KG 아이티뱅크

C 언어

C LANGUAGE

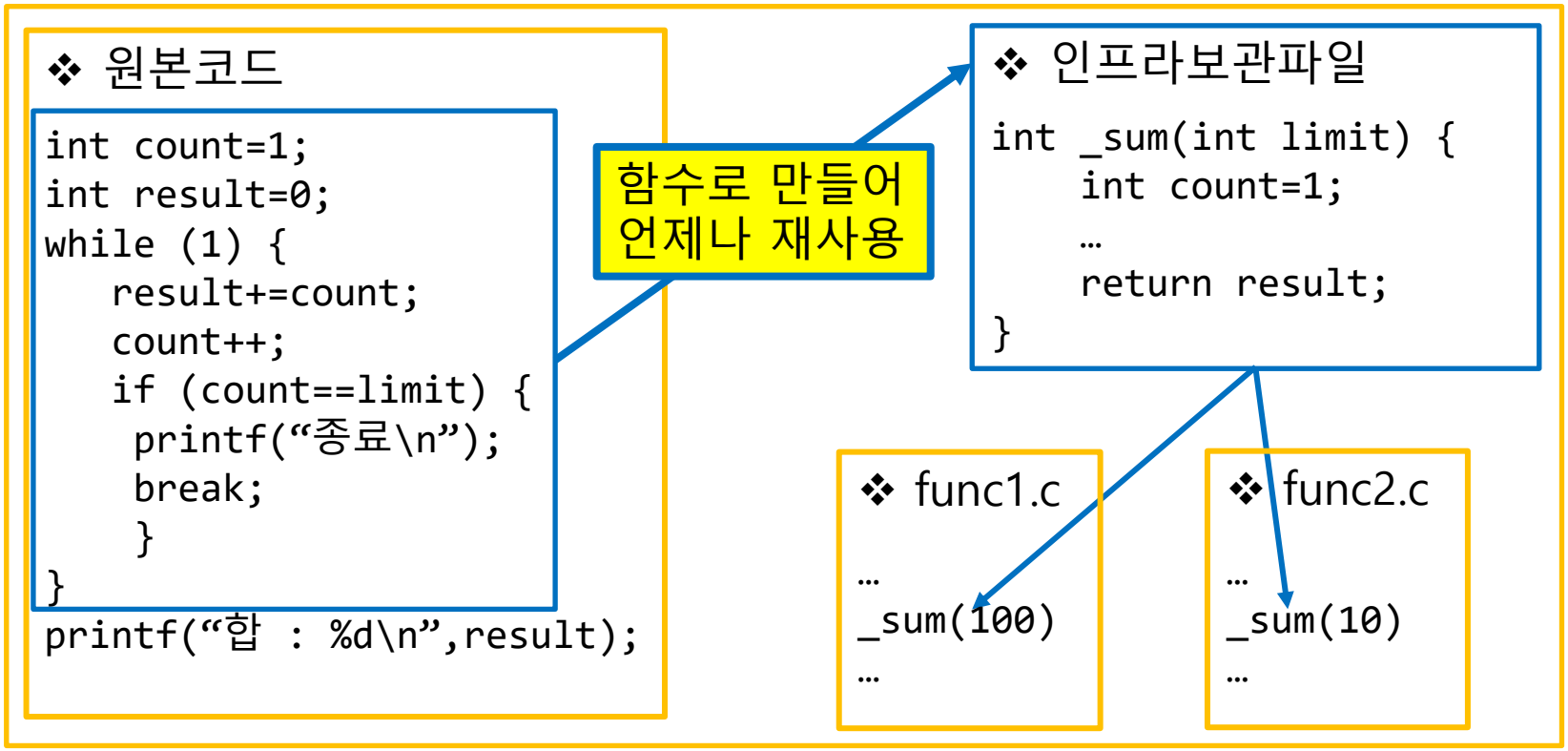
함수

함수(Function)

❖ **하나의 작업을 수행하도록 구성한 코드의 집합**

- **불규칙적으로 반복되는 같은 코드**를 줄이기 위한 방법
- 특정 작업을 하는 코드의 집합에 **이름을 붙여 사용하는 것**
- **나만의 인프라를 구축하여 작업효율을 올리기 위한 것**

❖ **개념도**



함수(Function)

❖ 함수의 필수조건

- **하나의 기능**으로 구성 : 다양한 기능을 수행하지 않음
- **독립적**인 코드 : 다른 코드에 종속되지 않음
- **재사용**이 가능 : 한번만 쓰이는 코드는 의미 없음

❖ 함수의 특징

- 다른 코드와 **격리된 공간**을 보유 : 공간을 **공유하지 않음**
  - 작성된 내용은 코드영역에 저장
  - **내부에서 사용되는 변수는 STACK영역을 사용**
- 격리된 공간을 오가기 위한 **출구와 입구**가 존재

❖ 함수의 장점

- 기능단위로 구분되어 코드의 가독성이 상승
- 에러 및 버그의 관리 및 해결이 용이해짐

함수(Function)

❖ **함수를 정의하다 : 원하는 기능으로 작동하도록 코드를 작성**

- 정의할 때 **main 함수 외부의 상단에 순차적으로 작성**
- **main 내부에서는 절대로 생성하지 않음**

```
  ①      ②        ③
자료형 함수명(매개변수) {
    종속문장1;
    종속문장2;  ④
  ⑤ return 처리결과;
}
```

❖ 코드

```
int print_num(int num) {
    printf("PRINT!\n");
    printf("%d\n", num);
    return num+5;
}
```

① 함수가 실행후 되돌려주는 값의 자료형
② 함수명이며, 변수명 규칙과 동일
③ 전달되는 값을 받아서 함수 내에서 사용하기 위한 변수
④ 해당 함수가 원하는 작업을 하도록 사용자가 직접 작성하는 영역
⑤ 해당 함수가 실행된 후 반환하는 값

함수(Function)

❖ **함수를 호출하다 : 함수를 불러와 사용하는 것**

- 사용시 새롭게 생성하는 함수에서도 사용할 수 있음
- 사용된 지점에서 자신에게 정의된 처리과정을 수행

```
 ①      ②      ③
변수 = 함수명(전달인자);
  ④
함수명(전달인자);
```

❖ 코드

```
int result=print_num(5);

print_num(5);
```

① return이 설정되어 있을 경우 처리결과를 활용하기 위한 변수
② 사용하고자 하는 함수의 이름
③ 매개변수를 초기화시키기 위한 값
④ return이 없을 경우의 함수 사용

함수(Function)

❖ **함수에는 어느정도 준수하는 형식이 존재**

- **반드시 지켜야 하는 것은 아니며 변동될 수 있음**

① **매개변수 있음, return 있음**

❖ 값의 연산 혹은 처리 **과정**을 함수로 만들 때 이용

❖ 연산이나 처리는 **연결되는 과정**이기 때문에 둘 다 필요

② **매개변수 있음, return 없음**

❖ 값의 연산 혹은 처리 **결과**를 함수로 만들 때 이용

❖ return이 없어서 반드시 **자체적인 출력**이 필요함

③ **매개변수 없음, return 있음**

❖ 값의 연산 혹은 처리 **시작**을 함수로 만들 때 이용

❖ 매개변수가 없어 **자체적으로 값을 준비하거나 입력**을 받음

④ **매개변수 없음, return 없음**

❖ 지나치게 긴 코드를 줄이기 위한 용도.

함수(Function)

< 파일이름 : functionEX1.c >

❖ **실습예제1. 아래의 조건을 달성하는 코드를 작성하세요.**

조건

1. 두 수의 합을 구하는 함수를 정의하세요.
2. 두 수의 차를 구하는 함수를 정의하세요.
3. 두 수의 곱을 구하는 함수를 정의하세요.
4. 두 수의 몫을 구하는 함수를 정의하세요.

*위의 함수들은 모두 매개변수와 return이 있습니다.

결과

```
--출력--
_sum((값1),(값2)) : (결과1) // 합
_sub((값1),(값2)) : (결과2) // 차
_mul((값1),(값2)) : (결과3) // 곱
_div((값1),(값2)) : (결과4) // 몫
```

함수(Function)

< 파일이름 : functionEX2.c >

❖ 실습예제2. 아래의 조건을 달성하는 코드를 작성하세요.

조건

1. 두 수의 비교해서 더 큰 수를 출력하는 함수를 정의하세요.
2. 하나의 정수에 대하여 절대값을 출력하는 함수를 정의하세요.
3. 두 수의 몫, 나머지, 나누기를 출력하는 함수를 정의하세요.

*위의 함수들은 모두 매개변수는 있고, return이 없습니다.

결과(값1 : 6, 값2 : 3, 값3 : -8로 가정)

```
--출력--
(값1)이 더 큽니다.                    // _comp((값1),(값2))
(값3)의 절대값은 (절대값3) 입니다. // _abs_num((값3))
결과 : (몫), (나머지), (나누기)    // _div3((값1), (값2))
```

함수(Function)

< 파일이름 : functionEX3.c >

❖ **실습예제3. 아래의 조건을 달성하는 코드를 작성하세요.**

조건

1. 값을 입력을 받아 변수를 초기화시킬 수 있는 함수를 정의하세요.
   문자, 정수, 실수 3가지를 구분하여 만듭니다.
2. 1부터 지정한 정수까지의 합을 구하는 함수를 정의하세요.

*모두 매개변수가 없고, return이 있습니다.

결과

```
char ch=get_char()          : (문자값)
int num=get_num()           : (정수값)
double fnum=get_fnum()      : (실수값)
int result=get_sum()        : (결과값)
```

함수(Function)

< 파일이름 : functionEX4.c >

❖ **실습예제4. 아래의 조건을 달성하는 코드를 작성하세요.**

조건

1. 소문자를 대문자로 변경하는 함수를 정의하세요.
   소문자는 97, 대문자는 65부터 시작합니다.

2. 지정한 횟수만큼 입력을 받아 평균을 구하는
   함수를 정의하세요.

*매개변수 / return 모두 임의로 구성합니다.
*함수의 필수조건을 모두 만족해야 합니다.

결과

지정한 형식에 따른 결과가 나옵니다.
직접 구성하세요.