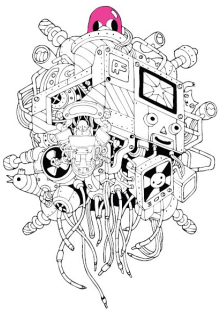
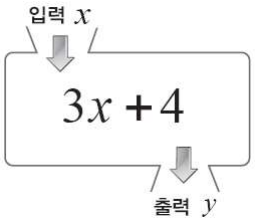


# Ch 2. 프로그램의 기본구성

# Hello world! 들여다보기

## C 언어는 프로그래밍 언어이다.

- 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.

3

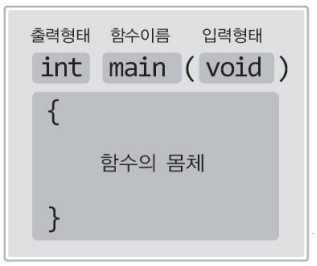
## 예제 hello.c에서의 함수는 어디에?

- 프로그램의 시작
  - 첫 번째 함수가 호출이 되면서 프로그램은 시작이 된다.
- 제일 먼저 호출되는 함수는?
  - main이라는 이름의 함수!
  - C 언어로 구현된 모든 프로그램은 시작점에 해당하는 main이라는 이름의 함수를 반드시 정의해야 한다.
  - main이라는 이름의 함수가 자동으로 호출이 되면서 프로그램 실행 시작

4

## 예제 Hello.c에서의 함수는 어디에?

- 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

5

## 세미콜론 ;

- 함수 내에 존재하는 문장의 끝에는 세미콜론 문자 ; 을 붙여준다.
  - 문장 하나는 명령어 하나를 의미란다.
  - 세미콜론은 문장 (명령)의 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 바꿈을 뜻하는 것이 아니다.
- 한 줄에 하나의 문장을 표시하는 것이 가장 일반적이고 보기도 좋다.
  - 다음 페이지의 세 main 함수는 모두 동일한 프로그램이다.
  - 줄 바꿈의 차이가 프로그램의 차이로 이어지지 않는다.
  - 그러나 첫 번째 모양이 가장 바람직하다.

6

## 세미콜론 ;

```
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; }
```

7

## 소스 코드의 세세한 분석

- #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

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

- 표준함수
  - 이미 만들어져서 기본적으로 제공이 되는 함수!
  - printf 함수는 표준함수이다.
- 표준 라이브러리
  - 표준함수들의 모임을 뜻함
  - 즉, printf 함수는 표준 라이브러리의 일부이다

8

# 주석이 들어가야 완성된 프로그램

## 주석의 필요성과 블록단위 주석

- **주석의 이해**
  - 주석 (comments)은 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 간단하게 라도 달아놓을 필요가 있다.
  - 즉 주석은 선택이 아닌 필수이다.

## 주석의 필요성과 블록단위 주석

- **블록 단위 주석**

한 행의 주석처리

```
/* 주석처리 된 문장 */
```

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

여러 행의 주석처리

- **행 단위 주석**

한 행 단위로의 주석처리

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

**주석을 다는 방식은**
**회사에서는 프로젝트 별로 팀원과 상의하여 결정하게 된다.**

11

## 주석 처리의 예

HelloComment.c

```
/*
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.**

12

## 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.

13

## printf 함수의 기본적인 이해

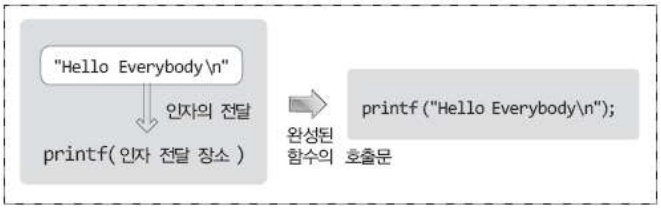
## printf 함수를 이용한 정수의 출력

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

```
Hello Everybody
1234
10 20
```

실행결과

PrintfOne.c




15

## printf 함수를 이용한 정수의 출력

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

```
Hello Everybody
1234
10 20
```

실행결과

- printf 함수의 기본 동작 원칙
  - ₩는 키보드의 ₩와 동일함
  - " " 내에 있는 것을 출력
  - " " 내에 ₩가 있으면 그 뒤에 오는 것이 무엇인가에 따라 다르게 동작 (예, ₩n, ₩t 등)
  - " " 내에 %가 있으면 " " 뒤에 있는 인자로 교체한 후 출력 (예, %d 등)

16

## printf 함수를 이용한 정수의 출력

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

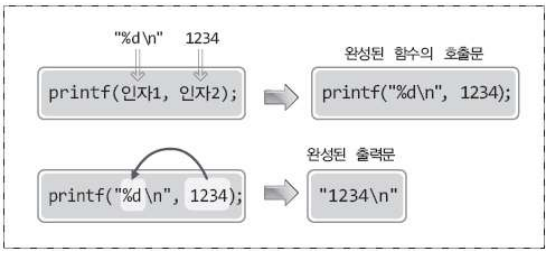
```
Hello Everybody
1234
10 20
```

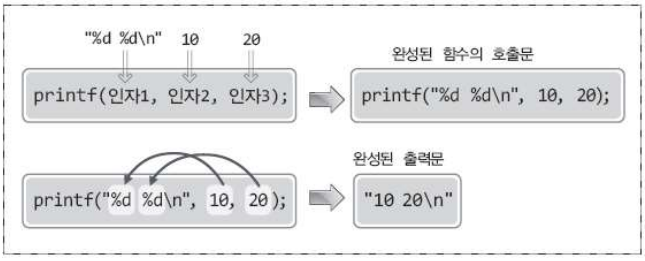
실행결과

- ₩n
  - 이스케이프 시퀀스(escape sequence) 또는 특수문자라 불리며 줄 바꿈을 의미하는 용도로 사용된다.
- %d
  - 문자열에 삽입된 %d를 '서식문자'라 한다.
  - 서식문자는 출력의 형태를 지정하는 용도로 사용이 된다.
  - %d는 부호가 있는 10진수 정수의 형태로 출력하라는 의미를 담는다!
- 출력의 대상은?
  - 큰 따옴표로 표시되는 문자열의 뒤에 이어서 표시를 하며,
  - 콤마로 각각을 구분한다.
  - 서식문자 %d가 두 개 등장하면, 출력의 대상도 두 개 등장해야 한다.

17

## 정수의 출력에 사용된 서식문자 %d




18

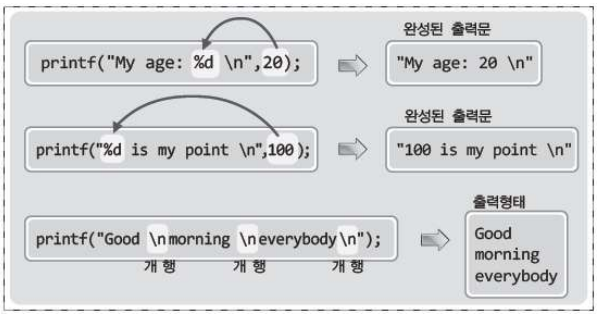
## 출력의 형태를 다양하게 조합하는 것이 가능하다.

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

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

PrintfTwo.c

실행결과




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

19

## 프로그램 실습

- 아래와 같이 출력되도록 프로그래밍 하시오.
  - printf 함수를 이용하여 출력하시오.
  - 단, 모든 숫자는 printf 함수의 " " 안에 적지 말고 밖에 적도록 한다.

나의 이름은 아무개입니다.
나의 학번은 20191234입니다.
나의 생일은 8월 15일입니다.

20

## 정수 값을 저장할 방의 선언

- 변수 (방)
  - 프로그램에서 임시로 자료 값을 저장할 수 있는 저장 장소
- 변수 선언 방법

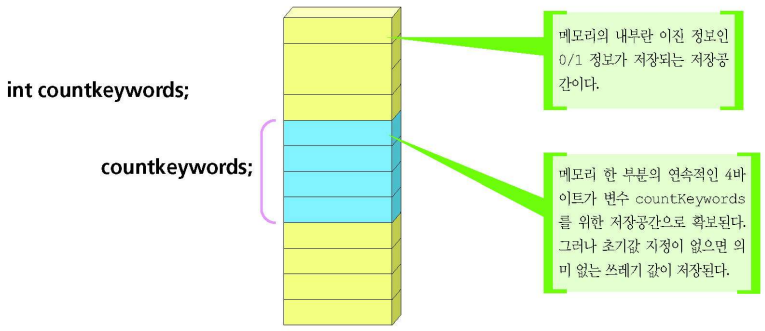
```
[저장할 방의 타입]  [방의 이름];
        int countkeywords;
```

  - 정수를 저장할 countkeywords라는 방을 만들라는 명령
  - int는 integer (정수)의 약어로 방의 타입을 나타냄
  - 방에 저장할 값의 타입 : 정수, 실수, 문자 등
  - 방의 타입에 따라 방의 크기가 달라짐
  - countkeywords는 방의 이름이며 프로그램 작성자가 정해 주어야 함

21

## 정수 값을 저장할 방의 선언

- 변수




22

## 변수의 초기화

- 변수의 초기 값
  - 변수(variables)는 값을 저장할 방의 이름
  - 변수 선언과 동시에 그 초기값 저장 가능
  - 변수는 초기값이 무엇이든 항상 마지막 값만을 저장

```
int   a; // 정수 저장할 a라는 방을 만들어라. 현재 들어있는 값은 쓰레기
int   b = 100; // 정수 저장할 b 방을 만들고, 100을 저장하라
a = 10; // a 방에 10을 저장해라
a = 25; // a 방에 25를 저장하라. a 방의 값은 10에서 25로 바뀜
b = 200; // b 방에 200을 저장하라
a = 50; // a 방에 50을 저장하라
```



## 변수의 초기화

- 변수의 초기 값
  - 같은 자료 유형의 변수는 여러 개를 한 문장으로 선언 가능
  - 같은 자료 유형의 변수는 선언하면서 변수의 초기 값을 저장

```
int   a, b; // 정수 값을 저장할 a, b 방을 각각 만들어라
int   c;    // 정수 값을 저장할 c 방을 만들어라
int   d;    // 정수 값을 저장할 d 방을 만들어라
int   e = 10, f = 20; // 정수 값을 저장할 e, f 방을 만들고
                      // e 방에는 10을, f 방에는 20을 저장하라
```

## 대입문은 값의 저장

- "=" : 대입연산자 (assignment operator)
  - a = 100; // 100을 a 방에 저장하라
  - 우측의 값 또는 결과를 좌측의 저장 장소에 저장하라는 의미
  - = 우측의 계산을 모두 끝낸 후 = 좌측의 방에 그 결과를 저장하게 됨
  - 대입 연산자인 "=" 의 우측에는 좌측의 저장공간 (방 이름)에 저장할 자료 값 또는 값을 알 수 있는 표현식이 올 수 있음
    - int a, b, c, d, e, f;
    - a = 100; // 100을 a 방에 저장하라.
    - b = 200;
    - c = a + b; // a 방의 값과 b 방의 값을 더한 결과 값을 c 방에 저장하라.
    - d = a – b; // 이 명령 이후에 d 방에는 -100이 저장된다.
    - e = a * b;
    - f = 10/2;

25

## 대입문은 값의 저장

- "="의 잘못된 사용
  - 대입 연산자인 "="의 좌측에는 항상 저장 공간인 변수 (방 이름)만 가능
    - 7 = 3 + 4;
    - a + b = 5;
    - b + 1 = 1 + b;

26

## 대입문 예제

```
#include <stdio.h> //printf() 의 이용을 위한 헤더 파일 포함

int main(void)  {
    int age;
     age = 16;

    printf("age =   16; 이후에 age에 저장된 값은 %d 입니다. ₩n", 16);
    printf("age =   16; 이후에 age에 저장된 값은 %d 입니다. ₩n", age);

    return 0;
}
```

27

## 프로그램 실습 (Lab 1)

- 아래와 같이 출력되도록 프로그래밍 하시오.
  - printf 함수를 이용하여 출력하시오.
  - 단, 모든 숫자는 printf 함수의 " " 안에 적지 말고 밖에 적도록 한다.

```
나의 이름은 아무개입니다.
나의 학번은 20191234입니다.
나의 생일은 8월 15일입니다.
```

28

## 프로그램 실습 (Lab 2)

- 아래에 주어진 순서에 따라 프로그래밍 하시오.
  - 정수 값을 저장할 변수 3개 (변수 이름은 각각 a, b, sum)를 만드시오.
  - 변수 a에 3을 저장하고, 변수 b에는 4를 저장하시오.
  - 변수 a와 변수 b를 더한 결과를 변수 sum 에 저장하시오.
  - printf 함수를 이용하여 다음과 같이 출력하시오.
    - 단, printf 함수 사용 시 3, 4, 7 등의 숫자는 보이지 않아야 한다.

```
a 방의 값은 3이고, b 방의 값은 4입니다.
a + b의 값은 7입니다.
```

29

## 실습시간 (2019년 3월 13일)

- 교재 예제:
  - HelloComment.c, PrintfOne.c, PrintTwo.c
- 실습문제:
  - Lab1, Lab2

30

2장이 끝났습니다.