- 9장 함수와 변수 연습문제 풀이

01 다음의 소스 안에 적합한 변수를 선언하여 보라.

```
#include <stdio.h>
void f(void);

int main(void)
{
        ...
}
void f(void)
{
        ...
}
```

① 함수 f() 안에서만 사용하는 int형 변수 number
② 모든 함수에서 사용하는 double형 변수 radio
③ 함수 f() 안에서 사용되고 CPU 레지스터에 저장되어야 하는 변수 index
④ 모든 함수에서 사용되고 다른 소스 파일에서 정의된 변수 counter
⑤ 함수 main()에서 사용되고 이전의 값이 유지되는 변수 setting

02 다음 프로그램에 등장하는 변수들의 범위, 생존 기간, 연결을 설명하라.

```
#include <stdio.h>

int a;
static int b;
extern int c;

int main(void)
{
        int d;
        register int e;
        static int f;
        {
            int g;
        }
        return 0;
}
```

1.

```
#include <stdio.h>
void f(void);
double ratio;     // (b)
extern int counter;       // (d)

int main(void)
{
        static int setting;         // (f)
        ...
}
void f(void)
{
        int number; // (a)
        register int index; // (c)
        extern int total; // (e)
        ...
}
```

2.

```
#include <stdio.h>

int a;          // 파일 전체, 정적, 연결 가능
static int b;   // 파일 전체, 정적, 연결 불가능
extern int c;   // 파일 전체, 정적, 외부 변수 참조

int main(void)
{
        int d;   // 블록, 자동, 연결 불가능
        register int e;   // 블록, 자동, 연결 불가능
        static int f;     // 블록, 정적, 연결 불가능
        {
                int g;   // 블록, 자동, 연결 불가능
        }
        return 0;
}
```

```
#include <stdio.h>
void f(void);

int i;
int main(void)
{
        for(i = 0;i < 3; i++)
        {
                printf("*");
                f();
        }
        return 0;
}
void f(void)
{
        for(i = 0;i < 5; i++)
                printf("#");
}
```

```
*#####
```

```
#include <stdio.h>
void f(void);
int x = 1;

int main(void)
{
        int x = 2;
        printf("%d\n", x);
        {
                int x = 3;
                printf("%d\n", x);
        }
        printf("%d\n", x);
        return 0;
}
```

```
2
3
2
```

03 다음 프로그램의 출력을 쓰고, 이유를 설명하라.

(a)

```c
#include <stdio.h>
void f(void);

int i;
int main(void)
{
        for(i = 0;i < 3; i++)
            f();
        return 0;
}
void f(void)
{
        for(i = 0;i < 5; i++)
            printf("#");
}
```

(b)

```c
#include <stdio.h>
void f(int);
int n = 10;
int main(void)
{
        f(n);
        printf("n=%d\n", n);
        return 0;
}
void f(int n)
{
        n = 20;
}
```

```
n=10
```

(a)

```c
#include <stdio.h>
void f(void);
int x = 1;

int main(void)
{
        int x = 2;
        {
            int x = 3;
            printf("%d\n", x);
        }
        printf("%d\n", x);
        return 0;
}
```

(b)

```c
#include <stdio.h>
void f(void);

int main(void)
{
        f();
        f();
        return 0;
}
void f(void)
{
        static int count = 0;
        printf("%d\n", count++);
}
```

```
0
1
```

04 다음 소스에 오류가 있는지를 먼저 판단하고 오류가 있다면 오류를 지적하라. 논리적인 오류도 포함된다.

(a)

```c
#include <stdio.h>

register auto int x = 20;

int main(void)
{
        printf("%d\n", x);
        return 0;
}
```

(b)

```c
int recursive(int n)
{
        if( n == 1 ) return 0;
        return n * recursive(n);
}
```

4.

(a) 하나 이상의 저장 유형 지정자를 붙이면 안 된다.

(b) 재귀 호출할 때 매개 변수의 값이 줄어들지 않아서 무한히 재귀 호출됨

05 다음 함수를 주석과 같이 호출하는 경우에 화면에 출력되는 내용과 함수의 반환값을 구하라.

(a)

```c
// sum(5)로 호출
int sum(int n)
{
        printf("%d\n", n);
        if( n < 1) return 1;
        else return( n + sum(n - 1) );
}
```

(b)

```c
// recursive(5)로 호출
int recursive(int n)
{
        printf("%d\n", n);
        if( n < 1) return 2;
        else return( 2 * recursive(n - 1) + 1 );
}
```

(a)

```
5
4
3
2
1
0
```

반환값은 16

(b)

```
5
4
3
2
1
0
```

반환값은 95

06 다음의 순환적인 프로그램을 반복 구조를 사용한 비순환적 프로그램으로 바꾸어 보자.

```c
int recursive(int n)
{
        if( n == 0 ) return 1;
        else return (n + recursive(n-1));
}
```

6.

```c
int recursive(int n)
{
        int i, sum=0;
        for(i=n; i>=1; i--)
                sum += i;
        return sum;
}
```

01 덧셈, 뺄셈, 곱셈, 나눗셈을 지원하는 계산기 프로그램을 작성하여 보자. 이번에는 각 연산들이 몇 번씩 계산되었는지를 기억하게 하자. 각 연산을 지원하는 함수들은 자신이 호출된 횟수를 화면에 출력한다.

실행결과

① 정적 지역 변수를 사용하여 프로그램을 작성하라.

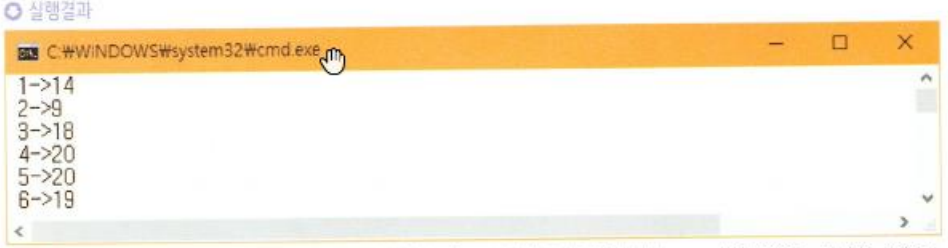
② 전역 변수를 사용하여 프로그램을 작성하라.

HINT 정적 지역 변수는 static int count;와 같이 선언한다.

```c
int main(void)
{
    char op;
    int x, y;
    int i;

    for (i = 0; i < 10; i++)
    {
        printf("연산을 입력하시오: ");
        scanf("%d %c %d", &x, &op, &y);
        if (op == '+')
            printf("연산 결과: %d \n", add(x, y));
        else if (op == '-')
            printf("연산 결과: %d \n", sub(x, y));
        else if (op == '*')
            printf("연산 결과: %d \n", mul(x, y));
        else if (op == '/')
            printf("연산 결과: %d \n", div(x, y));
        else
            printf("지원되지 않는 연산자입니다. \n");
    }
    return 0;
}
int add(int x, int y)
{
    static int count;
    count++;
    printf("덧셈은 총 %d번 실행되었습니다.\n", count);
    return (x + y);
}
```

02 주사위를 던져서 각각의 면이 몇 번 나왔는지를 출력하는 프로그램을 작성하라. 주사위의 면은 난수를 이용하여 생성한다. 주사위를 던지는 함수 get_dice_face()를 만들고 이 함수 안에서 각각의 면이 나올 때마다 그 횟수를 정적 지역 변수를 이용하여 기억하게 하라. get_dice_face() 호출 횟수가 100이 되면 각 면의 횟수를 출력한다.

실행결과



HINT 주사위 면은 0에서 5까지의 정수로 표현할 수 있고 0부터 5까지의 난수는 rand() % 6으로 생성할 수 있다.

```c
int main(void)
{
    int i;
    for (i = 0; i < 101; i++)
        get_dice_face();
    return 0;
}
void get_dice_face()
{
    static int one, two, three, four, five, six;
    static int call_count = 0;
    int face;

    call_count++;
    face = rand() % 6;
    if (face == 0) one++;
    else if (face == 1) two++;
    else if (face == 2) three++;
    else if (face == 3) four++;
    else if (face == 4) five++;
    else six++;
    if (call_count % 100 == 0)
        printf(" 1->%d\n 2->%d\n 3->%d\n 4->%d\n 5->%d\n 6->%d\n",
            one, two, three, four, five, six);
}
```

03 로그인시에 아이디를 검사하는 함수 int check()를 작성해서 테스트하라. check()가 한번 호출 될 때마다 비밀번호를 질문하고 일치 여부를 0과 1으로 반환한다. 비밀번호는 숫자 1234로 고정되어 있다고 가정한다. check()가 3번 이상 호출되고 아이디가 일치하지 않으면 check()는 "로그인 시도 횟수 초과" 메시지를 출력한다. check() 함수 안에 정적 변수를 선언하여 사용해보자.

실행결과

HINT 정적 지역 변수에 시도 횟수를 저장한다. 사용자가 로그인을 할 때마다 이 정적 지역 변수를 1만큼 증가시킨다.

```c
int check()
{
    static int call_count = 0;
    call_count++;
    if (call_count > 3) {
        printf("로그인 시도횟수 초과\n ");
        return 0;
    }
    printf("비밀번호: ");
    int n;
    scanf("%d", &n);
    if (n == 1234) {
        printf("로그인 성공!!\n ");
        return 1;
    }
    return 0;
}
int main(void)
{
    check();
    check();
    check();
    check();
    return 0;
}
```

04 본문에서 설명한 바와 같이 정적 변수는 초기화를 딱 한번만 수행하는 경우에도 사용된다. 난수를 생성하여 반환하는 함수 get_random()을 작성하여 테스트하라. get_random()이 처음으로 호출되는 경우에는 srand()를 호출하여서 난수의 시드를 초기화하고 그렇지 않으면 단순히 rand()를 호출하여 난수를 반환한다.

실행결과

HINT 정적 지역 변수 inited를 작성하여 사용해본다. 본문의 LAB 문제를 참조한다.

```c
int get_random()
{
    static int inited = 0;
    if (inited == 0)
    {
        printf("초기화 실행\n");
        srand((unsigned)time(NULL));
        inited = 1;
    }
    return rand();
}
int main(void)
{
    printf("%d\n", get_random());
    printf("%d\n", get_random());
    printf("%d\n", get_random());
    return 0;
}
```

05 1부터 n까지의 합 (1+2+3+...+n)을 계산하는 문제를 순환기법을 이용하여 작성해보자.

실행결과

HINT sum(3)은 3+sum(2)로 바꿀 수 있다.

```c
int main()
{
    int number, result;

    printf("정수를 입력하시오: ");
    scanf("%d", &number);

    result = sum(number);

    printf("1부터 %d까지의 합=%d\n", number, result);
}

int sum(int num)
{
    if (num != 0)
        return num + sum(num - 1);
    else
        return num;
}
```

06 순환기법을 이용하여 지수값을 계산하는 함수 power(int base, int power_raised)를 작성하고 테스트해보자. power(2, 3)가 호출되면 2^3을 계산하여 반환한다.

실행결과

HINT power(b, p)가 호출되면 이것을 b*power(b, p-1)로 바꾸어서 호출한다.

```c
int main(void)
{
    int base, powerRaised, result;
    printf("밑수: ");
    scanf("%d", &base);
    printf("지수: ");
    scanf("%d", &powerRaised);
    result = power(base, powerRaised);
    printf("%d^%d = %d\n", base, powerRaised, result);
    return 0;
}

int power(int base, int powerRaised)
{
    if (powerRaised != 0)
        return (base * power(base, powerRaised - 1));
    else
        return 1;
}
```

07 순환 호출을 이용하여 정수의 각 자리수를 출력하는 함수 show_digit(int x)를 작성하고 테스트하라. 즉 정수가 1234이면 화면에 1 2 3 4와 같이 출력한다. 함수는 일의 자리를 출력하고 나머지 부분을 대상으로 다시 같은 함수를 순환 호출한다. 예를 들어서 1234의 4를 출력하고 나머지 123을 가지고 다시 같은 함수를 순환 호출한다. 1234를 10으로 나누면 123이 되고 4는 1234를 10으로 나눈 나머지이다.

실행결과

HINT 정수 x를 10으로 나눈 값을 가지고 show_digit(x/10);와 같이 다시 순환호출하면 된다.

```c
int show_digit(int x)
{
    if (x / 10 > 0)
        show_digit(x / 10);
    printf("%d ", x % 10);
}

int main(void)
{
    int n;
    printf("정수를 입력하시오: ");
    scanf("%d", &n);
    show_digit(n);
    printf("\n");
    return 0;
}
```

08 주어진 정수가 몇 개의 자리수를 가지고 있는지를 계산하는 프로그램을 순환을 이용하여 작성해보자. 예를 들어서 12345의 경우에는 5가 출력된다.

실행결과

HINT 순환 호출 시에 어떤 데이터를 저장하려면 정적 지역 변수를 사용할 수 있다.

```c
int get_no_digits(int n);
int main()
{
    int n, ctr;
    printf("정수를 입력하시오: ");
    scanf("%d", &n);
    ctr = get_no_digits(n);
    printf("자리수의 개수: %d\n", ctr);
    return 0;
}
int get_no_digits(int n)
{
    if (n == 0)
        return 0;
    if (n < 10)
        return 1;
    else
        return (1 + get_no_digits(n / 10));
}
```

09 앞의 문제와 유사한 문제이다. 자리수의 합계를 계산하는 프로그램을 순환을 이용하여 작성해보자. 예를 들어서 123의 경우에는 6이 출력된다.

실행결과

HINT get_digit_sum(n)은 (n % 10) + get_digit_sum(n / 10))으로 나눌 수 있다.

```c
int main()
{
    int n, sum;
    printf("정수를 입력하시오: ");
    scanf("%d", &n);
    sum = get_digit_sum(n);
    printf("자리수의 합: %d\n", sum);
    return 0;
}

int get_digit_sum(int n)
{
    if (n == 0)
        return 0;

    return ((n % 10) + get_digit_sum(n / 10));
}
```

10 다음과 같은 수식의 값을 계산하는 순환적인 프로그램을 작성하라.

$$1/1+1/2+1/3+ \ldots +1/n$$

실행결과

HINT 위 수열의 값을 계산하는 식은 return 1.0/n + recursive(n-1);이다.

```c
double recursive(int n)
{
    if (n == 1) return 1;
    else return 1.0 / n + recursive(n - 1);
}

int main(void)
{
    int n;
    scanf("%d", &n);
    printf("%f\n", recursive(n));
    return 0;
}
```

11 이항 계수(binomial coefficient)를 계산하는 순환 함수를 작성하라. 이항 계수는 다음과 같이 순환적으로 정의된다.

$$ {}_nC_k = \begin{cases} {}_{n-1}C_{k-1} + {}_{n-1}C_k & \text{if } 0<k<n \\ 1 & \text{if } k=0 \text{ or } k=n \end{cases} $$

실행결과

HINT 순환적으로 정의된 수학식을 그대로 변환하면 된다. recursive(n-1, k-1) + recursive(n-1, k);와 같이 수식을 작성한다.

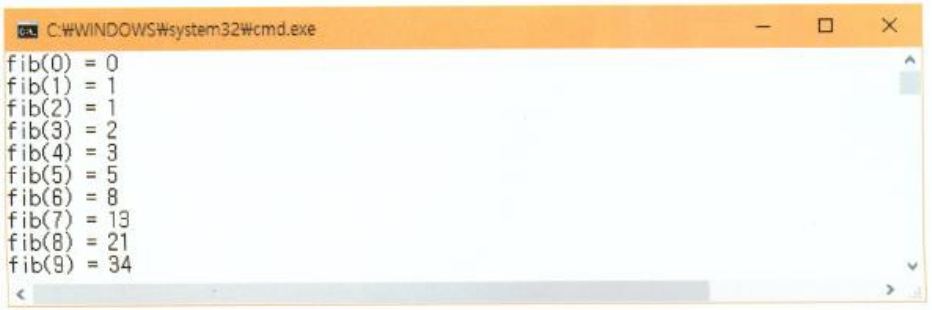
```c
int recursive(int n, int k)
{
    if (k == 0 || n == k) return 1;
    else return recursive(n - 1, k - 1) + recursive(n - 1, k);
}

int main(void)
{
    printf("n=");
    int n;
    scanf("%d", &n);
    printf("k=");
    int k;
    scanf("%d", &k);
    printf("%d\n", recursive(n, k));
    return 0;
}
```

12 순환 호출을 이용하여 피보나치 수열을 계산해 보자. 피보나치 수열이란 다음과 같이 정의되는 수열이다.

$$ fib(n) = \begin{cases} 0 & n=0 \\ 1 & n=1 \\ fib(n-2)+fib(n-1) & otherwise \end{cases} $$

즉 일반적인 경우, 앞의 두 개의 숫자를 더해서 뒤의 숫자를 만들면 된다. 정의에 따라 수열을 만들어 보면 다음과 같다.

$$ 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, \ldots $$




```c
int fib(int n)
{
    if (n == 0) return 0;
    if (n == 1) return 1;
    return (fib(n - 1) + fib(n - 2));
}
int main(void)
{
    for (int i = 0; i < 10; i++)
        printf("fib(%d) = %d\n", i, fib(i));
    return 0;
}
```