

# C - 배열

# Visual Studio 2022



# 배열(Array)

## ❖ 배열은 왜 써야 할까?, 사용의 필요성

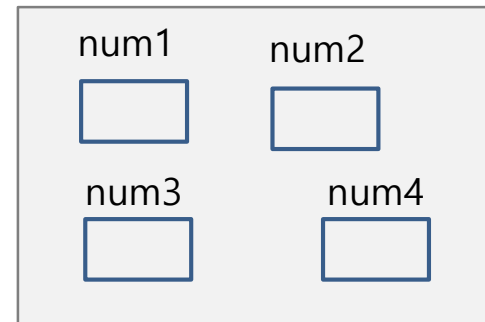
- 정수 10개를 이용한 프로그램을 할 때 10개의 정수 타입의 변수를 선언

`int num1, int num2, int num3... num10;`

정보가 흩어진 채 저장되므로

비효율적이고 관리하기 어렵다.

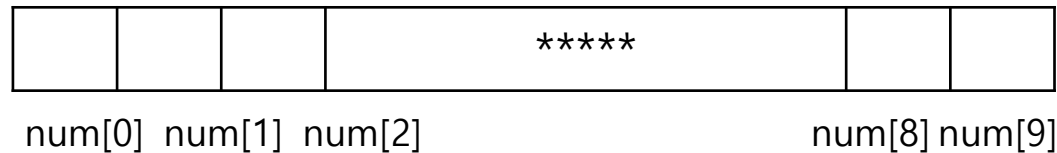
**메모리**



- 배열은 동일한 자료형의 변수를 한꺼번에 순차적으로 관리할 수 있다.

`int num[10];`

배열 이름  
1개



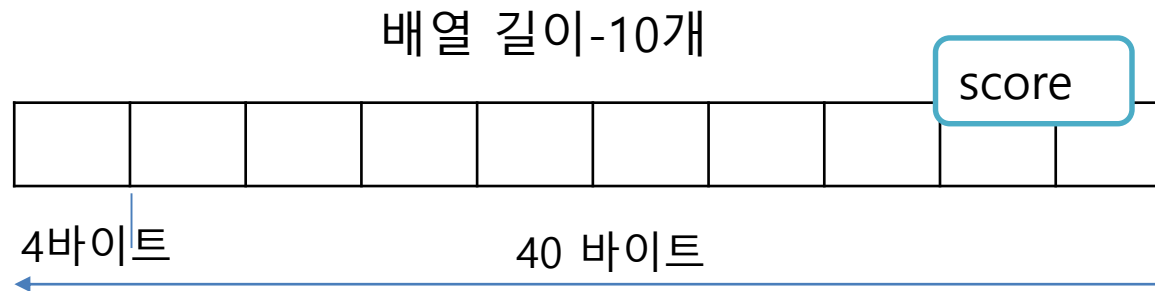
# 배열(Array)

- 배열이란?

여러 개의 연속적인 값을 저장하고자 할 때 사용하는 자료형이다.  
배열 변수는 [ ]안에 설정한 값만큼 메모리를 할당하여 저장한다.

- 배열 변수의 선언과 사용

```
int score[10];
```



# 배열(Array)

- 정수형 배열

```
int arr[] = { 10, 20, 30, 40 };

//배열의 이름은 배열의 시작 주소
printf("%x %x %x\n", &arr[0], &arr[1], &arr[2]);
printf("%x %x %x\n", arr, arr+1, arr+2);

//요소 접근(조회)
printf("%d\n", arr[0]);

//요소 수정
arr[1] = 55;

//요소 전체 조회
for (int i = 0; i < 4; i++) {
    printf("%d\n", arr[i]);
}
```



# 배열(Array)

```
//요소의 합계
int sum = 0;
for (int i = 0; i < 4; i++) {
    sum += arr[i];
}
printf("합계: %d\n", sum);

//요소의 평균
double avg = 0.0;
avg = (double)sum / 4; //실수형 형변환
printf("평균: %.11f\n", avg);

//요소의 최대값
int max = arr[0];
for (int i = 1; i < 4; i++) {
    if (arr[i] > max)
        max = arr[i];
}
printf("최대값: %d\n", max);
```



# 배열(Array)

- 문자형 배열

```
char c1, c2, c3;
c1 = 'C';
c2 = c1 + 1;
c3 = c2 - 1;

printf("%c %c\n", c2, c3);

//26개 크기를 문자열 배열 가진 생성
char alphabets[26];
char ch = 'A';

for (int i = 0; i < 26; i++) {
    alphabets[i] = ch;
    ch++;
}

for (int i = 0; i < 26; i++) {
    printf("%c %d\n", alphabets[i], alphabets[i]);
}
```



# 배열(Array)

- 문자열 배열

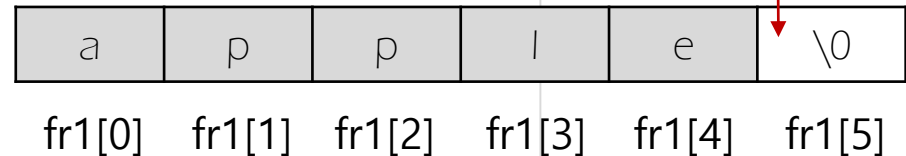
```
char fr1[] = "apple"; //맨 뒤에 널문자(NULL)  
char fr2[] = { 'a', 'p', 'p', 'l', 'e', '\0' };  
char fr3[] = "사과";
```

```
//sizeof() - 자료형의 크기를 바이트 단위로 반환('\0' 포함)  
printf("%d\n", sizeof(fr1));  
printf("%s\n", fr1);
```

```
printf("%d\n", sizeof(fr2));  
printf("%d\n", sizeof(fr3));
```

```
char name[20];  
printf("당신의 이름은 무엇입니까? ");  
scanf("%s", name);  
//scanf_s("%s", name, sizeof(name));  
  
printf("당신의 이름은 %s이군요.", name);
```

문자열의 끝을 나타내는 NULL문자 '\0' 자동으로 추가



# 배열(Array)

- 문자열 배열 크기 및 출력

```
char str[] = "Hello";
int num[] = { 1, 2, 3, 4 };

printf("문자 배열에 할당된 메모리의 크기 : %d바이트\n", sizeof(str));
printf("문자 배열에 할당된 메모리의 크기 : %d바이트\n", sizeof(num));

//첫번째 요소 출력
printf("%c\n", str[0]); //H
printf("%d\n", num[0]); //H

//배열의 크기
int size1 = sizeof(str) / sizeof(str[0]); // 6 / 1 = 6
int size2 = sizeof(num) / sizeof(num[0]); // 16 / 4 = 4
```





# 배열(Array)

- 문자열 배열 크기 및 출력

```
//배열의 크기
int size1 = sizeof(str) / sizeof(str[0]); // 6 / 1 = 6
int size2 = sizeof(str) / sizeof(num[0]); // 16 / 4 = 4

//'\0' NULL문자는 문자열(배열)의 끝을 의미함(필수)
//char msg1[9] = "Thank you";
char msg2[9] = { 'T', 'h', 'a', 'n', 'k', ' ', 'y', 'o', 'u' };
char msg3[] = { 'T', 'h', 'a', 'n', 'k', ' ', 'y', 'o', 'u', '\0' };

//printf("%s\n", msg1);
printf("%s\n", msg2);
printf("%s\n", msg3);
```



# 배열(Array)

- 배열 요소 저장 및 삭제

```
int arr[5];

//요소 추가
arr[0] = 1;
arr[1] = 2;
arr[2] = 3;
arr[3] = 4;
arr[4] = 5;

//출력
for (int i = 0; i < 5; i++)
{
    printf("%3d", arr[i]);
}

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

```
//2번 인덱스 삭제
//2번 인덱스 0으로 초기화
arr[2] = 0;

//배열의 인덱스 왼쪽으로 이동
for (int i = 2; i < 4; i++)
{
    arr[i] = arr[i + 1];
}

//출력
for (int i = 0; i < 4; i++)
{
    printf("%3d", arr[i]);
}
```



# 배열(Array)

- 배열 요소 저장 및 삭제

```
#include <stdio.h>
#define MAX_LEN 4

int main()
{
    int carts[MAX_LEN];
    int idxOfCarts = 0;

    // 요소 추가
    /*if (idxOfCarts >= MAX_LEN)
    {
        return;
    }

    carts[idxOfCarts++] = 80;
    carts[idxOfCarts++] = 70;
    carts[idxOfCarts++] = 95;
    carts[idxOfCarts++] = 90;
    //carts[idxOfCarts++] = 100;*/
```

```
//요소 삭제
carts[idxOfCarts--];
carts[idxOfCarts--];
carts[idxOfCarts--];
carts[idxOfCarts--];
//carts[idxOfCarts--];

if (idxOfCarts < 0)
{
    printf("더 이상 삭제할 수 없습니다.\n");
}
```



# 배열(Array)

- 배열 요소 입력 받기

```
while (1)
{
    printf("점수 입력: ");
    scanf_s("%d", &carts[idxOfCarts++]);

    if (idxOfCarts >= MAX_LEN)
    {
        puts("더 이상 추가할 수 없음!");
        break;
    }
}

for (int i = 0; i < idxOfCarts; i++)
{
    printf("%-4d", carts[i]);
}
return 0;
}
```



# 배열(Array)

- 배열의 복사

```
char a1[] = "NET";
char a2[4];

printf("%c\n", a1[0]);
printf("%c\n", a1[1]);
printf("%c\n", a1[3]); //NULL 문자
printf("%c\n", a1[2]);

for (int i = 0; i < 4; i++) {
    a2[i] = a1[i];
}
printf("%s\n", a2);
printf("=====\n");

//NET -> TEN으로 거꾸로 복사
for (int i = 0; i < 4; i++) {
    a2[i] = a1[2-i];
}
a2[3] = '\0';
printf("%s\n", a2);
```



# 배열(Array)

- 최대값 구하기

```
int arr[] = { 2, 71, 59, 33, 94, 25 };
int maxVal = arr[0];
int i;

for (i = 0; i < 6; i++) {
    if (maxVal < arr[i])
        maxVal = arr[i];
}

printf("최대값 : %d\n", maxVal);

int maxIdx = 0;
for (i = 0; i < 6; i++) {
    if (arr[maxIdx] < arr[i])
        maxIdx = i;
}
printf("최대값의 위치 : %d\n", maxIdx);
```



# 배열(Array) 예제

- 5개의 정수를 배열에 입력 받아 최소값 구하는 프로그램

```
1번째의 수 입력 : 70
2번째의 수 입력 : 60
3번째의 수 입력 : 90
4번째의 수 입력 : 10
5번째의 수 입력 : 80
최소값은 10
```

```
int n[5];
int i, min = 999;
for (i = 0; i < 5; i++) {
    printf("%d번째의 수 입력 : ", i + 1);
    scanf_s("%d", &n[i]);
    if (n[i] < min)
        min = n[i];
}
printf("최소값은 %d\n", min);
```



# 2차원 배열

## ■ 배열의 확장 : 2차원 배열

이정후의 1반 학생들의 키를 배열에 저장

```
int class1[5]
```

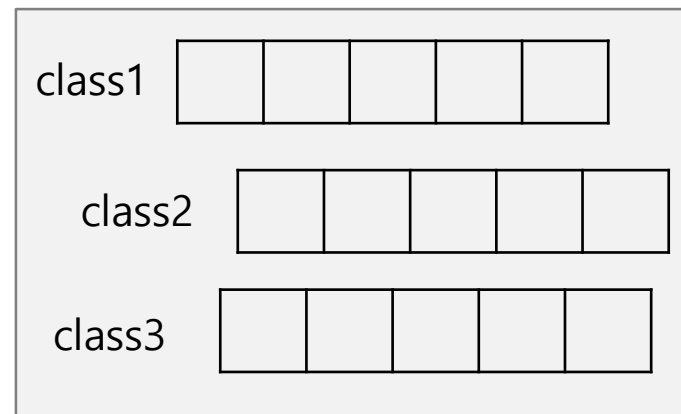
2반과 3반 학생들의 키를 배열에 저장

```
int class1[5]
```

```
int class2[5]
```

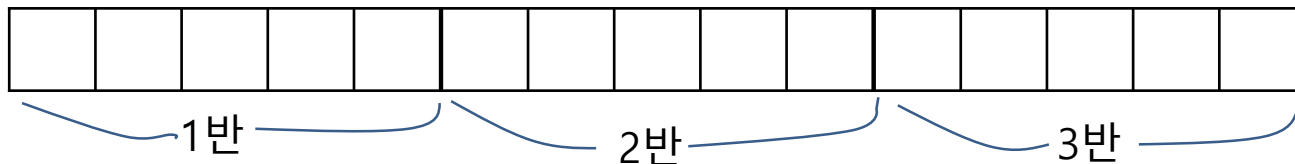
```
int class3[5]
```

메모리



## ■ 2차원 배열을 사용한 경우

```
int class[3][5]
```





# 2차원 배열

## ■ 배열의 확장 : 2차원 배열

1. 지도, 게임 등 평면이나 공간을 구현할 때 많이 사용됨.
2. 이차원 배열의 선언과 구조

```
int arr[2][3];
```

3. 선언과 초기화

```
int arr[ 2][3] = {  
    {1, 2, 3},  
    {4, 5, 6}  
};
```



arr[0][0]	arr[0][1]	arr[0][2]

arr[1][0]   arr[1][1]   arr[1][2]

arr[0][0]	arr[0][1]	arr[0][2]
1	2	3
4	5	6

arr[1][0]   arr[1][1]   arr[1][2]

# 이차원 배열(function)

- 이차원 배열 – 정수형 배열

학생 3명의 2과목 점수

Kim, Lee, Park

```
int a[3][2];
```

이름	수학	영어
Kim	75	80
Lee	85	95
Park	90	100



# 이차원 배열(function)

```
//저장 방법1
/*int a[3][2] = {75, 80, 85, 95, 90, 100}; */
```

```
//저장 방법2
int a[3][2] = {
    {75, 80},
    {85, 95},
    {90, 100}
};
```

```
int x, y;
```

```
//출력 방법1
```

```
for (x = 0; x < 3; x++) {
    printf("a[%d][0]=%d, a[%d][1]=%d\n", x, a[x][0], x, a[x][1]);
}
```

```
//출력 방법2
```

```
printf("=====이중 for=====\n");
for (x = 0; x < 3; x++) {
    for (y = 0; y < 2; y++) {
        printf("a[%d][%d]=%d, ", x, y, a[x][y]);
    }
    printf("\n");
}
```



# 이차원 배열(function)

- 이차원 배열 – 정수형 배열

```
int i, j, k = 0;
int a[2][3];

//k를 1 ~ 6까지 초기화(저장)
for (i = 0; i < 2; i++)
{
    for (j = 0; j < 3; j++)
    {
        a[i][j] = k + 1;
        k++;
    }
}

for (i = 0; i < 2; i++)
{
    for (j = 0; j < 3; j++)
    {
        printf("%d\n", a[i][j]);
    }
}
```



# 이차원 배열(function) 예제

- 학생에 5명에 대한 영어, 수학 과목의 합계와 평균 구하기

번호	국어	수학
1	70	90
2	85	85
3	90	95
4	80	70
5	65	50

```
//학생 5명의 국어, 수학 점수
int score[5][2] = {
    {90, 70},
    {84, 81},
    {95, 90},
    {80, 70},
    {75, 60}
};

int i, j;
int total[2] = { 0, 0 };
float avg[2] = { 0.0, 0.0 };

//출력
for (i = 0; i < 5; i++) {
    for (j = 0; j < 2; j++) {
        printf("%3d", score[i][j]);
    }
    printf("\n");
}
```



# 이차원 배열(function) 예제

- 학생에 5명에 대한 영어, 수학 과목의 합계와 평균 구하기

```
//합계
for (i = 0; i < 5; i++) {
    total[0] += score[i][0];
    total[1] += score[i][1];
}

//평균
avg[0] = (float)total[0] / 5;
avg[1] = (float)total[1] / 5;

printf("국어 합계 : %d\n", total[0]);
printf("수학 합계 : %d\n", total[1]);
printf("국어 평균 : %.2f\n", avg[0]);
printf("수학 평균 : %.2f\n", avg[1]);
```



# 이차원 배열(function) 예제

- 2차원 문자열 배열

```
char greet[] = "hello";
int i, j;

//문자열로 출력
printf("%s\n", greet);

//문자로 출력
for (i = 0; greet[i] != NULL; i++)
{
    printf("%c", greet[i]);
}

printf("\n=====");

//words[단어의 개수][최대 문자의 수]
char words[3][10] = {
    "sun",
    "moon",
    "earth"
};
```



# 이차원 배열(function) 예제

- 2차원 문자열 배열

```
//요소 전체 조회(문자로 출력)
for (i = 0; i < 3; i++)
{
    for (j = 0; words[i][j] != NULL; j++)
    {
        printf("%c", words[i][j]);
    }
    printf("\n");
}

//printf("\n=====");
//요소 전체 조회(문자열로 출력)

int size = sizeof(words) / sizeof(words[0]); //요소의 개수
printf("%d\n", size);

for (i = 0; i < size; i++)
{
    printf("%s\n", words[i]);
}
```





# 실습 문제

- 학생에 5명에 대한 영어, 수학 점수를 입력받아 평균 계산하기

번호	영어	수학
1	70	90
2	85	85
3	90	95
4	80	70
5	65	50

```
int score[5][2];
int total[2] = { 0, 0 };
int i, j;
printf("각 학생의 영어 점수와 수학 점수를 입력하세요\n");
for (i = 0; i < 5; i++)
{
    printf("%d번 학생의 영어 점수 ", i + 1);
    scanf_s("%d", &score[i][0]);
    printf("%d번 학생의 수학 점수 ", i + 1);
    scanf_s("%d", &score[i][1]);
}
```



# C - 함수

# Visual Studio 2022



# 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
}
```

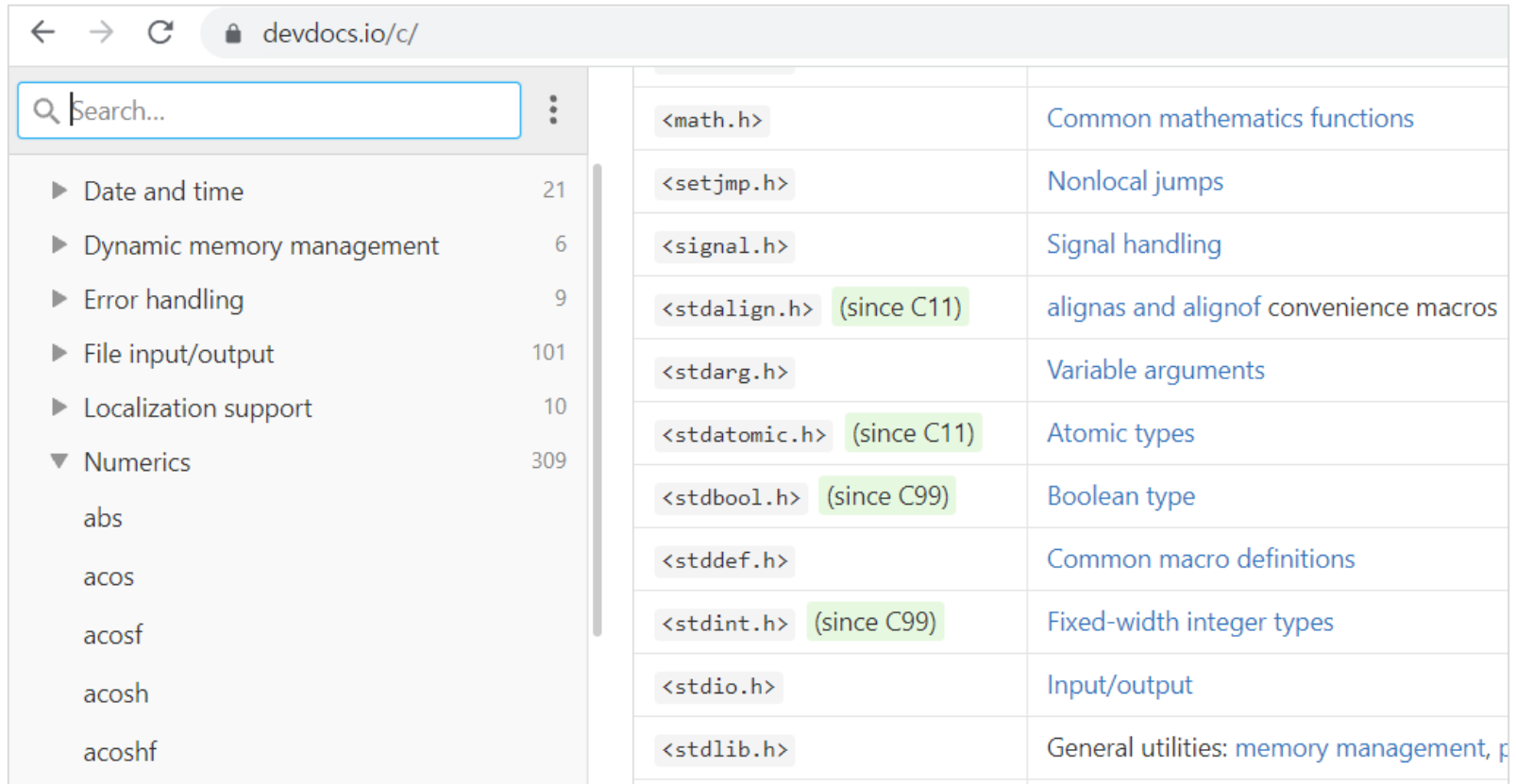
```
int getArea(x, y)
{
    return x * y
}
```



# 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
- ▼ Numerics 309
  - abs
  - acos
  - acosf
  - acosh
  - acoshf

On the right, there is a table listing 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>

int main() {
    //반올림
    printf("%.2f\n", round(2.54));
    printf("%.2f\n", round(-2.54)); //작은쪽 정수로 결과 출력
    printf("%.2lf\n", round(2.54));

    //내림
    printf("%.2f\n", floor(11.3));
    printf("%.2f\n", floor(-11.3));
    printf("%.2lf\n", floor(11.3));

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

    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));
}
```



# 시간 함수(function)

## ✓ 수행 시간 측정하기

```
//수행시간 측정하기
time_t start, end;
start = time(NULL); //시작
printf("시작시간: %ld\n", start); // %ld - long decimal

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

end = time(NULL); //종료
printf("종료시간: %ld\n", end);

printf("수행시간: %ld초\n", (end-start));
```



# 시간 함수(function)

## ✓ 현재 날짜와 시간 표시하기

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

int main() {
    // 현재 시간을 가져오기 위한 time_t 변수 선언
    time_t ct;
    struct tm* now; //현재 날짜와 시간(tm 구조체 포인터 객체)

    // 현재 시간 가져오기
    ct = time(NULL);
    now = localtime(&ct); //localtime 함수로 포매팅

    // 날짜 및 시간 출력
    printf("현재 년도: %d\n", now->tm_year + 1900);
    printf("현재 월: %d\n", now->tm_mon + 1);
    printf("현재 일: %d\n", now->tm_mday);
    printf("현재 날짜: %d. %d. %d.\n",
        now->tm_year + 1900, now->tm_mday, now->tm_mday);
}
```





# 실습 문제

✓ 현재 요일을 아래와 같이 출력하시오.

👉 실행 결과

```
현재 요일: 1
오늘은 월요일입니다.
```

```
printf("현재 시: %d\n", now->tm_hour);
printf("현재 분: %d\n", now->tm_min);
printf("현재 초: %d\n", now->tm_sec);
printf("현재 시간: %d : %d : %d.\n",
      now->tm_hour, now->tm_min, now->tm_sec);

//현재 요일
printf("현재 요일: %d\n", now->tm_wday); //0-일, 1-월, 2-화...

//현재 요일을 출력(조건문 사용)

return 0;
}
```



# rand() 함수(function)

- rand() 함수 – srand() 함수 필요함

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

int main(void)
{
    // 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);
}
```



# 실습 문제

## ■ 주사위 10번 던지기

```
// 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);

//실습 - 주사위 10번 던지기

return 0;
```

## 👉 실행 결과

```
주사위 눈 : 4
3
4
4
2
3
6
2
1
6
3
```



# rand() 함수(function)

- rand() 함수 – srand() 함수 필요함

```
//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>gets</b> (char* Bufffer)	<stdio.h>	문자열 저장[ scanf()와 유사함 ](사용안함)
<b>puts</b> (char* Buffer)	<stdio.h>	문자열 출력[ printf()와 유사함 ]
<b>fgets</b> (char* Buffer, int MaxCount, FILE* Stream)	<stdio.h>	공백을 포함한 문자열 입력 가능 엔터(\n)까지 포함하여 저장됨
<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

- 문자열 출력 및 입력 – fgets(), puts()

```
/* scanf()
   공백(space) 또는 개행(\n)을 만나면 입력이 끝남.
   입력 버퍼에는 '\n'이 남아있음.
   즉, 공백 포함문자를 입력받을 수 없음
*/
char name[20];
printf("이름을 입력하세요: "); //name - apple
//scanf_s("%s", name, sizeof(name));
//gets(name); //지금 사용하지 않음
fgets(name, sizeof(name), stdin);

//printf("입력된 이름: %s\n", name);
printf("입력된 이름:");
puts(name);
```



# 문자열 처리 함수

- 문자열 복사, 개수, 비교 – strcpy(), strlen(), strcmp()

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

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

    //문자열의 개수
    printf("%d\n", strlen(msg1));

    //문자열의 복사
    printf("%s\n", strcpy(msg2, msg1));

    return 0;
}
```



# 문자열 처리 함수

- 문자열 복사, 개수, 비교 – strcpy(), strlen(), strcmp()

```
//문자열의 비교
char greet1[] = "hello";
char greet2[10];

printf("문자열을 입력하세요: ");
scanf_s("%s", greet2, sizeof(greet2));

//일치 - 0, 불일치 - 1
if (strcmp(greet1, greet2) == 0)
{
    printf("문자열이 일치합니다.\n");
}
else
{
    printf("문자열이 일치하지 않습니다.\n");
}

printf("%d\n", strcmp(greet1, greet2));
```





# 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 – 매개변수는 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 – 매개변수는 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 – 매개변수는 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) 예제

- 정사각형과 삼각형의 넓이 구하는 프로그램

- 정사각형

- 한 변의 길이 : 4cm

- ▷ 삼각형

- 밑변 : 3cm, 높이 : 4cm

```
#include <stdio.h>
int square(int);
int triangle(int, int);
int main()
{
    int rec_area;
    int tri_area;
    rec_area = square(4);
    tri_area = triangle(3, 4);
    printf("정사각형의 넓이 : %d\n", rec_area);
    printf("삼각형의 넓이 : %d\n", tri_area);
}

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

int triangle(int b, int h) {
    return b * h / 2;
}
```



# 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;
}
```





# 함수(function) 예제

- 소문자를 대문자로 바꾸기

```
void UpperCase(char);
int main()
{
    char buf[] = "I am a student";
    int length, i;

    /*printf("%c\n", buf[0]);
    printf("%c\n", buf[1]);
    printf("%c\n", buf[2]);*/

    length = strlen(buf);
    //printf("%d\n", length);

    for (i = 0; i < length; i++)
    {
        UpperCase(buf[i]);
    }

    return 0;
}
```

```
void UpperCase(char data)
{
    if (data >= 'a' && data <= 'z')
    {
        //data = data - ('a' - 'A'); //대문자로 변환
        data -= ('a' - 'A');
    }
    printf("%c", data);
}
```



# 함수(function) 예제

- 소문자를 대문자로 바꾸기

```
// 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);
}
```



# 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

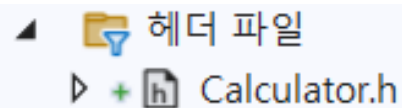
    x += 1;
    printf("현재 호출은 %d번째입니다.\n", x);
}
```



# 헤더 파일 사용하기

## ❖ 헤더파일 사용하기

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



<Calculator.h>

```
//Calculator.h
```

```
int count = 0; //변수 선언  
int add(int, int); //함수
```

<Calculator.c>

```
//Calculator.c
```

```
int add(int x, int y)  
{  
    int total;  
    total = x + y;  
    return total;  
}
```



# 헤더 파일 사용하기

## ❖ 헤더파일 사용하기

<CalculatorMain.c>

```
#include "Calculator.h"; //쌍따옴표("") 사용
#include <stdio.h>

int main()
{
    int x = 3, y = 4, result;
    count++;

    result = add(x, y);
    printf("count = %d\n", count);
    printf("result = %d\n", result);

    return 0;
}
```





# 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 0입니다. 아쉽게 실패했어요  $\pi.\pi$ " 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

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

int main_UpAndDown()
{
    //Up And Down 게임
    srand(time(NULL));
    int randNum = rand() % 50 + 1;

    int guessNum = 0;
    int count = 5;
    //printf("숫자: %d\n", randNum);

    while (1) //1-true, 0-false
    {
        printf("남은 횟수 %d 번\n", count--);

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



# 실습 – 숫자를 추측해서 맞추는 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 0입니다. 아쉽게 실패했어요ㅠ.ㅠ.\n");
    break;
}
}
return 0;
```



# 실습 – 영어 타이핑 게임

- 게임 방법
  - 컴퓨터가 저장된 영어 단어를 랜덤하게 출력함
  - 사용자가 단어를 따라 입력함
  - 출제된 단어와 입력한 단어가 일치하면 "통과!", 아니면 "오타! 다시 도전" 출력
  - 횟수는 총 10번이고, 끝나면 "게임을 종료합니다." 출력

영어 타자 게임, 준비되면 엔터 >

```
문제 1
bear
bear
통과!
문제 2
dog
dog
통과!
문제 3
cow
cow
통과!
문제 4
elephant
elepan
오타! 다시 도전!
문제 4
dog
dog
통과!
문제 5
bear
```



# 실습 – 영어 타이핑 게임

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

int main()
{
    char* words[] = {"ant", "bear", "chicken", "deer", "elephant", "fox",
                    "horse", "monkey", "lion", "tiger"};
    char* question; //출제된 문제
    char answer[30]; //사용자 입력

    int size = sizeof(words) / sizeof(words[0]); //배열의 크기

    srand(time(NULL)); //seed 설정

    int n = 1; //문제
    printf("영어 타자 게임, 준비되면 엔터 >>\n");
    getchar();

    time_t start, end;
    time(&start);
```



# 실습 – 영어 타이핑 게임

```
while (n <= 10)
{
    printf("문제 번호: %d\n", n);
    int rnd = rand() % size;
    question = words[rnd];
    printf("%s\n", question);

    scanf_s("%s", answer, sizeof(answer)); //입력

    if (strcmp(question, answer) == 0)
    {
        printf("통과!\n");
        n++;
    }
    else
    {
        printf("오타! 다시 도전!\n");
    }
}
time(&end);

printf("게임 소요시간: %ld초", (end - start));
```

