C Language

Lecture 8. 함수

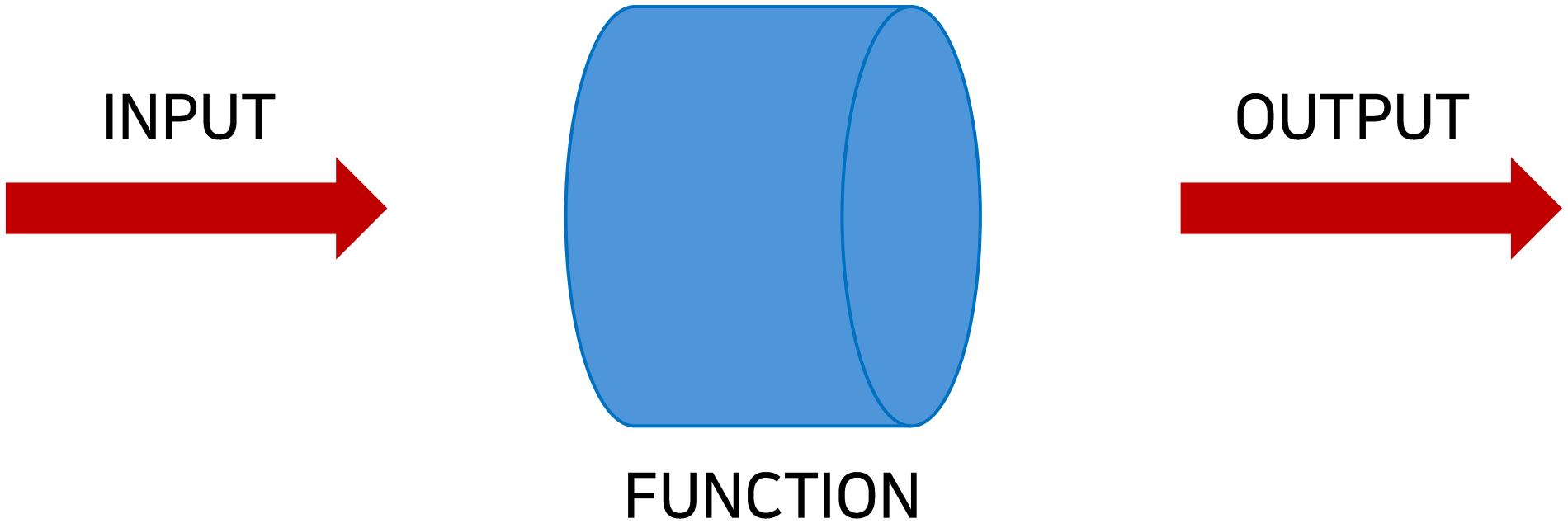
BAE SEKANG(배세강)

Contents

Function

♣ 함수

- 특정 작업을 수행하기 위해 모아둔 코드의 집합



Function

♣ 함수 생성

- [반환 타입] [함수명](매개변수) {실행할 코드}
- main()도 함수이다.

```
#include <stdio.h>

int main() {
    int result = 10;
    printf("결과: %d", result);

    return 0;
}
```

Function

♣ 함수 생성

- double_function 생성

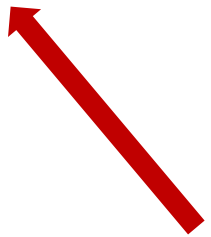
```
#include <stdio.h>

int double_function(int x) {
    return x * 2;
}

int main() {
    int result = double_function(10);
    printf("결과: %d", result);
    return 0;
}
```

Function

♣ 함수 호출

- 함수 호출: double_function(매개변수)

```
#include <stdio.h>

int double_function(int x) {
    return x * 2;
}

int main() {
    int result = double_function(10);
    printf("결과: %d", result);
    return 0;
}
```



Function

♣ 반환값

- return 키워드를 활용
- 반환값이 필요 없는 경우에는 void 사용

```
#include <stdio.h>

int double_function(int x) {
    return x * 2;
}

int main() {
    int result = double_function(10); // 20이 result에 저장됨
    printf("결과: %d", result);
    return 0;
}
```

Function

♣ 매개변수

- 콤마(,)로 구분하여 여러 개의 매개변수도 사용 가능

```
#include <stdio.h>

int add(int a, int b) {
    return a + b;
}

int main() {
    int result = add(3, 5);
    printf("결과: %d", result);
    return 0;
}
```

Function

[Practice]

두 정수 A, B를 입력 받을 때, A와 B 중 더 큰 수를 반환하는 함수를 활용해 이를 출력한다.

[입력]
10 30

[출력]
30

Function

[Practice]

정수 A를 입력 받을 때, |A|를 반환하는 함수를 활용하여 이를 출력한다.

[입력 1]
-5

[입력 2]
123

[출력]
5

[출력 2]
123

Function

♣ Prototype

- 모든 식별자(함수, 변수)는 사용 전에 선언되어야 한다.

```
#include <stdio.h>

int abs(int x);                 // 미리 abs()의 존재만 알림

int main() {
   int a, b;
   scanf("%d %d", &a, &b);
   printf("%d", abs(a - b));

   return 0;
}

int abs(int x) {                // 실제 함수의 구조는 뒤에 정의
   return x < 0 ? -x : x;
}
```

Function

♣ 가변 인자(Variable Arguments)

- 매개 변수의 개수를 마음대로 정해서 넘길 수 있는 기능
- #include <stdarg.h> 필요
- va_list: 가변 인자들이 저장된 목록을 가리키는 *포인터 ap
- va_start(ap, 마지막 고정 인자명): 가변 인자 목록을 읽기 시작할 준비
- va_arg(ap, TYPE): 목록에서 하나씩 인자를 꺼내옴
- va_end(ap): 가변 인자 처리를 끝냄
- 고정 인자는 최소 1개 이상 사용해야 함

Function

♣ 가변 인자(Variable Arguments) 예시 1/5

```
#include <stdio.h>
#include <stdarg.h> // stdarg.h 선언

int sum_all(int count, ...) {
    int total = 0;
    va_list ap;
    va_start(ap, count);
    for (int i = 0; i < count; i++) total += va_arg(ap, int);
    va_end(ap);
    return total;
}

int main() {
    printf("합계: %d₩n", sum_all(3, 10, 20, 30));
    printf("합계: %d₩n", sum_all(2, 5, 5));
    return 0;
}
```

Function

♣ 가변 인자(Variable Arguments) 예시 2/5

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

int sum_all(int count, ...) { // 고정 인자 count & 가변 인자 위치인 ... 표시
    int total = 0;
    va_list ap;
    va_start(ap, count);
    for (int i = 0; i < count; i++) total += va_arg(ap, int);
    va_end(ap);
    return total;
}

int main() {
    printf("합계: %d₩n", sum_all(3, 10, 20, 30));
    printf("합계: %d₩n", sum_all(2, 5, 5));
    return 0;
}
```

Function

♣ 가변 인자(Variable Arguments) 예시 3/5

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

int sum_all(int count, ...) {
    int total = 0;
    va_list ap; // 목록 point 선언
    va_start(ap, count); // 마지막 고정 인자명 count 사용
    for (int i = 0; i < count; i++) total += va_arg(ap, int);
    va_end(ap);
    return total;
}

int main() {
    printf("합계: %d₩n", sum_all(3, 10, 20, 30));
    printf("합계: %d₩n", sum_all(2, 5, 5));
    return 0;
}
```

Function

♣ 가변 인자(Variable Arguments) 예시 4/5

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

int sum_all(int count, ...) {
    int total = 0;
    va_list ap;
    va_start(ap, count);
    for (int i = 0; i < count; i++) total += va_arg(ap, int); // 인자를 int 타입으로 하나씩 꺼냄
    va_end(ap);
    return total;
}

int main() {
    printf("합계: %d₩n", sum_all(3, 10, 20, 30));
    printf("합계: %d₩n", sum_all(2, 5, 5));
    return 0;
}
```

Function

♣ 가변 인자(Variable Arguments) 예시 5/5

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

int sum_all(int count, ...) {
    int total = 0;
    va_list ap;
    va_start(ap, count);
    for (int i = 0; i < count; i++) total += va_arg(ap, int);
    va_end(ap); // 목록 포인터 사용 종료
    return total;
}

int main() {
    printf("합계: %d₩n", sum_all(3, 10, 20, 30));
    printf("합계: %d₩n", sum_all(2, 5, 5));
    return 0;
}
```

Scope

♣ local variable(지역 변수)

- 함수 내에서 선언한 변수
- 함수 실행 시 생성되고, 함수의 실행이 끝나면 소멸된다.
- 매개변수도 지역 변수이다.

```
#include <stdio.h>

int main() {
    int a, b; // main 함수의 지역 변수 a, b
    scanf("%d %d", &a, &b);
    printf("%d", a + b);
    return 0;
}
```

Scope

♣ local variable(지역 변수)

```
#include <stdio.h>

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

int main() {
    int a, b;
    for(int i=0; i<5; ++i) {
        scanf("%d %d", &a, &b);
        printf("%d", max(a, b));
    }
}
```

Scope

♣ global variable(전역 변수)

- 함수 밖에서 선언한 변수
- 모든 함수에서 사용 가능하다.

```
#include <stdio.h>

double pi = 3.14;          // 전역 변수 선언

double area(double r) {
    return pi * r * r;
}

int main() {
    double r;
    scanf("%lf", &r);
    printf("Pi: %.2lf, Area: %.2lf", pi, area(r));
}
```

Scope

♣ global variable(전역 변수)

- 자동으로 0으로 초기화됨
- 어느 함수에서나 접근이 가능
- 프로그램 실행 내내 메모리를 차지함 ➔ 꼭 필요한 경우만 사용하기

  예) 프로그램 실행 내내 값을 기억해야 할 경우, 함수와 함수 간의 데이터 교환이 빈번한 경우 등

Scope

♣ static

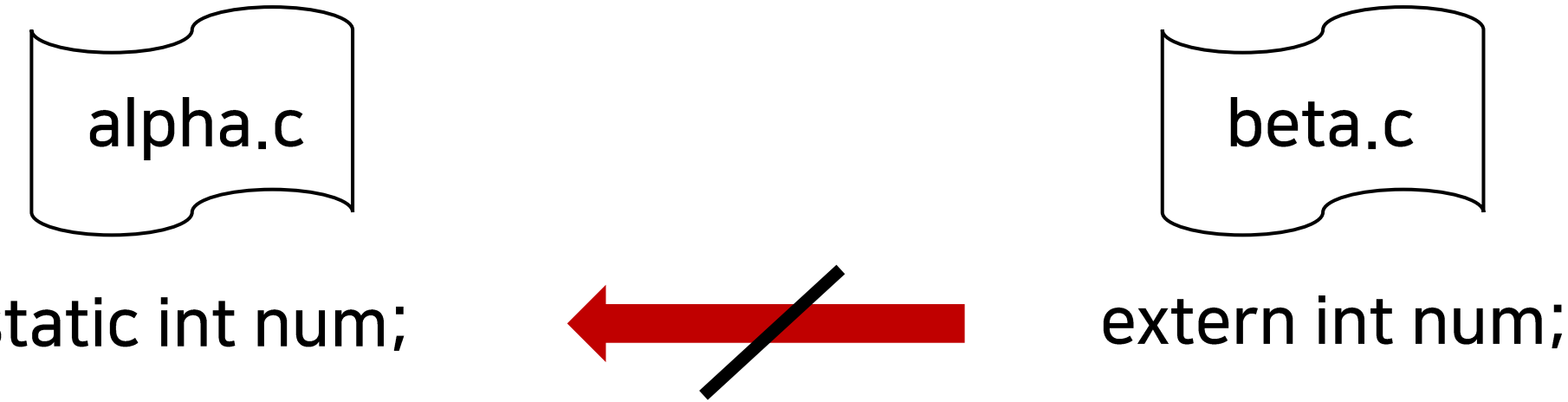
- 함수가 종료되어도 프로그램이 종료되기 전까지 유지되는 지역 변수
- 생성 지점부터 프로그램 종료전까지 메모리를 차지함

```
#include <stdio.h>

void countUp() {
    static int arr = 0; // 프로그램 시작 시 딱 1번 초기화됨
    arr++;
    printf("현재 값: %d₩n", arr);
}

int main() {
    countUp(); // 출력: 1
    countUp(); // 출력: 2 (값이 유지)
}
```

Scope

♣ static

- 함수 내부에 선언된 경우: scope는 해당 함수 내부
- 전역으로 선언된 경우: 전역변수이지만 외부 파일에서는 접근 불가



Scope

♣ shadowing

- Scope: 식별자(함수, 변수 등)가 사용될 수 있는 범위
- Shadowing: Scope 내에서 선언된 변수가 더 바깥쪽 범위에 있는 동일한 이름의 변수를 일시적으로 숨기는 현상

```
#include <stdio.h>

int main() {
   int x = 10;

   if (1) {
      int x = 20;
      printf("안쪽 x: %d₩n", x);
   }
   printf("바깥쪽 x: %d₩n", x);
}
```

Other Functions

♣ main()

- 프로그램이 실행될 때 OS가 가장 먼저 호출하는 Entry Point 함수

```
// 인자가 없는 경우
#include <stdio.h>

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

```
// 외부(터미널/OS)로부터 인자를 받는 경우
#include <stdio.h>

int main(int argc, char *argv[]) {
    return 0;
}
```

Other Functions

♣ main()

- int argc: 명령창에 입력된 단어의 개수
- char *argv[]: 입력된 문자들의 배열

Example)

./my_proc hello 123

argc == 3
argv[0] == “./my_proc”, argv[1] == “hello”, argv[2] == “123”

Other Functions

♣ main()

- main 함수의 return값: OS에게 프로그램의 최종 종료 상태를 알려줌

```c
#include <stdio.h>

int main() {
    int a, b;
    scanf("%d %d", &a, &b);
    printf("%d", a + b);
    return 123;
}
```

0: 정상 종료
0 이외: 에러 or 특이 사항 발생

```
🚀 C-Coding >
🚀 C-Coding > gcc -o test.exe test.c
🚀 C-Coding > ./test.exe
3 4
7
🚀 C-Coding > $lastExitCode
123
```

$lastExitCode: 프로그램의 종료 상태/반환 코드

Other Functions

♣ math.h

- pow(base, exp): 거듭제곱을 계산
- sqrt(x): 제곱근을 계산
- abs(x): 절댓값을 계산
- …

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

int main() {
    double result = pow(2, 10);
    printf("%.1f", result);                    // 1024.0
    return 0;
}
```

Other Functions

♣ string.h

- strlen(s): 문자열의 길이 반환
- strcpy(dest, src): src의 문자열을 dest로 복사함
- strcmp(s1, s2): s1과 s2의 문자열을 비교함
  (s1==s2면 0, s1<s2면 음수, s1>s2면 양수)

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

int main() {
    char a[20] = "Apple", b[20];
    strcpy(b, a);
    strcat(b, " Pie");
    if (strcmp(a, b) != 0) printf("결과: %s₩n", b);
    return 0;
}
```

Other Functions

♣ stdlib.h

- rand(): 난수 생성
- exit(): 프로그램 종료
- atoi(): 문자열을 정수로 변환함
- …

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h> // 시간 관련 라이브러리
int main() {
    char str[] = "100";
    int num = atoi(str);
    srand(time(NULL));
    int random = rand() % num;
    printf("변환 숫자: %d, 랜덤 숫자: %d₩n", num, random);
    return 0;
}
```

Other Functions

[Practice]

두 양의 정수 A, B를 입력 받을 때, A와 B의 최대공약수를 출력한다.

[입력 1]
3 7

[입력 2]
12 48

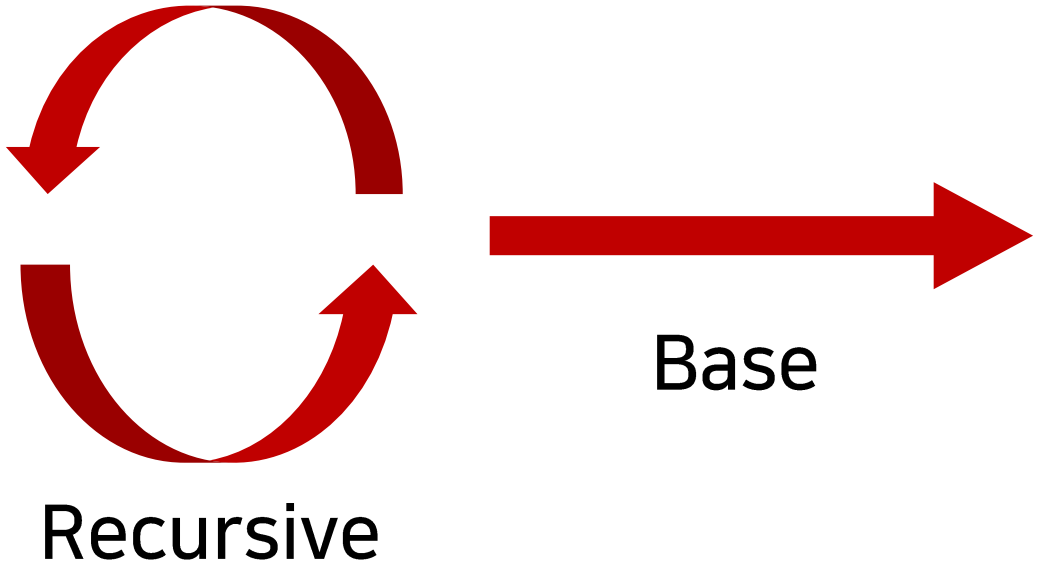
[출력]
1

[출력 2]
12

Recursions

♣ recursion

- 함수 내부에서 자기 자신을 호출하는 함수
- Base Case: 재귀를 멈추는 지점
- Recursive Case: 자기 자신을 호출하는 지점



Recursions

♣ recursion

- 팩토리얼 계산 함수

```
#include <stdio.h>

int factorial(int n) {
    if (n <= 1) return 1;              // Base Case
    return n * factorial(n - 1);  // Recursive Case
}

int main() {
    int result = factorial(5);
    printf("5! 결과: %d₩n", result);
    return 0;
}
```

Recursions

[Practice]

자연수 n을 입력 받을 때, 피보나치 수열의 n번 째 항을 출력한다.
(피보나치 수열은 1, 1, 2, 3, 5, 8, 13, 21, …이며, 앞선 두 항을 더한 값이 현재 항의 값이 된다.)

[입력 1]
7

[출력 1]
13

Exercise

[Problem 1]

정수 N을 입력 받은 후, N개의 단어(최대 20글자)를 입력 받을 때, 가장 길이가 긴 단어를 출력한다. 길이가 같다면 사전 순으로 앞선 단어를 출력한다.
(단, 1 <= 100 <= N)

[입력 1]
3
apple
mango
melon

[출력 1]
apple

Exercise

[Problem 1] - Answer

```
#include <stdio.h>
#include <string.h>
int main() {
    int n;
    char current_word[21];
    char best_word[21] = "";
    int max_len = 0;
    scanf("%d", &n);
    for (int i = 0; i < n; i++) {
        scanf("%s", current_word);
        int cur_len = strlen(current_word);
        if (cur_len > max_len) {
            max_len = cur_len;
            strcpy(best_word, current_word);
        }
        else if (cur_len == max_len) {
            if (strcmp(current_word, best_word) < 0) {
                strcpy(best_word, current_word);
            }
        }
    }
    printf("%s₩n", best_word);
    return 0;
}
```

Exercise

[Problem 2]

10진수 양의 정수를 입력 받아, 이를 2진수로 변환하여 출력한다. 단, 반드시 **재귀함수**를 사용해야 하며, 반복문을 사용하지 않는다.

[입력 1]
123

[출력 1]
1111011

[입력 2]
12

[출력 2]
1100

[입력 2]
5

[출력 2]
101

Exercise

[Problem 2] - Answer

```
#include <stdio.h>

void print_binary(int n) {
    if (n == 0) return;
    print_binary(n / 2);
    printf("%d", n % 2);
}

int main() {
    int n;
    scanf("%d", &n);
    if (n == 0) {
        printf("0₩n");
    } else {
        print_binary(n);
        printf("₩n");
    }

    return 0;
}
```

THANK YOU!

Any Question?