

C – 배열(array)



Visual Studio 2022



배열(Array)

❖ 배열은 왜 써야 할까?, 사용의 필요성

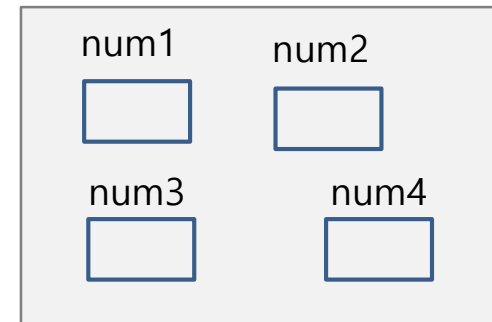
- 정수 10개를 이용한 프로그램을 할 때 10개의 정수 타입의 변수를 선언

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

정보가 흩어진 채 저장되므로

비효율적이고 관리하기 어렵다.

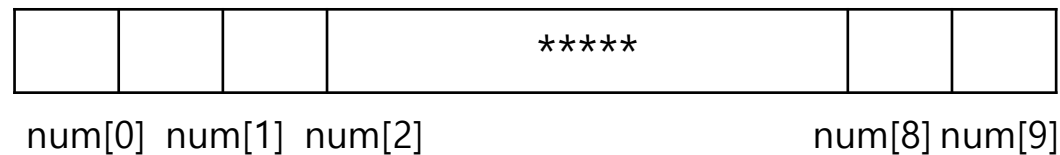
메모리



- 배열은 동일한 자료형의 변수를 한꺼번에 순차적으로 관리할 수 있다.

`int num[10];`

배열 이름
1개

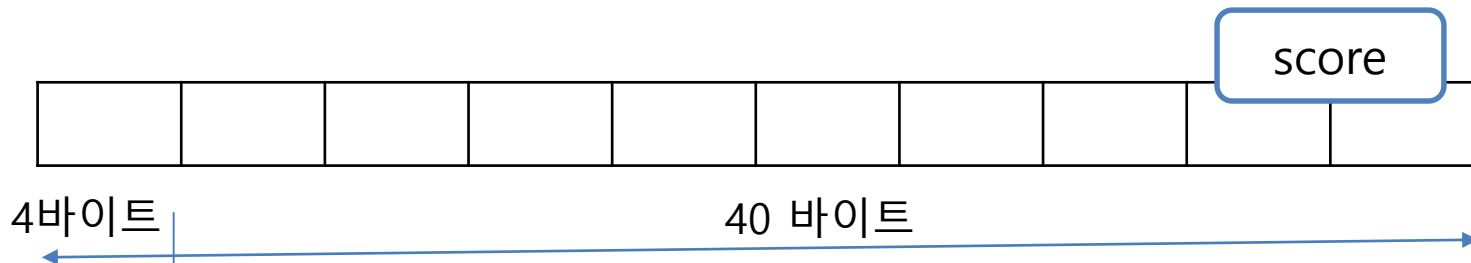


배열(Array)

- 배열이란?
 - 동일한 자료형의 여러 개의 연속적인 값을 저장할 때 사용하는 자료형이다.
 - 배열 변수는 대괄호([]) 안에 설정한 값만큼 메모리를 할당하여 저장한다.
 - 배열에서 변수하나에 해당하는 공간을 요소라고 하고, 인덱스(index)라는 번호로 구분한다.
- 배열 변수의 선언과 초기화

자료형 배열명[배열크기] 예) `int score[10];`

자료형 배열명[배열크기] = {요소1, 요소2, 요소3};



배열(Array)

- 정수형 배열 생성 및 관리

```
//정수형 배열 선언
int arr[4];

//요소 추가
arr[0] = 10;
arr[1] = 20;
arr[2] = 30;
arr[3] = 40;

//배열 선언과 동시에 초기화
//int arr[4] = { 10, 20, 30, 40 };

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



배열(Array)

- 정수형 배열 생성 및 관리

```
//특정 요소 검색
printf("%d\n", arr[0]);

//요소 수정
arr[1] = 55;

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

```
f86ff8f8 f86ff8fc f86ff900
f86ff8f8 f86ff8fc f86ff900
10
10 55 30 40
```



배열(Array)

- 문자형 배열 생성 및 관리

```
//문자형 배열 생성
/*char msg[5];
...
msg[0] = 'h';
msg[1] = 'e';
msg[2] = 'l';
msg[3] = 'l';
msg[4] = 'o';*/
```

```
//배열의 크기가 6인 문자형 배열 생성
char msg[6] = { 'h', 'e', 'l', 'l', 'o' };

//특정 요소 검색
printf("%c\n", msg[4]);

//요소 수정
msg[0] = 'y';

//요소 추가
msg[5] = 'w';

//요소 출력
for (int i = 0; i < 6; i++) {
    printf("%c ", msg[i]);
}
```

```
o
y e l l o w
```



배열(Array)

- 알파벳 소문자를 저장한 배열

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

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

//26개 크기를 가진 문자형 배열 생성
char alphabets[26];
char ch = 'a';

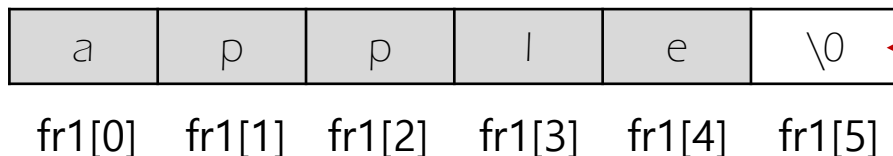
//저장
for (int i = 0; i < 26; i++) {
    alphabets[i] = ch;
    ch++;
}
//a ~ z와 아스키 코드값 출력
for (int i = 0; i < 26; i++) {
    printf("%c %d\n", alphabets[i], alphabets[i]);
}
```

```
b a
a 97
b 98
c 99
d 100
e 101
f 102
g 103
h 104
i 105
j 106
k 107
l 108
m 109
n 110
o 111
p 112
q 113
r 114
s 115
t 116
u 117
v 118
w 119
x 120
y 121
z 122
```



배열(Array)

■ 문자열 배열 생성 및 관리



문자열의 끝을 나타내는 NULL문자 '\0' 자동으로 추가

```
//문자열 배열
char fr1[5] = "apple";
char fr2[6] = "apple"; //맨 뒤에 널(NULL)문자 있음
char fr3[] = {'a', 'p', 'p', 'l', 'e', '\\0'};
char fr4[] = "바나나"; //한글은 1자에 2byte

//sizeof() - 자료형의 크기를 바이트 단위로 변환('\\0' 포함)
printf("%s %d\\n", fr1, sizeof(fr1));
printf("%s %d\\n", fr2, sizeof(fr2));
printf("%s %d\\n", fr3, sizeof(fr3));
printf("%s %d\\n", fr4, sizeof(fr4));
```

```
apple做做做做做做做做做做?pple 5
apple 6
apple 6
바나나 7
```



배열(Array)

- 배열의 크기 및 출력

```
//문자형 배열의 크기 및 출력
char msg[] = "Good Luck";

printf("배열 msg의 메모리 크기: %dByte\n", sizeof(msg));
printf("첫째 요소의 메모리 크기: %dByte\n", sizeof(msg[0]));

//배열의 크기
size = sizeof(msg) / sizeof(msg[0]);
printf("배열의 크기: %d\n", size);

//문자로 출력
for (int i = 0; i < size; i++) {
    printf("%c", msg[i]);
}

//문자열 출력
printf("%s", msg);
```



회원 정보

- 회원 정보 입력 및 출력

```
char id[20], password[256], name[30];  
float weight, height;  
  
printf("\n==== 회원 정보 입력 =====\n");  
printf("아이디 입력: ");  
//sizeof(id) - 입력 크기를 제한해 버퍼 오버플로우 방지  
scanf_s("%s", id, sizeof(id));  
  
printf("비밀번호 입력: ");  
scanf_s("%s", password, sizeof(password));  
  
printf("이름 입력: ");  
scanf_s("%s", name, sizeof(name));  
  
printf("몸무게 입력: ");  
scanf_s("%f", &weight);
```



회원정보

- 회원 정보 입력 및 출력

```
printf("키 입력: ");  
scanf_s("%f", &height);  
  
printf("\n==== 회원 정보 출력 =====\n");  
printf("아이디: %s\n", id);  
printf("비밀번호: %s\n", password);  
printf("이름: %s\n", name);  
printf("몸무게: %.1f\n", weight);  
printf("키: %.1f\n", height);
```

```
==== 회원 정보 입력 ====  
아이디 입력: cloud12  
비밀번호 입력: k1234  
이름 입력: 장그래  
몸무게 입력: 65.27  
키 입력: 172.34  
  
==== 회원 정보 출력 =====  
아이디: cloud12  
비밀번호: k1234  
이름: 장그래  
몸무게: 65.3  
키: 172.3
```



배열(Array)의 복사

- 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]);

//a1을 a2에 복사
for (int i = 0; i < 4; i++)
{
    a2[i] = a1[i];
}

//a2를 문자로 출력
for (int i = 0; i < 4; i++)
{
    printf("%c", a2[i]);
}
```

```
N
E

T
NET
NET
=====
TEN
```



배열(Array)의 복사

- 배열의 복사

```
//a2를 문자열로 출력
printf("%s\n", a2);
printf("=====\n");

//a1을 a2에 거꾸로 복사하기(NET -> TEN)
for (int i = 0; i < 4; i++)
{
    a2[i] = a1[2 - i];
}
a2[3] = '\0';

//a2를 문자열로 출력
printf("%s\n", a2);
```



배열(Array)의 정렬

- 배열 요소값 교환하기

```
int a[5] = { 3, 2, 5, 1, 4 };
int i, temp;

//현 위치의 요소와 다음 요소 교환
for (i = 0; i < 4; i++)
{
    temp = a[i];
    a[i] = a[i + 1];
    a[i + 1] = temp;
}

for (i = 0; i < 5; i++)
{
    printf("%d ", a[i]);
}
```

```
/*
i=0, 2 3 5 1 4
i=1, 2 5 3 1 4
i=2, 2 5 1 3 4
i=3, 2 5 1 4 3
*/
```

2 5 1 4 3



배열(Array)의 정렬

- 버블 정렬(Bubble Sort)

인접한 두 요소를 비교하며 큰 값을 오른쪽으로 이동시킨다.

```
for (i = 0; i < 5; i++)
{
    for (j = 0; j < 4; j++)
    {
        if (a[j] > a[j + 1])
        {
            temp = a[j];
            a[j] = a[j + 1];
            a[j + 1] = temp;
        }
    }
}

for (i = 0; i < 5; i++)
{
    printf("%d ", a[i]);
}
```

```
a[] = { 3, 2, 5, 1, 4 }
i=0, j=0, 2 3 5 1 4 //교환
      j=1, 2 3 5 1 4 //유지
      j=2, 2 3 1 5 4 //교환
      j=3, 2 3 1 4 5 //변경 저장
i=1, j=0, 2 3 1 4 5
      j=1, 2 1 3 4 5 //변경 저장
i=2, j=0, 1 2 3 4 5 //최종 저장
i=3, 교환없음
i=4, 교환없음
```



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) 관리

- 배열 요소 저장 및 삭제

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

int main()
{
    //배열의 크기는 변수를 사용할 수 없음
    int len = 4;
    //int array[len]; //컴파일 에러 발생

    //배열의 크기가 4인 carts 생성
    int carts[MAX_LEN];
    int idxOfCarts = 0; //배열의 인덱스

    //요소 추가
    if (idxOfCarts < MAX_LEN) carts[idxOfCarts++] = 80;
    if (idxOfCarts < MAX_LEN) carts[idxOfCarts++] = 70;
    if (idxOfCarts < MAX_LEN) carts[idxOfCarts++] = 95;
    if (idxOfCarts < MAX_LEN) carts[idxOfCarts++] = 80;
```



배열(Array) 관리

- 배열 요소 저장 및 삭제

```
//요소 삭제
if (idxOfCarts > 0) idxOfCarts--;
if (idxOfCarts > 0) idxOfCarts--;
if (idxOfCarts > 0) idxOfCarts--;
//if (idxOfCarts > 0) idxOfCarts--;

// 출력
printf("현재 배열 상태:\n");
if (idxOfCarts == 0) {
    printf("(비어 있음)\n");
}
else {
    printf("남은 요소 수: %d\n", idxOfCarts);
    for (int i = 0; i < idxOfCarts; i++) {
        printf("%d\n", carts[i]);
    }
}
```

```
현재 배열 상태 :
남은 요소 수 : 1
80
```



배열(Array) 연산

- 정수형 배열의 연산

```
// 1부터 10까지 더하기
int i, a[10];
int total = 0; //합계

// 1 ~ 10 저장 및 계산
for (i = 0; i < 10; i++)
{
    a[i] = i + 1;
    total += a[i];
    printf("a[%d]=%d, total=%d\n", i, a[i], total);
}
printf("합계: %d\n", total);
```

```
a[0]=1, total=1
a[1]=2, total=3
a[2]=3, total=6
a[3]=4, total=10
a[4]=5, total=15
a[5]=6, total=21
a[6]=7, total=28
a[7]=8, total=36
a[8]=9, total=45
a[9]=10, total=55
합계 : 55
```



배열(Array) 연산

- 정수형 배열의 연산

```
int score[5] = { 85, 75, 90, 75, 80 };
int i; //반복 변수
int sum = 0; //합계
double avg; //평균
int min, max; //최소값, 최대값

//배열의 크기(개수) - count
int count = sizeof(score) / sizeof(score[0]); //20byte / 4byte = 5
printf("배열의 크기: %d\n", count);

//성적의 합계
//score[0] + score[1] + score[2] ...
for (i = 0; i < count; i++)
{
    sum += score[i]; //sum = sum + score[i];
}
printf("합계: %d\n", sum);
```



배열(Array) 연산

- 정수형 배열의 연산

```
//평균 = 합계 / 개수
avg = (double)sum / count; //우측이 int가 되므로 (double)로 강제 형변환
printf("평균: %.1f\n", avg);

//최소값
min = score[0]; //배열의 첫째값을 최소값 설정
for (i = 1; i < count; i++)
{
    if (score[i] < min) //비교할 점수가 최소값보다 작으면
        min = score[i]; //그 점수를 최소값에 저장
}
/*
    i=1; 85 < 75, min=75
    i=2; 90 < 75, min=75
    i=3; 75 < 75, min=75
    i=4; 80 < 75, min=75
*/
printf("최소값: %d\n", min);
```

배열의 크기: 5
합계: 405
평균: 81.0
최소값: 75
최대값: 90



데이터 입력 받기

- 5개의 정수를 배열에 입력 받아 최소값 구하는 프로그램

```
1번째의 수 입력: 70
2번째의 수 입력: ha
잘못된 입력입니다! 숫자를 입력하세요.
2번째의 수 입력: 80
3번째의 수 입력: 90
4번째의 수 입력: 75
5번째의 수 입력: 99
최소값은 70
```

```
int arr[5];    //배열 선언
int idx = 0;   //배열의 인덱스
int min = 999; //최소값 설정

while (idx < 5)
{
    printf("%d번째의 수 입력: ", idx + 1);
    /*scanf_s("%d", &arr[idx]);
    idx++; */
```



데이터 입력 받기

- 5개의 정수를 배열에 입력 받아 최소값 구하는 프로그램

```
//문자 입력시 오류 처리
if (scanf_s("%d", &arr[idx]) == 1)
{
    if (arr[idx] < min)
        min = arr[idx];
    idx++; //인덱스 1증가
}
else
{
    puts("잘못된 입력입니다! 숫자를 입력하세요.");
    while (getchar() != '\n'); //입력 버퍼 비우기
}
}
printf("최소값은 %d\n", min);
```



2차원 배열

배열의 확장 : 2차원 배열

이정후의 1반 학생들의 키를 배열에 저장

```
int class1[5]
```

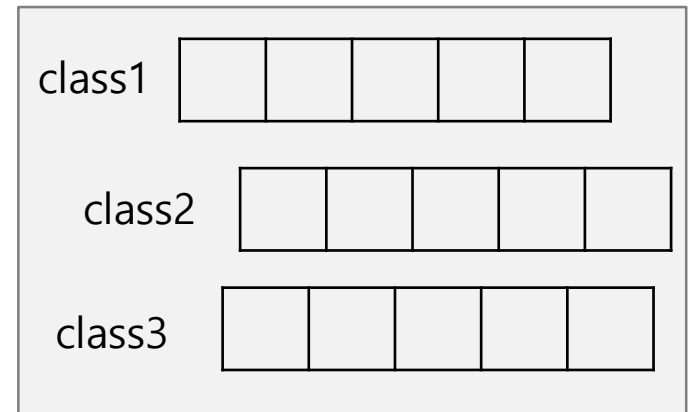
2반과 3반 학생들의 키를 배열에 저장

```
int class1[5]
```

```
int class2[5]
```

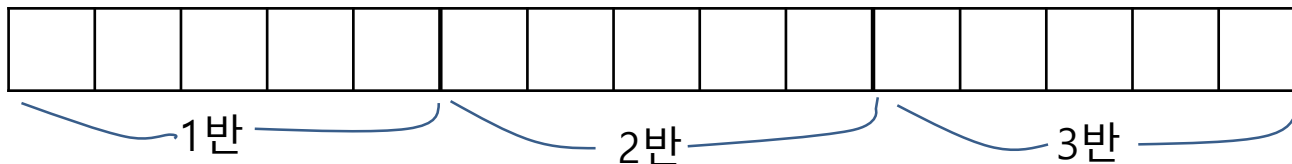
```
int class3[5]
```

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 a[3][3];
int k = 0;

//요소 값 저장
for (int i = 0; i < 3; i++)
{
    for (int j = 0; j < 3; j++)
    {
        k++;
        a[i][j] = k;
    }
}

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

1	2	3
4	5	6
7	8	9



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

```
//1차원 배열 - 문자열
char greet[] = "hello";
int i, j;

printf("%s\n", greet);

//2차원 배열 - words[단어의 개수][최대 문자의 수]
char words[3][10] = {
    "sun",
    "moon",
    "earth"
};
int i, j;

//특정 단어 조회
printf("%s\n", words[0]);
printf("%s\n", words[1]);
printf("%s\n", words[2]);
```



이차원 배열(function) 예제

- 2차원 문자열 배열

```
//요소 전체 조회(문자열로 출력)
int size = sizeof(words) / sizeof(words[0]); //요소의 개수

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

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



실습 문제 – 배열 입력

- 학생에 5명에 대한 영어, 수학 점수를 입력받아 평균 계산하기

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

```
*** 영어, 수학의 총점과 평균 ***
1번 학생의 영어 점수 : 70
1번 학생의 수학 점수 : 90
2번 학생의 영어 점수 : 85
2번 학생의 수학 점수 : 85
3번 학생의 영어 점수 : 90
3번 학생의 수학 점수 : 95
4번 학생의 영어 점수 : 80
4번 학생의 수학 점수 : 70
영어 평균 : 325
수학 평균 : 340
영어 평균 : 81.2
수학 평균 : 85.0
```



실습 문제 1 – 배열

배열 길이가 5인 정수 배열을 선언하고, 1~10중 홀수만을 배열에 저장한 후 그 합과 평균을 계산하세요.

👉 실행 결과

합 계 : 25
평균 : 6.2



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
}
```



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) 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;
}
```



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

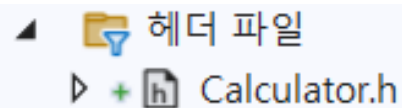
    x += 1;
    printf("현재 호출은 %d번째입니다.\n", x);
}
```



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  
  
int count = 0;    //전역 변수  
int add(int, int); //함수 선언부  
int factorial(int);
```



헤더 파일 사용하기

❖ 헤더파일 사용하기

<Calculator.c>

```
//함수 구현부
int add(int x, int y)
{
    int sum;
    sum = x + y;
    return sum;
}
```

```
int factorial(int n)
{
    int facto = 1;
    for (int i = 1; i <= n; i++)
        facto *= i;
    /*
        n = 4인 경우
        i = 1, facto = 1 * 1
        i = 2, facto = 1 * 2
        i = 3, facto = 2 * 3
        i = 4, facto = 6 * 4
    */
    return facto;
}
```



헤더 파일 사용하기

❖ 헤더파일 사용하기

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

<CalculatorMain.c>

```
int main()
{
    int x = 3, y = 4, value1, value2;

    count++; //count 1증가
    value1 = add(x, y); //add() 호출
    value2 = factorial(y); //factorial() 호출

    printf("count = %d\n", count);
    printf("value1 = %d\n", value1);
    printf("%d! = %d\n", y, value2);
    printf("5! = %d\n", factorial(5));

    return 0;
}
```

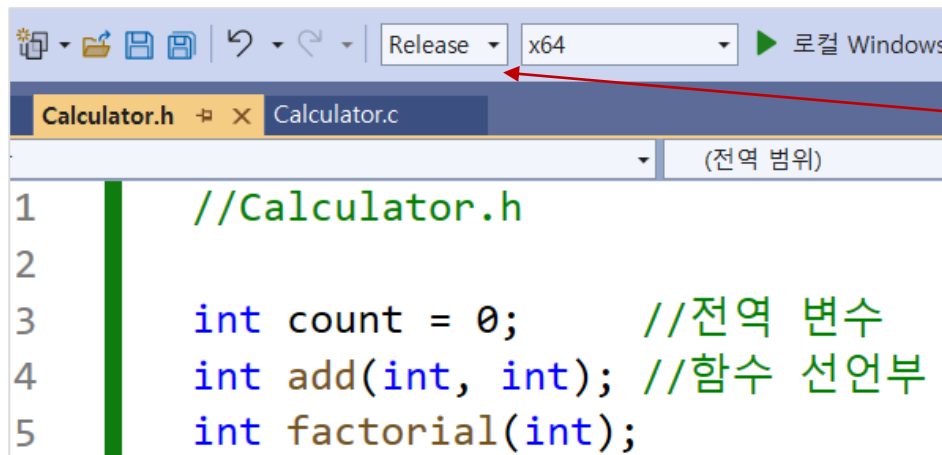
```
count = 1
value1 = 7
4! = 24
5! = 120
```



파일 배포

◆ 파일 배포

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



The screenshot shows a code editor window with two tabs: 'Calculator.h' and 'Calculator.c'. The 'Calculator.c' tab is active, showing the following code:

```
1 //Calculator.h
2
3 int count = 0; //전역 변수
4 int add(int, int); //함수 선언부
5 int factorial(int);
```

At the top of the editor, there is a toolbar with a dropdown menu set to 'Release' and another dropdown set to 'x64'. A red arrow points from the 'Release' dropdown to a text box on the right.

Debug 모드를
Release 모드로 바꾼다.



Ctrl + F5로 실행



파일 배포

◆ exe 파일이 실행되지 않는 문제 해결

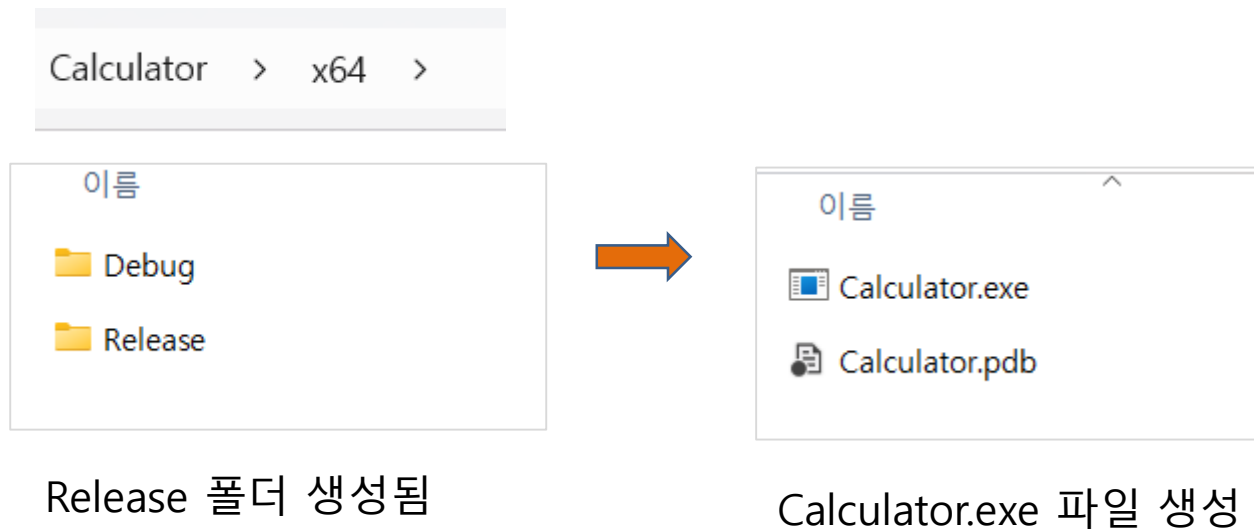
system("pause") 를 명시함

```
printf("count = %d\n", count);  
printf("value1 = %d\n", value1);  
printf("%d! = %d\n", y, value2);  
printf("5! = %d\n", factorial(5));  
  
system("pause"); //exe 파일 실행시 프로세스 유지  
  
return 0;  
}
```



파일 배포

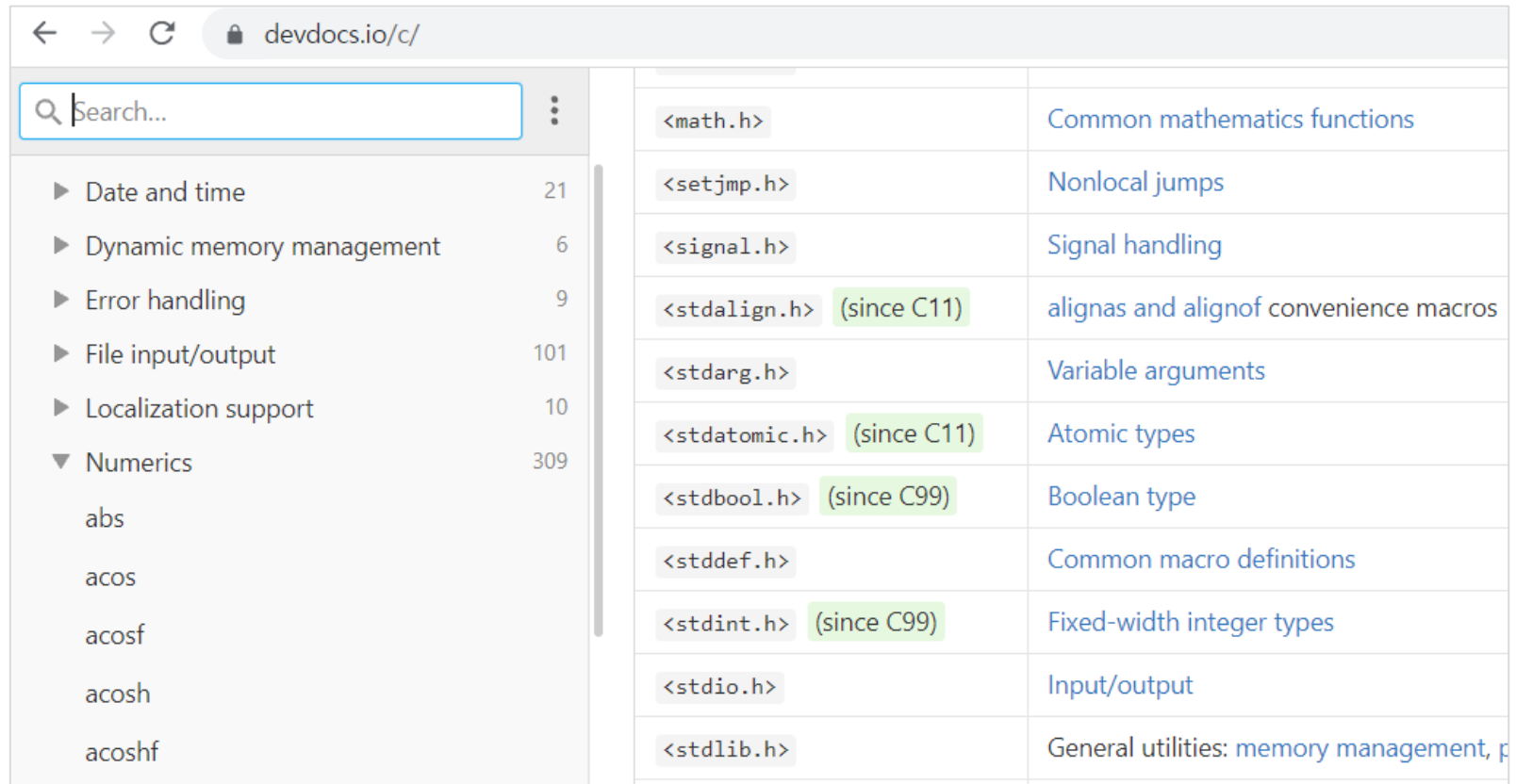
◆ 파일 배포



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. The 'Numerics' category is expanded, showing a list of functions. On the right, there is a table listing C standard library headers and their descriptions.

Header	Description
<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> //round(), floor()... 사용

int main()
{
    //반올림
    printf("%.1f\n", round(2.54));
    printf("%.1f\n", round(2.14));

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

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



수학 함수(function)

- ✓ 수학 관련 함수 – math.h를 include 해야 함

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

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

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

return 0;
}
```

```
3.0
2.0
2
2
3
3
8
8
16
1000
4
10
```



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
getchar(void)	<stdio.h>	문자 1개 입력
while(getchar() != '\n')		버퍼에서 ('\n')을 비움
fgets (char* Bufffer, int MaxCount, File* stream)	<stdio.h>	공백을 포함한 문자열 입력 가능
puts (char* Buffer)	<stdio.h>	문자열 출력[printf()와 유사함]
strcpy (char *string1, const char *string2)	<string.h>	string2 문자열을 string1로 복사
strlen (const char* Str)	<string.h>	저장된 문자열의 길이를 반환(개수)
strcmp (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

- 문자열과 숫자 입력 예제

```
char name[20];
int age;

puts("이름 입력: ");
scanf_s("%s", name, sizeof(name));
//공백을 포함한 이름 입력 가능
//fgets(name, sizeof(name), stdin);

printf("이름: %s\n", name);

while (getchar() != '\n');

puts("나이 입력: ");
scanf_s("%d", &age);

printf("나이: %d\n", age);
```

```
이름 입력 :
신 유 빈
이름 : 신
나이 입력 :
나이 : -858993460
```

```
이름 입력 :
신 유 빈
이름 : 유 빈

나이 입력 :
20
나이 : 20
```



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);
    printf("%d\n", len);

    //문자열의 복사
    //strcpy(msg2, msg1); //strcpy(복사본, 원본)
    //sizeof(msg2) - 버퍼 오버플로우 방지
    strcpy_s(msg2, sizeof(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);
}
```



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

```
#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

```
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 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()

