민 선 생 코 딩 학 원 시 작 반

수업노트 LV-09

배우는 내용

1. SWAP
2. 구조체
3. Counting 의 응용
4. if 안에 함수 호출 여러 개 두기
5. Call by value

SWAP 이란

- 두개의 변수에 있는 값을 교체하는 것을 SWAP 이라고 함

Ex)

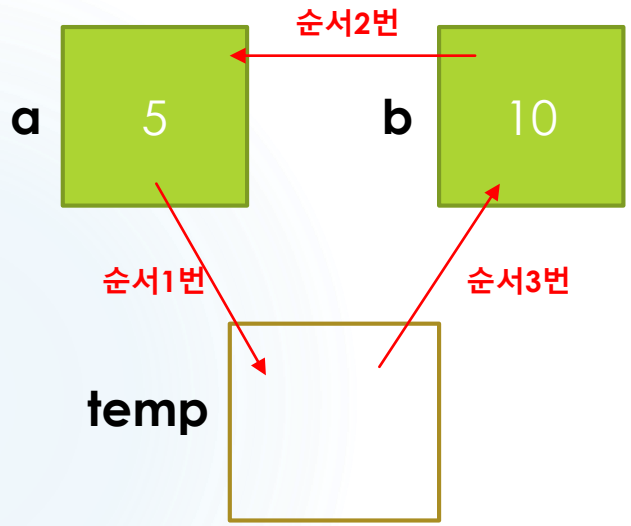

```
int a = 5;
int b = 10;

a = b;
b = a;
```

버그발생
a = b;를 수행하면 a, b 모두 10이 됩니다
b = a;를 수행하면 a, b 모두 10이 됩니다

SWAP 해결책

- **임시 변수를 하나 더 만들어** 이용하면
  두개의 변수에 있는 값을 교체할 수 있음

Ex)




```
int a = 5;
int b = 10;
int temp;

temp = a;
a = b;
b = temp;
```

정상적으로 교체 성공
temp라는 변수에 a값을 백업 해 둡니다
그리고 a에다가 b값을 넣고
b값에다 백업한 값을 넣습니다

[TIP] SWAP 소스코드 암기 방법

- SWAP 소스코드에는 일정한 패턴이 있다.

```
int a = 5;
int b = 10;
int temp;

        temp = a;

        a    =  b;

        b    = temp;
```

```
int a = 7;
int b = 2;
int temp;

        temp = b;

        b    =  a;

        a    = temp;
```

[중요] Type(타입)이란

- 변수의 형태를 Type이라고 함

  ex) **int** type / **char** type (인티져타입 / 캐릭터타입)

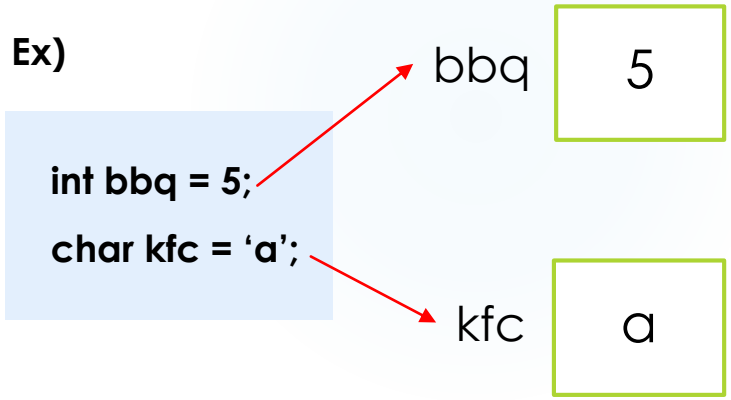
- 변수를 만드는 방법

  ➔ type의 종류를 적고 그 뒤에 변수 이름을 적음

변수 만드는 방법은 **Type종류 + 변수이름** 입니다
Type이라는 용어를 알아두세요

Ex)

```
int bbq = 5;
char kfc = 'a';
```



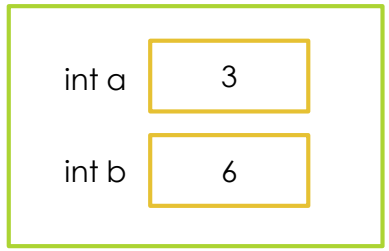
구조체 (=struct)

- 프로그래머가 int / char 외 새로운 Type을 만드는 것
- **변수들을 묶어서** 새로운 Type을 만드는 것

```
struct ABC
{
   int a;
   int b;
};
```

1. ABC라는 **구조체 type**을 생성

```
ABC x;

x.a = 3
x.b = 6;
```



2. **새로운 Type을 만들었으니**
   이제 ABC type의 변수 x를 선언

   그리고 구조체 변수 x에다가 값을 채운다

```
struct ABC
{
   int a;
   int b;
};

ABC x;

x.a = 3
x.b = 6;
```

구조체 정의 / 구조체 변수 선언

```
int main()
{
        struct ABC
        {
          int a;
          int b;
        };

        ABC x;


        cin >> x.a >> x.b;

        x.a++;
        x.b--;

        cout << x.a << x.b;

        return 0;
}
```

세미콜론을 넣어야 합니다

- struct ABC { int a; int b; }; 이 부분을 통해 ABC라는 구조체 Type이 만들어 짐

  이 동작을 **"ABC 구조체를 정의한다"** 라고 함

- ABC x; 이 부분을 통해 구조체 변수 x가 만들어 짐

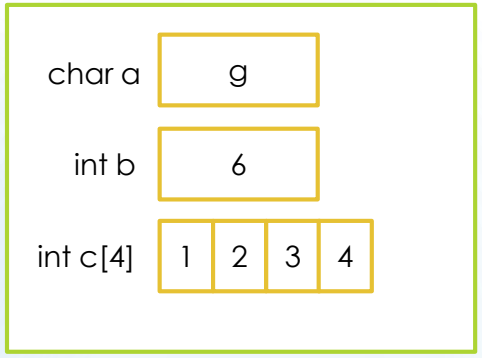
  이 동작을 **"ABC 구조체변수 x를 선언한다"** 라고 함

구조체를 쓰는 이유

- **변수들을 묶은 형태로 그룹화 하고 싶을 때** 사용 함
- 특정 상황에서 소스코드를 보다 간결하게 가능
- 차후 자료구조 및 알고리즘을 배울 때 구조체가 쓰이는데
  이를 위해 미리 익숙해져야 함

구조체 예제

구조체 BBQ의 변수 w




```
struct BBQ
{
    char a;
    int b;
    int c[4];
};

int main()
{
    BBQ w;
    int x;

    w.a = 'g';
    w.b = 6;

    for (x = 0; x < 4; x++)
    {
        w.c[x] = x + 1;
    }

    return 0;
}
```

이렇게 main 밖에다가 **BBQ 구조체를 정의하면** 다른 함수들도 BBQ Type의 구조체 변수를 만들 수 있음

Counting 복습

- 배열에 특정 값이 몇 개 있는지 세는 코딩 방법
- count 변수 하나 만들고
  for문 돌리면서 내가 찾는 값이 발견 될 때마다 count++;

```
int vect[3][3] = {
    {4, 2, 5},
    {6, 3, 1},
    {1, 2, 3}
};

int x, y;
int input;

cin >> input;

for (y = 0; y < 3; y++)
{
    for (x = 0; x < 3; x++)
    {
        if (vect[y][x] == input)
        {
            count++;
        }
    }
}

cout << count;
```

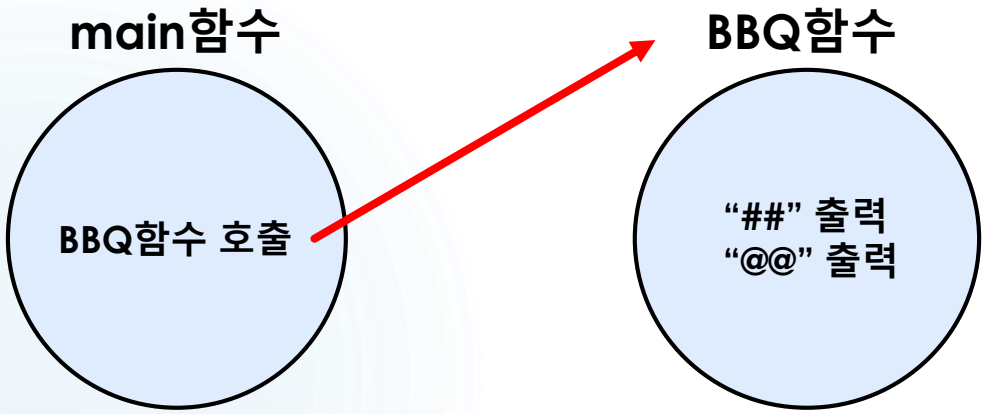
Counting 응용

```
int vect[5] = { 4, 2, 5, 1, 8 };

int cnt;
int x;
int target;

cnt = 0;
cin >> target;
for (x = 0; x < 5; x++)
{
    if (vect[x] == target)
    {
        cnt++;
    }
}

if (cnt > 0)
{
    cout << "발견";
}
else
{
    cout << "없다";
}
```

- 내가 입력받은 숫자를 Counting을 한 후 존재 여부를 판단할 수 있다.

기본 함수호출 예시

- 기본적인 함수호출 방법 예시




```
void BBQ()
{
    cout << "##" << endl;
    cout << "@@" << endl;
}

int main()
{
    BBQ();
    return 0;
}
```

함수 호출 응용 예시1

- x가 3일때 BBQ함수를 호출 하는 소스코드
- 필요할 때 자유롭게 함수를 호출 가능

```
void BBQ()
{
    cout << "##" << endl;
    cout << "@@" << endl;
}

int main()
{

    int x;
    for (x = 0; x < 5; x++)
    {
        if (x == 3)
        {
            BBQ();
        }
    }

    return 0;
}
```

함수 호출 응용 예시2

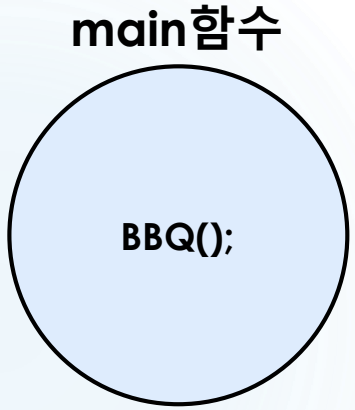
- vect[x] 를 탐색하는데

  vect[x]가 홀수면 print 함수 한번 호출
  vect[x]가 짝수면 print 함수 두번 호출

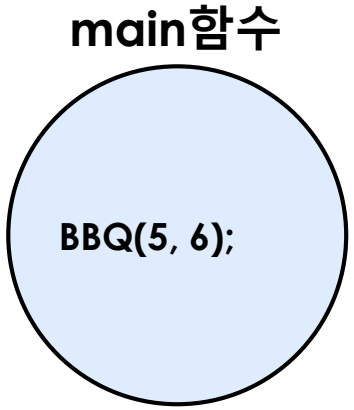
```
void print()
{
    cout << "###" << endl;
}

int main()
{
    int x;
    int vect[5] = { 4, 5, 1, 2, 1 };

    for (x = 0; x < 5; x++)
    {
        if (vect[x] % 2 == 1)
        {
            print();
        }
        else
        {
            print();
            print();
        }
    }

    return 0;
}
```

함수 호출할 때 값을 보내기

- 함수를 호출하면서 값들을 보낼 수 있다.
- 이렇게 숫자나 문자 같은 값들을 보내는 것을 **Call by value** (콜바이벨류)라고 한다.



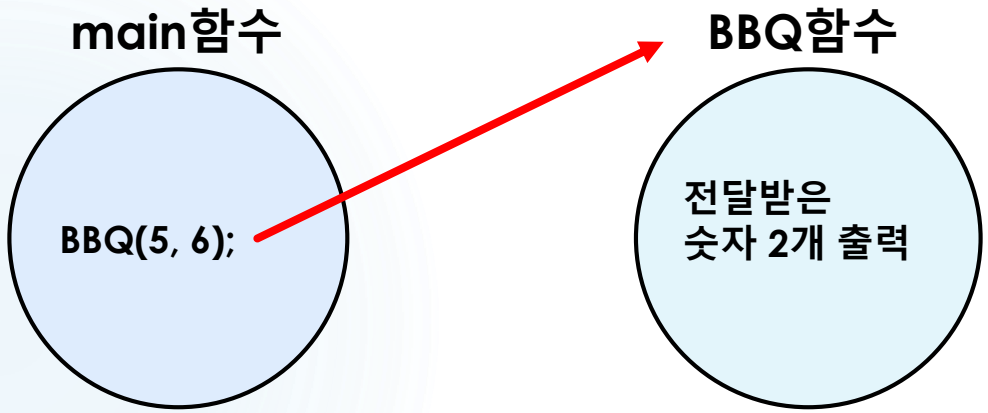
BBQ함수 호출하기
값 보내는 것 없음



BBQ함수 호출하면서
숫자 5와 6을 보냄

Call by value 예제1

- main함수에서 BBQ에 숫자 2개 보내기



변수 a, b에 숫자 5, 6이 들어감

```
void BBQ( int a, int b )
{
    cout << a << b;
}

int main()
{
    BBQ(5, 6);
    return 0;
}
```

Call by value 분석

변수 2개를 만들어 두고,
변수 2개에다가 5, 6을 받음

```
void BBQ( int a, int b )
{
    cout << a << b;
}

int main()
{
    BBQ(5, 6);
    return 0;
}
```

BBQ를 호출하면서
숫자 5, 6을 보냄

- 숫자를 2개를 보냈기 때문에
  변수 2개를 만들어 줘야 받을 수 있다.
- 숫자 n개를 보낸다면
  정확히 변수 n개로 받아줘야 한다.

Call by value 예제2

변수 2개를 생성하고
b, t 값을 복사해서 넣음

```
void BBQ( int a, int b )
{
    cout << a << b;
}

int main()
{
    int b = 2, t = 1;
    BBQ(b, t);
    return 0;
}
```

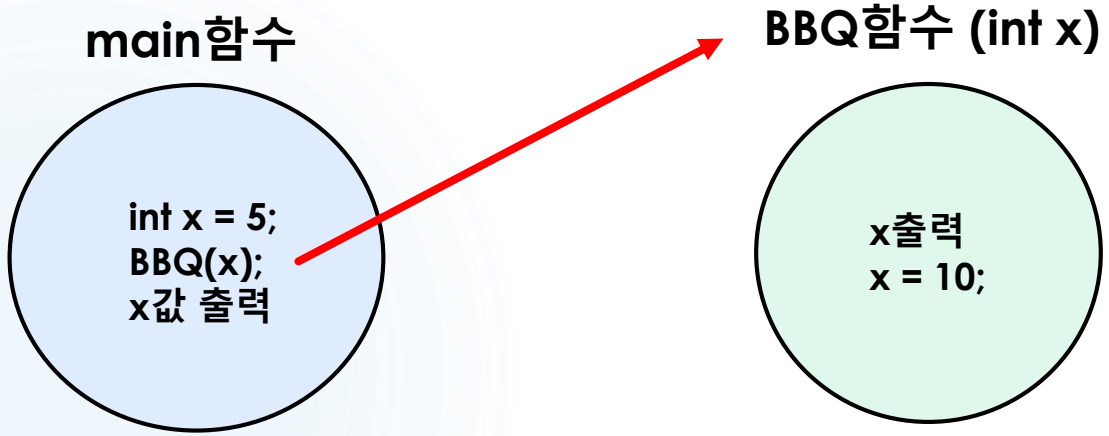
BBQ를 호출하면서
b, t 값을 전달 함

- b와 t 값이 **복사되어**
  a, b 변수 안으로 들어간다.

- b, t 변수 값이 **복사된다는 것을 유의하자**

주의! 이름만 같은 서로 다른 변수

- main 세상에서 만들어지는 변수 x와
  BBQ 세상에서 만들어지는 변수 x는 **서로 다른 변수.**

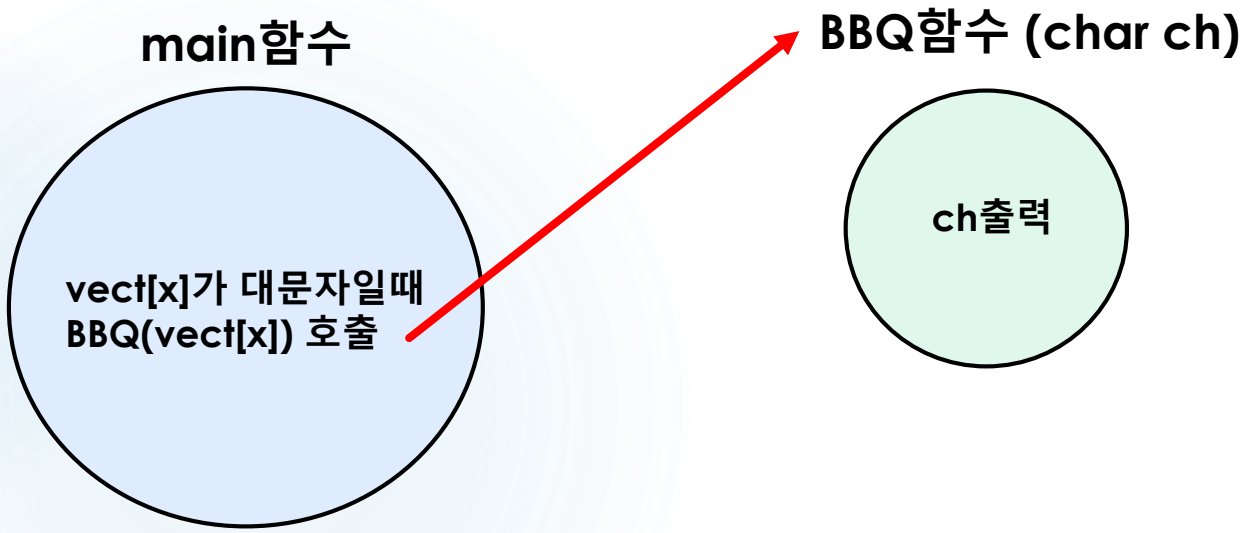


BBQ 함수에서 x 변수에 값을 고치더라도
main함수에서 변수 x의 값은 **수정되지 않음**
main함수의 변수 x와 BBQ함수 안의 변수 x는 다르기 때문

```
void BBQ( int x )
{
    cout << x;
    x = 10;
}

int main()
{
    int x = 5;
    BBQ(x);
    cout << x;
    return 0;
}
```

출력결과 : 5 5

Call by value 예제3

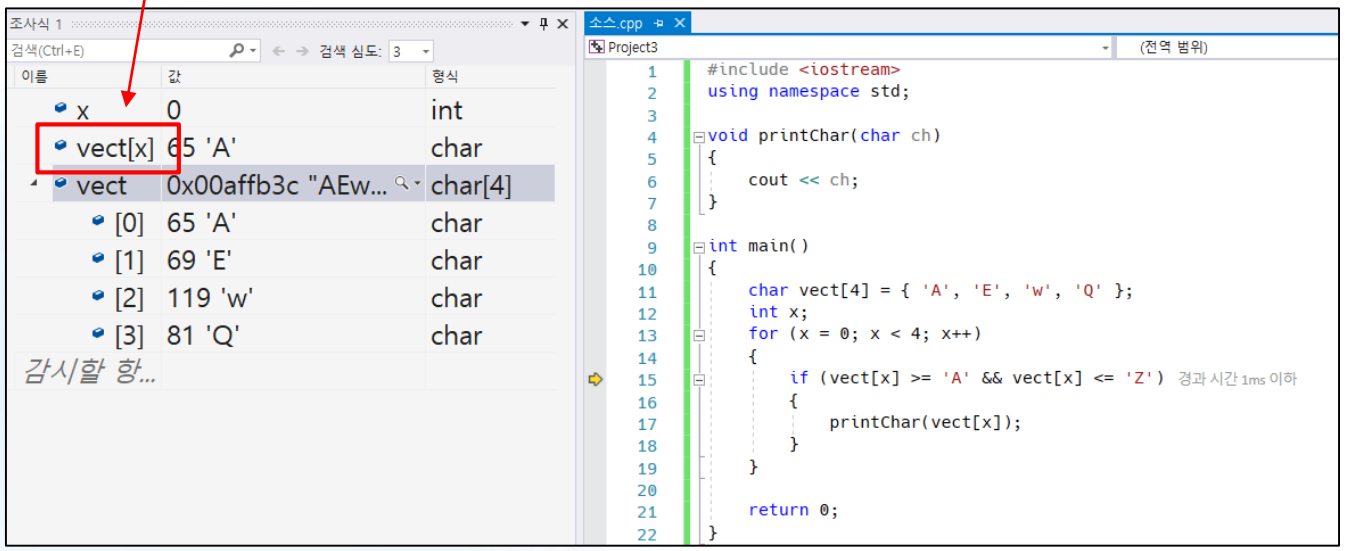



```
void printChar(char ch)
{
    cout << ch;
}

int main()
{
    char vect[4] = { 'A', 'E', 'w', 'Q' };
    int x;
    for (x = 0; x < 4; x++)
    {
        if (vect[x] >= 'A' && vect[x] <= 'Z')
        {
            printChar(vect[x]);
        }
    }

    return 0;
}
```

출력결과 : AEQ

중요! Trace 하실 때 꼭 지켜주세요

조사식에 이렇게 등록시킬 수도 있어요



- F10 버튼을 **누르기 전**
  미리 다음 동작을 예상해야 합니다.

내가 여기서 F10을 누르면
vect[x]가 대문자이기 때문에
if문 안으로 진입을 하겠지?

- 이렇게 예상을 먼저 하고
  F10을 눌러서 동작을 봐야 합니다.

  만약 예상과 다르게 동작되면
  그 곳에 바로 **버그가 발생한 곳** 입니다
  (또는 코드를 잘못 이해 한 것입니다)

- 조사식에 적절한 변수를 등록을 하고
  다음 동작을 예상을 미리 하면서 F10을 눌러야
  정확한 디버깅을 할 수 있습니다.

[중요] Trace할 때 중요한 점

- F10을 눌러놓고, 이 상황에서 값이 맞는지 확인하는 것이 아니라

  F10을 누르기전에 값이 어떻게 바뀔 지 미리 예상을 한 후에 F10 누르는 것.

- **단순하지만, 정말 중요한 디버깅 노하우**

  반드시 예상을 하면서 트레이스 하는 것을 습관 들여주세요.