SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

삼성 청년 SW 아카데미

알고리즘

<알림>

본 강의는 삼성 청년 SW아카데미의 컨텐츠로
보안서약서에 의거하여
강의 내용을 어떠한 사유로도 임의로 복사,
촬영, 녹음, 복제, 보관, 전송하거나
허가 받지 않은 저장매체를
이용한 보관, 제3자에게 누설, 공개,
또는 사용하는 등의 행위를 금합니다.

SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

Day5. 재귀

SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

사전 지식

지역 변수 vs 전역 변수

코드에서 { } 는 하나의 지역을 나타낸다.

지역변수 : { } 안에서 생성된 변수

전역변수 : { } 에 둘러 쌓여 있지 않은 변수

- 변수 a 만 전역 변수이고,
- 나머지 b, c, d, i 는 모두 지역 변수이다.

```
int a;
int main() {
    int b;

    if (a < 10) {
        int c;
    }
    for (int i = 0; i < 10; i++) {
        int d;
    }

    return 0;
}
```

코드를 기능 단위로 묶은 것

다양한 형식이 있다.

```
return type 함수이름 ( 매개변수 ) {
        명령문;
        return 값;
}
```

```
void hello() {
    cout << "HELLO";
}
```

```
void hello(string name) {
    cout << name << " HELLO";
}
```

```
int hello(string name) {
    cout << name << " HELLO";
    return 1;
}
```

함수의 변수 호출은 전역 변수가 우선이다.

```
10
```

```
int a = 10;

void hello() {
    int a = 20;
}
int main() {
    cout << a;
    return 0;
}
```

SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

재귀함수 기초

나 자신을 호출하는 함수이다.

- Let me introduce myself
- 러시아 그 인형

```
void hello() {
    hello();
}
```

재귀 함수를 학습하는 이유

재귀 함수는 정말 매우 확실히 리얼리 다양하게 활용된다.

- DFS, 미로찾기, 순열/조합 과 같은 탐색 및 백트래킹 알고리즘
- 병합 정렬, 퀵 정렬, 이진 탐색과 같은 분할 정복 알고리즘
- 팩토리얼, 피보나치 수열 과 같은 수학적 계산 및 DP 알고리즘
- 이진 트리 순회, 그래프 탐색 과 같은 트리와 그래프 구조의 순회

재귀 함수 만들기

코드를 작성하고 빌드한다.

```
void hello() {
    cout << "HELLO" << '\n';
    hello();
}
int main() {
    hello();

    return 0;
}
```

https://gist.github.com/hoconoco/d2f0a62eed7750a1f77872d60ae70083

계속 반복하는 재귀 함수를 멈추려면?

특정 횟수가 되면 멈춰야 한다. (종료 시점==기저조건)

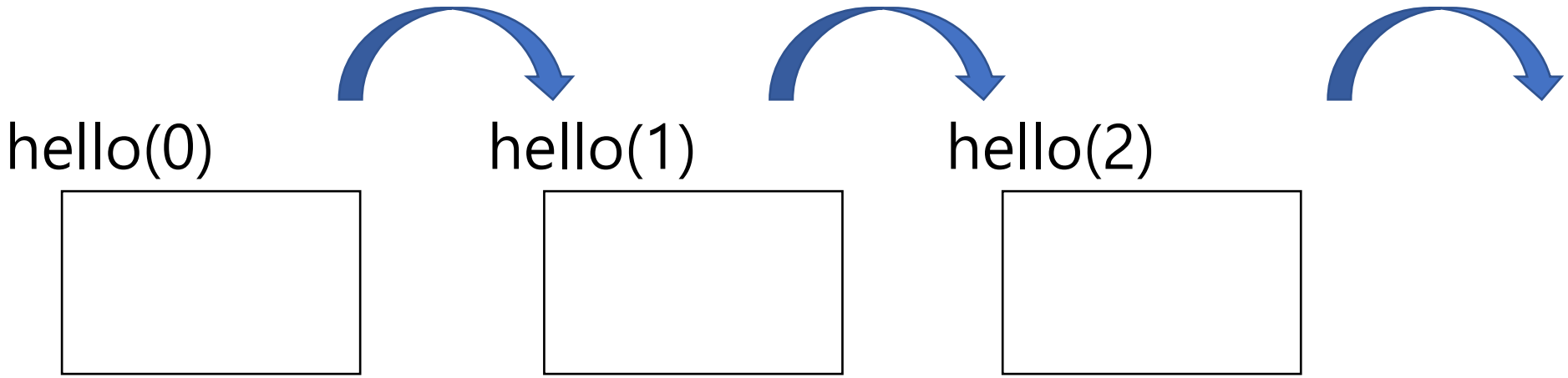
- **호출되는 함수에 indexing 을 한다.**
- **특정 횟수가 되면 다시 돌아가게 한다.**

재귀 함수 종료하기 1

재귀 함수에 번호를 붙인다.

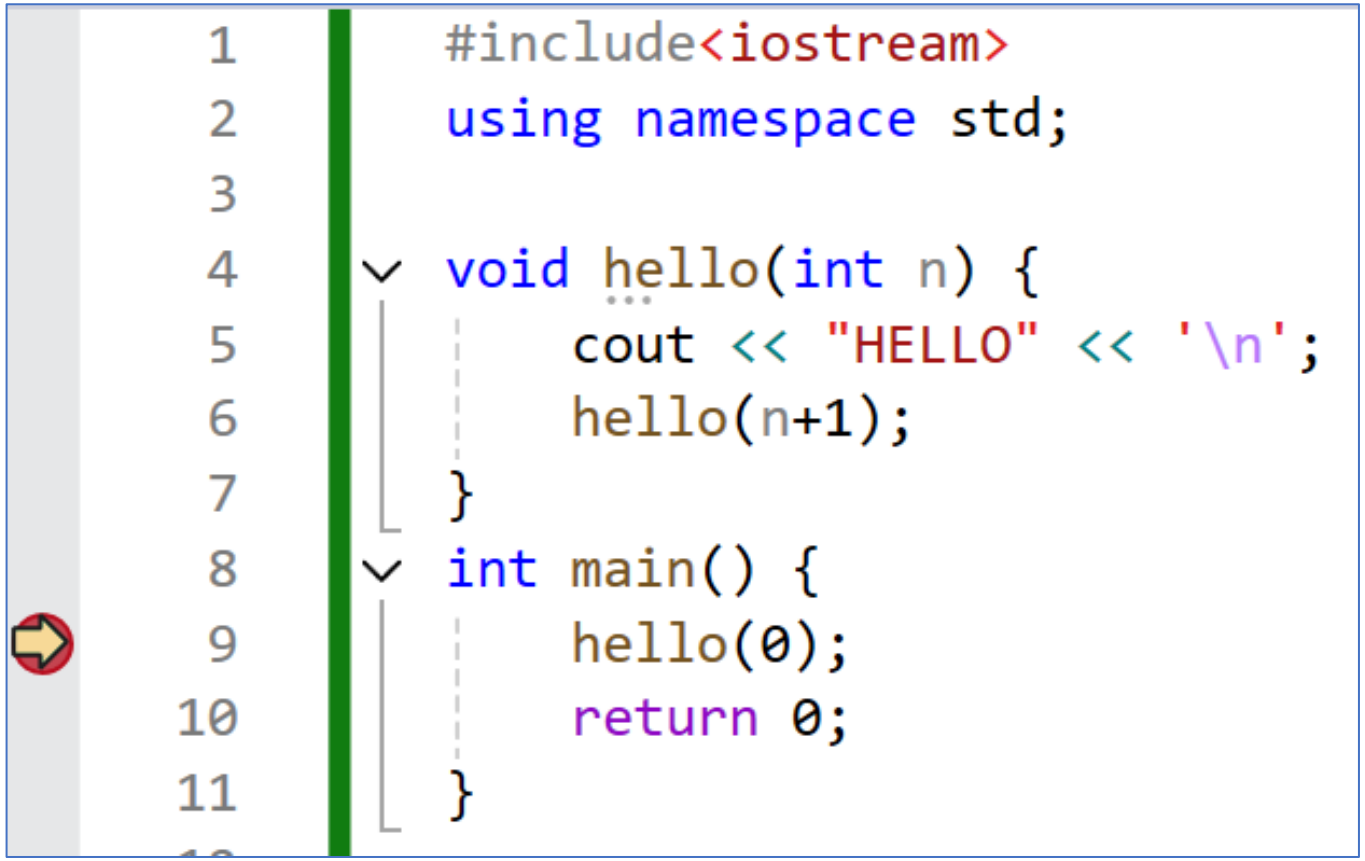
- 디버깅으로 확인하자.

```
void hello(int n) {
    cout << "HELLO" << '\n';
    hello(n+1);
}
```



함수 호출 스택 확인하기 1

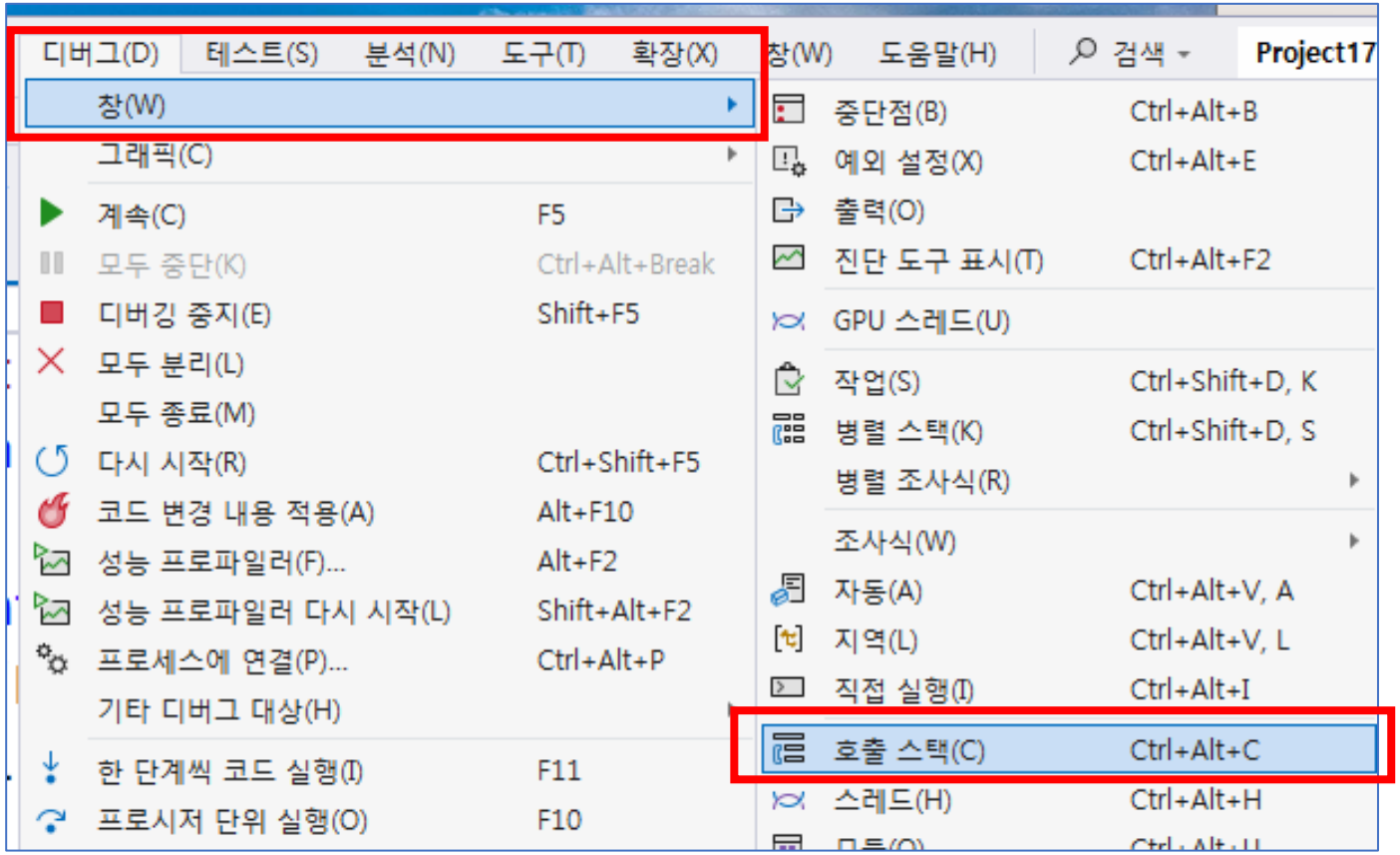
hello(0) 에 Break Point(F9) 걸고
디버깅(F5)를 누른다.



```
1   #include<iostream>
2   using namespace std;
3
4   void hello(int n) {
5       cout << "HELLO" << '\n';
6       hello(n+1);
7   }
8   int main() {
9       hello(0);
10      return 0;
11  }
```

함수 호출 스택 확인하기 2

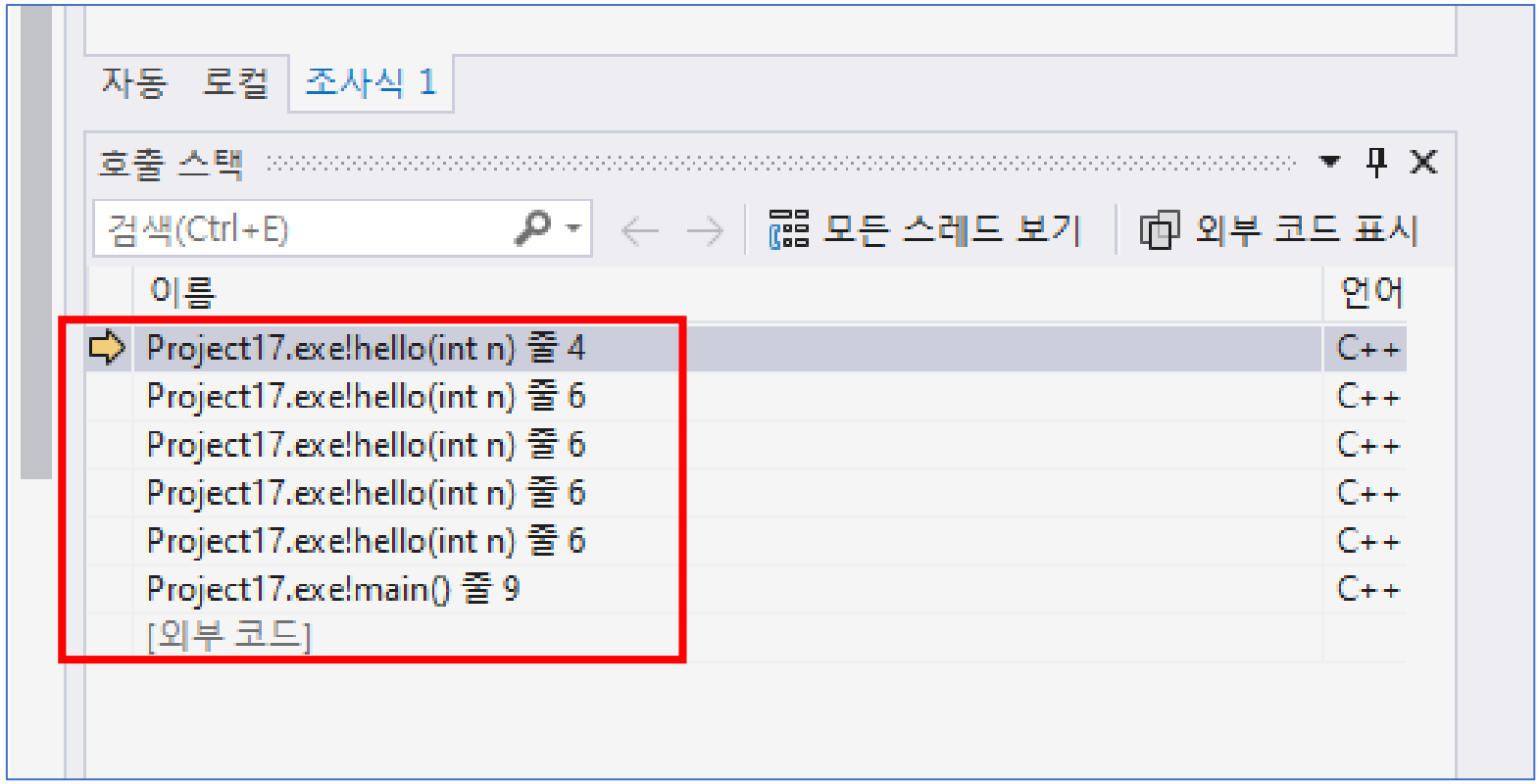
디버그 → 창 → 호출 스택 클릭



함수 호출 스택 확인하기 3

F11 키를 누른다.

호출 스택 창에 함수가 계속 쌓인다.

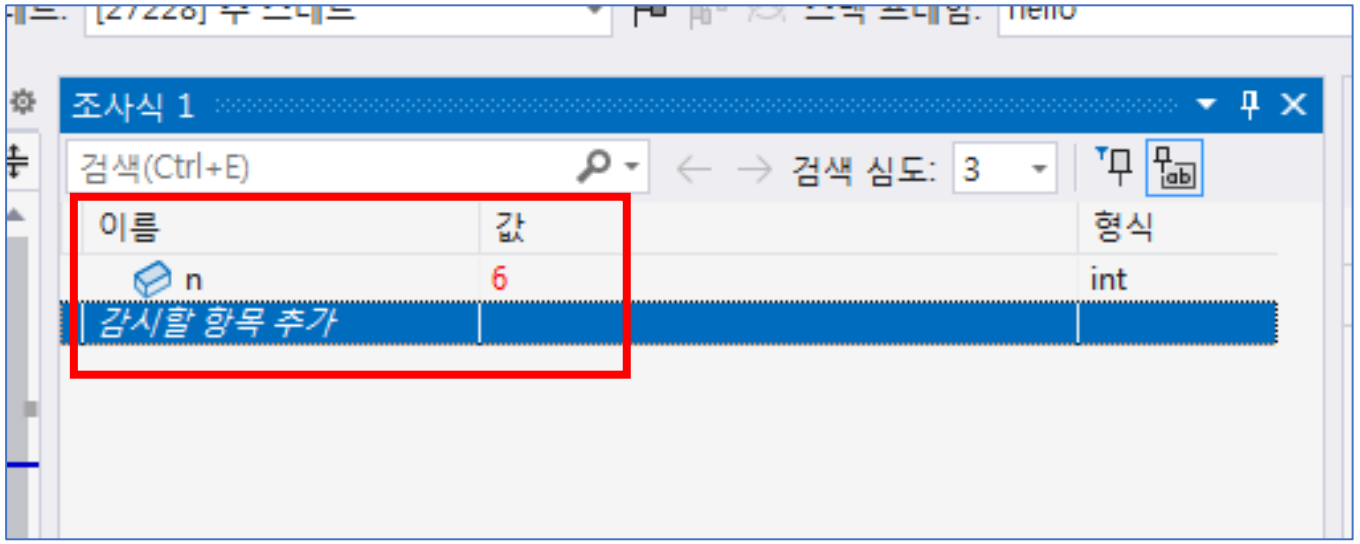


함수 호출 스택 확인하기 4

조사식 창도 띄워서 n 값의 변화를 같이 살펴본다.

조사식 창 추가하기

- **디버그 → 창 → 조사식 → 조사식1 클릭**



재귀 함수 종료하기 2

특정 횟수가 되면 다시 돌아가게 한다.

- **종료 시점**

코드를 작성하고 디버깅한다.

```
void hello(int n) {
    if (n >= 10) {
        return;
    }
    cout << "HELLO" << '\n';
    hello(n+1);
    int dummy; //dummy 코드
}
```

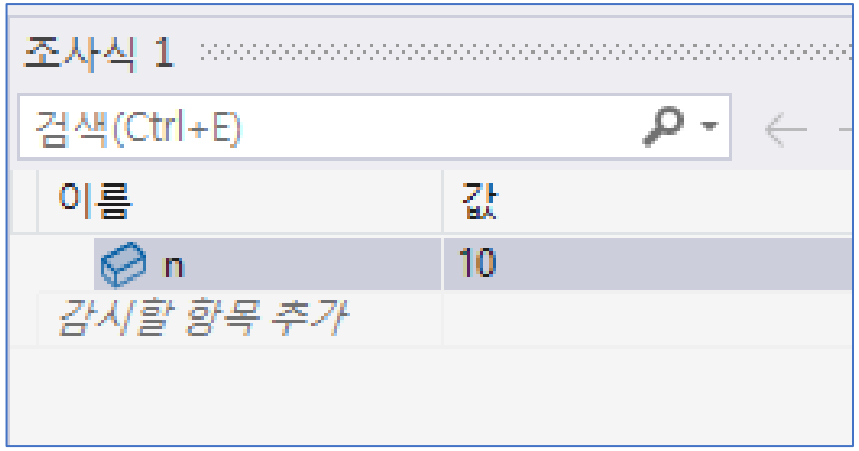
https://gist.github.com/hoconoco/e669bbf2ef7383617d1efd7e93022eaf

함수 호출 스택 확인하기

int dummy 에 Break point(F9)

디버깅(F5)를 하고, F11을 눌러서 호출 스택 창을 살펴본다!

n 이 10 이상이 되면 어떤 일이 발생할까?



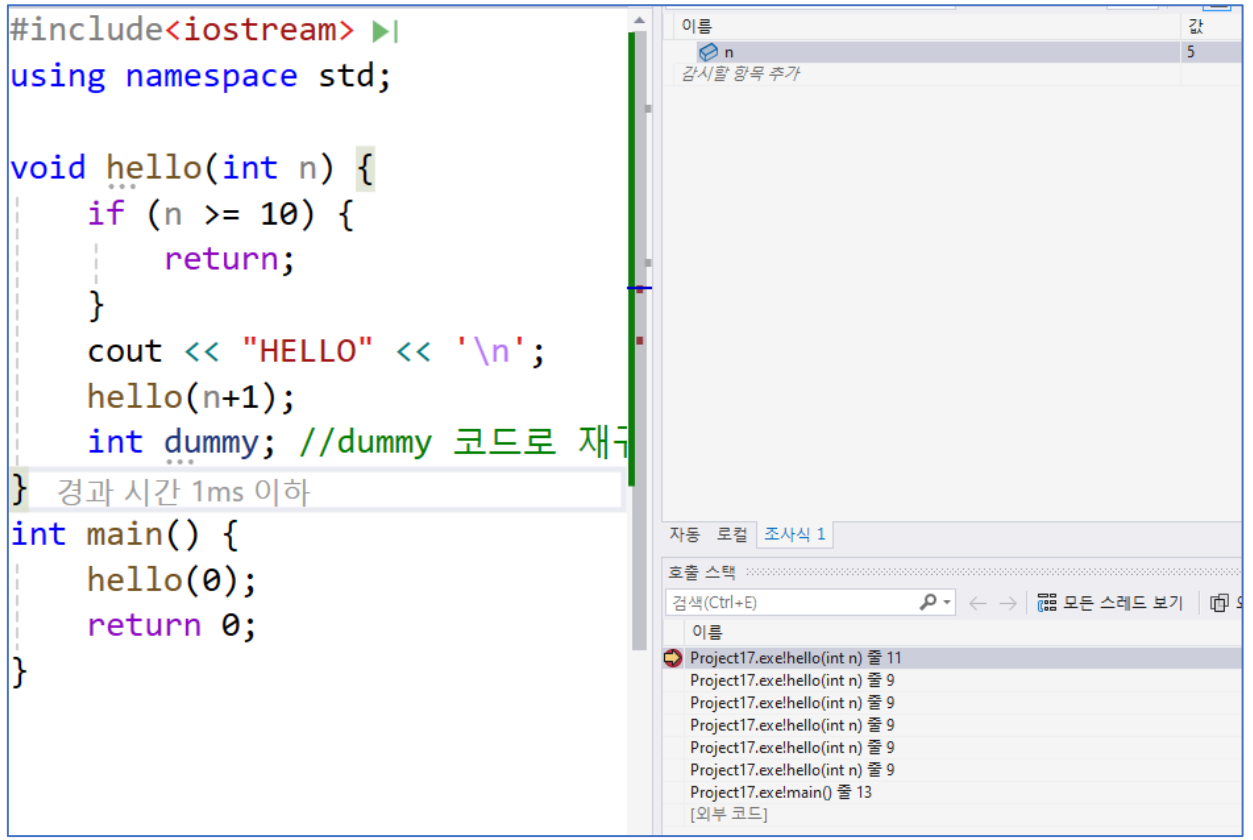

```
3
4   void hello(int n) {
5       if (n >= 10) {
6           return; 경과 시간
7       }
8       cout << "HELLO" <<
9       hello(n+1);
10      int dummy; //dummy
11  }
```

재귀 함수의 동작 원리를 파악한다

재귀함수가 종료하는 것이 아니라 다시 처음으로 돌아간다

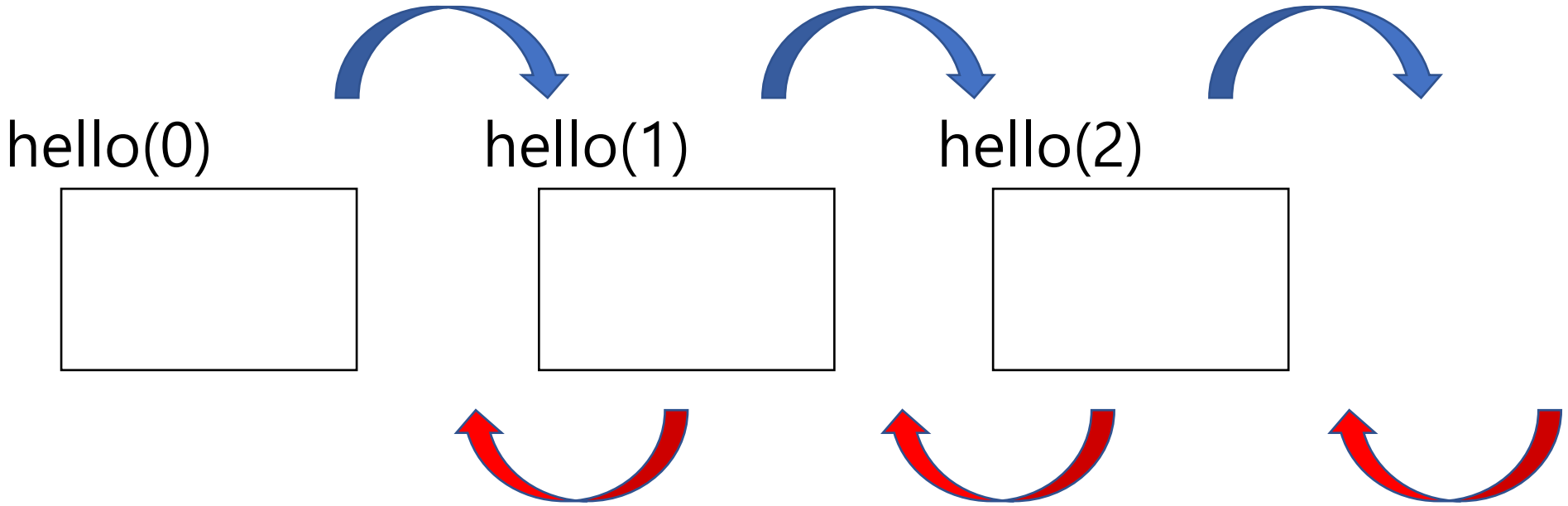
다시 n이 줄어들고, 호출 스택에 쌓인 함수가 사라진다.

이것이 재귀 함수랄까?!



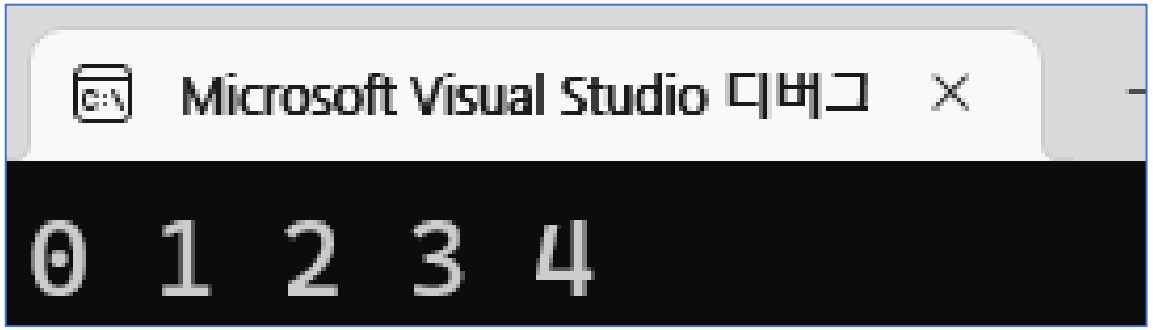
재귀 함수의 동작 원리

재귀 함수는 종료 시점을 만나면 다시 돌아간다!



왼쪽 기본 코드로 0 1 2 3 4 출력한다.

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



종료 시점의 기준을 잘 잡아야 한다.

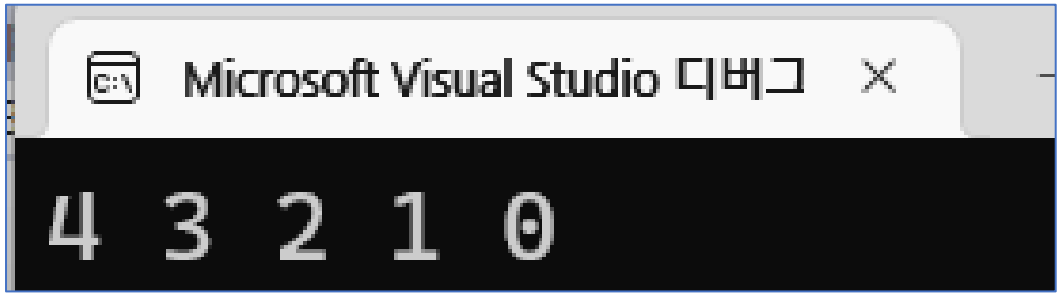
```
void func(int level) {
    //종료 시점
    if (level >= 5) {
        return;
    }
    cout << level << " ";
    func(level + 1);
}
```

https://gist.github.com/hoconoco/dfe4f34ba8aaf02fd264a103ab40e679

왼쪽 기본 코드로 4 3 2 1 0 출력한다.

- 힌트! : 재귀 함수가 종료 시점 후 호출 스택에서 어떻게 동작했는 지 다시 한번 떠올려 본다.

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



QUIZ 2 정답

기본 골조를 바꾸지 않아도 원리만 이해하면 할 수 있다!

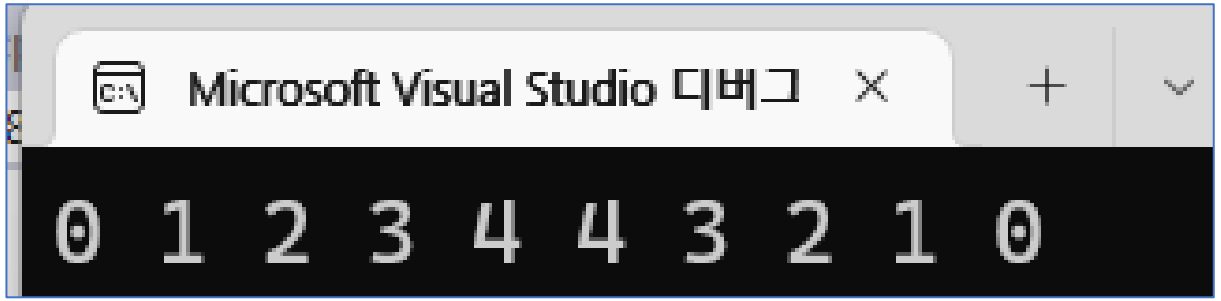
```
void func(int level) {
    //종료 시점
    if (level >= 5) {
        return;
    }
    func(level + 1);
    cout << level << " ";
}
```

https://gist.github.com/hoconoco/cab0ad89c4bb7f61e4b804e2a7f619cc

왼쪽 기본 코드를 갖고 0 1 2 3 4 4 3 2 1 0 을 호출한다.

- QUIZ1,2 로 단련된 이 몸이라면 할 수 있지!

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



QUIZ 3 정답

재귀 함수의 기본 구조를 파악하였다!

```
void func(int level) {
    if (level >= 5) {
        return;
    }
    cout << level << " ";
    func(level + 1);
    cout << level << " ";
}
```

https://gist.github.com/hoconoco/7d62c985aa2e9b1a1599ffa2161d06a3

재귀 함수 기본 구조

재귀 함수의 원리를 이해하고

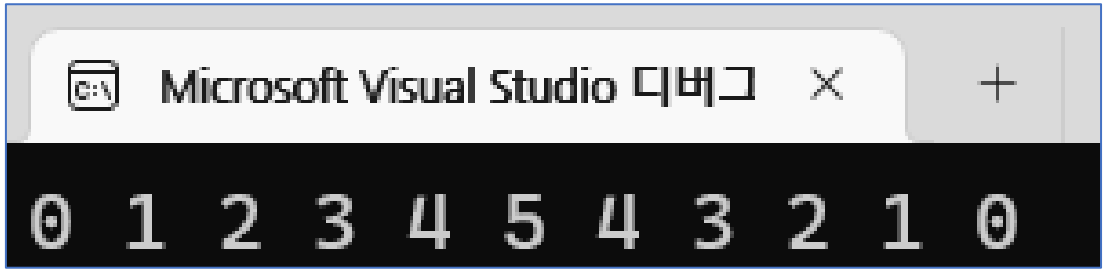
기본 구조를 깨우쳤다면 이제 문제를 풀 시간이다

```
void func(int level) {
    //종료 시점, 언제 종료할 것인가?
    if (level >= 5) {
        return;
    }
    //함수를 실행하면서 처리되는 코드
    cout << level << " ";
    //재귀 함수
    func(level + 1);
    //다시 돌아오면서 처리되는 코드
    cout << level << " ";
}
```

왼쪽 기본 코드를 갖고 0 1 2 3 4 5 4 3 2 1 0 을 호출한다.

- **이제 진짜 끝이야!!**

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



QUIZ 4 정답

그래. 자네가 재귀함수 마스터인가?

- **사실 우린 재귀 함수 기초를 마스터 한 것이다 하하**

```
void func(int level) {
    if (level >= 5) {
        cout << level << " ";
        return;
    }
    cout << level << " ";
    func(level + 1);
    cout << level << " ";
}
```

https://gist.github.com/hoconoco/3a98e2dedd84d6d49eb342c55c2031bc

SAMSUNG SW ACADEMY FOR YOUTH

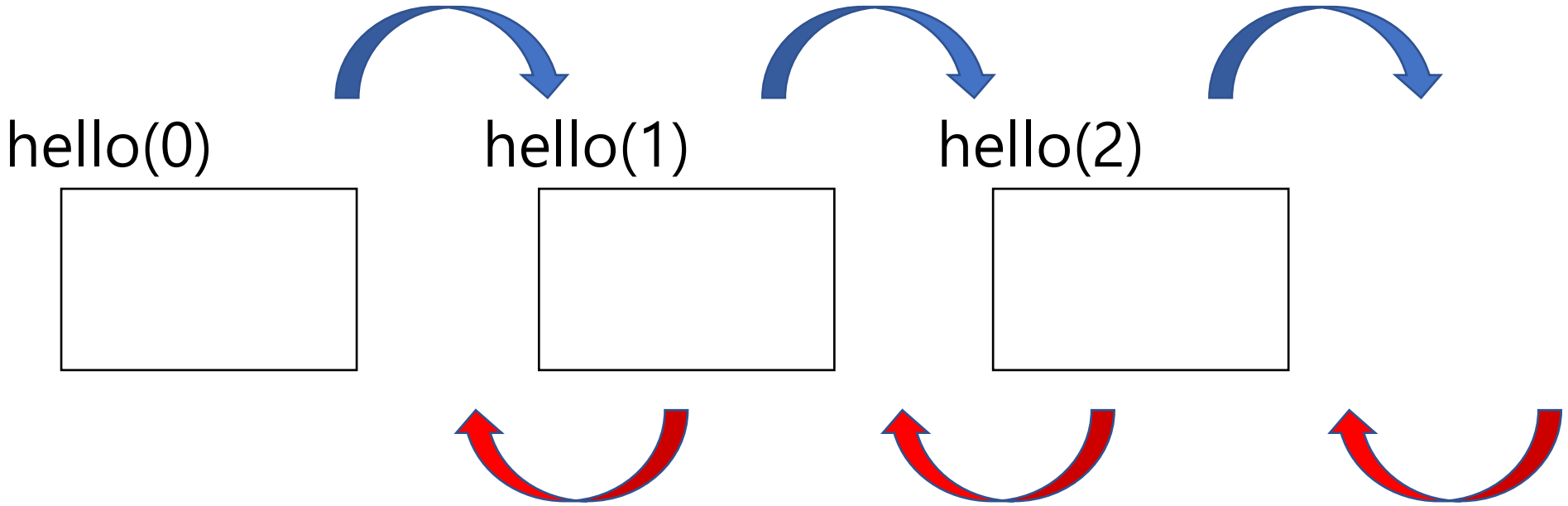
함께가요 미래로!
Enabling People

level 과 branch

지금까지 학습한 것

재귀 함수가 다음 재귀 함수를 호출할 때

하나의 함수만 호출하였다.

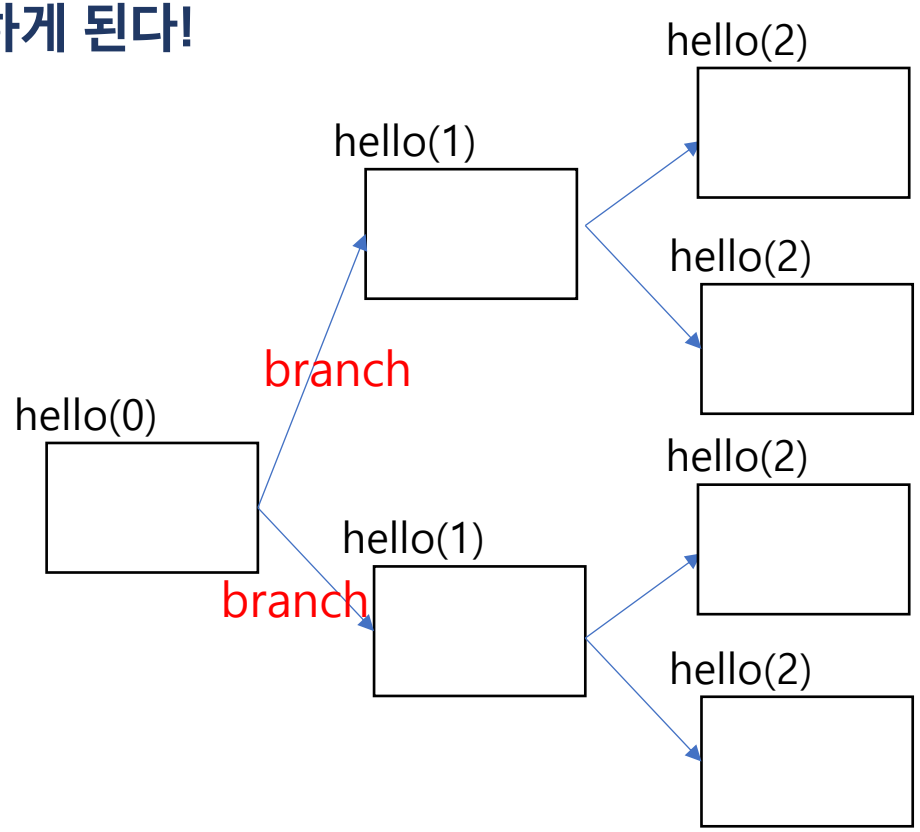


한번에 여러 호출을 진행한다면?!

hello(0) 에서 hello(1) 을 두 번 호출한다면?!

hello(1) 에서 hello(2) 를 몇 번 호출할까?!

- 재귀 함수이기 때문에 당연하게도 두 번 호출하게 된다!
- 이러한 분기를 branch 라 한다.
- 오른쪽 그림은 branch 개수 2인 상황



다음 코드의 실행 결과는?!

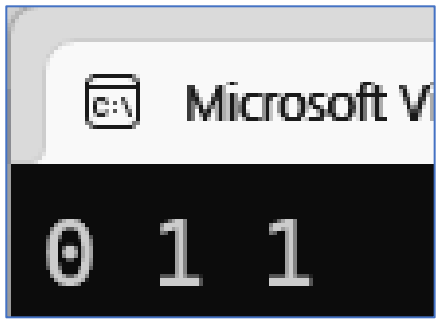
- func() 2회 호출
- 함수를 실행하면서 level 출력 처리

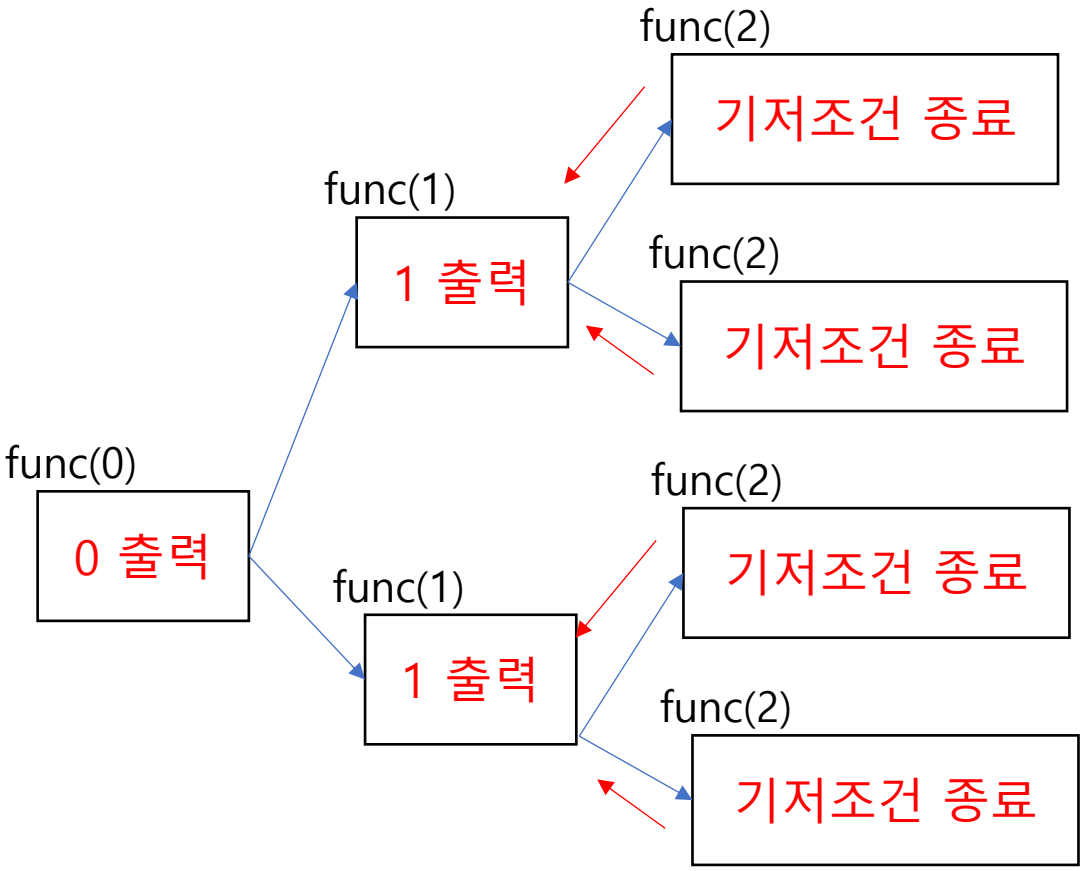
```
void func(int level) {
    if (level >= 2) {
        return;
    }
    cout << level << " ";
    func(level + 1);
    func(level + 1);
}

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

QUIZ 5 정답

정답 : 0 1 1

- **디버깅과 호출 스택 창을 확인하며 다시 검토한다!**





https://gist.github.com/hoconoco/e4238f4028a2bf5fd8c1799c23cc0647

다음 코드의 실행 결과는?!

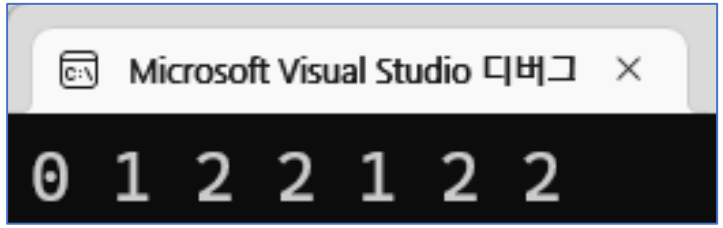
- level 3
- 반복문?!

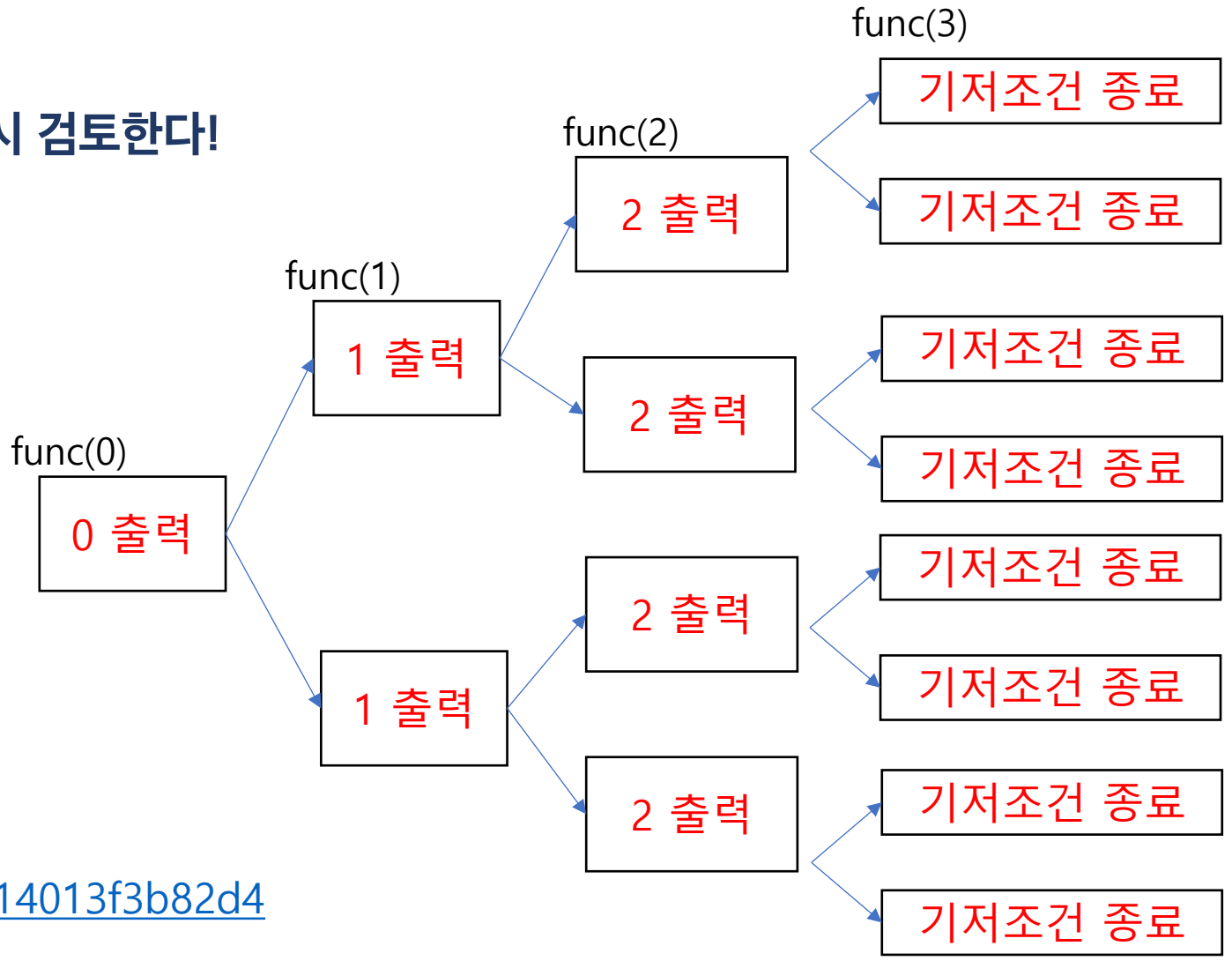
```
void func(int level) {
    if (level >= 3) {
        return;
    }
    cout << level << " ";
    for (int i = 0; i < 2; i++) {
        func(level + 1);
    }
    int dummy;
}
```

QUIZ 6 정답

정답 : 0 1 2 2 1 2 2

- **디버깅과 호출 스택 창을 확인하며 다시 검토한다!**





https://gist.github.com/hoconoco/ef6ca7914013f3b82d41ebf61e46f14c

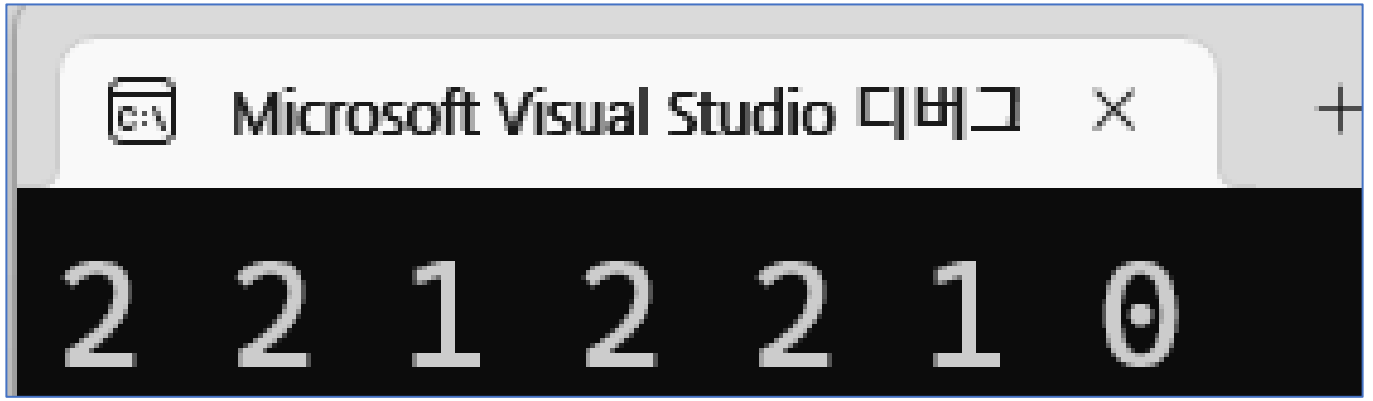
다음 코드의 실행 결과는?!

- 출력문이 아래에?!
- 종료 시점 이 후 돌아오면서 level 출력 처리

```
void func(int level) {
    if (level >= 3) {
        return;
    }
    for (int i = 0; i < 2; i++) {
        func(level + 1);
    }
    cout << level << " ";
    int dummy;
}
```

QUIZ 7 정답

재귀함수가 종료 시점 이 후 돌아오면서 출력된다



https://gist.github.com/hoconoco/99e91f000a6c30828d21b20120cf930a

다음 코드의 실행 결과는?!

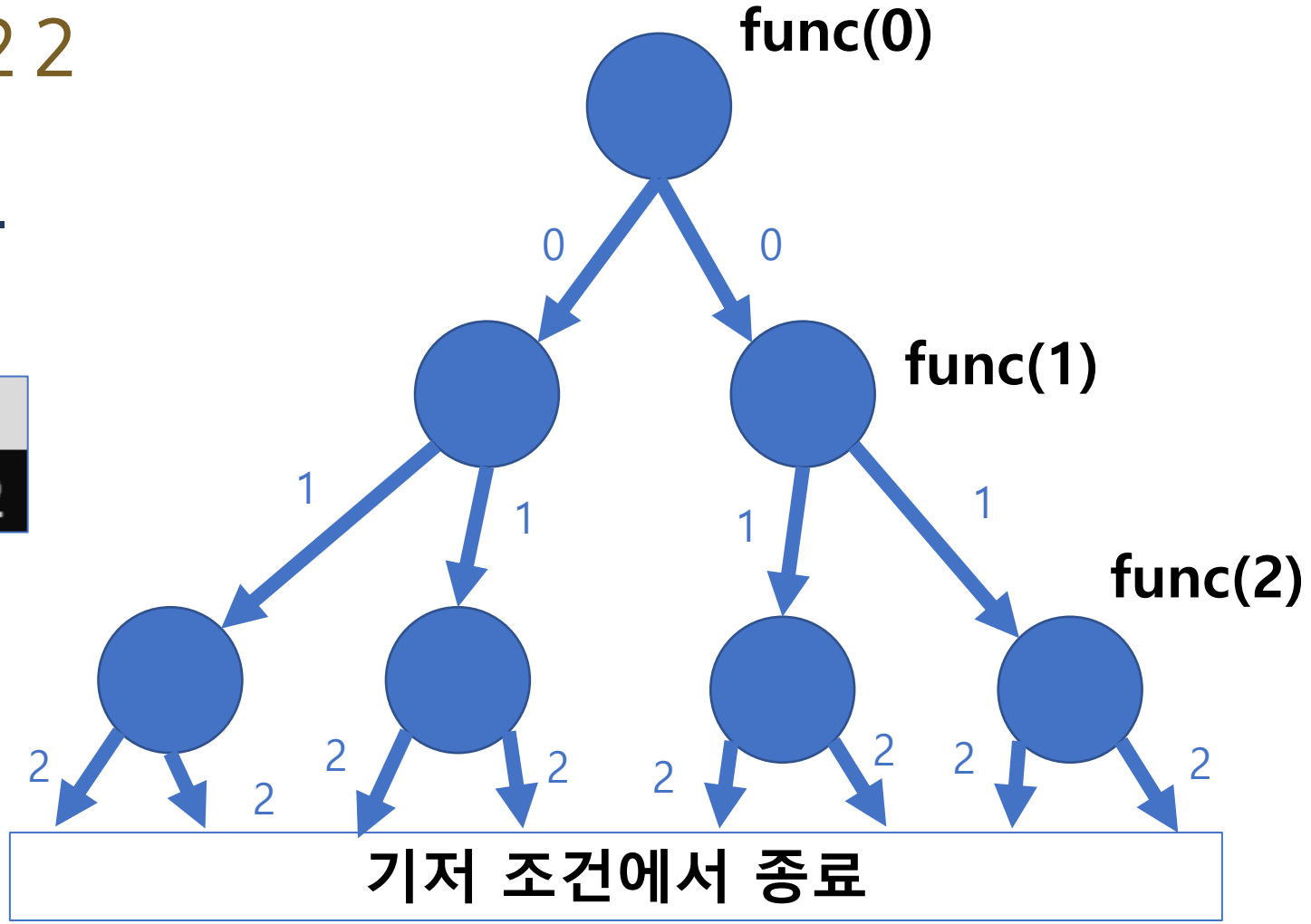
- 출력문이 반복문 안에?!

```
void func(int level) {
    if (level >= 3) {
        return;
    }
    for (int i = 0; i < 2; i++) {
        cout << level << " ";
        func(level + 1);
    }
    int dummy;
}
```

QUIZ 8 정답

정답 : 0 1 2 2 1 2 2 0 1 2 2 1 2 2

- 디버깅을 통해 호출 스택을 추적한다.
- 함수를 실행하면서 반복문에서 출력한다.



https://gist.github.com/hoconoco/9a8b691280a586d540cd1e57980f6cc5

다음 코드의 실행 결과는?!

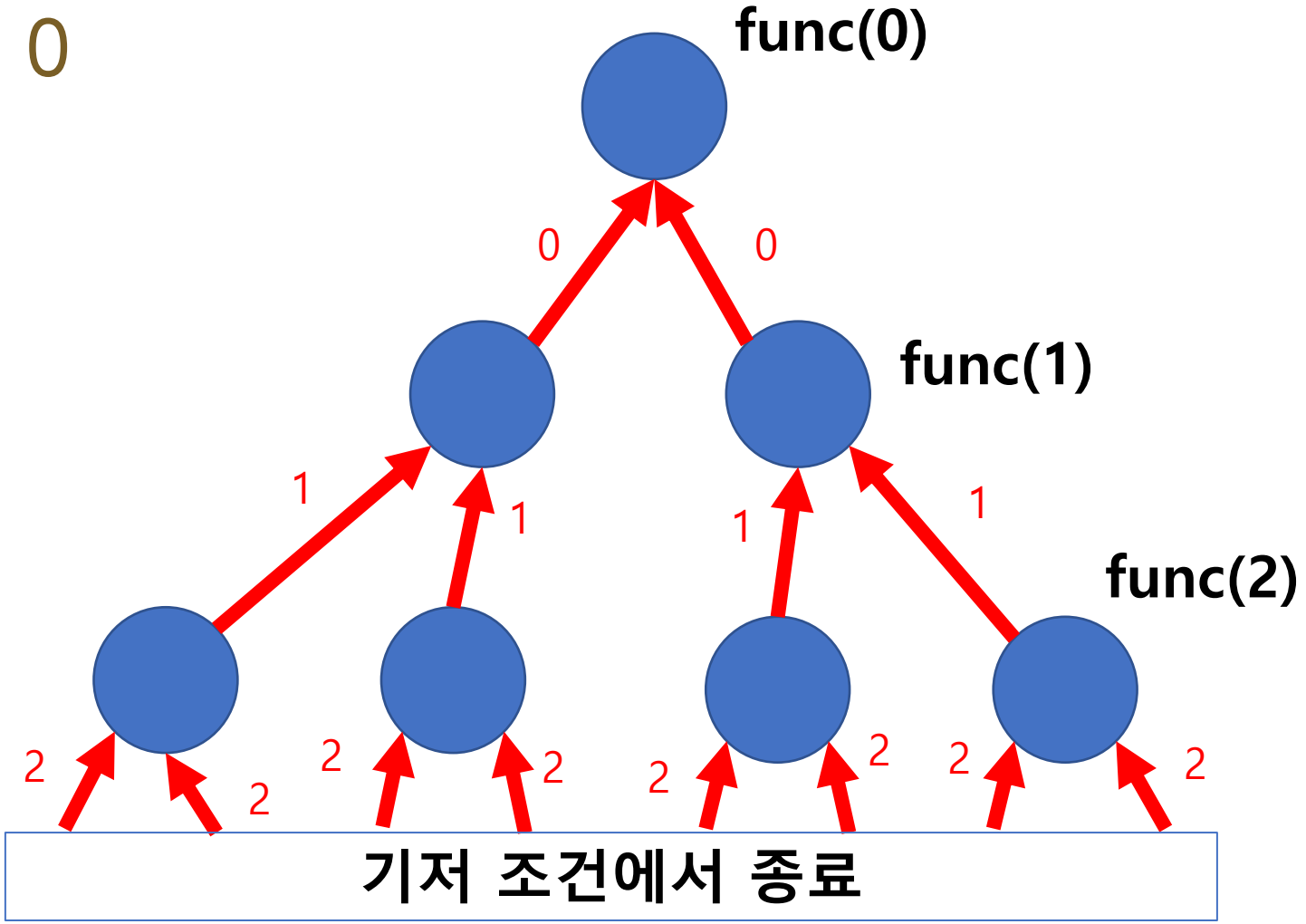
- 출력문이 반복문 안에?!

```
void func(int level) {
    if (level >= 3) {
        return;
    }
    for (int i = 0; i < 2; i++) {
        func(level + 1);
        cout << level << " ";
    }
    int dummy;
}
```

QUIZ 9 정답

정답 : 2 2 1 2 2 1 0 2 2 1 2 2 1 0

- 디버깅을 통해 호출 스택을 추적한다.
- 종료 지점에서 돌아오면서 출력한다.



https://gist.github.com/hoconoco/8d81302fb5cb84e3068fc68ac6c2de4b

다음 코드의 실행 결과는?!

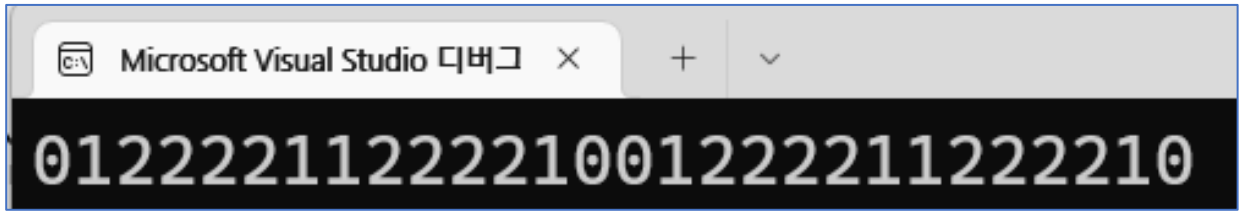
- 출력문이 반복문 안에?!

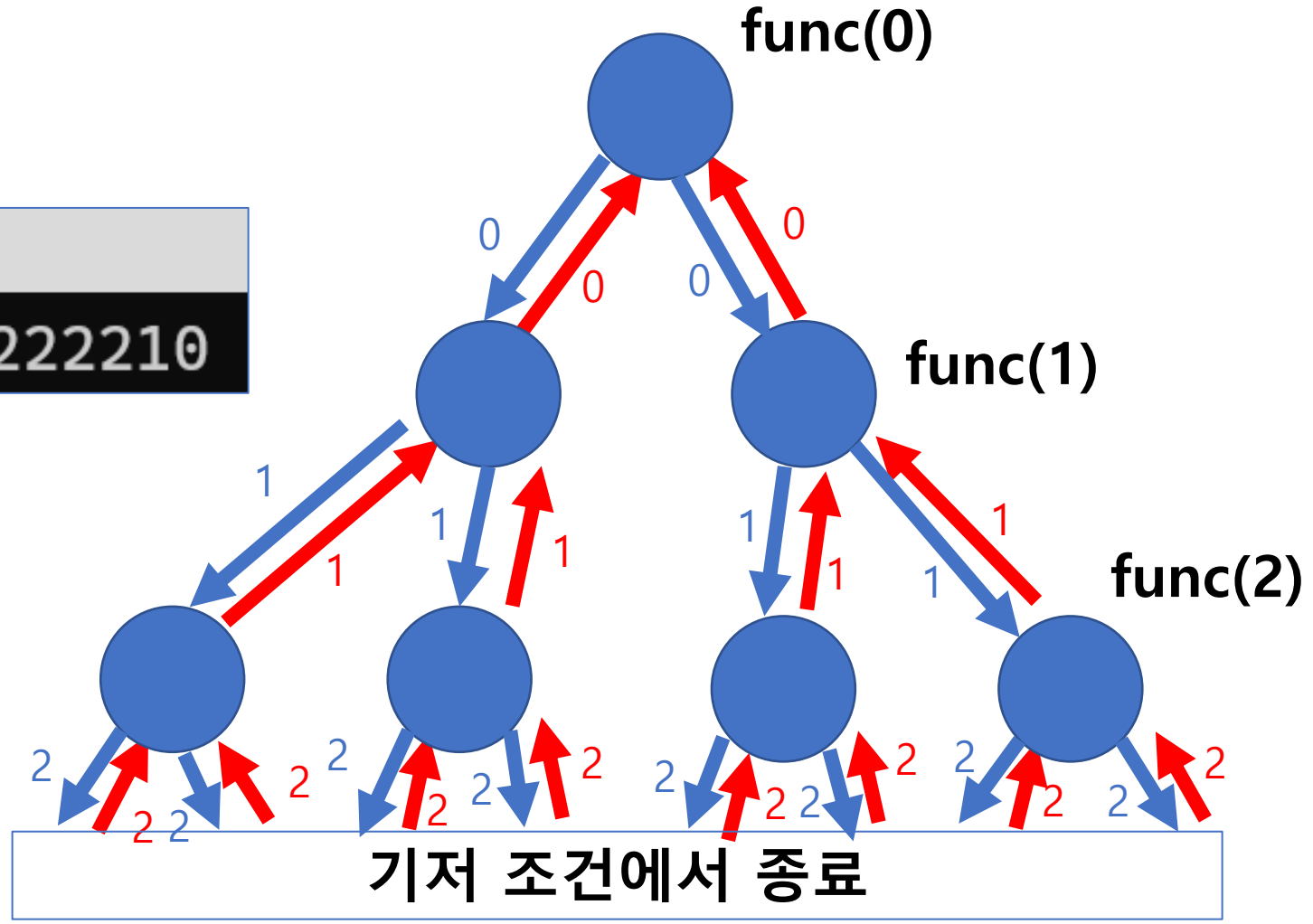
```
void func(int level) {
    if (level >= 3) {
        return;
    }
    for (int i = 0; i < 2; i++) {
        cout << level;
        func(level + 1);
        cout << level;
    }
    int dummy;
}
```

QUIZ 10 정답

정답 : 012222112222100122221122222 10

- 이제 그림 없이 이해가 된다면 마스터다.





https://gist.github.com/hoconoco/8d6d52a06ac3ca2e8cc67725f60b0c4f

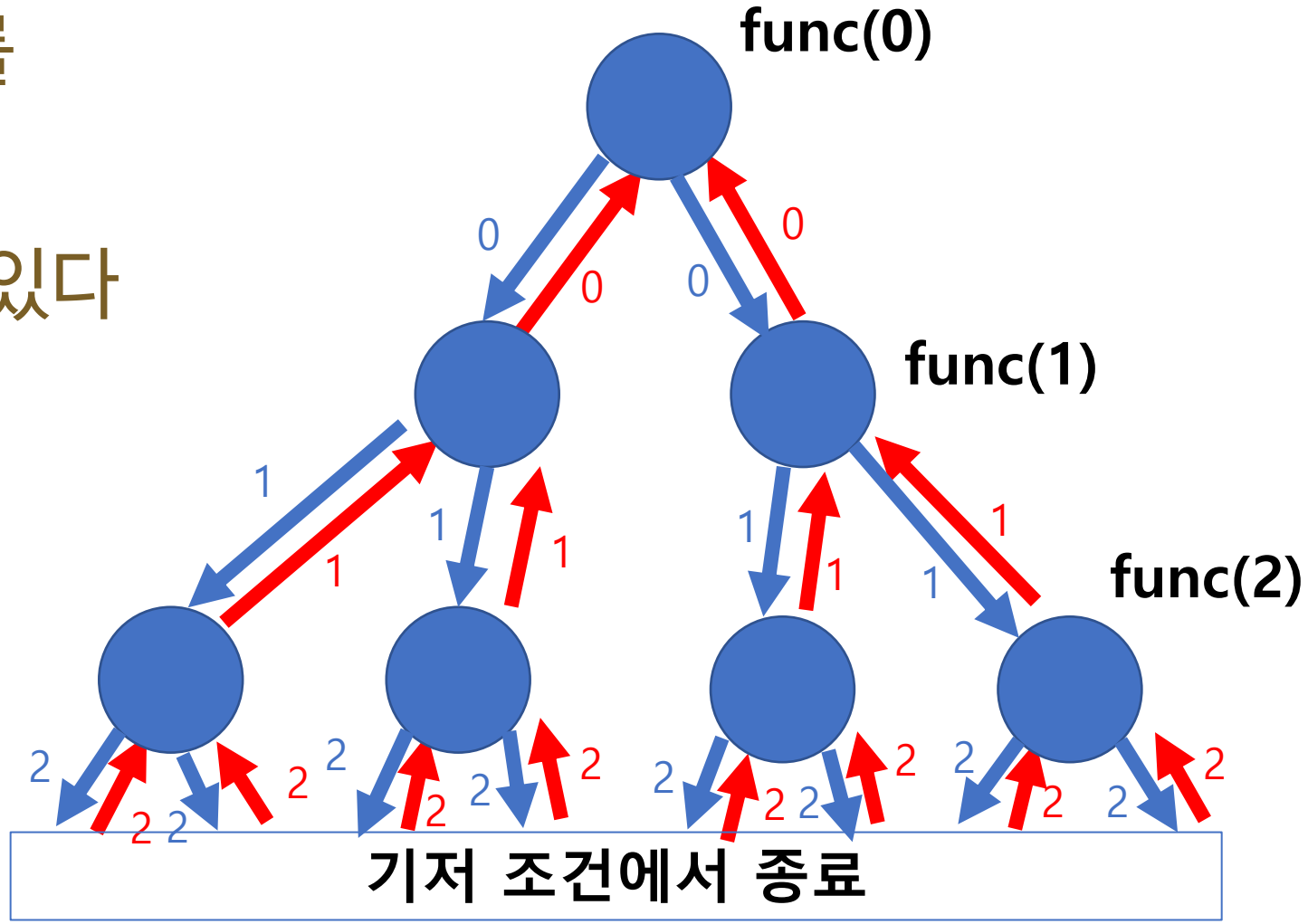
SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

재귀함수 + DAT

DAT와 재귀 함수

오른쪽 이미지처럼
재귀 함수의 호출 순서(경로)를
DAT에 기록하여
다양한 알고리즘에 사용할 수 있다

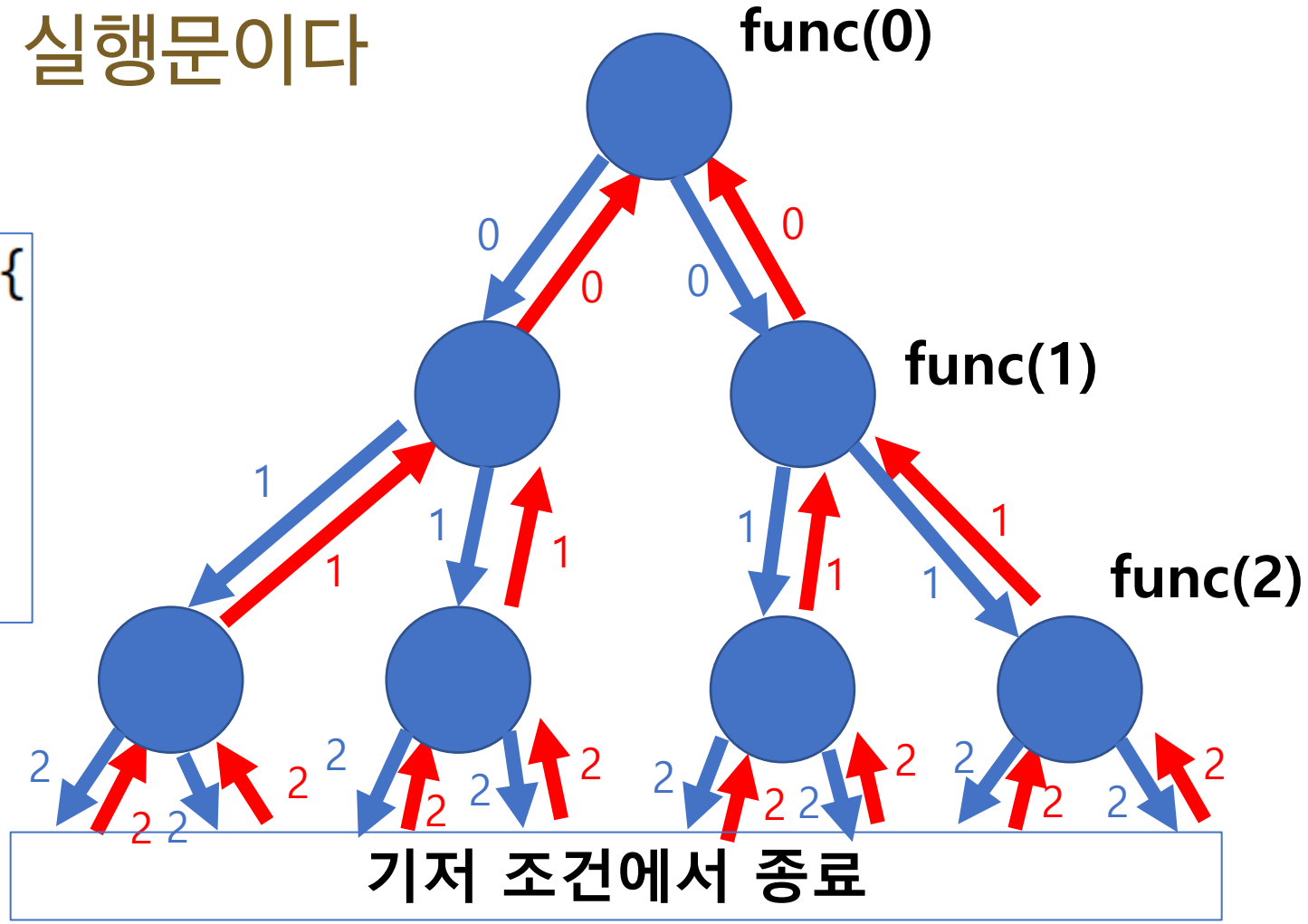


코드를 다시 한번 보자

함수 호출 위 아래에 있는 코드가

각각 진행 방향 및 복귀 방향의 실행문이다

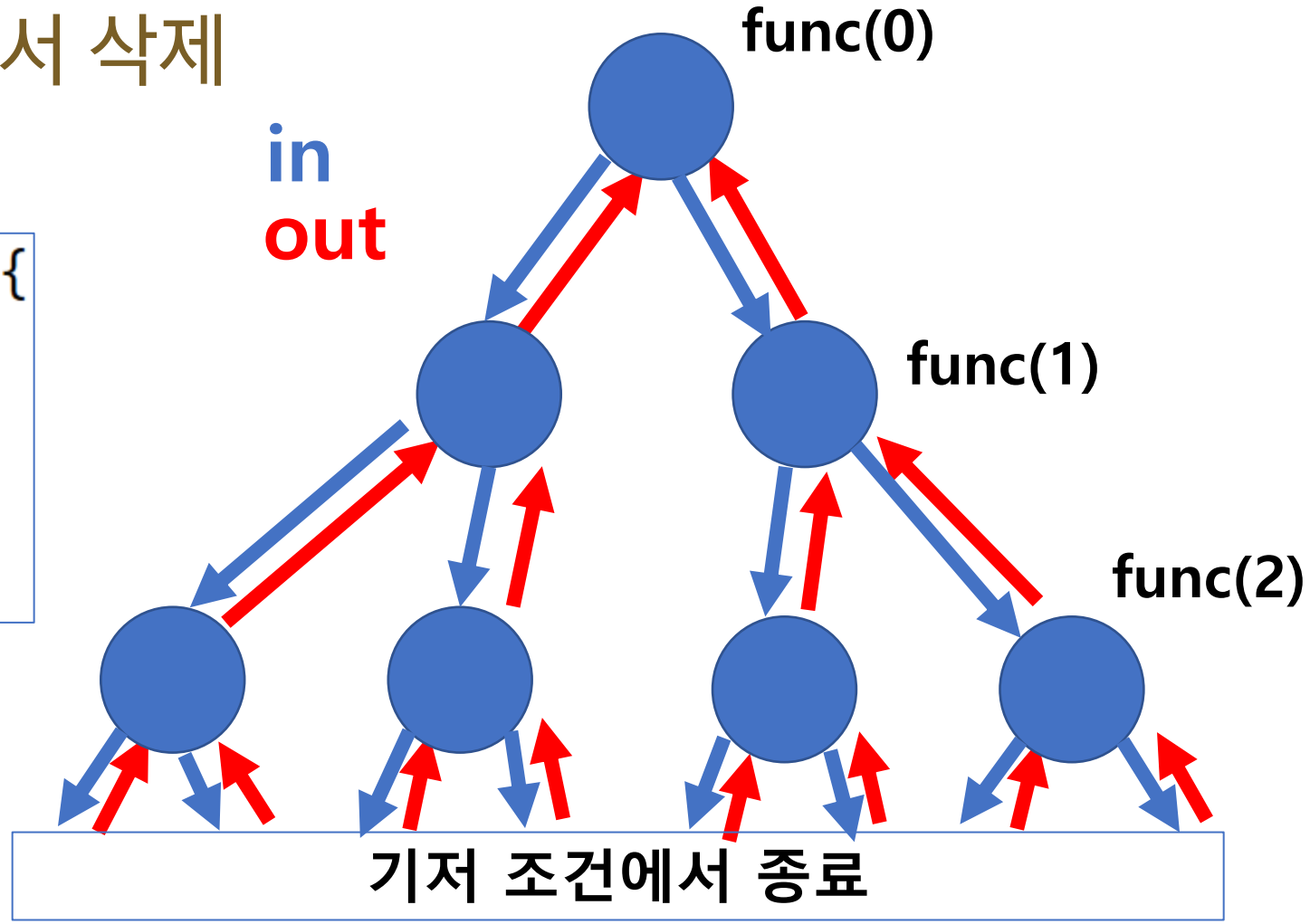
```
for (int i = 0; i < 2; i++) {
    cout << level;
    func(level + 1);
    cout << level;
}
```



위쪽 코드가 in! → DAT 기록

아래쪽 코드가 out! → DAT 에서 삭제

```
for (int i = 0; i < 2; i++) {
    cout << level;      in
    func(level + 1);
    cout << level;      out
}
```



코드를 실행한 결과를 이해해보자.

- 디버깅을 통해 DAT 배열의 변화를 추적한다.

```
000
001
010
011
100
101
110
111
```

```
int DAT[5];
void func(int level) {
    if (level >= 3) {
        for (int i = 0; i < 3; i++) {
            cout << DAT[i];
        }
        cout << '\n';
        return;
    }
    for (int i = 0; i < 2; i++) {
        //진입, DAT 기록
        DAT[level] = i;
        func(level + 1);
        //아웃, DAT 기록 삭제
        DAT[level] = 0;
    }
    int dummy;
}
```

https://gist.github.com/hoconoco/62c83cb47f66fa1580a38e6d123ae891

굉장한 결과가 나왔다.

DAT를 통해
재귀함수의 경로를 기록했더니
0,1 을 3개 선택할 수 있는 모든 경우의 수를 출력하게 되었다.

```
000
001
010
011
100
101
110
111
```

0,1 두 개의 숫자를 3번 뽑았을 경우를 보여준다.

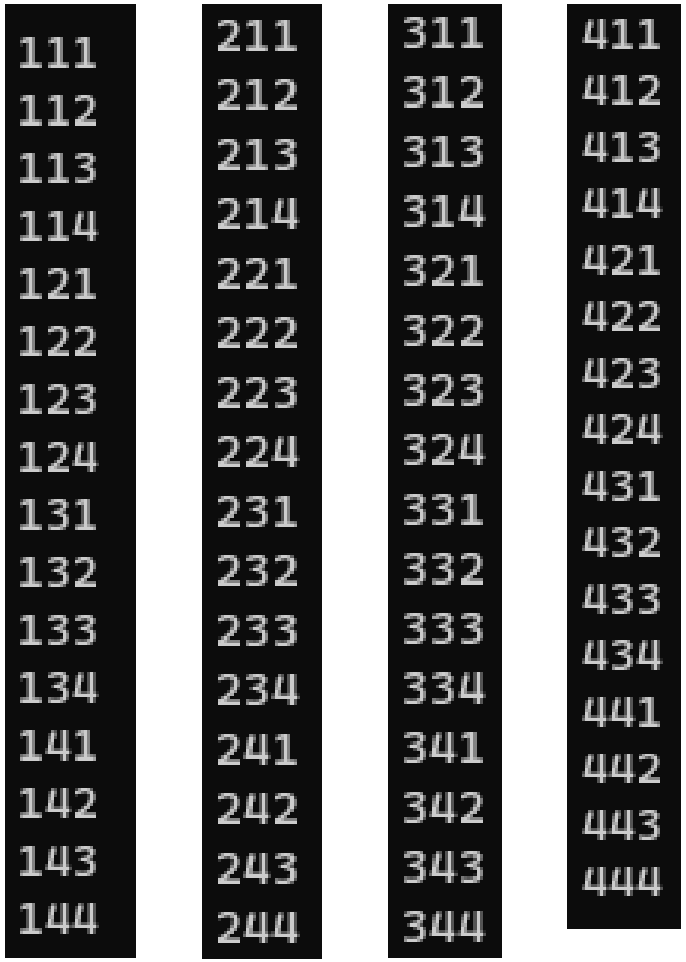
- 코드의 어느 부분이 결과에 영향을 주었을 까?

```
    if (level >= 3) {
        for (int i = 0; i < 3; i++) {
            cout << DAT[i];
        }
        cout << '\n';
        return;
    }
    for (int i = 0; i < 2; i++) {
```

```
000
001
010
011
100
101
110
111
```

1~4 까지 4개의 숫자 중 3개를 뽑는 경우의 수가 출력하기

- 111 112 113 114 121 122 123 124 ...
- 211 212 213 214 221 222 223 224 ...
- 311 312 313 314 321 322 323 324 ...
- 411 412 413 414 421 422 423 424 ...



QUIZ 11 정답

1~4 까지 4개의 숫자 중 3개를 뽑는 경우의 수

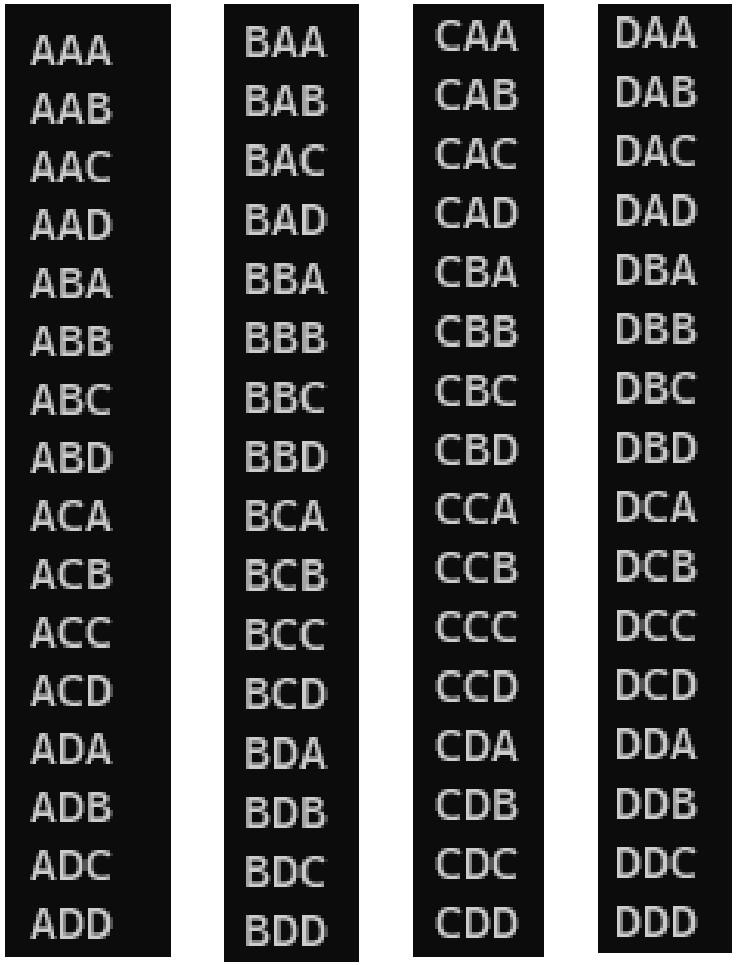
- **코드를 이해하고 직접 짤 수 있어야 한다.**

```
void func(int level) {
    if (level >= 3) {
        for (int i = 0; i < 3; i++) {
            cout << DAT[i];
        }
        cout << '\n';
        return;
    }
    for (int i = 1; i <= 4; i++) {
        //진입, DAT 기록
        DAT[level] = i;
        func(level + 1);
        //아웃, DAT 기록 삭제
        DAT[level] = 0;
    }
    int dummy;
}
```

https://gist.github.com/hoconoco/2080e04b8e236ff0783bbdf029879a80

ABCD 중 3개를 뽑는 경우의 수 출력

- AAA AAB AAC AAD ABA ABB ABC ABD …
- BAA BAB BAC BAD BBA BBB BBC BBD …
- CAA CAB CAC CAD CBA CBB CBC CBD …
- DAA DAB DAC DAD DBA DBB DBC DBD …



QUIZ 12 정답

ABCD 중 3개를 뽑는 경우의 수

- **코드를 이해하고 직접 짤 수 있어야 한다.**

```
char DAT[5];
void func(int level) {
    if (level >= 3) {
        for (int i = 0; i < 3; i++) {
            cout << DAT[i];
        }
        cout << '\n';
        return;
    }
    for (int i = 'A'; i <= 'D'; i++) {
        //진입, DAT 기록
        DAT[level] = i;
        func(level + 1);
        //아웃, DAT 기록 삭제
        DAT[level] = 0;
    }
    int dummy;
}
```

https://gist.github.com/hoconoco/1bd16e5a4a5fa0399e5e0236a35b6e21

QWER 중 2개를 뽑는 경우의 수 출력

```
QQ
QW
QE
QR
WQ
WW
WE
WR
EQ
EW
EE
ER
RQ
RW
RE
RR
```

QUIZ 13 정답

QWER 중 2개를 뽑는 경우의 수

- 코드를 이해하고 직접 짤 수 있어야 한다.
- 우리는 이제 재귀함수 마스터다

https://gist.github.com/hoconoco/add95a748483ccb5eb2532f518a279be

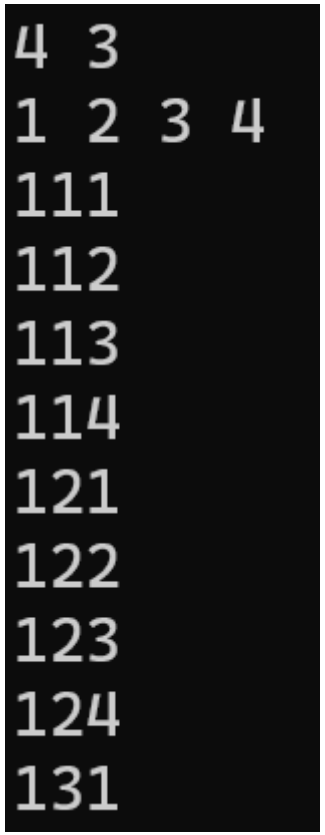
```
char DAT[5];
char str[5] = "QWER";
void func(int level) {
    if (level >= 2) {
        for (int i = 0; i < 2; i++) {
            cout << DAT[i];
        }
        cout << '\n';
        return;
    }
    for (int i = 0; i < 4; i++) {
        //진입, DAT 기록
        DAT[level] = str[i];
        func(level + 1);
        //아웃, DAT 기록 삭제
        DAT[level] = 0;
    }
    int dummy;
}
```

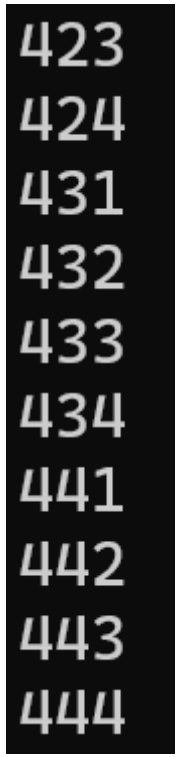
순열과 조합

N,K를 입력 받고

N개의 정수를 입력 받아서 K개를 뽑는 경우의 수 출력

- **입력 예시**
  - 4 3
  - 1 2 3 4
- **출력 예시**
  - 111
  - 112
  - 113
  - 121
  - …
  - 444





QUIZ 14 정답

지금까지 우리가 학습한 내용의 모든 것

```
void func(int now) {
    if (now == k) {
        for (int i = 0; i < k; i++) {
            cout << DAT[i];
        }
        cout << '\n';
        return;
    }

    for (int i = 0; i < n; i++) {
        DAT[now] = arr[i];
        func(now + 1);
        DAT[now] = 0;
    }
}
```

https://gist.github.com/hoconoco/09fba3ab76676093d6d2b62484b79149

1, 2, 3, 4 를 나열하는 방법은 어떤 게 있을까?

- **순서를 고려할 지?**
- **중복을 고려할 지?**

[참고] 순열과 조합

순열 : 중복을 허용하지 않고 순서대로 숫자를 나열한다.

- 순서가 다르면 다른 순열이다. ex) 1 2 3 / 3 2 1 (서로 다른 순열 O)
- 중복된 숫자를 뽑지 않는다. ex) 1 1 2 (허용 x)

조합 : 중복을 허용하지 않고 선택할 수 있는 숫자의 조합

- 순서를 고려하지 않고 숫자를 선택한 경우이다. ex) 1 2 3 / 3 2 1 (같은 조합 O, 한 가지 경우만 존재)
- 중복된 숫자를 뽑지 않는다. ex) 1 1 2 (허용 x)

중복 순열 : 중복을 허용하며 순서대로 숫자를 나열한다.

- 순서가 다르면 다른 순열이다. ex) 1 2 3 / 3 2 1 (서로 다른 순열 O)
- 중복된 숫자를 허용한다. ex) 1 1 2 (허용 O)

중복 조합 : 중복을 허용하며 선택할 수 있는 숫자의 조합

- 순서를 고려하지 않고 숫자를 선택한 경우이다. ex) 1 2 3 / 3 2 1 (같은 조합 O, 한 가지 경우만 존재)
- 중복된 숫자를 허용한다. ex) 1 1 2 (허용 O)

그동안 재귀에서 다룬 숫자 뽑기

우리가 그동안 했던 경우의 수들은 중복 순열이었다.

- **중복 순열 : 중복을 허용하며 순서대로 숫자를 나열한다.**

1 2 3 4 중 3개의 숫자를 뽑는 모든 경우의 수 = 중복 순열

- **중복 O (1 1 1)**
- **순열 (순서대로 나열, 1 1 2 , 1 2 1 모두 뽑음)**

- **총 64개이다.**
- **DAT 대신 path 사용**
- **visited[] 로 경로 체크**

```
for (int i = 0; i < n; i++) {
    path[level] = number[i];
    func(level + 1);
    path[level] = 0;
}
```

https://gist.github.com/hoconoco/a4ba86a5390beaa90aadeae2d3cac4e6

N,K를 입력 받고

N개의 정수를 입력 받아서 K개를 뽑는 순열 출력

개수도 출력한다

- **입력 예시**
  - 4 3
  - 1 2 3 4

- **출력 예시**
  - 1 2 3
  - …
  - 4 3 2
  - 24 개

```
4 3
1 2 3 4
1 2 3
1 2 4
1 3 2
1 3 4
1 4 2
1 4 3
2 1 3
2 1 4
2 3 1
2 3 4
2 4 1
2 4 3
3 1 2
3 1 4
3 2 1
3 2 4
3 4 1
3 4 2
4 1 2
4 1 3
4 2 1
4 2 3
4 3 1
4 3 2
24
```

QUIZ 15 정답

순열은 중복 허용 X

순서 다르면 허용 O

- **코드를 이해하고 직접 짤 수 있어야 한다.**

```
for (int i = 0; i < n; i++) {
    if (visited[i] == 0) {
        visited[i] = 1;
        path[level] = number[i];
        func(level + 1);
        path[level] = 0;
        visited[i] = 0;
    }
}
```

https://gist.github.com/hoconoco/0e71602231b114f0cc34b3af77ce8bf4

N,K를 입력 받고
N개의 정수를 입력 받아서 K개를 뽑는 중복 조합 출력

- **입력 예시**
  - 4 3
  - 1 2 3 4

- **출력 예시**
  - 1 1 1
  - 1 1 2
  - 1 1 3
  - 1 1 4
  - 1 2 2
  - …
  - 4 4 4
  - 20 개

```
4 3
1 2 3 4
1 1 1
1 1 2
1 1 3
1 1 4
1 2 2
1 2 3
1 2 4
1 3 3
1 3 4
1 4 4
2 2 2
2 2 3
2 2 4
2 3 3
2 3 4
2 4 4
3 3 3
3 3 4
3 4 4
4 4 4
20
```

QUIZ 16 정답

중복 조합은 순서 다르면 허용 X

- 이전 값보다 작은 선택을 할 수 없게 한다.
- 코드를 이해하고 직접 짤 수 있어야 한다.

```
for (int i = 0; i < n; i++) {
    //이전 값보다 작은 선택은 하지 않는다.
    if (level > 0 && path[level - 1] > number[i]) continue;
    path[level] = number[i];
    func(level + 1);
    path[level] = 0;
}
```

https://gist.github.com/hoconoco/2e849ec47832e208ffdd52b9e77abea4

N,K를 입력 받고

N개의 정수를 입력 받아서 K개를 뽑는 조합 출력

개수도 출력한다

- **입력 예시**
  - 4 3
  - 1 2 3 4

- **출력 예시**
  - 1 2 3
  - 1 2 4
  - 1 3 4
  - 2 3 4
  - 4개

```
4 3
1 2 3 4
1 2 3
1 2 4
1 3 4
2 3 4
4
```

조합은 중복 허용 X

순서 다르면 허용 X

- **코드를 이해하고 직접 짤 수 있어야 한다.**

```
for (int i = 0; i < n; i++) {
    //중복 허용 X
    if (visited[i] == 0) {
        //이전 값보다 작은 선택은 하지 않는다.
        if (level > 0 && path[level - 1] > number[i]) continue;
        visited[i] = 1;
        path[level] = number[i];
        func(level + 1);
        path[level] = 0;
        visited[i] = 0;
    }
}
```

https://gist.github.com/hoconoco/08ee36d86f40d1b36c5a249de70e4a03

SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

내일 방송에서 만나요!

삼성 청년 SW 아카데미