KG 아이티뱅크

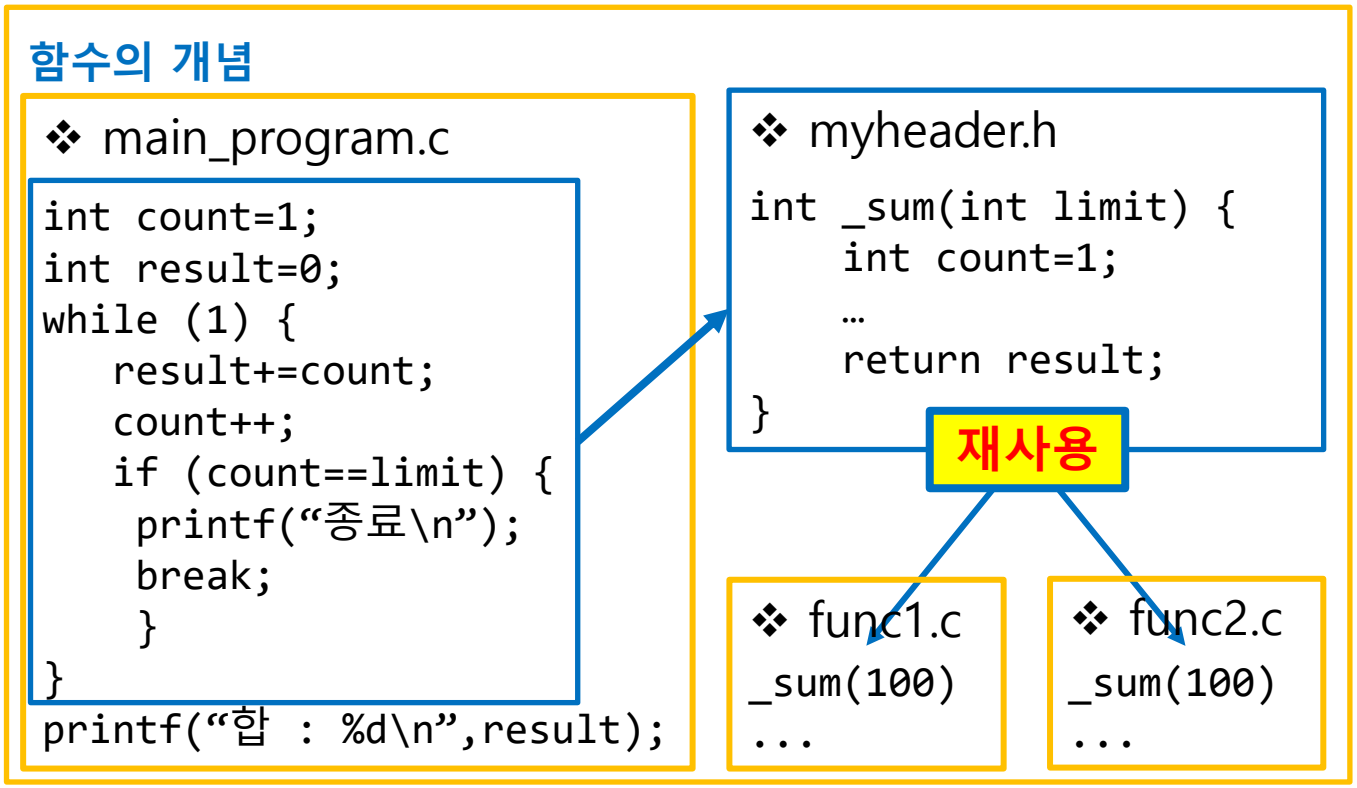
C 언어

C LANGUAGE

함수

❖ 코드를 변수에 저장하여 사용하기 위한 문법

- 불규칙적이지만 반복적으로 나타나는 코드를 저장
- 저장된 코드를 사용하여 반복되는 코드 작성을 줄임
- **기본적인 접근은 변수와 동일하나, 더 넓게 봐야 함**



함수

❖ **함수를 선언하고, 값을 저장하는 과정 : 함수를 정의한다**

- 값이 아닌 **코드를 저장하며, 코드는 실행**되어야 함
- 실행되는 내용에 따라, **특정 동작을 수행**하게 됨
- 실제로는 변수의 선언 및 초기화와 비슷한 과정

함수의 정의

```
void HELLO(void) {
   printf("HELLO!");
}
```

구성요소

```
②      ③
반환형 함수명(매개변수) {
①   종속코드;
   ④
}
```

① **반환형** : 함수가 만들어내는 값의 자료형을 의미

② **함수명** : 동사로 이름을 붙이는 편이 좋음

③ **매개변수** : 외부에서 주는 값을 받기 위한 변수

④ **종속문장** : 함수에 저장하여 **실행시킬 예정**인 코드

함수

❖ **함수를 이용하여, 값을 불러오는 과정 : 함수를 불러온다**

- **함수명만으로는 저장된 코드를 말 그대로 불러만 옴**
- 불러온 값을 실행시켜야 하며, 이를 위해 ( ) 를 붙임

❖ **함수로 불러온 코드를 실행하는 과정 : 함수를 호출한다**

- 불러온 값이 **코드라면 이를 실행하는 연산자 ( )** 를 붙임
- 함수가 작성된 곳에서 실행되며, 초점(Focus)이 옮겨짐

함수의 호출

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

불러오기 : 기본적으로 쓸모없음

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

구성요소

```
자료형 사용위치( ) {
    함수명(전달인자);
}
```

① ② ③

① **사용위치** : 다른 함수에서만 가능

② **함수명** : 불러올 함수를 구별

③ **전달인자** : 필요한 값을 배치

❖ **가장 기본이 되는 형태 : 매개변수가 없고 반환형이 없음**

➢ 이러한 함수를 4형식이라고 편의상 명명하여 구분함

4형식함수 예시

```
void main_program(void) {
    int num1;
    int num2;
    printf("정수 2개 입력 >> ");
    scanf_s("%d%d", &num1, &num2);
    printf("두 정수의 합 : %d\n",num1 + num2);
}
```

❖ **4형식 함수의 특징 : 외부와 완전히 차단된 독립된 코드**

- 독립된 코드를 정리하여 가독성을 높이기 위한 용도
- 프로그램 내에 다른 프로그램을 만들어서 구분하는 용도

✓ **함수 내부에 저장된 코드에서 모든 것이 완결되어야 함**

함수

< 파일이름 : 08. 함수_EX1.c >

실습문제1. 아래의 조건을 만족하는 코드를 작성하세요.

✓ **조건**

1. 2개의 정수를 입력을 받아 합/차/곱/몫을 출력하는 함수를 정의하세요.
2. 몫을 구할 때 연산이 안되는 경우의 수는 <연산불가>라고 출력합니다.
3. 함수명은 **calculator**로 합니다.

✓ **결과(10과 3을 입력했을 경우)**

```
// calculator(); 만 작성하여 실행
--입력--
정수 2개 입력 >> 10 3
--출력--
합 : 13
차 : 7
곱 : 30
몫 : 3  // 또는 연산불가로 출력
```

❖ **외부에 간섭할 수 있음 : 매개변수가 없고 반환형이 있음**

➢ 이러한 함수를 3형식이라고 편의상 명명하여 구분함

3형식함수 예시

```
int get_data(void) {
    int num1;
    int num2;
    printf("정수 2개 입력 >> ");
    scanf_s("%d%d", &num1, &num2);
    return num1 + num2;
}
```

❖ **3형식 함수의 특징 : 반환형과 return 명령어가 작성됨**

- 외부로부터 필요한 것들은 관상용이 아니면 입력을 받음
- 입력을 받아 준비된 것을 외부로 복사하여 전달함

✓ **함수가 사용된 곳에 지정한 변수/연산식의 값이 복사됨**

함수

< 파일이름 : 08. 함수_EX2.c >

실습문제2. 아래의 조건을 만족하는 코드를 작성하세요.

✓ **조건**

1. 정수 하나를 입력을 받아 1부터 해당 정수까지 1씩 증가하는 정수들의 합을 구합니다.
2. 구한 합을 외부에 복사하는 함수를 정의하세요.
3. 함수에서 결과를 출력하지 않습니다.
4. 함수명은 **get_sum**으로 합니다.

✓ **결과(15를 입력했을 경우)**

```
--입력--
// int result = get_sum();
합을 구할려는 정수 입력 >> 15

--출력--
// printf("함수에서 구한 합 : %d\n",result);
함수에서 구한 합 : 120
```

❖ **외부가 함수에 간섭함 : 매개변수가 있고 반환형이 없음**

➢ 이러한 함수를 2형식이라고 편의상 명명하여 구분함

2형식함수 예시

```
void show_result(int num1, int num2) {
    printf("두 정수의 합 : %d\n",num1 + num2);
}
```

❖ **2형식 함수의 특징 : 매개변수가 작성되어 값을 복사함**

- 함수를 호출할 때 넣어주는 값을 받는 변수
- 외부에서 넣어준 것은 **값만 복사**되고, 변수는 그대로 있음

✓ **호출될 때 초기화가 되는 변수를 준비하여 값을 이용함**

함수

< 파일이름 : 08. 함수_EX3.c >

실습문제3. 아래의 조건을 만족하는 코드를 작성하세요.

✓ **조건**

1. 두 정수의 크기를 비교하여 더 큰 값을 출력하세요.
2. 함수에서 입력을 받지 않습니다.
3. 함수명은 **show_bigger**로 합니다.

✓ **결과**

```
--입력-- // 필요하시면 함수 외부에 추가하세요.

--출력--
제일 큰 값     : 15   // show_bigger(15, 4);
제일 큰 값     : 15   // show_bigger(4, 15);
서로 같은 값   : 4    // show_bigger(4, 4);
```

❖ **중간과정으로 바꿔 씀 : 매개변수가 있고 반환형이 있음**

➢ 이러한 함수를 1형식이라고 편의상 명명하여 구분함

1형식함수 예시

```
int make_sum(int num1, int num2) {
    return num1 + num2;
}
```

❖ **1형식 함수의 특징 : 프로그램의 처리를 담당하는 함수**

- 오로지 연산만 하며, 필요하면 입/출력을 수행함
- 특수한 경우(안내)가 아니면 입/출력은 수행하지 않음
- 일반적으로 복잡한 코드를 만들며 단순연산(위)은 안 만듬
- 함수를 만들면서 가장 많이 나타나게 되는 형식

✓ **중간과정으로 바꿔 쓰지만, 중간과정은 많은 의미를 가짐**

함수

< 파일이름 : 08. 함수_EX4.c >

실습문제4. 아래의 조건을 만족하는 코드를 작성하세요.

✓ **조건**

1. 짝수는 1만큼 증가, 홀수는 1만큼 감소한 값이 나옵니다.
2. 함수에서 입력/출력 안합니다.
3. 함수명은 **change**입니다.

✓ **결과**

```
--입력-- // 필요하시면 함수 외부에 추가하세요.
--출력-- // 함수 외부에서 출력됩니다.
결과1 : 12  // printf("결과1 : %d\n",change(13));
결과1 : 17  // printf("결과2 : %d\n",change(16));
결과3 : -15 // printf("결과3 : %d\n",change(-16));
결과4 : -14 // printf("결과4 : %d\n",change(-13));
```

< 파일이름 : 08. 함수_EX5.c >

실습문제5. 아래의 조건을 만족하는 코드를 작성하세요.

✓ **조건**

1. 정수값 2개를 이용하여 거듭제곱 연산 결과를 줍니다.
2. 첫번째 정수에 대하여 두번째 정수만큼 거듭제곱합니다.
3. 거듭제곱수가 0이하이면 1로 통일합니다.
4. 함수명은 **make_exp**이며 함수에서는 입출력을 하지 않습니다.

✓ **결과**

```
--입력-- // 필요하시면 함수 외부에 추가하세요.
--출력-- // 함수 외부에서 출력됩니다.
결과1 : 16 // printf("결과1 : %d\n", make_exp(2,4));
결과2 : 81 // printf("결과2 : %d\n", make_exp(3,4));
결과3 : 1  // printf("결과3 : %d\n", make_exp(9,0));
결과4 : 64 // printf("결과4 : %d\n", make_exp(-2,6));
결과5 : 1  // printf("결과5 : %d\n", make_exp(2,-5));
```