



## § 14. C和C++知识点补充

### 1. 位运算

#### 1.1. 位运算的基本概念

##### 1.1.1. 字节和位

字节: byte, 计算机中数据表示的**基本**单位

位 : bit, 计算机中数据表示的**最小**单位

1 byte = 8 bit

##### 1.1.2. 位运算

以bit为单位进行数据的运算

##### 1.1.3. 位运算的基本方法

★ 按位进行 (只有0、1)

★ 要求运算数据长度相等, 若不等, 则右对齐, 按**符号位**补齐左边

再次强调:

有符号数: 符号位是最高位 (0/1)

无符号数: 符号位是0

char a=0x37;	0000 0000 0000 0000 0000 0000 0011 0111
short b=0x1234;	0000 0000 0000 0000 0001 0010 0011 0100
char a=0xA7;	1111 1111 1111 1111 1111 1111 1010 0111
short b=0x8341;	1111 1111 1111 1111 1000 0011 0100 0001
unsigned char a=0xA7;	0000 0000 0000 0000 0000 0000 1010 0111
short b=0x8341;	1111 1111 1111 1111 1000 0011 0100 0001

★ 数在计算机内是用补码表示的

★ 整型提升适用于位运算



## § 14. C和C++知识点补充

### 1. 位运算

#### 1.2. 常用的位运算

##### 1.2.1. 与(&)

运算规则：遇0得0

例：char a=3, b=5; 求a&b

```
0000 0011
& 0000 0101
0000 0001          a&b=1
```

例：char a=0x87; short b=0x9c52; 求a&b

```
1111 1111 1000 0111
& 1001 1100 0101 0010
1001 1100 0000 0010  a&b=0x9c02 (-25598)
```

例：unsigned char a=0x87; short b=0x9c52; 求a&b

```
0000 0000 1000 0111
& 1001 1100 0101 0010
0000 0000 0000 0010  a&b=0x2
```

例：char a=0xb6, b=0xc2; 求a&b

```
1011 0110
& 1100 0010
1000 0010          a&b=0x82 (-126)
```

● 竖式简写，实际应该32bit

● hex结果简写，省略了前面若干f

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

int main()
{
    char a1 = 3, b1 = 5;
    //问题：为什么打印a/b的值需要转int?
    cout << "a=" << (int)a1 << " b=" << (int)b1 << " a&b=" << (a1&b1) << endl;
    cout << sizeof(a1&b1) << " " << typeid(a1&b1).name() << endl;

    char a2 = 0x87;
    short b2 = 0x9c52;
    cout << "a=0x" << hex << (int)a2 << " b=0x" << b2 << " a&b=0x" << (a2&b2)
        << " a&b=0x" << short(a2&b2) << " " << dec << " a&b=" << (a2&b2) << endl;

    unsigned char a3 = 0x87;
    short b3 = 0x9c52;
    cout << "a=0x" << hex << (int)a3 << " b=0x" << b3 << " a&b=0x" << (a3&b3) << endl;

    char a4 = 0xb6, b4 = 0xc2;
    cout << "a=0x" << hex << (int)a4 << " b=0x" << (int)b4 << " a&b=0x" << (a4&b4)
        << " " << dec << " a&b=" << (a4&b4) << endl;

    return 0;
}
```

Microsoft Visual Studio 调试控制台

读懂运行结果!!!

```
a=3 b=5 a&b=1
4 int
a=0xffffffff87 b=0x9c52 a&b=0xffff9c02 a&b=0x9c02 a&b=-25598
a=0x87 b=0x9c52 a&b=0x2
a=0xffffffb6 b=0xfffffc2 a&b=0xffff82 a&b=-126
```



## § 14. C和C++知识点补充

### 1. 位运算

#### 1.2. 常用的位运算

##### 1.2.1. 与(&)

运算规则：遇0得0

应用：

##### ★ 清零

例：char a=0xb6;现要求将该数清零，则：

1011 0110

& 0?00 ?00?    要清零数为1的位，本数对应位为0

0000 0000

##### ● 竖式简写，实际应该32bit

a&0x0    a&0x1    a&0x8    a&0x9

a&0x40   a&0x41   a&0x48   a&0x49

##### ★ 取指定位

例：char a=0xb6;现要求只保留低4位，

而高4位清0，则：

1011 0110

& 0000 1111    要保留的位，本数对应位为1

0000 0110

##### ● 竖式简写，实际应该32bit

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

int main()
{
    /* &的应用：清零 */
    char a1=0xb6;
    cout << "char a=" << hex << (int)a1 << endl
         << "    a&0x0 =" << dec << (a1&0x0) << endl
         << "    a&0x1 =" << dec << (a1&0x1) << endl
         << "    a&0x8 =" << dec << (a1&0x8) << endl
         << "    a&0x9 =" << dec << (a1&0x9) << endl
         << "    a&0x40=" << dec << (a1&0x40) << endl
         << "    a&0x41=" << dec << (a1&0x41) << endl
         << "    a&0x48=" << dec << (a1&0x48) << endl
         << "    a&0x49=" << dec << (a1&0x49) << endl;

    /* &的应用：取指定位 */
    char a2=0xb6;
    cout << "char a=0x" << hex << (int)a2
         << "    a&0x0F=" << dec << (a2&0x0F)
         << endl;

    return 0;
}
```

读懂运行结果!!!

```
Microsoft Visual Studio 调试控制台
char a=fffffb6
a&0x0 =0
a&0x1 =0
a&0x8 =0
a&0x9 =0
a&0x40=0
a&0x41=0
a&0x48=0
a&0x49=0
char a=0xfffffb6 a&0x0F=6
```



## § 14. C和C++知识点补充

### 1. 位运算

#### 1.2. 常用的位运算

##### 1.2.2. 或(|)

运算规则：遇1得1

例：char a=3, b=5; 求a|b

```
0000 0011
| 0000 0101
0000 0111    a|b=7
```

例：char a=3; short b=5; 求a|b

```
0000 0000 0000 0011
| 0000 0000 0000 0101
0000 0000 0000 0111    a|b=7
```

例：char a=0xb6, b=0xc2; 则a|b

```
1011 0110
| 1100 0010
1111 0110    a|b=0xF6    有符号10进制：-10
```

应用：★ 设定某些位为1

例：char a=0xb6; 要求1,4位设为1, 其它不变

```
1011 0110
| 0000 1001    要设置的位，本数对应位为1
1011 1111    (0xBF)
```

● 竖式简写，实际应该32bit

● hex结果简写，省略了前面若干f

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

int main()
{
    char a1=3, b1=5;
    cout << "a=" << (int)a1 << " b=" << (int)b1 << " a|b=" << (a1|b1) << endl;

    char a2=3;
    short b2=5;
    cout << "a=" << (int)a2 << " b=" << b2 << " a|b=" << (a2|b2) << endl;

    char a3=0xb6, b3=0xc2;
    cout << "a=" << hex << (int)a3 << " b=" << (int)b3;
    cout << " a|b=0x" << hex << (a3|b3) << " " << dec << (a3|b3) << endl;

    /* |的应用，将1、4 bit位设为1，其它不变 */
    char a4=0xb6;
    cout << "a=" << hex << (int)a4 << " a|0x9=0x" << (a4|0x9) << endl;

    return 0;
}
```

读懂运行结果!!!

Microsoft Visual Studio 调试控制台

```
a=3 b=5 a|b=7
a=3 b=5 a|b=7
a=fffffb6 b=fffffc2 a|b=0xfffffff6 -10
a=fffffb6 a|0x9=0xfffffff6
```



## § 14. C和C++知识点补充

### 1. 位运算

#### 1.2. 常用的位运算

##### 1.2.3. 异或(^)

运算规则：相同为0，不同为1

例：char a=3, b=5; 求a^b

```
0000 0011
^ 0000 0101
-----
0000 0110    a^b=6
```

例：char a=3; short b=5; 求a^b

```
0000 0000 0000 0011
^ 0000 0000 0000 0101
-----
0000 0000 0000 0110    a^b=6
```

例：char a=0xb6, b=0xc2; 则a^b

```
1011 0110
^ 1100 0010
-----
0111 0100    a^b=0x74
```

有符号10进制：116

● 竖式简写，实际应该32bit

● hex结果简写，省略了前面若干f

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

int main()
{
    char a1=3, b1=5;
    cout << "a=" << (int)a1 << " b=" << (int)b1 << " a^b=" << (a1^b1) << endl;

    char a2=3;
    short b2=5;
    cout << "a=" << (int)a2 << " b=" << b2 << " a^b=" << (a2^b2) << endl;

    char a3=0xb6, b3=0xc2;
    cout << "a=" << hex << (int)a3 << " b=" << (int)b3;
    cout << " a^b=0x" << hex << (a3^b3) << " " << dec << (a3^b3) << endl;

    return 0;
}
```

读懂运行结果!!!

```
Microsoft Visual Studio 调试控制台
a=3 b=5 a^b=6
a=3 b=5 a^b=6
a=ffffffb6 b=fffffc2 a^b=0x74 116
```



## § 14. C和C++知识点补充

### 1. 位运算

#### 1.2. 常用的位运算

##### 1.2.3. 异或(^)

运算规则：相同为0，不同为1

应用：

#### ★ 特定位置翻转（0，1互换）

例：char a=0xb6；高4位翻转，低4位不变

```
1011 0110
^ 1111 0000  要翻转的位，本数对应位为1
-----
0100 0110
```

#### ★ 两数交换

例：char a=0xb6, b=0xc2；要求a, b互换

三步：a=a^b    b=b^a    a=a^b

(1) a=1011 0110

b=1100 0010

a=0111 0100    a=a^b=0x74

(2) b=1100 0010

a=0111 0100

b=1011 0110    b=b^a=0xb6

(3) a=0111 0100

b=1011 0110

a=1100 0010    a=a^b=0xc2

● 竖式简写，实际应该32bit

● hex结果简写，省略了前面若干f

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

int main()
{
    /* ^的应用：特定位置翻转（特别提醒：有无(char)的差别） */
    char a1=0xb6;
    cout << "a=" << hex << (int)a1 << " a^0xF0=0x" << (a1^(char)0xF0)
         << " a^0xF0=0x" << (a1^0xF0) << endl;

    /* ^的应用：两数交换 */
    char a2=0xb6, b2=0xc2;
    cout << "a=" << hex << (int)a2 << " b=" << (int)b2 << endl;
    a2 = a2^b2;
    b2 = b2^a2;
    a2 = a2^b2;
    cout << "a=" << hex << (int)a2 << " b=" << (int)b2 << endl;

    return 0;
}
```

读懂运行结果!!!

```
Microsoft Visual Studio 调试控制台
a=ffffffb6 a^0xF0=0x46 a^0xF0=0xffffffff46
a=ffffffb6 b=ffffffc2
a=ffffffc2 b=ffffffb6
```



## § 14. C和C++知识点补充

### 1. 位运算

#### 1.2. 常用的位运算

##### 1.2.3. 异或(^)

运算规则：相同为0，不同为1

应用：甲乙双方通过**公共通信**方式传递情报

##### ★ 前提条件

甲乙双方持有相同的密钥，用于加密/解密

##### ★ 基本步骤

- 甲：要发送的情报 ^ 密钥串 => 要传递的信息(公共可见)
- 通过公共通信方式将信息传给乙（其他人也能收到，看不懂）
- 乙：接收到的信息 ^ 密钥串 => 要阅读的情报

##### ★ 前提条件

```
const char* msg = "This is my student"; //甲持有
const char* secret_key = "周伯通黄药师郭靖黄蓉"; //甲乙持有
char encryped_msg[80], decryped_msg[80];
```

##### ★ 基本步骤

- 甲：情报msg，用密钥串secret\_key加密，得到encryped\_msg  
encrypt(msg, secret\_key, encryped\_msg);
- 公共渠道传递encryped\_msg
- 乙：收到encryped\_msg，用密钥串secret\_key解密，得到msg  
decrypted(encryped\_msg, secret\_key, decryped\_msg);

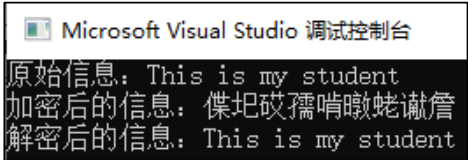
```
#include <iostream>
using namespace std;
void encrypt(const char* msg, const char* secret_key, char *encryped_msg)
{
    const char* p1 = msg, * p2 = secret_key;
    char* p3 = encryped_msg;
    /* 加密 */
    for (; *p1; p1++, p2++, p3++)
        *p3 = *p1 ^ *p2;
    *p3 = 0;
}

void decrypted(const char* encryped_msg, const char* secret_key, char* decryped_msg)
{
    const char* p1 = encryped_msg, * p2 = secret_key;
    char* p3 = decryped_msg;
    /* 解密(与解密操作完全一致) */
    for (; *p1; p1++, p2++, p3++)
        *p3 = *p1 ^ *p2;
    *p3 = 0;
}

int main()
{
    const char* msg = "This is my student";
    const char* secret_key = "周伯通黄药师郭靖黄蓉";
    char encryped_msg[80], decryped_msg[80];

    cout << "原始信息: " << msg << endl;
    encrypt(msg, secret_key, encryped_msg);
    cout << "加密后的信息: " << encryped_msg << endl; //这个信息允许公共传播
    decrypted(encryped_msg, secret_key, decryped_msg);
    cout << "解密后的信息: " << decryped_msg << endl;

    return 0;
}
```



原始信息、密钥串、加密信息，  
任意两个可以还原出第三个，  
因此要注意保护密钥串



## § 14. C和C++知识点补充

### 1. 位运算

#### 1.2. 常用的位运算

##### 1.2.4. 取反(~)

运算规则：0/1互反

例：char a=0x5c; 求~a

a=0101 1100

~a=1010 0011    ~a=0xa3    有符号10进制：-93

- ~a简写，实际应该32bit
- hex结果简写，省略了前面若干f

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

int main()
{
    char a=0x5c;
    cout << "a=" << hex << (int)a
         << " ~a=0x" << (~a) << " " << dec << (~a) << endl;

    return 0;
}
```

读懂运行结果!!!

Microsoft Visual Studio 调试控制台  
a=5c ~a=0xfffffa3 -93





## § 14. C和C++知识点补充

### 1. 位运算

#### 1.2. 常用的位运算

##### 1.2.5. 左移(<<)

运算规则：左移数据，右补0

例：char a=0x12;

a=0001 0010

0010 0100 a<<1=0x24

0100 1000 a<<2=0x48

1001 0000 a<<3=0x90

● 二进制结果简写，应32bit

例：int b=0x12;

a<<1=0x24

a<<2=0x48

a<<3=0x90

0x12 = 18

0x24 = 36

0x48 = 72

0x90 = -112

无符号:144

0x24 = 36

0x48 = 72

0x90 = 144

```
#include <iostream>
using namespace std;
int main()
{
    /* char型 */
    char a=0x12;
    cout << "a=0x" << hex << int(a) << " " << dec << int(a) << endl;
    cout << "a<<1=0x" << hex << (a<<1) << " " << dec << (a<<1) << endl;
    cout << "a<<2=0x" << hex << (a<<2) << " " << dec << (a<<2) << endl;
    /* 特别提醒，看懂两个a<<3的差异 */
    cout << "1. a<<3=0x" << hex << (a<<3) << " " << dec << (a<<3) << endl;
    cout << "2. a<<3=0x" << hex << (int)(char)(a<<3) << " "
        << dec << (int)(char)(a<<3) << endl;

    cout << endl;

    /* 直接是int型的情况 */
    int b=0x12;
    cout << "b=0x" << hex << b << " " << dec << b << endl;
    cout << "b<<1=0x" << hex << (b<<1) << " " << dec << (b<<1) << endl;
    cout << "b<<2=0x" << hex << (b<<2) << " " << dec << (b<<2) << endl;
    cout << "b<<3=0x" << hex << (b<<3) << " " << dec << (b<<3) << endl;

    return 0;
}
```

1、看懂运行结果!!!  
2、为什么两个a<<3不同?

Microsoft Visual Studio 调试控制台

```
a=0x12 18
a<<1=0x24 36
a<<2=0x48 72
1. a<<3=0x90 144
2. a<<3=0xffffffff90 -112

b=0x12 18
b<<1=0x24 36
b<<2=0x48 72
b<<3=0x90 144
```



## § 14. C和C++知识点补充

### 1. 位运算

#### 1.2. 常用的位运算

##### 1.2.5. 左移(<<)

运算规则：左移数据，右补0

例：char a=0x12; 求a<<3

a=0001 0010

1001 0000 a<<3=0x90 有符号 -112

无符号144

★ 在不溢出(1不被舍去)的情况下，左移n位等于乘2的n次方(当做无符号数理解)

例：char a=0x12; 求a<<4

a=0001 0010

1 0010 0000 a<<4=0x20 0x12=18 0x20=32  
32+256(2<sup>8</sup>)=288=18\*16(2<sup>4</sup>)

例：char a=0x9c; 求a<<2

a=1001 1100

10 0111 0000 a<<2=0x70 0x9c=156 0x70=112  
112+512(2<sup>9</sup>)=624=156\*4(2<sup>2</sup>)

例：char a=0xc2; 求a<<2

a=1100 0010

11 0000 1000 a<<2=0x8 0xc2=194 0x8=8  
8+512(2<sup>9</sup>)+256(2<sup>8</sup>)=776=194\*4(2<sup>2</sup>)

```
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
    char a1=0x12;
    cout << "a<<4=0x" << hex << (int)(char)(a1<<4) << " "
         << dec << (int)(char)(a1<<4) << endl;

    char a2=0x9c;
    cout << "a<<2=0x" << hex << (int)(char)(a2<<2) << " "
         << dec << (int)(char)(a2<<2) << endl;

    char a3=0xc2;
    cout << "a<<2=0x" << hex << (int)(char)(a3<<2) << " "
         << dec << (int)(char)(a3<<2) << endl;

    return 0;
}
```

读懂运行结果!!!

Microsoft Visual Studio 调试控制台

```
a<<4=0x20 32
a<<2=0x70 112
a<<2=0x8 8
```



## § 14. C和C++知识点补充

### 1. 位运算

#### 1.2. 常用的位运算

##### 1.2.6. 右移(>>)

运算规则：右移数据，左补0（逻辑右移）

右移数据，左补符号位（算术右移）<= C/C++的位运算时算术右移

★ 算术右移，无符号数仍补0

例：char a=0x18;

a=0001 1000

0000 1100 a>>1=0xc

0000 0110 a>>2=0x6

0000 0011 a>>3=0x3

0000 0001 a>>4=0x1

0x18 = 24

0xc = 12

0x6 = 6

0x3 = 3

0x1 = 1

● 二进制结果简写，忽略24bit的0

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

int main()
{
    char a=0x18;
    cout << "a>>1=0x" << hex << (a>>1)
          << " " << dec << (a>>1) << endl;
    cout << "a>>2=0x" << hex << (a>>2)
          << " " << dec << (a>>2) << endl;
    cout << "a>>3=0x" << hex << (a>>3)
          << " " << dec << (a>>3) << endl;
    cout << "a>>4=0x" << hex << (a>>4)
          << " " << dec << (a>>4) << endl;

    return 0;
}
```

读懂运行结果!!!

Microsoft Visual Studio 调试控制台

```
a>>1=0xc 12
a>>2=0x6 6
a>>3=0x3 3
a>>4=0x1 1
```



## § 14. C和C++知识点补充

### 1. 位运算

#### 1.2. 常用的位运算

##### 1.2.6. 右移(>>)

运算规则：右移数据，左补0（逻辑右移）

右移数据，左补符号位（算术右移）<= C/C++的位运算时算术右移

★ 算术右移，无符号数仍补0

★ 在不溢出(1不被舍去)的情况下，右移n位等于除2的n次方  
(当作有符号数理解)

例：char a=0x84; 求a>>1

a=1000 0100

1100 0010 a>>1=0xc2

最高位为1，若作为符号位，则表示负数

0x84 = -124

无符号: 132

0xc2 = -62

无符号: 194

a=1000 0100

-) 1

1000 0011

0111 1100

补码 => 原码

(1) 减1

(2) 取反

(3) 绝对值

|a|=124

|a>>1|=62

a=1100 0010

-) 1

1100 0001

0011 1110

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

```
int main()
{
```

```
    char a=0x84; //有符号数补1 !!!
```

```
    cout << "a=0x" << hex << int(a) << " " << dec << int(a) << endl;
```

```
    cout << "a>>1=0x" << hex << (a>>1) << " " << dec << (a>>1) << endl;
```

```
    cout << endl;
```

```
    unsigned char b=0x84; //无符号数补0 !!!
```

```
    cout << "b=0x" << hex << int(b) << " " << dec << int(b) << endl;
```

```
    cout << "b>>1=0x" << hex << (b>>1) << " " << dec << (b>>1) << endl;
```

```
    return 0;
```

```
}
```

读懂运行结果!!!

Microsoft Visual Studio 调试控制台

```
a=0xffffffff84 -124
a>>1=0xffffffffc2 -62

b=0x84 132
b>>1=0x42 66
```



## § 14. C和C++知识点补充

### 1. 位运算

#### 1.2. 常用的位运算

##### 1.2.6. 右移(>>)

例: char a=0x18;

a=0001 1000	(24)
0000 1100 a>>1=0xc	(12)
0000 0110 a>>2=0x6	(6)
0000 0011 a>>3=0x3	(3)
0000 0001 a>>4=0x1	(1) 溢出舍去了1
0000 0000 a>>5=0x0	(0) 再次溢出舍去1
0000 0000 a>>6=0x0	(0) >>6以上都是0

例: char a=0x84;

a=1000 0100	(-124)
1100 0010 a>>1=0xc2	(-62)
1110 0001 a>>2=0xe1	(-31)
1111 0000 a>>3=0xf0	(-16) 溢出舍去了1
1111 1000 a>>4=0xf8	(-8)
1111 1100 a>>5=0xfc	(-4)
1111 1110 a>>6=0xfe	(-2)
1111 1111 a>>7=0xff	(-1)
1111 1111 a>>8=0xff	(-1) >>8以上都是-1

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

```
int main()
{
    char a=0x18;
    int i;

    for(i=1; i<=6; i++) {
        a = a>>1;
        cout << "a>>" << i << "=0x" << hex << int(a) << " " << dec << int(a) << endl;
    }
    return 0;
}
```

读懂运行结果!!!

Microsoft Visual Studio 调试控制台

```
a>>1=0xc 12
a>>2=0x6 6
a>>3=0x3 3
a>>4=0x1 1
a>>5=0x0 0
a>>6=0x0 0
```

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

```
int main()
{
    char a=0x84;
    int i;

    for(i=1; i<=8; i++) {
        a = a>>1;
        cout << "a>>" << i << "=0x" << hex << int(a) << " " << dec << int(a) << endl;
    }
    return 0;
}
```

读懂运行结果!!!

Microsoft Visual Studio 调试控制台

```
a>>1=0xffffffffc2 -62
a>>2=0xffffffe1 -31
a>>3=0xfffffff0 -16
a>>4=0xfffffff8 -8
a>>5=0xfffffff4 -4
a>>6=0xffffffe2 -2
a>>7=0xfffffff1 -1
a>>8=0xfffffff1 -1
```



## § 14. C和C++知识点补充

### 1. 位运算

#### 1.3. 复合位运算符

&=    |=    ^=    <<=    >>=

- ★ 将上例中 `a = a>>1;`  
改为 `a >>= 1;` 结果相同

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

```
int main()
{
    char a=0x18;
    int i;

    for(i=1; i<=6; i++) {
        a >>= 1;
        cout << "a>>" << i << "=0x" << hex << int(a) << " " << dec << int(a) << endl;
    }
    return 0;
}
```

读懂运行结果!!!

Microsoft Visual Studio 调试控制台

```
a>>1=0xc 12
a>>2=0x6 6
a>>3=0x3 3
a>>4=0x1 1
a>>5=0x0 0
a>>6=0x0 0
```

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

```
int main()
{
    char a=0x84;
    int i;

    for(i=1; i<=8; i++) {
        a >>= 1;
        cout << "a>>" << i << "=0x" << hex << int(a) << " " << dec << int(a) << endl;
    }
    return 0;
}
```

读懂运行结果!!!

Microsoft Visual Studio 调试控制台

```
a>>1=0xffffffffc2 -62
a>>2=0xffffffe1 -31
a>>3=0xfffffff0 -16
a>>4=0xfffffff8 -8
a>>5=0xfffffff4 -4
a>>6=0xffffffe -2
a>>7=0xfffffff -1
a>>8=0xfffffff -1
```



## § 14. C和C++知识点补充

### 2. 带参数的main函数

#### 2.1. 引入

可执行文件运行时，目前只能简单的运行，如果能加上参数，则使用中可以更灵活

#### 例1：两数交换(常规方法)

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

int main()
{
    int a, b, t;
    cout << "请输入两个整数" << endl;
    cin >> a >> b;
    cout << "交换前: a=" << a << "    b=" << b << endl;
    t = a;
    a = b;
    b = t;
    cout << "交换后: a=" << a << "    b=" << b << endl;

    return 0;
}
```



## § 14. C和C++知识点补充

### 2. 带参数的main函数

#### 2.1. 引入

可执行文件运行时，目前只能简单的运行，如果能加上参数，则使用中可以更灵活

#### 2.2. 带参数的main函数的定义形式

```
int main(int argc, char **argv)
```

```
int main(int argc, char *argv[])
```

两者均可

#### ★ 参数解释

argc: 参数的个数，若不带参数，则为1(自身)

argv: 参数的内容，用指针数组表示，每个元素是一个字符串(char \*)，最后一个为 NULL

- argv数组共有argc+1个元素，下标[0]~[argc]（例如：argc为3，则argv[0]是自身，argv[3]是NULL）
- 参数名argc/argv可变，类型不能变（例如：int ac, char \*\*av）
- VS系列可以 long ac, unsigned char \*\*av, gcc系列不可以，因此不建议其它类型





## § 14. C和C++知识点补充

### 2. 带参数的main函数

#### 2.3. 使用

例2: 两数交换(main函数带参数方法)

```
#include <iostream>
#include <cstdlib> //atoi函数用到
using namespace std;
int main(int argc, char *argv[])
{
    int a, b, t;

    cout << "argc=" << argc << endl;
    cout << "argv[0]=" << argv[0] << endl;
    a = atoi(argv[1]); //atoi是将字符串转为整数的函数
    b = atoi(argv[2]);

    cout << "交换前:a=" << a << " b=" << b << endl;
    t = a;
    a = b;
    b = t;
    cout << "交换后:a=" << a << " b=" << b << endl;

    return 0;
}
```

假设编译后形成demo.exe

1、集成环境运行 (出错,为什么?)

2、命令行运行

demo (出错,为什么?)

demo 10 (出错,为什么?)

demo 10 15 (正确)

demo 10 15 20 (正确)



## § 14. C和C++知识点补充

### 2. 带参数的main函数

#### 2.3. 使用

例3: 两数交换(main函数带参数方法 - 改进)

```
#include <iostream>
#include <cstdlib> //atoi函数用到
using namespace std;
int main(int argc, char *argv[])
{
    int a, b, t;
    if (argc<3) { /* 参数不足3个则出现提示 */
        cout << "请带两个整数作为参数"<< endl;
        return -1;
    }
    for (t=0; t<argc; t++) /* 打印所有的参数值 */
        cout << "argv[" << t << "]= " << argv[t] << endl;
    a = atoi(argv[1]); //atoi是将字符串转为整数的函数
    b = atoi(argv[2]);
    cout << "交换前: a=" << a << " b=" << b << endl;
    t = a;
    a = b;
    b = t;
    cout << "交换后: a=" << a << " b=" << b << endl;
    return 0;
}
```

假设编译后形成形成  
demo.exe

1、集成环境运行

2、命令行运行

demo

demo 10

demo 10 15

demo 10 15 20



## § 14. C和C++知识点补充

### 2. 带参数的main函数

#### 2.4. 综合应用

##### 例4: 作业相似度检查程序的参数设计

###### (1) 学生的匹配

要求能在两个特定的学生之间检查

某个特定学生和全体学生之间检查

全体学生之间相互检查

###### (2) 文件的匹配

要求既可以是单文件，也可以全部文件

###### (3) 相似度设置

要求值在60-100间浮动

###### (4) 输出方式

可选文件/屏幕(通过重定向实现)

假设Linux下编译后形成形成 check

```
./check 1234567 7654321 12-b2.cpp 80
```

```
./check 1234567 7654321 all 80
```

```
./check 1234567 all 12-b2.cpp 75 > result.txt
```

```
./check 1234567 all all 85
```

```
./check all all all 85 > final.txt
```

★ 1234567 和 7654321 是两个学号



## § 14. C和C++知识点补充

### 2. 带参数的main函数

#### 2.5. 参数个数不固定的带参main函数

例5: 在Windows的命令行下输入 ping, 可以看到ping 命令的很多选项, 下列命令都是正确的

```
ping 10.10.108.117
```

```
ping -t 10.10.108.117
```

```
ping -n 10 10.10.108.117
```

```
ping -n 10 -l 50000 192.168.80.230
```

```
ping -t -l 50000 192.168.80.230
```

```
ping -l 50000 -t 192.168.80.230
```

★ 参数个数不固定, 且部分参数要2个一组

★ 参数出现顺序任意

★ 具体通过作业方式来理解实现过程

思考: 如果输入ping后用人机交互形式, 该如何做? 从用户操作方便性角度而言, 可行吗?

```
cmd.exe

D:\>ping

用法: ping [-t] [-a] [-n count] [-l size] [-f] [-i TTL] [-v TOS]
          [-r count] [-s count] [[-j host-list] | [-k host-list]]
          [-w timeout] [-R] [-S srcaddr] [-c compartment] [-p]
          [-4] [-6] target_name

选项:
    -t          Ping 指定的主机, 直到停止。
                若要查看统计信息并继续操作, 请键入 Ctrl+Break;
                若要停止, 请键入 Ctrl+C。
    -a          将地址解析为主机名。
    -n count    要发送的回显请求数。
    -l size     发送缓冲区大小。
    -f          在数据包中设置“不分段”标记(仅适用于 IPv4)。
    -i TTL      生存时间。
    -v TOS      服务类型(仅适用于 IPv4。该设置已被弃用,
                对 IP 标头中的服务类型字段没有任何影响)。
    -r count    记录计数跃点的路由(仅适用于 IPv4)。
    -s count    计数跃点的时间戳(仅适用于 IPv4)。
    -j host-list 与主机列表一起使用的松散源路由(仅适用于 IPv4)。
    -k host-list 与主机列表一起使用的严格源路由(仅适用于 IPv4)。
    -w timeout  等待每次回复的超时时间(毫秒)。
    -R          同样使用路由标头测试反向路由(仅适用于 IPv6)。
                根据 RFC 5095, 已弃用此路由标头。
                如果使用此标头, 某些系统可能丢弃回显请求。
    -S srcaddr  要使用的源地址。
    -c compartment 路由隔离舱标识符。
    -p          Ping Hyper-V 网络虚拟化提供程序地址。
    -4          强制使用 IPv4。
    -6          强制使用 IPv6。

D:\>
```



## § 14. C和C++知识点补充

### 2. 带参数的main函数

#### 2.6. 带参数的main函数的扩展形式 (仅了解)

形式: `int main(int argc, char **argv, char **env)`

或: `char *env[]`

参数解释: {  
  `argc`: 同前  
  `argv`: 同前  
  `env`: 操作系统的环境变量, 用指针数组来表示, 每个元素是一个字符串 (`char *`), 最后一个元素是 `NULL`

使用: 需要判断/取操作系统的某些设置时才用到

例6: 取操作系统的环境变量(在Windows/Linux下分别运行)

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

int main(int argc, char **argv, char **env)
{
    int i;
    for (i=0; env[i]; i++)
        cout<< "env[" << i << "]=" << env[i] << endl;

    return 0;
}
```

拓展问题: 如何在Windows/Linux下  
增加一个环境变量?



## § 14. C和C++知识点补充

### 2. 带参数的main函数

#### 2.7. 带参数main函数的作用

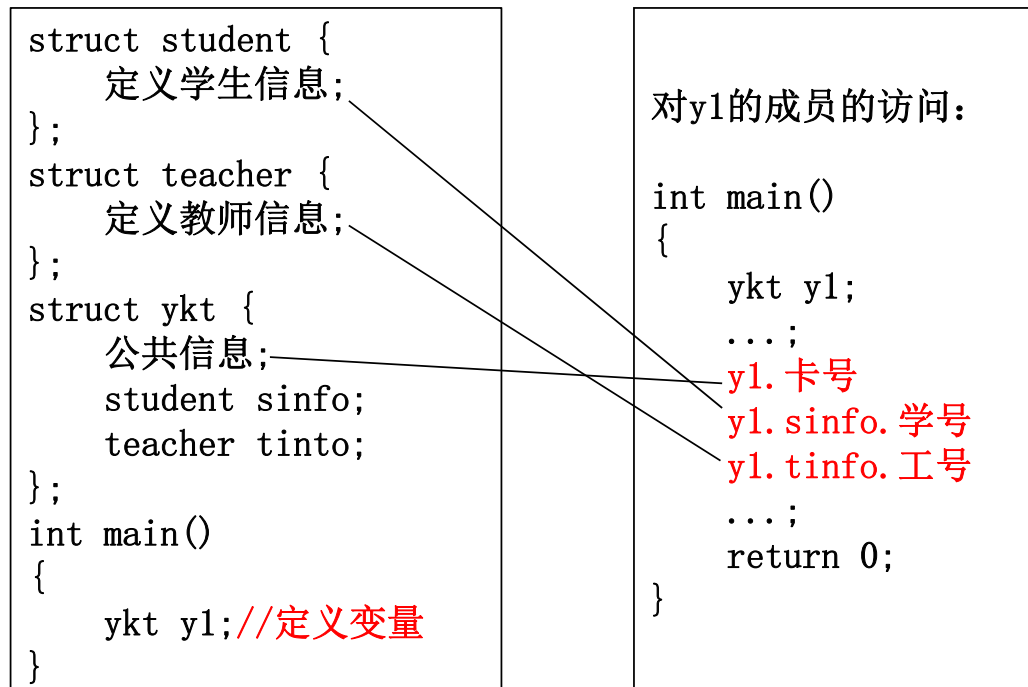
	带参main函数方式	运行时键盘交互方式
运行方法	运行命令后直接跟各参数，不再进行人机交互	运行命令后进入人机交互
是否需要人机交互	不需要人机交互	需要人机交互 (可用输入重定向方式取消人机交互，但不方便)
适用程序	1、守护进程(开机自启动) 2、后台运行程序 3、类似ping的不定参数形式命令，必须用此形式	前端程序



## § 14. C和C++知识点补充

### 3. 共用体

例： 定义一个用于一卡通管理系统的结构，要求包含卡号、余额、消费限额、消费密码等**公共信息**，此外，若持卡人是学生，要包含学号、姓名、专业等**学生特有的信息**，若持卡人是教师，则包含工号、姓名、职称等**教师特有的信息**



缺陷：无论持卡人何种身份sinfo和tinfo中必然有一个是不需要填写任何信息的，从而导致存储空间的浪费

解决：能否使sinfo/tinfo共用一段空间，当持卡人是学生时，这段空间按student方式访问，当持卡人是教师时按teacher方式访问

=> (共用体)



## § 14. C和C++知识点补充

### 3. 共用体

```
union 共用体名 {  
    共用体成员1 (类型名 成员名)  
    ...  
    共用体成员n (类型名 成员名)  
};  
  
union data {  
    short a;  
    long b;  
    char c;  
};
```

- ★ 所有成员从同一内存开始，共用体的大小为其中占用空间最大的成员的大小
- ★ 给一个共用体成员赋值后，会覆盖其它成员的值，因此只有最后一次存放的成员是有效的
- ★ 其它所有定义、使用方法同结构体

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

```
struct data1 {  
    short a;  
    long b;  
    char c;  
};
```

```
union data2 {  
    short a;  
    long b;  
    char c;  
};
```

```
int main()  
{
```

```
    data1 d1;  
    data2 d2;
```

```
    cout << sizeof(d1) << ' ' << sizeof(d2) << endl;  
    cout << &d1.a << ' ' << &d1.b << ' ' << (void *)&d1.c << endl;  
    cout << &d2.a << ' ' << &d2.b << ' ' << (void *)&d2.c << endl;
```

```
    return 0;  
}
```

12: 所有成员所占空间之和(含填充字节)

4 : 所有成员中最大成员所占空间

d1	2000	a
	2001	
	2002	///
	2003	
	2004	b
	2005	
	2006	
	2007	
	2008	c
	2009	///
	2010	
	2011	

d2	3000	a	b	c
	3001			
	3002			
	3003			

12 4  
地址: X X+4 X+8  
地址: Y Y Y

Microsoft Visual Studio 调试控制台

```
12 4  
005BFE40 005BFE44 005BFE48  
005BFE34 005BFE34 005BFE34
```





## § 14. C和C++知识点补充

### 3. 共用体

★ 所有成员从同一内存开始，共用体的大小为其中占用空间最大的成员的大小

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

```
union data {
    int  a;
    short b;
    char  c;
};
```

```
int main()
{
    union data d;
    d.a=70000;
    cout << d.a << ' ' << d.b << ' ' << d.c << endl;
    d.b=7000;
    cout << d.a << ' ' << d.b << ' ' << d.c << endl;
    d.c='A';
    cout << d.a << ' ' << d.b << ' ' << d.c << endl;

    return 0;
}
```

70000 = 00000000 00000001 00010001 01110000

d: 低位在前存放

2000	01110000	a	b	c
2001	00010001			
2002	00000001			
2003	00000000			

70000 4464 p

72536 7000 X

72513 6977 A



## § 14. C和C++知识点补充

### 3. 共用体

★ 所有成员从同一内存开始，共用体的大小为其中占用空间最大的成员的大小

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

```
union data {
    int  a;
    short b;
    char  c;
};
```

```
int main()
{
```

```
    union data d;
```

```
    d.a=70000;
```

```
    cout << d.a << ' ' << d.b << ' ' << d.c << endl;
```

```
    d.b=7000;          70000 4464 p
```

```
    cout << d.a << ' ' << d.b << ' ' << d.c << endl;
```

```
    d.c='A';          72536 7000 X
```

```
    cout << d.a << ' ' << d.b << ' ' << d.c << endl;
```

```
          72513 6977 A
```

```
    return 0;
```

```
}
```

70000 = 00000000 00000001 00011011 01011000

d: 低位在前存放

2000

01011000

2001

00011011

2002

00000001

2003

00000000

c

b

a

7000 = 00011011 01011000



## § 14. C和C++知识点补充

### 3. 共用体

★ 所有成员从同一内存开始，共用体的大小为其中占用空间最大的成员的大小

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

```
union data {
    int  a;
    short b;
    char  c;
};
```

```
int main()
{
    union data d;
    d.a=70000;
    cout << d.a << ' ' << d.b << ' ' << d.c << endl;
    d.b=7000;
    cout << d.a << ' ' << d.b << ' ' << d.c << endl;
    d.c='A';
    cout << d.a << ' ' << d.b << ' ' << d.c << endl;

    return 0;
}
```

70000 = 00000000 00000001 00011011 01000001

d: 低位在前存放

2000	01000001	a	b	c
2001	00011011			
2002	00000001			
2003	00000000			

70000 4464 p

72536 7000 X

72513 6977 A

A = 01000001



## § 14. C和C++知识点补充

### 3. 共用体

★ 所有成员从同一内存开始，共用体的大小为其中占用空间最大的成员的大小

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

```
union data {
    int  a;
    short b;
    char  c;
};
```

```
int main()
{
```

```
    union data d;
    d.c='A';
```

```
    cout << d.a << ' ' << d.b << ' ' << d.c << endl;
```

```
    d.b=7000;    不确定 不确定 A
```

```
    cout << d.a << ' ' << d.b << ' ' << d.c << endl;
```

```
    d.a=70000;    不确定 7000 X
```

```
    cout << d.a << ' ' << d.b << ' ' << d.c << endl;
```

70000 4464 p

```
    return 0;
}
```

d:低位在前存放

2000	01000001
2001	???
2002	???
2003	???

d:低位在前存放

2000	01011000
2001	00011011
2002	???
2003	???

d:低位在前存放

2000	01110000
2001	00010001
2002	00000001
2003	00000000



## § 14. C和C++知识点补充

### 3. 共用体

★ 所有成员从同一内存开始，共用体的大小为其中占用空间最大的成员的大小

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

```
union data {
    int    a;
    short b;
    char  c;
};
```

```
int main()
{
    union data d;
    d.a=70000;
    cout << d.a << ' ' << d.b << ' ' << d.c << endl;
    d.b=7000;
    cout << d.a << ' ' << d.b << ' ' << d.c << endl;
    d.c='A';
    cout << d.a << ' ' << d.b << ' ' << d.c << endl;

    return 0;
}
```

70000 = 00000000 00000001 00010001 01110000

d: 低位在前存放

2000	01110000	a	b	c
2001	00010001			
2002	00000001			
2003	00000000			

d: 高位在前存放

2000	00000000	a	b	c
2001	00000001			
2002	00010001			
2003	01110000			

不同的CPU(和操作系统、编译器无关)有不同的字节序类型，这些字节序是指整数在内存中保存的顺序，称为主机序，常见的有两种：

★ Little endian: 将低序字节存储在起始地址，地址低位存储值的低位，地址高位存储值的高位(小头序/小字序)

★ Big endian : 将高序字节存储在起始地址，地址低位存储值的高位，地址高位存储值的低位(大头序/大字序)

思考：大字序系统中，本题的运行结果？

(注：本题无法通过Intel/AMD等小字序系统运行测试程序得到答案，需要手动计算)



## § 14. C和C++知识点补充

### 3. 共用体

例： 定义一个用于一卡通管理系统的结构，要求包含卡号、余额、消费限额、消费密码等**公共信息**，此外，若持卡人是学生，要包含学号、姓名、专业等**学生特有的信息**，若持卡人是教师，则包含工号、姓名、职称等**教师特有的信息**

```
struct student {  
    定义学生信息;  
};  
  
struct teacher {  
    定义教师信息;  
};  
  
struct ykt {  
    公共信息;  
    student sinfo;  
    teacher tinfo;  
};  
  
int main()  
{  
    ykt y1;//定义变量  
}
```

空间浪费

```
struct student {  
    定义学生信息;  
};  
struct teacher {  
    定义教师信息;  
};  
union owner {  
    student s;  
    teacher t;  
};  
struct ykt {  
    公共信息;  
    char type; //持卡人类别  
    owner info;  
};  
int main()  
{  
    ykt y1;//定义变量  
}
```

此处保证s/t  
共用一段空间

```
int main()  
{  
    ykt y1;//定义变量  
    ...;  
    y1. 卡号;  
    if (y1.type=='s') {  
        y1.info.s. 学号;  
    }  
    else {  
        y1.info.t. 工号;  
    }  
    ...;  
    return 0;  
}
```



## § 14. C和C++知识点补充

### 4. 条件编译

§ 13中已讲过，课件已单独下发



## § 14. C和C++知识点补充

### 5. 枚举类型

#### 5.1. 含义

当变量的取值在**有限范围**内且**离散**时(可一一列出), 称为枚举类型

**例: 性别、星期、月份、血型**

#### 5.2. 枚举类型的定义

```
enum 枚举类型名 {枚举元素1, ..., 枚举元素n};
```

```
enum sex {male, female};
```

```
enum week {sun, mon, tue, wed, thu, fri, sat};
```

```
enum month {Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov, Dec};
```

```
enum blood_type {A, B, O, AB};
```

★ 枚举类型和元素的命名同变量

★ 枚举元素也称**枚举常量**, 不是字符串, 不加", 作为整型常量处理, 值从0开始顺序递增, 也可自行指定, 在程序执行中不可变

```
enum week {sun, mon, tue, wed, thu, fri, sat};
```

0 1 2 3 4 5 6

```
enum week {sun=7, mon=1, tue, wed, thu, fri, sat};
```

7 1 2 3 4 5 6

```
enum week {sun=7, mon, tue, wed, thu, fri, sat};
```

7 8 9 10 11 12 13

★ 如果执行的常量值出现重叠, 不算错误 (**但会造成误解**)

```
enum week {sun=3, mon=1, tue, wed, thu, fri, sat};
```

3 1 2 3 4 5 6





## § 14. C和C++知识点补充

### 5. 枚举类型

#### 5.3. 枚举类型变量的定义

`enum` 枚举类型名 变量名 (红色阴影: 在C++下定义时, `enum`关键字可省略)

`enum` week w1, w2;

#### 5.4. 枚举类型变量的使用

赋值: w1=mon w2=fri w2=w1

★ 不能直接赋整型量, 需要进行强制类型转换

w1 = 3;

w1 = week(3);

w1 = (week)3;

错误

正确

正确

error C2440: “=”: 无法从“int”转换为“week”  
message : 强制转换为枚举类型要求显式强制转换(static\_cast、C 样式强制转换或函数样式强制转换)

In function 'int main()':  
[Error] invalid conversion from 'int' to 'week' [-fpermissive]

比较: w1==mon w2>sat w2>=w1

★ 按枚举常量对应的值进行比较

输出: 直接输出: (按整型输出)

w1 = wed;

cout << w1 << endl; 3

间接输出: (可以是自定义的任意格式)

```
switch(w1) {  
    case sun:  
        cout << "Sunday";  
        break;  
    ...  
}
```

```
if (w1 == sun)  
    cout << "星期天";  
else if (...)  
    ...
```

输入: 直接输入: 只能用C方式, 且不检查范围

week w1;

scanf("%d", &w1);

cin >> w1; 不允许

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)

间接输入: 可以多种格式

char s[80];

cin >> s;

if (!strcmp(s, "sun"))

w1=sun;

else if (...)

int w;

cin >> w;

w1=(week)w;



## § 14. C和C++知识点补充

### 5. 枚举类型

#### 5.4. 枚举类型变量的使用

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

enum month {Jan=1, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov, Dec};

int main()
{
    int i, m[13], *p;

    for (i=Jan; i<=Dec; i++)
        if (i==Feb)
            m[i] = 28;
        else if (i==Apr || i==Jun || i==Sep || i==Nov)
            m[i] = 30;
        else
            m[i] = 31;

    for (p=&m[1]; p<m+13; p++)
        cout << *p << " ";
    cout << endl;

    return 0;
}
```

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

int main()
{
    int i, m[13], *p;

    for (i=1; i<=12; i++)
        if (i==2)
            m[i] = 28;
        else if (i==4 || i==6 || i==9 || i==11)
            m[i] = 30;
        else
            m[i] = 31;

    for (p=&m[1]; p<m+13; p++)
        cout << *p << " ";
    cout << endl;

    return 0;
}
```

#### 特别说明:

- ★ enum枚举类型不属于一定要使用的概念，左右例子分别是使用/未使用enum的情况，功能相同
- ★ 请大家自行评估可读性并选择适合自己的风格，不强求



## § 14. C和C++知识点补充

### 6. 用typedef声明类型

#### 6.1. 含义

用新的名称来等价代替已有的数据类型

★ 不产生新类型，仅使原有类型有新的名称

★ 建议声明的新类型名称为大写，与系统类型区分

#### 6.2. 使用

声明名称：

typedef 已有类型 新名称;

typedef int INTEGER;

typedef struct student STUDENT; //C++无此需求

typedef int ARRAY[10];

typedef char \* STRING;

用新名称定义变量及等价对应关系：

INTEGER i, j;

STUDENT s1, s2[10], \*s3;

ARRAY a, b[5];

STRING p, x[10];

等价方式

int i, j;

student s1, s2[10], \*s3;

int a[10], b[5][10];

char \*p, \*x[10];

```
struct student {  
    ...  
};  
struct student s1; //C方式  
STUDENT s1; //C方式  
student s1; //C++方式
```

C方式的另一种小技巧，将  
无名结构体声明为student

```
typedef struct {  
    ...  
} student;  
student s1; //C方式
```



## § 14. C和C++知识点补充

### 6. 用typedef声明类型

#### 6.3. 声明新类型的一般步骤

- ① 以现有类型定义一个变量
- ② 将变量名替换为新类型名
- ③ 加typedef
- ④ 完成, 可定义新类型的变量

```
① int i;  
② int INTEGER;  
③ typedef int INTEGER;  
④ INTEGER i, j;
```

```
① int a[10];  
② int ARRAY[10];  
③ typedef int ARRAY[10];  
④ ARRAY a, b[5];
```

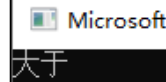
```
① char *s;  
② char *STRING;  
③ typedef char *STRING;  
④ STRING p, x[10];
```

#### 特别说明:

- ★ typedef声明新类型不属于必须使用的方法
- ★ 请大家自行评估可读性并选择适合自己的风格, 不强求

```
#include <iostream>  
#include <cstring>  
using namespace std;  
  
typedef const char* STRING;  
  
int main()  
{  
    const char *p1="house";  
    STRING p2="horse";  
  
    if (strcmp(p1, p2)>0)  
        cout<<"大于"<<endl;  
    else  
        cout<<"不大于"<<endl;  
  
    return 0;  
}
```

正确



```
demo.cpp - x  
demo.cpp (全局范围)  
1  #include <iostream>  
2  using namespace std;  
3  typedef int INTEGER;  
4  int fun(int a)  
5  {  
6      return 0;  
7  }  
8  INTEGER fun(INTEGER a)  
9  {  
10     return 0;  
11 }  
12 int main()  
13 {  
14     return 0;  
15 }
```

demo.cpp(8,9): error C2084: 函数“int fun(int)”已有主体  
demo.cpp(4,5): message : 参见“fun”的前一个定义

★ 使用方法与原来的类型一致, 与原类型可直接混用, 不需要进行强制类型转换