

민선생코딩학원 시작반

수업노트 LV-13



배우는 내용

1. Call by reference
2. 배열 비교하기
3. 배열 전달하기
4. 백업변수 두기

1. Call by reference

return의 한계

함수를 종료할 때 or 함수에서 값을 돌려줄 때 return 명령어를 사용 함

return값 없음

```
void kfc(int d)
{
    if (d == 9)
    {
        return;
    }

    printf("%d", d);
}
```

0 또는 1을 return 함

```
int kfc(int d)
{
    if (d == 9)
    {
        return 0;
    }

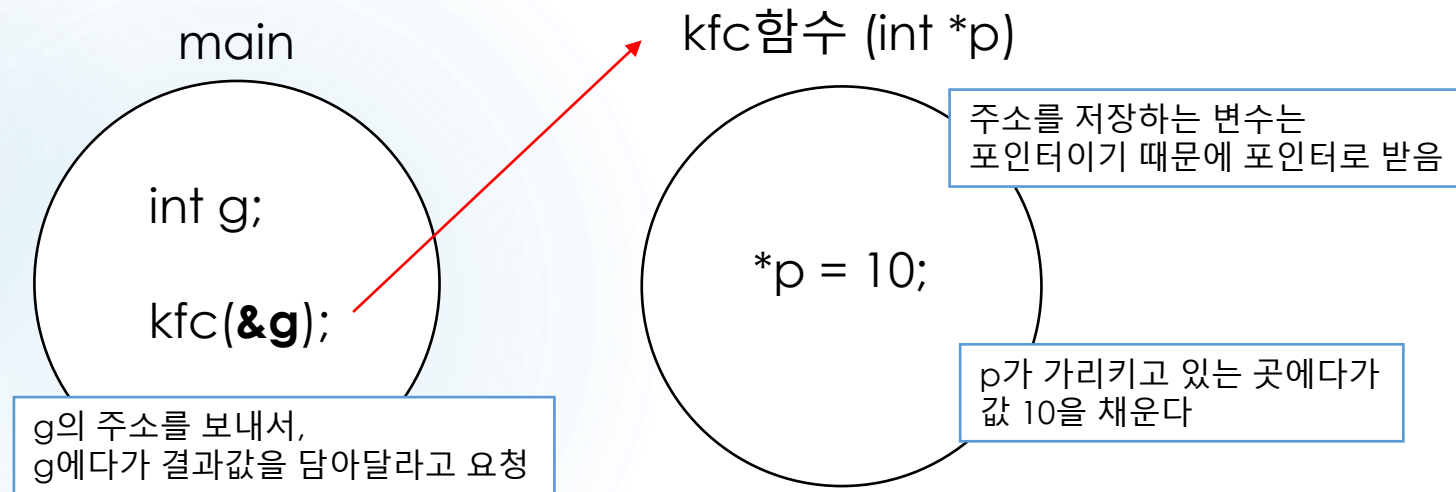
    return 1;
}
```

return의 한계 : 값을 최대 한 개만 return할 수 있음
2개 이상의 값을 return하고 싶을 때는 return명령어로 불가능 함

→ Call by reference로 해결 가능

포인터를 쓰는 방식, Call by reference

[중요] 함수 호출하기 전, **값을 받을 변수를** 먼저 만들어 둔다



```
void kfc(int *p)
{
    *p = 10;
}

int main()
{
    int g;
    kfc(&g);

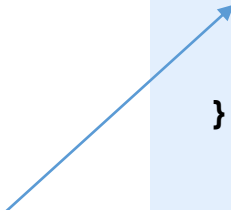
    cout << g;
    return 0;
}
```

여러 개의 값을 return하고 싶을 때

호출할 때 결과값을 받을 '변수의 주소'를 보냄
이 방법을 **Call by reference**라고 함

```
void kfc(int *a, int *b)
{
    *a = 10;
    *b = 5;
}

int main()
{
    int a, b;
    kfc(&a, &b);
    cout << a << " " << b;
    return 0;
}
```



변수 a, b를 준비하고,

a와 b에 값을 담아 달라는 의미

Call by reference 응용

숫자 2개를 입력 받고, 합과 곱을 구해주는 sumGop 함수를 만들어보자.

```
#include <iostream>
using namespace std;

void sumGop(int a, int b, int* sum, int* gop)
{
    *sum = a + b;
    *gop = a * b;
}

int main()
{
    int a, b;
    cin >> a >> b;

    int sum;
    int gop;

    sumGop(a, b, &sum, &gop);
    cout << sum << " " << gop;

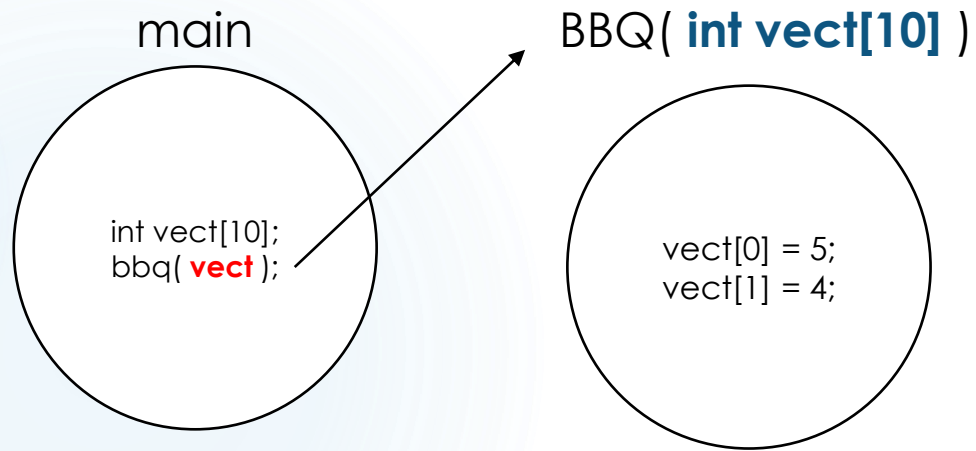
    return 0;
}
```

Call by reference로
숫자 2개를 return하는 것과
같은 효과를 낼 수 있다.

2. 배열 전달하기

배열을 다른 함수로 전달하기

배열을 다른 함수에 보낼 때는 배열 이름만 써주면 됨

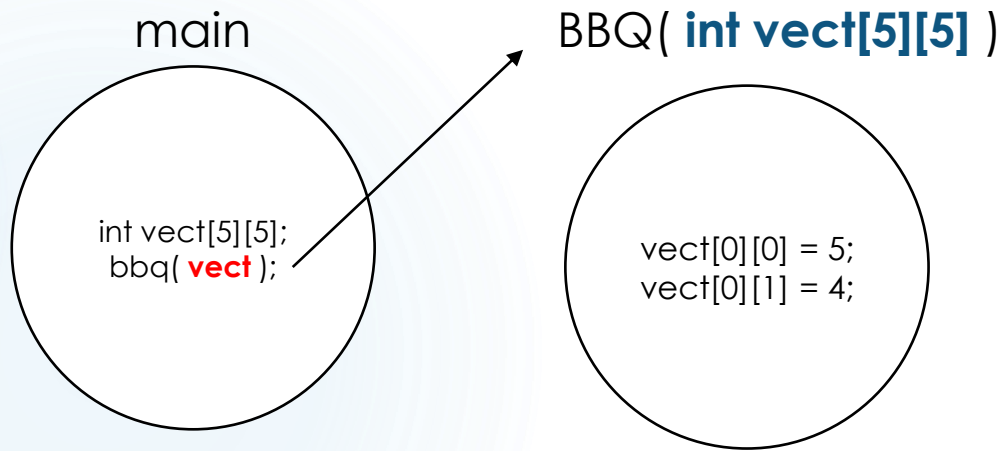


```
void bbq(int vect[10])
{
    vect[0] = 5;
    vect[1] = 4;
}

int main()
{
    int vect[10];
    bbq( vect );
    return 0;
}
```

2차 배열을 다른 함수로 전달하기

1차 배열과 마찬가지로 배열 이름만 써주면 됨

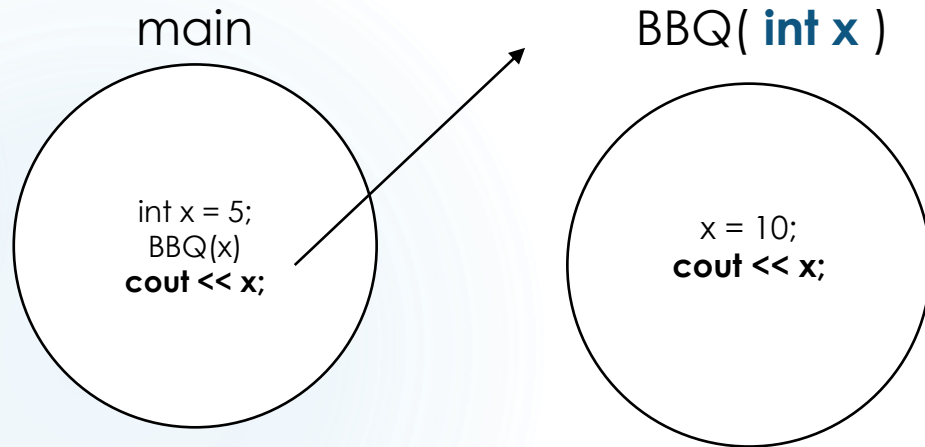


```
void bbq(int vect[5][5])
{
    vect[0][0] = 5;
    vect[0][1] = 4;
}

int main()
{
    int vect[5][5];
    bbq( vect );
    return 0;
}
```

[주의] 같은 이름의 변수

main에 있는 변수 x와 BBQ에 있는 변수 x는 이름만 같을 뿐 **완전히 다른 변수이다.**



```
#include <iostream>
using namespace std;

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

int main()
{
    int x = 5;
    BBQ(x);
    cout << x;

    return 0;
}
```

Micro

10
5

배열의 이름이 의미하는 것

배열의 이름은 "배열 첫 번째 칸의 주소" 를 나타낸다.

```
#include <iostream>
using namespace std;

int main()
{
    int vect[3] = { 1, 2, 3 };
    return 0;
}
```

vect	1	2	3
	0xA	0xB	0xC

여기서 `cout << vect;` 를 수행하면, 0xA가 출력된다.

* 컴퓨터마다 메모리 주소는 달라질 수 있다.

변수 주소를 포인터에 저장하기

포인터는 메모리 주소를 저장할 수 있는 변수이다.

따라서 포인터 p는 vect[0]의 주소를 저장할 수 있다.

배열의 이름이 나타내는 것은 &vect[0] 이다.

따라서 p = vect가 가능하다.

```
#include <iostream>
using namespace std;

int main()
{
    int vect[3] = { 1, 2, 3 };

    int* p;
    p = &vect[0];
    p = vect;

    return 0;
}
```

포인터가 가리키는 값 출력하기

다음 예제에서 p가 저장하고 있는 값은 vect[0]의 메모리 주소이다.

따라서, *p를 출력하면 vect[0] 값이 출력 된다.

```
#include <iostream>
using namespace std;

int main()
{
    int vect[3] = { 1, 2, 3 };

    int* p = vect;

    cout << *p;
    cout << *(p + 0);
    cout << *(p + 1);
    cout << *(p + 2);

    return 0;
}
```

CN Microsoft

1123

[어려움] C언어 문법, 줄임표현

```
#include <iostream>
using namespace std;

int main()
{
    int vect[3] = { 1, 2, 3 };

    int* p = vect;

    cout << *(p + 0);
    cout << p[0];

    cout << *(p + 1);
    cout << p[1];

    cout << *(p + 2);
    cout << p[2];

    return 0;
}
```

*(p + 0) 을, 줄여서 p[0]으로 간추려 표현이 가능하다.

이 간추린 표현으로, 마치 "배열처럼" 포인터를 사용할 수 있다.

배열과 포인터는 같은 것이 아니다.

단지 포인터가 배열의 첫 번째 주소를 가리키고 있을 때,

배열처럼 사용할 수 있는 것이다.

[어려움] 배열 전달의 원리 1

배열을 보낼 때, `abc(vect)` 처럼, 배열의 이름을 적어준다.

배열의 이름은 `vect[0]`의 주소를 나타낸다. 따라서 주소를 보내는 것이다.

```
void abc(int vect[3])
{
}

int main()
{
    int vect[3] = { 1, 2, 3 };
    abc(vect);

    return 0;
}
```

여기서는 보내진 주소를 받아야 한다. (`&vect[0]`)

C언어는 내부적으로 `int vect[3]`을

`void abc(int *vect)`로 바꾸어서 처리한다.

[어려움] 배열 전달의 원리 2

따라서 `int vect[3]` 으로 받지 않고, `int *vect` 라고 써도 된다.

그리고 다음 소스코드 처럼 포인터를 **마치 배열처럼** 사용할 수 있다.

```
#include <iostream>
using namespace std;

void abc(int *vect)
{
    vect[0] = 3; // *(vect + 0) = 3;
    vect[1] = 4; // *(vect + 1) = 4;
    vect[2] = 5; // *(vect + 2) = 5;
}

int main()
{
    int vect[3] = { 1, 2, 3 };
    abc(vect);

    return 0;
}
```

3. 배열 비교하기

같은 글자 수인 문자열 비교하기

(버그발생 소스코드)

```
#include <iostream>
using namespace std;

int main()
{
    char name1[10] = "MIN";
    char name2[10] = "KFC";

    if (name1 == name2)
    {
        cout << "같은문장";
    }
    else
    {
        cout << "다른문장";
    }

    return 0;
}
```

왼쪽 소스코드는 버그 발생 소스코드

버그이유 : 배열은 비교가 되지 않는다.

→ 한 글자씩 비교를 해 주는 소스코드를 작성해야 함

같은 글자수 문자열 비교

Flag를 사용하면 두 문자열을 비교할 수 있다.

만약 다른 글자가 **단 한 글자라도** 발견 된다면

flag = 1 켜고, 종료하기

```
char name1[10] = "MIN";
char name2[10] = "KFC";

int x;
int flag = 0;

for (x = 0; x < 3; x++)
{
    if (name1[x] != name2[x])
    {
        flag = 1;
        break;
    }
}

if (flag == 0) cout << "같은";
else cout << "다름";
```

두 int 배열이 완전히 동일한지 비교

5	1	3	7
---	---	---	---

v2

5	1	4	3
---	---	---	---

1중 for문을 돌며
같은 글자가 있는지 찾는 것이 아니라
다른 글자가 있는지 찾는다

두 int 배열이 완전히 동일한지 비교

5	1	3	7
---	---	---	---

v2

5	1	4	3
---	---	---	---

```
int v1[4] = { 5, 1, 3, 7 };
int v2[4] = { 5, 1, 4, 3 };

int x;
int flag = 0;

for (x = 0; x < 4; x++)
{
    if (v1[x] != v2[x])
    {
        flag = 1;
        break;
    }
}

if (flag == 0) cout << "동일";
else cout << "다른배열";
```

한 글자라도 다르면
flag = 1로 세팅한다

함수를 이용하여 비교하자

배열을 보내는 방식을 사용한다.

두 배열 중 한 글자라도 다른 글자가 발견되면 바로 함수를 종료 시킨다.

for문이 끝날 때 까지 함수가 종료되지 않았다면, **다른 글자가 하나도 없는 것이다.**

따라서 이 때는 return 1을 한다.

```
#include <iostream>
using namespace std;

int compare(int a[4], int b[4])
{
    for (int i = 0; i < 4; i++) {
        if (a[i] != b[i]) return 0;
    }
    return 1;
}
```

```
int main()
{
    int a[4] = { 2, 3, 4, 5 };
    int b[4] = { 2, 3, 6, 5 };

    int ret = compare(a, b);

    if (ret == 1) cout << "같음";
    else cout << "다름";

    return 0;
}
```

4. 백업변수 두기

복습. MAX값 찾기

5	6	1	12	5	2	6
---	---	---	----	---	---	---

배열에서 MAX 값을 구하는 소스코드

- ▶ max변수보다 더 큰 값을 찾을 때 마다
max값을 더 큰 값으로 갱신한다.

```
#include <iostream>
using namespace std;

int main()
{
    int vect[7] = { 5, 6, 1, 12, 5, 2, 6 };
    int max = 0;
    int x;

    for (int x = 0; x < 7; x++)
    {
        if (vect[x] > max)
        {
            max = vect[x];
        }
    }

    cout << max;

    return 0;
}
```

MAX값의 Index 찾기

5	6	1	12	5	2	6
---	---	---	----	---	---	---

MAX값이 있는 Index를 출력 해 보자

- ▶ max값을 찾을 때 마다 max를 갱신하고, x값을 maxIndex 변수에 백업한다.

```
#include <iostream>
using namespace std;

int main()
{
    int vect[7] = { 5, 6, 1, 12, 5, 2, 6 };
    int max = 0;
    int maxIndex;
    int x;

    for (int x = 0; x < 7; x++)
    {
        if (vect[x] > max)
        {
            max = vect[x];
            maxIndex = x;
        }
    }

    cout << maxIndex;

    return 0;
}
```

백업변수 추가

2차원 배열에서 min값의 위치 찾기

2차원 배열에서 min값의 좌표를 찾는 문제

동작원리

min값을 찾을 때마다 min값을 갱신하고,
dy, dx 변수에 y, x 값을 백업 함

```
#include <iostream>
using namespace std;

int main()
{
    int vect[3][3] =
    {
        {3, 4, 1},
        {5, 5, 7},
        {8, 3, 2}
    };

    int min = 999;
    int dy, dx;

    int x, y;
    for (y = 0; y < 3; y++)
    {
        for (x = 0; x < 3; x++)
        {
            if (vect[y][x] < min)
            {
                min = vect[y][x];
                dy = y;
                dx = x;
            }
        }
    }

    cout << "MIN : " << min << endl;
    cout << "(" << dy << ", " << dx << ")" ;

    return 0;
}
```

