

C - 배열



Visual Studio 2022



배열(Array)

❖ 배열은 왜 써야 할까?, 사용의 필요성

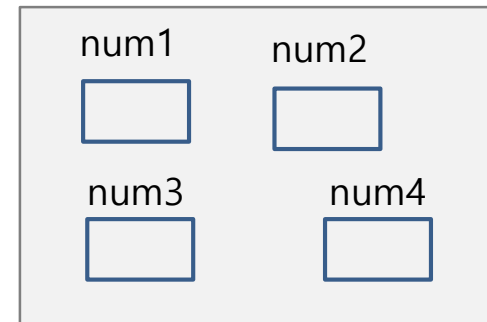
- 정수 10개를 이용한 프로그램을 할 때 10개의 정수 타입의 변수를 선언

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

정보가 흩어진 채 저장되므로

비효율적이고 관리하기 어렵다.

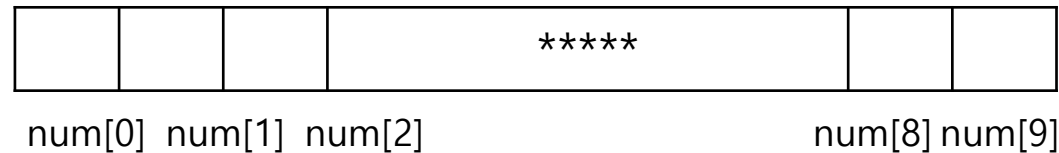
메모리



- 배열은 동일한 자료형의 변수를 한꺼번에 순차적으로 관리할 수 있다.

`int num[10];`

배열 이름
1개



배열(Array)

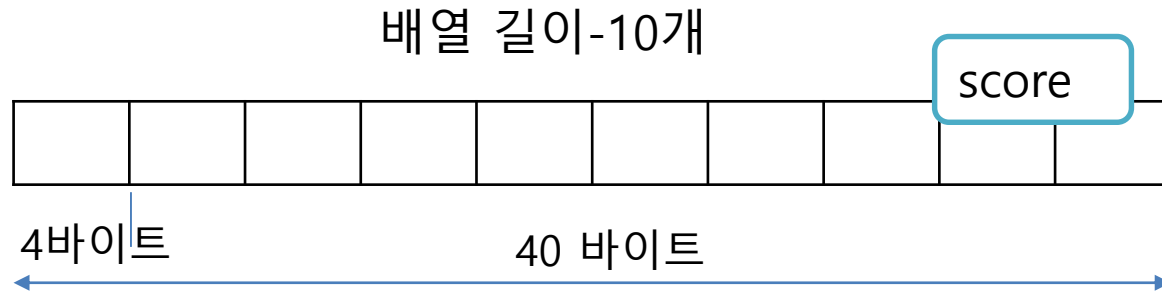
- 배열이란?

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 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

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

//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));
```

```
apple 6
apple 6
사과 5
```



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);
```



배열(Array)

- 회원 정보 입력 및 출력

```
char id[20], password[256], name[30];  
float weight, height;  
  
printf("\n==== 회원 정보 입력 =====\n");  
printf("아이디 입력: ");  
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);
```



배열(Array)

- 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)

- 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)

- 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)

- 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()
{
    //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)

- 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)

- 정수형 배열의 연산

```
int score[5] = { 85, 75, 90, 75, 80 };
int sum = 0;    //총점
double avg;     //평균
int min, max;   //최소, 최대

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

for (int i = 0; i < size; i++) {
    sum += score[i];
    printf("score[%d]=%d, sum=%d\n", i, score[i], sum);
}

avg = (double)sum / 4;
printf("합계: %d, 평균: %.11f\n", sum, avg);

//최소값
min = score[0]; //첫번째 요소를 최소값으로 설정
for (int i = 0; i < 4; i++) {
    if (score[i] < min) //요소값이 최소값보다 작으면
        min = score[i]; //요소값을 최소값에 저장
}
```

```
배열의 크기 : 5
score[0]=85, sum=85
score[1]=75, sum=160
score[2]=90, sum=250
score[3]=75, sum=325
score[4]=80, sum=405
합계 : 405, 평균 : 101.2
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
    continue; //while문 처음으로 돌아감
}
}
printf("최소값은 %d\n", min);
```



2차원 배열

배열의 확장 : 2차원 배열

이정후의 1반 학생들의 키를 배열에 저장

```
int class1[5]
```

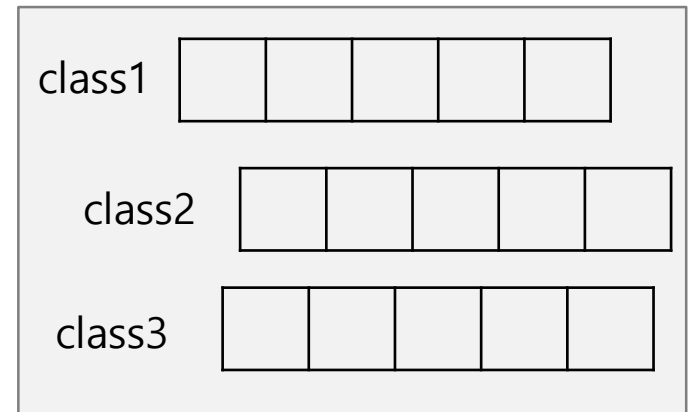
2반과 3반 학생들의 키를 배열에 저장

```
int class1[5]
```

```
int class2[5]
```

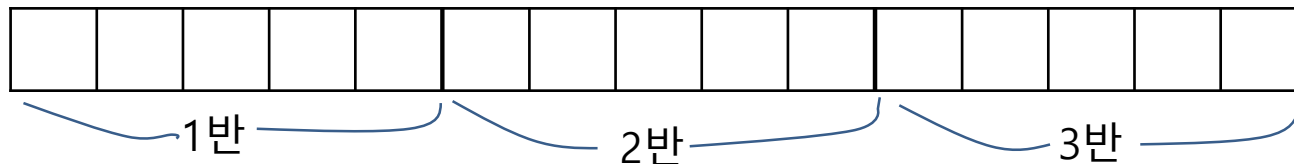
```
int class3[5]
```

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

```
//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

- 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]);
}
```



실습 문제 1 – 배열

- 실습 예제

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

👉 실행 결과

```
합 계 : 25  
평 균 : 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 – 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

- 아스키 코드값 출력

```
// 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);
}
```



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);
}
```



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;
}
```

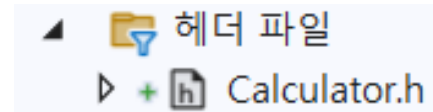
```
void call()
{
    //int x = 0; //지역변수
    static int x = 0; //정적 변수-전역변수화 함

    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);

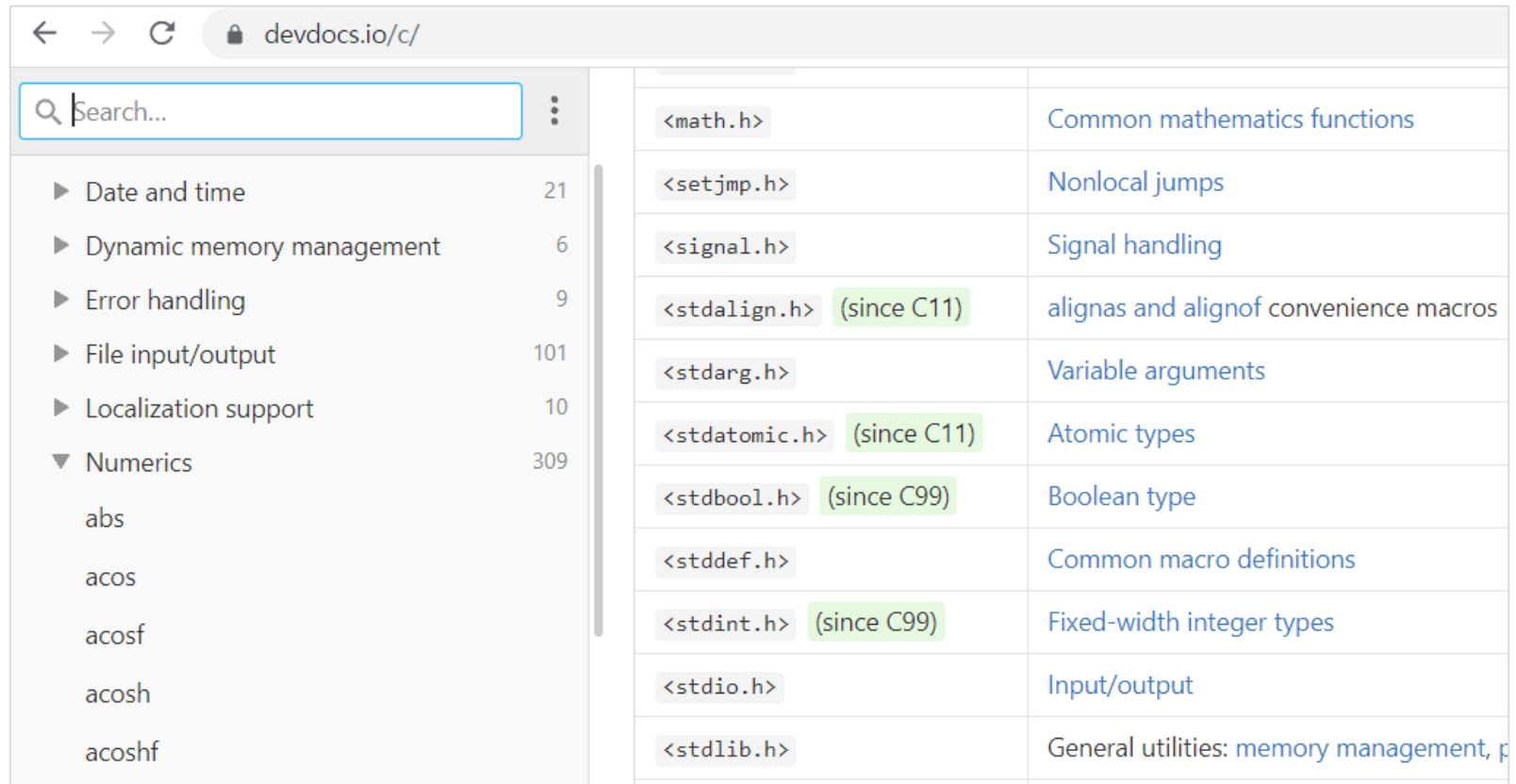
    return 0;
}
```



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>

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)

- ✓ 수행 시간 측정하기 - 소수로 출력

```
clock_t start, end;
double elapsed_time;

start = clock(); // 시작 시간 기록

for (int i = 0; i < 10; i++) {
    printf("%d\n", i);
    Sleep(500); // 0.5초 대기
}

end = clock(); // 종료 시간 기록

// 실행 시간 계산 (초 단위, 소수점 3자리까지 출력)
elapsed_time = (double)(end - start) / CLOCKS_PER_SEC;

printf("소요 시간: %.2f초\n", elapsed_time);
```



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

- 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

```
#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

```
//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
gets(char* Bufffer)	<stdio.h>	문자열 저장[scanf()와 유사함](사용안함)
puts(char* Buffer)	<stdio.h>	문자열 출력[printf()와 유사함]
fgets(char* Buffer, int MaxCount, FILE* stream)	<stdio.h>	공백을 포함한 문자열 입력 가능
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')) 문이
버퍼(임시기억장소에 남아 있던 '\n'을 비움

```
char c1, c2;  
  
c1 = getchar(); //예) 'a' 입력 받음(버퍼에는 '\n' 남아 있음)  
while (getchar() != '\n'); //'\n'이 비워짐  
c2 = getchar(); //예) 'b'를 입력 받음(버퍼에는 '\n' 남아 있음)  
  
printf("%c %c", c1, c2);
```



문자열 처리 함수

- 문자 1개 입력 및 버퍼 비우기 – getchar()

```
//이름과 나이 입력받기
char name[20];
int age;

printf("이름 입력 :");
scanf_s("%s", name, sizeof(name));

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

printf("나이 입력 :");
scanf_s("%d", &age);

printf("이름: %s, 나이: %d\n", name, age);
```



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));
```



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

```
#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

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

영어 타자 게임, 준비되면 엔터>

문제 1
monkey
monkey
통과!

문제 2
deer
deer
통과!

문제 3
deer
der
오타! 다시 도전!

문제 8
horse
horse
통과!

문제 9
tiger
tiger
통과!

문제 10
fox
fox
통과!
게임 소요 시간 : 20.9초



실습 – 영어 타이핑 게임

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

int main()
{
    char* words[] = { "ant", "bear", "chicken", "deer", "elephant", "fox",
                      "horse", "monkey", "lion", "tiger"};
    char* question; //문제
    char answer[20]; //사용자
    int n = 1; //문제 번호
    clock_t start, end;
    double elapsedTime; //게임 소요 시간

    srand(time(NULL)); //seed 설정
    int size = sizeof(words) / sizeof(words[0]);

    printf("영어 타자 게임, 준비되면 엔터>");
    getchar();

    start = clock(); //시작 시간
```



실습 – 영어 타이핑 게임

```
while (n <= 10)
{
    printf("\n문제 %d\n", n);
    int rndIdx = rand() % size;
    question = words[rndIdx];

    printf("%s\n", question); //문제 출제
    scanf_s("%s", answer, sizeof(answer)); //사용자 입력

    if (strcmp(question, answer) == 0)
    {
        printf("통과!\n");
        n++; //다음 문제
    }
    else
    {
        printf("오타! 다시 도전!\n");
    }
}

end = clock(); //종료 시간
elapsedTime = (double)(end - start) / CLOCKS_PER_SEC;
printf("게임 소요 시간: %.1f초\n", elapsedTime);
```

