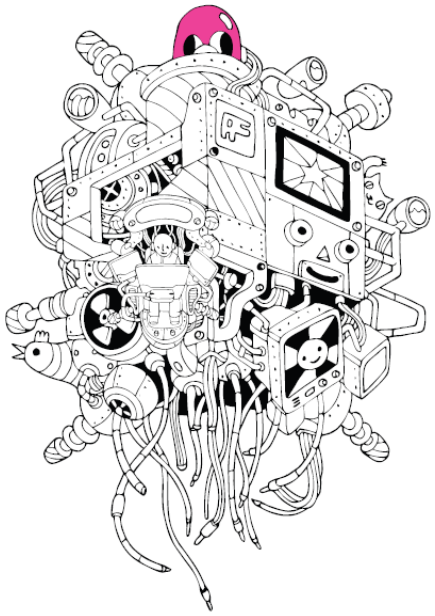
C 프로그래밍



윤성우 저 열혈강의 C 프로그래밍 개정판

Chapter 02. 프로그램의 기본구성

C 프로그래밍

Chapter 02-1. Hello world! 들여다보기

Chapter 02. 프로그램의 기본구성

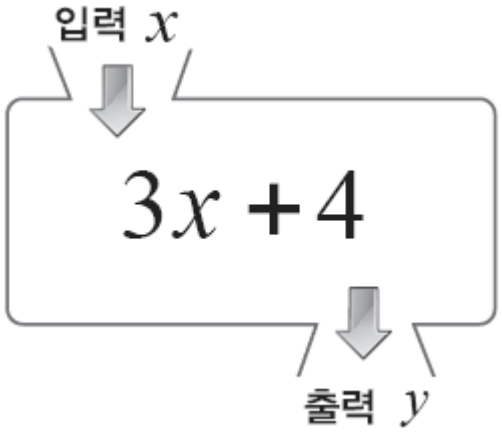
C언어의 기본단위인 '함수'의 이해

√ C언어의 기본단위는 함수이다.

함수를 만들고, 만들어진 함수의 실행순서를 결정하는 것이 C언어로 프로그램을 작성하는 것이다.

√ 함수의 기본특성

수학적으로 함수에는 입력과 출력이 존재한다.



√ C언어의 함수

C언어의 함수에도 입력과 출력이 존재한다.

√ C언어의 함수와 관련된 용어의 정리

- 함수의 정의 만들어진 함수, 실행이 가능한 함수를 일컬음
- 함수의 호출 함수의 실행을 명령하는 행위
- 인자의 전달 함수의 실행을 명령할 때 전달하는 입력 값

C언어는 함수로 시작해서 함수로 끝이 난다.

예제 Hello.c에서의 함수는 어디에?

√ 프로그램의 시작

첫 번째 함수가 호출이 되면서 프로그램은 시작이 된다.

√ 제일 먼저 호출되는 함수는?

main이라는 이름의 함수! 따라서 C언어로 구현된 모든 프로그램은 시작점에 해당하는 main이라는 이름의 함수를 반드시 정의해야 한다.

즉, main이라는 이름의 함수가 자동으로 호출이 되면서 프로그램은 실행된다.

√ 함수의 기능

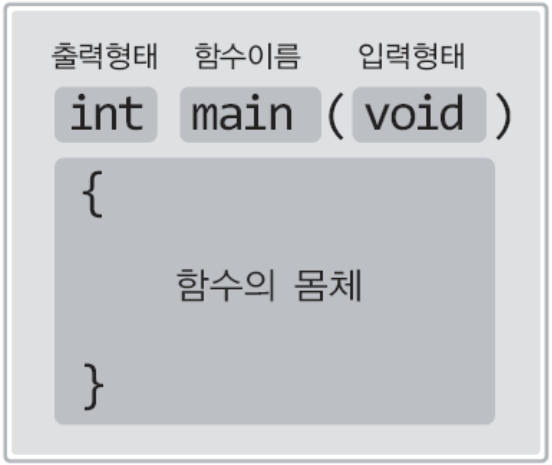
함수의 기능은 중괄호 안에 표현이 되며, 중괄호 안에 표현된 함수의 기능을 가리켜 함수의 몸체라 한다.

√ C언어의 함수에 표시가 되는 세 가지

· 함수의 이름 함수를 호출할 때 사용하게 되는 이름

· 출력형태 실행의 결과! 일반적으로 반환형(return type)이라 한다.

· 입력형태 함수를 호출할 때 전달하는 입력 값의 형태




```
int main(void)
{
    printf("Hello world! \n");
    return 0;
}
```

순차적으로 실행

세미콜론

√ 함수 내에 존재하는 문장의 끝에는 세미콜론 문자 ; 을 붙여준다.

세미콜론은 문장의 끝을 표현하기 위한 문자이다.

√ 열 줄에 표현된 코드는 열 개의 문장인가?

하나의 문장이 둘 이상의 줄에 표시될 수도 있고, 한 줄에 둘 이상의 문장이 표시될 수도 있다.

즉, 줄 바꿈은 문장의 바뀜을 뜻하는 것이 아니다.

√ 한 줄에 하나의 문장을 표시하는 것이 가장 일반적이고 또 보기도 좋다.

다음 세 main 함수는 모두 동일한 프로그램이다. 줄 바꿈의 차이가 프로그램의 차이로 이어지지 않는다.

```
int main(void)
{
    printf("Hello world! ₩n"); return 0;
}
```

```
int main(void)
{
    printf("Hello world! ₩n");
    return 0;
}
```

```
int main(void) { printf("Hello world! ₩n"); return 0; }
```

소스코드의 세세한 분석

```
#include <stdio.h>
int main(void)
{
    printf("Hello world! ₩n");
    return 0;
}
```

헤더파일 선언문

처음 보는 함수의 호출문

√ 표준함수

이미 만들어져서 기본적으로 제공이 되는 함수!

printf 함수는 표준함수이다.

√ 표준 라이브러리

표준함수들의 모임을 뜻하는 말이다.

즉, printf 함수는 표준 라이브러리의 일부이다

#include <stdio.h>

- stdio.h 파일의 내용을 이 위치에 가져다 놓으라는 뜻
- printf 함수의 호출을 위해서 선언해야 하는 문장
- stdio.h 파일에는 printf 함수호출에 필요한 정보 존재

printf("Hello world! ₩n");

- printf라는 이름의 함수를 호출하는 문장
- 인자는 문자열 "Hello world! ₩n"
- 인자는 소괄호를 통해서 해당 함수에 전달이 된다.

return 0;

- 함수를 호출한 영역으로 값을 전달(반환)
- 현재 실행중인 함수의 종료

C 프로그래밍

Chapter 02-2. 주석이 들어가야 완성된 프로그램

Chapter 02. 프로그램의 기본구성

주석의 필요성과 블록단위 주석

√ 주석의 이해

주석은 소스코드에 삽입된 메모를 뜻한다. 이는 컴파일의 대상에서 제외가 되기 때문에 주석의 유무는 컴파일 및 실행의 결과에 영향을 미치지 않는다.

√ 주석의 필요성

코드의 분석은 글을 읽는 것 만큼 간단하지 않다. 때문에 코드를 분석해야 하는 남을 위해서, 그리고 코드를 작성한 작성자 스스로를 위해서라도 코드에 대한 설명인 주석을 간단히나마 달아놓을 필요가 있다. 즉 주석은 선택이 아닌 필수이다.

√ 블록 단위 주석

한 행의 주석처리

```
/* 주석처리 된 문장 */
```

```
/*
    주석처리 된 문장1
    주석처리 된 문장2
    주석처리 된 문장3
*/
```

여러 행의 주석처리

√ 행 단위 주석

```
// 주석처리 된 문장1
// 주석처리 된 문장2
// 주석처리 된 문장3
```

한 행 단위로의 주석처리

주석을 다는 방식은
프로젝트 별로 팀원과 상의하여 결정하게 된다.

주석 처리의 예

```
/*
제 목: Hello world 출력하기
기 능: 문자열의 출력
파일이름: HelloComment.c
수정날짜: 2014. 07. 15
작성자: 윤성우
*/
#include <stdio.h>    // 헤더파일 선언

int main(void)    // main 함수의 시작
{
    /*
    이 함수 내에서는 하나의 문자열을 출력한다.
    문자열은 모니터로 출력된다.
    */
    printf("Hello world! \n");    // 문자열의 출력
    return 0;    // 0의 반환
}    // main 함수의 끝
```

과도하게 처리된 주석(주석도 과하면 좋지 않다)!
주석을 다는 방법을 소개하기 위한 예제일 뿐이다.

주석처리에 있어서의 주의점

```
1.  /*
2.      주석처리 된 문장1
3.      /* 단일 행 주석처리 */
4.      주석처리 된 문장2
5.  */
```

잘못 달린 주석(컴파일 시 오류 발생)

```
1.  /*
2.      주석처리 된 문장1
3.      // 단일 행 주석처리
4.      주석처리 된 문장2
5.  */
```

잘 달린 주석(컴파일 시 오류 발생하지 않음)

주석을 달다 보면 주석이 겹치는(중첩되는) 경우가 발생하기도 한다. 그런데 블록 단위 주석은 겹치는 형태로 달 수 없다.

윤성우의

열혈 C 프로그래밍

Chapter 02-3. printf 함수의 기본적인 이해

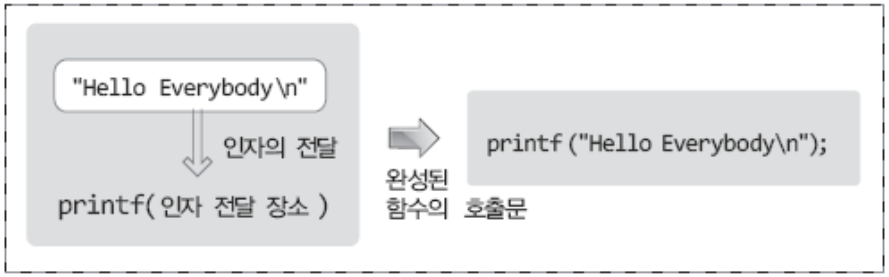
Chapter 02. 프로그램의 기본구성

printf 함수를 이용한 정수의 출력

```
int main(void)
{
    printf("Hello Everybody\n");
    printf("%d\n", 1234);
    printf("%d %d\n", 10, 20);
    return 0;
}
```

실행결과

```
Hello Everybody
1234
10 20
```



%d

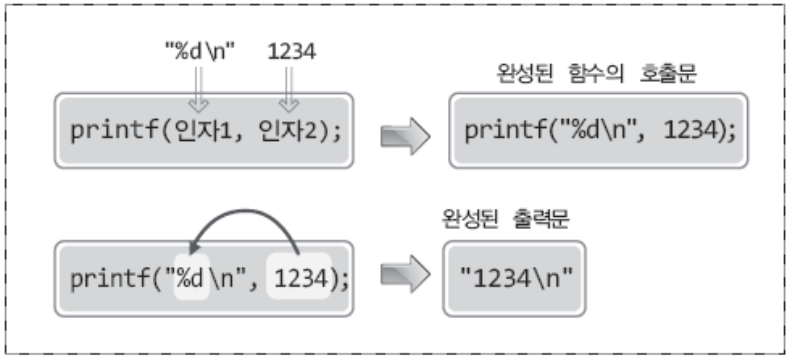
- 문자열에 삽입된 %d를 가리켜 '서식문자'라 한다.
- 서식문자는 출력의 형태를 지정하는 용도로 사용이 된다.
- %d는 부호가 있는 10진수 정수의 형태로 출력하라는 의미를 담는다!

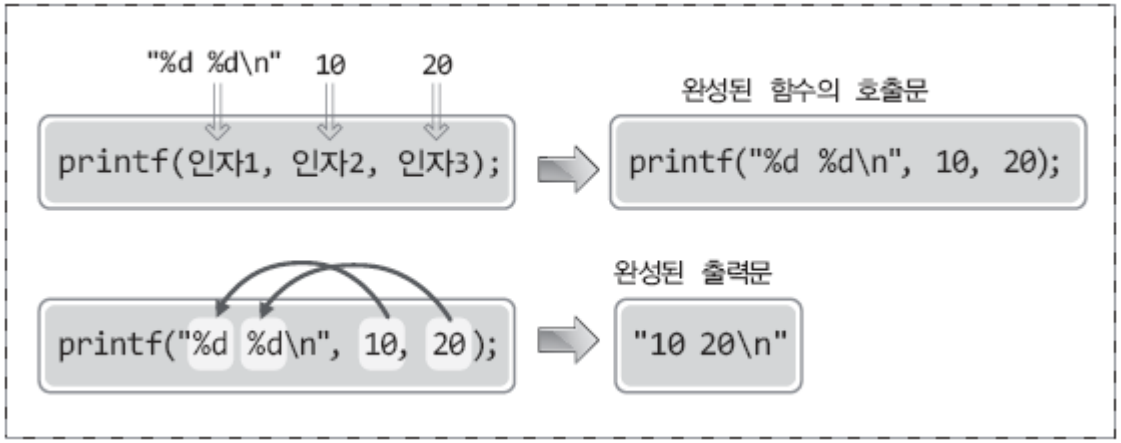
₩n은 이스케이프 시퀀스(escape sequence) 또는 특수문자라 불리며 개 행을 의미하는 용도로 사용된다.

출력의 대상은?

- 큰 따옴표로 표시되는 문자열의 뒤에 이어서 표시를 하며,
- 콤마로 각각을 구분한다.
- 서식문자 %d가 두 개 등장하면, 출력의 대상도 두 개 등장해야 한다.

정수의 출력에 사용된 서식문자 %d



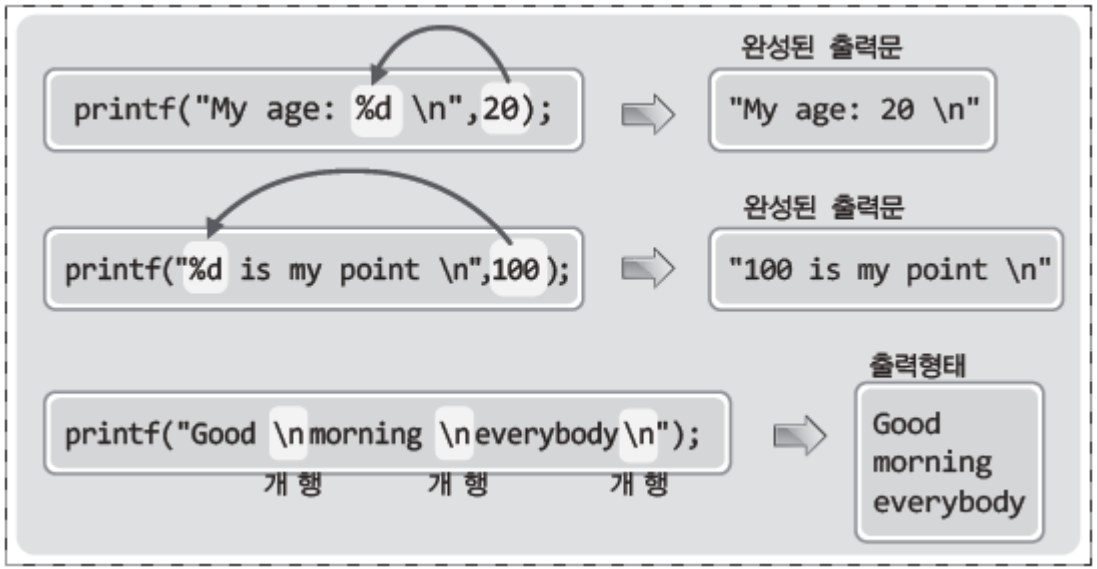


출력의 형태를 다양하게 조합하는 것이 가능하다.

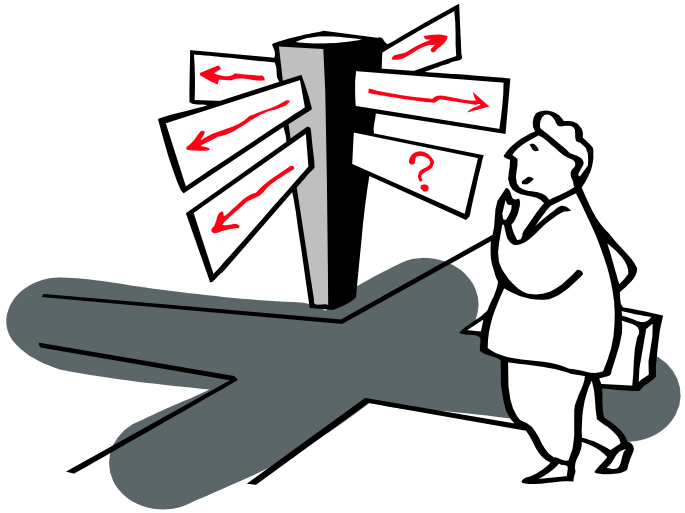
```
int main(void)
{
    printf("My age: %d \n", 20);
    printf("%d is my point \n", 100);
    printf("Good \nmorning \neverybody\n");
    return 0;
}
```

실행결과

```
My age: 20
100 is my point
Good
morning
everybody
```



이후에는 보다 다양한 서식문자를 공부하게 된다. 그리고 그렇게 되면 보다 다양한 형태로 출력의 형태를 조합할 수 있게 된다.



Chapter 02가 끝났습니다. 질문 있으신지요?