| 프로그래밍 | **8. 함수의 정의와 사용 : 반환타입 void편** | 일자: 2024. . . |
|---|---|---|
| 학습 목표 | ☐ 함수의 이해<br>☐ 사용자 정의 함수의 사용 (1) | 하이디: |

## 1. 함수

### 1) 함수의 이해

① 프로그래밍에서의 함수(function)란 일정한 작업을 수행하기 위해 독립적으로 설계된 프로그램 코드의 집합이다.

② C언어로 구현되는 모든 프로그램의 기본 단위는 ____이다.

③ 함수의 장점은 아래와 같다.

- 함수를 재사용하여 코드의 반복을 피할 수 있다. 특정 작업을 여러 번 반복해야 할 때, 해당 작업을 관리하는 코드를 함수로 만들어 여러 번 호출하면 코드의 반복을 최소화하면서도 작업을 여러 번 수행할 수 있다.
- 프로그램을 여러 개의 함수로 나누어 모듈화하면 전체적인 코드의 가독성이 좋아진다.
- 프로그램의 유지보수 측면에서 여러 번 쓰인 코드를 각각 수정하는 것보다 함수 하나를 수정하는 것이 더 편리하다.

### 2) 표준 함수와 사용자 정의 함수

① 표준 함수는 printf, scanf와 같이 **그 기능과 사용법이 C언어의 표준**으로 정의되어 있으며, 라이브러리로서 포함되어 사용자가 불러서 쓸 수 있는 함수이다. 표준 함수는 각종 헤더 파일에 수록되어 있기에 _____을 사용하여 해당 헤더 파일을 미리 편입한 후에야 사용할 수 있다.

② 사용자 정의 함수는 C언어에서 미리 제공되는 것이 아니라 **사용자가 직접 정의하여 사용**하는 함수이다.

## 2. 사용자 정의 함수의 사용

### 1) 값을 반환하지 않는 함수의 정의

```
void[1] calcComb[2](int n, int m, char ch)[3] // 함수의 헤더 (반환타입 + 함수명 + 매개변수)
{
    함수의 바디[4]
}
```

① 반환 타입(return type): 함수의 반환값이 없을 때는 반환 타입에 **void**를 적는다.

② 함수 이름: 함수를 구분하고 호출하기 위한 이름이다.

③ 매개변수 목록(parameters): 함수 호출 시에 전달된 인수의 값을 저장할 변수의 타입과 이름을 순서대로 명시한 것이다. 함수의 매개변수가 없을 때는 괄호 안에 void를 적거나, 괄호만 두고 내용을 생략하여 나타낸다.

④ 함수 몸체(body): 함수의 고유 기능을 수행할 코드들의 집합이다. _____로 묶인다.

- **사용자 정의 함수를 사용하기 위해서는 main 함수 전에 정의하여 컴파일러에게 함수에 대한 정보를 전해주어야 한다!** 그러지 않으면 컴파일러가 함수 호출에 대응하지 못하고 에러가 발생한다.
- 그러나 컴파일러에게 먼저 함수에 대한 정보를 제공할 수 있다면, 그 함수는 main 함수 뒤에 정의해도 문제가 없을 것이다.

**2) 함수의 원형 선언**

```
void calcComb1(int n, int m, char ch);
void calcComb2(int, int, char);
```

① 함수의 원형은 헤더와 동일하게 반환 타입, 함수 이름, 매개변수 목록을 적는다.
② 단, **매개변수 목록에서 매개변수의 이름은 포함되지 않아도 된다.**

**3) 값을 반환하지 않는 함수의 호출**

```
calcComb[1](n, 5, 'A')[2];
```

① 함수 이름을 불러 어떤 함수를 호출할 것인지 지정한다.
② 인수(인자): 함수의 매개변수로 전달해 줄 값을 적는다. ____ 연산자를 이용해 여러 개의 인수를 전달하는 것이 가능하다. 함수로 전달할 인수가 없을 때는 괄호만 두고 내용을 생략하여 나타낸다.

## 3. 개념 확인 문제

**1) 아래 코드를 읽으며, 코드의 흐름을 따라가 보아라.**

```c
#include <stdio.h>

void printHi();
void sum(int n, int m);

int main(void) {
    int n = 5, m = 10;

    printHiHello();
    printSum(n, m);

    return 0;
}
```

```c
void printHiHello() {
    printf("Hi\n");
    return;
    printf("Hello\n");
    return;
}

void printSum(int n, int m) {
    printf("%d\n", n + m);
}
```

- 함수의 return문은 함수를 종료하고 _____으로 돌아가게 한다.
- return문이 없으면 닫는 중괄호까지 모든 문장을 실행하고 호출한 곳으로 돌아가게 된다.

## 4. 실습

### 1) LAB8_1 (매개변수 없는 void 함수)

아래 함수 헤더를 보고 기능을 추측해 정의하여, 우측과 같이 실행되도록 하라.

```
*****
*****
*****
```

```
void print5Stars(void)
```

### 2) LAB8_2 (매개변수를 가지는 void 함수)

정수를 하나 받아 해당 정수를 단으로 갖는 구구단을 출력하는 프로그램을 작성하라. 이때 구구단 출력부는 함수화하라.

```
몇 단을 출력할까요? : 7
구구단 7단의 출력입니다
7 * 1 = 7
7 * 2 = 14
7 * 3 = 21
7 * 4 = 28
7 * 5 = 35
7 * 6 = 42
7 * 7 = 49
7 * 8 = 56
7 * 9 = 63
```

```
몇 단을 출력할까요? : 9
구구단 9단의 출력입니다
9 * 1 = 9
9 * 2 = 18
9 * 3 = 27
9 * 4 = 36
9 * 5 = 45
9 * 6 = 54
9 * 7 = 63
9 * 8 = 72
9 * 9 = 81
```

### 3) LAB8_3 (매개변수를 가지는 void 함수)

다음의 헤더를 보고 기능을 추측해 함수를 정의하여 아래와 같은 실행결과를 갖도록 하라.

```
void printManyStars(int num)
```

```
Enter the number of *: 5
*****
```

### 4) LAB8_4 (타입이 같은 매개변수를 여러 개 가지는 void 함수)

정수를 세 개 입력받아 가장 큰 수를 출력하는 프로그램을 작성하라. 정수들을 비교하고 가장 큰 정수를 출력하는 기능을 함수화하라.

```
Enter three numbers: 5 2 9
9
```

```
Enter three numbers: 1 8 8
8
```

### 5) LAB8_5 (타입이 다른 매개변수를 여러 개 가지는 void 함수)

LAB8_3을 수정하여 * 뿐만이 아닌 원하는 문자를 원하는 수만큼 출력하게 하라.

```
Enter a character to print:+
Enter the number of characters:7
+++++++
```

```
Enter a character to print:0
Enter the number of characters:1
0
```

### 6) LAB8_6 (매개변수를 가지는 void 함수)

n개의 피보나치 수열 값을 출력하는 프로그램을 작성하라.

```
void printFibonacci(int n) // (n + 1)번째 피보나치 값을 출력한다
```

힌트는 아래와 같다.

- 피보나치 수열에서 n번째 수는 n - 1번째 수와 n - 2번째 수의 합이다.

```
몇 개의 피보나치 수열 값을 출력할까요? (3보다 큰 정수): 13
1 1 2 3 5 8 13 21 34 55 89 144 233
```

**7) LAB8_7 (매개변수를 가지는 void 함수)**

정수 n을 입력받아 다음의 규칙에 집어넣고 n이 1이 될 때까지 반복하며 변화하는 n의 값과 총 변화 횟수를 출력하여라.

- n이 짝수인 경우, n을 2로 나눈다. 그리하여 나온 값을 다시 n에 대입한다.
- n이 홀수인 경우, n에 3을 곱한 값에서 1을 더하여 나온 값을 다시 n에 대입한다.

```
Enter a number: 10
10 5 16 8 4 2 1
길이는 7
```

```
Enter a number: 6
6 3 10 5 16 8 4 2 1
길이는 9
```