chapter 7. 비트단위 연산자와 열거형

연습문제

7-1. 정수가 16비트의 2의 보수 표현을 사용한다고 가정하자. -1,-5,-101,-1023의 이진표현을 기술하여라. 음수에 대한 2의 보수 표현은 그 음수에 대응되는 양수의 비트표현을 구하고, 보수를 취한후, 1을 더한후 구할 수 있다.

Sol)

-1	1111 1111 1111 1111	0000 0000 0000 0001 (1)
-5	1111 1111 1111 1011	0000 0000 0000 0101 (5)
-101	1111 1111 1001 1011	0000 0000 0110 0101 (101)
-1023	1111 1100 0000 0001	0000 0011 1111 1111 (1023)

7-2. Alice, Betty, Carole이 16가지 투표를 한다고 가정하자. 그리고 각 개인의 16개의 투표는 16비트 정수에 비트별로 저장된다고 가정하자. 다음과 같이 시작되는 함수를 작성하여라.

```
Short majority(short a, short b, short c)
{
   ……
```

이 함수는 a,b,c에 저장된 Alice, Betty, Carole의 투표를 입력으로 한다. 그리고, 이 함수는 그 투표 결과를 16비트에 저장하여 리턴해야 한다. 즉, a,b,c를 비트 단위로 비교하여, 1의 값이 더 많으면 그 결과 비트를 1로 하고, 0이 더 많으면 결과 비트를 0으로 한다.

Sol)

C code

```
#include <stdio.h>
short majority (short,short,short);
int main(void)
```

```
{
short d,e,f;
scanf("%u",&d);
scanf("%u",&e);
scanf("%u",&f);

printf("MAJORITY = %u ₩n",majority(d,e,f));
return 0;
}

short majority(short a, short b, short c)
{
 short k;
 k = a+ b+ c;
 if (k >= 2) return 1;
 else return 0;
}
```

7-3. 다음과 같이 시작되는 함수를 작성하여라.

int circular_shift(int a, int n)

{ ….

이 함수는 a를 n만큼 왼쪽으로 이동하는 함수로, 이동할 때 상위 비트가 다시 하위 비트로 들어오도록해야 한다. 다음은 char형에 대해 정의된 순환 이동연산의 두가지 예이다. 1000001을 1만큼 순환이동하면 00000011이다.

Sol)

C code

```
#include <stdio.h>

int main(void)
{
int b,c;
scanf("%x",&b);
scanf("%x",&c);
```

```
printf("0x%x₩n",circular_shift(b,c));



return 0;
}
int circular_shift(int a,int n)
{
int c;
while ( n > 0)
{
   c = (a << 1) && 0x80000000;
   if ( c == 0x80000000)   { a = (a << 1) +  1; }
   else { a = a << 1; }
   --n;
}
return a;
}
```

Result (결과)

```
[romance@161s ch7_code]$ cc 7_3.c
[romance@161s ch7_code]$ ./a.out
0x8fffffff
3
0x7ffffff8
[romance@161s ch7_code]$
```

7-4. 각자의 컴퓨터에서 부호 비트가 삽입되는지를 실험해 보아라. 다음 코드는 이것을 알 수 있도록 도와 준다. 이 코드가 왜 그것을 수행하는지 설명하여라.

```
int   i = -1;
unsigned u = -1;
if (i >> 1 == u >> 1)
 printf("Zeros are Shifted in. ₩n");
else printf("Sign bits are shifted in. ₩n");
```

Sol)

i는 –1이므로 0xffff ffff 이다. 여기서 right shift를 행하면 비트가 하나씩 밀리게 된다. 이것과 싸인비트가 없는 –1이므로 역시 0xffff ffff 이다. 따라서 이것 역시 right shift를 행하게 되면 싸인 비트 다음 자리에 1 이 들어 올지 0이 들어올지 를 알 수 있게 된다.

7-5. int의 비트 표현을 다음과 같이 역전시키는 함수를 작성하여라. 다음은 char에 대한 예제이다.
01110101을 역전시키면 10101110이 된다.

Sol)

C code

```
#include <stdio.h>
int main(void)
{
int a;
scanf("%x",&a);
printf("Original %D ₩n",a);
printf("New = %x ₩n",n(a));
return 0;
}
int n(int a)
{ return ~a; }
```

Result (결과)

```
[romance@161s ch7_code]$ ./a.out
0
Original = 0
New = ffffffff
[romance@161s ch7_code]$
```

7-6. 32 비트 수식의 비트들로부터 1 비트씩 건너뛰면서 비트를 추출하는 함수를 작성하여라. 결과는 16비트 수식으로 리턴되어야 한다. 또한 작성한 함수는 2 바이트나 4바이트 워드 컴퓨터에서 모두 수행될 수 있어야 한다.

Sol)

C code

```
#include <stdio.h>

int main(void)
{
int a,i;
scanf("%x",&a);
for (i = 0; i <32; + + i){
printf("%d ",extra(a));
a = a << 1;                }
printf("₩n");
return 0;
}
int extra(int a)
{
 return !!(a & 0x80000000) ;
}
```

Result (결과)

```
[romance@161s ch7_code]$ ./a.out
0xf8ff8fff
1 1 1 1 1 0 0 0 1 1 1 1 1 1 1 1 1 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1
[romance@161s ch7_code]$
```

7-7. 10진 정수의 열을 입력으로 하는 함수를 작성하여라. 그 열의 각 문자는 하나의 10진 숫자로 볼 수 있다. 각 숫자는 4비트의 이진수 열로 변환되어야 하고, 그것은 int형에 패킹되어야 한다. 만일 int형이 32비트이면, 8개의 숫자가 패킹될 수있다. 이 함수의 수행결과를 다음과 같이 나오도록 하여라.

Input a string of decimal digits : 12345678
12345678 = 0001 0010 0011 0100 0101 0110 0111 1000

Sol)

C code

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

int main(void)
{
char a[20];
int i=0,j=0,k=0;
printf("Input a string of decimal digits : ");
scanf("%s",&a);
printf(" %s : ",a);
while (i<strlen(a))
{
k = (int)a[i];
printf("%d%d%d%d",k&0x8 ? 1 : 0,k&0x4 ? 1: 0,k&0x2 ? 1 :0 ,k&0x1? 1: 0);
+ + i;
printf(" ");
}
printf("₩n");
return 0;
}
```

Result (결과)

```
[romance@161s ch7_code]$ ./a.out
Input a string of decimal digits : 123
 123 : 0001 0010 0011
[romance@161s ch7_code]$ ./a.out
Input a string of decimal digits : 1234
 1234 : 0001 0010 0011 0100
[romance@161s ch7_code]$
```

7-8. bit_print() 함수를 사용하여 n = 0,1,2,3,..... ,32에 대해서 2^n과 2^{n+1} 의 이진수 표현을 표로 출력하는 프로그램을 작성하여라.

Sol)

C code

```
#include <stdio.h>
#include <limits.h>
void bit_print(int);
int main(void)
{
int i;
for (i=0; i < 33 ; + + i)
{ printf("%3d : ",i);
   bit_print(i);
   printf("    ");
   bit_print(i-1);
   printf("₩n");
}

return 0;
}
```

```
void bit_print(int a)
{
 int i;
 int n = sizeof(int)* CHAR_BIT;
 int mask = 1 << (n-1);
 for (i=1; i<=n; + + i)
 { putchar(((a & mask) == 0)?'0' : '1');
   a <<= 1;
   if (i % CHAR_BIT == 0 && i < n)
 putchar (' ');
 }
}
```

Result (결과)

```
0 : 00000000 00000000 00000000 00000000   11111111 11111111 11111111 11111111
1 : 00000000 00000000 00000000 00000001   00000000 00000000 00000000 00000000
2 : 00000000 00000000 00000000 00000010   00000000 00000000 00000000 00000001
3 : 00000000 00000000 00000000 00000011   00000000 00000000 00000000 00000010
 .....
30 : 00000000 00000000 00000000 00011110   00000000 00000000 00000000 00011101
31 : 00000000 00000000 00000000 00011111   00000000 00000000 00000000 00011110
32 : 00000000 00000000 00000000 00100000   00000000 00000000 00000000 00011111
```

7-9. 이 장의 표들에 있는 이진 표현 중 어떤 것은 오류를 검사하기 용이하나 일부는 그렇지 못하다. Bit_print() 함수를 이용하여 좀 더 어려운 표현을 검사해 보아라.

Sol) 생략

7-10. 2바이트 워드나 4바이트워드 컴퓨터에서 모두 동작할 수 있도록 bit_print()함수를 수정하여라.

Sol)

```
void bit_print(int a)
{
 int i;
 int n = sizeof(int)* CHAR_BIT;
 int mask = 1 << (n-1);
 for (i=1; i<=n; ++i)
 { putchar(((a & mask) == 0)?'0' : '1');
  a <<= 1;
  if (i % CHAR_BIT == 0 && i < n)
 putchar (' ');
 }
}
```

7-11. 상수 0xff, 0xff00, 0xff0000, 0xff000000을 마스크로 사용하는데 익숙하지않다면, bit_print() 함수를 사용하여 이들의 값을 비트열로 출력하는 프로그램을 작성해 보아라.

Sol)

C code

```
#include <stdio.h>
#include <limits.h>
void bit_print(int);
int main(void)
{
int i,a=0xff;
for (i=0; i<4; ++i)
{ printf("%10d : ",a);
  bit_print(a);
```

```
   printf("₩n");
   a <<= 8;
}

return 0;
}

void bit_print(int a)
{
 int i;
 int n = sizeof(int)* CHAR_BIT;
 int mask = 1 << (n-1);
 for (i=1; i<=n; ++i)
 { putchar(((a & mask) == 0)?'0' : '1');
   a <<= 1;
   if (i % CHAR_BIT == 0 && i < n)
 putchar (' ');
 }
}
```

Result (결과)

```
[romance@161s ch7_code]$ ./a.out
         255 : 00000000 00000000 00000000 11111111
       65280 : 00000000 00000000 11111111 00000000
    16711680 : 00000000 11111111 00000000 00000000
 -16777216 : 11111111 00000000 00000000 00000000
[romance@161s ch7_code]$
```

7-12. 만일 4바이트 워드 컴퓨터를 사용하고 있다면, bit_print() 함수를 사용하여 다중 바이트 문자 ‘abc’가 어떻게 저장되는지 알아보는 프로그램을 작성하여라. 2바이트 워드 컴퓨터에서는 단지 두 문자만이 하나의 다중 바이트 문자로 저장될 수 있다. 그 경우에는 ‘ab’ 를 사용해 보아라.

Sol)

C code

```
#include <stdio.h>
#include <limits.h>
#include <string.h>
void bit_print(int);
int main(void)
{
int i;
 bit_print('abc');
printf("₩n");
return 0;
}

void bit_print(int a)
{
 int i;
 int n = sizeof(int)* CHAR_BIT;
 int mask = 1 << (n-1);
 for (i=1; i<=n; ++i)
 { putchar(((a & mask) == 0)?'0' : '1');
  a <<= 1;
  if (i % CHAR_BIT == 0 && i < n)
 putchar (' ');
 }
}
```

Result (결과)

```
[romance@161s ch7_code]$ ./a.out
00000000 01100001 01100010 01100011
[romance@161s ch7_code]$
```

7-13. 룰렛 프로그램을 작성하여라. 룰넷은 무작위로 0에서 35사이의 수 하나를 선택 한다. 경기하는 사람은 짝수인지 홀수인지 맞추는 홀수/짝수 게임에 돈을 걸거나, 특정숫자를 맞추는 게임에 돌을 걸 수 있다. 룰넷의 결과가 0이면 짝수/홀수 게임에 건 돈은 모두 잃게 되고, 이 경우를 제외하고는 홀수/짝수 게임에 건 돈의 2배로 받는다. 경기하는 사람이 특정숫자에 돈을 걸었고 룰넷의 결과가 그 숫자라면, 건돈의 35배의 돈을 받는다. 이 룰넷 게임에서 한 게임당 1달러를 건다면, 10달러를 잃을 때까지 몇 게임이나 할 수 있겠는가?

Sol)
홀수/짝수 게임을 한 게임해서 얻을 수 있는 기대값은 (17/36)*2 달러이다.
따라서 홀수/짝수 게임만 해서는 –(1/18)*180 = -10 이므로 180게임을 예상할 수 있다. 다음은 특정숫자를 맞추는 게임의 경우는 기대값이 35*(1/36)이므로 이 게임을 반복 했을 경우 –(1/36)*360 = -10 이므로 360 게임이 가능하다. 그러므로 양쪽 게임을 같은 번 했을 경우 홀수/짝수 게임은 90게임 특정 숫자를 맞추는 게임은 180 게임이 가능하다. 따라서 270게임이 이론상으로 가능하다.

7-14. 이전 달을 리턴하는 previous_month() 함수를 작성하여라. 이 함수는 다음과 같이 시작해야 한다.

- enum month { jan,feb, …. , dec}
- typedef enum month month;

jan이 함수의 인자로 전달되면, dec이 리턴되어야 한다. 또한 달의 이름을 출력하는 함수도 작성하여라. 좀더 명확하게 말하자면, 열거자 jan이 인자로서 전달되면, january가 출력되어야한다.

Sol)

C code

```
#include <stdio.h>
#include <string.h>
enum month {jan,feb,mar,apr,may,jun,jul,aug,sep,oct,nov,dec};
typedef  enum month  month;
month previus_month(month);
```

```
char *this_month(month);
char this_month2[20];
int main(void)
{
month a;
month i;
printf("Input the month :");
scanf("%d",&a);
printf("now %d %s₩n",a,this_month(a));
i = previus_month(a);
printf("previus_month is %d %s₩n",i,this_month(i));
return 0;
}
month previus_month(month a)
{
 month previus_month;
 switch(a) {
 case jan: previus_month = dec; break;
 case feb: previus_month = jan; break;
 case mar: previus_month = feb; break;
 case apr: previus_month = mar; break;
 case may: previus_month = apr; break;
 case jun: previus_month = may; break;
 case jul: previus_month = jun; break;
 case aug: previus_month = jul; break;
 case sep: previus_month = aug; break;
 case oct: previus_month = sep; break;
 case nov: previus_month = oct; break;
 case dec: previus_month = nov;
                }
 return previus_month;
}
char *this_month(month a)
{
 switch(a) {
 case jan: strcpy(this_month2,"jan"); break;
```

```
 case feb: strcpy(this_month2,"feb"); break;
 case mar: strcpy(this_month2,"mar"); break;
 case apr: strcpy(this_month2,"apr"); break;
 case may: strcpy(this_month2,"may"); break;
 case jun: strcpy(this_month2,"jun"); break;
 case jul: strcpy(this_month2,"jul"); break;
 case aug: strcpy(this_month2,"aug"); break;
 case sep: strcpy(this_month2,"sep"); break;
 case oct: strcpy(this_month2,"oct"); break;
 case nov: strcpy(this_month2,"nov"); break;
 case dec: strcpy(this_month2,"dec");
           }
```

Result (결과)

```
[romance@161s ch7_code]$ ./a.out
Input the month :5
now 5 jun
previus_month is 4 may
[romance@161s ch7_code]$
```

7-15. 특정한 년의 다음날을 출력하는 프로그램을 작성하여라. 프로그램은 두 정수, 예를를 들어 17 May를 나타내는 17과 5의 입력을 받아들여, 다음날인 18 May를 출력해야 한다. 프로그램에서 열거형을 사용하고, 다음날이 다른 달로 넘어갈 때 주의하여라.

Sol)

C code

```
enum month {jan,feb,mar,apr,may,jun,jul,aug,sep,oct,nov,dec};
typedef enum month month;
#include <stdio.h>
int next_day(month,int);

int main(void)
```

```
{
month a;
int b = 0;
printf("Input the Month ");
scanf("%d",&a);
printf("Input the Day ");
scanf("%d",&b);
b = next_day(a,b);
if (b == 1) { a = a +  1; }
printf("Next Day is Month %d Day %d ₩n",a,b);

return 0;
}

int next_day(month a,int b)
{
   switch(a)
   {
    case jan : if (b == 31) { b = 1; }
               else { b + = 1; }
               break;
    case feb : if (b == 28) { b = 1; }
               else { b + = 1; }
               break;

    case mar : if (b == 31) { b = 1; }
               else { b + = 1; }
               break;

    case apr : if (b == 30) { b = 1; }
               else { b + = 1; }
               break;

    case may : if (b == 31) { b = 1; }
               else { b + = 1; }
               break;
```

```
    case jun : if (b == 30) { b = 1; }
               else { b += 1; }
               break;

    case jul : if (b == 31) { b = 1; }
               else { b += 1; }
               break;

    case aug : if (b == 31) { b = 1; }
               else { b += 1; }
               break;

    case sep : if (b == 30) { b = 1; }
               else { b += 1; }
               break;

    case oct : if (b == 31) { b = 1; }
               else { b += 1; }
               break;

    case nov : if (b == 30) { b = 1; }
               else { b += 1; }
               break;

    case dec : if (b == 31) { b = 1; }
               else { b += 1; }
               break;

    }
    return b;
}
```

7-16. 20세기의 날짜는 day/month/year의 형태의 정수들로 표시할 수 있다. 예를 들어 1/7/33은 1933년 7월 1일을 나타낸다. Day,month,year를 압축해서 저장하는 함수를 작성하여라. Day는 31개, month는 12개,year는 100개의 다른 값을 가질 수 있다. 따라서 day를 나타낸기 위해서는 5비트면 되고, month는 4비트, year는 7비트면 된다. 이 함수는 day,month,year를 정수로 입력 받아서 16비트 정수에 패킹하는 것이다. 언패킹을 수행하는 함수도 작성해 보아라. 그리고 작성한 함수를 검사하는 프로그램도 만들어 보아라.

Sol)

C code

```
#include <stdio.h>
#include <math.h>
#include <limits.h>
int packing(int,int,int);
int unpacking_month(int);
int unpacking_year(int);
void bit_print(int);
int main(void)
{
int day,mon,year,x;
scanf("%d",&day);
printf("Day : %d ₩n",day);
scanf("%d",&mon);
printf("Month : %d₩n",mon);
scanf("%d",&year);
printf("Year : %d₩n",year);

x = packing (day,mon,year);
bit_print(x);
printf("=   %d ₩n",x);
printf("Unpacking Day : %d ₩n",unpacking_day(x));
printf("Unpacking Month : %d ₩n",unpacking_month(x));
printf("Unpacking Year : %d ₩n",unpacking_year(x));
```

```
return 0;
}

int packing (int day,int month,int year)
{
   int re_num;
   re_num = (day<<11)+ (month<<7)+ year;
   return re_num;
}

int unpacking_day (int re_num)
{
   return (re_num >> 11);
}
int unpacking_month (int re_num)
{ int x = 0x00000780;
   return ((x & re_num)>>7);
}
int unpacking_year(int re_num)
{
 int x = 0x0000007f;
   return (x & re_num);
}
void bit_print(int a)
{
 int i;
 int n = sizeof(int)* CHAR_BIT;
 int mask = 1 << (n-1);
 for (i=1; i<=n; ++i)
 { putchar(((a & mask) == 0)?'0' : '1');
  a <<= 1;
  if (i % CHAR_BIT == 0 && i < n)
 putchar (' ');
 }
}
```

7-17. 연습문제 16번에서 언급한 패킹한 날짜를 입력받아서 다음 날을 리턴하는 함수를 작성하여라.

Sol) 생략

7-18. 4.10절 " 예제 : 부울 변수"에서 제시한 프로그램을 다시 작성하여라. 이때, char 형 변수 b의 하위 5개 비트를 사용하여 5개의 부울 변수 b1, ... , b5를 나타내도록하여라.

Sol)

C code

```
#include <stdio.h>
int bit_ret(char,int);
int main(void)
{
  char b = 0;
  int b1,b2,b3,b4,b5;
  int cnt = 0;
  printf("₩n%5s%5s%5s%5s%5s%5s%7s%7s%11s₩n₩n",
          "Cnt","b1","b2","b3","b4","b5","fct1","fct2","majority");
  while((int) b <= 0x0000001f)
  { b1 = bit_ret(b,1);
    b2 = bit_ret(b,2);
    b3 = bit_ret(b,3);
    b4 = bit_ret(b,4);
    b5 = bit_ret(b,5);
    printf("%5d%5d%5d%5d%5d%5d%6d%7d%9d₩n",++cnt,b1,b2,b3,b4,b5,
           b1||b3||b5,b1&&b2||b4&&b5,b1+b2+b3+b4+b5 >= 3);
    b += 1;
  }
return 0;
}
int bit_ret(char a,int c)
{
```

```
    int e = (int) a;
    int d;
    d = 0x1 << (5-c);
    e = (e & d) >> (5-c);
    return e;
}
```

Result (결과)

```
[romance@161s ch7_code]$ ./a.out
```

Cnt	b1	b2	b3	b4	b5	fct1	fct2	majority
1	0	0	0	0	0	0	0	0
2	0	0	0	0	1	1	0	0
3	0	0	0	1	0	0	0	0
4	0	0	0	1	1	1	1	0
5	0	0	1	0	0	1	0	0
6	0	0	1	0	1	1	0	0
7	0	0	1	1	0	1	0	0
8	0	0	1	1	1	1	1	1
9	0	1	0	0	0	0	0	0
10	0	1	0	0	1	1	0	0
11	0	1	0	1	0	0	0	0
12	0	1	0	1	1	1	1	1
13	0	1	1	0	0	1	0	0
14	0	1	1	0	1	1	0	1
15	0	1	1	1	0	1	0	1
16	0	1	1	1	1	1	1	1
17	1	0	0	0	0	1	0	0
18	1	0	0	0	1	1	0	0
19	1	0	0	1	0	1	0	0
20	1	0	0	1	1	1	1	1
21	1	0	1	0	0	1	0	0
22	1	0	1	0	1	1	0	1
23	1	0	1	1	0	1	0	1
24	1	0	1	1	1	1	1	1
25	1	1	0	0	0	1	1	0

```
 26   1   1   0   0   1   1   1   1
 27   1   1   0   1   0   1   1   1
 28   1   1   0   1   1   1   1   1
 29   1   1   1   0   0   1   1   1
 30   1   1   1   0   1   1   1   1
 31   1   1   1   1   0   1   1   1
 32   1   1   1   1   1   1   1   1
[romance@161s ch7_code]$
```

7-19. 연습문제 18번에서 작성한 프로그램을 컴퓨터의 산술연산을 이용하여 다시 작성하여라. 손으로 b의 비트 표현에 1을 더하는 것은 중첩된 for 문을 사용하는 것과 같은 효과를 가져옴을 보여라.

Sol) 위의 7-18. 문제에서 이미 그러한 방식으로 프로그램을 작성하였다.
여기에 while문을 for문으로 바꾸기만 하면 해결된다.

7-20. 5가지의 기본적인 음식그룹들을 정의하기 위해 열거형을 사용하여라: 어류,과일류, 곡류,육류,채소류. 각 음식그룹에서부터 한 가지 음식을 선택하기위해 난수 발생기를 사용 하여라. 5개의 음식 그룹으로부터 한가지씩의 음식을 선택하여 메뉴를 출력하는 함수 meal()을 작성하여라. 20가지 메뉴를 출력하여라.

Sol) 생략

7-21. 한 벌의 카드에서 임의로 5장의 카드를 선택하는 함수를 작성하여라. 작성된 함수는 선택한 모든 카드가 유일 한지를 검사해야 한다. 각 카드는 점수와 무늬가 있다. 7과 하트가 그려진카드는 점수값이 7이고, 무늬값은 하트이다. 에이스에 대한 점수는 1이고, 듀스는 2, ….. 그리고 킹은 13이다. 작성하는 함수에서 점수와 무늬의 값을 표현하기 위해 열거형을 사용하여라. 그리고, 선택된 카드를 출력하는 다른 함수를 작성하여라.

Sol)

C code

```
#include <stdio.h>
#include <time.h>
#include <stdlib.h>
#include <ctype.h>
enum card_num {A,Two,Three,Four,Five,Six,Seven,Eight,Nine,J,Q,K };
enum card_pic {Dia,Hea,Clo,Spa};
typedef enum card_num card_num;
typedef enum card_pic card_pic;
void card_prn(card_num,card_pic);
int main(void)
{
card_num a;
card_pic b;
int c;
srand(time(NULL));

a = rand() % 12;
b = rand() % 4;
printf("%d,%d₩n",a,b);
card_prn (a,b);

return 0;
}

void card_prn(card_num a,card_pic b)
{
   switch(a)
   { case A : printf("[A,"); break;
      case Two : printf("[2,"); break;
      case Three : printf("[3,"); break;
      case Four : printf("[4,"); break;
      case Five : printf("[5,"); break;
      case Six : printf("[6,"); break;
      case Seven : printf("[7,"); break;
```

```
      case Eight : printf("[8,"); break;
      case Nine : printf("[9,"); break;
      case J : printf("[J,"); break;
      case Q : printf("[Q,"); break;
      case K : printf("[K,"); break;
   }
   switch(b)
   { case Dia : printf(" ◆ ]₩n"); break;
      case Hea : printf(" ♥ ]₩n"); break;
      case Clo : printf(" ♣ ]₩n"); break;
      case Spa : printf(" ♠ )₩n"); break;
   }
}
```

Result (결과)

```
romance@161s ch7_code]$ ./a.out
11,2
[K, ♣ ]
[romance@161s ch7_code]$ ./a.out
6,2
[7, ♣ ]
[romance@161s ch7_code]$ ./a.out
7,2
[8, ♣ ]
[romance@161s ch7_code]$ ./a.out
4,1
[5, ♥ ]
[romance@161s ch7_code]$
```

7-22. 연습문제 21번의 함수에 의해 생성된 카드 패의 내용이 스트레이트, 플러시, 또는 풀하우스 인지를 검사하는 루틴을 작성하여라.

Sol) 생략.

7-23. "가위,바위,보" 게임에서 무승부가 아닌 경우의 출력은 다음 중 하나이다.
You win.
You lose.
출력되는 메시지가 다음과 같은 형태가 되도록 프로그램을 수정하여라.
You Chose paper and I chose rock. You win.

Sol) selection_by_machine() 함수에서

```
- p_r_n selection_by_machine(void)
- { p_r_n c = rand() % 3;
-   switch(c) { case paper : printf("You Chose paper "); break;
-               case rock : printf("You Chose rock");   break;
-               case scissors : printf("You Chose scissors");
-             }
-   return c;
- }
```

마찬가지로 selection_by_player에도 이와 같은 부분을 끼워 넣게 되면 된다.

7-24. 7.4절 "패킹과 언패킹"에서 제시한 pack() 함수를 생각해 보자. 이 함수의 몸체는 4개의 문장으로 이루어져 있다. 이 4개의 문장이 하나의 return 문을 통합되도록 함수를 다시 작성하여라.
Sol)

```
int pack(char a,char b,char c,char d)
{ return (( ((((a << CHAR_BIT) | b)<<CHAR_BIT) | c)<<CHAR_BIT) | d); }
```

7-25. pack() 함수를 산술 연산만을 사용하여 작성하여라.

Sol) 생략

7-26. 어떤 컴퓨터에서는 long형의 마스크를 허용한다. 그러나 2 바이트 컴퓨터에서 다음과 같이 초기화 할 경우, 작성된 코드는 기대했던 것 처럼 동작하지 않는다.

- long mask = 1 << 31;

Sol) 잘 동작한다.

7-27. 'abc'와 같은 다중문자 상수의 저장 방법은 시스템에 따라 다르다. 프로그래머는 때때로 "abc"를 'abc'라고 쓸 수 있기 때문에, 어떤 컴파일러는 다중문자 문자상수가 사용되면 비록 그 내용이 타당하더라도 경고 메시지를 생성한다. 사용하고 있는 시스템에서는 어떻게 되는가? 다으 코드를 수행해 보아라.

- int c = 'abc';
- printf("'abc' = ");
- bit_print(c);
- printf("₩n");

Sol)

Result (결과)

```
[romance@161s ch7_code]$ cc 7_27.c
7_27.c: In function `main':
7_27.c:6: warning: multi-character character constant
[romance@161s ch7_code]$./a.out
'abc' = 00000000 01100001 01100010 01100011
[romance@161s ch7_code]$
```

7-28. 집합에 대한 수학적 개념의 유용한 구현은 unsigned long을 최대 32개의 원소로 구성된 집합으로 다루는 것이다.

- typedef unsigned long set;
- const set empty = 0x0;

비트 연산자를 사용하여 두 집합을 합하는 루틴을 작성하여라.

- set Union(set a,set b);

마스크를 사용하면 원하는 비트가 1인지를 검사할 수 있다. 이 아이디어를 이용하여 집합의 구성원소를 출력하는 다음과 같은 함수 원형을 갖는 함수를 작성하여라.

- void display(set a);

이 함수를 검사하기 위해 다음 코드를 사용할 수 있을 것이다.

- set a = 0x7;
- set b = 0x5;
- display(Union(a,b));

Sol)

C code

```
#include <stdio.h>
#include <limits.h>
typedef unsigned long set;
const set empty = 0x0;
set Union(set,set);
void display(set);
void bit_print(int);
int main(void)
{
set a = 0x7;
set b = 0x55;
display(a);
display(b);
display(Union(a,b));
return 0;
}
set Union(set a,set b)
{ return a | b; }
void display(set a)
{ bit_print(a);
   putchar('₩n');
}
void bit_print(int a){ int i; int n = sizeof(int)* CHAR_BIT; int mask = 1 << (n-1);
```

```
for (i=1; i<=n; ++i) { putchar(((a & mask) == 0)?'0' : '1');  a <<= 1;  if (i % CHAR_BIT == 0 && i < n) putchar (' '); }}
```

Result (결과)

```
[romance@161s ch7_code]$ ./a.out
00000000 00000000 00000000 00000111
00000000 00000000 00000000 01010101
00000000 00000000 00000000 01010111
[romance@161s ch7_code]$
```

7-29. 연습문제 28번에 기술된 아이디어를 사용하여, 원소의 개수가 32개이하인 집합들을 다루는 완전한 집합 패키지를 개발하여라. 따라서, 다음과 같은 함수들을 작성해야 한다.

- set Union(set a,set b);
- set intersection(set a, set b);
- set complement(set a);

Sol)

C code

```
#include <stdio.h>
#include <limits.h>
typedef unsigned long set;
const set empty = 0x0;
set Union(set,set);
void display(set);
void bit_print(int);
set intersection(set,set);
set complement(set);
int main(void)
{
set a = 0x7;
set b = 0x55;
display(a);
display(b);
```

```
display(Union(a,b));
display(intersection(a,b));
display(complement(a));
display(complement(b));
return 0;
}
set Union(set a,set b)
{ return a | b; }
set intersection(set a,set b)
{ return a & b; }
set complement(set a)
{ return ~a ; }
void display(set a)
{ bit_print(a);
   putchar('₩n');
}
void bit_print(int a){ int i; int n = sizeof(int)* CHAR_BIT; int mask = 1 << (n-1);
for (i=1; i<=n; ++i) { putchar(((a & mask) == 0)?'0' : '1');  a <<= 1;  if (i %
CHAR_BIT == 0 && i < n) putchar (' '); }}
```

Chapter 7 End Point.