C 언어 스터디 4주차

복습

```c
#include<stdio.h>

int main(void)
{
    int base, exponent;
    int i, result = 1;

    printf("밑과 지수를 입력하세요 : ");
    scanf("%d %d", &base, &exponent);

    for (i = 0;i < exponent;i++)
    {
        result *= base;
    }

    printf("결과는 %d입니다.\n", result);

    return 0;
}
```

복습

```
3   int main(void)
4   {
5       int num;
6       int i, j;
7       int count;
8
9       printf("숫자를 입력하세요 : ");
10      scanf("%d", &num);
11
```

```
12      for (i = 2;i <= num;i++)
13      {
14          count = 0;
15          for (j = 2;j <= i / 2;j++)
16          {
17              if (i%j == 0)
18              {
19                  count++;
20                  break;
21              }
22          }
23
24          if (!count)
25          {
26              printf("%d\t", i);
27          }
28      }
29
30      printf("\n");
31
```

복습

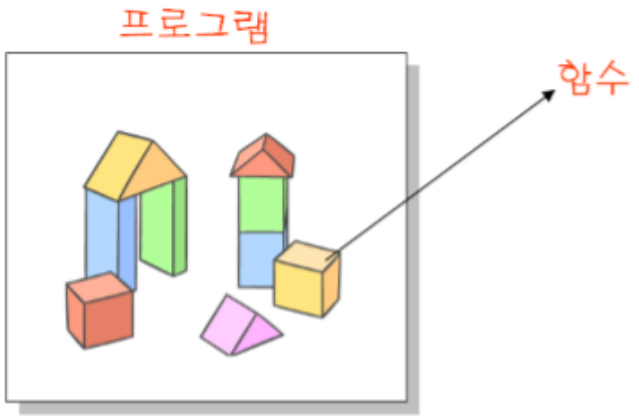
```
5       int i, j;
6       int number;
7
8       printf("줄 수를 입력하세요 : ");
9       scanf("%d", &number);
10
11      for (i = 1;i <= number;i++)
12      {
13          for (j = 1;j < i;j++)
14              printf(" ");
15
16          for (j = 1;j <= number - i + 1;j++)
17              printf("%d", j);
18
19          for (j = number - i;j > 0;j--)
20              printf("%d", j);
21
22          printf("\n");
23      }
```

함수란?

- 함수(function)은 프로그램을 구성하는 기본 요소.
- 한번 만들어지면 재사용할 수 있는 점과 가독성 증대, 유지관리 용이 등 장점이 있다.
- 함수는 입력 받아서 특정한 작업을 수행한 뒤 결과를 반환하는 역할.

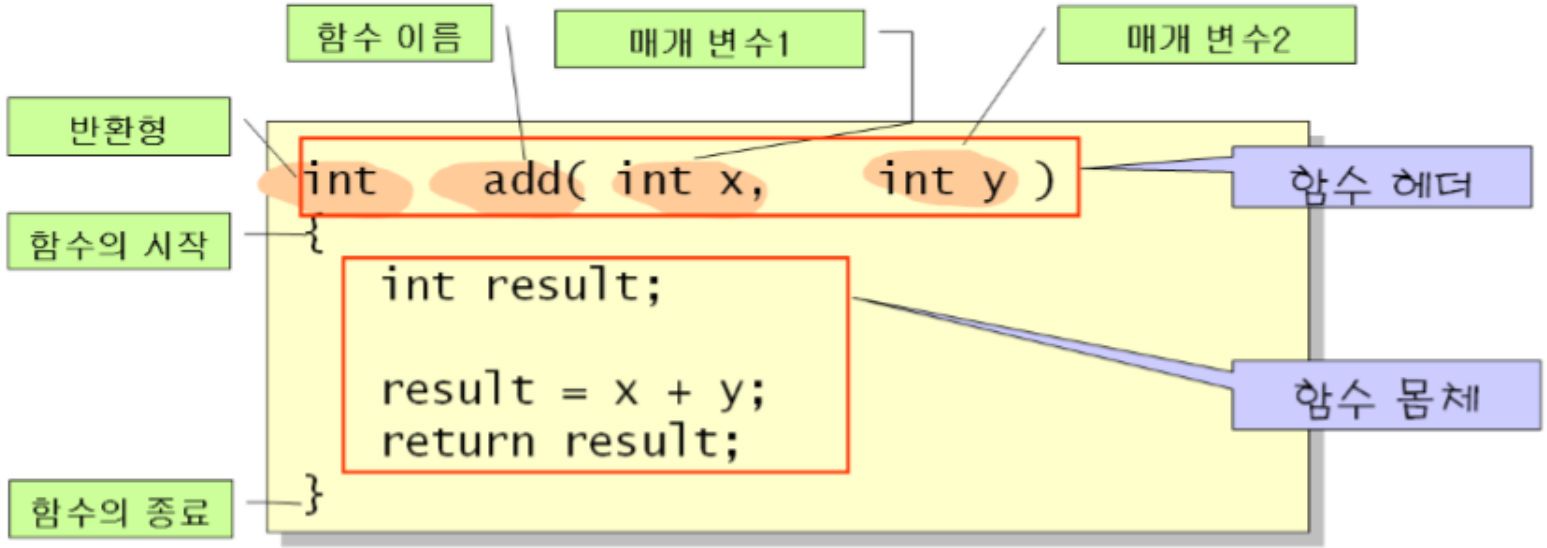
특징)

- 함수는 서로 구별되는 이름을 가지고 있다.
- 함수는 특정한 작업을 수행한다.
- 함수는 입력을 받을 수 있고 수행 결과를 반환할 수 있다.



함수 정의

- **함수 헤더**(function header)와 **함수 몸체**(function body)로 크게 나눈다.
- 함수 헤더 : 반환형 + 함수 이름 + 매개변수 목록.
- 함수 몸체 : 중괄호로 둘러싸인 부분. 작업에 필요한 문장.



함수 정의

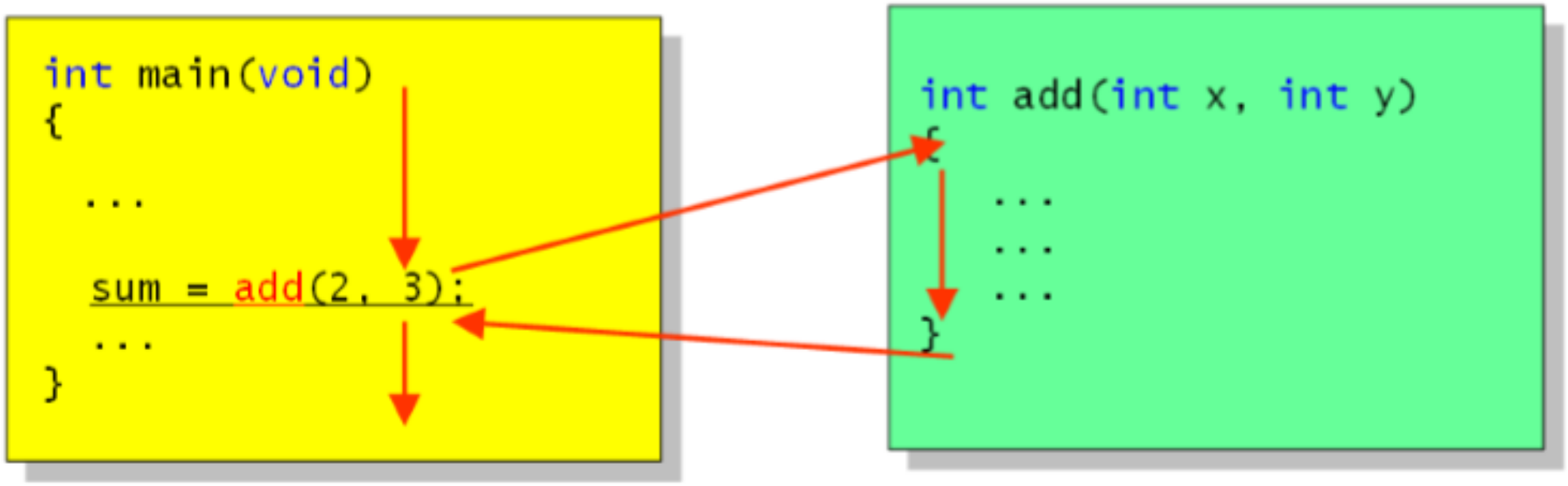
- **반환형** : 함수가 처리를 종료한 후 반환되는 데이터의 유형을 지정.
  (int, double, long, char 등등)
  ex) int square(); // int 형의 값을 반환한다.
  double average(); // double 형의 값을 반환한다.

- **매개변수** : 작업에 필요한 **데이터를 받는 변수**. 여러 개도 가능하며 쉼표로 구분한다. 각 매개변수에 대하여 자료형과 이름을 지정한다.
  ex) int square(int n);
  double average(double x, double y);

- **함수 이름** : 식별자 규칙만 만족한다면 무엇이든 OK. 기능과 관련한 이름으로 짓는 것이 좋음.

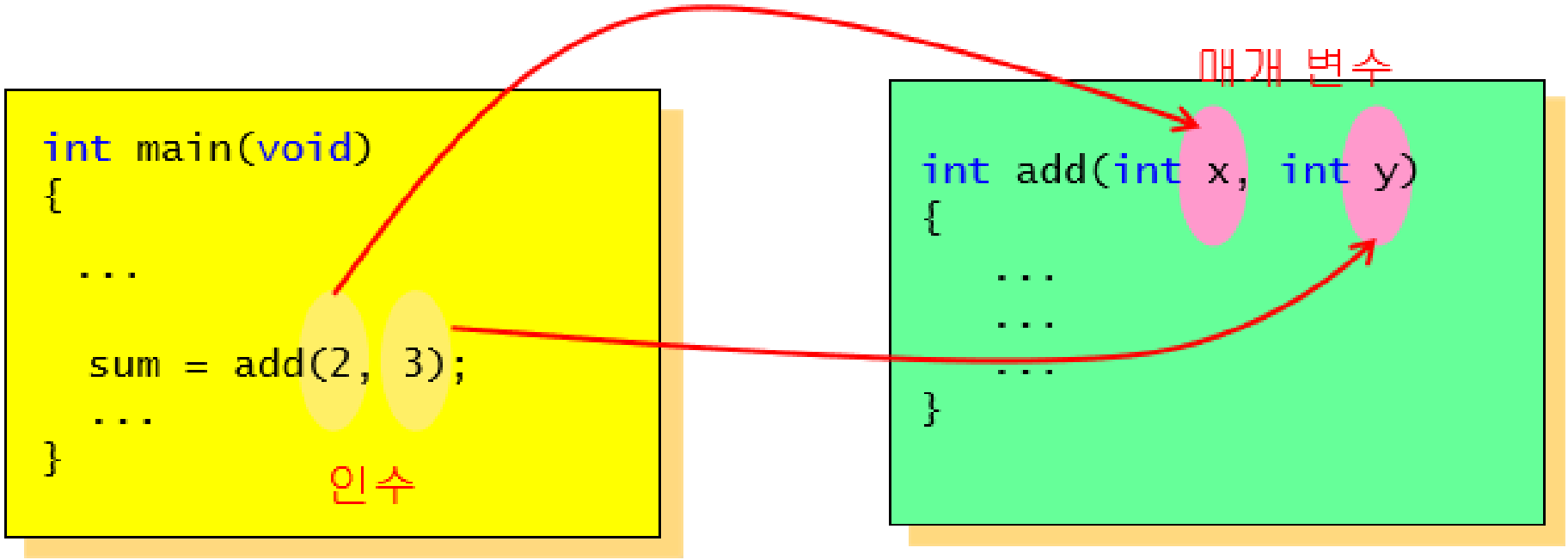
함수 호출과 반환

함수를 호출하게 되면 현재 실행하고 있는 코드는 잠시 중단되고, 호출된 함수로 이동한다.



함수 호출과 반환

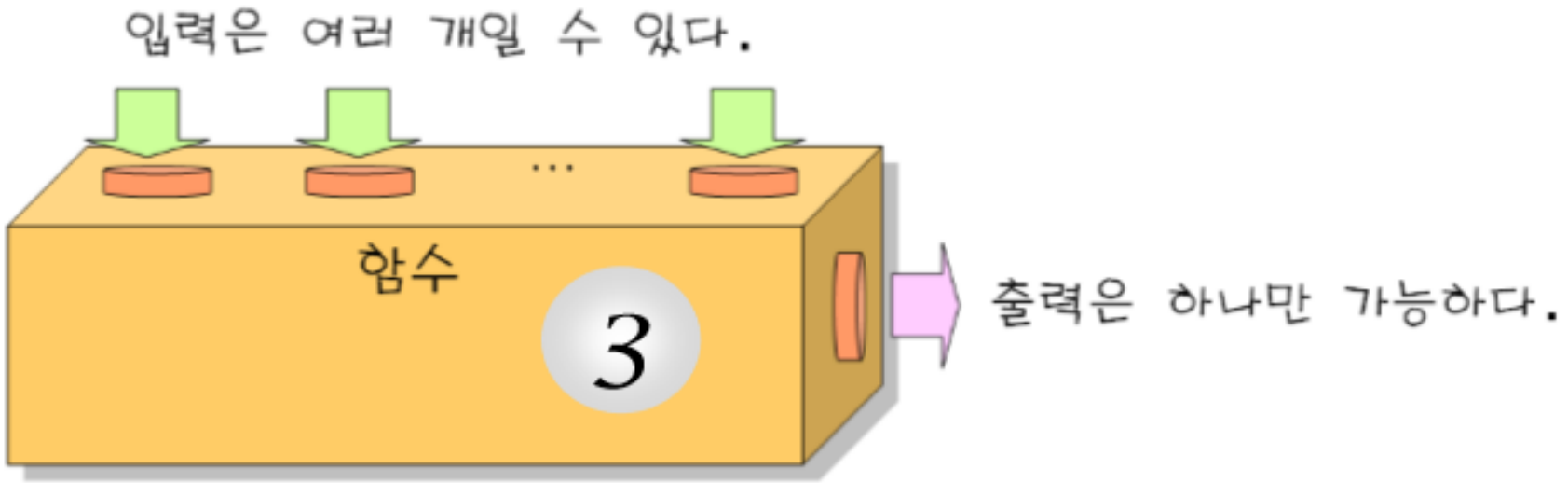
인수와 매개변수 – 함수가 호출될 때마다 인수는 함수의 매개변수로 전달된다.



함수 호출과 반환

반환값 – 함수가 호출된 곳으로 반환하는 작업의 **결과값**.

인수는 여러 개가 있을 수 있으나 반환값은 **하나만** 가능하다.
return 뒤에 반환하고자 하는 수식을 쓰면 된다. 유효한 수식이라면 무엇이든 가능!
Ex) return 30; , return x; , return x*x;



함수 정의 예제

```
int square(int n)
{
    return n * n;
}
```

제곱값을 구하는 함수.

```
int get_max(int x, int y)
{
    if (x > y)
        return x;
    else
        return y;
}
```

더 큰 수를 구하는 함수.

```
void draw_rect(int side)
{
    int x, y;

    for (y = 0; y < side; y++)
    {
        for (x = 0; x < side; x++)
        {
            printf("*");
        }
        printf("\n");
    }
}
```

정사각형을 그리는 함수.
return값이 없어도 되므로 void!!

함수 정의 예제

그래서 이걸 어떻게 쓰느냐??? 이렇게 씁니다.

```c
#include<stdio.h>

void draw_rect(int side)
{
    int x, y;

    for (y = 0; y < side; y++)
    {
        for (x = 0; x < side; x++)
        {
            printf("*");
        }
        printf("\n");
    }
}

int main() {
    draw_rect(5);
    return 0;
}
```

```c
#include<stdio.h>

int square(int n)
{
    return n * n;
}

int main() {
    int x;

    printf("제곱할 정수를 입력하세요 : ");
    scanf("%d", &x);

    printf("입력한 정수의 제곱은 %d입니다.\n", square(x));
    return 0;
}
```

함수 원형

일반적으로 함수를 사용할 때는 미리 함수 원형을 정의하여야 한다.

(전역 범위)

```
1   #include<stdio.h>
2
3   int compute_sum(int n);
4
5   int main() {
6
7       int sum = compute_sum(100);
8       printf("1부터 100까지의 합은 %d입니다.\n", sum);
9       return 0;
10  }
11  int compute_sum(int n)
12  {
13      int i;
14      int result = 0;
15      for (i = 1; i <= n; i++)
16          result += i;
17      return result;
18  }
```

라이브러리 함수

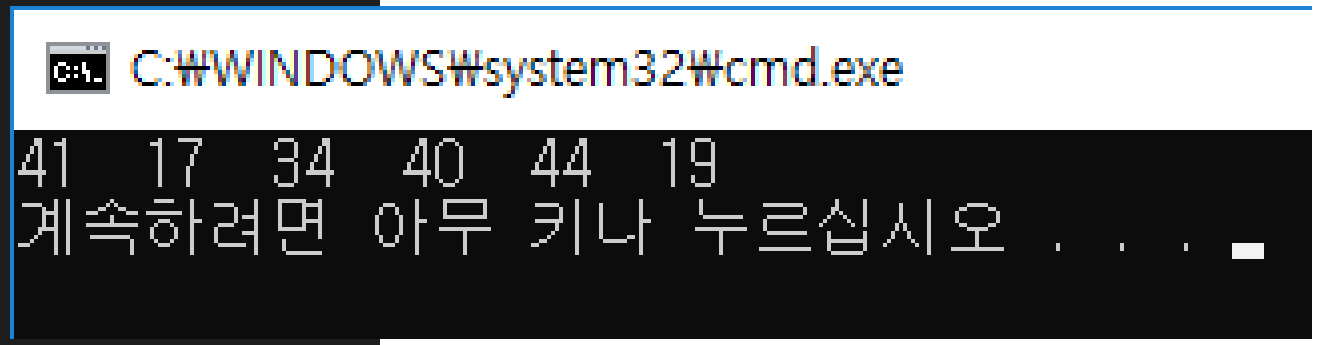
표준 라이브러리 함수는 컴파일러에서 제공하는 함수이다.

- 표준 입출력
- 수학 연산
- 문자열 처리
- 시간 처리
- 오류 처리
- 데이터 검색과 정렬

난수 함수

난수(random number) : 규칙성이 없이 임의로 생성되는 수
rand(); 처럼 사용하고 <stdlib.h> 헤더파일에 포함되어 있다.

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

int main()
{
    int i;

    for (i = 0; i < 6; i++)
    {
        printf("%d  ", 1+(rand() % 45));
    }
    printf("\n");
    return 0;
}
```

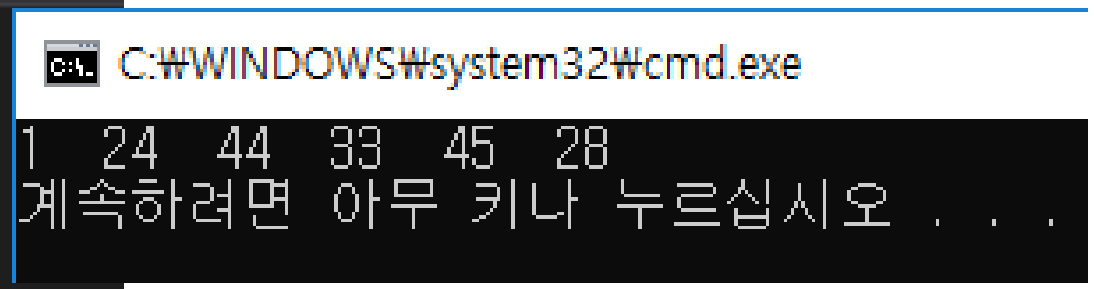


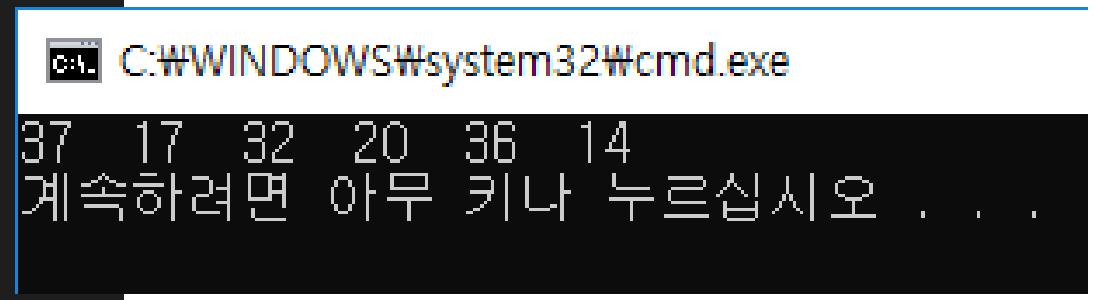
// % 45 를 한 이유는 45로 나눈 나머지는 45보다 클 수 없다는 사실을 이용
(45로 나눈 나머지의 범위는 0~44)

난수 함수

몇 번을 실행해도 결과가 같을 것임. 매번 난수를 다르게 생성하려면 시드 값을 바꿔준다.
srand((unsigned)time(NULL)); 와 같이 사용하며 time() 함수는 <time.h>에 포함되어 있다.

```c
#include<stdio.h>
#include<stdlib.h>
#include<time.h>

int main()
{
    int i;
    srand((unsigned)time(NULL));

    for (i = 0; i < 6; i++)
    {
        printf("%d  ", 1+(rand() % 45));
    }
    printf("\n");
    return 0;
}
```





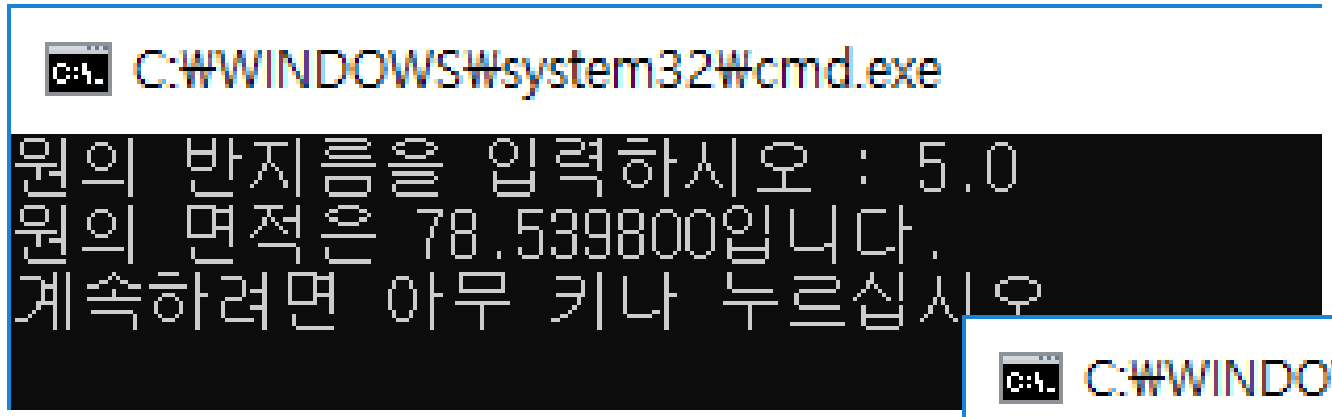
라이브러리 함수

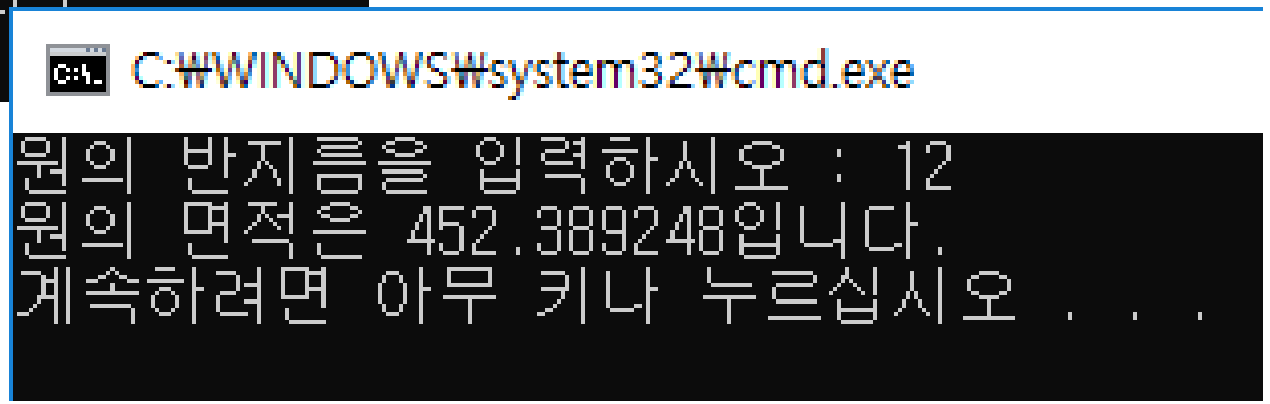
분류	함수	설명
삼각함수	double sin(double x)	사인값 계산
	double cos(double x)	코사인값 계산
	double tan(double x)	탄젠트값 계산
역삼각함수	double asin(double x)	역사인값 계산
	double acos(double x)	역코사인값 계산
	double atan(double x)	역탄젠트값 계산
지수함수	double exp(double x)	e^x
	double log(double x)	$log_e x$
	double log10(double x)	$log_{10} x$
기타함수	double fabs(double x)	실수 x의 절대값
	int abs(int x)	정수 x의 절대값
	double pow(double x, double y)	x^y
	double sqrt(double x)	$\sqrt{x}$

실습 문제

- 원의 면적을 구하는 문제를 함수로 작성하여라. 원의 면적을 구하는 함수 GetArea(double radius)를 작성하고 함수를 호출하여 전체 프로그램을 완성하여라.

hint) 1. 파이는 기호상수로 작성하여 보자.





실습 문제

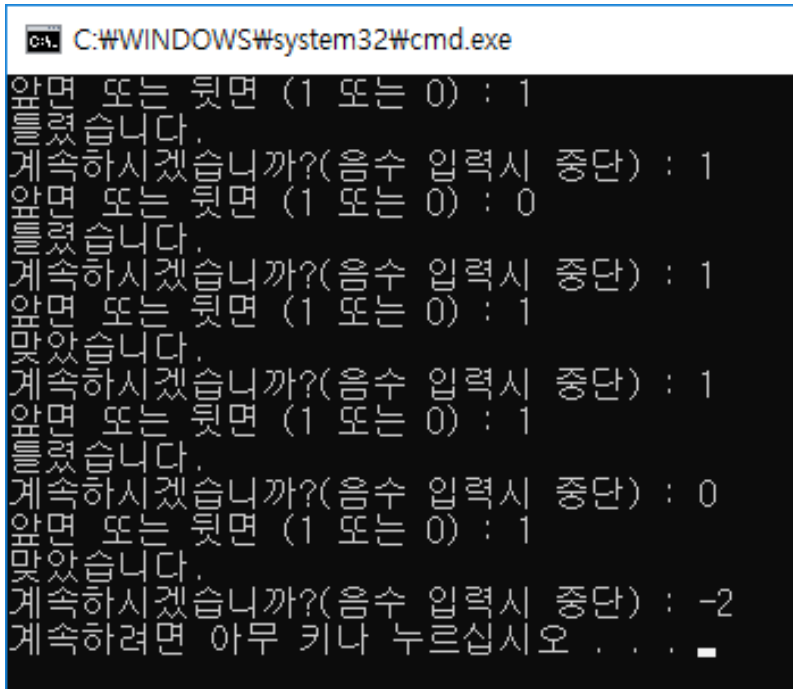
```c
#include<stdio.h>
#define PI 3.141592
double GetArea(double r);

int main(void)
{
    double radius, area;

    printf("원의 반지름을 입력하시오 : ");
    scanf("%lf", &radius);

    area = GetArea(radius);
    printf("원의 면적은 %lf입니다.\n", area);

    return 0;
}

double GetArea(double r)
{
    return PI * r * r;
}
```

실습 문제

- rand()함수를 이용하여 0 또는 1을 반환하는 b_rand()함수를 작성하고 이를 이용하여 동전 던지기 게임을 작성하라.

hint) 1. 반복루프를 만들어 사용자가 n을 입력할 때까지 반복한다.

2. 1 → 앞면, 0 → 뒷면



실습 문제

```c
#include<stdio.h>
#include<stdlib.h>
#include<time.h>
int b_rand();

int main(void)
{
    int guess;
    int after = 1;

    srand((unsigned)time(NULL));

    while (after >= 0)
    {
        printf("앞면 또는 뒷면 (1 또는 0) : ");
        scanf("%d", &guess);

        if (guess == b_rand())
            printf("맞았습니다.\n");
        else
            printf("틀렸습니다.\n");
        printf("계속하시겠습니까?(음수 입력 시 중단) : ");
        scanf("%d", &after);
    }

    return 0;
}

int b_rand()
{
    return rand() % 2;
}
```

변수의 속성

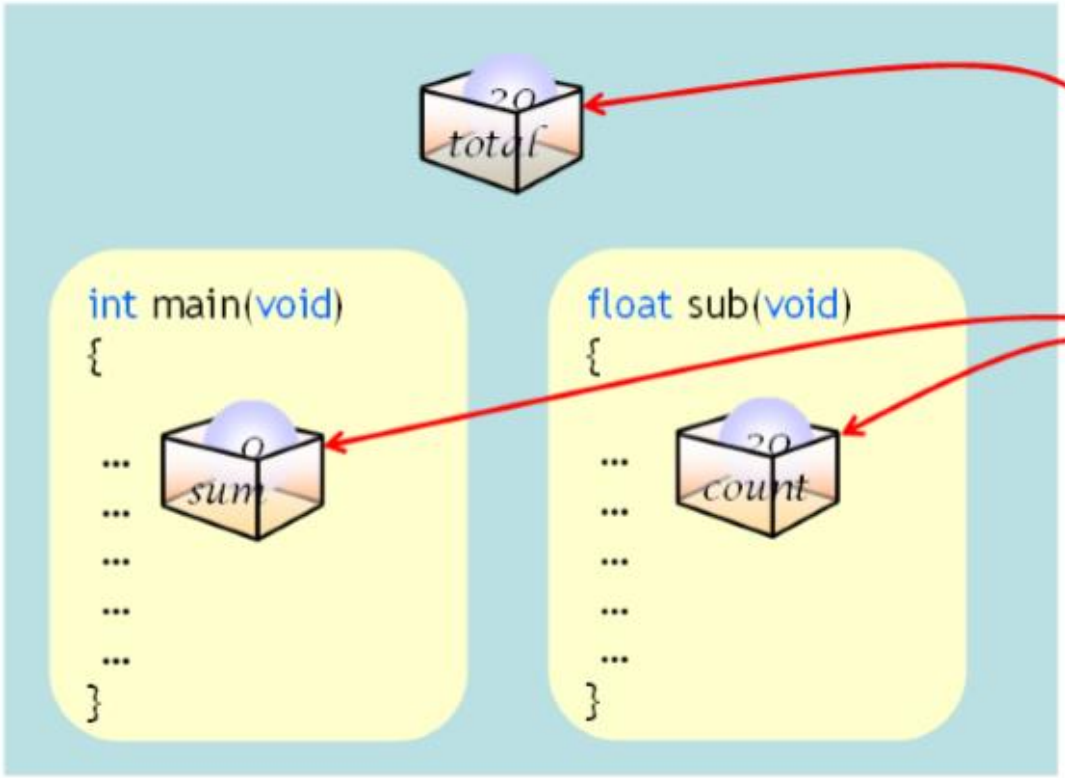
변수는 기본적으로 이름, 타입 같은 속성도 갖지만 3가지 추가 속성을 갖는다.

- **범위** : 변수가 어떤 범위에서 사용가능한가. 이 속성은 주로 변수가 어디에서 정의되는지에 따라 달라진다.
- **생존시간** : 변수가 메모리상에 얼마나 오랫동안 존재하는지. 변수가 선언되는 위치에 따라 잠깐 생존하였다가 없어지기도 함.
- **연결** : 서로 다른 영역에 있는 변수들을 연결하는데 사용되는 속성.

범위

변수는 선
크게 전역

- 지역변수 :
  사용 가능
- 전역변수 :
  하다.

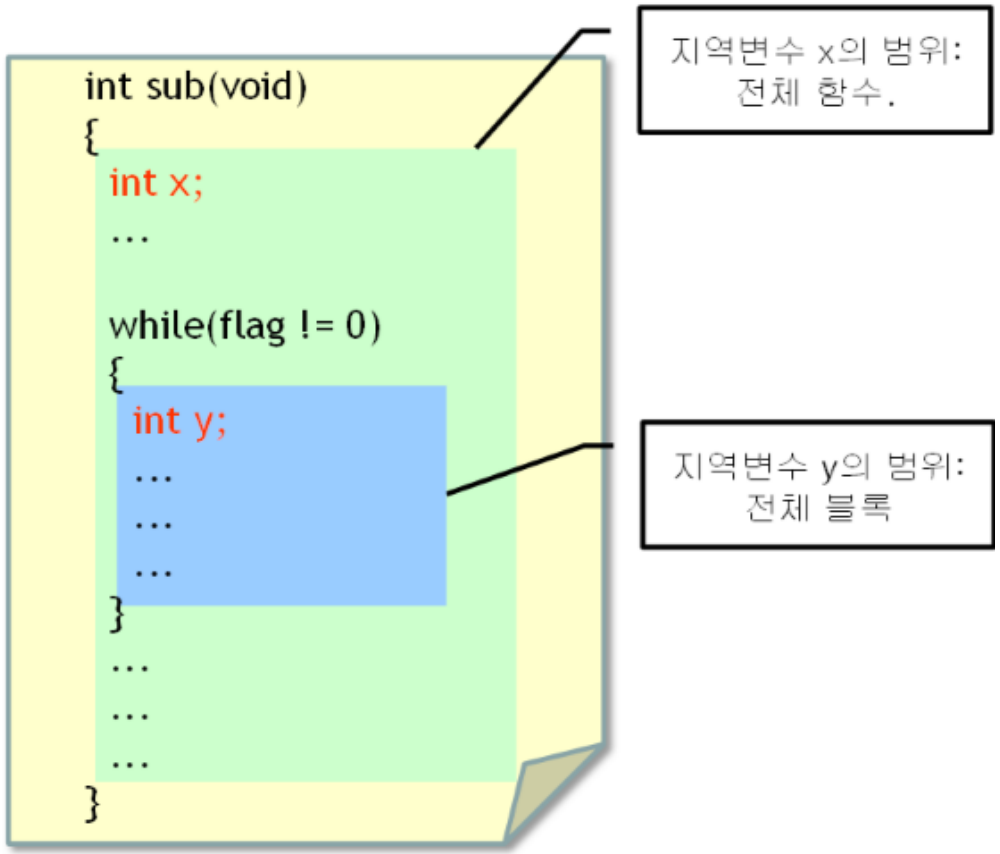


이나 함수 안에서만

에서도 사용이 가능

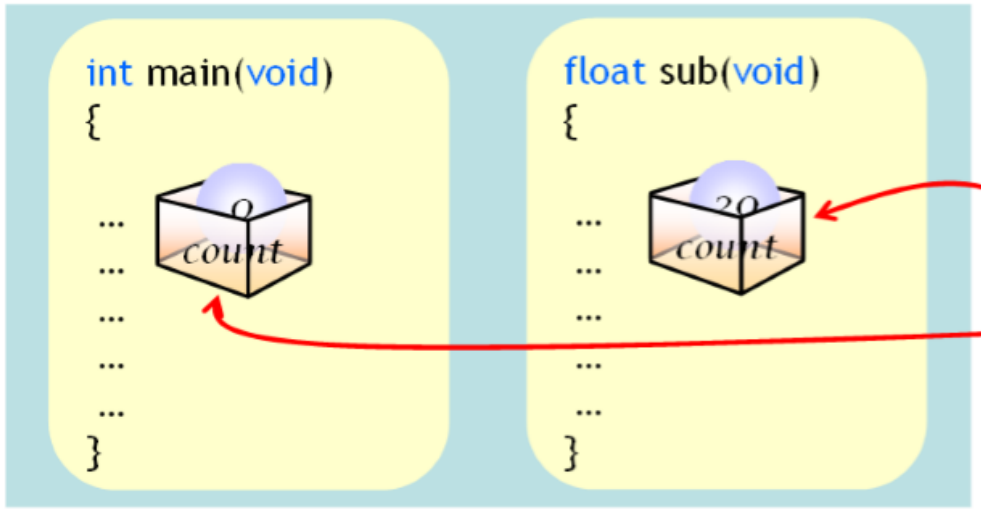
지역변수

지역변수 : 블록 안에 선언되는 변수.



지역변수

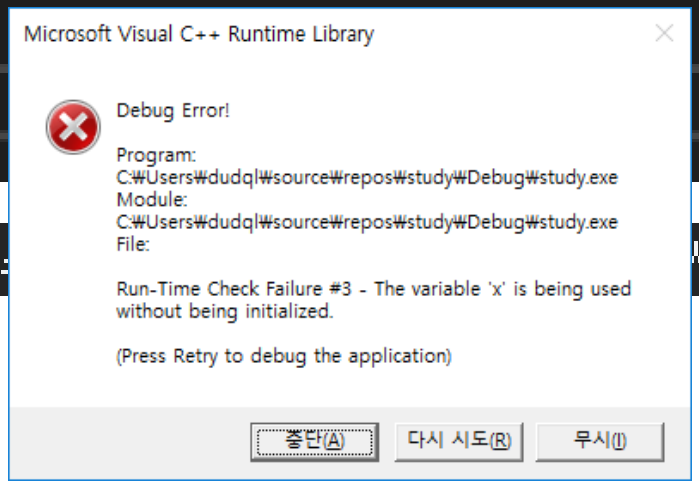
이름이 같은 지역변수 : 블록만 다르면 이름은 같아도 됨.



상관없음!!

지역변수의 초기값 : 초기화를 해주지 않는다면 아무 의미 없는 값, 쓰레기 값이 들어가 있음.

```
int main() {
    int x;
}
```




```
warning C4700: 
```

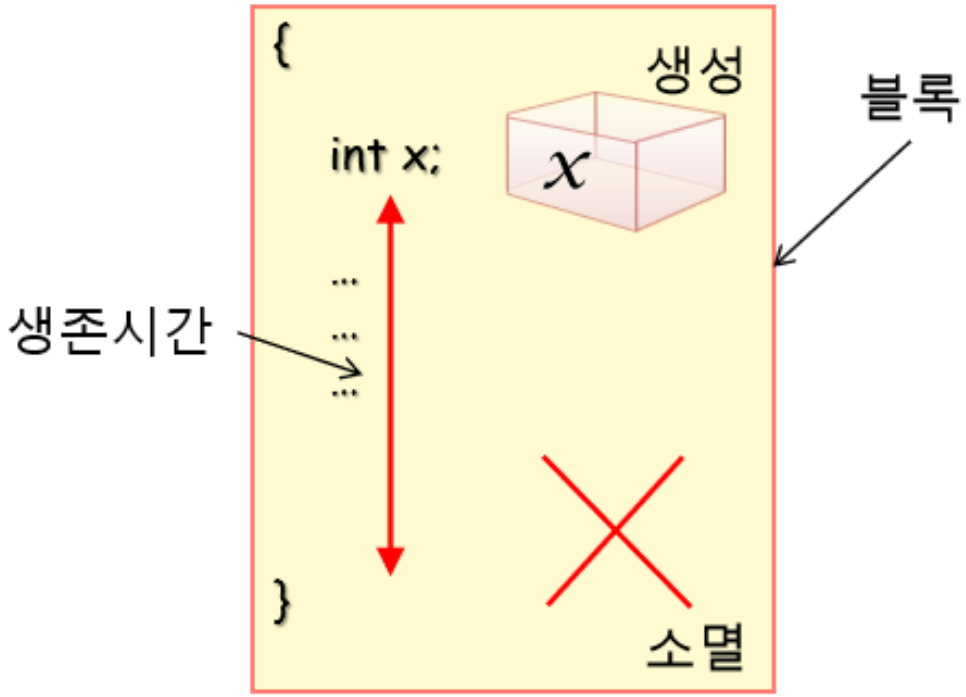
사용했습니다.

지역변수

지역변수의 생존시간 : 지역변수는 선언된 블록이 시작할 때 스택이라 불리는 메모리 공간에 만들어진다. 지역변수에 할당된 공간은 블록 끝에서 반환된다.

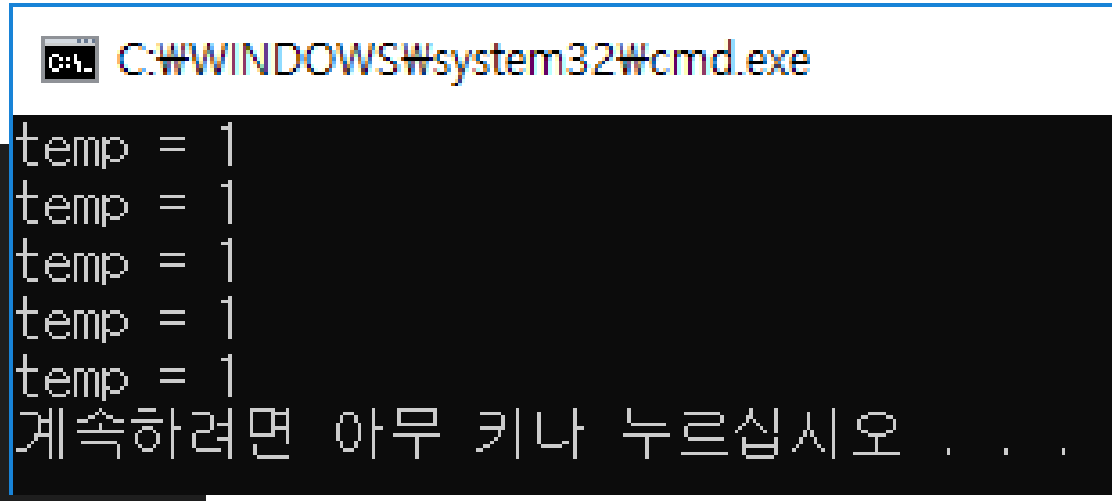
지역변수의 범위 : 선언된 블록 내에서만 접근 및 사용 가능.

```
void sub1()
{
    {
        int y;
    }
    y = 4;     //y가 선언된 블록을 벗어났으므로 오류!
}
```



지역변수

```c
int main()
{
    int i;

    for (i = 0; i < 5; i++)
    {
        int temp = 1;
        printf("temp = %d\n", temp);
        temp++;
    }
    return 0;
}
```



지역변수

```c
#include<stdio.h>

int inc(int counter)//매개변수도 지역변수!
{
    counter++;
    return counter;
}

int main()
{
    int i = 10;

    printf("함수 호출전 i=%d\n", i);
    inc(i);
    printf("함수 호출후 i=%d\n", i);

    return 0;
}
```

counter와 i는 완전히 다른 별도의 변수.

함수 호출 시에는 변수 i의 값이 counter로 복사돼서 전달됨.

이것을 값에 의한 호출(call by value)라고 하며 인수 값이 매개변수로 복사돼서 전달된다는 의미이다.

C언어에는 이것 말고도 참조에 의한 호출(call by reference)도 있다. 이는 나중에 다룬다.

전역변수

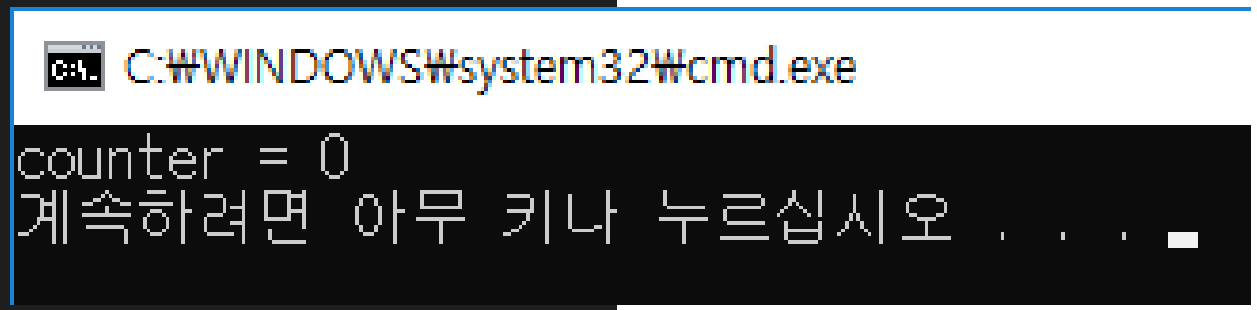
전역 변수 : 함수 외부에 선언되는 변수

```c
#include<stdio.h>

int x = 123;

void sub1()
{
    printf("In sub1()  x = %d\n", x);
}
void sub2()
{
    printf("In sub2()  x = %d\n", x);
}

int main()
{
    sub1();
    sub2();
    return 0;
}
```

전역 변수는 함수 어디든 접근이 가능하고 사용도 가능하다.

전역변수

```
1   #include<stdio.h>
2
3   int counter;
4
5   int main()
6   {
7       printf("counter = %d\n", counter);
8       return 0;
9   }
```



전역변수를 초기화하지 않으면 컴파일러에 의하여 0으로 초기화된다.
생존시간은 프로그램 시작과 동시에 생성되고, 종료되기 전까지 메모리에 존재한다.

전역변수

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

int i;
int main(void)
{
        for(i = 0;i < 5; i++)
        {
                f();
        }
        return 0;
}
void f(void)
{
        for(i = 0;i < 10; i++)
                printf("#");
}
```

전역 변수는 상당히 편리할 것처럼 보이지만 전문가들은 사용을 권하지 않는다. 어디서나 접근이 가능하다는 점이 단점이 될 수 있기 때문이다.

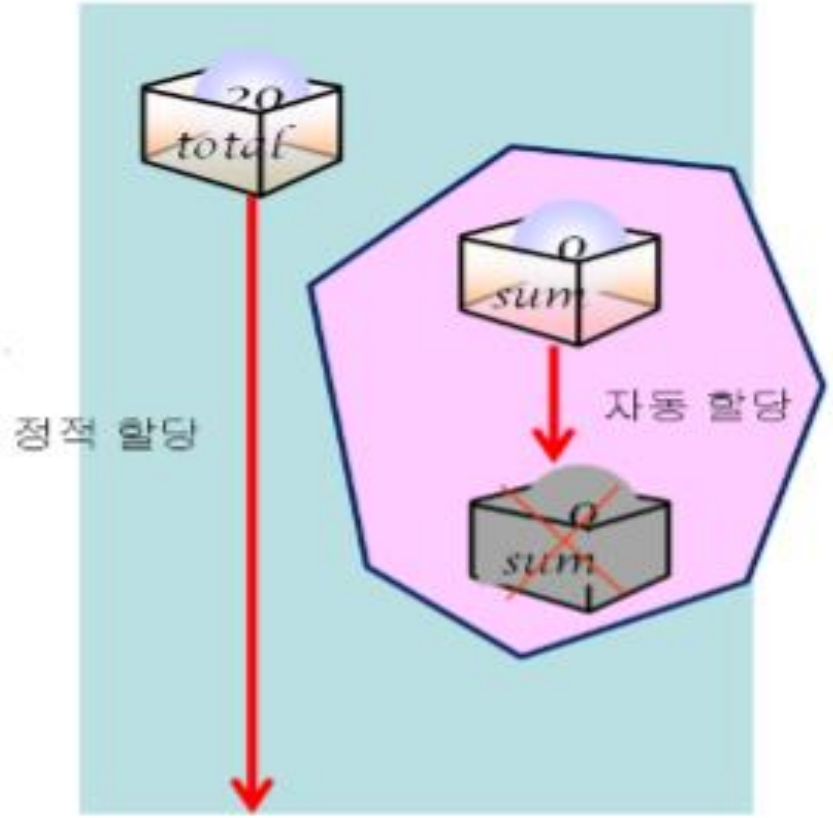
프로그램이 복잡해지면 전역변수를 어디서 변경했는지 알 수 없으며, 하나의 전역변수를 변경하면 다른 함수에서도 변경하여야 하는 경우가 허다하기 때문이다.

이처럼 복잡한 코드를 스파게티 코드라고 하며 이러한 코드는 되도록 짜지 않도록 해야 한다.

다시 말해, 꼭 써야하는 상황이 아니면 쓰지 맙시다.

생존 시간

- **정적할당(static allocation)** : 프로그램이 실행되는 동안 계속 변수에 저장 공간이 할당되어 있는 방법.
- **자동할당(automatic allocation)** : 블록이 시작되면서 변수에 저장 공간이 할당되고 블록이 종료되면 저장 공간이 회수되는 방법.



생존 시간을 결정하는 요소

1. **변수가 선언되는 위치** : 전역변수는 정적할당, 지역변수는 자동할당 된다.

2. **저장 유형 지정자** : 변수를 선언할 때 지정자를 붙여 생존 시간을 결정할 수 있다. 수식어들은 auto, static, register, extern 등이 있다.지정자를 쓰지 않는다면 선언위치에 따라 자동으로 결정된다.

생존 시간

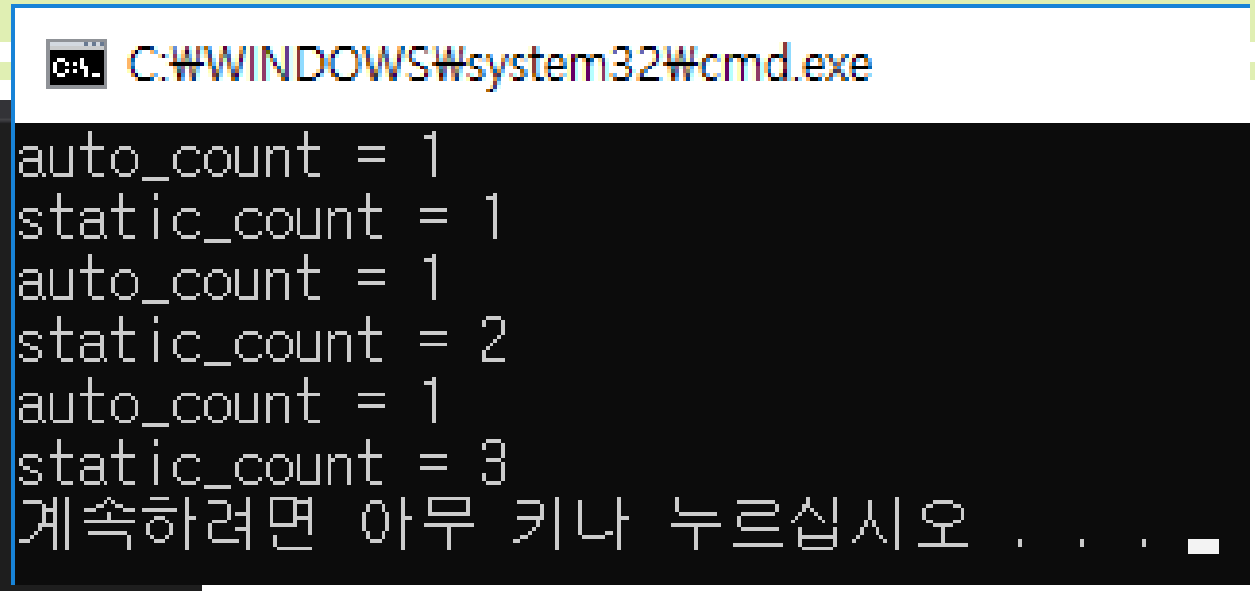
- **auto 지정자** : 블록 내에 선언되는 지역 변수는 기본적으로 **자동 할당**이 된다. 이러한 지역변수를 자동 변수라고 한다. 원칙적으로 auto 키워드를 붙여야 하지만 블록내에서는 auto가 생략되어도 자동 변수로 취급된다.

  int sum; == auto int sum;

- **static 지정자** : 지역 변수처럼 블록 내에서만 사용되지만 블록을 벗어나도 자동으로 제거되지 않는 변수를 선언하려면 **static** 키워드를 붙이면 된다.

생존 시간

```c
#include<stdio.h>

void sub() {
    int auto_count = 0;
    static int static_count = 0;

    auto_count++;
    static_count++;

    printf("auto_count = %d\n", auto_count);
    printf("static_count = %d\n", static_count);
}

int main()
{
    for (int i = 0; i < 3; i++) {
        sub();
    }
    return 0;
}
```



순환 – 팩토리얼

순환 : 자기 자신을 호출하는 함수.

```c
#include<stdio.h>

long factorial(int n)
{
    printf("factorial(%d)\n", n);
    if (n <= 1)
        return 1;
    else
        return n * factorial(n - 1);
}
int main()
{
    int n;
    printf("정수를 입력하세요 : ");
    scanf("%d", &n);
    printf("%d의 팩토리얼 : %d\n", n, factorial(n));
    return 0;
}
```

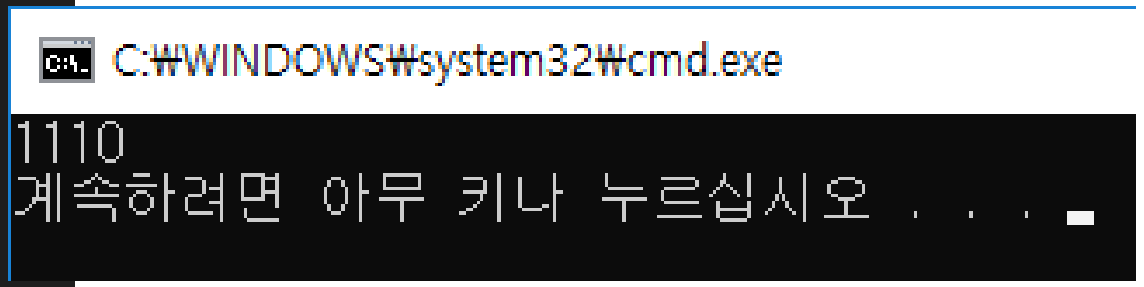
팩토리얼의 정의

$$n! = \begin{cases} 1 & n = 1 \\ n*(n-1)! & n \geq 2 \end{cases}$$

순환 - 2진수 형식으로 출력하기

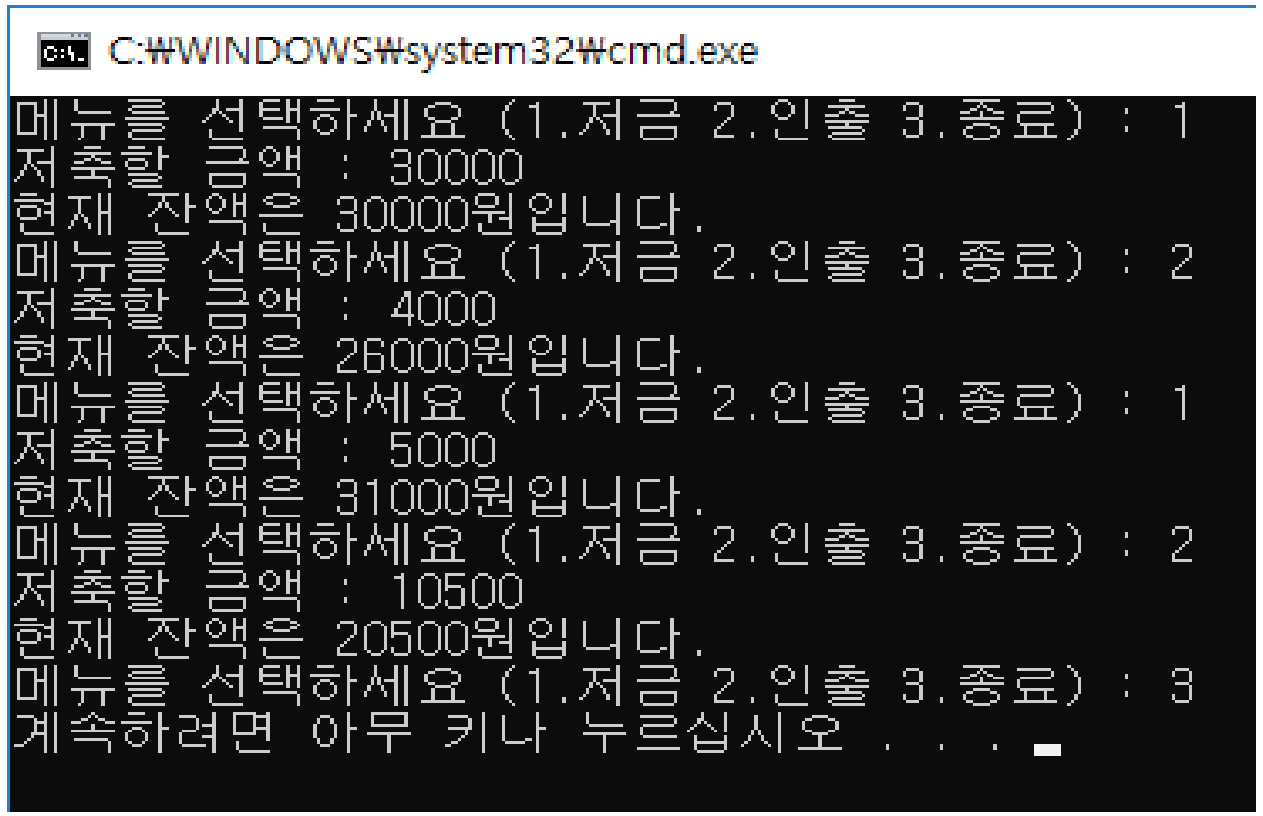
알고리즘 : 2로 나누어서 몫이 0이 될 때까지 나머지를 기록했다가 이를 역순으로 출력.

```c
#include<stdio.h>

void print_binary(int x)
{
    if (x > 0)
    {
        print_binary(x / 2);
        printf("%d", x % 2);
    }
}
int main()
{
    print_binary(14);
    printf("\n");
    return 0;
}
```



실습 문제

- 입출금 프로그램을 작성해보자. 입출금 함수인 save(int mount), draw(int amount)를 작성한 후 예금 잔액은 전역변수를 이용하라.
- hint) 무한반복문, break를 활용한다.



심화

1,2,3 이외의 응답이 들어왔을 경우와 잔고가 마이너스 일경우를 처리하는 경로도 작성해보자.

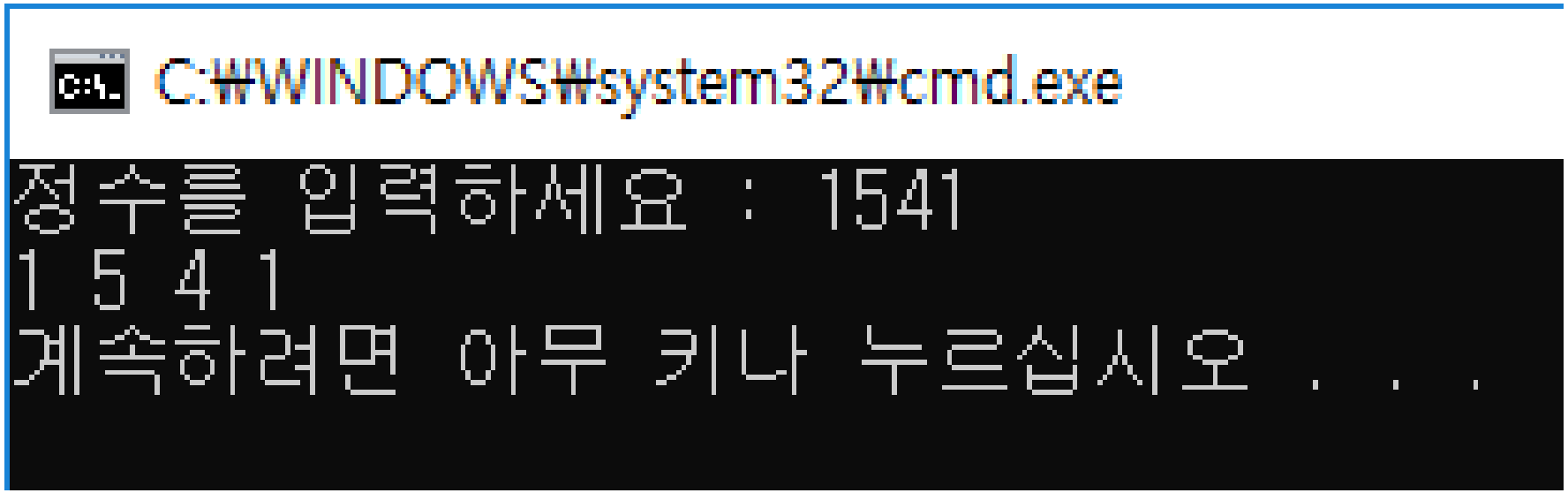
실습 문제

```c
#include<stdio.h>
void save(int amount);
void draw(int amount);

int money = 0;

int main(void)
{
    int menuSelector;
    int amount;

    while (1)
    {
        printf("메뉴를 선택하세요 (1.저금 2.인출 3.종료) : ");
        scanf("%d", &menuSelector);

        switch (menuSelector)
        {
        case 1:
            printf("저축할 금액 : ");
            scanf("%d", &amount);
            save(amount);
            break;
        case 2:
            if (money < 0)
            {
                printf("잔고가 마이너스입니다.\n");
            }
            else
            {
                printf("인출할 금액 : ");
                scanf("%d", &amount);
                draw(amount);
            }
            break;
        case 3:
            return 0;
        default:
            printf("잘못된 입력입니다.\n");
            break;
        }

        printf("현재 잔액은 %d원입니다.\n", money);
    }

    return 0;
```

실습 문제

- 순환 호출을 이용하여 각 자리수를 출력하는 함수 show_digit(int x)를 작성해보자.

  Hint) 1234에서 4는 10으로 나눈 나머지이며 123은 10으로 나눈 몫이다.



실습 문제

```c
#include<stdio.h>
void show_digit(int x);

int main(void)
{
    int num;
    printf("정수를 입력하세요 : ");
    scanf("%d", &num);

    show_digit(num);
    printf("\n");

    return 0;
}

void show_digit(int x)
{
    if (x)  //x가 0이 아니면
    {
        show_digit(x / 10);
        printf("%d ", x % 10);
    }
}
```

다음주

배열

감사합니다