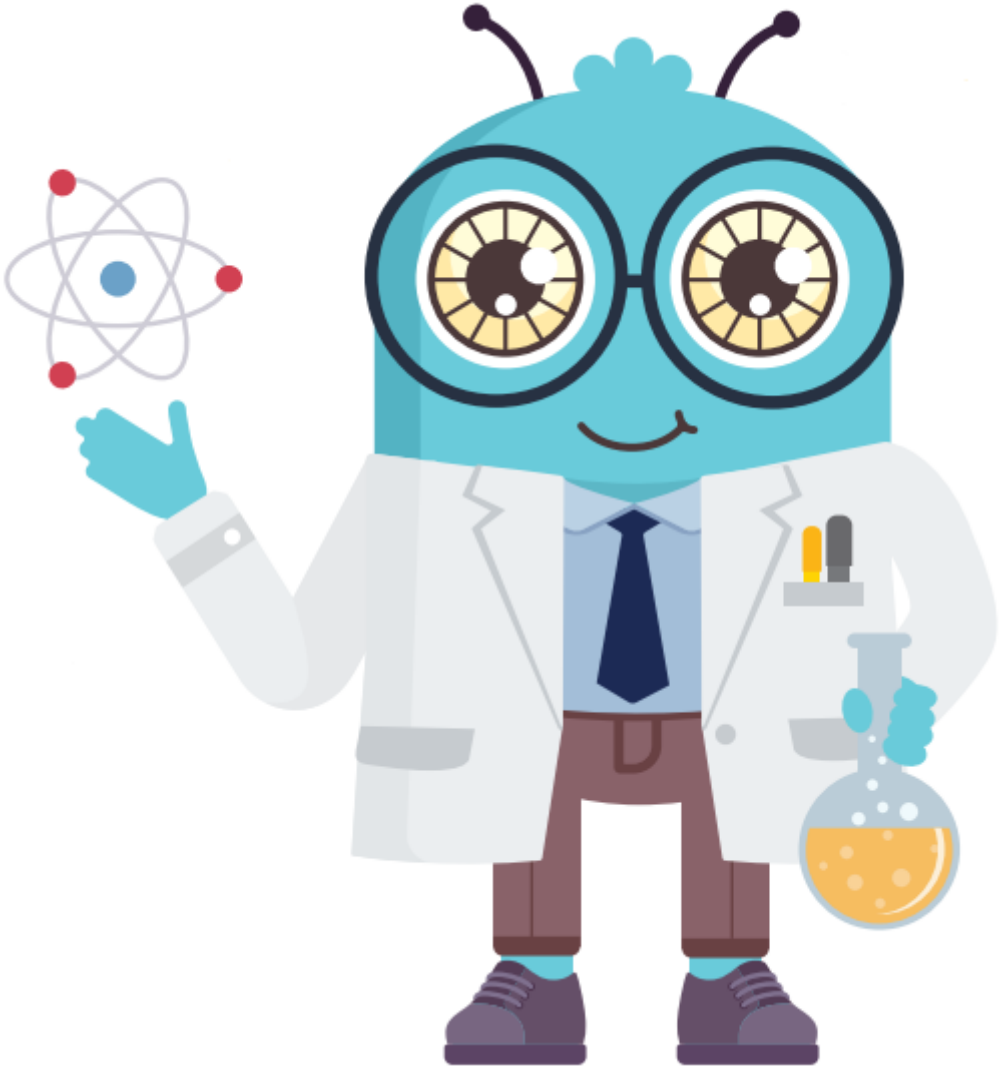

*Chapter 10*

# 함수

# 목차

# 01
# 함수의 이해

# 1. 함수의 이해

## 1. 함수의 개념

- 함수(Function) : “무엇을 넣으면, 어떤 것을 돌려주는 요술상자”
- C 언어 프로그램 자체에서 제공, 직접 만들어서 사용 가능

```
함수_이름( );
```

- printf( ) 함수 : C 언어에서 자체 제공

```
printf("Basic-C");
```

➔ 출력 – Basic-C

## 1. 함수의 개념

- **함수를 자판기에 비교**
  - 자판기가 없는 경우 → 매번 같은 작업을 반복

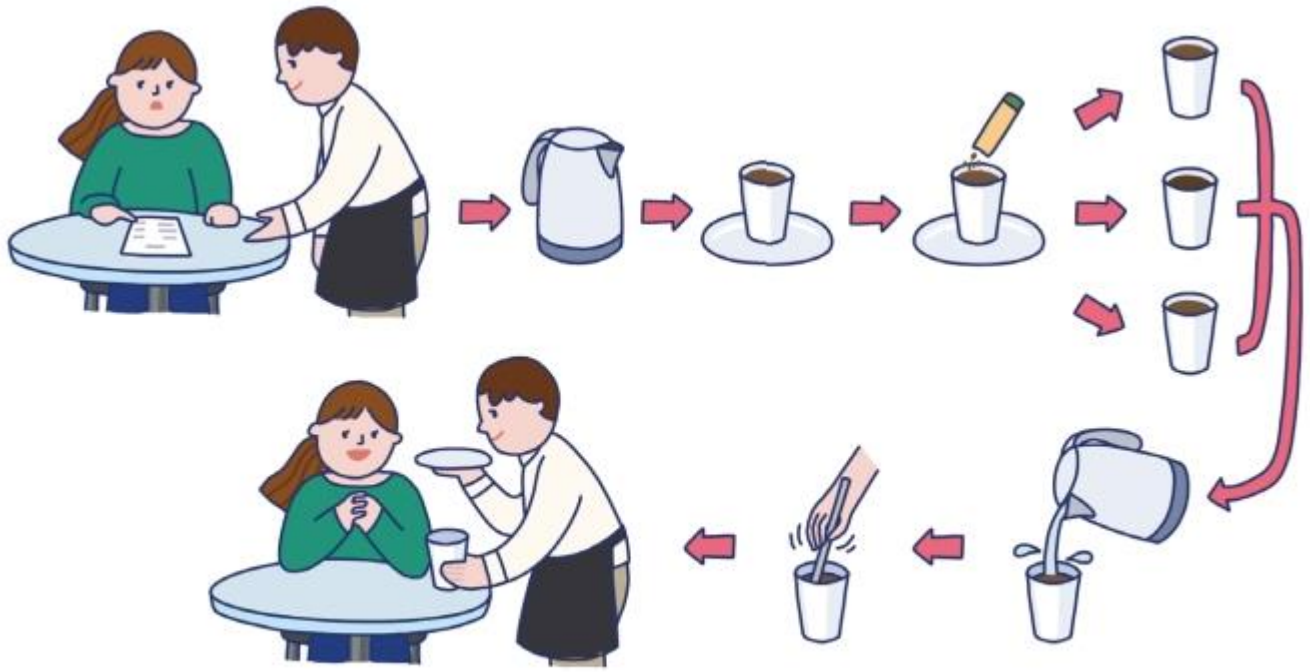

그림 10-1 직접 커피를 서비스하는 과정

# 1. 함수의 이해

## 1. 함수의 개념

**기본 10-1** 직접 커피를 서비스하는 과정의 예 10-1.c

```
01 #define _CRT_SECURE_NO_WARNINGS
02 #include <stdio.h>
03 void main( )
04 {
05    int coffee;                          ----- 커피의 종류를 선택하는 변수이다.
06
07    printf("어떤 커피 드릴까요? (1:보통, 2:설탕, 3:블랙) ");
08    scanf("%d", &coffee);                ----- 커피의 종류를 입력받는다.
09
10    printf("\n# 1. 뜨거운 물을 준비한다\n");
11    printf("# 2. 종이컵을 준비한다\n");
12
13    switch(coffee)                       ----- 커피의 종류에 따라 안내문을
14    {                                          출력한다.
15    case 1: printf("# 3. 보통커피를 탄다\n"); break;
16    case 2: printf("# 3. 설탕커피를 탄다\n"); break;
17    case 3: printf("# 3. 블랙커피를 탄다\n"); break;
18    default: printf("# 3. 아무거나 탄다\n"); break;
19    }
```

# 1. 함수의 이해

## 1. 함수의 개념

```
20
21      printf("# 4. 물을 붓는다\n");
22      printf("# 5. 스푼으로 저어서 녹인다\n\n");
23
24      printf("손님~ 커피 여기 있습니다.\n\n");
25  }
```

실행 결과

```
어떤 커피 드릴까요? (1:보통, 2:설탕, 3:블랙) 2

# 1. 뜨거운 물을 준비한다
# 2. 종이컵을 준비한다
# 3. 설탕커피를 탄다
# 4. 물을 붓는다
# 5. 스푼으로 저어서 녹인다

손님~ 커피 여기 있습니다.
```

# 1. 함수의 이해

## 1. 함수의 개념

- 손님 3명이 연속해서 온다고 가정

```
⋮
07    printf("어떤 커피 드릴까요? (1:보통, 2:설탕, 3:블랙) ");
08    scanf("%d", &coffee);
09
10    printf("\n# 1. 뜨거운 물을 준비한다\n");
11    printf("# 2. 종이컵을 준비한다\n");
12
13    switch(coffee)
14    {
15    case 1: printf("# 3. 보통커피를 탄다\n"); break;
16    case 2: printf("# 3. 설탕커피를 탄다\n"); break;
17    case 3: printf("# 3. 블랙커피를 탄다\n"); break;
18    default: printf("# 3. 아무거나 탄다\n"); break;
19    }
20
```

# 1. 함수의 이해

## 1. 함수의 개념

```
21    printf("# 4. 물을 붓는다\n");
22    printf("# 5. 스푼으로 저어서 녹인다\n\n");
23
24    printf("손님~ 커피 여기 있습니다.\n\n");
25
26 [두 번째 손님을 위해 07~24행을 다시 반복한다.]
27
28 [세 번째 손님을 위해 07~24행을 다시 반복한다.]
```

그림 10-2 커피 자판기를 사용하는 과정

## 1. 함수의 개념

**기본 10-2** 함수를 사용하여 변경한 [기본 10-1] 10-2.c

```
01 #define _CRT_SECURE_NO_WARNINGS
02 #include <stdio.h>
03 int coffee_machine(int button)
04 {
05    printf("\n# 1.(자동으로) 뜨거운 물을 준비한다\n");
06    printf("# 2. (자동으로) 종이컵을 준비한다\n");
07
08    switch(button)
09    {
10    case 1: printf("# 3. (자동으로) 보통커피를 탄다\n"); break;
11    case 2: printf("# 3. (자동으로) 설탕커피를 탄다\n"); break;
12    case 3: printf("# 3. (자동으로) 블랙커피를 탄다\n"); break;
13    default: printf("# 3. (자동으로) 아무거나 탄다\n"); break;
14    }
15
16    printf("# 4. (자동으로) 물을 붓는다\n");
17    printf("# 5. (자동으로) 스푼으로 저어서 녹인다\n\n");
18
```

커피 자판기 함수를 구현한다.

선택한 버튼에 따라 안내문을 출력한다.

# 1. 함수의 이해

## 1. 함수의 개념

```
19    return 0;                                          ------ 30행으로 돌아간다.
20 }
21
22 void main( )
23 {
24    int coffee;                                        ----- 커피 종류 변수와
25    int ret;                                                 반환값 변수를
26                                                             선언한다.
27    printf("어떤 커피를 드릴까요? (1:보통, 2:설탕, 3:블랙) ");
28    scanf("%d", &coffee);                              ----- 커피를 주문받는다.
29
30    ret = coffee_machine(coffee);                      ----- 커피 자판기의 버튼을 누른다
                                                               (coffee_machine( ) 함수를 호출한다).
31
32    printf("손님~ 커피 여기 있습니다.\n\n");
33 }
```

실행 결과

```
어떤 커피를 드릴까요? (1:보통, 2:설탕, 3:블랙) 2

# 1.(자동으로) 뜨거운 물을 준비한다
# 2. (자동으로) 종이컵을 준비한다
# 3. (자동으로) 설탕커피를 탄다
# 4. (자동으로) 물을 붓는다
# 5. (자동으로) 스푼으로 저어서 녹인다

손님~ 커피 여기 있습니다.
```

## 1. 함수의 개념

**기본 10-3** 여러 명에게 주문을 받도록 변경한 [기본 10-2] 10-3.c

```
01 #define  _CRT_SECURE_NO_WARNINGS
02 #include <stdio.h>
03 int coffee_machine(int button)
04 {
05    printf("\n# 1.(자동으로) 뜨거운 물을 준비한다\n");
06    printf("# 2. (자동으로) 종이컵을 준비한다\n");
07
08    switch(button)
09    {
10    case 1: printf("# 3. (자동으로) 보통커피를 탄다\n"); break;
11    case 2: printf("# 3. (자동으로) 설탕커피를 탄다\n"); break;
12    case 3: printf("# 3. (자동으로) 블랙커피를 탄다\n"); break;
13    default: printf("# 3. (자동으로) 아무거나 탄다\n"); break;
14    }
15
16    printf("# 4. (자동으로) 물을 붓는다\n");
17    printf("# 5. (자동으로) 스푼으로 저어서 녹인다\n\n");
18
```

커피 자판기 함수를 구현한다.

선택한 버튼에 따라 안내문을 출력한다.

# 1. 함수의 이해

## 1. 함수의 개념

```
19    return 0;
20 }
21
22 void main( )
23 {
24    int coffee;
25    int ret;
26
27    printf("A님, 어떤 커피 드릴까요? (1:보통, 2:설탕, 3:블랙) ");
28    scanf("%d", &coffee);
29    ret = coffee_machine(coffee);
30    printf("A님 커피 여기 있습니다.\n\n");
31
32    printf("B님, 어떤 커피 드릴까요? (1:보통, 2:설탕, 3:블랙) ");
33    scanf("%d", &coffee);
34    ret = coffee_machine(coffee);
35    printf("B님 커피 여기 있습니다.\n\n");
36
37    printf("C님, 어떤 커피 드릴까요? (1:보통, 2:설탕, 3:블랙) ");
```

각각을 호출한 곳으로 돌아간다(29행, 34행, 39행).

주문을 받고 커피 자판기의 버튼을 누른다 (함수를 호출한다).

주문을 받고 커피 자판기의 버튼을 누른다 (함수를 호출한다).

## 1. 함수의 개념

```
38    scanf("%d", &coffee);
39    ret = coffee_machine(coffee);
40    printf("C님 커피 여기 있습니다.\n\n");
41 }
```

----- 주문을 받고 커피 자판기의 버튼을 누른다 (함수를 호출한다).

실행 결과

```
A님, 어떤 커피 드릴까요? (1:보통, 2:설탕, 3:블랙) 1

# 1.(자동으로) 뜨거운 물을 준비한다
# 2. (자동으로) 종이컵을 준비한다
# 3. (자동으로) 보통커피를 탄다
# 4. (자동으로) 물을 붓는다
# 5. (자동으로) 스푼으로 저어서 녹인다

A님 커피 여기 있습니다.

B님, 어떤 커피 드릴까요? (1:보통, 2:설탕, 3:블랙) 2

# 1.(자동으로) 뜨거운 물을 준비한다
# 2. (자동으로) 종이컵을 준비한다
# 3. (자동으로) 설탕커피를 탄다
# 4. (자동으로) 물을 붓는다
# 5. (자동으로) 스푼으로 저어서 녹인다

B님 커피 여기 있습니다.

C님, 어떤 커피 드릴까요? (1:보통, 2:설탕, 3:블랙) 3
⋮
```

## 2. 함수의 모양과 활용

### ▪ 함수의 기본 형태

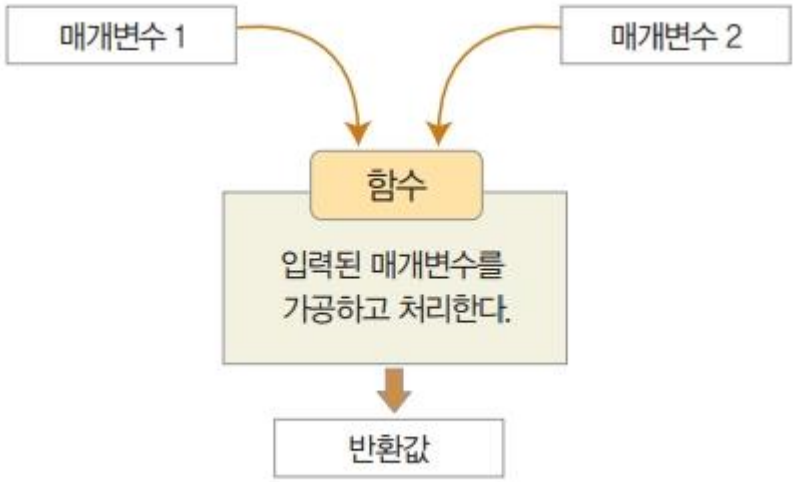


그림 10-3 함수의 형태

- 함수는 '매개변수(또는 '인수')'를 입력받은 후 그 매개변수를 가공하고 처리한 후 '반환값'을 돌려줌(커피 자판기를 예로 들면 '동전 넣기'과 '버튼 입력'이라는 매개변수를 받아서 커피를 탄 후 반환값으로 '커피'를 돌려줌)

# 1. 함수의 이해

## 2. 함수의 모양과 활용

**기본 10-4** 본격적인 함수 사용 예 1 — 10-4.c

```
01 #include <stdio.h>
02
03 int plus(int v1, int v2)          ----- plus( ) 함수를 정의한다.
04 {
05    int result;
06    result = v1 + v2;              ------ 3행에서 받은 두 매개변수의 합을 구한다.
07    return result;                 ------ plus( ) 함수를 호출한 곳에 result 값을 반환한다.
08 }
09
10 void main( )
11 {
12    int hap;
13
14    hap = plus(100, 200);          ----- 매개변수 2개를 지정하여 plus( ) 함수를
                                           호출하고 반환값은 hap에 저장한다.
15
16    printf("100과 200의 plus( ) 함수 결과는 : %d\n", hap);
17 }
```

실행 결과

```
100과 200의 plus( ) 함수 결과는 : 300
```

# 1. 함수의 이해

- **plus( ) 함수를 정의와 호출 과정**

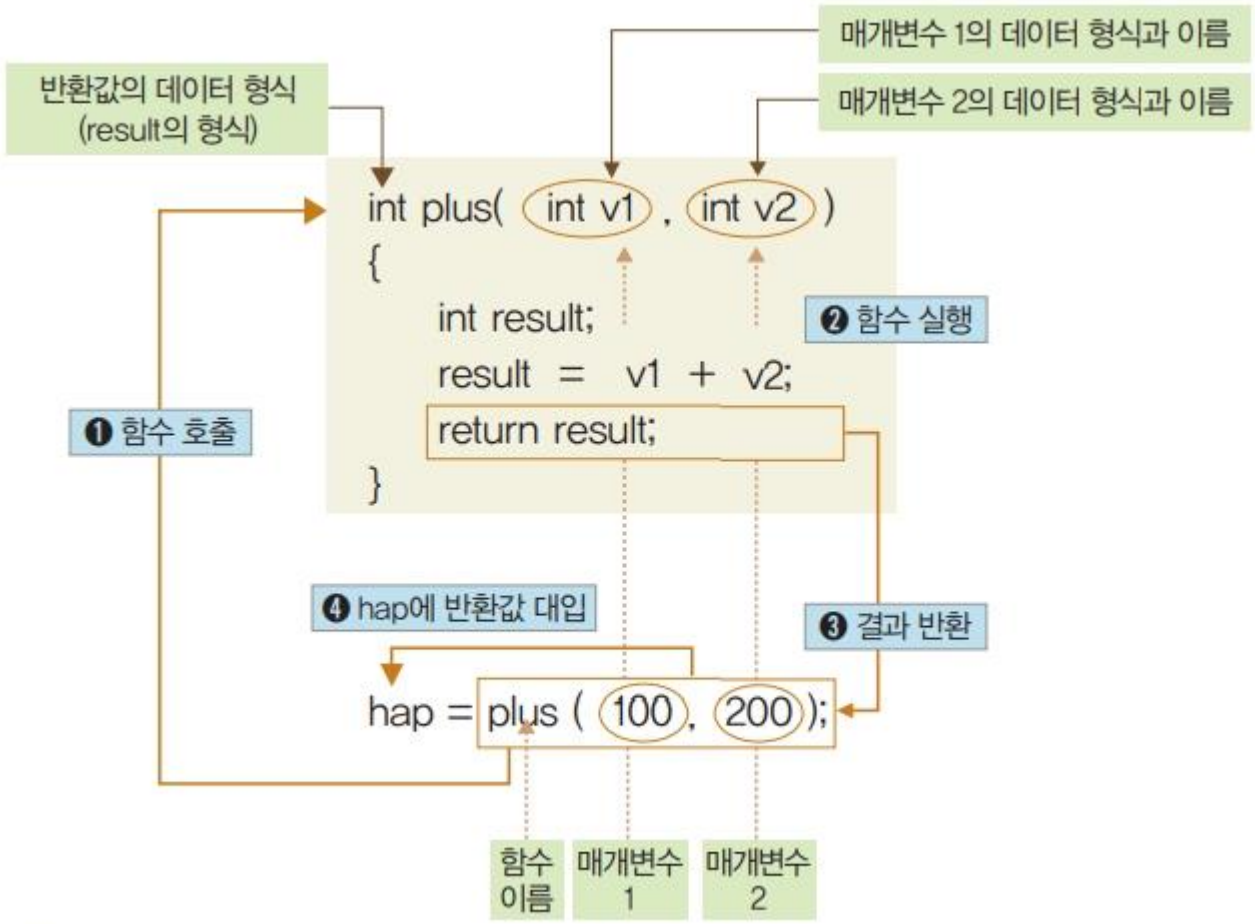


그림 10-4 plus( ) 함수의 형태와 호출 순서

# 1. 함수의 이해

❶ **함수 호출 :** plus(100, 200); 으로 함수 호출

❷ **함수 실행** : v1과 v2를 더해 result에 대입시킨 후 이 함수를 호출했던 곳으로 돌아감

❸ **결과 반환** : 결과 result값(300)을 plus( ) 함수를 호출했던 곳으로 돌려줌

❹ **hap에 반환값 대입** : result값 300을 변수 hap에 대입. plus(100, 200)의 결과를 hap에 넣어야 하므로 hap과 plus( ) 함수의 데이터 형식이 같아야 함

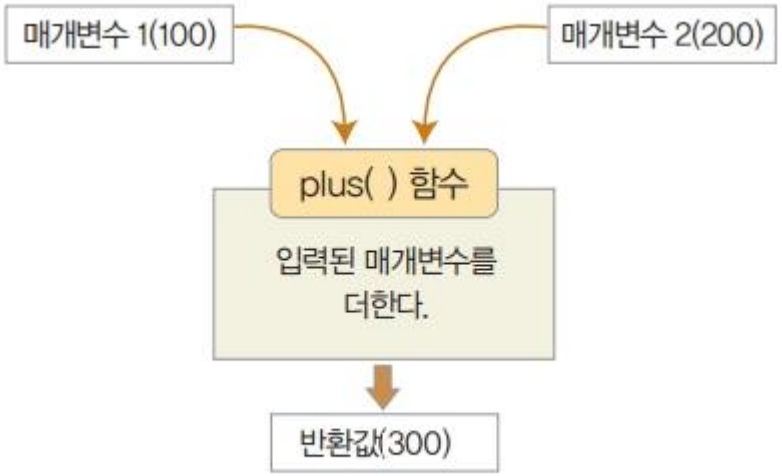


그림 10-5 plus( ) 함수의 호출을 간단하게 표현한 형태

## 2. 함수의 모양과 활용

**응용 10-5** 본격적인 함수 사용 예 2 — 10-5.c

```
01 #define _CRT_SECURE_NO_WARNINGS
02 #include <stdio.h>
03 int calc(int v1, int v2, int op)
04 {
05    int result;
06
07    switch(__1__)
08    {
09    case 1:   result = v1 + v2; break;
10    case 2:   result = v1 - v2; break;
11    case 3:   result = v1 * v2; break;
12    case 4:   result = v1 / v2; break;
13    }
14
15    return result;
16 }
17
```

03 ~~~ 매개변수 3개를 받아서 계산하는 함수이다.

07 ~~~ 매개변숫값에 따라서 '1: 덧셈, 2: 뺄셈, 3: 곱셈, 4: 나눗셈'을 실행한다.

15 ~~~ 계산 결과를 반환한다.

# 1. 함수의 이해

## 2. 함수의 모양과 활용

```
18 void main( )
19 {
20    int res;
21    int oper, a, b;
22
23    printf("계산 입력 (1:+, 2:-, 3:*, 4:/) : ");
24    scanf("%d", &oper);
25
26    printf("계산할 두 숫자를 입력 : ");
27    scanf("%d %d", &a, &b);
28
29    res = calc( __2__ );
30
31    printf("계산 결과는 : %d\n", res);
32 }
```

----- 계산 결과, 연산자, 입력 숫자에 대한 변수이다.

----- 연산자를 입력한다.

----- 계산할 두 숫자를 입력한다.

----- 매개변수 3개를 넣고 calc( ) 함수를 호출한다. 계산 결과는 res에 저장한다.

**실행 결과**

```
계산 입력 (1:+, 2:-, 3:*, 4:/) : 3
계산할 두 숫자를 입력 : 7  8
계산 결과는 : 56
```

# 02
# 지역변수와 전역변수

# 2. 지역변수와 전역변수

- 지역변수 : 한정된 지역(local)에서만 사용되는 변수
- 전역변수 : 프로그램 전체(global)에서 사용되는 변수


그림 10-6 지역변수와 전역변수의 생존 범위

# 2. 지역변수와 전역변수

- ❶에서 a는 현재 '함수 1' 안에 선언, 그러므로 a는 '함수 1' 안에서만 사용될 수 있고, '함수 2'에서는 a의 존재를 모름
- ❷는 전역변수 b를 보여줌. b는 함수(함수 1, 함수 2) 안이 아니라 함수 바깥에 선언되어 있으므로 모든 함수에서 b의 존재를 알게 됨

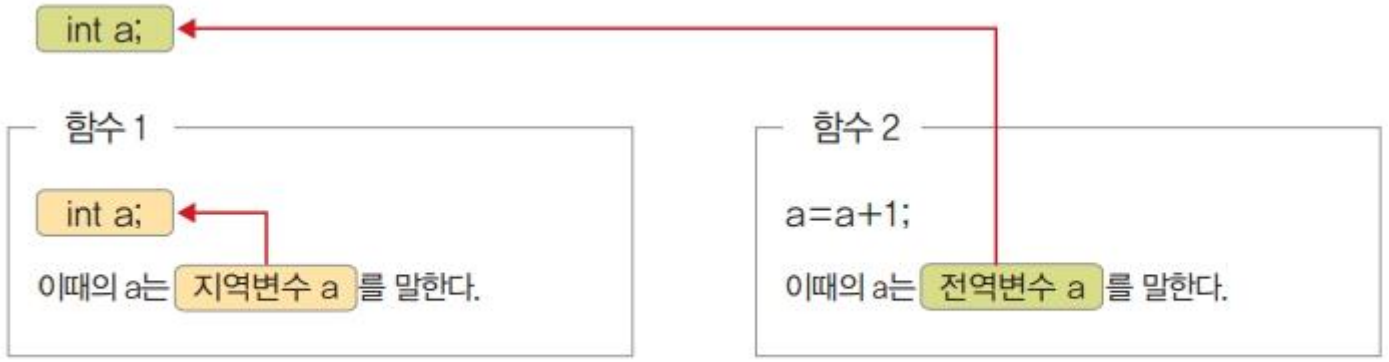


그림 10-7 이름이 같은 지역변수와 전역변수의 구분

- 같은 a라고 해도 '함수 1의 a'는 함수 내에서 따로 정의했으므로 지역변수,

  '함수 2의 a'는 함수 안에 정의된 것이 없으므로 전역변수

# 2. 지역변수와 전역변수

**기본 10-6** 지역변수와 전역변수의 구분 10-6.c

```
01 #include <stdio.h>
02
03 int a = 100;                          ----- 전역변수 a를 선언하고 초깃값을 대입한다.
04
05 void func1( )
06 {
07    int a = 200;                       ----- 지역변수 a를 선언하고 초깃값을 대입한다.
08    printf("func1( )에서 a의 값==> %d\n", a);   ----- 지역변수를 출력한다.
09 }
10
11 void main( )
12 {
13    func1( );                          ----- 함수를 호출한다.
14    printf("main( ) 에서 a의 값==> %d\n", a);   ----- 전역변수를 출력한다.
15 }
```

실행 결과

```
func1( )에서 a의 값==> 200
main( ) 에서 a의 값==> 100
```

# 2. 지역변수와 전역변수

## 여기서 잠깐 main( ) 함수의 반환값

- 원칙적으로 main( ) 함수의 맨 아래에 'return 정숫값;'과 같은 반환값을 지정해야 하 지만, main( ) 함수의 끝이 프로그램의 끝이어서 return문을 사용하지 않아도 별 문제가 없음
- main( ) 함수는 void main( )이나 int main( ) 중 아무거나 사용해도 무방

# 03
# 함수의 반환값과 매개변수

# 3. 함수의 반환값과 매개변수

## 1. 반환값 유무에 따른 함수 구분

### ▪ 반환값이 있는 함수

- 함수를 실행한 후에 나온 결과값은 함수의 데이터형을 따름

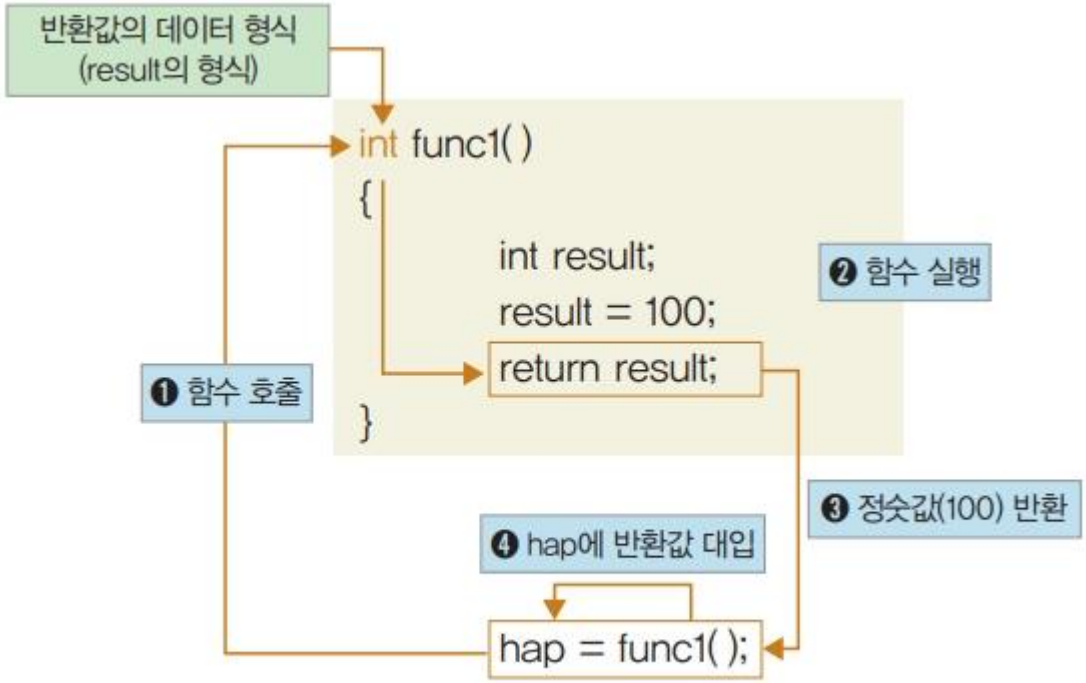


그림 10-8 int 형 값의 반환

# 3. 함수의 반환값과 매개변수

- **반환값이 없는 함수**
  - 함수를 실행한 결과, 돌려줄 것이 없는 경우
  - void로 함수 표시 : void 형 함수를 호출할 때는 함수 이름만 표시

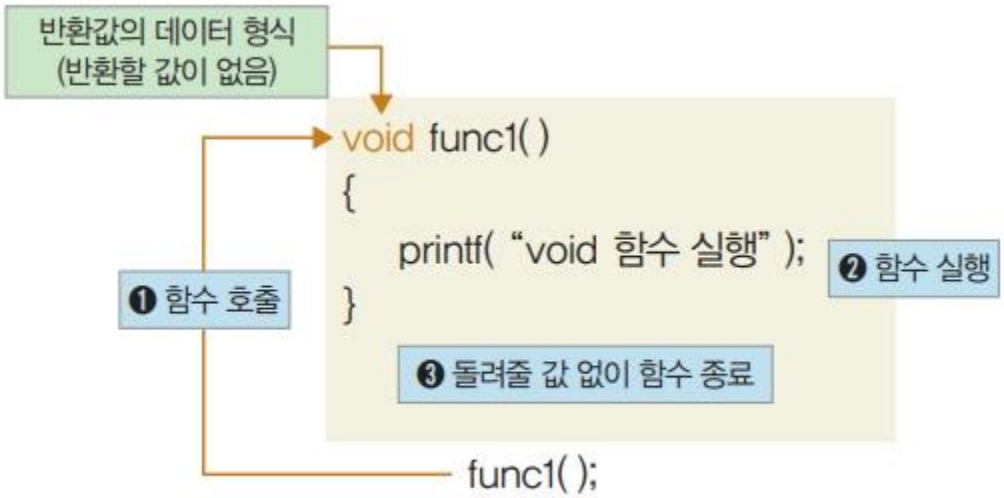


그림 10-9 void 형 함수의 작동

# 3. 함수의 반환값과 매개변수

- **반환값이 없는 함수**

**기본 10-7** 반환값 유무에 따른 함수 비교 10-7.c

```
01 #include <stdio.h>
02
03 void func1( )                    ----- void 형 함수이므로 반환값이 없다.
04 {
05    printf("void 형 함수는 돌려줄 게 없음.\n");
06 }
07
08 int func2( )                     ----- int 형 함수이므로 반환값이 있다.
09 {
10    return 100;
11 }
12
13 void main( )
14 {
15    int a;
16
17    func1( );                     ----- void 형 함수를 호출한다.
18
19    a = func2( );                 ----- int 형 함수를 호출한다.
20    printf("int 형 함수에서 돌려준 값 ==> %d\n", a);
21 }
```

실행 결과

```
void 형 함수는 돌려줄 게 없음.
int 형 함수에서 돌려준 값 ==> 100
```

# 3. 함수의 반환값과 매개변수

## 2. 매개변수 전달 방법

- **값의 전달(call by value)**
  - 값 자체를 함수에 넘겨주는 방법

**기본 10-8** 매개변수 전달 방법: 값으로 전달 10-8.c

```
01 #include <stdio.h>
02
03 void func1(int a)
04 {
05    a = a + 1;          ---- 전달받은 a 값을 1 증가시킨 후 출력한다.
06    printf("전달받은 a ==> %d\n", a);
07 }
08
09 void main( )
10 {
11    int a=10;           ---- 변수 a를 선언한다.
12
13    func1(a);           ---- a 값을 매개변수로 넘겨 함수를 호출한다.
14    printf("func1( ) 실행 후의 a ==> %d\n", a);  ---- 함수를 호출한 후 a 값을 출력한다.
15 }
```

실행 결과

```
전달받은 a ==> 11
func1( ) 실행 후의 a ==> 10
```

# 3. 함수의 반환값과 매개변수

## 2. 매개변수 전달 방법

- **값의 전달(call by value)**
  - 11행에서 a에 10을 입력하고 13행에서 func1(a)를 호출
  - 3행의 a에 10이 들어가면 5행에 서는 a 값을 1 증가시킨 후 출력
  - func1( ) 함수가 종료된 후 14행에서 a를 출력하니 원래의 10이 출력

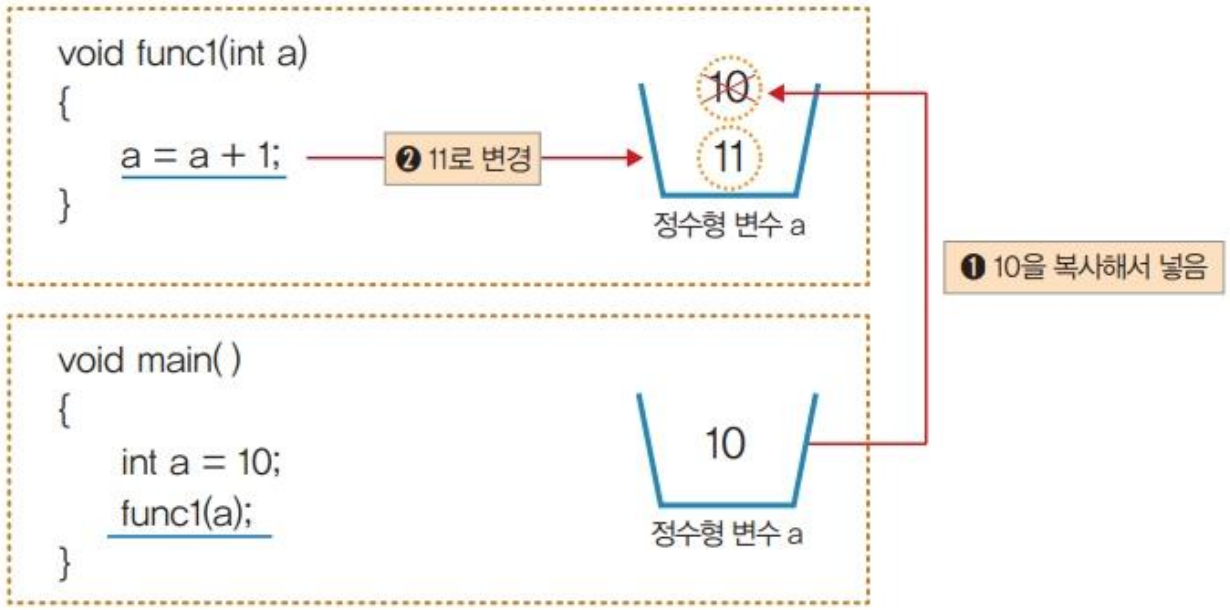


그림 10-10 매개변수 전달: 값으로 전달

# 3. 함수의 반환값과 매개변수

## 2. 매개변수 전달 방법

- 주소(또는 참조)로 전달

**기본 10-9** 매개변수 전달 방법: 주소로 전달 — 10-9.c

```
01 #include <stdio.h>
02
03 void func1(int *a)                  ----- 매개변수로 주솟값(포인터)을 받는다.
04 {
05    *a = *a + 1;                    ----- a가 가리키는 곳의 실제 값+1을 수행한다.
06    printf("전달받은 a ==> %d\n", *a);  ----- a가 가리키는 곳의 실제 값을 출력한다.
07 }
08
09 void main( )
10 {
11    int a=10;                       ----- a를 10으로 초기화한다.
12
13    func1(&a);                      ----- 함수를 호출할 때 a의 주소를 전달한다.
14    printf("func1( ) 실행 후의 a ==> %d\n", a);  ----- 함수를 호출한 후 a 값을 출력한다.
15 }
```

실행 결과

```
전달받은 a ==> 11
func1( ) 실행 후의 a ==> 11
```

# 3. 함수의 반환값과 매개변수

## 2. 매개변수 전달 방법

- 주소(또는 참조)로 전달

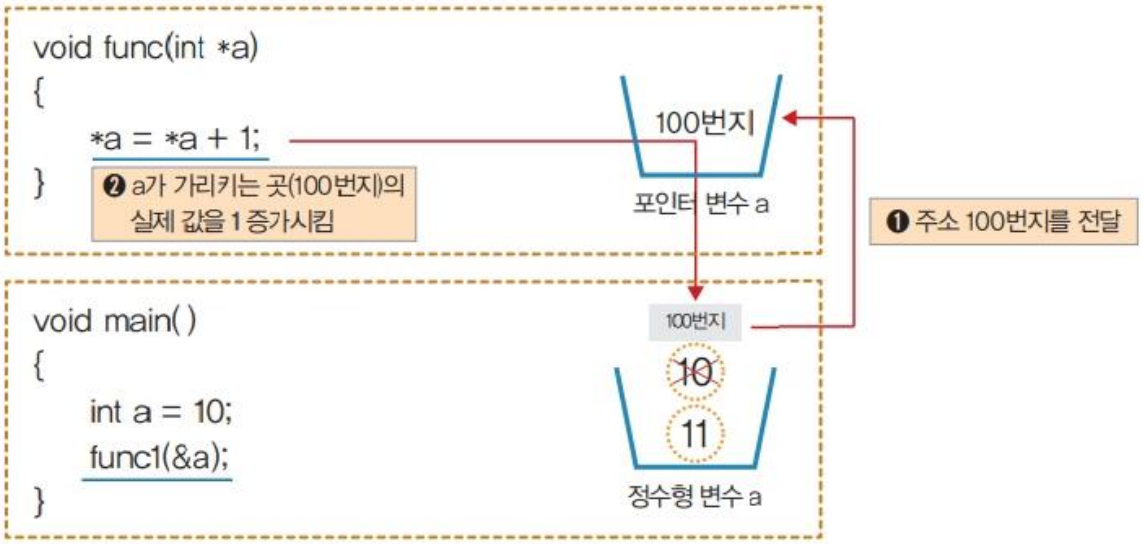


그림 10-11 매개변수 전달: 주소로 전달

# 3. 함수의 반환값과 매개변수

## 2. 매개변수 전달 방법

- 주소(또는 참조)로 전달

**응용 10-10** 매개변수 전달 방법 비교 — 10-10.c

```
01 #include <stdio.h>
02
03 void func1(char a, char b)          ---- 매개변수가 값인 함수이다.
04 {
05    int imsi;
06
07    imsi = a;                       ------ 두 문자를 교환한다.
08    __1__
09    b = imsi;
10 }
11
12 void func2(char *a, char *b)        ---- 매개변수가 주소인 함수이다.
13 {
14    int imsi;
15
16    imsi = *a;                      ------ 두 문자를 교환한다.
17    __2__
18    *b = imsi;
19 }
```

# 3. 함수의 반환값과 매개변수

## 2. 매개변수 전달 방법

- 주소(또는 참조)로 전달

```
20
21 void main( )
22 {
23    char x = 'A', y = 'Z';
24
25    printf("원래 값        : x=%c, y=%c\n", x, y);
26
27    func1( __3__ );
28    printf("값을 전달한 후  : x=%c, y=%c\n", x, y);
29
30    func2( __4__ );
31    printf("주소를 전달한 후: x=%c, y=%c\n", x, y);
32 }
```

25 ---- 원래 문자를 출력한다.

27 ---- 값을 전달해서 func1( ) 함수를 호출한다.

30 ---- 주소를 전달해서 func2( ) 함수를 호출한다.

정답_ 1 a = b;  2 *a = *b;  3 x, y  4 &x, &y

**실행 결과**

```
원래 값        : x=A, y=Z
값을 전달한 후  : x=A, y=Z
주소를 전달한 후: x=Z, y=A
```

# 3. 함수의 반환값과 매개변수

## 2. 매개변수 전달 방법

- 주소(또는 참조)로 전달

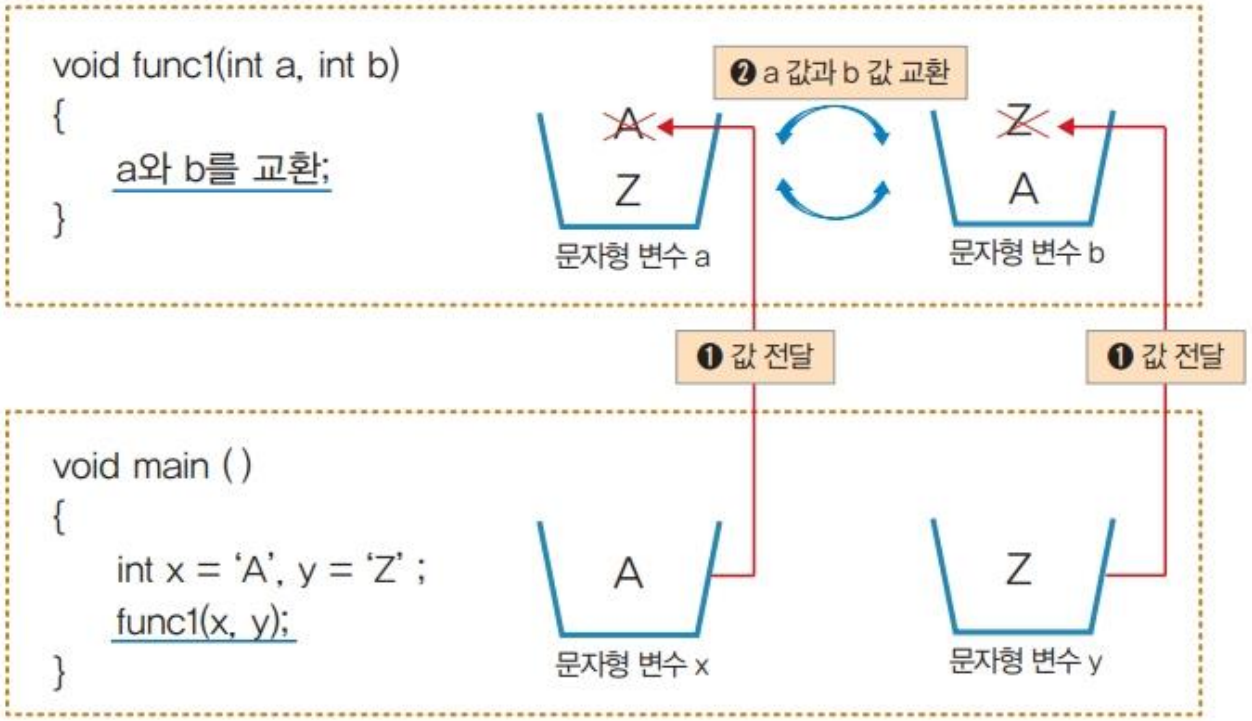


그림 10-12 값으로 전달을 통한 교환

# 3. 함수의 반환값과 매개변수

## 2. 매개변수 전달 방법

### ▪ 주소(또는 참조)로 전달

- 30행에서는 주소로 전달하므로 [그림 10-13]과 같이 작동
- 주솟값을 매개변수로 주었기 때문에 func2( ) 함수를 호출하면 main( ) 함수에서 출력되는 값에도 영향을 미침

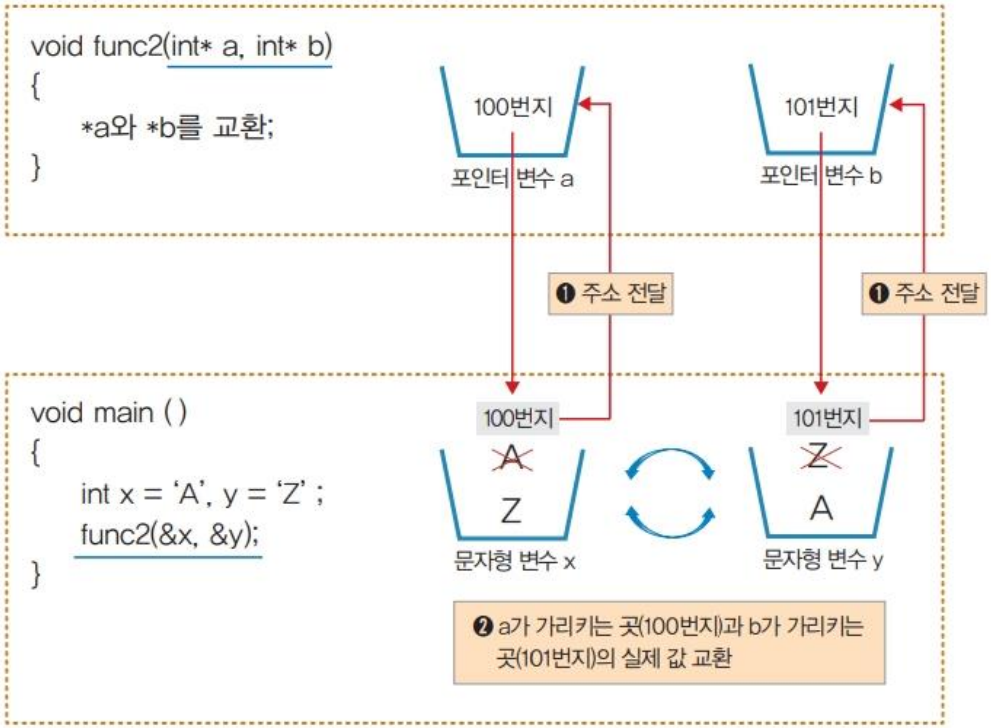


그림 10-13 주소로 전달을 통한 교환

*

# 예제 모음

# [예제모음 27] 함수를 이용한 구구단 프로그램

**예제 설명** 함수를 사용하여 구구단을 출력하는 프로그램이다.

**실행 결과**

```
출력하고 싶은 단을 입력 : 7
7 X  1=  7
7 X  2= 14
7 X  3= 21
7 X  4= 28
7 X  5= 35
7 X  6= 42
7 X  7= 49
7 X  8= 56
7 X  9= 63
```

# [예제모음 27] 함수를 이용한 구구단 프로그램

```
01 #define _CRT_SECURE_NO_WARNINGS
02 #include <stdio.h>
03 void gugu(int dan)
04 {
05    int i;
06
07    for(i=1; i <= 9; i++)
08    {
09       printf("%2d X %2d= %2d  \n", dan, i, dan*i);
10    }
11 }
12
13 void main( )
14 {
15    int input;
16
17    printf("출력하고 싶은 단을 입력 : ");
18    scanf("%d", &input);
19
20    gugu(input);
21 }
```

gugu( ) 함수를 정의한다 (매개변수는 정수형 dan이다).

1~9를 반복하며 매개변수로 받은 dan의 단을 출력한다.

출력할 단을 입력한다.

구구단을 계산하고 출력할 함수를 내보낸다.

# [예제모음 28] 함수를 이용한 대·소문자 변환 프로그램

**예제 설명** 대문자는 소문자로, 소문자는 대문자로 변환하는 프로그램이다.

❶ 대문자 변환 방법: 소문자에서 대·소문자 차이를 뺀다.
❷ 소문자 변환 방법: 대문자에서 대·소문자 차이를 더한다.

**실행 결과**

```
문자열을 입력(100자 이내) : Hanbit_Academy
변환된 결과 ==> hANBIT_aCADEMY
```

# [예제모음 28] 함수를 이용한 대·소문자 변환 프로그램

```
01 #define  _CRT_SECURE_NO_WARNINGS
02 #include <stdio.h>
03 char upper(char ch)                          ----- 대문자로 변환하는 함수이다.
04 {  return ch - ('a' - 'A');  }
05
06 char lower(char ch)                          ----- 소문자로 변환하는 함수이다.
07 {  return ch + ('a' - 'A');  }
08
09 void main( )
10 {
11    char in[100], out[100];
12    char ch;
13    int i =0;
14
15    printf("문자열을 입력(100자 이내) : ");
16    scanf("%s", in);                          ----- 문자열을 입력받는다.
```

# [예제모음 28] 함수를 이용한 대·소문자 변환 프로그램

```
17
18    do {
19       ch = in[i];
20       if(ch >= 'A' && ch <= 'Z')
21          out[i] = lower(ch);
22       else if(ch >= 'a' && ch <= 'z')
23          out[i] = upper(ch);
24       else
25          out[i] = ch;
26       i++;
27    } while(ch != '\0');
28
29    out[i] = '\0';
30    printf("변환된 결과 ==> %s\n",out);
31 }
```

문자열이 널이 아닌 동안 반복한다. 즉 문자열의 끝까지 반복한다.

문자형 배열에서 한 문자만 추출한다.

문자가 대문자이면 lower( ) 함수를, 소문자이면 upper( ) 함수를 호출한다. 숫자나 기호 등은 그대로 사용한다.

출력 문자열의 맨 뒤에 널 문자를 추가한다.

# [예제모음 29] 숫자 자동 추첨 프로그램

**예제 설명** 1~45 중에서 숫자 6개를 자동으로 뽑는 프로그램이다.

**실행 결과**

```
** 로또 추첨을 시작합니다. **

추첨된 로또 번호 ==> 11  40  35  34  36  15
```

# [예제모음 29] 숫자 자동 추첨 프로그램

```
01 #include <stdio.h>
02 #include <stdlib.h>          ----- 관련 함수를 사용하기 위해 포함한 헤더 파일이다.
03 #include <time.h>
04
05 int getNumber( ) {           ----- 1~45 중에서 숫자 하나를 추출하는 함수이다.
06   return rand( ) % 45 + 1;    ------ rand( ) 함수는 0~32767 중 하나를 임의로 반환한다.
07 }
08
09 void main( )
10 {
11   short int lotto[6] = {0,};  ----- 추첨된 숫자를 담을 배열이다.
12   int i, k, num;              ----- 반복 변수 i, k와 뽑힌 숫자를 담을 변수 num이다.
13   char dup='N';               ----- 이미 뽑힌 숫자인지 체크하기 위한 변수이다.
14
```

# [예제모음 29] 숫자 자동 추첨 프로그램

```
15   printf("** 로또 추첨을 시작합니다. ** \n\n");
16   srand((unsigned)time(NULL));
17
18   for(i=0;i < 6;) {
19     num = getNumber( );
20
21     for(k=0; k < 6; k++)
22       if(lotto[k] == num)
23         dup = 'Y';
24
25    if(dup == 'N')
26      lotto[i++] = num;
27    else
28      dup = 'N';
29   }
30
```

rand( ) 함수를 초기화하는 함수이다. 이 행이 없으면 늘 같은 숫자가 뽑힌다.

로또 숫자를 1개 뽑는다.

다른 숫자 6개가 뽑힐 때까지 반복한다(18~28행). 다른 숫자가 뽑히면 18행에서 i를 1 증가시킨다.

뽑은 숫자가 이전에 뽑은 숫자와 동일한지 체크하고, 동일하면 중복 확인 변수에 'Y'를 대입한다.

뽑은 숫자가 처음 나온 숫자라면 로또 배열에 넣고 i(뽑힌 개수)를 1 증가시킨다(25~28행). 아니면 다시 중복 확인 변수에 'N'을 대입한다.

# [예제모음 29] 숫자 자동 추첨 프로그램

```
31    printf("추첨된 로또 번호 ==> ");
32    for(i= 0 ; i < 6 ; i++) {          ----- 뽑힌 로또 숫자 6개를 출력한다.
33      printf("%d   ", lotto[i]);
34    }
35
36    printf("\n\n");
37  }
```

# 감사합니다!