

---

# **Introduction to Software Design**

## **C01. C Basics**

Yoonsang Lee  
Spring 2020

# 실습 Q&A 정책

---

- 기본적으로 채팅창이 아니라, 손을 들어 질문 의사를 표시하는 방식으로 질문.
- 조교는 학생이 질문하면 답을 한 번에 가르쳐주는 것이 아니라, 그 학생에게 계속 질문을 던지면서 답을 스스로 찾아갈 수 있도록, '고기를 잡아주는 것이 아니라 고기잡는 법을 가르쳐준다'는 느낌으로 질문에 응대.
  - 문제푸는 것을 최대한 도와주지만 해결 방법을 직접적으로 알려주는 것이 아니라 학생이 찾아봐야 할 것, 적용해야 할 것에 대한 질문을 던지며 학생 스스로가 해결 방법을 깨달을 수 있도록 한다는 의미.
- 다음 주 강의 및 실습은 정상적으로 진행.

# Topics Covered

---

- Why C?
- Compiler & Interpreter
- Setting Environment
- printf(), scanf()
- Variables, Operators
- Loop statements (while, for)
- Conditional statements (if - else if - else, switch)

# Why C?

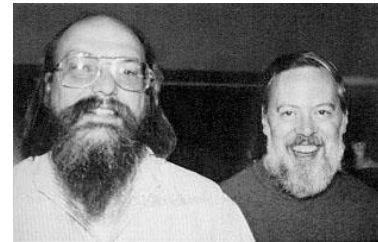
- 컴퓨터 시스템에 대한 이해

- Unix를 만들기 위해 Dennis Ritchie가 1972년에 개발
  - 운영체제를 만들기 위해 개발된 언어이기 때문에, 운영체제/하드웨어 등 컴퓨터 시스템과 근접한 언어

- 저급(low-level) 언어적 특징

- 프로그래머가 직접 포인터(pointer), 메모리 관리
- 비트(bit), 바이트(byte) 단위로 데이터 조작이 수월
- 저수준의 제어가 가능

- 컴퓨터 시스템에 대한 깊이 있는 이해를 위해 전공자로서 배울 필요가 있다!



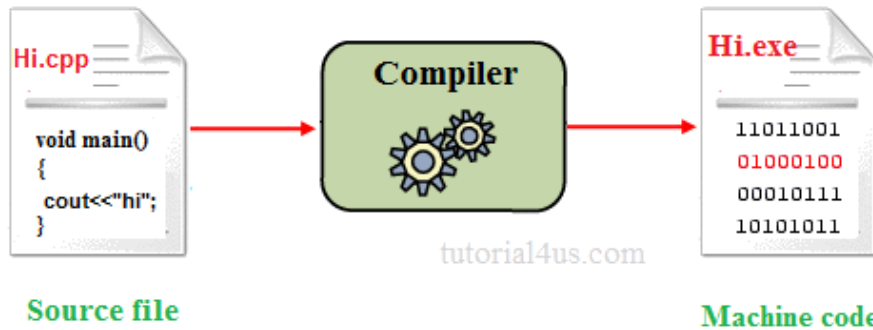
[Ken Thompson](#) (left) with [Dennis Ritchie](#) (right)

# Why C?

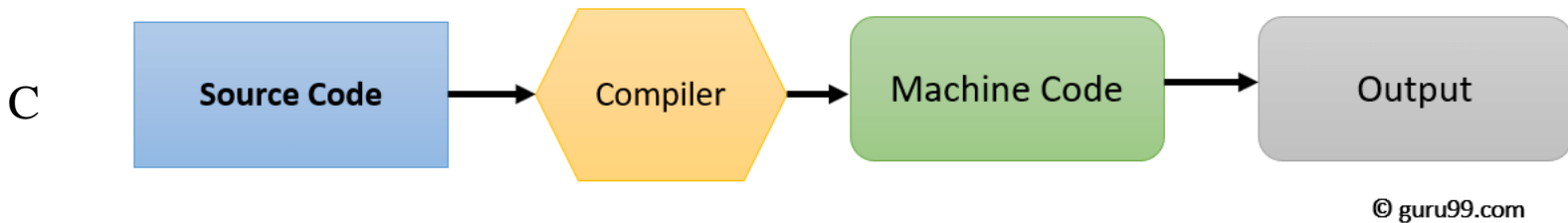
- 가장 기본이 되는 언어
  - 다른 프로그래밍 언어에 많은 영향 (Python 포함)
- 가볍고 빠르다!
  - 프로그램 크기 ↓, 메모리 사용량 ↓, 실행 속도↑
- 여전히 널리 쓰인다
  - 운영체제 (Unix 등)
  - 디바이스 드라이버
  - 웹서버 (Apache 등)
  - 임베디드 시스템 (냉장고, 전기밥솥 등)



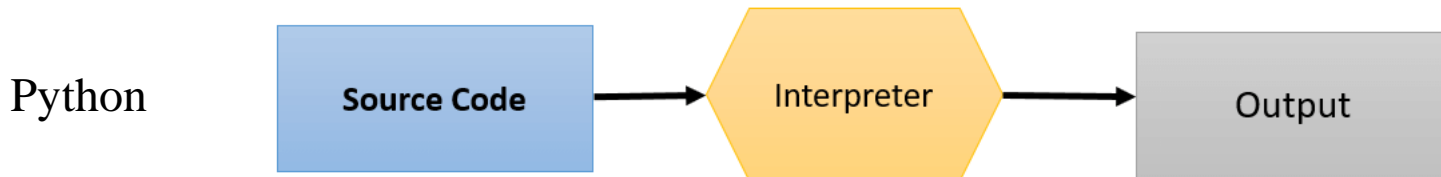
# Compiler & Interpreter



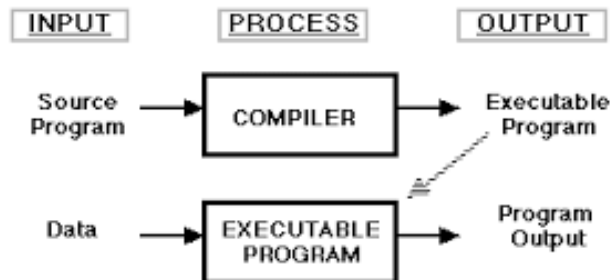
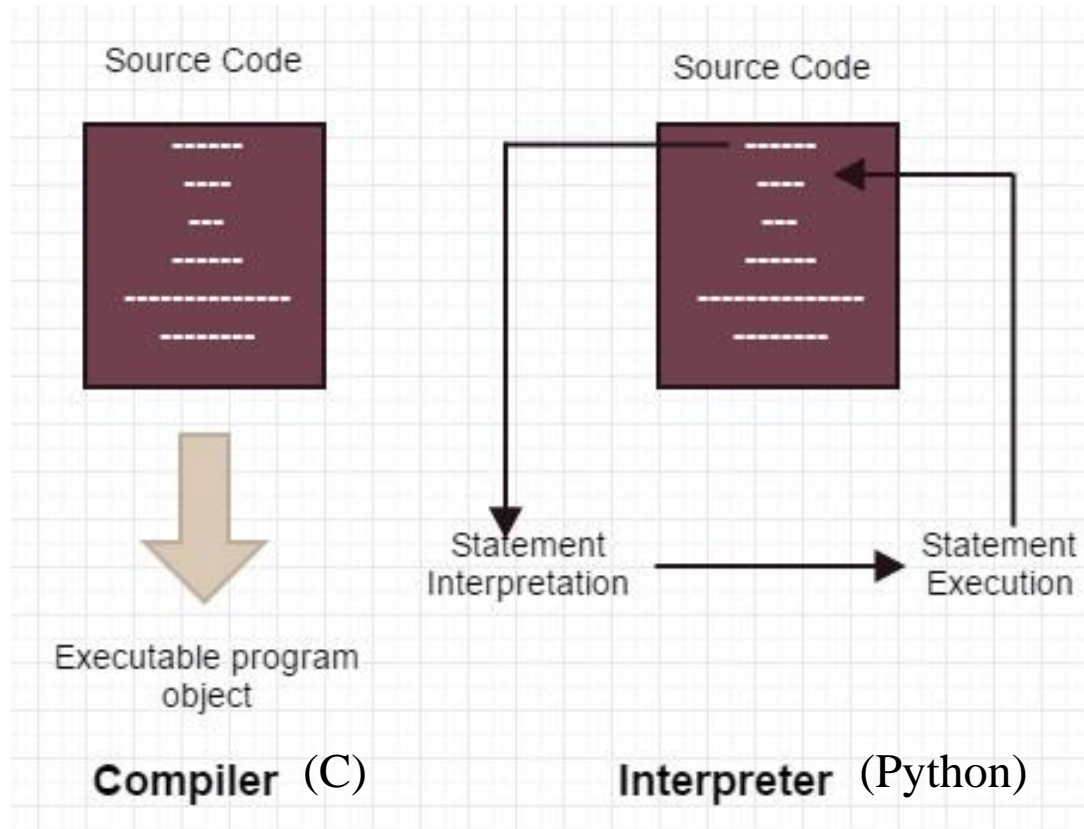
## How Compiler Works



## How Interpreter Works



# Compiler & Interpreter



# Setting Environment

---

- C컴파일러에도 여러 종류가 있다
  - MSVC, gcc, clang, ...
- 본 강좌의 C언어 실습 환경: **Ubuntu + gcc**
  - Ubuntu에는 gcc가 기본 설치되어 있음.
  - Windows 혹은 다른 OS를 사용하고 있다면 virtual machine software를 설치하여 Ubuntu를 설치.
  - 과제 채점 역시 Ubuntu에서 gcc로 할 것임.
    - Windows에서 mingw-w64를 설치하여 gcc를 이용할 수도 있으나, Ubuntu gcc에서 문제없이 컴파일 되고 실행되는 것은 본인이 확인해야 함 (문제가 있으면 감점)



# Setting Environment

---

- 추천 환경:
  - Virtual machine: <http://www.virtualbox.org/>
  - Ubuntu: <http://releases.ubuntu.com/18.04>
  - Editor: vim
  - 자세한 환경 설정 및 vim 사용법은 lab에서 다룰 예정
- 참고:
  - 임시로 사용할만한 online C complier
    - [https://www.onlinegdb.com/online\\_c\\_compiler](https://www.onlinegdb.com/online_c_compiler)
    - [https://www.tutorialspoint.com/compile\\_c\\_online.php](https://www.tutorialspoint.com/compile_c_online.php)
  - Visualize C code execution
    - <http://www.pythontutor.com/cpp.html#mode=display>

# Hello World 예제

---

```
#include <stdio.h>
```

```
int main(void)
```

```
{
```

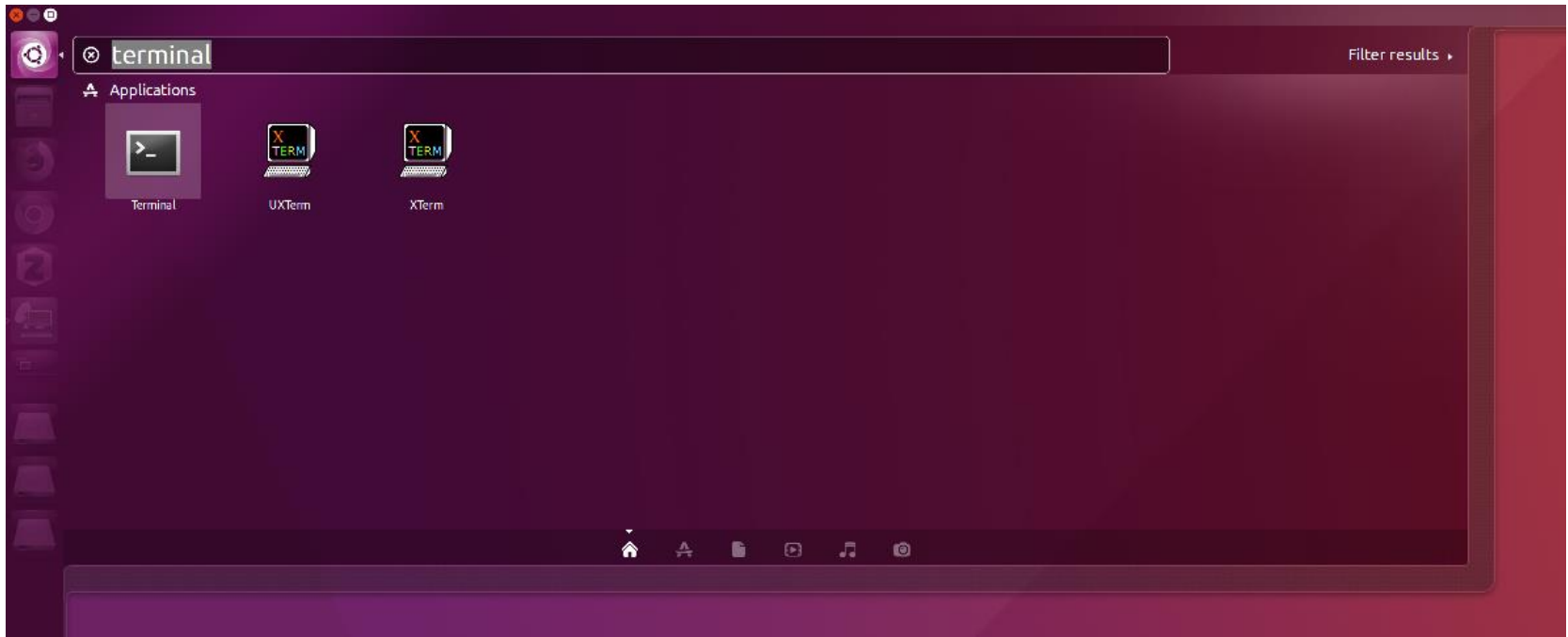
```
    printf("Hello world! \n");
```

```
    return 0;
```

```
}
```

# Hello World 컴파일 & 실행 (Ubuntu)

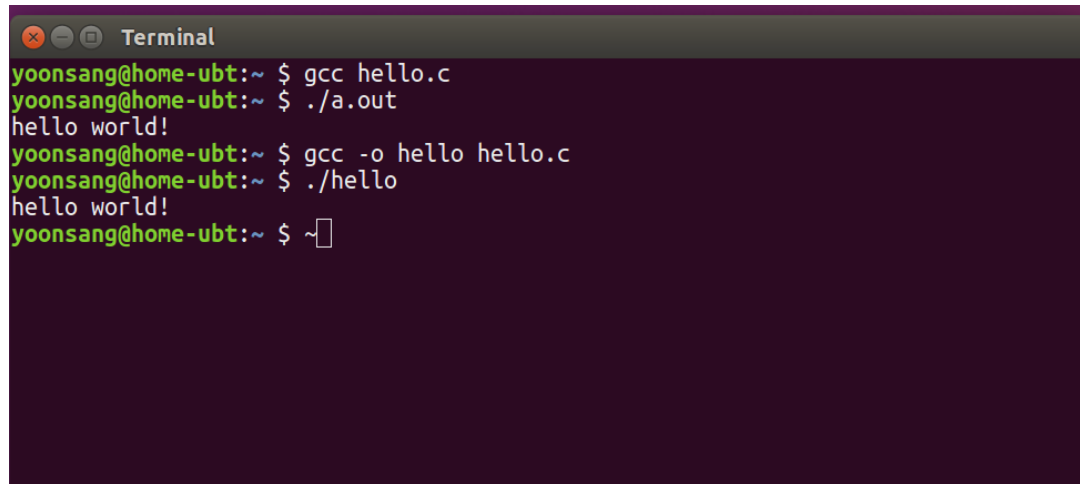
- Click Dash button (Start button)
- Type “terminal” and click Terminal





# Hello World 컴파일 & 실행 (Ubuntu)

- `$ gcc hello.c` (컴파일 & 링크)
- `$ ./a.out` (생성된 실행파일 실행)
- 혹은
- `$ gcc -o hello hello.c` (실행파일의 이름을 hello로 지정)
- `$ ./hello` (생성된 실행파일 실행)



```
Terminal
yoonsang@home-ubt:~ $ gcc hello.c
yoonsang@home-ubt:~ $ ./a.out
hello world!
yoonsang@home-ubt:~ $ gcc -o hello hello.c
yoonsang@home-ubt:~ $ ./hello
hello world!
yoonsang@home-ubt:~ $ ~
```

# Hello World 컴파일 & 실행 (Ubuntu)

---

- vim 대신 gedit와 같은 다른 text editor를 사용해서 편집 및 저장한 후 terminal에서 gcc로 컴파일 및 실행할 수도 있다.
- 하지만 vim에 익숙해지면 Linux/Unix 시스템 사용 시 도움이 되는 부분이 많기 때문에, 본 강좌에서는 vim을 사용하는 것을 권함.

# 함수 (Function)

```
#include <stdio.h>
```

출력형태      함수이름      입력형태

```
int main(void)
```

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

함수의 몸체(body)

- **main** 함수가 호출이 되면서 프로그램이 시작됨
- C언어로 구현된 모든 프로그램은 시작점에 해당하는 **main**이라는 이름의 함수를 반드시 정의해야 함

# 세미콜론 (Semicolon)

```
#include <stdio.h>
```

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

- 하나의 명령문은 세미콜론(;)으로 끝난다

```
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; }
```

: 컴파일러 입장에서는  
모두 동일한 소스코드!



# 주석(Comment)

```
#include <stdio.h> // 헤더 파일

int main(void) // main 함수
{
    /*
    hello world 출력 프로그램
    */
    printf("Hello world! \n"); // 문자열 출력
    return 0;
}
```

- 주석: 소스코드에 삽입된 메모 (컴파일의 대상에서 제외)
- 블록단위 주석
  - `/* 주석처리 내용 */`
- 행단위 주석
  - `// 주석처리 내용`

# Hello World 예제 분석

---

- `#include <stdio.h>`
  - `stdio.h` 파일의 내용을 이 위치에 가져다 놓으라는 뜻
  - `printf` 함수 호출을 위해 필요함. 즉, `stdio.h`에는 `printf` 함수 호출을 위해 필요한 정보가 들어있음.
- `printf("Hello world! \n");`
  - `printf` 함수를 호출.
  - 함수의 인자로 문자열 "Hello world! \n"을 전달함.
- `return 0;`
  - 현재 함수의 실행을 종료하면서 함수를 호출한 곳으로 특정 값을 반환

# C & Python Examples

---

- C

```
#include <stdio.h>

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

- Python

```
print('Hello world!')
```

# printf 함수의 인자(Argument)

```
#include <stdio.h>

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

(실행결과)  
Hello world! \n  
1234  
10 20

서식문자 or 서식지정자 (format specifier)

# printf 함수의 인자(Argument)

```
printf("%d\n", 1234);
```

↑            ↑  
인자1        인자2

(실행결과)  
1234

```
printf("%d %d\n", 10, 20);
```

↑            ↑            ↓  
인자1        인자2        인자3

(실행결과)  
10 20

# printf()와 서식문자 (서식지정자, format specifier)

- %d: 부호가 있는 10진수 형태로 출력하라는 뜻
- 이와 같은 문자를 서식 문자 (서식지정자)라고 부름. 출력의 형태를 지정하는 용도로 사용.

```
#include <stdio.h>

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

`\n` (**Wn**)은 특수 문자를 나타내는 escape sequence 중 하나로서, 개행 (줄바꿈)을 의미한다.

<http://ko.cppreference.com/w/cpp/language/escape>

(실행결과)

```
Hello world! \n
1234
10 20
```

# C & Python Examples

- C

```
#include <stdio.h>

int main(void)
{
    printf("%d and %d\n", 3, 4);
    return 0;
}
```

- Python

```
print('%d and %d'%(3, 4))
```

- Python에서도 **C와 동일한 format specifier**를 사용 가능.
- Python에서는 **%와 tuple**을 이용하며, printf() 함수 안에서만 사용하는 C와는 달리 **string을 생성하는 모든 경우에 format specifier** 사용.
- format specifier를 1개만 사용하는 경우 tuple 생략 가능.

```
a = 'name: %s, score: %d'%( 'John', 95)
b = 'The number is %d.'%10

print(a)          # name: John, score: 95
print(b)          # The number is 10.
```

# 변수 (Variable)

```
#include <stdio.h>

int main(void)
{
    int num;
    num = 20;
    printf("%d", num);
    return 0;
}
```

- `int num;`
  - `int`: 정수의 저장을 위한 메모리 공간을 할당
  - `num`: 할당된 메모리 공간의 이름은 `num` (`int`형 변수 `num`)



# 변수의 다양한 선언 및 초기화 방법

- 변수 2개 선언

```
int num1;  
int num2; == int num1, num2;
```

선언만 한 경우임. 값이 대입되기 전까지 쓰레기 값 (의미없는 값)을 가짐

- 변수 2개 선언 및 초기화

```
int num1;  
int num2;  
num1 = 10;  
num2 = 20; == int num1 = 10, num2 = 20;
```

선언한 후 값을 대입

선언과 동시에 초기화

# C & Python Examples

- C

```
int main(void)
{
    int a = 3;
    double b = 4.5;
    return 0;
}
```

- Python

```
a = 3
b = 4.5
```

- Python은 실행 시점(run time)에 변수의 자료형을 결정하는 동적타입언어 (dynamically typed language)이다.
- 그러므로 명시적으로 자료형을 지정하며 변수를 선언할 필요가 없다.
- C는 컴파일 시점(compile time)에 변수의 자료형을 결정하는 정적타입언어 (statically typed language)이다.

# scanf()

```
#include <stdio.h>

int main(void)
{
    int num;
    scanf("%d", &num);
    printf("%d", num);
    return 0;
}
```

뒤에 **Wn**을 붙이지 않는다!

`scanf("%d", &num);`

변수 num에 저장하라

부호 있는 10진수  
형태로 입력 받아서

변수 이름 앞에 **&**을 붙인다!

# scanf()

```
#include <stdio.h>
```

```
int main(void)
```

```
{
```

```
    int num1, num2, num3;
```

```
    scanf("%d %d %d", &num1, &num2, &num3);
```

```
    return 0;
```

```
}
```



- 숫자 3개를 어떻게 입력할까?

- 2 3 4↵

- 2↵3↵4↵

- 2<tab>3<tab>4↵

- 공백(space), 줄바꿈(newline), 탭(tab)을 기준으로 구분

# C & Python Examples

- C

```
#include <stdio.h>
int main(void)
{
    int num1, num2;
    printf("Enter two numbers: ");
    scanf("%d %d", &num1, &num2);
    printf("Entered numbers are %d and %d\n", num1, num2);
    return 0;
}
```

- Python

```
print('Enter two numbers: ', end='')
num1 = int(input())
num2 = int(input())
print('Entered numbers are %d and %d'%(num1, num2))
```

- C의 scanf()는 space, newline, tab을 기준으로 데이터를 분리.
- Python의 input()은 newline만을 기준으로 데이터를 분리.

# 증가, 감소연산자

++num	값을 1 증가 후, 속한 문장을 진행 예) val = ++num;
num++	속한 문장을 진행한 후, 값을 1 증가 예) val = num++;
--num	값을 1 감소 후, 속한 문장을 진행 예) val = --num;
num--	속한 문장을 진행한 후, 값을 1 감소 예) val = num--;

```
int num1=12;
int num2=12;

printf("num1: %d\n", num1);
printf("num1++: %d\n", num1++);
printf("num1: %d\n", num1);

printf("num2: %d\n", num2);
printf("++num2: %d\n", ++num2);
printf("num2: %d\n", num2);
```

```
num1: 12
num1++: 12
num1: 13
```

```
num2:12
++num2:13
num2: 13
```

# C & Python Examples

- C

```
#include <stdio.h>

int main(void)
{
    int num1 = 10, num2 = 10;

    num1++;
    num2++;

    printf("num1: %d\n", num1);
    printf("num2: %d\n", num2);

    return 0;
}
```

- Python

```
num1 = 10
num2 = 10

num1 += 1
num2 += 1

print('num1: %d'%num1)
print('num2: %d'%num2)
```

- Python에는 증가연산자(++)**혹은** 감소연산자(--)**가 없다.**

# 논리연산자

A && B	논리 AND: A와 B 모두 '참'이면 연산결과로 '참'을 반환
A    B	논리 OR: A와 B 둘 중 하나라도 '참'이면 연산결과로 '참'을 반환
!A	논리 NOT: A가 '참'이면 '거짓', '거짓'이면 '참'을 반환

```
int num1 = 10;
int num2 = 12;
int r1, r2, r3;

r1 = (num==10 && num==12);
r2 = (num1<12 || num2>12);
r3 = !num1;
```

```
r1: 1
r2: 1
r3: 0
```

- C언어에서,
- 거짓(False): 0
- 참(True): 0이 아닌 모든 숫자. 주로 1을 사용.
- 그래서 관계 및 논리연산자는 0 or 1을 리턴.



# C & Python Examples

- C

```
#include <stdio.h>

int main(void)
{
    int num1 = 10, num2 = 12;
    int result1, result2, result3;

    result1 = (num1==10 && num2==12);
    result2 = (num1<12 || num2>12);
    result3 = (!num1);

    printf("%d\n", result1);
    printf("%d\n", result2);
    printf("%d\n", result3);

    return 0;
}
```

- Python

```
num1 = 10
num2 = 12

result1 = (num1==10 and num2==12)
result2 = (num1<12 or num2>12)
result3 = not num1

print(result1)
print(result2)
print(result3)
```

# 연산자의 우선순위와 결합방향

- $3 + 4 * 5 / 2 - 10$ 
  - 연산자의 우선 순위에 의해 곱셈, 나눗셈이 먼저 계산
  - 결합 방향에 의해 곱셈이 나눗셈보다 먼저 계산
- 우선순위(precedence): 어떤 연산자를 먼저 계산할 것인가
  - 곱셈, 나눗셈이 덧셈, 뺄셈보다 우선순위가 높다
- 결합방향(associativity): 우선순위가 동일한 연산자 사이에서 계산을 진행하는 방향
- 만일  $3+4$ 를 먼저 하고 싶다면?
  - :  $(3+4) * 5 / 2 - 10$
- 전체 연산자 우선순위 및 결합방향 표:
  - [http://en.cppreference.com/w/c/language/operator\\_precedence](http://en.cppreference.com/w/c/language/operator_precedence)

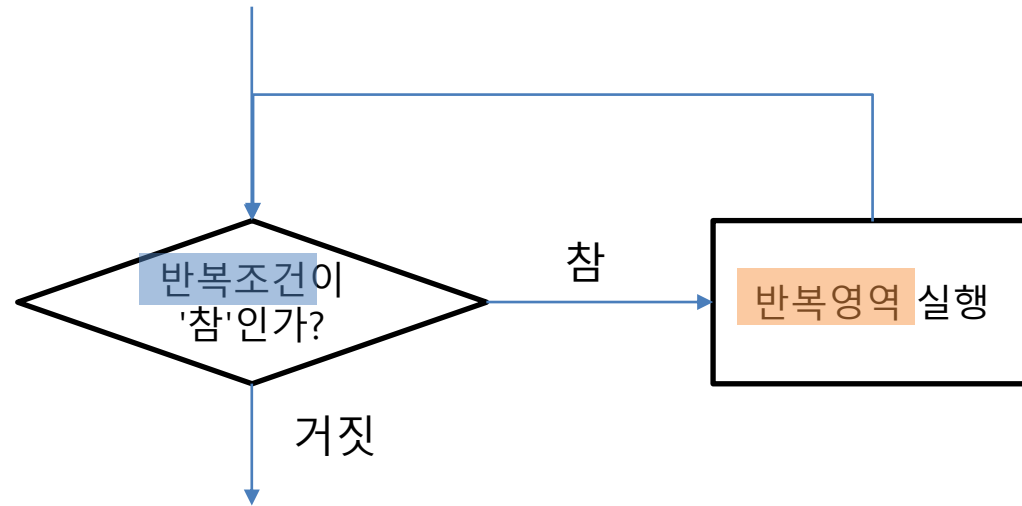
# Quiz #1

---

- Go to <https://www.slido.com/>
- Join #isd-hyu
- Click “Polls”
  
- Submit your answer in the following format:
  - **Student ID: Your answer**
  - e.g. **2017123456: 4)**
  
- Note that you must submit all quiz answers in the above format to be checked as “attendance”.

# while문

```
while (num < 5)
{
    printf("%d\n", num);
    num++;
}
```



반복영역이 한 문장이면 중괄호 { } 생략 가능

```
while (num < 5)
    printf("%d\n", num++);
```

# C & Python Examples

- C

```
#include <stdio.h>

int main(void)
{
    int num = 0;
    while (num < 5)
    {
        printf("%d\n", num);
        num++;
    }
    return 0;
}
```

- Python

```
num = 0
while num < 5:
    print(num)
    num+=1
```

# for

1)                    2)                    3)

```
for (int i=0; i<3; i++)  
{  
    printf("%d\n", i);  
}                    4)
```

첫 번째 반복:

1) → 2) → 3) → 4) : i==1

두 번째 반복:

2) → 3) → 4) : i==2

세 번째 반복:

2) → 3) → 4) : i==3

네 번째 반복:

2)

- 1) 초기식: 반복을 시작하기 전에 한 번만 실행
- 2) 조건식: 매 반복 시작에 앞서 실행되어 반복을 할지 결정
- 3) 증감식: 매 반복 실행 후 실행
- while문과 마찬가지로 반복영역이 한 문장이면 중괄호 {} 생략 가능

# C & Python Examples

- C

```
#include <stdio.h>

int main(void)
{
    int sum = 0;
    for(int i=0; i<5; ++i)
        sum += i;
    printf("%d\n", sum);
    return 0;
}
```

- Python

```
sum = 0
for i in range(5):
    sum += i
print(sum)
```

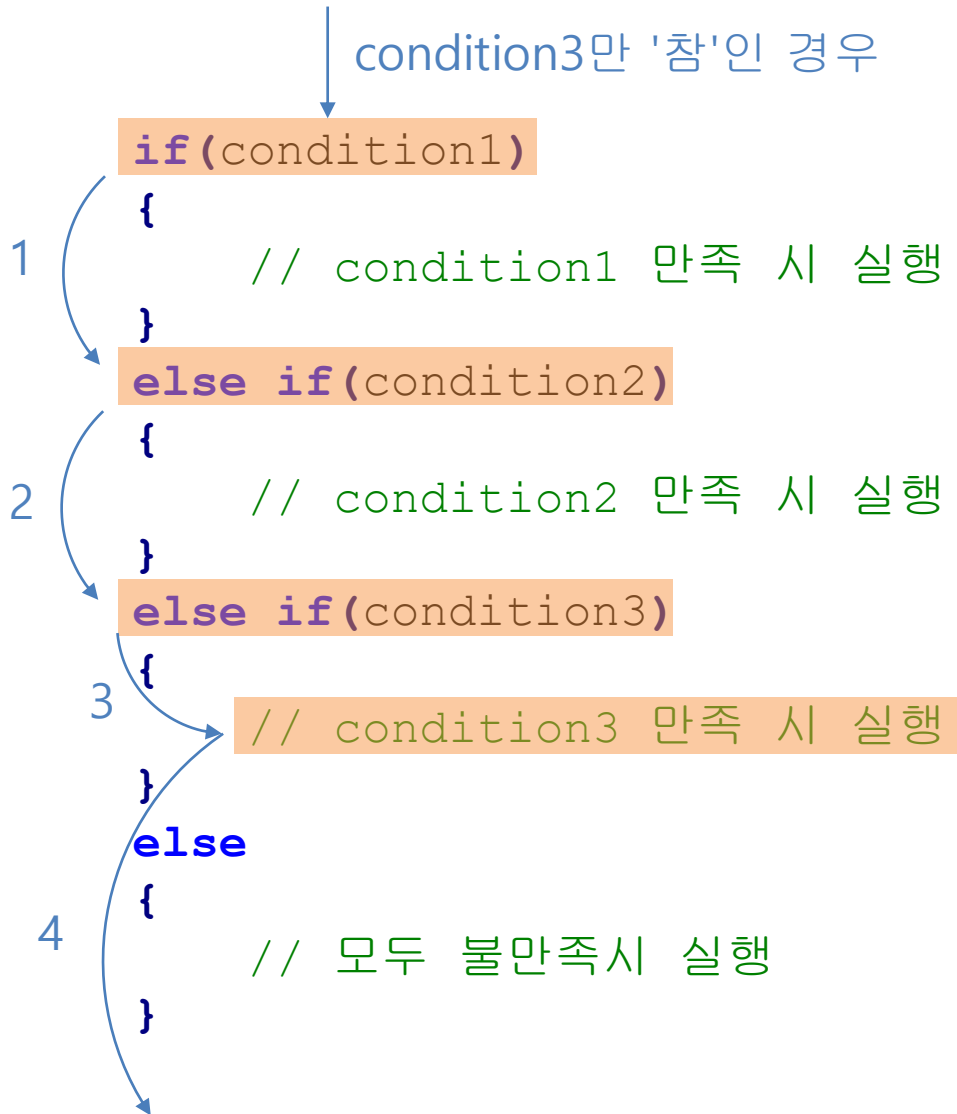
# Quiz #2

---

- Go to <https://www.slido.com/>
- Join #isd-hyu
- Click “Polls”
  
- Submit your answer in the following format:
  - **Student ID: Your answer**
  - e.g. **2017123456: 4)**
  
- Note that you must submit all quiz answers in the above format to be checked as “attendance”.



# if - else if - else



- if 하나만
- if와 else if들만
- if와 else만
- if, else if들, else 함께 사용하는 것이 모두 가능!

# if문의 사용 예

의미	if문
x < 0 이면	
x가 10이면	
x가 5가 아니면	
x가 참이면	
x가 거짓이면	
0 < x <= 5 이면	

~~if(0 < x <= 5)~~

// Python에서는  
가능하지만 C에서는  
불가능한 문법!

# if문의 사용 예

의미	if문
x < 0 이면	if(x < 0)
x가 10이면	if(x == 10)
x가 5가 아니면	if(x != 5)
x가 참이면	if(x) or if(x!=0)
x가 거짓이면	if(!x) or if(x==0)
0 < x <= 5 이면	if(x>0 && x<=5)

~~if(0 < x <= 5)~~

// Python에서는  
가능하지만 C에서는  
불가능한 문법!

# C & Python Examples

- C

```
#include <stdio.h>

int main()
{
    int a = 3;
    if(a < 0)
        printf("case 1");
    else if(a >= 0 && a < 5)
        printf("case 2");
    else
        printf("case 3");
    return 0;
}
```

- Python

```
a = 3
if a < 0:
    print('case 1')
elif 0 <= a < 5:
    # elif a >= 0 and a < 5:
    이것도 가능
    print('case 2')
else:
    print('case 3')
```

# break & continue

- break : 반복을 중단하고 빠져나감

```
int main(void)
{
    ...
    while(1)
    {
        if(x > 20)
            break;
        ...
    }
    ...
}
```

- continue : 나머지 반복영역 건너뛰고 다시 반복조건 확인

```
int main(void)
{
    ...
    while(1)
    {
        if(x/2 == 1)
            continue;
        ...
    }
    ...
}
```

# C & Python Examples

- C

```
#include <stdio.h>

int main(void)
{
    int num = 0;
    while(1)
    {
        num++;
        if(num%2==0)
            continue;
        if(num>10)
            break;
        printf("%d\n", num);
    }

    return 0;
}
```

- Python

```
num = 0
while True:
    num += 1
    if num%2==0:
        continue
    if num>10:
        break
    print(num)
```

# switch문 : 또 다른 종류의 조건문

---

```
switch (n)
{
case 1:    if(n==1)
           printf("1");
           printf("11");
           break;
case 2:    else if(n==2)
           printf("2");
           printf("22");
           break;
default:  else
           printf("default")
}
```

# if else & switch

if else

```
if (n==1)
    print("1");
else if (n==2)
    print("2");
else if (n==3)
    printf("3");
else
    printf("4");
```

switch

```
switch (n)
{
case 1:
    printf("1");
    break;
case 2:
    printf("2");
    break;
case 3:
    printf("3");
    break;
default:
    printf("4");
}
```

값의 비교가 나열되는 경우 switch문을 사용하는 것이 더 편한 경우가 많다.  
하지만 어느 것을 사용할 지는 개인의 선호도에 따라 정하면 된다.



# if else & switch

if else

```
if(n>=0 && n<10)
    print("1");
else if(n>=10 && n<20)
    print("2");
else if(n>=20 && n<30)
    printf("3");
else
    printf("4");
```

switch

```
switch(n)
{
    case ??:
        printf("1");
        break;
    case ??:
        printf("2");
        break;
    case ??:
        printf("3");
        break;
    default:
        printf("4")
}
```

하지만 모든 if else 문을 switch 문으로 대체할 수 있는 것은 아니다.

# goto문

---

- 쓰지 말 것. 굳이 알 필요도 없음.

# Quiz #3

---

- Go to <https://www.slido.com/>
- Join #isd-hyu
- Click “Polls”
  
- Submit your answer in the following format:
  - **Student ID: Your answer**
  - e.g. **2017123456: 4)**
  
- Note that you must submit all quiz answers in the above format to be checked as “attendance”.

# Next Time

---

- Labs in this week:
  - Lab1: 6-Lab-CEnvSetting, 과제 6-1
  - Lab2: 과제 6-2
- Next lecture:
  - 7-C02. Data Representation