실 습 문 제 제15장

1. [비트 연산자] 정수를 입력으로 받아서 이진수의 형태로 출력하는 함수를 작성하여 보자.

```
#include <stdio.h>

void print_binary(int v)
{
    int i;

    for(i = 0; i < 32; i++)
    {
        if( (v << i) & 0x80000000 )
            printf("1");
        else
            printf("0");
    }
    printf("\n");
}

int main(void)
{
    print_binary(0xff);
    return 0;
}
```

(a) 위의 프로그램을 컴파일하여 실행하고, 함수의 인수로 0, 1, 32, 0xff을 대입하여 보고 그 결과를 기록하라.

실행결과

(b) 4비트마다 스페이스를 출력하여 보다 알아보기 쉽게 출력하여 보라.

(c) 사용자로부터 2개의 정수를 받아서 이진수 형태로 출력한 후에, 비트 논리곱 연산을 하고 그 결과도 이진수 형태로 출력하여 보라.

```
첫번째 정수: 5
두번째 정수: 3
첫번째 정수: 0000 0000 0000 0000 0000 0000 0000 0101
첫번째 정수; 0000 0000 0000 0000 0000 0000 0000 0011
결과값     : 0000 0000 0000 0000 0000 0000 0000 0001
```

(d) 비트 논리합, 비트 배타적 논리합, 비트 부정 연산에 대해서도 이진수 형태로 출력하여 보라.

2. [비트 이동 연산자] 비트 이동 연산을 이용하여 문자 4개를 받아서 하나의 unsigned int 형의 변수 안에 저장하는 프로그램을 작성하라. 첫번째 문자는 비트 0부터 비트 7까지에 저장되고 두번째 문자는 비트 8부터 비트 15까지 세번째 문자는 비트 16에서 비트 23까지, 네번째 문자는 비트 24부터 비트 31까지에 저장된다. 결과로 생성되는 정수값을 비트 단위로 출력하도록 한다. 비트 이동 연산과 비트 논리곱 및 비트 논리합 연산을 사용하라.

```
#include <stdio.h>

void print_binary(int v)
{
    ...
}

int pack_char(char a, char b, char c, char d)
{
    return a << 24 | b << 16 | c << 8 | d;
}

int main(void)
{
    int v;

    v = pack_char('a', 'b', 'c', 'd');
    print_binary(v);
    unpack(v);

    return 0;
}
```

(a) 위의 프로그램을 컴파일하여 실행하고 그 결과를 기록하라.

실행결과

(b) 사용자로부터 한 줄을 입력 받아 문자 배열에 저장한 후에, 문자 배열에서 문자 4개를 읽어서 정수에 저장한 후에 이진 파일에 쓰도록 프로그램을 수정하라.

(c) 텍스트 파일을 열어서 텍스트 파일에서 한 줄을 읽은 후에, 문자 배열에서 문자 4개를 읽어서 정수에 저장한 후에 이진 파일에 쓰도록 프로그램을 수정하라.

(d) `unpack()` 함수를 작성하고 실행하라. `unpack()` 함수는 정수에 채워진 4개의 문자를 분리한다.

(e) `unpack()` 함수를 이용하여 생성된 이진 파일을 다시 읽어서 문자열의 형태로 변환하여 보자. 원본 파일과 비교하여 검증하라.

3. [비트 필드]

```
#include <stdio.h>

struct test {
    unsigned a:8;
    unsigned b:1;
    unsigned c:3;
    unsigned d:4;
};

int main(void)
{
    struct test bf;

    bf.a = 0xff;
    bf.b = 0;
    bf.c = 1;
    bf.d = 0xf;

    printf("bf = %x\n", bf);
}
```

(a) 위의 프로그램을 컴파일하여 실행하고 그 결과를 기록하라. 비트 필드는 32비트의 unsigned int 변수 안에서 어디부터 만들어지는가?

실행결과

(b) sizeof 연산자를 bf에 적용시켜 보라. 어떤 값이 출력되는가?

(c) 현재 사용하지 않는 16비트를 사용하기 위하여 12비트 크기의 비트 필드 e와 6비트 크기의 비트필드 f를 만들고 거기에 각각 3과 4를 대입하여 보라.

(d) 비트 필드의 크기를 넘어서는 값을 대입하면 어떻게 되는지 실험하여 보라. 예를 들어서 c에 10을 대입하여 보라.

(e) 비트 필드와 일반 멤버를 같이 선언할 수 있는가? double형 변수 g와 int형 변수 h를 같이 선언하여 보라. 값을 대입해보고 어디에 만들어지는지를 알아보라.

4. [전처리]

```
#include <stdio.h>

#define SQUARE1(x)  (x * x)
#define SQUARE2(x)  (x) * (x)
#define SQUARE3(x)  ((x) * (x))

int main(void)
```

```
{
    printf("3의 제곱 = %d\n", SQUARE1(3));
    printf("3의 제곱 = %d\n", SQUARE2(3));
    printf("3의 제곱 = %d\n", SQUARE3(3));

    return 0;
}
```

(a) 위의 프로그램을 컴파일하여 실행하고 그 결과를 기록하라.

실행결과

(b) (1+2)를 3개의 매크로 함수에 인수로 주어서 호출해보라. 즉 SQUARE1(1+2), SQUARE2(1+2), SQUARE3(1+2)와 같이 호출하여 계산하여 보라. 결과를 설명하라.

(c) 8 / SQUARE1(2), 8 / SQUARE2(2), 8 / SQUARE2(2)와 같이 호출하여 계산하여 보라. 결과를 설명하라.

(d) SQUARE()를 정의할 때 줄 끝에 세미콜론을 붙이면 어떻게 되는가? 실험하여 보고 결과를 설명하라.

```
#define SQUARE3(x)  ((x) * (x));
```

(e) 내장 매크로인 __DATE__, __TIME__, __LINE__, __FILE__을 DEBUG라는 기호 상수가 1인 경우에만 출력하는 문장을 작성하여 보라. DEBUG를 1로 선언하고 컴파일하여 보라.