
    cout << "a[2]=" << a[2] << endl;

    return 0;
}
```

输出为：

```
a[0][2]=C
a[1][20]=u
a[0][2]=CDEFGHIJKLMNOPQRSTUVWXYZ
a[1][20]=uvwxyz
a[0]=ABCDEFGHIJKLMNOPQRSTUVWXYZ
a[2]=0123456789
```

Microsoft Visual Studio 调试控制台

```
a[0][2]=C
a[1][20]=u
a[0][2]=CDEFGHIJKLMNOPQRSTUVWXYZ
a[1][20]=uvwxyz
a[0]=ABCDEFGHIJKLMNOPQRSTUVWXYZ
a[2]=0123456789
```

问1：同样双下标形式（第1/2组），

怎样输出单个字符？ C方式 : printf("%c", 数组名+双下标)

C++方式: cout << 数组名+双下标

怎样输出字符串？ C方式 : printf("%s", &+数组名+双下标)

C++方式: cout << &+数组名+双下标

问2：如何修改第2组的输出（必须保持双下标形式不变），使输出结果与第3组一致？

改为对应一维数组中第一个元素的地址，即&a[0][2]改为&a[0][0]，&a[1][20]改为&a[2][0]。



# §. 基础知识题 - 字符数组的输入与输出

## 5. 二维字符数组的输入/输出

★ 数组名加双下标表示元素，单下标表示一维数组

例28：二维字符数组从任一位置开始输入字符串

```
#define _CRT_SECURE_NO_WARNINGS //VS需要
#include <iostream>
using namespace std;

int main()
{
    char a[3][30]={"ABCDEFGHIJKLMNOPQRSTUVWXYZ",
                   "abcdefghijklmnopqrstuvwxyz",
                   "0123456789" };

    scanf("%s", &a[1][3]); //&+数组名+双下标

    cout << "a[0]=" << a[0] << endl;
    cout << "a[1]=" << a[1] << endl;
    cout << "a[2]=" << a[2] << endl;

    return 0;
}
```

The screenshots show the Microsoft Visual Studio Debug Console with three separate outputs. Each output shows the state of a 3x30 character array 'a'. The first output shows the initial state with all elements containing ASCII values for letters and digits. The second output shows the state after inputting 10 '#' characters into index [1][3], with indices [0] through [2] containing the original string and index [1] containing '#'. The third output shows the state after inputting 30 '#' characters, with indices [0] through [2] all containing '#'. This demonstrates that the input did not affect the last element of the array.

1、输入≤26个字符，输出为：（以连续输入10个#为例）

```
a[0]=ABCDEFGHIJKLMNOPQRSTUVWXYZ
a[1]=abc#####
a[2]=0123456789
```

2、输入27~56个字符，输出为：（以连续输入30个#为例）

```
a[0]=ABCDEFGHIJKLMNOPQRSTUVWXYZ
a[1]=abc#####
a[2]=###
```

3、输入56个以上字符，输出为：（以连续输入60个#为例）（弹窗报错）

```
a[0]=ABCDEFGHIJKLMNOPQRSTUVWXYZ
a[1]=abc#####
a[2]=#####
```

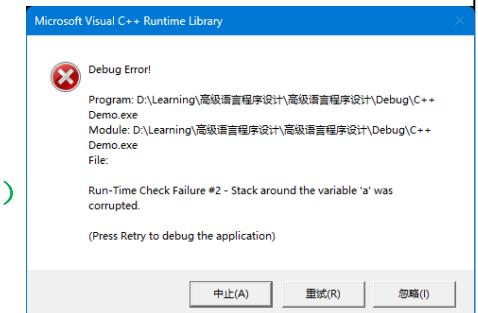
将scanf换为cin>>&a[1][3];再重复1、2、3，观察结果：结果相同

问1：输入27~56个字符为什么不出错？a[2]中是什么？

第28~56个字符赋值给a[2]数组，未超过整个二维字符数组的合法空间，并未越界，所以不出错。a[2]中是第28~56个字符。

问2：如果想不影响a[2]，例26中是≤29个字符，本例中是≤26个字符，差别在哪？

例26中是从一维数组中的第一个元素开始赋值，所以若想不影响a[2]应该≤29个字符，例28中是从一维数组中的第四个元素开始赋值，所以若想不影响a[2]应该≤26个字符。





# §. 基础知识题 - 字符数组的输入与输出

## 6. 尾零的输出

例29：在不同的控制台及字体设置下尾零输出的差异

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

int main()
{
    int i;
    char a[10] = { 'c', 'h', 'i', 'n', 'a' };

    cout << "0      1      2" << endl; //标尺
    cout << "012345678901234567890123456789" << endl; //标尺

    for (i = 0; i < 10; i++)
        cout << a[i] << '$'; //确认a[i]是否输出

    cout << '#' << endl; //加行尾识别符

    return 0;
}
```

### 1、新版控制台+新宋体28点阵

```
0      1      2
012345678901234567890123456789
c$h$i$n$a$$$$$#
```

### 2、旧版控制台+新宋体28点阵

```
0      1      2
012345678901234567890123456789
c$h$i$n$a$a$a$a$a$a$#
```

### 3、旧版控制台+新宋体16点阵

```
0      1      2
012345678901234567890123456789
c$h$i$n$a$ $ $ $ $ $#
```

结论：

- 1、不要以字符形式输出\0，因为看到的内容不可信。
- 2、如果想准确得知某字符的值，转为\_int类型输出即可。

# §. 基础知识题 – 字符数组的输入与输出



## 6. 尾零的输出

例30：在不同的控制台及字体设置下其它非图形字符输出的差异

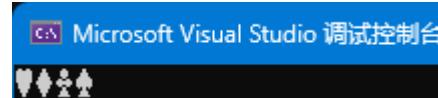
(去ASCII码表中查表示扑克牌四种花色的字符，用测试程序打印含这4个字符的字符串)

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

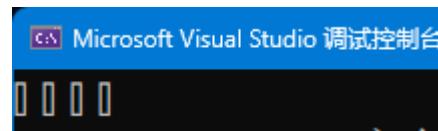
int main()
{
    char a[5] = "\3\4\5\6";
    cout << a;

    return 0;
}
```

1、新版控制台+点阵字体10×18点阵



2、新版控制台+新宋体28点阵



结论

上页的结论1也适用于其它非图形字符。



## §. 基础知识题 – C方式常用的字符串处理函数

1. strlen(const char s[])

例1：字符数组与字符串长度

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

int main()
{
    char str1[]="Hello";
    cout << sizeof(str1) << endl;
    cout << strlen(str1) << endl;

    char str2[]="china\0Hello\0\0";
    cout << sizeof(str2) << endl;
    cout << strlen(str2) << endl;

    return 0;
} //读操作，不需要加_CRT_SECURE_NO_WARNINGS
```

//给出程序的运行结果

```
Microsoft Visual Studio 调试控制台
6
5
14
5
```

问题：

- 1、求数组长度时，无论是否有显式\0，最后一定有隐式的\0。
- 2、当含有多个\0(显式/隐式)时，字符串长度计算到第一个\0为止。



## §. 基础知识题 – C方式常用的字符串处理函数

2. strcat(char dst[], const char src[])

例2：字符串连接

```
#define _CRT_SECURE_NO_WARNINGS //VS需要
#include <iostream>
#include <cstring>
using namespace std;

int main()
{
    char str1[30] = "Tongji "; //不能缺省，至少18字节
    char str2[] = "University";
    cout << strcat(str1, str2) << '#' << endl; //加#的目的：
                                                判断连接后字符串末尾位置

    return 0;
}
```

//给出程序的运行结果

Microsoft Visual Studio 调试控制台  
Tongji University#

问题：

- 1、str2数组的默认长度是11。
- 2、结合前面字符数组输入/输出的作业，strcat复制时包含src的\0。



## §. 基础知识题 – C方式常用的字符串处理函数

2. strcat(char dst[], const char src[])

例3：字符串连接（错误）

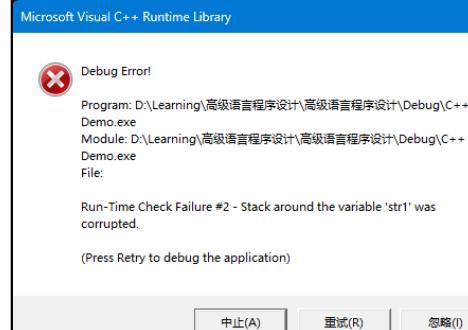
```
#define _CRT_SECURE_NO_WARNINGS //VS需要
#include <iostream>
#include <cstring>
using namespace std;

int main()
{
    char str1[]="Tongji ";
    char str2[]="University";
    cout << strcat(str1, str2) << '#' << endl; //加#的目的:
                                                    判断连接后字符串末尾位置

    return 0;
}
```

//给出程序的运行结果

```
Microsoft Visual Studio 调试控制台
Tongji University#
```



问题：

- 1、str1数组的大小必须给出，不能默认，其最小长度是\_18（针对本例的一个具体数字）。
- 2、dst数组的最小长度是\_dst和src两字符串总长度+1才能保证正确。



## §. 基础知识题 – C方式常用的字符串处理函数

3. strncat(char dst[], const char src[], const unsigned int n)

例4：字符串连接前n个字符

//例：字符串连接前n个字符

```
#define _CRT_SECURE_NO_WARNINGS //VS需要
#include <iostream>
#include <cstring>
using namespace std;

int main()
{
    char str1[30] = "Tongji ";
    char str2[30] = "Tongji ";
    char str3[] = "University";
    cout << strncat(str1, str3, 3) << '*' << endl;
    cout << strncat(str2, str3, 300) << '*' << endl;

    return 0;
}
```

//给出程序的运行结果

A screenshot of the Microsoft Visual Studio Debug Console window. It shows two lines of text: "Tongji Uni\*" and "Tongji University\*". The first line is the result of the strncat call with n=3, where only the first three characters of "University" were copied. The second line is the result of the strncat call with n=300, where the entire string "University" was copied, followed by a null terminator and the remaining characters from "Tongji ".

问题：

但n超过src表示的字符串的长度时，连接规则是只连接src字符串长度个字符。



## §. 基础知识题 – C方式常用的字符串处理函数

3. strncat(char dst[], const char src[], const unsigned int n)

例5：字符串连接前n个字符（错误）

//例：字符串连接前n个字符

```
#define _CRT_SECURE_NO_WARNINGS //VS需要
#include <iostream>
#include <cstring>
using namespace std;

int main()
{
    char str1[]="Tongji ";
    char str3[]="University"; //缺省长度为11
    cout << strncat(str1, str3, 3) << '*' << endl;

    return 0;
}
```

//给出程序的运行结果

Microsoft Visual Studio 调试控制台

Tongji Uni\*

Microsoft Visual C++ Runtime Library



Debug Error!

Program: D:\Learning\高级语言程序设计\高级语言程序设计\Debug\C++  
Demo.exe  
Module: D:\Learning\高级语言程序设计\高级语言程序设计\Debug\C++  
Demo.exe  
File:

Run-Time Check Failure #2 - Stack around the variable 'str1' was  
corrupted.

(Press Retry to debug the application)

中止(A) 重试(R) 忽略(I)

问题：

1、str1数组的大小必须给出，不能默认，其最小长度是11（针对本例的一个具体数字）。

2、dst数组的最小长度是原dst字符串长度+min(dst字符串长度, len)+1才能保证正确。



## §. 基础知识题 – C方式常用的字符串处理函数

4. strcpy(char dst[], const char src[])

例6：字符串拷贝

```
#define _CRT_SECURE_NO_WARNINGS //VS需要
#include <iostream>
#include <cstring>
using namespace std;
int main()
{
    int i;
    char a[]="student", b[]="hello";
    strcpy(a, b);
    cout << a << endl;
    for(i=0;i<8;i++)
        cout << int(a[i]) << ' ';
    cout << endl;

    return 0;
}
```

//给出程序的运行结果

```
Microsoft Visual Studio 调试控制台
hello
104 101 108 108 111 0 116 0
```

问题：

- 1、字符串复制时，复制到src的`\0`为止，包含`\0`，之后的字符不再复制。
- 2、在运行截图中用箭头指出证明结论1的位置。



## §. 基础知识题 – C方式常用的字符串处理函数

4. strcpy(char dst[], const char src[])

例7：字符串拷贝

```
#define _CRT_SECURE_NO_WARNINGS //VS需要
#include <iostream>
#include <cstring>
using namespace std;
int main()
{
    int i;
    char a[]="student", b[]="hello\0china";
    strcpy(a, b);
    cout << a << endl;
    for(i=0;i<8;i++)
        cout << int(a[i]) << ' ';
    cout << endl;

    return 0;
}
```

//给出程序的运行结果

Microsoft Visual Studio 调试控制台  
hello  
104 101 108 108 111 0 116 0

问题：1、a数组的默认大小是8，b数组的默认大小是12。

2、b数组的大小超过了a数组的大小，为什么运行不出错？字符串复制时，复制到src的\0为止，包含\0，之后的字符不再复制，因此复制的字符数量为6，未超过a数组的大小。

3、本例中，复制到b[5]就停止复制了。



## §. 基础知识题 – C方式常用的字符串处理函数

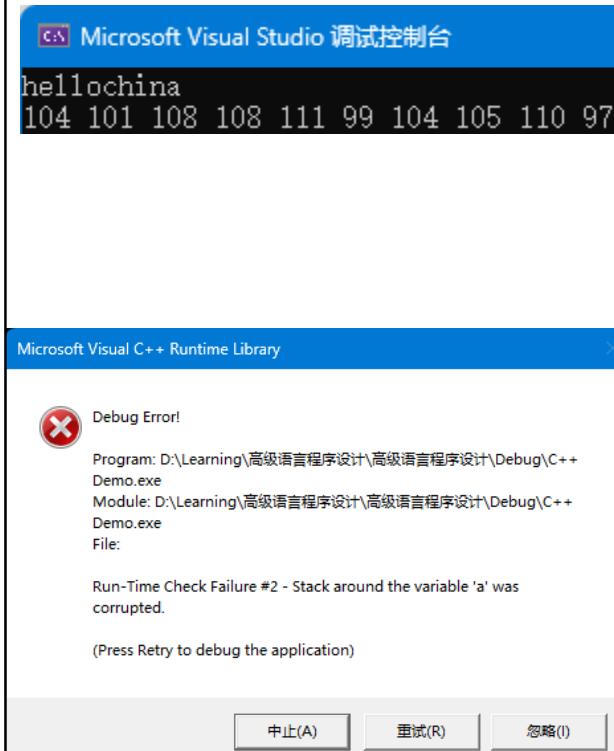
4. strcpy(char dst[], const char src[])

例8：字符串拷贝（有错）

```
#define _CRT_SECURE_NO_WARNINGS //VS需要
#include <iostream>
#include <cstring>
using namespace std;
int main()
{
    int i;
    char a[11] = "student", b[] = "hellochina";
    strcpy(a, b);
    cout << a << endl;
    for (i = 0; a[i] != '\0'; i++)
        cout << int(a[i]) << ',';
    cout << endl;

    return 0;
}
```

//给出程序的运行结果



问题：1、本程序为什么会错？因为b数组的大小超过了a数组的大小，且复制的字符数量超过了a数组的大小。

2、仅改a的定义使正确，如何做？（直接在源程序中修改）

3、dst数组的最小长度是\_max(dst字符串长度, src字符串长度)+1才能保证正确。



## §. 基础知识题 – C方式常用的字符串处理函数

5. `strncpy(char dst[], const char src[], unsigned int n)`

例9：字符串拷贝前n个字符

```
#define _CRT_SECURE_NO_WARNINGS //VS需要
#include <iostream>
#include <cstring>
using namespace std;
int main()
{
    int i;
    char a[]="student", b[]="hello";
    strncpy(a, b, 2);
    cout << a << endl;
    for(i=0;i<8;i++)
        cout << int(a[i]) << ',';
    cout << endl;

    return 0;
}
```

//给出程序的运行结果

Microsoft Visual Studio 调试控制台  
heudent  
104 101 117 100 101 110 116 0

问题:

本程序证明了`strncpy`复制时，不包含`\0`。



## §. 基础知识题 – C方式常用的字符串处理函数

5. strncpy(char dst[], const char src[], unsigned int n)

例10：字符串拷贝前n个字符

```
#define _CRT_SECURE_NO_WARNINGS //VS需要
#include <iostream>
#include <cstring>
using namespace std;
int main()
{
    int i;
    char a[]="student", b[]="hello";
    strncpy(a, &b[2], 2);
    cout << a << endl;
    for(i=0;i<8;i++)
        cout << int(a[i]) << ',';
    cout << endl;

    return 0;
}
```

//给出程序的运行结果

Microsoft Visual Studio 调试控制台  
11udent  
108 108 117 100 101 110 116 0

问题：

如果想从b[2]开始复制2个字符到a中，如何做？（期望输出：11udent）

（直接在源程序中修改）



## §. 基础知识题 – C方式常用的字符串处理函数

5. strncpy(char dst[], const char src[], unsigned int n)

例11：字符串拷贝前n个字符（深度讨论）

```
#define _CRT_SECURE_NO_WARNINGS //VS需要
#include <iostream>
#include <cstring>
using namespace std;

int main()
{
    int i;
    char a[] = "student", b[] = "hello";
    for (i = 0; i < 12; i++) //12已越界，目的:
        cout << int(a[i]) << ',';
    cout << endl;
    strncpy(a, b, 200); //观察数组越界后的输出内容，判断strncpy中n超过字符数组src的长度时拷贝是否到\0为止
    cout << a << endl;

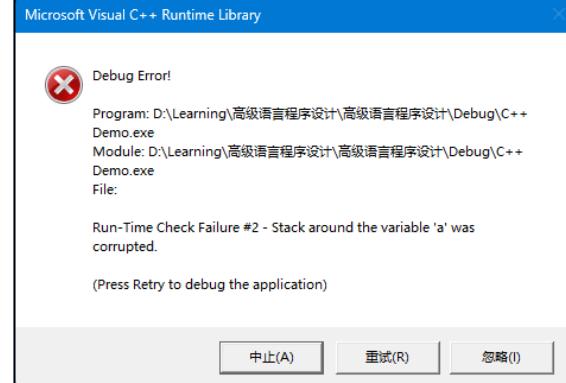
    for (i = 0; i < 12; i++) //12已越界，目的:
        cout << int(a[i]) << ',';
    cout << endl;
    return 0;
}
```

//给出VS下程序的运行结果

```
Microsoft Visual Studio 调试控制台
115 116 117 100 101 110 116 0 -52 -52 -52 -52
hello
104 101 108 108 111 0 0 0 0 0 0 0
```

//给出Dev下程序的运行结果

```
C:\Users\lenovo\Desktop\demo.exe
115 116 117 100 101 110 116 0 8 0 0 0
hello
104 101 108 108 111 0 0 0 8 0 0 0
```



问题：

观察两个for循环的后6个数字的输出，能得到什么结论？

strncpy中n超过字符数组src的长度时，拷贝不是到\0为止，继续拷贝导致出错。



## §. 基础知识题 - C方式常用的字符串处理函数

5. strncpy(char dst[], const char src[], unsigned int n)

例12：字符串拷贝前n个字符（深度讨论）

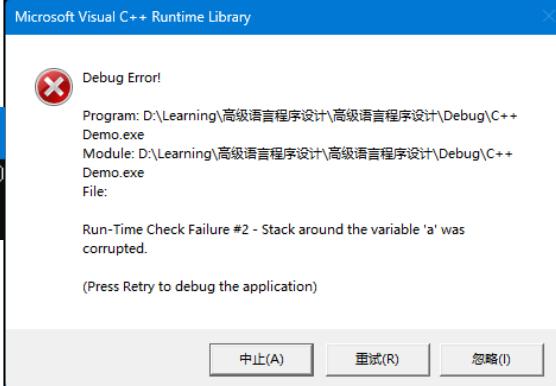
```
#define _CRT_SECURE_NO_WARNINGS //VS需要
#include <iostream>
#include <cstring>
using namespace std;

int main()
{
    int i;
    char a[] = "student", b[] = "hello";
    for (i = 0; i < 20; i++) //20已越界，目的:
        cout << int(a[i]) << ',';
    cout << endl;
    strncpy(a, b, 200); //观察数组越界后的输出内容，判断strncpy中n超过字符数组src的长度时拷贝是否到\0为止
    cout << a << endl;

    for (i = 0; i < 20; i++) //20已越界，目的:
        cout << int(a[i]) << ',';
    cout << endl;
    return 0;
}
```

//给出VS下程序的运行结果

```
Microsoft Visual Studio 调试控制台
115 116 117 100 101 110 116 0 -52 -52 -52 -52 -52 -52 -52 -52 -52 16 0 0 0
hello
104 101 108 108 111 0 0 0 0 0 0 0 0 0 0 0 0 16 0 0 0
```



//给出Dev下程序的运行结果

```
C:\Users\lenovo\Desktop\demo.exe
115 116 117 100 101 110 116 0 8 0 0 0 32 -68 64 0 -32 -2 120 0
hello
104 101 108 108 111 0 0 0 8 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

问题：

如果n超过了src的长度，则继续拷贝导致出错。



## §. 基础知识题 – C方式常用的字符串处理函数

6. strcmp(const char s1[], const char s2[])

例13：字符串比较

```
#include <iostream>
#include <cstring>
using namespace std;
int main()
{
    char str1[] = "house", str2[] = "horse";
    char str3[] = "abcd", str4[] = "abcde";
    char str5[] = "abcd", str6[] = "abc";
    char str7[] = "abcd", str8[] = "abcd";
    char str9[] = "abcd", str10[] = "abcd\0efgh";
    cout << strcmp(str1, str2) << endl;
    cout << strcmp(str3, str4) << endl;
    cout << strcmp(str5, str6) << endl;
    cout << strcmp(str7, str8) << endl;
    cout << strcmp(str9, str10) << endl;
    return 0;
}
```

//给出程序的运行结果

The screenshot shows the Microsoft Visual Studio Debug Console window titled "Microsoft Visual Studio 调试控制台". It displays the following output:  
1  
-1  
1  
0  
0

问题：两个字符串相等的条件是？

字符串长度相等且对应位置字符的ASCII码相同。



## §. 基础知识题 – C方式常用的字符串处理函数

6. strcmp(const char s1[], const char s2[])

例14：字符串比较（另一种形式）

```
#include <iostream>
#include <cstring>
using namespace std;
int main()
{
    char str1[]{"abcd", str2[]{"abcde"};
    int k = strcmp(str1, str2);
    if (k==0)
        cout << "串1 = 串2" << endl;
    else if (k<0)
        cout << "串1 < 串2" << endl;
    else
        cout << "串1 > 串2" << endl;

    return 0;
}
```

//给出程序的运行结果



Microsoft Visual Studio 调试控制台  
串1 < 串2

问题：给出两个字符串比较的执行过程。

两个字符串自左向右逐个字符相比（按ASCII值大小相比较），先比较第一个字符，如果第一个字符不相等则返回非0，如果第一个字符相等则比较下一个字符，自左向右逐个字符相比，直到出现不同的字符或遇到'\0'为止。



## §. 基础知识题 – C方式常用的字符串处理函数

6. strcmp(const char s1[], const char s2[])

例15：字符串比较（编译不报错但运行结果与期望不符合）

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

int main()
{
    char str1[]{"house", str2[]{"horse"};
    int k;

    k = str1 < str2;
    cout << k << endl;

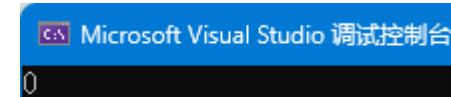
    return 0;
}
```

//给出程序的运行结果



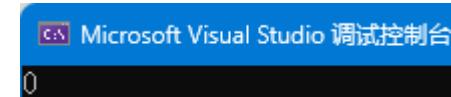
Microsoft Visual Studio 调试控制台  
0

//将str1和str2的内容互换，  
给出运行结果



Microsoft Visual Studio 调试控制台  
0

//将str1和str2都置为"house"，  
给出运行结果



Microsoft Visual Studio 调试控制台  
0

问题：

这个程序的运行结果是表示str1和str2的首地址进行比较。



## §. 基础知识题 – C方式常用的字符串处理函数

7. strncmp(const char s1[], const char s2[], const unsigned int n)

例16：字符串比较前n个字符

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

int main()
{
    char str1[] = "abcd", str2[] = "abcde";
    cout << strncmp(str1, str2, 3) << endl;
    cout << strncmp(str1, str2, 4) << endl;
    cout << strncmp(str1, str2, 5) << endl;
    cout << strncmp(str1, str2, 100) << endl;

    return 0;
}
```

//给出程序的运行结果

```
Microsoft Visual Studio 调试控制台
0
0
-1
-1
```

//将str2也置为"abcd"  
给出程序的运行结果

```
Microsoft Visual Studio 调试控制台
0
0
0
0
```

问题：

- 1、当n小于短串长度时，则比较到第n个字符。
- 2、当n大于等于短串长度时，则比较到短串末尾的\0为止。
- 3、如果n超过长串的长度，则比较到短串末尾的\0为止。



## §. 基础知识题 - C方式常用的字符处理函数

C方式常用的字符处理函数

```
#include <ctype.h> //C方式常用的字符处理函数所需头文件
```

```
isdigit(char ch); //判断是否数字(0~9)  
isalpha(char ch); //判断是否字母(A~Z, a~z)  
isalnum(char ch); //判断是否字母或数字(A~Z, a~z, 0~9)  
isxdigit(char ch); //判断是否16进制数字(0~9, A~F, a~f)  
isspace(char ch); //判断是否空格(含回车/换行/换页/TAB/竖向TAB)  
islower(char ch); //判断是否小写字母(a~z)  
isupper(char ch); //判断是否大写字母(A~Z)  
tolower(char ch); //大写转小写(其余原值返回)  
toupper(char ch); //小写转大写(其余原值返回)
```



## §. 基础知识题 - 与cin有关的成员函数的基本使用

### 1. 用于字符输入的流成员函数

例1: cin.get()

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

int main()
{
    char ch;

    ch = cin.get();
    cout << ch << int(ch) << endl;
    ch = cin.get();
    cout << ch << int(ch) << endl;

    return 0;
}
```

输入一个字符+回车，输出：该字符和该字符的ASCII码  
**(以输入a为例)**

空行

回车的ASCII码10

输入一串字符+回车，输出：第一个字符及其ASCII码  
**(以输入abc为例)**

第二个字符及其ASCII码

```
(0) Microsoft Visual Studio 调试控制台
a
a97
10
(0) Microsoft Visual Studio 调试控制台
abc
a97
b98
```



## §. 基础知识题 - 与cin有关的成员函数的基本使用

### 1. 用于字符输入的流成员函数

例2: cin.get()

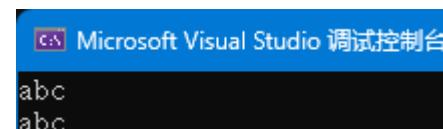
```
#include <iostream>
using namespace std;

int main()
{
    char ch;

    while((ch = cin.get()) != '\n')
        cout << ch;
    cout << endl;

    return 0;
}
```

输入一串字符+回车，输出：输入的字符串  
(以输入abc为例)





## §. 基础知识题 - 与cin有关的成员函数的基本使用

### 1. 用于字符输入的流成员函数

例3: cin.get()

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

int main()
{
    char ch;

    while((ch = cin.get()) != EOF)
        cout << ch;
    cout << endl;

    return 0;
}
```

输入: 连续多个一串字符+回车, 串中可含CTRL+Z

(以输入123, 回车, 输入a+CTRL+Z+b, 回车, 输入CTRL+Z为例)

输出: 第一个CTRL+Z之前的所有字符和一个不可见字符

输入: 连续多个一串字符+回车, 最后一行**单独**CTRL+Z

(以输入123, 回车, 输入abc, 回车, 输入CTRL+Z为例)

输出: 第一个CTRL+Z之前的所有字符和一个不可见字符, 最后一行单独CTRL+Z输入终止

```
Microsoft Visual Studio 调试控制台
123
123
a^Zb

Microsoft Visual Studio 调试控制台
123
123
abc
abc
^Z
```



## §. 基础知识题 - 与cin有关的成员函数的基本使用

### 1. 用于字符输入的流成员函数

例4: cin.get(字符变量)

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

int main()
{
    char ch;

    cin.get(ch);
    cout << ch << int(ch) << endl;
    cin.get(ch);
    cout << ch << int(ch) << endl;

    return 0;
}
```

输入一个字符+回车，输出：该字符和该字符的ASCII码  
**(以输入a为例)**

空行

回车的ASCII码10

输入一串字符+回车，输出：第一个字符及其ASCII码  
**(以输入abc为例)**

第二个字符及其ASCII码

```
Microsoft Visual Studio 调试控制台
a
a97
10
Microsoft Visual Studio 调试控制台
abc
a97
b98
```



## §. 基础知识题 - 与cin有关的成员函数的基本使用

### 1. 用于字符输入的流成员函数

例5: cin.get(字符变量)

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

int main()
{
    char ch;

    while(cin.get(ch))
        cout << ch;
    cout << endl;

    return 0;
}
```



输入: 连续多个一串字符+回车, 串中可含CTRL+Z (能否结束? 否)  
(以输入123, 回车, 输入a+CTRL+Z+b, 回车, 输入CTRL+Z为例)

输出: 第一个CTRL+Z之前的所有字符和一个不可见字符

输入: 连续多个一串字符+回车, 最后一行单独CTRL+Z

(以输入123, 回车, 输入abc, 回车, 输入CTRL+Z为例)

输出: 第一个CTRL+Z之前的所有字符和一个不可见字符, 最后一行  
单独CTRL+Z输入终止

问: 右侧为get无参例子, 左右两个程序的输出是否相同? 相同。

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

int main()
{
    char ch;

    while( (ch=cin.get())!=EOF )
        cout << ch;
    cout << endl;

    return 0;
}
```

//不需要写结果



## §. 基础知识题 - 与cin有关的成员函数的基本使用

### 1. 用于字符输入的流成员函数

例6: cin.get(字符变量)

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

int main()
{
    char ch;

    while((cin.get(ch)) != '\n')
        cout << ch;
    cout << endl;

    return 0;
}
```

**编译出错，为什么？**

cin.get(字符变量)返回cin(流对象自身)，无法进行逻辑运算比较，因此编译出错。

error C2678: 二进制“!=”：没有找到接受“std::basic\_istream<char, std::char\_traits<char>>”类型的左操作数的运算符(或没有可接受的转换)



## §. 基础知识题 - 与cin有关的成员函数的基本使用

### 1. 用于字符输入的流成员函数

例7: cin.get(字符变量)

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

int main()
{
    char ch;

    while((cin.get(ch))!=EOF)
        cout << ch;
    cout << endl;

    return 0;
}
```

**编译出错, 为什么?**

cin.get(字符变量)返回cin(流对象自身),  
无法进行逻辑运算比较, 因此编译出错。

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

int main()
{
    char ch;

    while( (ch=cin.get())!=EOF )
        cout << ch;
    cout << endl;

    return 0;
}
```

**编译正确, 为什么?**

cin.get()从输入流中读取一个字符并返回该字符,  
并将其赋值给char型变量ch, 进行!=运算的是cin.  
get()从输入流中读取并返回的字符, 因此编译正  
确。

error C2678: 二进制“!=”：没有找到接受“std::basic\_istream<char, std::char\_traits<char>>”类型的左操作数的运算符(或没有可接受的转换)



## §. 基础知识题 - 与cin有关的成员函数的基本使用

### 1. 用于字符输入的流成员函数

例8: cin.get(字符数组, 字符个数n, 中止字符)

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

int main()
{
    char ch[10];

    cin.get(ch, 10, '*');
    cout << ch << endl;

    return 0;
}
```

输入多于10个的字符串，输出：前9个字符

(以abcdefghijklmn输入为例)

输入小于10个的字符串，输出：光标闪烁，继续等待输入

(以abcdefg输入为例)

输入字符串，第9个及以前位置有\*，输出：\*之前的所有字符

(以abc\*defghijklmn输入为例)

输入字符串，第10个及以后位置有\*，输出：前9个字符

(以abcdefghijklm\*n输入为例)

The screenshots show the output of the program in the Microsoft Visual Studio Debug Console:

- Screenshot 1: Input: abcdefghijklmn. Output: abcdefghijklmn (The last character 'm' is not processed because it's at index 10 and '\*' is the stop character).
- Screenshot 2: Input: abcdefg. Output: abcdefg (The input ends at index 6, so the console waits for more input).
- Screenshot 3: Input: abc\*defghijklmn. Output: abc\*defghijklmn (The asterisk at index 3 is included in the output).
- Screenshot 4: Input: abcdefghijklm\*n. Output: abcdefghijklm\*n (The asterisk at index 10 is included in the output).



## §. 基础知识题 - 与cin有关的成员函数的基本使用

### 1. 用于字符输入的流成员函数

例9: cin.get(字符数组, 字符个数n, 中止字符)

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

int main()
{
    char ch[10];

    cin.get(ch, 10); //省略第3个参数
    cout << ch << endl;

    return 0;
}
```

输入多于10个的字符串，输出：前9个字符

(以abcdefgijklmn输入为例)

输入小于10个的字符串，输出：输入的所有字符

(以abcdefg输入为例)

The screenshot shows two separate instances of the Microsoft Visual Studio Debug Console. The top instance shows the output of the first code example: 'abcdefgijklmn' on the first line and 'abcdefghi' on the second line. The bottom instance shows the output of the second code example: 'abcdefg' on both the first and second lines.

```
Microsoft Visual Studio 调试控制台
abcdefgijklmn
abcdefghi

Microsoft Visual Studio 调试控制台
abcdefg
abcdefg
```



## §. 基础知识题 – 与cin有关的成员函数的基本使用

### 1. 用于字符输入的流成员函数

例10: cin.getline(字符数组, 字符个数n, 中止字符)

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

int main()
{
    char ch[10];
    cin.getline(ch, 10, '*');
    cout << ch << endl;
    return 0;
}
```

输入多于10个的字符串，输出：前9个字符

(以abcdefgijklmn输入为例)

输入小于10个的字符串，输出：光标闪烁，继续等待输入

(以abcdefg输入为例)

输入字符串，第9个及以前位置有\*，输出：\*之前的所有字符

(以abc\*defghijklmn输入为例)

输入字符串，第10个及以后位置有\*，输出：前9个字符

(以abcdefgijklm\*n输入为例)

是否与三个参数的cin.get相同？相同。

The screenshots show the Immediate Window of Microsoft Visual Studio with the following outputs:

- Screenshot 1: Input "abcdefgijklmn", Output "abcdefghijklmn" (truncated by the window border).
- Screenshot 2: Input "abcdefg", Output "abcdefg" followed by a cursor闪烁 (blinking) at the end.
- Screenshot 3: Input "abc\*defghijklmn", Output "abc\*defghijklmn" (truncated by the window border).
- Screenshot 4: Input "abcdefgijklm\*n", Output "abcdefgijklm\*n" (truncated by the window border).



## §. 基础知识题 – 与cin有关的成员函数的基本使用

### 1. 用于字符输入的流成员函数

例11：三个参数的cin.get与cin.getline的区别

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

int main()
{
    char ch[20];
    cout << "enter a sentence:"; //不需要endl
    cin >> ch; //直接cin, 空格结束
    cout << "The string with cin is:" << ch << '#' << endl;
    cin.getline(ch, 20, '/');
    cout << "The second part is:" << ch << '#' << endl;
    cin.getline(ch, 20); //缺省是回车结束
    cout << "The third part is:" << ch << '#' << endl;
}
```

运行结果：

enter a sentence:I like C++./I study C++./I am happy.

The string with cin is:I#

The second part is: like C++.#

The third part is:I study C++./I am h#

Microsoft Visual Studio 调试控制台

```
enter a sentence:I like C++./I study C++./I am happy.
The string with cin is:I#
The second part is: like C++.#
The third part is:I study C++./I am h#
```

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

int main()
{
    char ch[20];
    cout << "enter a sentence:"; //不需要endl
    cin >> ch; //直接cin, 空格结束
    cout << "The string with cin is:" << ch << '#' << endl;
    cin.getline(ch, 20, '/');
    cout << "The second part is:" << ch << '#' << endl;
    cin.getline(ch, 20, '/');
    cout << "The third part is:" << ch << '#' << endl;
}
```

运行结果：

enter a sentence:I like C++./I study C++./I am happy.

The string with cin is:I#

The second part is: like C++.#

The third part is:I study C++.#

Microsoft Visual Studio 调试控制台

```
enter a sentence:I like C++./I study C++./I am happy.
The string with cin is:I#
The second part is: like C++.#
The third part is:I study C++.#
```



## §. 基础知识题 – 与cin有关的成员函数的基本使用

### 1. 用于字符输入的流成员函数

例12：三个参数的cin.get与cin.getline的使用区别

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

int main()
{
    char ch[20];
    cout << "enter a sentence:"; //不需要endl
    cin >> ch; //直接cin, 空格结束
    cout << "The string with cin is:" << ch << '#' << endl;
    cin.get(ch, 20, '/');
    cout << "The second part is:" << ch << '#' << endl;
    cin.get(ch, 20, '/');
    cout << "The third part is:" << ch << '#' << endl;
}
```

和上页的差别：两句蓝色语句从getline变为get，则结果：

```
enter a sentence:I like C++./I study C++./I am happy.
The string with cin is:I#
The second part is: like C++.#
The third part is:#
```

getline：遇见终止字符，在缓冲区内清除终止字符

get：遇见终止字符，将终止字符保留在缓冲区中

```
Microsoft Visual Studio 调试控制台
enter a sentence:I like C++./I study C++./I am happy.
The string with cin is:I#
The second part is: like C++.#
The third part is:#
```



# §. 基础知识题 – 与cin有关的成员函数的基本使用

## 1. 用于字符输入的流成员函数

例13：三个参数的cin.get与cin.getline的使用区别

```
#include <iostream>
using namespace std;
int main()
{   char ch1[10], ch2[10];
    cin.get(ch1, 10, '*');
    cout << ch1 << endl;
    cin.get(ch2, 10, '*');
    cout << ch2 << endl;
    return 0;
}
```

输入一串大于20个字符的字符串，输出：

前九个字符及之后九个字符

(以输入abcdefghijklmnopqrstuvwxyz为例)

输入一串字符串，每9个以内含\*，输出：

第一个\*之前的所有字符

(以输入abcd\*efghi jkl\*mnopqrst\*uvwxyz为例)

输入一串小于9的字符串，加回车，输出：

光标闪烁，继续等待输入

(以输入abcdefg为例)

```
D:\Learning\高级语言程序设计\高级语言程序设计\Debug\C++ Demo.exe
abcdefg
```

```
#include <iostream>
using namespace std;
int main()
{   char ch1[10], ch2[10];
    cin.getline(ch1, 10, '*');
    cout << ch1 << endl;
    cin.getline(ch2, 10, '*');
    cout << ch2 << endl;
    return 0;
}
```

输入一串大于20个字符的字符串，输出：

前九个字符

(以输入abcdefghijklmnopqrstuvwxyz为例)

输入一串字符串，每9个以内含\*，输出：

第一个\*之前的字符及两个\*之间的前九个字符

(以输入abcd\*efghi jkl\*mnopqrst\*uvwxyz为例)

输入一串小于9的字符串，加回车，输出：

光标闪烁，继续等待输入

(以输入abcdefg为例)

```
D:\Learning\高级语言程序设计\高级语言程序设计\Debug\C++ Demo.exe
abcdefgijklmnopqrstuvwxyz
abcdefg
abcd*efghi jkl*mnopqrst*uvwxyz
abcd
efghi jkl
```

● 输入满： get满后 可继续输入

getline满后 停止输入

● 遇中止字符： get遇中止字符，下一个 仍为中止字符

getline遇中止字符，下一个 跳过中止字符

● 未满遇回车： get把回车当一个普通字符读入至满，下一个 继续读入回车之后字符

getline把回车当一个普通字符读入至满，下一个 继续读入回车之后字符



## §. 基础知识题 - 与cin有关的成员函数的基本使用

### 2. 与字符输入有关的其它成员函数

例14: cin.eof()

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

int main() //P. 430 例13.5
{
    char c;

    while (!cin.eof())
        if ((c=cin.get())!=' ')
            cout.put(c);

    return 0;
}
```

输入: 连续多个字符串(含空格及CTRL+Z)+回车, 最后一行**单独**CTRL+Z  
(以输入123空格456, 回车, 输入abc+CTRL+Z+def, 回车, 输入CTRL+Z为例)

输出: 第一个CTRL+Z之前的所有字符和一个不可见字符  
最后一行单独CTRL+Z输入终止

The screenshot shows the Microsoft Visual Studio Debug Console window titled "Microsoft Visual Studio 调试控制台". The console displays the following text:  
123 456  
123456  
abc^Zdef  
abc^Z



## §. 基础知识题 - 与cin有关的成员函数的基本使用

### 2. 与字符输入有关的其它成员函数

例14: cin.peek()

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

int main()
{
    char ch;

    ch = cin.peek();
    cout << ch << int(ch) << endl;
    ch = cin.get();
    cout << ch << int(ch) << endl;

    return 0;
}
```

输入: ab      输出为:

```
Microsoft Visual Studio 调试控制台
ab
a97
a97
```

输入: CTRL+Z    输出为:

```
Microsoft Visual Studio 调试控制台
^Z
-1
-1
```



## §. 基础知识题 - 与cin有关的成员函数的基本使用

### 2. 与字符输入有关的其它成员函数

例15: cin.putback(字符变量/字符常量)

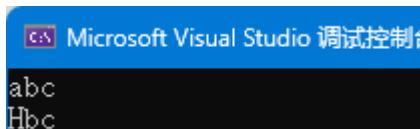
```
#include <iostream>
using namespace std;

int main()
{
    char ch;
    ch = cin.get(); //get()一次

    cin.putback('H'); //putback()一次

    while((ch=cin.get()) != '\n')
        cout.put(ch);

    return 0;
}
```

输入: abc    输出: 



## §. 基础知识题 - 与cin有关的成员函数的基本使用

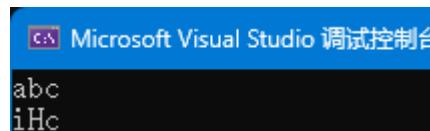
### 2. 与字符输入有关的其它成员函数

例16: cin.putback(字符变量/字符常量)

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

int main()
{
    char ch;
    ch = cin.get(); //get()两次
    ch = cin.get();
    cin.putback('H'); //putback()两次
    cin.putback('i');
    while((ch=cin.get()) != '\n')
        cout.put(ch);

    return 0;
}
```

输入: abc    输出: 



## §. 基础知识题 - 与cin有关的成员函数的基本使用

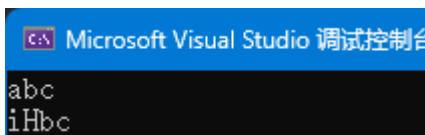
### 2. 与字符输入有关的其它成员函数

例17: cin.putback(字符变量/字符常量)

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

int main()
{
    char ch;
    ch = cin.get();      //get()一次
    cin.putback('H');   //putback()两次
    cin.putback(' i');
    while((ch=cin.get()) != '\n')
        cout.put(ch);
    return 0;
}
```

输入: abc

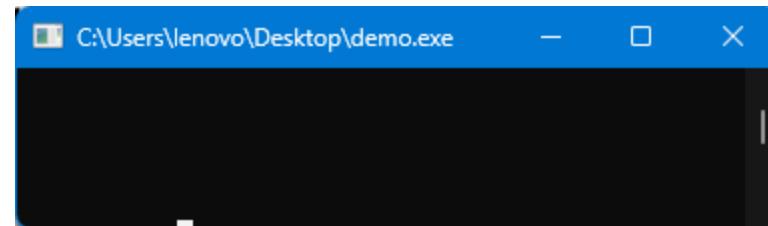


输出: VS : abc

Dev :

光标一直在动，什么意思？

持续输出不可见字符。



上两页的正确情况，本页的错误情况，综合起来，putback使用时要注意什么问题？  
putback使用时要注意插入的字符数不要超过原输入流长度。

## §. 基础知识题 - 与cin有关的成员函数的基本使用



## 2. 与字符输入有关的其它成员函数

例18: cin.putback(字符变量/字符常量)

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

int main()
{
    char ch;
    ch = cin.get();
    cin.putback('H');
    cin.putback('i');
    while((ch=cin.get()) != '\n')
        cout << int(ch) << ',';
    return 0;
}
```

输入: abc

输出: VS : abc  
105 72 98 9

Dey :  选择 C:\Users\lenovo\Desktop\demo.exe



## §. 基础知识题 - 与cin有关的成员函数的基本使用

### 2. 与字符输入有关的其它成员函数

例19: cin.putback(字符变量/字符常量)

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

int main()
{
    char ch;
    ch = cin.get();
    cin.putback('H');
    cin.putback('i');
    while((ch=cin.get())!=EOF) //判断条件换为!=EOF
        cout.put(ch);
    return 0;
}
```

输入: abc

输出: VS : abc  
Dev :

```
D:\Learning\高级语言程序设计\高级语言程序设计\Debug\C++ Demo.exe
abc
iHbc

C:\Users\lenovo\Desktop\demo.exe
abc
```



## §. 基础知识题 - 与cin有关的成员函数的基本使用

### 2. 与字符输入有关的其它成员函数

例20: cin.putback(字符变量/字符常量)

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

int main()
{   char c[20];
    int ch;
    cout << "please enter a sentense:" << endl;
    cin.getline(c, 15, '/');
    cout << "The first part is:" << c << endl;
    ch = cin.peek();
    cout << "The next char(ASCII):" << ch << endl;
    cin.putback(c[0]);
    cin.getline(c, 15, '/');
    cout << "The second part is:" << c << endl;
    return 0;
}
```

运行结果(红色为输入):

```
please enter a sentense:I am a boy./ am a student.
The first part is:I am a boy.
The next char(ASCII):32
The second part is:I am a student
```



## §. 基础知识题 - 与cin有关的成员函数的基本使用

### 2. 与字符输入有关的其它成员函数

例21: cin.ignore(字符个数n, 中止字符)

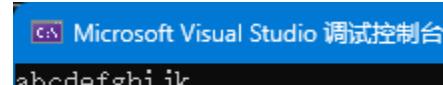
```
#include <iostream>
using namespace std;

int main()
{
    char ch;
    ch = cin.get();
    cout << ch;
    cin.ignore(5, 'A');
    ch = cin.get();
    cout << ch;

    return 0;
}
```

输入: abcdefghijkl

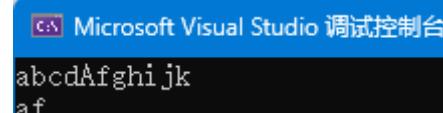
输出:



Microsoft Visual Studio 调试控制台  
abcdefghijkl

输入: abcdAfghijk

输出:



Microsoft Visual Studio 调试控制台  
abcdAfghijk



## §. 基础知识题 - 与cin有关的成员函数的基本使用

### 2. 与字符输入有关的其它成员函数

例22: cin.ignore(字符个数n, 中止字符)

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

int main()
{
    char ch;
    ch = cin.get();
    cout << ch;
    cin.ignore(); //缺省1个字符, 中止字符为EOF
    ch = cin.get();
    cout << ch;

    return 0;
}
```

输入: abcdefghijkl

输出:

ac

输入: abcdAfgijk

输出:

ac



## §. 基础知识题 - 与cin有关的成员函数的基本使用

### 2. 与字符输入有关的其它成员函数

例23: cin.ignore(字符个数n, 中止字符)

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

int main()
{
    char ch[20];
    cin.get(ch, 20, '/'); //指针停留在'/'处
    cout << "The first part is:" << ch << endl;

    cin.get(ch, 20, '/'); //从'/'处取, 为空
    cout << "The second part is:" << ch << endl;
    return 0;
}
```

输入: I like C++. /I study C++. /I am happy.

输出:

```
Microsoft Visual Studio 调试控制台
I like C++. /I study C++. /I am happy.
The first part is:I like C++.
The second part is:
```



## §. 基础知识题 - 与cin有关的成员函数的基本使用

### 2. 与字符输入有关的其它成员函数

例24: cin.ignore(字符个数n, 中止字符)

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

int main()
{
    char ch[20];
    cin.get(ch, 20, '/'); //指针停留在'/'处
    cout << "The first part is:" << ch << endl;
    cin.ignore(); //跳过'/'
    cin.get(ch, 20, '/'); //从'/'后取, 非空
    cout << "The second part is:" << ch << endl;
    return 0;
}
```

输入: I like C++. /I study C++. /I am happy.

输出:

```
Microsoft Visual Studio 调试控制台
I like C++. /I study C++. /I am happy.
The first part is:I like C++.
The second part is:I study C++.
```



## §. 基础知识题 - 与cin有关的成员函数的基本使用

### 3. 输入错误处理

#### 例25：C++方式cin输入错误处理

```
int x;
while (1) { //输入错误
    cout << "请输入x的值[0-100] : ";
    cin >> x;
    if (cin.fail()) {
        cin.clear();
        /* 方式1：清除缓冲区中字符，到'\n'为止，最大65536(可改) */
        cin.ignore(65536, '\n');
        /* 方式2：#include <limits>: INT_MAX */
        cin.ignore(INT_MAX, '\n');
        /* 方式3：numeric_limits<streamsize>::max(): 返回流缓冲区的最大容量 */
        /*       VS : 64位整数时直接可用，当 #include <Windows.h> 时不可用 */
        /*       Dev: 32位整数且 #include <limits> 时可用 */
        cin.ignore(numeric_limits<streamsize>::max(), '\n');
        continue;
    }
    else if (x >= 0 && x <= 100) { //输入正确且满足条件
        break;
    }
    else { //输入正确但不满足条件
        continue;
    }
}
```



## §. 基础知识题 - 与cin有关的成员函数的基本使用

### 3. 输入错误处理

#### 例26：C方式scanf输入错误处理

```
int x;
while (1) {
    printf("请输入x的值[0-100] : ");
    if (scanf("%d", &x) != 1) { //输入错误
        while (getchar() != '\n')
            continue;
        continue;
    }
    else if (x >= 0 && x <= 100) { //输入正确且满足条件
        break;
    }
    else { //输入正确但不满足条件
        continue;
    }
}
```



## §. 基础知识题 – 与cout有关的成员函数的基本使用

### 1. 用于字符输出的流成员函数

例1: cout.put()

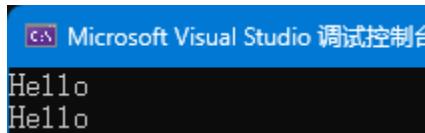
```
#include <iostream>
using namespace std;

int main()
{
    char str[] = "Hello";
    int i;

    for (i = 0; i < 5; i++)
        cout.put(str[i]);
    cout.put('n');
    cout.put('H').put('e').put('l').put('l').put('o').put(0x0A);

    return 0;
}
```

运行结果:





## §. 基础知识题 – 与cout有关的成员函数的基本使用

### 1. 用于字符输出的流成员函数

例2: cout.write()

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

int main()
{
    char s1[] = "Hello";
    cout.write(s1, 5);
    cout.put('\n');
    cout.write(s1, 10);
    cout.put('\n');

    char s2[] = { 'H', 'e', 'l', 'l', 'o' };
    cout.write(s2, 5);
    cout.put('\n');
    cout.write(s2, 10);
    cout.put('\n');

    return 0;
}
```

运行结果:

当write的参数是字符串(s1)，且write要写的长度超过字符串长度时的表现：

cout.write()成员函数不会判断字符串是否结尾，指定显示几个字符就显示几个，即使数组越界，当write要写的长度超过字符串长度时，则继续输出后续内存中未初始化的随机值。

当write的参数非字符串(s2)，且write要写的长度超过字符串长度时的表现：

cout.write()成员函数不会判断字符串是否结尾，指定显示几个字符就显示几个，即使数组越界，当write要写的长度超过字符串长度时，则继续输出后续内存中未初始化的随机值。

结论：用write向标准输出设备输出指定个数的字符时，输出缓冲区不要求是字符串



## §. 基础知识题 – 与cout有关的成员函数的基本使用

### 2. 用于字符输出控制的流成员函数

例3: cout.width()

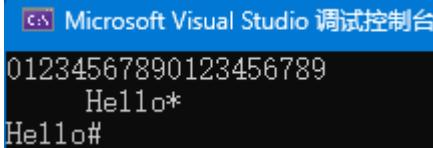
```
#include <iostream>
using namespace std;

int main()
{
    char s1[] = "Hello";

    cout << "01234567890123456789" << endl;
    cout.width(10);
    cout << s1 << '*' << endl;
    cout << s1 << '#' << endl;

    return 0;
}
```

运行结果:



Microsoft Visual Studio 调试控制台  
01234567890123456789  
Hello\*  
Hello#

结论:

- 1、cout.width(10) 等价于 cout << setw(10)
- 2、cout.width() 设置后仅1次有效



## §. 基础知识题 – 与cout有关的成员函数的基本使用

### 2. 用于字符输出控制的流成员函数

例4: cout.width() 与 cout.fill()

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

int main()
{
    char s1[] = "Hello";

    cout << "01234567890123456789" << endl;
    cout.width(10);
    cout.fill('$');
    cout << s1 << '*' << endl;
    cout.width(15);
    cout << s1 << '#' << endl;
    cout.width(12);
    cout.fill(' ');
    cout << s1 << '*' << endl;

    return 0;
}
```

运行结果:

```
Microsoft Visual Studio 调试控制台
01234567890123456789
$$$$$Hello*
$$$$$$$$$Hello#
Hello*
```

结论:

- 1、cout.fill() 等价于 cout << setfill()
- 2、cout.fill() 设置后始终有效
- 3、默认的cout.fill() 设置是哪个字符?  
ASCII码为32的空格字符(即' ')



## §. 基础知识题 – 与cout有关的成员函数的基本使用

### 2. 用于字符输出控制的流成员函数

例5: cout.width() 与 cout.setf()

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

int main()
{
    char s1[] = "Hello";

    cout << "01234567890123456789" << endl;
    cout.width(10);
    cout.setf(ios::left);
    cout << s1 << '*' << endl;
    cout.width(15);
    cout << s1 << '#' << endl;

    return 0;
}
```

运行结果:

Microsoft Visual Studio 调试控制台  
01234567890123456789  
Hello \*  
Hello #

结论:

- 1、cout.setf(ios::left) 等价于  
cout << setiosflags(ios::left)
- 2、cout.setf() 设置后始终有效



## §. 基础知识题 – 与cout有关的成员函数的基本使用

### 2. 用于字符输出控制的流成员函数

例6: cout.width() 与 cout.setf()

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

int main()
{
    char s1[] = "Hello";

    cout << "01234567890123456789" << endl;
    cout.width(10);
    cout.setf(ios::left);
    cout << s1 << '*' << endl;
    cout.setf(ios::right);
    cout.width(15);
    cout << s1 << '#' << endl;
    cout.width(10);
    cout.setf(ios::left);
    cout << s1 << '*' << endl;
    return 0;
}
```

运行结果:

```
Microsoft Visual Studio 调试控制台
01234567890123456789
Hello      *
          Hello#
Hello*#
```

结论:

- 1、cout.setf(ios::left) 等价于  
cout << setiosflags(ios::left)
- 2、cout.setf(ios::right) 等价于  
cout << setiosflags(ios::right)
- 3、cout.setf()设置后始终有效
- 4、不设置默认是右对齐
- 5、left后设置right是有效的的
- 6、right后设置left是无效的



## §. 基础知识题 – 与cout有关的成员函数的基本使用

### 2. 用于字符输出控制的流成员函数

例7: cout.width() 与 cout.setf()、cout.unsetf()

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

int main()
{
    char s1[] = "Hello";

    cout << "01234567890123456789" << endl;
    cout.width(10);
    cout.setf(ios::left);
    cout << s1 << '*' << endl;
    cout.setf(ios::right);
    cout.width(15);
    cout << s1 << '#' << endl;
    cout.width(10);
    cout.unsetf(ios::right); // 使用cout. 函数名
    cout.setf(ios::left);
    cout << s1 << '*' << endl;
    return 0;
}
```

将程序补充完整，得到期望的运行结果：

```
Microsoft Visual Studio 调试控制台
01234567890123456789
Hello *
Hello#
Hello *
```

所用的 cout.unsetf(ios::right) 等价于  
cout << resetiosflags(ios::right)



## §. 基础知识题 – 与cout有关的成员函数的基本使用

### 2. 用于字符输出控制的流成员函数

例8: cout.precision()

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

int main()
{
    double d = 123.456789123456;

    cout << d << '*' << endl;
    cout.precision(10);
    cout << d << '*' << endl;
    cout.precision(3);
    cout << d << '*' << endl;
    cout.precision(5);
    cout << d << '*' << endl;
    cout.precision(20);
    cout << d << '*' << endl;
    return 0;
}
```

运行结果:

Microsoft Visual Studio 调试控制台  
123.457\*  
123.4567891\*  
123\*  
123.46\*  
123.45678912345600509\*

结论:

- 1、不做任何设置的情况下，浮点数默认为小数方式；不设precision的输出宽度，默认为6
- 2、默认情况下，precision设定的宽度是全部数据
- 3、宽度不包含小数点
- 4、如果宽度超过有效位数，则可以显示，超出有效位数不可信



## §. 基础知识题 – 与cout有关的成员函数的基本使用

### 2. 用于字符输出控制的流成员函数

例9: cout.precision()

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

int main()
{
    double d = 123.456789123456;
    cout.setf(ios::fixed);
    cout << d << '*' << endl;
    cout.precision(10);
    cout << d << '*' << endl;
    cout.precision(3);
    cout << d << '*' << endl;
    cout.precision(5);
    cout << d << '*' << endl;
    cout.precision(20);
    cout << d << '*' << endl;
    return 0;
}
```

运行结果:

```
Microsoft Visual Studio 调试控制台
123.456789*
123.4567891235*
123.457*
123.45679*
123.45678912345600508615*
```

结论:

- 1、加ios::fixed后, precision默认的宽度为6, 设定的宽度是小数部分
- 2、宽度不包含小数点
- 3、如果宽度超过有效位数, 则可以显示, 超出有效位数不可信



## §. 基础知识题 – 与cout有关的成员函数的基本使用

### 2. 用于字符输出控制的流成员函数

例10: cout.precision()

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

int main()
{
    double d = 123.456789123456;
    cout.setf(ios::scientific);
    cout << d << '*' << endl;
    cout.precision(10);
    cout << d << '*' << endl;
    cout.precision(3);
    cout << d << '*' << endl;
    cout.precision(5);
    cout << d << '*' << endl;
    cout.precision(20);
    cout << d << '*' << endl;
    return 0;
}
```

运行结果:

Microsoft Visual Studio 调试控制台  
1.234568e+02\*  
1.2345678912e+02\*  
1.235e+02\*  
1.23457e+02\*  
1.23456789123456005086e+02\*

结论:

- 1、加ios::scientific后，precision默认的宽度为6，设定的宽度是小数部分
- 2、宽度不包含小数点
- 3、如果宽度超过有效位数，则可以显示，超出有效位数不可信



## §. 基础知识题 – 与cout有关的成员函数的基本使用

### 2. 用于字符输出控制的流成员函数

例11: cout.precision()

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

int main()
{
    double d = 123.456789123456;
    cout.setf(ios::fixed);
    cout.precision(10);
    cout << d << '*' << endl;

    cout.setf(ios::scientific);
    cout.precision(10);
    cout << d << '*' << endl;

    return 0;
}
```

运行结果:

Microsoft Visual Studio 调试控制台  
123.4567891235\*  
0x1.edd3c08729a5fp+6\*

结论:

先设ios::fixed后，再设ios::scientific，  
则输出显示错误



## §. 基础知识题 – 与cout有关的成员函数的基本使用

### 2. 用于字符输出控制的流成员函数

例12: cout.precision()

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

int main()
{
    double d = 123.456789123456;
    cout.setf(ios::scientific);
    cout.precision(10);
    cout << d << '*' << endl;

    cout.setf(ios::fixed);
    cout.precision(10);
    cout << d << '*' << endl;

    return 0;
}
```

运行结果:

Microsoft Visual Studio 调试控制台  
1.2345678912e+02\*  
0x1.edd3c08729a5fp+6\*

结论:

先设ios::scientific后，再设ios::fixed，  
则输出显示错误



## §. 基础知识题 – 与cout有关的成员函数的基本使用

### 2. 用于字符输出控制的流成员函数

例13: cout.precision()

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

int main()
{
    double d = 123.456789123456;
    cout.setf(ios::fixed);
    cout.precision(10);
    cout << d << '*' << endl;
    cout.unsetf(ios::fixed); // 使用cout. 函数名
    cout.setf(ios::scientific);
    cout.precision(10);
    cout << d << '*' << endl;

    return 0;
}
```

将程序补充完整，得到期望的运行结果：

Microsoft Visual Studio 调试控制台

```
123.4567891235*
1.2345678912e+02*
```

```
123.4567891235*
1.2345678912e+02*
```



## §. 基础知识题 – 与cout有关的成员函数的基本使用

### 2. 用于字符输出控制的流成员函数

例14: cout.precision()

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

int main()
{
    double d = 123.456789123456;
    cout.setf(ios::scientific);
    cout.precision(10);
    cout << d << '*' << endl;
    cout.unsetf(ios::scientific); // 使用cout. 函数名
    cout.setf(ios::fixed);
    cout.precision(10);
    cout << d << '*' << endl;

    return 0;
}
```

将程序补充完整，得到期望的运行结果：

Microsoft Visual Studio 调试控制台

1. 2345678912e+02*	1. 2345678912e+02*
123. 4567891235*	123. 4567891235*



## §. 基础知识题 – sscanf与sprintf函数的基本使用

1. 将格式化输出的内容放入字符串中

例1：

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

int main()
{
    char str[80];
    int k=123, ret;
    double pi=3.1415925;

    ret = sprintf(str, "k=%-4d*pi=%.*f#", k, pi);
    printf("ret : %d\n", ret);
    printf("str : %s\n", str);

    return 0;
}
```

输出结果：

```
Microsoft Visual Studio 调试控制台
ret : 15
str : k=123 *pi=3.14#
```



## §. 基础知识题 – sscanf与sprintf函数的基本使用

1. 将格式化输出的内容放入字符串中

例2:

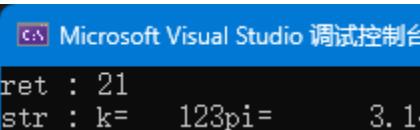
```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

int main()
{
    char str[80];
    int k=123, ret;
    double pi=3.1415925;

    ret = sprintf(str, "k=%dpi=%10.2f", k, pi);
    printf("ret : %d\n", ret);
    printf("str : %s\n", str);

    return 0;
}
```

输出结果:



```
Microsoft Visual Studio 调试控制台
ret : 21
str : k= 123pi= 3.14
```

结合例1和例2, sprintf的返回值是: 输出字符的个数



## §. 基础知识题 – sscanf与sprintf函数的基本使用

1. 将格式化输出的内容放入字符串中

例3:

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

int main()
{
    char str[15];
    int k=123, ret;
    double pi=3.1415925;

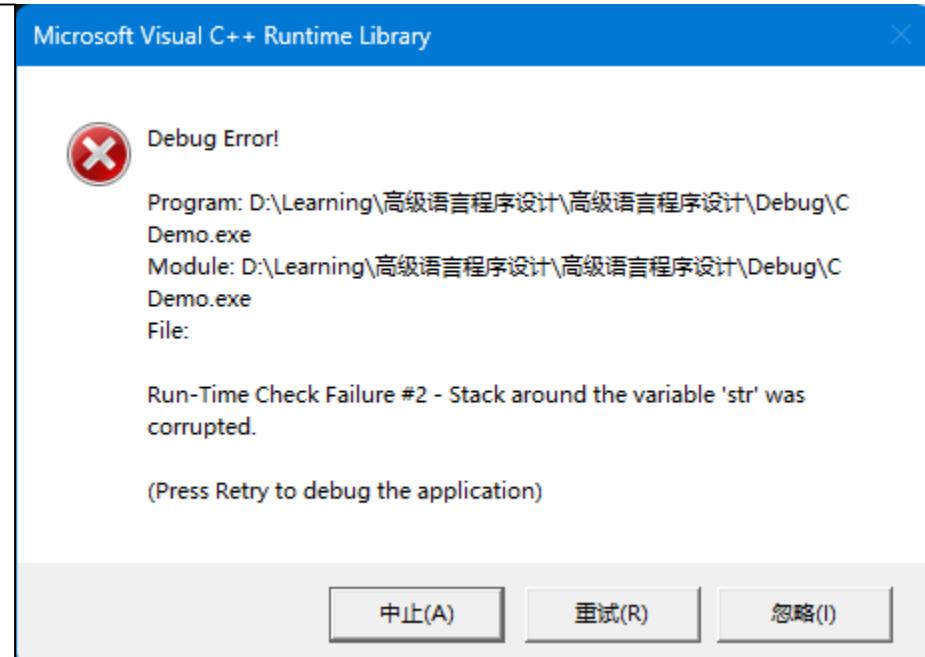
    ret = sprintf(str, "k=%-4d*pi=%.*f#", k, pi);
    printf("ret : %d\n", ret);
    printf("str : %s\n", str);

    return 0;
}
```

输出结果: VS: Microsoft Visual Studio 调试控制台  
ret : 15  
str : k=123 \*pi=3.14#

Dev: C:\Users\lenovo\Desktop\demo.exe  
ret : 15  
str : k=123 \*pi=3.14#

VS编译弹窗报错



结合例1/2/3, sprintf使用时对字符数组的要求是:  
字符数组要有足够空间容纳输出的数据(否则越界错)



## §. 基础知识题 – sscanf与sprintf函数的基本使用

### 2. 从字符串中进行格式化输入

例4:

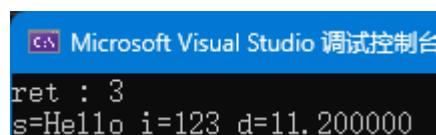
```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

int main()
{
    char str[80] = "Hello 123 11.2", s[10];
    int i, ret;
    double d;

    ret = sscanf(str, "%s %d %lf", s, &i, &d);
    printf("ret : %d\n", ret);
    printf("s=%s i=%d d=%f\n", s, i, d);

    return 0;
}
```

输出结果:



```
Microsoft Visual Studio 调试控制台
ret : 3
s=Hello i=123 d=11.200000
```



## §. 基础知识题 – sscanf与sprintf函数的基本使用

### 2. 从字符串中进行格式化输入

例5:

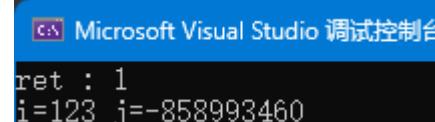
```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

int main()
{
    char str[80] = "123Hello";
    int i, j, ret;

    ret = sscanf(str, "%d%d", &i, &j);
    printf("ret : %d\n", ret);
    printf("i=%d j=%d\n", i, j);

    return 0;
}
```

输出结果:



```
Microsoft Visual Studio 调试控制台
ret : 1
i=123 j=-858993460
```

结合4例和例5, sscanf的返回值是: 正确读入的输入数据的个数



## §. 基础知识题 – sscanf与sprintf函数的基本使用

### 2. 从字符串中进行格式化输入

例6:

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

int main()
{
    char str[80] = "123 456";
    int i, j, ret;

    ret = sscanf(str, "%d%d", &i, &j);
    printf("ret : %d\n", ret);
    printf("i=%d j=%d\n", i, j);

    ret = sscanf(str, "%d%d", &j, &i); //顺序反
    printf("ret : %d\n", ret);
    printf("i=%d j=%d\n", i, j);

    return 0;
}
```

```
Microsoft Visual Studio 调试控制台
ret : 2
i=123 j=456
ret : 2
i=456 j=123
```

输出结果:

本例说明，str中的内容不可以被重复读取



## §. 基础知识题 – sscanf与sprintf函数的基本使用

### 3. 综合应用

例7：

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

int main()
{
    char str[80] = "123 456";
    int i, j, ret;

    ret = sscanf(str, "%d%d", &i, &j);
    printf("ret : %d\n", ret);
    printf("str=%s\ni=%d j=%d\n", str, i, j);

    ret = sprintf(str, "i=%d j=%d", i, j);
    printf("ret : %d\n", ret);
    printf("str=\"%s\"\n", str);

    return 0;
}
```

输出结果:



```
ret : 2
str=123 456
i=123 j=456
ret : 11
str="i=123 j=456"
```

本例说明，str中的内容可以被替换



## §. 基础知识题 – sscanf与sprintf函数的基本使用

### 3. 综合应用

例8:

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

int main()
{
    int x, w;
    printf("请输入[1..99999]间的整数及显示宽度[6..10]\n");
    scanf("%d %d", &x, &w); //不考虑输入错误
    printf("01234567890123456789\n"); //标尺

    char fmt[16];
    sprintf(fmt, "%%%dd*\n", w);
    printf(fmt, x);

    return 0;
}
```

1、 输入3 6， 输出：

```
Microsoft Visual Studio 调试控制台
请输入[1..99999]间的整数及显示宽度[6..10]
3 6
01234567890123456789
3*
```

2、 输入123 6， 输出：

```
Microsoft Visual Studio 调试控制台
请输入[1..99999]间的整数及显示宽度[6..10]
123 6
01234567890123456789
123*
```

3、 输入12345 6， 输出：

```
Microsoft Visual Studio 调试控制台
请输入[1..99999]间的整数及显示宽度[6..10]
12345 6
01234567890123456789
12345*
```

4、 输入3 9， 输出：

```
Microsoft Visual Studio 调试控制台
请输入[1..99999]间的整数及显示宽度[6..10]
3 9
01234567890123456789
3*
```

5、 输入123 9， 输出：

```
Microsoft Visual Studio 调试控制台
请输入[1..99999]间的整数及显示宽度[6..10]
123 9
01234567890123456789
123*
```

6、 输入12345 9， 输出：

```
Microsoft Visual Studio 调试控制台
请输入[1..99999]间的整数及显示宽度[6..10]
12345 9
01234567890123456789
12345*
```



# §. 基础知识题 – sscanf与sprintf函数的基本使用

## 3. 综合应用

例9：键盘输入一个长度[3..12]间字符串，再输入显示宽度[长度+1..20]，左对齐输出这个字符串（最后加\*分辨空格）

注：输入宽度小于等于串长则置为串长+1，不考虑其它输入错误

//给出相应的代码

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <string.h>

int main()
{
    char str[20] = { 0 };
    int w;

    printf("请输入长度[3..12]间的字符串及显示宽度[1en+1..20]\n");
    scanf("%s %d", str, &w);
    printf("01234567890123456789\n");
    if (w <= (int)strlen(str))
        w = strlen(str) + 1;
    char fmt[8];

    sprintf(fmt, "%-%ds*", w);
    printf(fmt, str);

    return 0;
}
```

1、输入abc 12，输出：

```
Microsoft Visual Studio 调试控制台
请输入长度[3..12]间的字符串及显示宽度[1en+1..20]
abc 12
01234567890123456789
abc *
```

2、输入abc 2，输出：

```
Microsoft Visual Studio 调试控制台
请输入长度[3..12]间的字符串及显示宽度[1en+1..20]
abc 2
01234567890123456789
abc *
```

3、输入abcdefg 3，输出：

```
Microsoft Visual Studio 调试控制台
请输入长度[3..12]间的字符串及显示宽度[1en+1..20]
abcdefg 3
01234567890123456789
abcdefg *
```

4、输入abcdefg 13，输出：

```
Microsoft Visual Studio 调试控制台
请输入长度[3..12]间的字符串及显示宽度[1en+1..20]
abcdefg 13
01234567890123456789
abcdefg *
```



## §. 基础知识题 – sscanf与sprintf函数的基本使用

### 3. 综合应用

例10：键盘输入一个double型数据，再输入总显示宽度及小数点后的位数，右对齐输出这个字符串（最后加\*分辨空格）

注：不考虑其它输入错误

//给出相应的代码

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

int main()
{
    double x;
    int w, h;

    printf("请输入double型数据及显示总宽度、小数点后位数\n");
    scanf("%lf%d%d", &x, &w, &h);
    printf("01234567890123456789\n");
    char fmt[20];

    sprintf(fmt, "%d.%dlf*", w, h);
    printf(fmt, x);

    return 0;
}
```

1、输入12.34 9 5，输出：

```
Microsoft Visual Studio 调试控制台
请输入double型数据及显示总宽度、小数点后位数
12.34 9 5
01234567890123456789
12.34000*
```

2、输入123.456789 12 2，输出：

```
Microsoft Visual Studio 调试控制台
请输入double型数据及显示总宽度、小数点后位数
123.456789 12 2
01234567890123456789
123.46*
```

3、输入12345678.9 5 2，输出：

```
Microsoft Visual Studio 调试控制台
请输入double型数据及显示总宽度、小数点后位数
12345678.9 5 2
01234567890123456789
12345678.90*
```

4、输入12345678.9 5 0，输出：

```
Microsoft Visual Studio 调试控制台
请输入double型数据及显示总宽度、小数点后位数
12345678.9 5 0
01234567890123456789
12345679*
```