ビット処理

０．目次

１．ビット演算

つぎのビット演算を使って、ビット単位の処理ができる。

演算	機能	例
&	ビット単位の論理積 0&0 = 0 0&1 = 0 1&0 = 0 1&1 = 1	01010011 & 00001111 00000011
|	ビット単位の論理和 0|0 = 0 0|1 = 1 1|0 = 1 1|1 = 1	01010011 | 00001111 01011111
^	ビット単位の排他的論理和 0^0 = 0 0^1 = 1 1^0 = 1 1^1 = 0	01010011 ^ 00001111 01011100
<<	左シフト 最右端から0が詰められる。	a << 2 aを2ビット左シフトする。 00001111 --> 00111100
>>	右シフト 最左端のビットが詰められる。	a >> 2 aを2ビット右にシフトする。 00001111 --> 00000011 11110000 --> 11111100

１．１　論理積　論理和　排他的論理和

●プログラム(b111.c)

```
/* << b111.c >> */
#include <stdio.h>
main() {
  char x,y,z; /* 1文字は8ビットで表される。
                 8進定数は先頭に0を付ける。
                 16進定数は先頭に0x(または0X)を付ける。*/
  x = 0x53;
  y = 0x0f;
  /* 論理積 */
  z = x & y; printf("論理積 : %02x & %02x = %02x¥n",x,y,z);
  /* 論理和 */
  z = x | y; printf("論理和 : %02x | %02x = %02x¥n",x,y,z);
  /* 排他的論理和* /
  z = x ^ y; printf("排他的論理和 : %02x ^ %02x = %02x¥n",x,y,z);
}
```

実行結果

```
% cc b111.c
% a.out
論理積 : 53 & 0f = 03
論理和 : 53 | 0f = 5f
排他的論理和 : 53 ^ 0f = 5c
```

１．２　左シフト 右シフト

char型変数の左シフト、右シフトを示す。

●プログラム(b112.c)

```
/* << b112.c >> */
#include <stdio.h>
main() {
  int i;
  char x,y;
  x = 0x0f;
  /* 左シフト */
  for( i=1; i<=5; i++ ) {
    y = x << i;
    printf("左シフト：%02x << %d =  %02x¥n",x,i,y);
  }
  printf("¥n");
  /* 右シフト */
  for( i=1; i<=5; i++ ) {
    y = x >> i;
    printf("右シフト：%02x >> %d =  %02x¥n",x,i,y);
  }
  printf("¥n");

  x = 0xf0;
  /* 左シフト */
  for( i=1; i<=5; i++ ) {
    y = x << i;
    printf("左シフト：%02x << %d =  %02x¥n",x,i,y);
  }
  printf("¥n");
  /* 右シフト */
  for( i=1; i<=5; i++ ) {
    y = x >> i;
    printf("右シフト：%02x >> %d =  %02x¥n",x,i,y);
  }
}
```

実行結果

```
% cc b112.c
% a.out
左シフト：0f << 1 =  1e
左シフト：0f << 2 =  3c
左シフト：0f << 3 =  78
左シフト：0f << 4 =  fffffff0
左シフト：0f << 5 =  ffffffe0

右シフト：0f >> 1 =  07
右シフト：0f >> 2 =  03
右シフト：0f >> 3 =  01
右シフト：0f >> 4 =  00
右シフト：0f >> 5 =  00

左シフト：fffffff0 << 1 =  ffffffe0
左シフト：fffffff0 << 2 =  ffffffc0
左シフト：fffffff0 << 3 =  ffffff80
左シフト：fffffff0 << 4 =  00
左シフト：fffffff0 << 5 =  00

右シフト：fffffff0 >> 1 =  fffffff8
右シフト：fffffff0 >> 2 =  fffffffc
右シフト：fffffff0 >> 3 =  fffffffe
右シフト：fffffff0 >> 4 =  ffffffff
右シフト：fffffff0 >> 5 =  ffffffff
```

int型変数の左シフト、右シフトを示す。

●プログラム(b113.c)

```
/* << b113.c >> */
#include <stdio.h>
main() {
  int x,y;
  x = 0x0000000f;
  /* 左シフト */
  y = x << 1; printf("左シフト：%08x << 1 =  %08x¥n",x,y);
  y = x << 1; printf("左シフト：%d << 1 =  %d¥n¥n",x,y);
  /* 右シフト */
  y = x >> 1; printf("右シフト：%08x >> 1 =  %08x¥n",x,y);
  y = x >> 1; printf("右シフト：%d >> 1 =  %d¥n¥n",x,y);

  x = 0xffffffff;
  /* 左シフト */
  y = x << 1; printf("左シフト：%08x << 1 =  %08x¥n",x,y);
  y = x << 1; printf("左シフト：%d << 1 =  %d¥n¥n",x,y);
  /* 右シフト */
  y = x >> 1; printf("右シフト：%08x >> 1 =  %08x¥n",x,y);
  y = x >> 1; printf("右シフト：%d >> 1 =  %d¥n",x,y);
}
```

実行結果

```
% cc b113.c
% a.out
左シフト：0000000f << 1 =  0000001e
左シフト：15 << 1 =  30

右シフト：0000000f >> 1 =  00000007
右シフト：15 >> 1 =  7

左シフト：ffffffff << 1 =  fffffffe
左シフト：-1 << 1 =  -2

右シフト：ffffffff >> 1 =  ffffffff
右シフト：-1 >> 1 =  -1
```

２．ビット列操作

２．１　char型変数の表示

●プログラム(b211.c)

```
1  /* << b211.c >> */
2  #include <stdio.h>
3  main() {
4    char a;     /* 文字。*/
5    int b[32], /* ビットを保存する配列。*/
6        i;
7    void bitshow(char a, int b[]);
8    while( 1 ) {
9      scanf("%c",&a); /* ^C で終了。*/
10     printf("文字       : %c¥n",a);
11     printf("文字コード : %d¥n",a);
12     bitshow(a,b);
13     for( i=7; i>=0; i-- ) { printf("%d",b[i]); }
14     printf("¥n");
15   }
16 }
17 void bitshow(char a, int b[]) {
18   int i,t;
19   t = 0x01;
20   for( i=0; i<8; i++ ) {
21     b[i] = 0;
22     if( (int)(a&t) == t ) { b[i] = 1; }
23     t = t << 1;
24   }
25 }
```

実行結果

```
% cc b211.c
% a.out
1a
文字       : 1
文字コード : 49
00110001
文字       : a
文字コード : 97
01100001
文字       :

文字コード : 10
00001010
```

２．２　int型変数の表示

●プログラム(b221.c)

```
/* << b221.c >> */
#include <stdio.h>
main() {
  int a,      /* 整数。*/
      i;
  void bitshow(int a);
  while( 1 ) {
    scanf("%d",&a); if( a == 0 ) { break; }
    bitshow(a);
  }
}
void bitshow(int a) {
  int i,t,
      b[32]; /* ビットを保存する配列。*/
  t = 0x00000001;
  for( i=0; i<32; i++ ) {
    b[i] = 0; if( (int)(a&t) == t ) { b[i] = 1; }
    t = t << 1;
  }
  for( i=31; i>=0; i-- ) {
    printf("%d",b[i]); if( i%8 == 0 ) { printf(" "); }
  }
  printf("¥n");
}
```

実行結果

```
% cc b221.c
% a.out
3
00000000 00000000 00000000 00000011
2
00000000 00000000 00000000 00000010
1
00000000 00000000 00000000 00000001
-1
11111111 11111111 11111111 11111111
-2
11111111 11111111 11111111 11111110
0
```

２．３　int型変数のビット数

int型変数のビット1の個数を求める。

●プログラム(b231.c)

```
 1  /* << b231.c >> */
 2  #include <stdio.h>
 3  main() {
 4    int a; /* 整数。*/
 5    int bitnum(int a);
 6    while( 1 ) {
 7      scanf("%d",&a); if( a == 0 ) { break; }
 8      bitnum(a);
 9      printf("整数 %d に含まれる1の個数=%d¥n",a,bitnum(a));
10    }
11  }
12  int bitnum(int a) {
13    int i,sum,t;
14    t = 0x00000001;
15    sum = 0;
16    for( i=0; i<32; i++ ) {
17      if( (int)(a&t) == t ) { sum++; }
18      t = t << 1;
19    }
20    return sum;
21  }
```

実行結果

```
% cc b231.c
% a.out
1024
整数 1024 に含まれる1の個数=1
10
整数 10 に含まれる1の個数=2
-1
整数 -1 に含まれる1の個数=32
0
```

２．４　ビット単位の設定

int型整数において、下位からi番目のビットをk（0 or 1）にセットする。

●プログラム(b241.c)

```
/* << b241.c >> */
#include <stdio.h>
main() {
  int a,c,    /* 整数。*/
      i,k;
  int bitset(int a, int i, int k);
  void bitshow(int a);
  while( 1 ) {
    scanf("%d%d%d",&a,&i,&k); if( a == 0 ) { break; }
    c = bitset(a,i,k);
    bitshow(c);
  }
}
int bitset(int a, int i, int k) {
  int j,t;
  if ( k == 1 ) {
    t = 0x00000001;
    t = t << (i-1); /* i-1ビット左にシフト。*/
    a = a | t; /* 1を埋め込む。*/
  } else {
    t = 0xfffffffe;
    /* i-1ビット左にシフト。*/
    for( j=1; j<=i-1; j++ ) {
      t = t << 1; t = t | 0x00000001;
    }
    a = a & t; /* 0を埋め込む。*/
  }
  return a;
}
void bitshow(int a) {
  int i,t,
      b[32]; /* ビットを保存する配列。*/
  t = 0x00000001;
  for( i=0; i<32; i++ ) {
    b[i] = 0; if( (int)(a&t) == t ) { b[i] = 1; }
    t = t << 1;
  }
  for( i=31; i>=0; i-- ) {
    printf("%d",b[i]); if( i%8 == 0 ) { printf(" "); }
  }
  printf("¥n");
}
```

実行結果

```
% cc b241.c
% a.out
15 1 0
00000000 00000000 00000000 00001110
1 8 1
00000000 00000000 00000000 10000001
0 0 0
```

３．課題

３．１　文字の詰め込みと取り出し

４文字を読み込み、１つのint型整数に詰め込む。

●プログラム(b311.c)

```
1  /* << b311.c >> */
2  #include <stdio.h>
3  main() {
4    char ch[4],w; /* 文字は8ビット。*/
5    int z,i;      /* int型整数は32ビット。*/
6    /* 文字の埋め込み。*/
7    printf("文字の埋め込み¥n");
8    z = 0x00000000; /* 詰め込む変数 */
9    ch[0] = 'a';
10   ch[1] = 'b';
11   ch[2] = 'c';
12   ch[3] = 'd';
13   for( i=0; i<=3; i++ ) {
14     z =          /* 8ビット左にシフト。*/
15     z =          /* 変数zの右端8ビットに文字ch[i]を埋め込む。*/
16     printf("z = %8x¥n",z); /* 16進数で表示。*/
17   }
18   /* 文字の取り出し。*/
19   printf("文字の取り出し¥n");
20   for( i=1; i<=4; i++ ) {
21     w =          /* 変数zの右端8ビットを取り出す。*/
22     printf("%c¥n",w);
23     z =          /* 8ビットを右にシフト。*/
24   }
25 }
```

実行結果

```
% cc b311.c
% a.out
文字の埋め込み
z =       61
z =     6162
z =   616263
z = 61626364
文字の取り出し
d
c
b
a
```

３．２　ビット反転

int型変数のビットを反転(0を1、1を0)する。

●プログラム(b321.c)

```
1  /* << b321.c >> */
2  #include <stdio.h>
3  main() {
4    int a,b; /* 正整数。*/
5    int bitreverse(int a);
6    void bitshow(int a);
7    while( 1 ) {
8      scanf("%d",&a); if( a == 0 ) { break; }
9      printf("a=%d¥n",a);
10     printf("反転前 : "); bitshow(a);
11     b = bitreverse(a);
12     printf("b=%d¥n",b);
13     printf("反転後 : "); bitshow(b);
14   }
15 }
16 int bitreverse(int a) {
17   int b,i,t;
18   b = 0x00000000;
19   t = 0x00000001;
20   for( i=0; i<32; i++ ) {
21     if(                    ) { b = b|t; }
22     t = t << 1;
23   }
24   return b;
25 }
26 void bitshow(int a) {
27   int i,t,
28       b[32]; /* ビットを保存する配列。*/
29   t = 0x00000001;
30   for( i=0; i<32; i++ ) {
31     b[i] = 0;
32     if(                    ) { b[i] = 1; }
33     t = t << 1;
34   }
35   for( i=31; i>=0; i-- ) {
36     printf("%d",b[i]);
37     if(                ) { printf(" "); }
38   }
39   printf("¥n");
40 }
```

実行結果

```
% cc b321.c
% a.out
2
a=2
反転前：00000000 00000000 00000000 00000010
b=-3
反転後：11111111 11111111 11111111 11111101
1
a=1
反転前：00000000 00000000 00000000 00000001
b=-2
反転後：11111111 11111111 11111111 11111110
-1
a=-1
反転前：11111111 11111111 11111111 11111111
b=0
反転後：00000000 00000000 00000000 00000000
-2
a=-2
反転前：11111111 11111111 11111111 11111110
b=1
反転後：00000000 00000000 00000000 00000001
0
```

３．３　巡回シフト

int型変数を右にシフトする。移動したビットは、最右端から挿入する。
たとえば、00000011 を意義に1ビットシフトすると、10000001となる。

●プログラム(b331.c)

```
 1  /* << b331.c >> */
 2  #include <stdio.h>
 3  main() {
 4    int a,b, /* 正整数。*/
 5        k;
 6    int bitshift(int a, int k);
 7    void bitshow(int a);
 8    while( 1 ) {
 9      scanf("%d%d",&a,&k); if( a == 0 ) { break; }
10      printf("巡回シフト前："); bitshow(a);
11      b = bitshift(a,k);
12      printf("巡回シフト後："); bitshow(b);
13    }
14  }
15  int bitshift(int a, int k) {
16    int b,i,s0,s1,t;
17    t = 0x00000001; s0 = 0x80000000; s1 = 0x7fffffff;
18    b = a;
19    for( i=1; i<=k; i++ ) {
20      if(                    ) {
21        b = b >> 1;
22      } else {
23        b = b >> 1;
24      }
25    }
26    return b;
27  }
28  void bitshow(int a) {
29    int i,t,
30        b[32]; /* ビットを保存する配列。*/
31    t = 0x00000001;
32    for( i=0; i<32; i++ ) {
33      b[i] = 0;
34      if(                    ) { b[i] = 1; }
35      t = t << 1;
36    }
37    for( i=31; i>=0; i-- ) {
38      printf("%d",b[i]);
39      if(           ) { printf(" "); }
40    }
41    printf("¥n");
42  }
```

実行結果

```
% cc b331.c
% a.out
2 1
巡回シフト前：00000000 00000000 00000000 00000010
巡回シフト後：00000000 00000000 00000000 00000001
2 2
巡回シフト前：00000000 00000000 00000000 00000010
巡回シフト後：10000000 00000000 00000000 00000000
2 3
巡回シフト前：00000000 00000000 00000000 00000010
巡回シフト後：01000000 00000000 00000000 00000000
-2 1
巡回シフト前：11111111 11111111 11111111 11111110
巡回シフト後：01111111 11111111 11111111 11111111
-2 2
巡回シフト前：11111111 11111111 11111111 11111110
巡回シフト後：10111111 11111111 11111111 11111111
-2 3
巡回シフト前：11111111 11111111 11111111 11111110
巡回シフト後：11011111 11111111 11111111 11111111
0 0
```