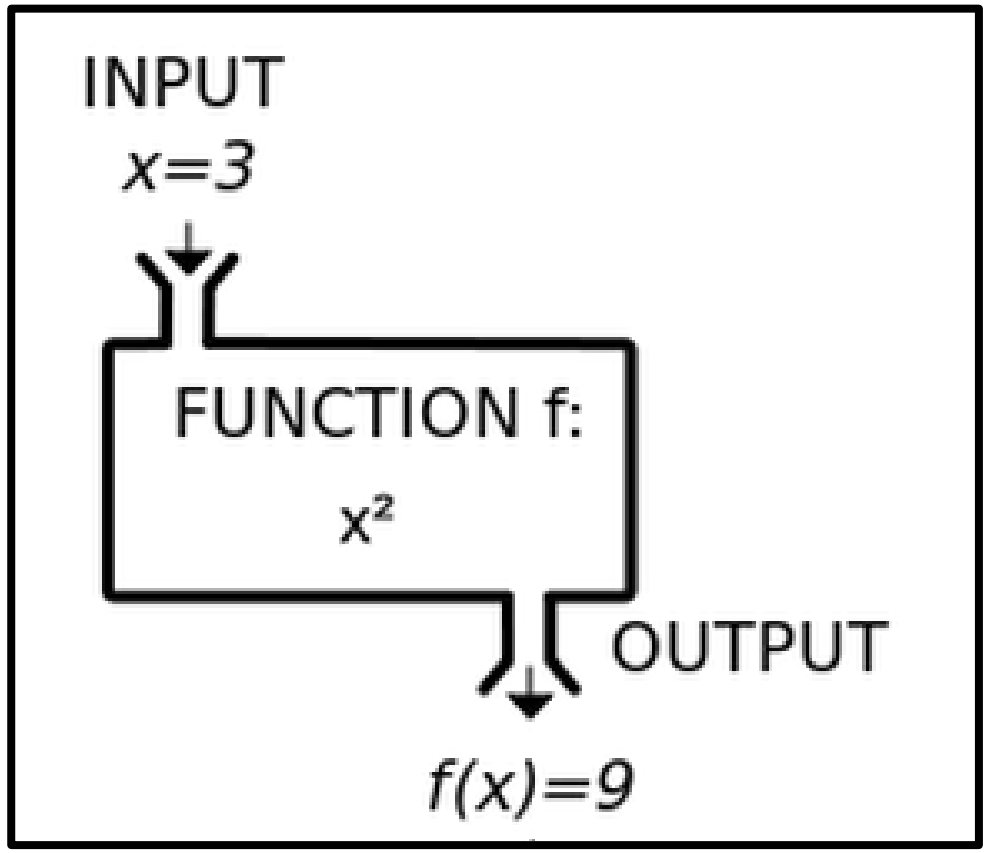
C++ 스터디

2. 함수

D.Com

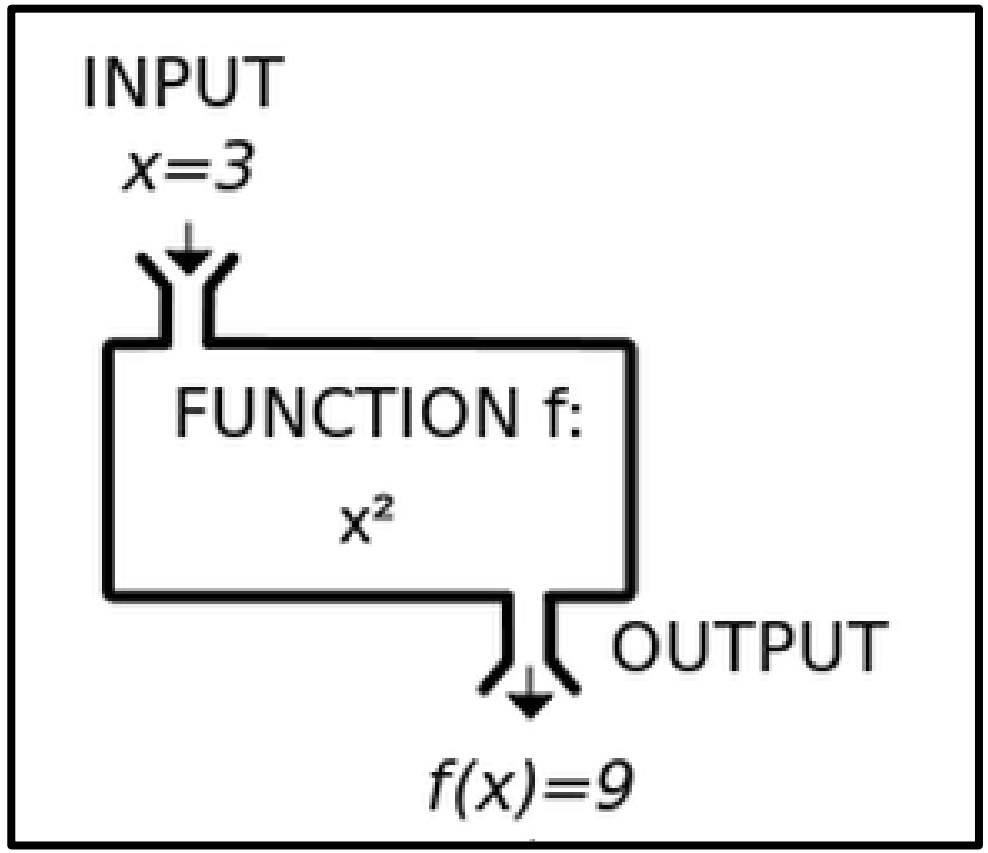
함수의 개념 (수학)

- 수학에서의 함수는 약속된 계산에 이름을 붙인 것이다.



함수의 개념 (C++)

- C++에서의 함수는 약속된 동작에 이름을 붙인 것이다.




```
int iSqrt(int p)
{
    return p * p;
}
```

```
int main() {
    int x, y;
    x = 3;
    y = iSqrt(x);
}
```

Python 때 함수 쓰는 것처럼 쓰되, 타입을 명시 해주는 것이 다르다! 또한 Python의 함수와 마찬가지로 매개변수로 들어간 함수는 함수내의 변수 역할을 한다.

D.Com

함수 이해하기 (선언, 정의, 호출)

- 함수 선언하기 : 함수의 이름과 타입을 정의해준다.

```
int iSqrt(int p);
```

- 함수 정의하기 : 함수의 이름과 타입을 함수의 본체와 합쳐준다.

```
int iSqrt(int p)
{
    return p * p;
}
```

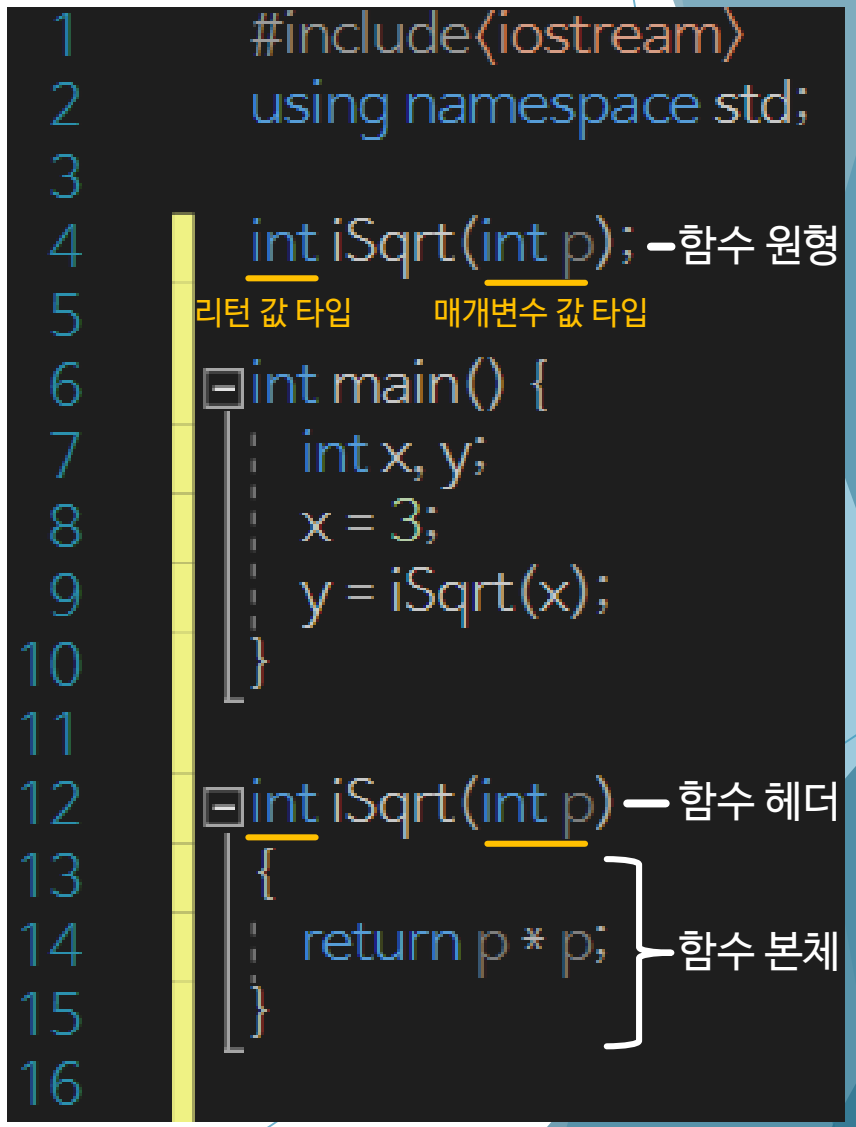
- 함수 호출하기 : 정의된 함수를 특정한 행동을 수행하기 위해 사용한다.

```
int main() {
    int x, y;
    x = 3;
    y = iSqrt(x);
}
```

D.Com

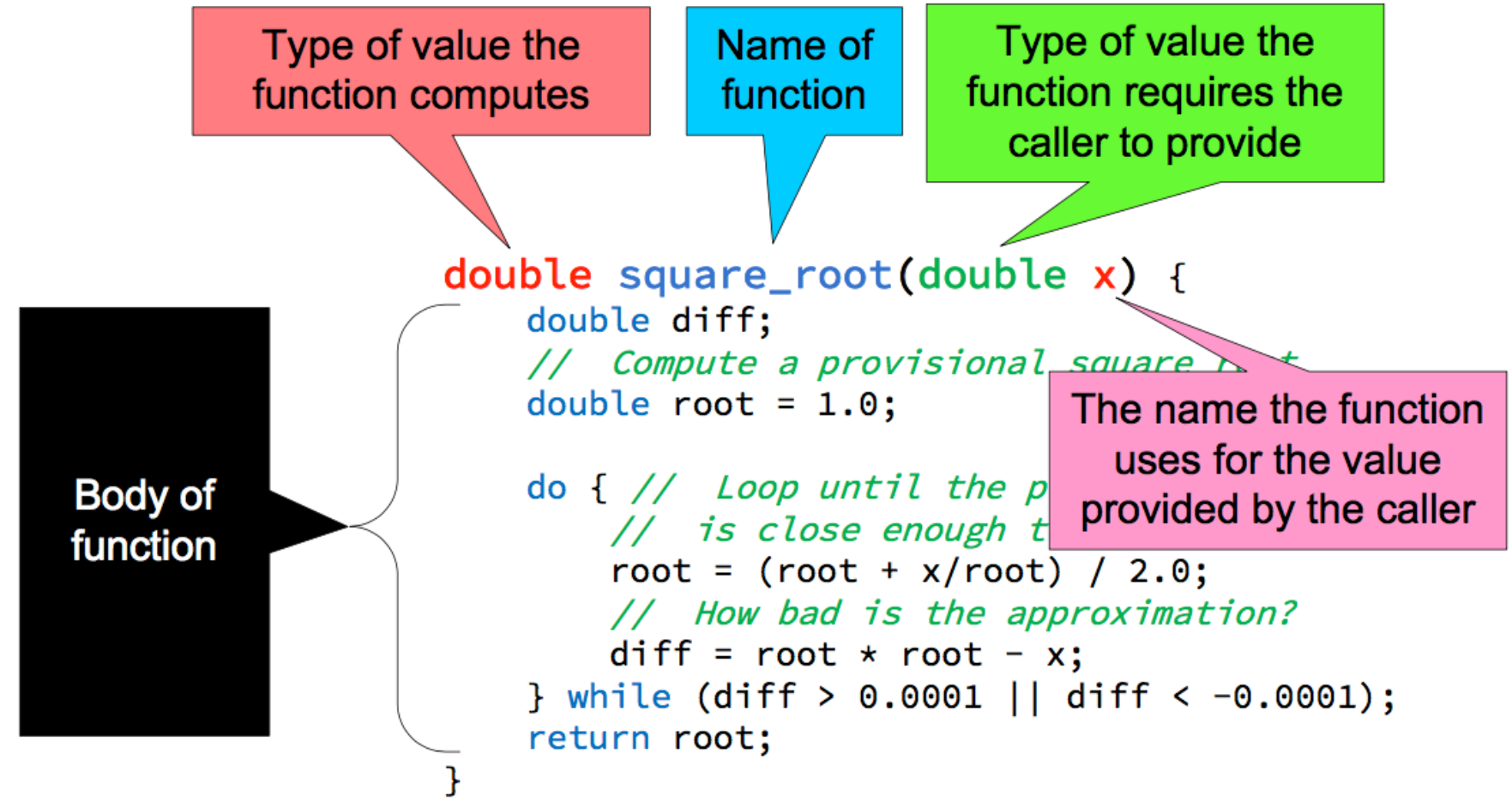
함수 이해하기 (선언, 정의, 호출)

- 선언 부분을 함수 원형(function prototype), 정의 부분을 함수 헤더(function header), 함수 본체(function body)라고 부른다.
- 함수가 짧을 경우 선언과 정의를 합쳐서 만들어도 되지만 대부분은 선언과 호출을 분리한다.
- 선언은 메인 함수 위쪽에, 정의는 아래쪽에 한다. 선언과 정의를 합쳐서 할 경우에는 메인 함수 위쪽에서 한다.
- Python의 함수와 다르게 C++에서는 매개변수의 타입, 리턴 값의 반환 타입을 꼭 명시해주어야 한다. 왜냐하면 C++은 Python과 다르게 변수에도 Type이 존재하기 때문이다.



D.Com

함수 작성하기


Type of value the function computes
Name of function
Type of value the function requires the caller to provide
Body of function
The name the function uses for the value provided by the caller
double square_root(double x) {
double diff;
// Compute a provisional square
double root = 1.0;
do { // Loop until the p
// is close enough t
root = (root + x/root) / 2.0;
// How bad is the approximation?
diff = root * root - x;
} while (diff > 0.0001 || diff < -0.0001);
return root;
}

D.Com

함수 작성하기 (void)

- 매개 변수와 리턴 값이 없을 수도 있다. 그럴 경우 void를 써서 값이 없음을 나타내 준다.

```
void say(void) {
    cout << "제드, 탈론 너프좀" << endl;
}
```

- 매개 변수의 void는 생략이 가능하다.

```
void say() {
    cout << "객프 개꿀잼~" << endl;
}
```

D.Com

함수 작성하기 (예시)

```cpp
#include<iostream>

void prompt() {
    std::cout << "정수를 입력해주세요 : ";
}

int main() {
    int value1, value2, sum;
    std::cout << "이 프로그램은 두 정수를 더합니다.₩n";

    prompt();
    std::cin >> value1;
    prompt();
    std::cin >> value2;

    sum = value1 + value2;
    std::cout << value1 << "+" << value2 << "=" << sum << '₩n';
}
```

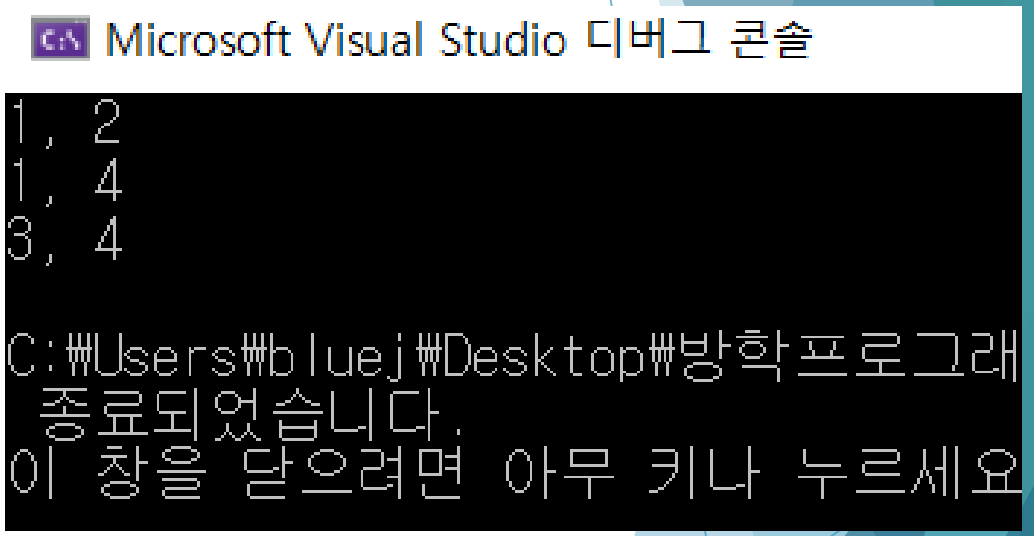
```cpp
#include<iostream>

int prompt() {
    int result;
    std::cout << "정수를 입력해주세요 : ";
    std::cin >> result;
    return result;
}

int main() {
    int value1, value2, sum;
    std::cout << "이 프로그램은 두 정수를 더합니다.₩n";
    value1 = prompt();
    value2 = prompt();
    sum = value1 + value2;
    std::cout << value1 << "+" << value2 << "=" << sum << '₩n';

}
```

D.Com

함수 작성하기 (Default Arguments)

- Python에서 사용했던 것처럼, 파라미터에 아무 값을 입력하지 않았을 때 기본 값을 넣어 줄 수 있다. 먼저 사용했던 매개변수부터 채워준다.

```
1   #include<iostream>
2   using namespace std;
3
4   void point(int x = 3, int y = 4) {
5       cout << x << ", " << y << endl;
6   }
7
8   int main() {
9       point(1, 2);
10      point(1);
11      point();
12  }
```



D.Com

지역 변수 (Local Variables)

- 변수(Variable)은 정의된 장소에 따라 사용 범위를 가지게 된다.
- 지역 변수(Local Variables)는 함수 내부에서 생성되어 스택(Stack)에 저장되며 선언된 함수 내부에서 사용되다가 함수가 종료되면 소멸한다.
- 우측의 프로그램에서 2번과 3번이 지역 변수이다. 지역 변수는 선언된 함수 내에서만 사용할 수 있다.
- 주석 처리된 문장 10번에서 num2는 main 함수의 지역 변수이기에 ex_func 함수에서 사용할 수 없다.
- 스택과 지역변수의 life cycle에 대한 자세한 설명은 아래 참조

  http://tcpschool.com/c/c_memory_stackframe

```
1   #include <iostream>
2   using namespace std;
3
4   int num1 = 100; // 1번
5
6   void ex_func() {
7       int num3 = 300; // 2번
8       cout << num3 << endl;
9       cout << num1 << endl;
10      //cout << num2 << endl;
11  }
12
13  int main() {
14      int num2 = 200; // 3번
15      cout << num1 << endl;
16      cout << num2 << endl;
17      ex_func();
18      return 0;
19  }
```

D.Com

전역 변수 (Global Variables)

- 지역 변수와 다르게 전역 변수는 프로그램이 시작될 때 생성되어 데이터(Data)에 저장되며 프로그램 전체에서 사용 가능하며 프로그램이 종료되면 소멸한다.

- 우측 프로그램에서 1번이 전역 변수로 함수 밖에서 생성 되었다.

- ex_func 함수 안에서 num1을 선언하거나 매개 변수로 주지 않았음에도 쓸 수 있는 이유가 num1이 전역 변수이기 때문이다.

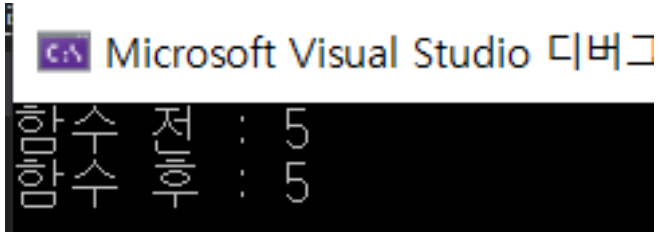
```
1   #include <iostream>
2   using namespace std;
3
4   int num1 = 100; // 1번
5
6   void ex_func() {
7       int num3 = 300; // 2번
8       cout << num3 << endl;
9       cout << num1 << endl;
10      //cout << num2 << endl;
11  }
12
13  int main() {
14      int num2 = 200; // 3번
15      cout << num1 << endl;
16      cout << num2 << endl;
17      ex_func();
18      return 0;
19  }
```

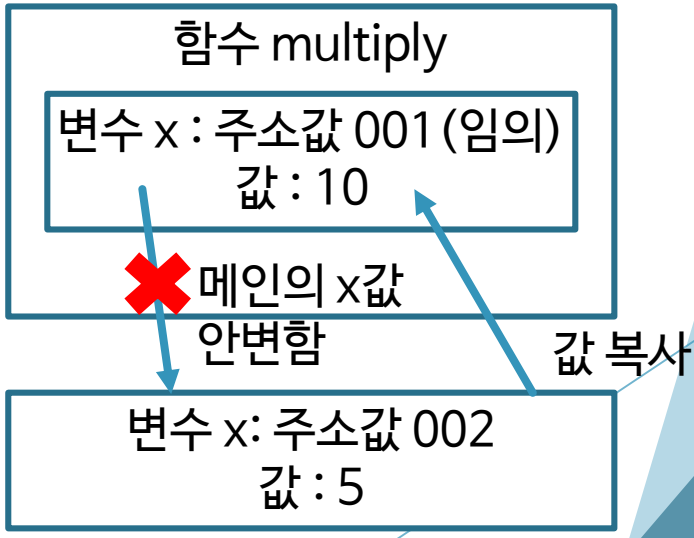
D.Com

Pass By Value, Pass By Reference

- C++에서는 기본적으로 함수 호출시에 인자로 넘겨준 변수의 값을 복사하여 파라미터로 전달해준다. 따라서 아래 프로그램에서 파라미터로 받은 변수의 값을 변경해도 인자로 넘겨준 원래 변수의 값은 변경되지 않는다.

```
1   #include <iostream>
2   using namespace std;
3
4   void multiply(int x) {
5       x *= 2;
6   }
7
8   int main() {
9       int x = 5;
10      cout << "함수 전 : " << x << endl;
11      multiply(x);
12      cout << "함수 후 : " << x << endl;
13  }
```



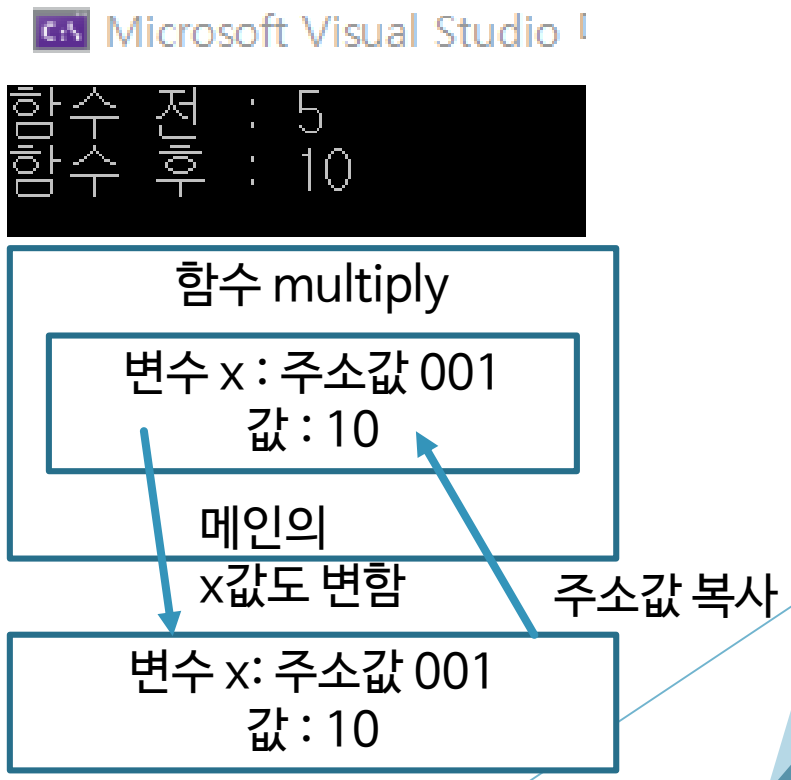


D.Com

Pass By Value, Pass By Reference

- 하지만 Pass By Reference로 넘겨주면 함수 안에서도 외부 매개 변수를 수정 가능하게 된다. 매개변수를 선언할 때 “타입& 변수 명” 으로 선언하게 되면 함수의 매개변수는 Pass By Reference로 값을 받게 된다. 주소 값을 참조한다는 뜻이다. 그렇게 되면 main에 있는 x를 multiply 함수에서 참조할 수 있게 되어 main에 있는 x의 값을 조정할 수 있다.

```
1   #include <iostream>
2   using namespace std;
3
4   void multiply(int& x) {
5       x *= 2;
6   }
7
8   int main() {
9       int x = 5;
10      cout << "함수 전 : " << x << endl;
11      multiply(x);
12      cout << "함수 후 : " << x << endl;
13  }
```



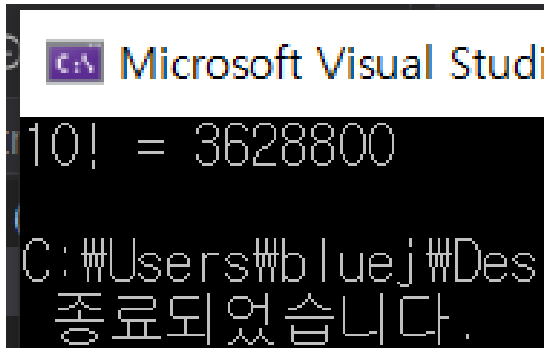
D.Com

재귀 함수

- 재귀함수란 어떤 함수에서 자신을 다시 호출하여 수행하는 방식의 함수이다. 반복분을 사용하는 코드는 모두 재귀함수를 통해 구현이 가능하다.
- 재귀함수를 쓰는 이유에 대해서는 아래 링크 참조.

  https://medium.com/sjk5766/재귀함수를-쓰는-이유-ed7c37d01ee0

- 대표적인 재귀 함수로는 팩토리얼이 있다.




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

int factorial(int n) {
    if (n == 0)
        return 1;
    else
        return n * factorial(n - 1);
}

int main() {
    cout << "10! = " << factorial(10) << endl;
}
```

재귀 함수

- 재귀함수 동작 과정

```
1   #include <iostream>
2   using namespace std;
3
4   int factorial(int n) {
5       if (n == 0)
6           return 1;
7       else
8           return n * factorial(n - 1);
9   }
10
11  int main() {
12      cout << "10! = " << factorial(10) << endl;
13  }
```

```
factorial(6) = 6 * factorial(5)
             = 6 * 5 * factorial(4)
             = 6 * 5 * 4 * factorial(3)
             = 6 * 5 * 4 * 3 * factorial(2)
             = 6 * 5 * 4 * 3 * 2 * factorial(1)
             = 6 * 5 * 4 * 3 * 2 * 1 * factorial(0)
             = 6 * 5 * 4 * 3 * 2 * 1 * 1
             = 720
```

D.Com

Referance Variable

- &를 변수를 선언할 때 사용할 수도 있다.

```
int x;
int& r = x;
```

- r을 참조 변수라고 하며 r이 x와 같은 메모리 공간을 나타내는 변수가 된다. (둘이 같아진다고 보면 된다.)
- r을 수정하면 x도 변하고, x를 수정하면 r도 변하게 된다.

D.Com

과제 2번

- 랩 2번을 푸세용~

D.Com