# 함수

## 함수

- 적당한 코드들의 묶음, 같은 행동이 반복될 때 주로 사용함
- 변수의 종류
  - 지역 변수 : { } 안에서 선언하여, 그 함수에서만 접근할 수 있는 변수
  - 전역 변수 : { } 밖에서 선언하여, 모든 함수에서 접근할 수 있는 변수
  - 매개변수 : 함수를 실행할 때 들어가는 변수로 매개변수 또한 선언한 함수 안에서만 접근할 수 있음

```
#include <stdio.h>

// 전역 변수
int i_itemCnt = 0;
int i_money = 100;

void buyItem(int cost, int cnt) { //매개 변수
  i_itemCnt += cnt;
  i_money -= cost;
  printf("아이템을 구매했습니다.\n");
  printf("  아이템 개수: %d\n", itemCnt);
  printf("  잔액: %d\n", money);
}

int main() {
  buyItem(30, 5);

  /* 중략 */

  buyItem(50, 7);
  return 0;
}
```

```
실행 결과

아이템을 구매했습니다.
  아이템 개수: 5
  잔액: 70
아이템을 구매했습니다.
  아이템 개수: 12
  잔액: 20
```

## 함수의 반환

- return : 돌려주다, 반환
  - printf( ) 함수도 출력한 문자의 수라는 반환값이 있음
  - 함수를 선언할 때, 반환값의 자료형을 언급해야 함 *ex) int, void ...*
  - 함수가 실행된 뒤, return 값이 반환되어 함숫값을 정의함
  - 함수가 실행되는 중 return을 만나면 함수가 종료됨. (break와 비슷한 역할)

```
#include <stdio.h>

int noMeaning() { //반환값의 자료형 : int
  printf("first\n");
  return 1; // 이 아래는 실행되지 않는다

  printf("second\n");
  return 2;
}

int main() {
  int a;
  a = noMeaning();

  printf("반환된 값 : %d\n", a);
  return 0;
}
```

## Call-By-Value, Call-By-Reference

- Call-By-Value : 값에 의한 호출
  - 함수에 변수의 값을 복사해서 전달함.
  - 함수의 실행 여부와 관계없이 변수의 값은 변함이 없음
- Call-By-Reference : 참조에 의한 호출
  - 함수에 변수의 주소값을 전달함
  - 함수가 실행됨에 따라 변수의 값이 변하여 저장됨

```
#include <stdio.h>

//call-by-value
void swap(int x, int y) {
  int tmp = x;
```

```
  x = y;
  y = tmp;
}

int main() {
  int a, b;

  scanf("%d%d", &a, &b);

  swap(a, b);

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

입력

2 3

실행 결과

a = 2, b = 3

```
#include <stdio.h>

//call-by-reference
void swap(int *x, int *y) {
  int tmp = *x;
  *x = *y;
  *y = tmp;
}

int main() {
  int a, b;

  scanf("%d%d", &a, &b);

  swap(&a, &b);

  printf("a = %d, b = %d\n", a, b);

  return 0;
}
```

입력

2 3

실행 결과

```
a = 3, b = 2
```

## Prototype

- C언어는 컴파일을 하며 위부터 아래로 소스 코드를 읽어나가기 때문에, 함수가 사용되기 전에 정의되어야 함
- 특히 두 함수가 서로를 이용하는 경우에는 서로의 앞에 정의되어야 하므로 불가능한 상황이 됨
- 따라서 Prototype을 이용하여 함수의 return type, 이름, 매개변수의 자료형만을 선언한 뒤 호출하는 방법을 사용함
- 함수의 순서에 구애받지 않는다는 점 이외에도 어떠한 함수가 존재하는지 알 수 있다는 장점을 지님

```
#include <stdio.h>

//prototype
void walk(int);
void rotate(int);
void drawSquare();

int main() {
  drawSquare();
  return 0;
}

void walk(int distance) {
  printf("%dcm를 걸었습니다.\n", distance);
}

void rotate(int angle) {
  printf("%d도 회전했습니다.\n", angle);
}

void drawSquare() {
  for (int i = 1; i <= 4; i++) {
    walk(10);
    rotate(90);
  }
}
```

```
실행 결과

10cm를 걸었습니다.
```

```
90도 회전했습니다.
10cm를 걸었습니다.
90도 회전했습니다.
10cm를 걸었습니다.
90도 회전했습니다.
10cm를 걸었습니다.
90도 회전했습니다.
```

## 재귀 함수

- 자기 자신을 호출하는 함수
- 종료 조건이 없다면 무한히 반복됨
- 메모리를 많이 차지하기에 적절히 사용해야 함

```
#include <stdio.h>

//n을 매개변수로 받아오면 n!을 return하는 함수
int factorial(int n) {
  if (n == 1) return 1;
  return n * factorial(n - 1);
}

int main() {
  printf("%d\n", factorial(7));
  return 0;
}
```

```
실행 결과

5040
```

## 배열을 매개변수로 넘기기

- 배열은 포인터와 밀접한 관계가 있음
- 따라서 배열을 매개변수로 넘길 때, 넘겨주는 값이 어떠한 주소값이 되는지 생각해보면 왜 Call-By-Value가 되는지 알 수 있을 것

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

```
// call-by-reference
void printArr(int arr[4]) { //arr[4] == *arr
  for (int i = 0; i < 4; i++) {
    arr[i] *= 2;
  }
}

int main() {
  int arr[4] = { 1, 2, 3, 4 };

  printArr(arr); //arr == &arr[0]

  for (int i = 0; i < 4; i++) {
    printf("%d ", arr[i]);
  }

  return 0;
}
```

```
실행 결과

2 4 6 8
```

## Section2. 함수 종합문제

- 정수 하나를 매개변수로 받아 그 수가 짝수이면 0, 홀수이면 1을 반환하는 함수를 작성해보세요

```
#include <stdio.h>

int parity(int n){
    return (n % 2) * (n % 2);
}

int main() {

    printf("%d\n", parity(5));
    printf("%d\n", parity(-3));
    printf("%d\n", parity(6));

    return 0;
}
```

```
실행 결과

1
```

```
1
0
```

- 다음 프로그램의 실행 결과는? _ 1

```
#include <stdio.h>

int useCnt[5] = { 0 };

void useItem(int);

int main() {
  useItem(4);
  useItem(2);
  useItem(1);
  useItem(4);
  useItem(0);

  for (int i = 0; i < 5; i++) {
    printf("슬롯%d의 아이템을 %d번 썼습니다.\n", i, useCnt[i]);
  }

  return 0;
}

//선언될 때마다 useCnt[itemNum]을 증가시키고, int main()에서 출력함
void useItem(int itemNum) {
  useCnt[itemNum]++;
}
```

```
실행 결과

슬롯0의 아이템을 1번 썼습니다.
슬롯1의 아이템을 1번 썼습니다.
슬롯2의 아이템을 1번 썼습니다.
슬롯3의 아이템을 0번 썼습니다.
슬롯4의 아이템을 2번 썼습니다.
```

- 다음 프로그램의 실행 결과는? _ 2

```
#include <stdio.h>

void rec(int n) {
  if (n == 0) return;
  printf("%d", n); //rec(5) -> rec(4) -> rec(3) -> rec(2) -> rec(1) ->
  rec(n - 1);
  printf("%d", n); //rec(1) -> rec(2) -> rec(3) -> rec(4) -> rec(5)
}

int main() {
```

```
  rec(5);
  return 0;
}
```

```
실행 결과

5432112345
```

- 문자열을 매개변수로 받아 그 문자열에서 공백을 제거하여 출력하는 함수를 작성해보세요

```
#include <stdio.h>

void print_noSpace(char str[]) {
  for (int i = 0; str[i] != '\0'; i++) { //문자열의 끝(NULL)까지 문자를 하나하나 받아옴
    if(str[i] != ' ') //공백이 아니라면
      printf("%c", str[i]); //출력
  }
}

int main() {
  print_noSpace("Hello, World\n");
  print_noSpace("My name");

  return 0;
}
```

```
실행 결과

Hello,World
Myname
```