《计算概论A》课程 程序设计部分

链表　枚举类型　共同体

李　戈

北京大学　信息科学技术学院　软件研究所

lige@sei.pku.edu.cn

北京大学

关于 new & delete

- new
  - C++运算符
  - 动态地分配内存空间，并将所分配的内存的地址赋给指针变量
- delete
  - C++运算符
  - 将动态分配的内存空间归还给系统

关于 new & delete

- 用法一：
  - <指针变量> = new <类型> ;
    - 分配某种类型大小的一片连续内存空间，并将内存空间的首地址赋给指针变量。
  - delete <指针变量>;

```
int main()
{
    int *p = new int;
    *p = 10;
    cout<<*p<<endl;
    delete p;
    system("pause");
    return 0;
}
```

```
10
请按任意键继续. . .
```

关于 new & delete

```
int main()
{
    int *p = new int;
    cout<<*p<<endl;
    delete p;
    system("pause");
    return 0;
}
```

```
-842150451
请按任意键继续. . .
```

关于 new & delete

```
int main()
{
    int *p = new int;
    *p = 10;
    cout<<*p<<endl;
    delete p;
    cout<<*p<<endl;
    system("pause");
    return 0;
}
```

```
10
-17891602
请按任意键继续. . .
```

关于 new & delete

- 用法二：
  - <指针变量> = new <类型>(初值);
    - 分配空间，并将初始值存入所分配的空间中

```
int main()
{
    int *p = new int(5);
    cout<<*p<<endl;
    delete p;
    cout<<*p<<endl;
    system("pause");
    return 0;
}
```

```
5
-17891602
请按任意键继续. . .
```

关于 new & delete

- 用法三：
  - <指针变量> = new <类型>[<常量表达式>];
    - 分配指定类型的数组空间，并将数组的首地址赋给指针变量。
  - delete [ ]<指针变量>;
    - 将指针变量所指向一维数组内存空间归还给系统。

关于 new & delete

```
int main()
{
    int *p = new int[5];
    memset(p,0,20);
    for(int i=0;i<5;i++)
        cout<<*(p+i)<<endl;
    delete []p;
    for(int i=0;i<5;i++)
        cout<<*(p+i)<<endl;
    system("pause");
    return 0;
}
```

```
0
0
0
0
0
-17891602
-17891602
-17891602
-17891602
-17891602
请按任意键继续. . .
```

北京大学

顺便说一下 memset

```
int main()
{
    int *p = new int[5];
    memset(p,1,20);
    for(int i=0;i<5;i++)
        cout<<*(p+i)<<endl;
    delete []p;
    for(int i=0;i<5;i++)
        cout<<*(p+i)<<endl;
    system("pause");
    return 0;
}
```

```
16843009
16843009
16843009
16843009
16843009
-17891602
-17891602
-17891602
-17891602
-17891602
请按任意键继续. . .
```

北京大学

顺便说一下 memset

```
int main()
{
    int *p = new int[5];
    memset(p,1,20);
    for(int i=0;i<5;i++)
        cout<<hex<<*(p+i)<<endl;
    delete []p;
    for(int i=0;i<5;i++)
        cout<<*(p+i)<<endl;
    system("pause");
    return 0;
}
```

```
1010101
1010101
1010101
1010101
1010101
feeefeee
feeefeee
feeefeee
feeefeee
feeefeee
请按任意键继续. . .
```

当 new & delete 用于结构体

```
#include<iostream>
using namespace std;
struct Node
{
    int n;
    Node * next;
};
int main()
{
    Node *p = new Node;
    cout<<p->n<<endl;
    cout<<p->next<<endl;
    system("pause");
    return 0;
}
```

```
-842150451
CDCDCDCD
请按任意键继续. . .
```

当 new & delete 用于结构体

```
#include<iostream>
using namespace std;
struct Node
{
    int n;
    Node * next;
};
int main()
{
    Node *p = new Node;
    cout<<p->n<<endl;
    cout<<p->next<<endl;
    delete p;
    cout<<p->n<<endl;
    cout<<p->next<<endl;
    system("pause");
    return 0;
}
```

```
-842150451
CDCDCDCD
-17891602
FEEEFEEE
请按任意键继续. . .
```

当 new & delete 用于结构体

```
#include<iostream>
using namespace std;
struct Node
{
    int num[5];
    Node * next;
};
int main()
{
    Node *p = new Node;
    for(int i=0; i<5; i++)
        cout<<p->num[i]<<endl;
    cout<<p->next<<endl;
    delete p;
    for(int i=0; i<5; i++)
        cout<<p->num[i]<<endl;
    cout<<p->next<<endl;
    system("pause");
    return 0;
}
```

```
-842150451
-842150451
-842150451
-842150451
-842150451
CDCDCDCD
-17891602
-17891602
-17891602
-17891602
-17891602
FEEEFEEE
请按任意键继续. . .
```

字符串处理中常用new

- 问题
  - 将一个位数不定的整数三位分解后输出
  - 如：123456789　　123，456，789
- 要求：
  - 按照输入整数的长短，动态申请存储空间
- 思路：
  - 将整数存入字符数组，并加上逗号，然后输出
    - 需要判断字符数组的长度（整数长度+逗号个数）
    - 按照整数转换成字符的规律，从后向前赋值；

```
int main() {
    int num, ilen, clen, k = 0;
    cout<<"Please input a number ";
    cin >> num;
    ilen = getlength(num);
    clen = ilen + ilen / 3;
    char *p = new char [clen+1];
    p = p + clen;
    *p = '\0';
    while (num != 0) {
        *(--p) = num % 10 + '0';
        num = num / 10;
        k++;
        if (k == 3) {
            *--p = ',';
            k = 0;
    }}
    if (*p == ',')
        cout<<p+1<<endl;
    else
        cout << p << endl;
    return 0;
}
```

```
int getlength(int num) {
    int count =0;
    while (num != 0) {
        num = num / 10;
        count ++;
    }
    return count;
}
```

约瑟夫环问题

- 约瑟夫环（Josephus）问题：
  - 古代某法官要判决n个犯人的死刑，他有一条荒唐的法律，将犯人站成一个圆圈，从第s个人开始数起，每数到第d个犯人，就拉出来处决，然后再数d个，数到的人再处决.....直到剩下的最后一个可赦免.

约瑟夫序列

描述

n 个小孩围坐成一圈，并按顺时针编号为1,2,…,n，从编号为 p 的小孩顺时针依次报数，由1报到m ，当报到 m 时，该小孩从圈中出去，然后下一个再从1报数，当报到 m 时再出去。如此反复，直至所有的小孩都从圈中出去。请按出去的先后顺序输出小孩的编号(假设小孩的个数不多于300个)。

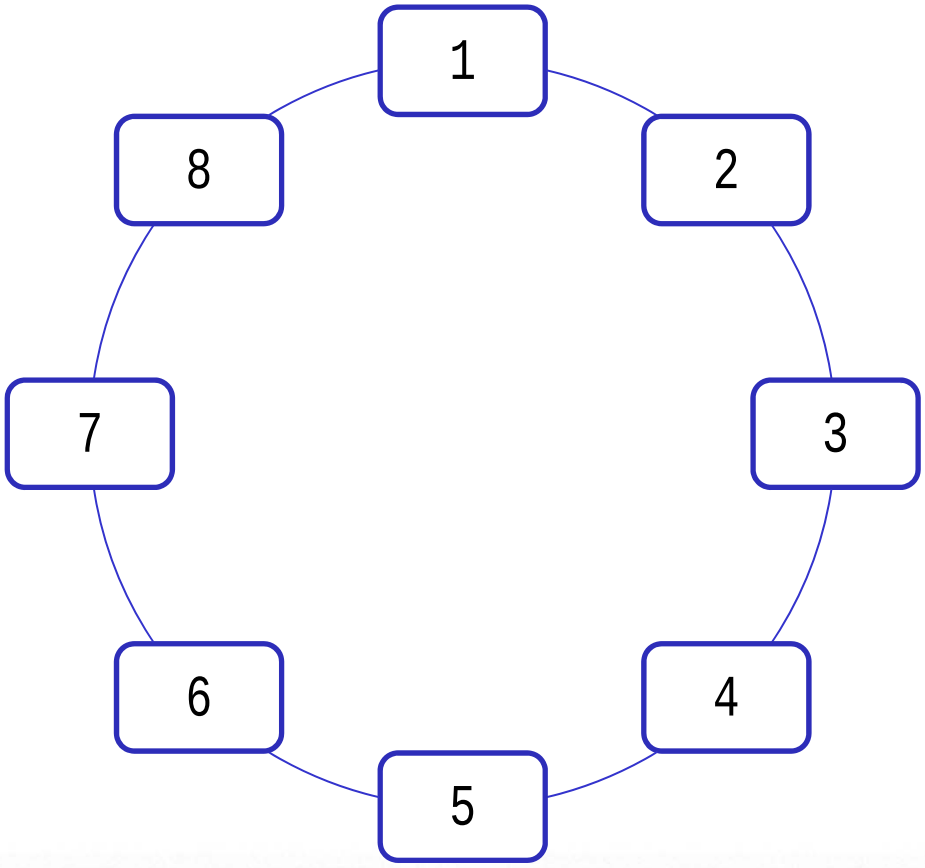
关于输入

n,p,m 的值在1行内输入，以空格间隔。

关于输出

按出圈的顺序输出编号，编号之间以逗号间隔。

约瑟夫序列



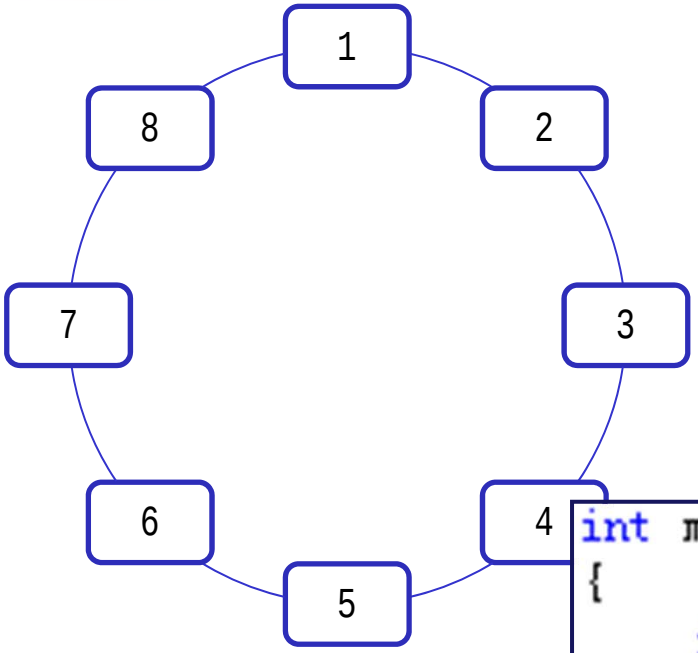
例子输入

```
8 3 4
```

例子输出

```
6, 2, 7, 4, 3, 5, 1, 8
```

约瑟夫序列

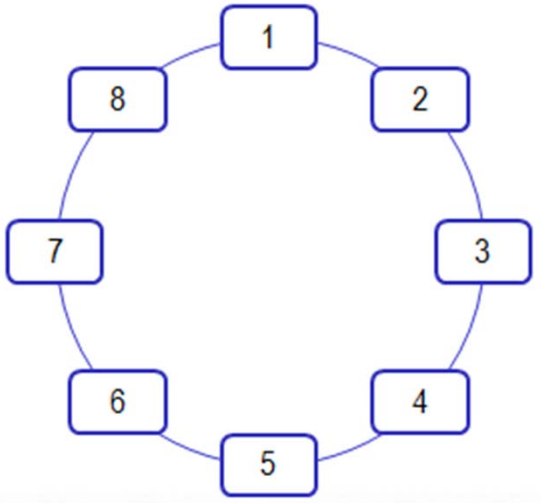



```
struct Node
{
    int num;
    Node * ahead;
    Node * next;
};
```

```
int main()
{
    int N, P, M = 0;  //N-起始节点数，P-开始节点
    cin>>N>>P>>M;     //每次释放第M个节点
    Node * head = Create(N);   //创建N个节点的环
    head = Search(head, P);    //找到第P个节点
    while(head->next != head)  //不断释放第M个元素
    {                          //直到只剩一个元素
        head = Release(head,M); //释放第M个节点
    }
    cout<<head->num;
    return 0;
}
```

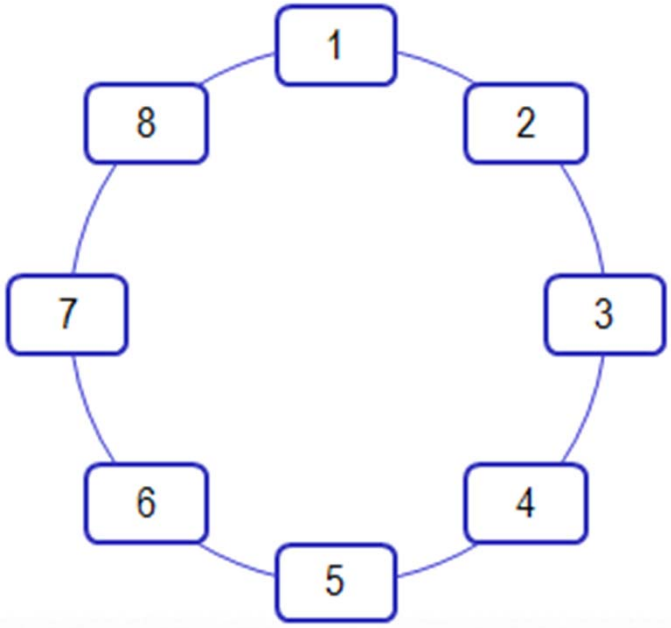
约瑟夫序列

```
Node *Create(int N)   //创建包含N个节点的双向循环链表
{
      int n = 1;
      Node * node = new Node;
      node->num = n;
      Node * head = node;          //指向第一个节点
      Node * tail = head;          //指向最后一个节点
      while( n++ < N)
      {
            node = new Node;       //建新节点
            node->num = n;         //赋值
            tail->next = node;     //接入新节点
            node->ahead = tail;
            tail = tail->next;     //尾巴后移
      }
      tail->next = head;           //尾巴处理
      head->ahead = tail;
      return head;
}
```



约瑟夫序列

```
Node *Search(Node *head, int P) //从Head开始寻找第P个节点
{
      while(head->num != P)
      {
            head = head->next;
      }
      return head;
}
```



注：这里没有错，因为题意要求是从编号为p的小朋友开始。

约瑟夫序列

```
Node *Release(Node *head, int M) //释放Head开始的第M个节点
{
    int count = 1;
    Node *temp = head;
    while(count < M)              //寻找第M个节点
    {
        temp = temp->next;
        count++;
    }
    temp->ahead->next = temp->next;      //移除第M个节点
    temp->next->ahead = temp->ahead;     //移除第M个节点
    cout<<temp->num<<",";
    head = temp->next;      //释放第M个节点所占内存空间
    delete temp;
    return head;
}
```

枚举类型、共同体类型

枚举类型

- 枚举
  - 如果一个变量只有几种可能的取值，则可以将该变量定义为“枚举类型”。
- 定义
  - 声明一个枚举数据类型**weekday**

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

    花括号中**sun,mon,…,sat**等称为枚举元素
  - 定义枚举变量：

    enum weekday workday, weekend;

    或　　**weekday workday, weekend;**
  - 枚举变量赋值：

    workday = sun;　　weekend = mon;

枚举类型使用注意事项

1. 枚举元素按常量处理，不能对它们赋值。

```
sun = mon;    （错误）
```

2. 枚举类型不能直接输出元素的名字.

[例如]
```
enum color {red, green,white,black};
color cloth = red;
cout<<cloth;   //结果为0
```

3. 枚举型可以比较

```
if (cloth > white)  count++;
```

4. 一个整型不能直接赋给一个枚举变量

```
workday = 2;  //错误！
```

枚举类型

- 枚举元素有值
  - 定义时枚举元素如未指定值，编译系统按定义顺序取默认值依次为0，1，2，3...
  - 也可以给枚举值指定对应值

enum day { Sun=7,Mon=1,Tue,Wed,Thu,Fri,Sat };

这时，**Sun=7，Mon=1，Tue=2，Wed=3 ……**

- 整数不能直接赋给枚举变量

如：**workday = 2；**错误！

  - 应先进行强制类型转换如：

workday = (enum weekday) 2;

枚举类型

```
#include<iostream>
using namespace std;
enum color {red,yellow,green=3,blue };
enum color c1;
int main()
{
     c1=blue;
     cout<<"red="<<red<<" yellow="<<yellow<<" greeen="<<green<<endl;
     cout<<"blue="<<blue<<" c1="<<c1<<endl;
     system("pause");
}
```

```
red=0 yellow=1 greeen=3
blue=4 c1=4
请按任意键继续. . .
```

北京大学

枚举类型的使用

- 如何输出枚举型变量的内容

```
enum color {red,green,blue,brown,white,black};
enum color choice;
 switch(choice)
{  case red:      cout<<“red\n”;    break;
   case green:    cout<<“green\n”; break;
   case blue:     cout<<“blue\n”;   break;
   case green:    cout<<“brown\n”;break;
   case red:      cout<<“white\n”; break;
   case green:    cout<<“black\n”; break;
}
```

枚举类型应用举例

- 问题
  - 按计时工资方法计算一周的付费。每周有7个工作日，周一至周五，按实际工作小时计算，周六工作时间按实际工作小时的1.5倍计算，周日工作时间按实际工作小时2.0倍计算。
- 用户输入
  - 每小时正常应付工资金额
  - 周一至周日每天的工作时间
- 程序输出
  - 计算出一周应付的工资。要求周一至周日用枚举类型表示；

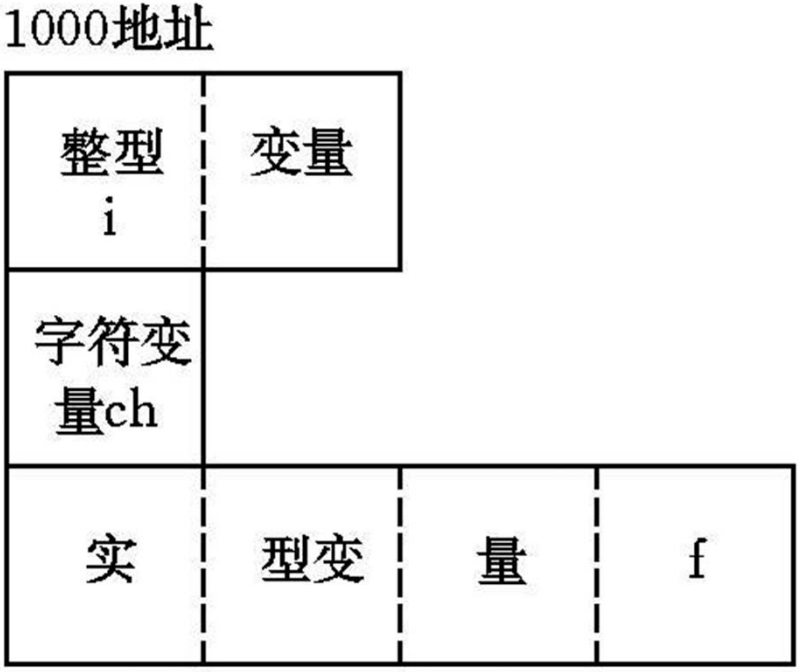
```
#include<iostream>
using namespace std;
int main()
{    enum day{Mon,Tue,Wed,Thu,Fri,Sat,Sun};
     day workDay;
     double times, wages = 0, hourlyPay, hours;
     cout<<"Enter the hourly wages rate."<<endl;
     cin>>hourlyPay;
     cout<<"Enter hours worked daily\n";
     for (int i=0; i<7; i++)
     {
          cin>>hours;
          switch((day)i)
          {
               case Sat:   times=1.5*hours; break;
               case Sun:   times=2.0*hours; break;
               default:    times=hours;
          }
          wages = wages + times*hourlyPay;
     }
     cout<<"The wages for the week are "<<wages;
     return 0;
     system("pause");
}
```

共用体

共用体

- 共用体是什么？
  - 为了节省内存空间，可以将几种不同类型的变量存放到同一段内存单元中，这段内存单元所对应的数据结构称为共用体。

- 例如：
  - 一个整型变量、一个字符型变量、一个实型变量可以放在同一个地址开始的内存单元中。



共用体的定义

- 共用体的定义

```
union 共用体名
{
    成员列表;
} 变量表列;
```

```
union data
{
    int i ;
    char ch ;
    float f ;
} a, b, c;
```

直接定义

```
union data
{
    int i ;
    char ch ;
    float f ;
}
data   a, b, c;
```

分开定义

北京大学

共用体的引用

- 不能引用共用体变量，只能引用共用体变量中的成员。
  - 例如：

```
union data
{
    int i ;
    char ch ;
    float f ;
} a, b, c;
```

a.i (引用共用体变量中的类型变量i)

a.ch (引用共用体变量中的字符变量ch)

a.f (引用共用体变量中的实型变量f)

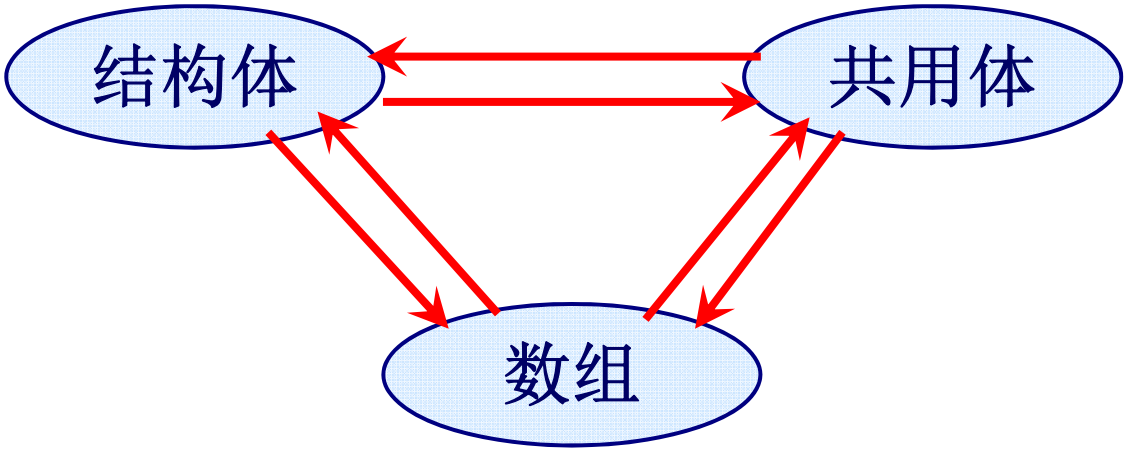
不能只引用共用体变量，例如：

cout<<a; 错误

共用体类型数据的特点

- 同一内存段可以存放几种不同类型的成员，但在同一时刻时只能存放其中一种；
- 共用体变量中起作用的成员是最后一次存放的成员，在存入一个新的成员后原有的成员就失去作用。
- 共用体变量的地址和它的各成员的地址都是同一地址。
  - 例如： **&a, &a.i, &a.c, &a.f** 都是同一地址值

共用体、结构体、数组



- 结构体 中可含 共用体；
- 共用体 中可含 结构体；
- 结构体 和 共用体 中可以包含 数组；
- 可以定义 共用体数组 和 结构体数组；

举 例

例11.12 设有若干个人员的数据，其中有学生和教师。

- 学生数据：姓名、号码、性别、职业、班级。
- 教师数据：姓名、号码、性别、职业、职务。

设计一个数据结构将多个学生和老师的数据用同一个数组存放。

name	num	sex	job	class(班) / position(职务)
Li	1011	f	s	501
wang	2085	m	t	prof

包含 共用体 的 结构体 数组

```
struct
{   int num;
    char name[10];
    char sex;
    char job;
    union
    {
        int class;
        char position[10];
    } category;
} person[2];
```

结构体

共用体

数组

共用体小结

① 共用体内的成员是互斥的；

② 最后一次赋值保留在共用体中；

③ 共用体只有一个首地址；

④ 共用体不能初始化,不能对整个共用体赋值；

⑤ 在函数中，可以使用共用体的指针，但不能使用名字做函数参数；

⑥ 共用体的空间是所有成员中最大的一个；

补充

位运算

- 位运算
  - 所谓位运算是指进行二进制位的运算。
- C++语言中的位运算符
  - 按位与（&）　　双目运算符
  - 按位或（|）　　双目运算符
  - 按位异或（^）　　双目运算符
  - 取反（~）　　单目运算符
  - 左移（<<）　　单目运算符
  - 右移（>>）　　单目运算符

位运算符（1）

- “按位与”运算符(&)
  - 参加运算的两个数据，各位均独立进行“与”运算。
- 例如：对于表达式：a = 3 & 5，有：

```
    3 = 00000011
(&) 5 = 00000101
----------------
    00000001
```

因此，3&5的值为：1

位运算符（2）

- “按位或”运算符（|）
  - 参加运算的两个数据，各位均独立进行“或”运算。
- 例如：对于表达式：a = 3 | 5，有：

```
    3 = 00000011
( | ) 5 = 00000101
-----------------
    00000111
```

因此，3 | 5的值为：7

位运算符（3）

- “异或”运算符（∧）
  - 异或运算符∧也称XOR运算符。参加运算的两个数，各位均独立进行 异或运算：
  - 即0 ∧ 0 = 0；0 ∧ 1 = 1；1 ∧ 0 = 1；1 ∧1 = 0；

例如：

```
        00111001   (十进制数57，八进制数071)
(∧)     00101010   (十进制数42，八进制数052)
--------------------------------------------
        00010011   (十进制数19，八进制数023)
```

即071∧052，结果为023(八进制数)。

位运算符（4）

- 取反运算符“～”是一个单目(元)运算符，用来对一个二进制数按位取反，
  - 即将0变1，1变0。
  - 例如：

000000000010101

(～)　　↓

111111111101010

因此，～(025) $_8$的值为 $(177752)_8$

位运算符（5）

- 左移运算符(<<)
  - 用来将一个数的各二进位全部左移若干位。
  - 高位左移后溢出，舍弃不起作用。
  - 若a=15，即二进制数00001111，左移2位得00111100，即十进制数60

    a = a<<2
  - 左移1位相当于该数乘以2，左移2位相当于该数乘以2^2=4。
    - 只适用于该数左移时被溢出舍弃的高位中不包含1的情况。

位运算符（6）

- 右移运算符(>>)
  - 用来将一个数的各二进位全部右移若干位。
  - 移到右端的低位被舍弃，对无符号数，高位补0。
  - 若a=15，即二进制数00001111，右移2位得00000011，即十进制数3；

$$a = a >> 2$$

  - 当没有非零数位被抛弃时，右移一位相当于除以2，右移n位相当于除以2^n。

关于位运算的几个问题(1)

- 右移运算符号位的处理
  - 对无符号数，右移时左边高位移入0。
  - 对于有符号的值，
    - 若原来符号位为0(该数为正)，则左边移入0；
    - 若符号位原来为1(即负数)，则左边移入0还是1，要取决于所用的计算机系统。
      - 若移入0，称为“逻辑右移”或“简单右移”
      - 若移入1，称为“算术右移”（VC）

关于位运算的几个问题(2)

- 不同长度的数据进行位运算
  - 如果两个数据长度不同进行位运算时，系统会将二者按右端对齐。
    - 如a & b，而a为int型，b为short型
  - 如何补位
    - 如果b为无符号整数型，则左侧添满0。
    - 如果b为有符号整数型：
      - 如果b为正数，则左侧16位补满0。
      - 如果b为负数，则左端16位补满1。

关于位运算的几个问题(3)

- 位运算赋值运算符
  - 位运算符与赋值运算符可以组成复合赋值运算符如: &=，|=， >>=， <<=， ∧=
    - a & = b　相当于　a = a & b
    - a | = b　相当于　a = a | b
    - a >> =2　相当于　a = a >> 2
    - a << =2　相当于　a = a << 2
    - a∧= b　相当于　a = a ∧ b

关于位运算的几个问题(4)

- 算符优先级
  - 取反运算符“～”
  - 算数运算符
  - 左移<< 右移>>
  - 关系运算符
  - 按位与&
  - 按位异或^
  - 按位或 |
  - 逻辑运算符

高

低

位运算的使用（1）

- “按位与”运算的用途
  - 任意存储单元与0进行“按位与”运算可清零；
  - 取一个数中某些指定位
    - 如有一个整数a，想要其中的低字节。只需将a与 $(377)_8$ 按位与即可。

a	00 10 11 00	10 10 11 00
b	00 00 00 00	11 11 11 11
c	00 00 00 00	10 10 11 00

- “按位或”运算
  - 对一个数据的某些位取定值为1；

位运算的使用（2）

- “异或”运算符的使用

（1）使特定位翻转

- 使01111010低4位翻转（即1变为0，0变为1）可将其与00001111进行∧运算，即：

```
    01111010
(∧) 00001111
    --------
    01110101
```

（2）使特定位保持不变

- 与0相∧，保留原值如012∧00=012

```
    00001010
(∧) 00000000
    --------
    00001010
```

北京大学

“异或”运算符示例

- 交换两个值，而不必使用临时变量
  - 假如a = 3，b = 4。想将a和b的值互换，可以用以下赋值语句实现：

```
a = a ∧ b;
b = b ∧ a;
a = a ∧ b;
```

设：a = 3, b = 4

```
      a=011
(∧)   b=100
      ---------
      a=111 (a∧b的结果，a已变成7)
(∧)   b=100
      ---------
      b=011 (b∧a的结果，b已变成3)
(∧)   a=111
      ---------
      a=100 (a∧b的结果，a变成4)
```

输出一个数的二进制编码

```
#include <iostream>
using namespace std;
void main()
{
    int i;
    int n = -2147483648;
    unsigned int b = 0x80000000;
    for (i = 0; i < 32; i ++) {
        cout<<n & b ? 1 : 0;
        b >>= 1;
    }
}
```

条件运算符

表达式1? 表达式2： 表达式3

求值规则：如果表达式1的值为真，则以表达式2 的值作为条件表达式的值；否则以表达式2的值作为整个条件表达式的值。

例如：

```
max=(a>b)?a:b;
```

相当于：

```
if(a>b)  max=a;
else max=b;
```

好好想想，有没有问题?

谢　谢　！

北京大学