

7장 함수

한림대학교 소프트웨어학부 양은샘.



7장 함수

- 안녕하세요? 여러분!
- 이번 장에서는 함수에 대해 학습합니다.
- C 언어는 함수의 조합으로 프로그래밍이 이루어 지기 때문에 반드시 함수의 호출과 반환을 이해해야 합니다.
- 지난 시간에 학습한 내용을 리뷰한 후 학습을 시작하도록 하겠습니다.

지난 시간 Review

6.1 포인터의 의미와 사용

6.2 포인터 연산

6.3 배열과 포인터

6.4 문자열 상수 포인터

6.5 이차원 배열과 포인터

6.6 포인터 배열

6.7 다중 포인터

6.8 포인터 응용

☐ 개념 확인 학습

☐ 적용 확인 학습

☐ 응용 프로그래밍

7장 함수

7.1 함수의 정의

7.2 표준 라이브러리 함수

7.3 함수의 호출 및 복귀

7.4 매개변수의 전달 방식

7.5 함수와 변수영역

7.6 순환 호출(Recursive Call)

7.7 함수와 배열

7.8 main() 함수의 매개변수

7.9 함수 포인터

7.10 배열 포인터 함수 응용

☐ 개념 확인 학습

☐ 적용 확인 학습

☐ 응용 프로그래밍

학습 목표

- 함수의 호출 및 복귀를 이해한다.
 - 함수에서 배열을 이용할 수 있다.
 - 함수에서 포인터를 이용할 수 있다.
 - 함수에서 배열과 포인터를 응용하여 다양한 조건의 문제들을 해결 할 수 있다.
-
- 개념 확인 학습으로 배운 내용을 정리한다.
 - 적용 확인 학습으로 개념 습득 여부를 확인한다.
 - 응용 프로그래밍으로 문제해결력을 키운다.

함수의 정의

□ 함수의 정의 방법

```
자료형 함수명(매개변수1, 매개변수2, ...)
{
    //매개변수는 함수의 호출에서 전달되어온 인자
    프로그램 내용;
    return 자료형에 해당하는 반환 값 ;
}
```

□ 함수의 원형

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

□ 함수의 정의 예

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

⇒ return 형식

return;	//반환 값없이 복귀
return (식);	//식 : 상수, 변수, 수식

⇒ return 사용 예

return;	//반환 값이 없는 경우
return 0; return 1;	//반환 값이 0, 1의 값인 경우
return a; return (b);	//반환 값이 변수인 경우
return a+b; return(a+b);	//반환 값이 연산 결과인 경우

함수의 호출 및 복귀

[예제 7.1] 함수의 호출

```
#include <stdio.h>

void output1()
{
    printf("-- Function output1 --\n");
}

void output2()
{
    printf("-- Function output2 --\n");
}

int main()
{
    printf("-- Start main() --\n");
    output1();
    output2();
    printf("-- End main() --\n");
    return 0;
}
```

```
-- Start main() --
-- Function output1 --
-- Function output2 --
-- End main() --
```

[예제 7.2] 함수의 호출 및 복귀

```
#include <stdio.h>
int add(int i, int j); //사용자 정의 함수의 원형
int prodt(int i, int j);
```

정수 두 개 입력 : 25 20
25+20=45, 25*20=500

```
int add(int i, int j)
{
    int result = i + j;
    return result;
}

int prodt(int i, int j)
{
    return i * j;
}

int main()
{
    int a, b, sum=0, mul=0;

    printf("정수 두 개 입력 : ");
    scanf("%d %d", &a, &b);

    sum = add(a, b);
    mul = prodt(a, b);

    printf("%d+%d=%d, %d*%d=%d\n", a, b, sum, a, b, mul);
    return 0;
}
```

문자 관련 함수

[표 7.1] 문자 관련 함수

함수의 결과	함수명	설명
참(1) 거짓(0)	isalpha()	알파벳인지 검사.
	isdigit()	숫자인지 검사.
	isspace()	공백문자인지 검사
	islower()	소문자인지 검사
	isupper()	대문자인지 검사
	isctrl()	제어 문자인지 검사

문자열 관련 함수 (1)

함수 원형 / 함수 기능

`int strlen(char *str1);`

: str1에 포함된 문자열 중 널(NULL)을 제외한 문자의 개수를 반환한다.

`int strcmp(char *str1, char *str2);`

: 두 문자열을 대소문자를 구별하여 사전적으로 비교한다.

: 두 문자열이 같으면 0을 반환한다.

`char *strcpy(char *str1, char *str2);`

: str2가 가리키는 문자열을 str1으로 복사한다.

: str2는 str1과 같은 내용으로 변경되고, str2는 변경이 없다.

`char *strncpy(char *str1, char *str2, int n);`

: str2가 가리키는 문자열에서부터 숫자(n) bytes만큼의 문자열을 str1이 가리키고 있는 위치에 복사한다.

문자열 관련 함수 (2)

함수 원형 / 함수 기능

`int atoi(char *str);`

: 문자열을 정수로 변환한다.

`long atol(char *str)`

: 문자열을 long형 정수로 변환한다.

`double atof(char *str);`

: 문자열을 실수로 변환한다.

`char * strtok(char *str, const char * delim)`

: str문자열을 delim 문자열의 구성성분을 기준으로 토큰을 추출한다.

`char *pch, str[]="a.b c";`

`pch = strtok(str, " .");`

`while(pch != NULL) { pch = strtok(NULL, " ."); }`

`char * strstr(char *str1, const char *str2);`

: str1에서 str2 문자열을 찾아 그 위치를 반환한다.

함수의 매개변수 전달 방식

[예제 7.3] 함수의 매개 변수 전달 (값에 의한 호출)

```
#include <stdio.h>

void call_val(int a, int b)
{
    int temp = a;
    a = b;
    b = temp;
    printf("call_val() : a=%d, b=%d\n", a, b);
}

int main()
{
    int a, b;

    printf("정수 두 개 입력 : ");
    scanf("%d %d", &a, &b);

    printf("before call_val() : a=%d, b=%d\n", a, b);
    call_val(a, b);
    printf("after call_val() : a=%d, b=%d\n", a, b);
    return 0;
}
```

```
정수 두 개 입력 : 47 23
before call_val() : a=47, b=23
call_val() : a=23, b=47
after call_val() : a=47, b=23
```

[예제 7.4] 함수의 매개 변수 전달 (참조에 의한 호출)

```
#include <stdio.h>

void call_ref(int *a, int *b)
{
    int temp = *a;
    *a = *b;
    *b = temp;
    printf("call_ref() : a=%d, b=%d\n", *a, *b);
}

int main()
{
    int a, b;

    printf("정수 두 개 입력 : ");
    scanf("%d %d", &a, &b);

    printf("before call_ref() : a=%d, b=%d\n", a, b);
    call_ref(&a, &b);
    printf("after call_ref() : a=%d, b=%d\n", a, b);
    return 0;
}
```

```
정수 두 개 입력 : 47 23
before call_ref() : a=47, b=23
call_ref() : a=23, b=47
after call_ref() : a=23, b=47
```

지역 변수와 전역 변수

- 지역 변수

- 함수 내부에서 선언되어 그 함수의 내부에서만 유효
- 서로 다른 함수에서 같은 이름의 변수를 사용해도 별개의 변수로 인식.
- 초기화 필요.

- 전역 변수

- 함수 외부에서 선언된 변수
- 메인 메모리의 데이터 영역에 생성
- 변수가 선언된 이후의 모든 함수에서 참조 가능
- 초기 값 : 정수형 데이터는 0, 문자형 데이터는 널(' 0')로 자동 초기화.
- 동일한 이름의 전역 변수와 지역 변수가 선언되어 사용되고 있다면, 지역 변수 우선 처리.

[예제 7.5] 전역 변수의 활용

```
#include <stdio.h>

int x = 5; //전역변수
void func()
{
    int x = 10;
    x = x + 20;
    printf("func() : x = %dWn", x);
}

int main()
{
    x = x + 10;
    printf("Before func() : x = %dWn", x);
    func();
    printf("After func() : x = %dWn", x);
    return 0;
}
```

Before func() : x = 15
func() : x = 30
After func() : x = 15

변수의 기억 클래스

[표 7.3] 기억클래스의 종류

기억클래스	기억장소	생존기간	참조범위
자동(auto) 변수	stack(스택)	일시적	<ul style="list-style-type: none">- 일반적인 변수 선언 방식.- 변수가 선언된 블록 내에서만 참조 가능.
정적(static) 변수	일반적인 메모리	영구적	<ul style="list-style-type: none">static 데이터형 변수명;- 변수가 선언된 블록에서 참조 가능.- 블록이나 함수가 종료되어도 프로그램이 종료되기 전까지 값을 유지.- 정수형은 0, 문자형은 널('W0')로 자동 초기화.
레지스터 변수	CPU의 레지스터	일시적	<ul style="list-style-type: none">register 데이터형 변수명 ;- 속도는 가장 빠름.- 선언된 블록 내에서만 사용 가능(지역변수)
외부(extern) 변수	일반적인 메모리	영구적	<ul style="list-style-type: none">extern 데이터형 변수명- 함수의 외부 또는 외부 파일에서 선언된 전역 변수를 참조하고자 할 때 사용.- extern 지정자는 선언된 변수가 외부 변수임을 컴파일러에게 알려주는 역할.

변수의 기억 클래스 예

[예제 7.6] 정적(static) 변수의 활용

```
#include <stdio.h>

void func()
{
    static int s = 0;
    int a = 0;

    s = s + 2;
    a = a + 2;
    printf("s=%d, a=%d\n", s, a);
}

int main()
{
    for (int i = 0; i < 3; i++) {
        printf("%d ", i);
        func();
    }
    return 0;
}
```

0) s=2, a=2
1) s=4, a=2
2) s=6, a=2

[예제 7.7] 외부(extern) 변수의 활용

//ex7_7_1.c

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

```
int x;
```

```
void sub1();
```

```
void sub2();
```

```
void sub3();
```

```
void sub1() {
    x = x + 100;
    printf("1) x = %d\n", x);
}
```

```
main() {
    x = 100;
    printf("0) x = %d\n", x);
    sub1();      sub2();  sub3();
}
```

//ex7_7_2.c

//ex7_7_1.c의 전역 변수 x를 사용하기 위한
외부 변수 선언

```
extern int x;
```

```
sub2() {
    int x;      //지역 변수
    x = 9999;
    printf("2) x = %d\n", x);
}
```

```
sub3() {
    x = x + 200;
    printf("3) x = %d\n", x);
}
```

순환 함수에서 스택의 사용

[예제 7.8] 순환 함수에서 스택의 사용

9 8 7 6 5 4 3 2 1 0

```
#include <stdio.h>

void recurse(int i)
{
    if(i<10) {
        recurse(i+1);
        printf("%d ", i);
    }
}

int main()
{
    recurse(0);
    return 0;
}
```

0 1 2 3 4 5 6 7 8 9

```
#include <stdio.h>

void recurse(int i)
{
    if(i<10) {
        printf("%d ", i);
        recurse(i+1);
    }
}

int main()
{
    recurse(0);
    return 0;
}
```

상호 순환 예

[예제 7.9] 상호 순환

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

```
void f1(int b);
```

```
void f2(int b);
```

```
void f2(int b)
```

```
{
```

```
    if(b) f1(b - 1);
```

```
    printf(". ");
```

```
}
```

```
void f1(int a)
```

```
{
```

```
    if(a) f2(a - 1);
```

```
    printf("%d ", a);
```

```
}
```

```
int main()
```

```
{
```

```
    f1(10);
```

```
    return 0;
```

```
}
```

0 . 2 . 4 . 6 . 8 . 10

함수와 배열

함수의 원형 예	함수의 호출 예
<code>void func(int *n);</code>	<code>int num=2;</code> <code>func(&num);</code>
<code>void func(char *s);</code>	<code>char ch = 'a' ;</code> <code>func(&ch);</code>
<code>void func(int n[], int size);</code> <code>void func(int *n, int size);</code>	<code>int nums[5] = {5, 4, 3, 2, 1};</code> <code>func(nums, sizeof(nums)/sizeof(int));</code> //정수형 배열은 크기를 알 수 없기 때문에 배열의 크기를 전달해야 한다.
<code>void func(char s[]);</code> <code>void func(char *s);</code>	<code>char str[] = "abc" ;</code> <code>func(str);</code> //문자열은 마지막이 항상 널이기 때문에 크기를 전달할 필요가 없다.
<code>void func(int (*n)[5], int size);</code> //size는 배열의 행의 크기	<code>int nums[2][5] = {5, 4, 3, 2, 1,</code> <code>1, 2, 3, 4, 5};</code> <code>func(nums, sizeof(nums)/sizeof(nums[0]));</code>
<code>void func(char (*s)[10], int size);</code>	<code>char strs[2][10] = {"apple", "banana"};</code> <code>func(strs, sizeof(strs)/sizeof(strs[1]));</code> //문자열 배열의 행의 크기는 전달해야 한다.

함수와 정수형 배열

[예제 7.10] 함수와 1차원 정수형 배열

```
#include <stdio.h>
void f1(int n[], int size) {
    for (int i = 0; i < size; i++)
        printf("%d ", n[i]);
}
void f2(int *n, int size) {
    for (int i = 0; i < size; i++)
        printf("%d ", n[i]);
}
int main()
{
    int nums[5] = {5, 4, 3, 2, 1};
    f1(nums, sizeof(nums) / sizeof(int));
    puts("");
    f2(nums, sizeof(nums) / sizeof(int));
    return 0;
}
```

5	4	3	2	1
5	4	3	2	1

[예제 7.11] 함수와 2차원 정수형 배열

```
#include <stdio.h>

void func(int (*n)[5], int size)
{
    for(int i=0; i<size; i++) {
        for(int j=0; j<sizeof(n[0])/sizeof(int); j++) {
            printf("%d ", n[i][j]);
        }
        puts("");
    }
}

int main()
{
    int nums[2][5] = {5, 4, 3, 2, 1, 1, 2, 3, 4, 5};
    func(nums, sizeof(nums)/sizeof(nums[0]));
    return 0;
}
```

5	4	3	2	1
1	2	3	4	5

함수와 문자열형 배열

[예제 7.12] 사용자 함수와 1차원 문자열형 배열

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

int mycmp(char *sa, char *sb)
{
    return strcmp(sa, sb);
}

int main()
{
    char stra[] = "apple";
    char strb[] = "apple";

    int result = mycmp(stra, strb);
    if (result)
        printf("다른 문자열 입니다.\n");
    else
        printf("같은 문자열 입니다.\n");
    return 0;
}
```

같은 문자열 입니다.

[예제 7.13] 사용자 함수와 2차원 문자열형 배열

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

int mycmp(char (*s)[10], int size)
{
    return strcmp(s[0], s[1]);
}

int main()
{
    char str[2][10] = {"apple", "apple"};

    int result = mycmp(str, sizeof(str)/sizeof(str[1]));
    if (result)
        printf("다른 문자열 입니다.\n");
    else
        printf("같은 문자열 입니다.\n");
    return 0;
}
```

같은 문자열 입니다.

main() 함수의 매개변수

□ main() 함수의 매개변수(명령어 라인 매개변수)

```
main(int argc, char *argv[ ])
```

- argc 는 명령어 라인에 있는 인자들의 개수를 의미한다.
- argv 는 포인터배열로 명령어 라인 매개변수들이 문자열로 저장된다.
- argv[0]에는 실행파일의 경로와 이름이 전달된다.
- argv[1], argv[2], ..., argv[n]에는 명령어 라인의 인자들이 전달된다.
- 명령어 라인에서 두 개 이상의 인자가 있는 경우에 구분자는 공백을 사용한다.

□ main() 함수의 매개변수(명령어 라인 매개변수)의 사용 예

```
c:>ctest 30 rain
```

- argc = 3

	0	1	2	3	4	5	6	7	8	9	10	11	12
argv[0]	c	:	W	c	t	e	s	t	.	e	x	e	W0
argv[1]	3	0	W0										
argv[2]	r	a	i	n	W0								

main() 함수의 매개변수 사용법

[예제 7.14] main() 함수의 매개변수 사용 방법

```
#include <stdio.h>

int main(int argc, char *argv[])
{
    printf("argc = %d\n", argc);

    for (int i = 0; i < argc; i++) {
        printf("argv[%d] = %s\n",
            i, argv[i]);
    }

    return 0;
}
```



The screenshot shows a Windows command prompt window titled "명령 프롬프트". The user navigates to the directory C:\Users\hallym>cd C:\lecture_src and then runs the program ctest. The program output shows argc = 4 and the arguments argv[0] = ctest, argv[1] = red, argv[2] = blue, and argv[3] = yellow. The command prompt also shows the directory listing for C:\lecture_src\ctest, which includes files ctest.c (191 bytes) and ctest.exe (44,877 bytes), and a directory .vscode.

```
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Users\hallym>cd C:\lecture_src
C:\lecture_src>cd ctest
C:\lecture_src\ctest>dir
C 드라이브의 볼륨에는 이름이 없습니다.
볼륨 일련 번호: 7ED5-ADE6

C:\lecture_src\ctest 디렉터리
2020-08-04 오후 08:39 <DIR> .
2020-08-04 오후 08:39 <DIR> ..
2020-07-28 오후 06:27 <DIR> .vscode
2020-08-04 오후 08:28 191 ctest.c
2020-08-04 오후 08:39 44,877 ctest.exe
2개 파일 45,068 바이트
3개 디렉터리 46,736,056,320 바이트

C:\lecture_src\ctest>ctest red blue yellow
argc = 4
argv[0] = ctest
argv[1] = red
argv[2] = blue
argv[3] = yellow

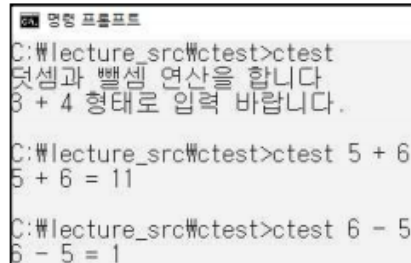
C:\lecture_src\ctest>
```

main() 함수의 매개변수 사용법

```
#include <stdio.h>
#include <stdlib.h>
int main(int argc, char *argv[])
{
    double x;
    int y;
    char op;

    if (argc != 4) {
        printf("덧셈과 뺄셈 연산을 합니다\n");
        printf("3 + 4 형태로 입력 바랍니다.\n");
        exit(0);
    }
    x = atof(argv[1]); //문자열을 실수로 변환
    op = *argv[2];
    y = atoi(argv[3]); //문자열을 정수로 변환

    switch (op) {
        case '+':
            printf("%d %c %d = %d\n", x, op, y, x + y);
            break;
        case '-':
            printf("%d %c %d = %d\n", x, op, y, x - y);
            break;
        default:
            printf("계산할 수 없습니다.\n");
    }
    return 0;
}
```



명령 프롬프트

```
C:\lecture_src\ctest>ctest
덧셈과 뺄셈 연산을 합니다
3 + 4 형태로 입력 바랍니다.

C:\lecture_src\ctest>ctest 5 + 6
5 + 6 = 11

C:\lecture_src\ctest>ctest 6 - 5
6 - 5 = 1
```

함수 포인터

□ 함수 포인터 선언 방법

반환자료형 (*함수포인터명) (자료형1, 자료형2,);

또는

반환자료형 (*함수포인터명) (자료형1 매개변수명1, 자료형2 매개변수명2,);

□ 함수 포인터 배열 선언 방법

반환자료형 (*함수포인터저장배열명[배열크기]) (자료형1, 자료형2,);

또는

반환자료형 (*함수포인터저장배열명[배열크기]) (자료형1 매개변수명1, 자료형2 매개변수명2,);

□ 함수 포인터 / 함수 포인터 배열 사용 예

```
void add(int *, int, int);
```

```
void sub(int *, int, int);
```

```
void (*pf)(int *, int, int);
```

```
pf = add;
```

```
pf = sub;
```

```
void (*pfa)(int *, int, int) = add; //pfa = sub;
```

```
void (*pfs)(int *, int, int) = sub; //pfs = add;
```

```
void (*pfarr[2])(int *, int, int);
```

```
pfarr[0] = add;
```

```
pfarr[1] = sub;
```

함수 포인터 예

[예제 7.16] 함수와 함수 포인터

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

```
void add(int *r, int a, int b) {  
    *r = a + b;  
}
```

```
void sub(int *r, int a, int b) {  
    *r = a - b;  
}
```

```
int main(int argc, char *argv[])  
{
```

```
    int ina, inb, rst;  
    void(*pf)(int *, int, int) = NULL;  
    void(*pfarr[2])(int *, int, int) = { add, sub };
```

```
    printf("정수 2개 입력 : ");  
    scanf("%d %d", &ina, &inb);
```

정수 2개 입력 : 33 55
함수 포인터 pf = add의 결과 (88)
함수 포인터 pf = sub의 결과 (-22)
함수 포인터 배열 pfarr의 결과 pfarr[0] = 88
함수 포인터 배열 pfarr의 결과 pfarr[1] = -22

```
    pf = add;  
    pf(&rst, ina, inb);  
    printf("함수 포인터 pf = add의 결과 (%d)\n", rst);
```

```
    pf = sub;  
    pf(&rst, ina, inb);  
    printf("함수 포인터 pf = sub의 결과 (%d)\n", rst);
```

```
    for (int i = 0; i < 2; i++) {  
        pfarr[i](&rst, ina, inb);  
        printf("함수 포인터 배열 pfarr의 결과 pfarr[%d] = %d\n", i,
```

```
rst);
```

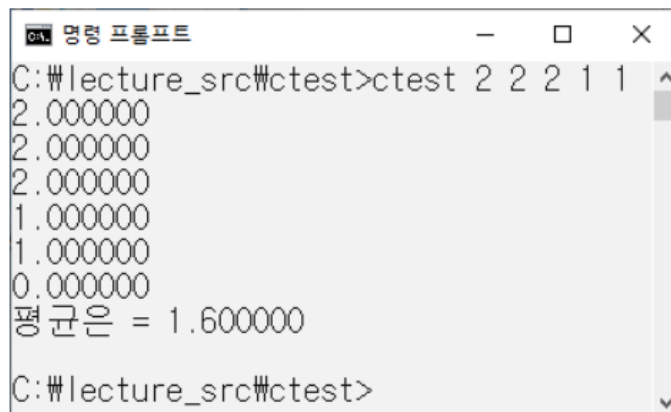
```
    }  
    return 0;
```

```
}
```


배열 포인터 함수 응용 (1)

[예제 7.17] main() 함수의 매개변수와 사용자 함수 (실수형)

```
#include <stdio.h>
#include <stdlib.h>
double avg(double *n, int size)
{
    double sum = 0;
    for (int i = 0; i < size; i++) {
        sum += *(n + i);
    }
    return sum / size;
}
```



```
C:\lecture_src\ctest>ctest 2 2 2 1 1
2.000000
2.000000
2.000000
1.000000
1.000000
0.000000
평균은 = 1.600000
C:\lecture_src\ctest>
```

```
int main(int argc, char *argv[])
{
    double avg_value, num[5];
    for (int i = 0; i < argc; i++) {
        num[i] = atof(argv[i + 1]);
        printf("%lf\\n", num[i]);
    }
    avg_value = avg(num, sizeof(num) / sizeof(double));
    printf("평균은 = %lf\\n", avg_value);
    return 0;
}
```

배열 포인터 함수 응용 (2)

[예제 7.18] main() 함수의 매개변수와 사용자 함수 (문자열형)

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

int has(char (*s)[10], int size, char *in)
{
    int has_flag = 0;
    for (int i = 0; i < size; i++) {
        if (strcmp(s[i], in) == 0)
            has_flag = 1;
    }
    return has_flag;
}
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

PS C:\lecture_src\ctest> cd "c:\lecture_
ctest"
입력이 바르지 않아 종료합니다.
(입력 예) ctest color
PS C:\lecture_src\ctest> .\ctest blue
입력하신 색이 존재하지 않습니다.
PS C:\lecture_src\ctest> .\ctest yellow
입력하신 색이 존재합니다.
PS C:\lecture_src\ctest>
```

```
int main(int argc, char *argv[])
{
    char strs[][10] = {"red", "green", "yellow"};

    if (argc != 2) {
        printf("입력이 바르지 않아 종료합니다.\n");
        printf("(입력 예) ctest color\n");
        exit(0);
    }
    if (has(strs, sizeof(strs)/sizeof(strs[1]), argv[1]))
        printf("입력하신 색이 존재합니다.\n");
    else
        printf("입력하신 색이 존재하지 않습니다.\n");
    return 0;
}
```

개념 확인학습 & 적용 확인학습 & 응용 프로그래밍

- 다음 파일에 있는 문제들의 해답을 스스로 작성 해 보세요.
 - c_07장_함수_ex.pdf
- 퀴즈와 과제가 출제되었다면 마감시간에 늦지 않도록 주의해 주세요.

Q & A

- “함수”에 대한 학습이 모두 끝났습니다.
- 모든 내용을 이해 하셨나요?
- 아직 이해가 안되는 내용이 있다면 다시 한번 복습하시기 바랍니다.
- 질문은 한림 SmartLEAD 쪽지 또는 e-mail 또는 전화상담을 이용하시기 바랍니다.



- 퀴즈와 과제가 출제되었다면 마감시간에 늦지 않도록 주의해 주세요.
- 다음 시간에는 “구조체”에 대해 알아보겠습니다.
- 수고하셨습니다.^^