민 선 생 코 딩 학 원 시 작 반

수업노트 LV-03

배우는 내용

소스코드의 반복 처리

1. if 안의 and, or
2. for 이해하기

복습 : 디버깅 Quiz

아래 소스코드를 이해하고, 어디서 버그가 발생할지 찾아보자.

```
int t;
cout << t;
```

- t를 선언하면, t 안에 쓰레기 값이 들어있다.
- 쓰레기 값을 출력하기 때문에 Error가 발생한다.

```
d = 10;
int d;
cout << d;
```

- 변수 d를 선언하기 전, d에다가 10을 넣었다.
- 변수를 선언을 먼저 해야, 변수를 사용할 수 있다.

```
int count;
count++;
```

- count를 선언하면, 쓰레기 값이 들어있다.
- 쓰레기 값에 1 더하려고 해서 Error 발생한다.

쓰레기 값을 출력하거나 연산을 하면 안된다.

소스코드는 윗줄에서 아랫줄 순서대로 순차적으로 실행된다.

참과 거짓

C, C++에서 0은 거짓, 0을 제외한 숫자는 참으로 처리를 한다.

```
if (1)
{
    cout << “참입니다”;
}
else
{
    cout << “거짓입니다”;
}
```

출력결과 : 참입니다

```
if (0)
{
    cout << “참입니다”;
}
else
{
    cout << “거짓입니다”;
}
```

출력결과 : 거짓입니다

```
if (-1)
{
    cout << “참입니다”;
}
else
{
    cout << “거짓입니다”;
}
```

출력결과 : 참입니다

=와 ==의 차이

```
int x = 5;

if (x == 5)
{
    cout << “x는 5 입니다”;
}
else
{
    cout << “x는 ” << x <<“입니다”;
}
```

아래 두 부호는 다른 의미를 가진다.

- = : 값을 넣는다. (**대입**한다.)
- == : 같은 값인지 **비교**한다.

if문에서 자주하는 실수이다. 실수로 바꿔 쓰는 경우가 많이 발생한다.

만약 실수로 잘못 썼다면?

```
int x;

cin >> x;

if (x = 10)
{
    cout << "x는 10";
}
else
{
   cout << "x는 "<< x << "입니다.";
}
```

(x == 10) 대신, 실수로 (x = 10) 으로 썼다면?

1. if (x = 10) 에서, 먼저 x에 10이 대입된다.
2. if (10) 으로 인식한다.
3. 결과적으로 if문은 '참'으로 인식한다.

if문에서 =와 ==를 혼동해서 쓰는 것을 주의하자.

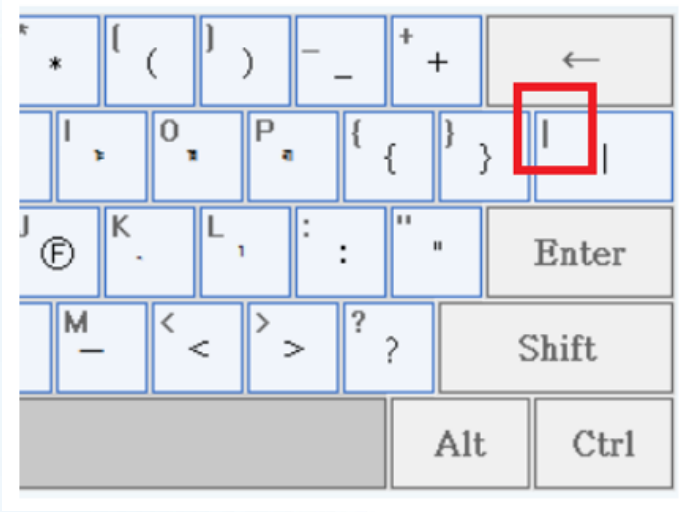
and, or 이해하기

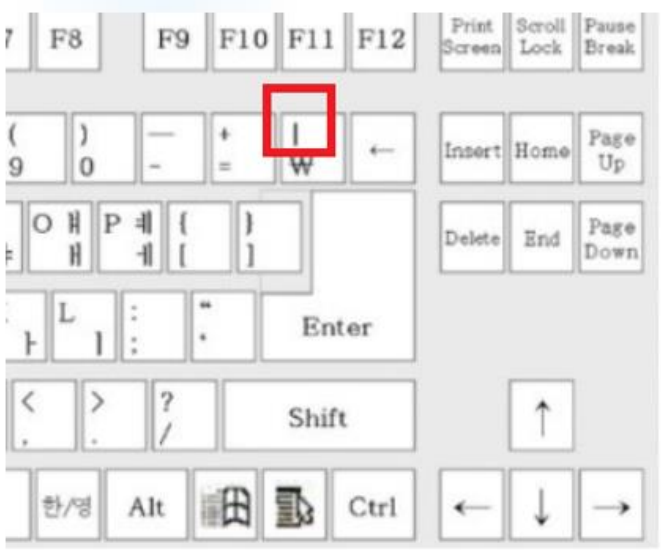
- if문 안에 여러 조건을 쓰고 싶을 때 사용한다.
- && : 모두 참일 때 조건에 진입한다.
  ➔ and 연산자라고 부른다.
- || : 하나라도 참일 때 조건에 진입한다.
  ➔ or 연산자라고 부른다.

```
int x = 5;

if (x > 3 && x < 10)
{
    cout << “#”;
}

if (x == 4 || x == 5)
{
    cout << “@”;
}
```

Bar의 키보드 위치

키보드의 형태에 따라 bar의 위치가 다를 수 있다.





다중 조건 if문 예제 1

[문제 1] 아래 조건에 맞게 출력할 것

입력 받은 숫자가 3보다 같거나 크고, 6보다 같거나 작을 때
YES 출력, 그렇지 않으면 **NO** 출력

```
int x;

cout << "숫자입력 : ";
cin >> x;

if (3 <= x && x <= 6)
{
    cout << "YES";
}
else
{
    cout << "NO";
}
```

자주하는 실수 1

if (3 <= x <= 6)

이런 표현은 불가능하다. && 를 써야한다.

다중 조건 if문 예제 2

[문제 2] 아래 조건에 맞게 출력할 것

입력 받은 숫자가 3, 5, 7 중 하나이면 **GOOD** 출력,
그렇지 않으면 **HOOT** 출력

```
int x;

cout << "숫자입력 : ";
cin >> x;

if (x == 3 || x == 5 || x == 7)
{
    cout << "GOOD";
}
else
{
    cout << "HOOT";
}
```

자주하는 실수 2

if (x == 3, 5, 7)

이런 표현은 불가능하다. 조건 하나씩 쓰고, || 로 연결해주어야 한다.

다중 조건 if문 예제 3

[문제 3] 아래 조건에 맞게 출력할 것

입력 받은 숫자가 0 ~ 3 사이에 있는 숫자이거나
7 ~ 10 사이에 있는 숫자라면 **Very Good** 출력,
아니면 **Bad** 출력

```
int x;

cin >> x;

if ((0 <= x && x <= 3) || (7 <= x && x <= 10))
{
    cout << “Very Good”;
}
else
{
    cout << “Bad”;
}
```

자주하는 실수 3

if (0 <= x && x <= 3 || 7 <= x && x <= 10)
먼저 처리해야 하는 조건을 괄호로 묶어주어야 한다.

for문 개념 이해하기

소스코드를 반복할 때 사용한다.

```
for (3번 반복)
{
    cout << “#”;
    cout << “ “;
}
```

출력결과 : # # #

```
for (3번 반복)
{
    for (2번 반복)
    {
        cout << “#”;
    }
}
```

출력결과 : ######

```
for (2번 반복)
{
    for (2번 반복)
    {
        cout << “#”;
    }

    for (3번 반복)
    {
        cout << “@”;
    }

    cout << “ “;
}
```

출력결과 : ##@@@ ##@@@

```
for (2번 반복)
{
    cout << “*”;
    for (2번 반복)
    {
        for (2번 반복)
        {
            cout << “#”;
        }
    }
}
```

출력결과 : *####*####

for문 원리 이해

조건 문 (if와 유사)

이 조건이 참인 동안

소스코드들을 반복한다.

반복자 값 변경 영역

한번 반복이 끝날 때 마다

한번씩 수행하는 영역이다.

초기 값 세팅 영역

For문을 시작할 때 한 번만 수행한다.

```
int x;

for (x=0; x<100; x++)
{
    cout << "#";
    cout << "@";
}
```

반복문을 쓰는 이유

- 반복되는 소스코드를 반복문을 사용할 경우, 코드가 간결 해진다.
- 반드시 써야하는 것은 아니다.

```
cout << "#";
cout << "@";
cout << "#";
cout << "@";
cout << "#";
cout << "@";
cout << "#";
cout << "@";
cout << "#";
cout << "@";
cout << "#";
cout << "@";
    ⋮
cout << "@";
```


for문으로
소스코드를 더 간결하게
작성할 수 있다.

```
int x;

for (x=0; x<100; x++)
{
    cout << "#";
    cout << "@";
}
```

변화되는 변수 값 출력

- For문 안에 x를 출력하는 소스코드를 넣으면, x값이 증가되는 것을 볼 수 있다.

```
int x;

for (x=0; x<3; x++)
{
    cout << x;
}
```

출력결과 : 012

```
int x;

for (x=3; x>0; x--)
{
    cout << x;
}
```

출력결과 : 321

[중요] for 기본형태 1

소스코드를 3회 반복 시키고 싶다면?

3회 반복하여 #을 출력하는 소스코드 예시

```
for (x=20; x<=22; x++)
{
    cout << "#"
}


for (x = 0; x>-3; x--)
{
    cout << "#"
}
```


아래 코드가 더 **가독성**이 좋다.
아래의 for문 형태를 암기하자.

```
for (x=0; x<3; x++)
{
    cout << "#"
}
```

항상 0부터 시작하고, 반복 횟수를 적어준다.
부등호는 < 로 한다.

[중요] for 기본형태 2

a값 부터 b값까지 출력하고 싶다면?

```
int a = 3;
int b = 7;

for (x=a; x<=b; x++)
{
    cout << x;
}
```

앞에는 시작 값을, 뒤에는 끝 값을 적어준다.
부등호에 =를 추가로 붙여준다.

```
int a = 15;
int b = 7;

for (x=a; x>=b; x--)
{
    cout << x;
}
```

큰 수에서 작은 수로 1씩 감소시키면서
반복하는 소스코드이다.

[중요] 두 가지 기본형태의 예시

- For문의 두 가지 형태를 익혀 두면, 다양한 응용도 쉽게 할 수 있다.
- 다음 예시들을 보며 for문의 기본 두 가지 형태를 익혀보자.

4부터 13까지 출력

```
int x;
for (x = 4; x <= 13; x++)
{
    cout << x;
}
```

5부터 1까지 출력 (부등호 주의)

```
int x;
for (x = 5; x >= 1; x--)
{
    cout << x;
}
```

2020회 # 출력

```
int x;
for (x = 0; x < 2020; x++)
{
    cout << "#";
}
```

Boss 문제 풀이

[문제]

숫자를 입력 받고, 입력 받은 숫자부터 증가한 숫자 3개 출력하기

ex) 3 을 입력 받았다면 3 4 5 출력

ex) 7 을 입력 받았다면 7 8 9 출력

```
int a;
int x;

cin >> a;

for (x = a; x < a + 3; x++)
{
    cout << x;
}

return 0;
```

시작 값을 a로 두고, a + 3 전까지 for문을 돌린다.

for문 쓸때 자주하는 실수

1부터 9까지 출력하는 프로그램

```
int x;

for (x=1; x>9; x++)
{
    cout <<  x;
}
```

for문을 시작할 때 x = 1 수행 후,
x > 10 에서 거짓이기 때문에, for문을 끝낸다.
즉, cout << x; 를 한번도 실행하지 않고 for문이 종료된다.

5부터 1까지 출력하는 프로그램

```
int x;

for (x=5; x>=1; x++)
{
    cout <<  x;
}
```

for문이 x >= 1인 동안 반복된다.
그러나 x가 줄어들지 않고, 계속 증가하기 때문에
무한 Loop가 발생한다.

의미없는 for문

```
int x;

for (x=1; x<2; x++)
{
    cout <<  x;
}
```

반복문이 1회만 수행하기 때문에,
for문을 쓰는 의미가 없다.


```
cout << "1";
```

왼쪽 코드와 같은 동작을 한다.

복잡한 For문 소스코드 이해하기

실행 결과를 예측 후 결과를 확인 해 보자.

```
int x;

for (x=1; x<=5; x++)
{
    if (x >= 2 && x <= 4)
    {
        cout <<  "V ";
    }
    else
    {
        cout <<  "G ";
    }
}
```

출력결과 : G V V V G

```
int x;

for (x=1; x<5; x++)
{
    cout << x << " ";

    if (x > 2)
    {
        cout <<  "# ";
    }

    if (x > 3)
    {
        cout <<  "$ ";
    }
}
```

출력결과 : 1 2 3 # 4 # $