

10. 함수

4주차

함수

- ◆ **함수의 필요성**
 - 매번 되풀이하는 일정한 작업을 반복적으로 처리하는 일이 흔히 발생
 - 모듈화 프로그래밍, 절차적 프로그래밍에 필수
- ◆ **함수(function)란**
 - 프로그램에서 **특정한 작업을 처리하도록 작성한 프로그램 단위**
 - 필요한 입력을 받아 원하는 기능을 수행한 후 결과를 반환(return)하는 프로그램 단위
 - 프로그램에서 원하는 특정한 작업을 수행하도록 설계된 독립된 프로그램 단위
- ◆ **C 프로그램이란**
 - 하나의 **main()** 함수와 다른 사용자 정의 함수로 구성되는 프로그램
- ◆ **함수의 구분**
 - 사용자 정의 함수(user defined function) : 필요에 의해서 개발자가 직접 개발하는 함수
 - 라이브러리 함수 (library function)(=표준 함수(standard function)) : printf()와 scanf()와 같이 이미 개발환경에 포함되어 있는 함수
- ◆ **함수 사용의 3요소**
 - **함수선언, 함수호출, 함수 정의**

함수의 정의

반환 형태
(return type)

함수 이름

형식매개 변수(들)

```
int FunctionName( char *ptr, int len )
```

몸체시작

몸체

몸체종료

```
{
```

```
}
```

함수원형(prototype)선언의 필요성

Add() 함수가, 호출한 위치보다 위에 있으므로 OK.

```
int Add(int a, int b)
{
    int result = a+b;
    return result;
}
```

```
int main()
{
    Add( 3, 4 );
    return 0;
}
```

‘함수원형선언’ 으로서 compiler에게 정보를 주기.

```
int Add( int a, int b );
```

```
int main()
{
    Add( 3, 4 );
    return 0;
}
```

Add() 함수가 호출한 위치보다 아래에 있으므로 compile 오류!

```
int Add(int a, int b)
{
    int result = a+b;
    return result;
}
```

함수원형 선언

반환 형태
(return type)

함수 이름

매개 변수 type
매개 변수 이름

```
int FunctionName( char *ptr, int len );
```

몸체 없이 ‘;’

함수 호출

(functionadd2.c)

```
#include <stdio.h>
int main(void) {
    int a = 3, b = 5;
    int add2(int a, int b);
    int sum = add2(a, b);
    printf("합: %d\n", sum);
    return 0;
}
int add2(int a, int b) {
    int sum = a + b;
    return (sum);
}
int findMin2(int x, int y) {
    int min = x < y ? x : y;
    return (min);
}
```

[결과]
합: 8

형식 매개변수 이름은 생략 가능

[호출 시 고려 사항]

- 반환 타입에 맞는 변수에 반환 값 저장
- 인자의 개수, 순서, 타입 맞추기

호출이 없으므로 이 함수
구현은 실행되지 않음

1~{입력받은수}까지 합을 구하는 함수 (getsum.c)

```
#define _CRT_SECURE_NO_WARNINGS
```

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

```
int getsum(int);
```

```
int main(void) {
```

```
    int max = 0;
```

```
    printf("1에서 n까지의 합을 구할 n을 입력하시오. >> ");
```

```
    scanf("%d", &max);
```

```
    printf("1에서 %d까지의 합: %d\n", max, getsum(max));
```

```
    return 0;
```

```
}
```

```
int getsum(int n) {
```

```
    int sum = 0;
```

```
    for (int i = 1; i <= n; i++)
```

```
        sum += i;
```

```
    return sum;
```

```
}
```

[결과]

1에서 n까지의 합을 구할 n을 입력하시오. >> 3

1에서 3까지의 합: 6

함수 원형선언, 호출, 정의

‘형식 매개변수’ 와 ‘실 매개변수’

형식 매개변수 :
함수 정의할 때 사용

```
int Add(int a, int b)
{
    int result = a+b;
    return result;
}
```

실 매개변수 :
함수 호출할 때 사용

```
int main()
{
    Add(3, 4);
    return 0;
}
```

값에 의한 호출(call by value) :
실인자의 값이 형식인자의 변수에 각각 복사된 후
함수가 실행

함수 구현과 호출

(functioncall.c)

```
#include <stdio.h>
int add2(int a, int b);
int findMax2(int, int);
void printMin(int, int);

int main(void) {
    int a = 3, b = 5;

    int max = findMax2(a, b);
    printf("최대: %d\n", max);
    printf("합: %d\n", add2(a, b));

    printMin(2, 5);
    return 0;
}
```

반환 값이 없는
함수호출

[결과]

최대: 5
합: 8
최소: 2

```
int add2(int a, int b) {
    int sum = a + b;
    return (sum);
}

int findMax2(int a, int b) {
    int max = a > b ? a : b;
    return max;
}

int findMin2(int x, int y) {
    int min = x < y ? x : y;
    return (min);
}

void printMin(int a, int b) {
    int min = a < b ? a : b;
    printf("최소: %d\n", min);
    return; //생략 가능
}
```

배열을 함수 매개변수로 사용

◆ 배열을 함수 매개변수로 전달하는 방법

- 함수의 매개변수 : **배열이름과 배열 원소수**를 전달
- 함수로 전달된 배열이름은 [] 형태를 갖더라도 무조건 포인터 변수가 되므로 sizeof 로 원소수를 알 수 없기 때문.

◆ 배열을 함수 매개변수로 전달할 때의 장점

- 한 번에 여러 개의 변수를 전달하는 효과
- Call by value의 한계 극복

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

```
int sumary(int *ary, int SIZE);
```

int sumaryf(int ary[], int SIZE);
와 같음.

```
int main(void) {  
    int point[] = { 95, 88, 76, 54, 85, 33, 65, 78, 99, 82 };  
    int *address = point;  
    int aryLength = sizeof(point) / sizeof(int);  
    int sum = 0;  
    for (int i = 0; i < aryLength; i++)  
        sum += *(point + i);  
    printf("메인에서 구한 합은 %d\n", sum);  
    address = point;  
    printf("합은 %d\n", sumary(point, aryLength));  
    printf("합은 %d\n", sumary(&point[0], aryLength));  
    printf("합은 %d\n", sumary(address, aryLength));  
    return 0;  
}
```

[결과]

메인에서 구한 합은 755
합은 755
합은 755

sum += *(point++); //오류
sum += *(address++); //가능

```
int sumary(int *ary, int SIZE) {  
    int sum = 0;  
    for (int i = 0; i < SIZE; i++) {  
        sum += *ary++;  
    }  
    return sum;  
}
```

sum += ary[i]; //가능
sum += *(ary + i); //가능
sum += *(ary++); //가능

함수에서 배열 인자 사 용하기 (arrayparam.c)

다차원 배열을 인자로 이용하는 경우

- ◆ 함수원형과 함수정의의 헤더에서
첫 번째 대괄호 내부를 제외한
다른 모든 크기는 반드시 상수로 기술해
야 함

이차원 배열을 함수로 전달 (1/2)

(twodarrayfunction.c)

```
#include <stdio.h>
double sum(double data[][3], int, int);
void printarray(double data[][3], int, int);
int main(void) {
    double x[][3] = { { 1, 2, 3 }, { 7, 8, 9 }, { 4, 5, 6 }, { 10, 11, 12 } };
    int rowsize = sizeof(x) / sizeof(x[0]);
    int colsize = sizeof(x[0]) / sizeof(x[0][0]);
    printf("2차원 배열의 자료값은 다음과 같습니다.\n");
    printarray(x, rowsize, colsize);
    printf("2차원 배열 원소합은 %.3lf 입니다.\n", sum(x, rowsize, colsize));
    return 0;
}
```

2차원 배열의 행 수와 열 수

[결과]

2차원 배열의 자료값은 다음과 같습니다.

1행원소: x[0][0] = 1.00 x[0][1] = 2.00 x[0][2] = 3.00

2행원소: x[1][0] = 7.00 x[1][1] = 8.00 x[1][2] = 9.00

3행원소: x[2][0] = 4.00 x[2][1] = 5.00 x[2][2] = 6.00

4행원소: x[3][0] = 10.00 x[3][1] = 11.00 x[3][2] =

12.00

2차원 배열 원소합은 78.000 입니다.

이차원 배열을 함수로 전달 (2/2)

(twodarrayfunction.c)

```
double sum(double(*data)[3], int rowsize, int colsize) {  
    double total = 0;  
    for (int i = 0; i < rowsize; i++)  
        for (int j = 0; j < colsize; j++)  
            total += data[i][j];  
    return total;  
}
```

배열값을 모두 더하는 함수.

```
double sum(double data[][3], int rowsize, int colsize)
```

```
void printarray(double(*data)[3], int rowsize, int colsize) {  
    for (int i = 0; i < rowsize; i++) {  
        printf("%d행원소: ", i + 1);  
        for (int j = 0; j < colsize; j++)  
            printf("x[%d][%d] = %5.2lf  ", i, j, data[i][j]);  
        printf("\n");  
    }  
    printf("\n");  
}
```

배열값을 모두 출력하는 함수.

```
void printarray(double data[][3], int rowsize, int colsize)
```

재귀 함수 사용 (factorial.c)

```
#include <stdio.h>
int factorial(int); //함수원형

int main(void) {
    for (int i = 1; i <= 10; i++)
        printf("%2d! = %d\n", i, factorial(i));
    return 0;
}

int factorial(int number) {
    if (number <= 1)
        return 1;
    else
        return (number * factorial(number - 1));
}
```

[결과]

1! = 1

2! = 2

3! = 6

4! = 24

5! = 120

6! = 720

7! = 5040

8! = 40320

9! = 362880

10! = 3628800

라이브러리 함수

(rand() 사용)

- ◆ 함수 rand()
 - 연속적인 난수(임의의 수)를 생성해 주는 함수
 - 임의의 수
 - 어느 수가 반환될 지 예측할 수 없으며 모든 수에 대해 선택될 확률이 동일하다는 의미
- ◆ 함수원형은 헤더파일 stdlib.h에 정의
 - 함수 rand()를 사용하려면...
#include <stdlib.h>
 - Visual C++에서는 함수 rand()
 - 0~ 32767(RAND_MAX=0x7fff)사이의 정수 중에서 임의로 수
- ◆ 매번 난수를 다르게 생성 하려면
 - 시드(seed)값(난수를 다르게 만들기 위해 처음에 지정하는 수)을 매번 다르게 줘야 함.
 - **srand(time(NULL)); 1회 호출 -> rand(); N회 호출**
- ◆ 매번 다른 seed 값을 지정하기 위해 함수 time() 이용
 - time(NULL) : 1970년 1월 1일 이후 현재까지 경과된 시간을 초 단위로 반환
 - #include <time.h>
- ◆ a에서 b까지의 난수를 발생시키는 방법
 - **rand() % (b - a + 1) + a**

난수 출력 (srand.c)

```
#include <stdlib.h> //rand(), srand()를 위한 헤더파일 포함
#include <time.h>   //time()을 위한 헤더파일 포함
#define MAX 100
```

```
int main(void) {
    long seconds = (long)time(NULL);
    srand(seconds);

    printf("1 ~ %5d 사이의 난수 5개:\n", MAX);
    for (int i = 0; i < 5; i++)
        printf("%5d ", rand() % MAX + 1);
    puts("");

    return 0;
}
```

[결과]

1 ~ 100 사이의 난수 5개:
92 19 38 67 30

수학 라이브러리 함수 (math.c)

```
#include <stdio.h>
#include <math.h> //수학 관련 다양한 함수헤더 포함 헤더파일
int main(void) {
    printf(" i   i제곱   i세제곱   제곱근(sqrt)\n");
    printf("-----\n");
    for (int i = 3; i < 7; i++)
        printf("%3d %7.1f %9.1f %9.1f\n", i, pow(i, 2), pow(i, 3), sqrt(i));
    printf("\n");
    printf("exp(1.0) == %5.2f, ", exp(1.0));
    printf("pow(2.72, 1.0) == %5.2f, ", pow(2.72, 1.0));
    printf("sqrt(49) == %5.2f\n", sqrt(49));
    printf("abs(-10) == %5d, ", abs(-10));
    printf("ceil(7.1) == %5.2f, ", ceil(7.1));
    printf("floor(6.9) == %5.2f\n", floor(6.9));
    return 0;
}
```

[결과]

i	i제곱	i세제곱	제곱근
3	9.0	27.0	1.732
4	16.0	64.0	2.000
5	25.0	125.0	2.236
6	36.0	216.0	2.449

exp(1.0) == 2.72, pow(2.72, 1.0) == 2.72

abs(-10) == 10, ceil(7.1) == 8.0

문자 라이브러리 함수 (char.c)

```

#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <ctype.h>
void print2char(char);
int main(void) {
    char ch;
    printf("알파벳(종료x) 또는 다른 문자 입력하세요.\n");
    do {
        printf("문자 입력 후 Enter: ");
        scanf("%c", &ch);
        getchar();          //enter 키 입력 받음
        if (isalpha(ch))
            print2char(ch);
        else
            printf("입력: %c\n", ch);
    } while (ch != 'x' && ch != 'X'); //입력이 x 또는 X이면 종료
    return 0;
}

```

[결과]

알파벳(종료x) 또는 다른 문자 입력하

문자 입력 후 Enter: A

입력: A, 변환: a

문자 입력 후 Enter: x

입력: x, 변환: X

문자 라이브러리 함수 (char.c)

```
void print2char(char ch) {  
    if (isupper(ch))  
        printf("입력: %c, 변환: %c\n", ch, tolower(ch));  
    else  
        printf("입력: %c, 변환: %c\n", ch, toupper(ch));  
    return;  
}
```

[결과]

알파벳(종료x) 또는 다른 문자 입력하세요.

문자 입력 후 Enter: A

입력: A, 변환: a

문자 입력 후 Enter: x

입력: x, 변환: X