

# C – 함수(function)

*function*



# 함수(function)

## ❖ 함수(Function)란?

- 하나의 기능을 수행하는 일련의 코드이다.(모듈화)
- 함수는 이름이 있고, 반환값과 매개변수가 있다.(함수의 형태)
- 하나의 큰 프로그램을 작은 부분들로 분리하여 코드의 중복을 최소화하고, 코드의 수정이나 유지보수를 쉽게 한다.(함수를 사용하는 이유)
  - 모든 코드를 `main(){...}` 함수 내에서 만들면 중복 및 수정의 복잡함이 있음

## ❖ 함수의 종류

- 내장 함수 – 수학, 시간, 문자열 함수 등
- 사용자 정의 함수 – 사용자(개발자)가 직접 만들어 사용하는 함수

```
반환자료형 함수이름(매개변수)
{
    구현 코드
}
```

```
int getArea(x, y)
{
    return x * y
}
```



# 함수(function)

## ❖ 사용자 정의 함수

- 사용자(개발자)가 직접 만들어 사용하는 함수

```
반환자료형 함수이름(매개변수)
{
    구현 코드
}
```

```
int getArea(x, y)
{
    return x * y
}
```



# 사용자 정의 함수(function)

## ❖ 함수의 정의와 호출

### 1. return값이 없는 경우(void 형)

```
#include <stdio.h>

void sayHello();
void sayHello2(char[]);

int main(void)
{
    sayHello();

    sayHello2("안중근");
    sayHello2("Elsa");
    return 0;
}
```

프로토타입(시그니처)

함수 호출

# 함수(function)의 유형

## 1. return값이 없는 함수(void 형)

```
void sayHello()  
{  
    printf("안녕하세요\n");  
}  
  
void sayHello2(char name[])  
{  
    printf("%s님~ 안녕하세요\n", name);  
}
```

함수 정의



# 함수(function)의 유형

## 2. return값이 있는 함수 – 매개변수가 1개 있는 경우

```
int main(void)
{
    int result = square(4);

    printf("제공한 값: %d\n", result);

    return 0;
}

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



# 함수(function)의 유형

## 2. return값이 있는 함수 – 매개변수가 1개 있는 경우

```
int MyAbs(int n)
{
    if (n < 0)
        return -n;
    else
        return n;

    return n;
}

int main(void)
{
    int value1 = MyAbs(-4);
    int value2 = abs(-4); //abs() - 내장 함수

    printf("절대값: %d\n", value1);
    printf("절대값: %d\n", value2);

    return 0;
}
```



# 함수(function)의 유형

## 2. return값이 있는 함수 – 매개변수가 2개 있는 경우

```
#include <stdio.h>
int add(int x, int y)
{
    return x + y;
}

int main()
{
    int result;
    result = add(10, 20);
    printf("두 수의 합 : %d\n", result);

    return 0;
}
```

함수 정의

함수 호출





# 함수(function)의 유형

- 1부터 n까지의 합과 곱을 계산하는 함수

```
int calcSum(int n)
{
    int sum = 0;
    for (int i = 1; i <= n; i++)
    {
        sum += i; //sum = sum + i
    }
    return sum;
}

int calcGob(int n)
{
    int gob = 1;
    for (int i = 1; i <= n; i++)
    {
        gob *= i; //gob = gob * i
    }
    return gob;
}
```



# 함수(function)의 유형

- 1부터 n까지의 합과 곱을 계산하는 함수

```
int main()
{
    int value1, value2;

    //1부터 10까지의 합
    value1 = calcSum(10);

    //1부터 5까지의 곱
    value2 = calcGob(5); //5! = 5x4x3x2x1

    printf("합계: %d\n", value1);
    printf("곱: %d\n", value2);

    return 0;
}
```



# 함수(function) 예제

- 배열에서 최대값 구하기

```
int findMax(int arr[], int len);
int main(void)
{
    int arr[] = { 21, 35, 71, 2, 97, 66 };
    int max = findMax(arr, 6);

    printf("최대값: %d\n", max);

    return 0;
}
```

```
int findMax(int arr[], int len)
{
    int maxVal = arr[0];

    for (int i = 1; i < len; i++)
    {
        if (arr[i] > maxVal)
            maxVal = arr[i];
    }

    return maxVal;
}
```



# 실습 문제 1 – 함수

정사각형과 삼각형의 넓이를 계산하는 함수를 각각 정의하고 아래와 같이 출력하세요.

👉 실행 결과

```
정사각형의 넓이 : 16cm  
삼각형의 넓이 : 7.5cm
```

□ 정사각형

- 한 변의 길이 : 4cm
- 함수명 : square()

▷ 삼각형

- 밑변 : 3cm, 높이 : 5cm
- 함수명: triangle()



# 변수의 메모리 영역

- **코드 영역** : 프로그램의 **실행 코드** 또는 **함수**들이 저장되는 영역
- **스택 영역** : **매개 변수 및 중괄호(블록)** 내부에 **정의된 변수**들이 저장되는 영역
- **데이터 영역** : **전역 변수**와 **정적 변수**들이 저장되는 영역
- **힙 영역** : **동적으로 메모리 할당하는 변수**들이 저장되는 영역



코드 영역  
(실행 코드, 함수)



스택 영역  
(지역 변수, 매개 변수)



데이터 영역  
(전역 변수, 정적 변수)



힙 영역  
(동적 메모리 할당)



# 변수의 적용 범위 - 지역변수

## ➤ 지역 변수(local variable)

- 하나의 코드 블록에서만 정의되어 사용되는 변수
- 함수 또는 제어문의 중괄호{ } 내부에서 사용

지역 변수의 메모리 생성 시점 - 블록(중괄호) 내에서 초기화할 때

지역 변수의 메모리 소멸 시점: - 블록(중괄호)을 벗어났을 때

```
int add10();

int main(void)
{
    int value = add10();
    printf("value = %d\n", add10());
    //printf("x = %d\n", x); //x는 정의되지 않음

    return 0;
}
```

```
int add10()
{
    int x = 1;
    x += 10;

    return x;
}
```



# 변수의 적용 범위 – 전역 변수

- 전역 변수(global variable)
  - 전체 소스 코드를 범위로 적용되는 변수
  - 소스 파일 내의 어디서든지 사용 가능한 변수

전역 변수의 메모리 생성 시점 - 프로그램이 시작되었을 때

전역 변수의 메모리 소멸 시점: - 프로그램이 종료되었을 때

```
int x = 1; //전역 변수
int add10();

int main(void)
{
    //printf("x = %d\n", x);
    int value = add10();
    printf("value = %d\n", add10());
    printf("x = %d\n", x);

    return 0;
}
```

```
int add10()
{
    x = x + 10;

    return x;
}
```



# 변수의 적용 범위 – 정적 변수

## ➤ 정적 변수(static variable)

- 선언된 함수가 종료하더라도 그 값을 계속 유지하는 변수
- **static** 키워드를 붙임

전역 변수의 메모리 생성 시점 - 중괄호 내에서 초기화될때

전역 변수의 메모리 소멸 시점: - 프로그램이 종료되었을 때

```
void call();

int main(void)
{
    call();
    call();
    call();

    return 0;
}
```

```
void call()
{
    //int x = 0; //지역변수
    static int x = 0; //정적 변수-전역변수화 함

    x += 1;
    printf("현재 호출은 %d번째입니다.\n", x);
}
```

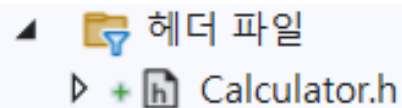




# 헤더 파일 사용하기

## ❖ 헤더파일 사용하기

- 다른 소스 파일에서 함수 또는 변수를 사용하는 방법이다.
- 헤더파일에서는 함수의 프로토타입을 선언한다.
- 헤더파일 > 추가 > 새항목 > Calculator.h



## ❖ Calculator 프로젝트 만들기

- Calculator.h – 헤더 파일(전역변수, 함수 선언부)
- Calculator.c – 함수 구현부
- Main.c – 실행 파일



# 헤더 파일 사용하기

## ❖ 헤더파일 사용하기

<Calculator.h>

```
//Calculator.h - 헤더 파일(함수 선언부 표기)  
  
int calcSum(int n); //덧셈 계산  
  
int calcGob(int n); //곱셈 계산
```



# 헤더 파일 사용하기

## ❖ 헤더파일 사용하기

<Calculator.c>

```
int calcSum(int n)
{
    int sum = 0;
    for (int i = 1; i <= n; i++){
        sum += i; //sum = sum + i
    }
    return sum;
}

int calcGob(int n)
{
    int gob = 1;
    for (int i = 1; i <= n; i++){
        gob *= i; //gob = gob * i
    }
    return gob;
}
```



# 헤더 파일 사용하기

## ❖ Main 함수

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
//헤더파일 포함 - 쌍따옴표 사용
#include "Calculator.h"

int main()
{
    int num1, num2;

    printf("==== Simple Calculator ==== \n\n");
    printf("1부터 몇 까지 더할까요? ");
    scanf("%d", &num1);
    printf("1부터 %d까지 더한 값은 %d입니다.\n", num1, calcSum(num1));

    printf("----- \n");
```



# 헤더 파일 사용하기

## ❖ Main 함수

```
printf("-----\n");

printf("1부터 몇 까지 곱할까요? ");
scanf("%d", &num2);
printf("1부터 %d까지 곱한 값은 %d입니다.\n", num2, calcGob(num2));

system("pause"); //exe 파일 실행시 프로세스 유지

return 0;
}
```

```
===== Simple Calculator =====
```

```
1부터 몇 까지 더할까요? 10
1부터 10까지 더한 값은 55입니다.
```

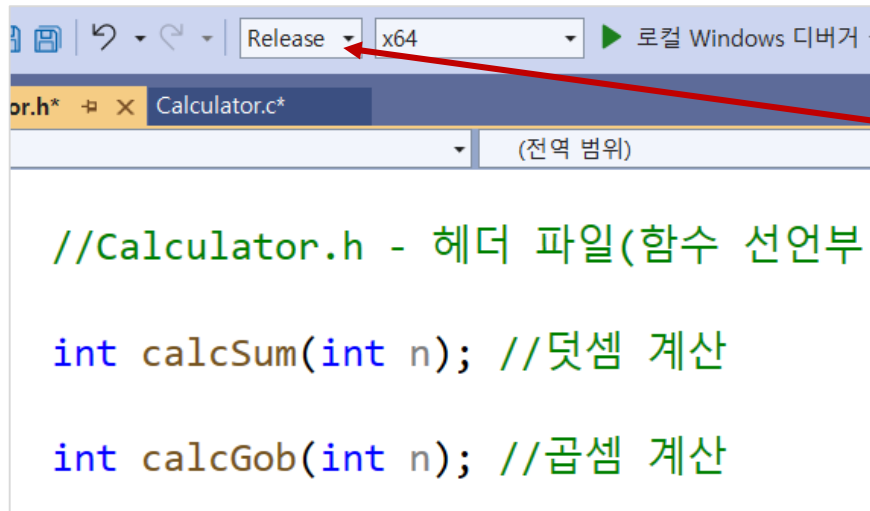
```
-----
1부터 몇 까지 곱할까요? 5
1부터 5까지 곱한 값은 120입니다.
계속하려면 아무 키나 누르십시오 . . . |
```



# 파일 배포

## ◆ 파일 배포

파일 배포란 c언어 소스파일을 .exe 실행 파일로 만들어 공개 및 서비스 하는 것을 말한다.



The screenshot shows a C++ IDE with the following details:

- Build configuration: Release (selected), x64 (selected), 로컬 Windows 디버거 (Local Windows Debugger).
- File: Calculator.c\*
- Scope: (전역 범위) (Global Scope)
- Code content:

```
//Calculator.h - 헤더 파일(함수 선언부  
  
int calcSum(int n); //덧셈 계산  
  
int calcGob(int n); //곱셈 계산
```

Debug 모드를  
Release 모드로 바꾼다.



Ctrl + F5로 실행



# 파일 배포

- ◆ exe 파일에서 창이 꺼지는 문제 해결

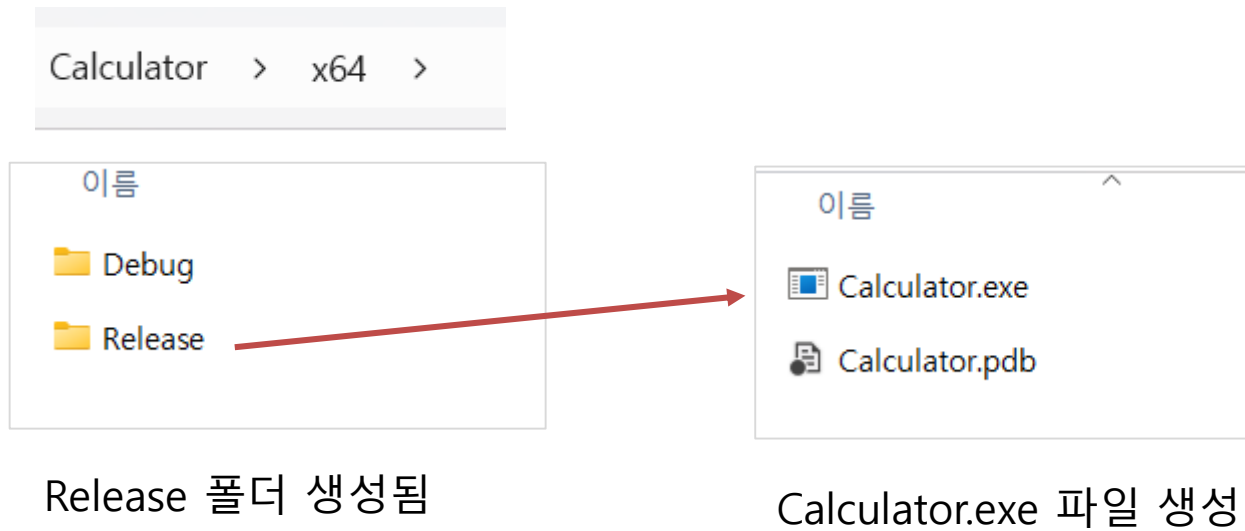
**system("pause")** 를 명시함

```
printf("1부터 몇 까지 곱할까요? ");  
scanf("%d", &num2);  
printf("1부터 %d까지 곱한 값은 %d입니다.\n", num2, calcGob(num2));  
  
system("pause"); //exe 파일 실행시 프로세스 유지  
  
return 0;  
}
```



# 파일 배포

## ◆ 파일 배포

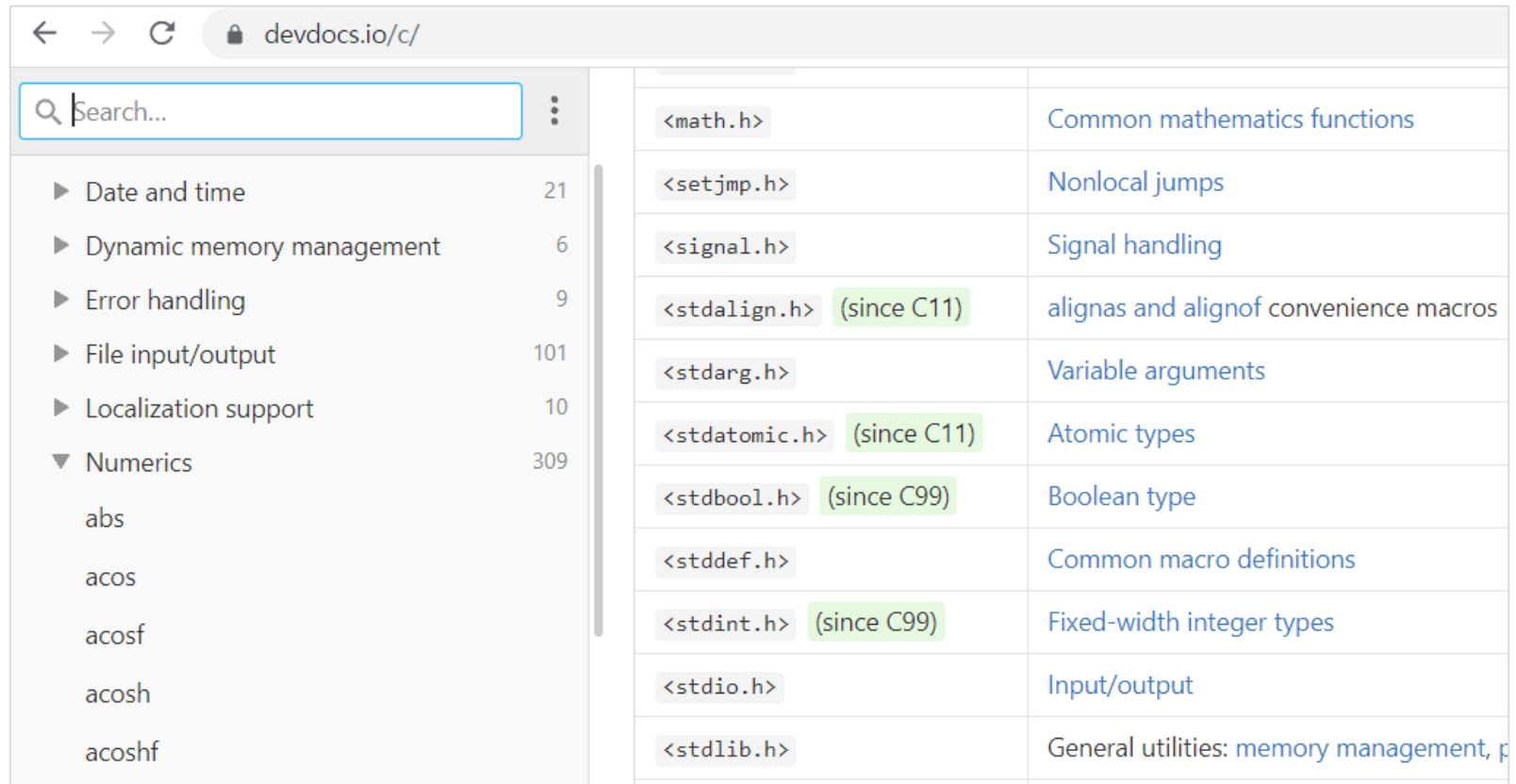




# 표준 라이브러리 함수(function)

## ❖ 내장 함수 – 표준 라이브러리 함수

C언어 Devdocs 검색 : <https://devdocs.io/c>



The screenshot shows the devdocs.io/c website. On the left, there is a search bar and a sidebar with a list of categories and their counts: Date and time (21), Dynamic memory management (6), Error handling (9), File input/output (101), Localization support (10), and Numerics (309). Under Numerics, several functions are listed: abs, acos, acosf, acosh, and acoshf. On the right, there is a table of C standard library headers and their descriptions:

<math.h>	Common mathematics functions
<setjmp.h>	Nonlocal jumps
<signal.h>	Signal handling
<stdalign.h> (since C11)	alignas and alignof convenience macros
<stdarg.h>	Variable arguments
<stdatomic.h> (since C11)	Atomic types
<stdbool.h> (since C99)	Boolean type
<stddef.h>	Common macro definitions
<stdint.h> (since C99)	Fixed-width integer types
<stdio.h>	Input/output
<stdlib.h>	General utilities: memory management, p

# 수학 함수(function)

- ✓ 수학 관련 함수 – math.h를 include 해야 함

```
#include <stdio.h>
#include <math.h> //수학 관련 함수 라이브러리

int main()
{
    //함수의 반환 자료형: double - %f, int - %d
    //반올림
    printf("%.1f\n", round(2.54)); //3.0
    printf("%.f\n", round(2.54)); //3 (정수 표현)

    //내림(버림)
    printf("%.1f\n", floor(2.54)); //2.0
    printf("%.f\n", floor(2.14)); //2

    //올림
    printf("%.1f\n", ceil(2.54)); //3.0
    printf("%.f\n", ceil(2.14)); //3
}
```



# 수학 함수(function)

- ✓ 수학 관련 함수 – math.h를 include 해야 함

```
//절대값
printf("%d\n", abs(-8)); //8
printf("%d\n", abs(8)); //8

//거듭제곱
printf("%.f\n", pow(2, 4)); //16
printf("%.f\n", pow(10, 3)); //1000

//제곱근
printf("%.f\n", sqrt(16)); //4
printf("%.f\n", sqrt(100)); //10

return 0;
}
```



# 수학 함수(function)

- ✓ 수학 관련 상수 - `_USE_MATH_DEFINES`를 정의해야 함

```
#define _USE_MATH_DEFINES //M_PI 상수 사용
#include <stdio.h>
#include <math.h>

int main()
{
    //원주율 상수 - M_PI
    printf("%f\n", M_PI);
    printf("%.2f\n", M_PI);

    int ans = pow(2, ceil(M_PI)); //올림
    printf("%d\n", ans);
}
```



# 수학 함수(function)

- ✓ 수학 관련 상수 - `_USE_MATH_DEFINES`를 정의해야 함

```
//원의 넓이 - 원주율 x 반지름 x 반지름
int radius = 4;
double area;

area = M_PI * radius * radius;

printf("원의 넓이: %.2f\n", area);

return 0;
}
```

```
3.141593
3.14
16
원의 넓이 : 50.27
```



# 수학 함수(function)

- ✓ 거듭 제곱 함수 정의하고 라이브러리 함수와 비교

```
int myPow(int x, int y) //x:밑, y:지수
{
    int num = 1; //거듭제곱 결과값
    for (int i = 0; i < y; i++)
    {
        num *= x; //num = num * x;
    }
    return num;

    /*
        x = 2, y = 3인 경우
        (2 x 2 x 2)
            num = num * x
        i=0,    2 = 1 * 2
        i=1,    4 = 2 * 2
        i=2,    8 = 4 * 2
    */
}
```



# 수학 함수(function)

- ✓ 거듭 제곱 함수 정의하고 라이브러리 함수와 비교

```
int main()
{
    int val1, val2;

    val1 = myPow(2, 3); //MyPow() 호출

    val2 = pow(2, 3); //라이브러리 함수 호출

    printf("%d, %d\n", val1, val2);

    return 0;
}
```



# 시간 함수(function)

- ✓ 시간 관련 함수 – time.h를 include 함

```
#include <stdio.h>
#include <time.h>
#include <Windows.h>

int main()
{
    //time_t 자료형
    //time_t now = time(NULL);
    long now = time(NULL);

    //초로 환산 : ld - long decimal
    printf("1970년 1월 1일(0시 0분 0초) 이후 : %ld초\n", now);
    //일로 환산
    printf("1970년 1월 1일(0시 0분 0초) 이후: %ld일\n", now / (24 * 60 * 60));
    //년으로 환산
    printf("1970년 1월 1일(0시 0분 0초) 이후: %ld년\n", now / (365 * 24 * 60 * 60));
```

1745885349초  
20207일  
55년





# 시간 함수(function)

## ✓ 수행 시간 측정하기

```
time_t start, end;    //time_t 자료형
start = time(NULL);   //시작 시각
printf("시작 시각: %ld초\n", start);

//0.5초 간격으로 1 ~ 10 출력
for (int i = 1; i <= 10; i++)
{
    printf("%d\n", i);
    Sleep(500);    //<Windows.h> 포함
}

end = time(NULL);    //종료 시각
printf("종료 시각: %ld초\n", end);
printf("%ld초\n", (end - start));
```

```
시작 시각 : 1745885313초
1
2
3
4
5
6
7
8
9
10
종료 시각 : 1745885318초
5초
```



# 시간 함수(function)

- ✓ 수행 시간 측정하기 - 소수로 출력

```
//수행 시간(정밀 측정)
clock_t start, end;
double elapsedTime;

start = clock(); //시작 시각

for (int i = 1; i <= 10; i++)
{
    printf("%d\n", i);
    Sleep(500);
}

end = clock(); //종료 시각

//CLOCKS_PER_SEC - 초당 시각 상수
elapsedTime = (double)(end - start) / CLOCKS_PER_SEC;
printf("소요시간: %.21f초\n", elapsedTime);
```

```
1
2
3
4
5
6
7
8
9
10
소요시간: 5.06초
```



# rand() 함수

- rand() 함수 - 난수(무작위)를 생성해 주는 함수

$\text{rand()} \% (\text{경우의 수}) + 1$

- rand() 함수를 사용하려면 srand() 함수가 반드시 먼저 사용되어야 한다.
- seed값을 설정하면 한번 만 난수로 되므로, 계속 무작위수가 나오려면 seed값에 시간의 흐름을 넣어준다.

**srand(6) -> srand(time(NULL))**

- srand(), rand()는 <stdlib.h>에 정의 되어 있다.
- 동전의 양면, 가위/바위/보, 주사위 눈의 수등 게임이나 통계 확률 등에서 많이 사용된다.



# rand() 함수

- rand() 함수 예제

```
#include <stdio.h>
#include <stdlib.h> //srand(), rand()
#include <time.h> //time()

int main()
{
    // srand(10); //seed값 설정(고정)
    srand(time(NULL)); //seed값 설정(변경)

    int rndVal = rand();
    printf("%d\n", rndVal);
    printf("=====\n");

    //동전(2가지 경우)
    int coin = rand() % 2;
    printf("%d\n", coin);
```

```
// 0-앞면, 1-뒷면
if (coin % 2 == 0)
{
    printf("앞면\n");
}
else
{
    printf("뒷면\n");
}
printf("=====\n");

//주사위(1~6)
int dice = rand() % 6 + 1;
printf("주사위 눈: %d\n", dice);
```



# rand() 함수

- rand() 함수 - 난수(무작위)를 생성해 주는 함수

```
//실습 - 주사위 10번 던지기
for (int i = 0; i < 10; i++)
{
    dice = rand() % 6 + 1;
    printf("%d\n", dice);
}

//가위 바위 보
int n = rand() % 3;

switch (n)
{
case 0: printf("가위\n"); break;
case 1: printf("바위\n"); break;
case 2: printf("보\n"); break;
default: printf("없음\n"); break;
}
```



# 문자열 처리 함수

## ● 문자열 처리 함수

함수의 원형	헤더파일	기능 설명
<b>getchar(void)</b>	<stdio.h>	문자 1개 입력
<b>while(getchar() != '\n')</b>		버퍼에서 ('\n')을 비움
<b>fgets</b> (char* Bufffer, int MaxCount, File* stream)	<stdio.h>	공백을 포함한 문자열 입력 가능
<b>puts</b> (char* Buffer)	<stdio.h>	문자열 출력[ printf()와 유사함 ]
<b>strcpy</b> (char *string1, const char *string2)	<string.h>	string2 문자열을 string1로 복사
<b>strlen</b> (const char* Str)	<string.h>	저장된 문자열의 길이를 반환(개수)
<b>strcmp</b> (const char* Str1, const char* Str2)	<string.h>	두 문자열의 비교 결과 반환 같으면 0, 다르면 1



# 문자열 처리 함수

- 문자 1개 입력 및 버퍼 비우기 – getchar()

**while(getchar() != '\n')** 문은

버퍼(임시기억장소)에 남아 있던 데이터를 '\n' 전까지 삭제

```
char c1, c2;  
//'\n'은 아스키 코드 - 10(LF-Line Feed)  
  
c1 = getchar();  
  
//이 구문이 없으면 엔터를 쳤을때 자동으로 '\n'이 실행됨  
while (getchar() != '\n');  
  
c2 = getchar();  
  
printf("%d %d\n", c1, c2);
```

a  
97 10

a  
b  
97 98



# 문자열 처리 함수

- 정수 입력시 문자 입력 오류 처리

```
int get_valid_integer() {  
    int num;  
  
    while (1) {  
        printf("정수를 입력하세요: ");  
        int input = scanf("%d", &num);  
        if (input) { //input == 1  
            return num;  
        }  
        else {  
            printf("유효한 정수를 입력하세요.\n");  
            while (getchar() != '\n'); // 입력 버퍼 비우기  
        }  
    }  
}
```





# 문자열 처리 함수

- 정수 입력시 문자 입력 오류 처리

```
int main() {  
    int number = get_valid_integer();  
    printf("입력한 숫자: %d\n", number);  
  
    return 0;  
}
```

```
정수를 입력하세요: haha  
유효한 정수를 입력하세요.  
정수를 입력하세요: 200  
입력한 숫자: 200
```



# 문자열 처리 함수

- 문자열 복사, 개수 – strcpy(), strlen()

```
#define _CRT_SECURE_NO_WARNINGS //strcpy() 처리
#include <stdio.h>
#include <string.h>

int main()
{
    char msg1[] = "Good Luck!";
    char msg2[20];
    int len;

    //문자열의 개수
    len = strlen(msg1); //10, 공백문자 포함
    printf("%d\n", len);

    //문자열의 복사
    //strcpy(msg2, msg1); //strcpy(복사할 장소, 복사할 내용)
    strcpy(msg2, msg1);
    printf("%s\n", msg2);
}
```



# 문자열 처리 함수

- 문자열 비교 – strcmp()

```
//문자열의 비교
char greet1[] = "hello";
char greet2[] = "Hello";
int result;

//0 - 일치, 1 - 불일치
//대소문자 구분함
result = strcmp(greet1, greet2);
printf("%d\n", result);

if (result == 0)
{
    puts("문자열이 일치합니다.");
}
else
{
    puts("문자열이 일치하지 않습니다.");
}

return 0;
}
```

```
10
Good Luck!
1
문자열이 일치하지 않습니다.
```



# 문자열 처리 함수

- 소문자를 대문자로 바꾸는 프로그램

```
// 아스키(ASCII) 코드
// 미국 ANSI에서 표준화한 정보 교환용 7비트 부호체계
// 7bit - 128개(0~127)
printf("%c\n", 'A');
printf("%d\n", 'A');

printf("%c\n", 'B');
printf("%d\n", 'B');

printf("%c\n", '\0'); //NULL문자 - 공백
printf("%d\n", '\0');

printf("%c\n", '1');
printf("%d\n", '1');

for (int i = 0; i < 128; i++) {
    printf("아스키코드 %d %c\n", i, i);
}
```



# 문자열 처리 함수

- 소문자를 대문자로 바꾸기

```
char sentence[] = "i am a student";
int length, i;

//문자열 인덱싱
printf("%c\n", sentence[0]);
printf("%c\n", sentence[1]);
printf("%c\n", sentence[2]);

length = strlen(sentence); //sentence 배열의 길이
printf("%d\n", length);

for (i = 0; i < length; i++)
{
    UpperCase(sentence[i]); //UpperCase() 호출
}
```

```
i
a
14
I AM A STUDENT
```



# 문자열 처리 함수

- 소문자를 대문자로 바꾸기

```
void UpperCase(char alpha)
{
    if (alpha >= 'a' && alpha <= 'z')
    {
        //소문자 b인경우 98-32=66 -> 대문자 B임
        //('a' - 'A') -> 97-65=32,
        alpha = alpha - ('a' - 'A');
    }
    printf("%c", alpha);
}
```



# 문자열 처리 함수

- strchr() – 개행 문자('\n') 찾을

```
char str[] = "Hello\nWorld";
int pos = strchr(str, "\n");

printf("%d\n", pos); //5

char msg[100];
fgets(msg, sizeof(msg), stdin); //사용자 입력(개행문자 포함됨)
str[strchr(msg, "\n")] = '\0'; // 개행 제거

/*
# strchr()은 '\n'의 위치를 찾아서 '\0'으로 수정
str[5] = '\0'
# fgets()로 저장된 실제 문자열
"Good Luck!\n\0"
# strchr() 처리후
"Good Luck!\0"
*/
printf("%s", msg);
```



# 문자열 자르기

- strtok(문자열, 구분기호) 함수

문자열을 구분 기호로 구분하여 잘라냄

```
#define _CRT_SECURE_NO_WARNINGS //strtok() 안전 모드
#include <stdio.h>
#include <string.h> //strtok() 사용
#include <stdlib.h> //srand(), rand() 사용
#include <time.h> //time() 사용

#define MAX_WORDS 10 //최대 단어 개수

int main()
{
    char words[] = "I am a student";
    char* wordList[MAX_WORDS]; //분리된 단어를 저장할 배열
    int idxOfWords = 0; //배열의 인덱스
```





# 문자열 자르기

- strtok(문자열, 구분기호) 함수

```
//첫째 단어 분리
//공백문자로 구분하여 ptr 포인터에 저장
char* ptr = strtok(words, " ");
printf("%s\n", ptr); //I

/*//두번째 단어 분리
ptr = strtok(NULL, " ");
printf("%s\n", ptr); //am */

//전체 단어 분리
while (ptr != NULL && idxOfWords < MAX_WORDS) {
    wordList[idxOfWords++] = ptr;
    ptr = strtok(NULL, " ");
}
printf("%d\n", idxOfWords); //인덱스의 크기
```



# 문자열 자르기

- 배열에서 랜덤하게 문자 추출하기

```
//분리된 단어 출력
for (int i = 0; i < idxOfWords; i++) {
    printf("%s ", wordList[i]);
}
printf("\n");

printf("\n== 문자열 추출(random) ===\n");
srand(time(NULL));
int rndIdx;

rndIdx = rand() % idxOfWords; //난수 저장
printf("%s\n", wordList[rndIdx]);
return 0;
}
```

```
I
4
I am a student
```

```
== 문자열 추출(random) ===
student
```



# 실습 – 숫자를 추측해서 맞추는 게임

## ■ 게임 방법

- 컴퓨터가 임의의 난수를 생성
- 사용자가 추측해서 1부터 50 사이의 수를 입력
- 추측한 수와 난수가 일치하면 "정답이에요" 출력  
추측한 수가 난수보다 크면 "너무 커요!", 아니면 "너무 작아요!" 출력
- 시도 횟수는 총 5번이고, 횟수가 0이면  
"남은 횟수가 0이에요. 아쉽게 실패했어요" 출력

```
남은 횟수 5 번  
맞혀 보세요 (1~50): 25  
너무 작아요!  
남은 횟수 4 번  
맞혀 보세요 (1~50): 35  
너무 작아요!  
남은 횟수 3 번  
맞혀 보세요 (1~50): 40  
정답이에요!
```



# 실습 – 숫자를 추측해서 맞추는 게임

- 숫자 추측 게임

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

int main()
{
    srand(time(NULL));
    int randNum = rand() % 50 + 1; //컴퓨터 난수
    int guessNum = 0; //사용자 추측한 수
    int count = 5; //시도한 회수

    //printf("%d\n", randNum);

    while (1)
    {
        printf("남은 횟수 %d번\n", count--);

        printf("맞혀보세요(1~50 입력): ");
        scanf_s("%d", &guessNum);
```



# 실습 – 숫자를 추측해서 맞추는 게임

- 숫자 추측 게임

```
    if (guessNum == randNum){
        printf("정답이에요!\n");
        break;
    }
    else if (guessNum > randNum){
        printf("너무 커요!\n");
    }
    else{
        printf("너무 작아요!\n");
    }

    if (count == 0){
        printf("남은 횟수가 0이에요! 아쉽게 실패했어요.\n");
        break;
    }
}
return 0;
}
```



# 실습 – 동전 던지기 게임

## ■ 게임 방법

- 동전을 던집니다.( 앞면은 1, 뒷면은 2)
- 앞면인지 뒷면인지 답을 합니다.
- 사용자와 동전이 같으면 "맞았음" 을 다르면 "꽝!"을 출력합니다.
- 1, 2 가 아닌 다른 값을 입력하면 종료합니다.

```
앞면은 1, 뒷면은 2, 종료는 다른 값을 입력하세요
동전을 던졌습니다. 앞면? 뒤면? : 1
사용자: 앞면 동전: 뒷면
꽝!
동전을 던졌습니다. 앞면? 뒤면? : 2
사용자: 뒷면 동전: 앞면
꽝!
동전을 던졌습니다. 앞면? 뒤면? : 1
사용자: 앞면 동전: 앞면
맞았음
동전을 던졌습니다. 앞면? 뒤면? : 3
게임을 종료합니다.
```



# 실습 – 동전 던지기 게임

- 동전 맞추기

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <time.h>
#include <stdlib.h>

int main()
{
    int you;
    int coin;
    //char aspect[][10] = { "", "앞면", "뒷면" };
    char* aspect[] = { "", "앞면", "뒷면" };
    srand(time(NULL));

    //printf("%s\n", aspect[0]); //앞면

    printf("앞면은 1, 뒷면은 2, 종료는 다른 값을 입력하세요\n");
```



# 실습 – 동전 던지기 게임

- 동전 맞추기

```
while (1) {  
    coin = rand() % 2 + 1;  
  
    printf("동전을 던졌습니다. 앞면? 뒤면? : ");  
    scanf("%d", &you);  
    if (you < 1 || you > 2) {  
        printf("게임을 종료합니다.\n");  
        break;  
    }  
    else {  
        printf("사용자: %s 동전: %s\n", aspect[you], aspect[coin]);  
        printf("%s\n", (you == coin) ? "맞았음" : "꽂!");  
    }  
}  
return 0;  
}
```

