



C언어 강의자료

문정욱

A decorative graphic consisting of several light blue squares of varying sizes arranged in a stepped pattern on the left side of the slide.

C언어 맛 보기 2

if 문의 개념

무조건 문장 수행

```
#include <stdio.h>
```

```
int main(void)
{
```

```
    int a;
```

```
    scanf("%d", &a);
```

```
    printf("a == one\n");
```

```
    return 0;
```

```
}
```

Logical Error

입출력 결과

```
1
a == one
계속하려면
```

입출력 결과

```
2
a == one
계속하려면 아무 키나 누르십시오 . . .
```

조건 문장 수행

```
#include <stdio.h>
```

```
int main(void)
{
```

```
    int a;
```

```
    scanf("%d", &a);
```

```
    if(a == 1)
```

```
        printf("a == one\n");
```

```
    return 0;
```

```
}
```

Ok

입출력 결과

```
1
a == one
계속하려면
```

입출력 결과

```
2
계속하려면 아무 키나 누르십시오 . . .
```

if 문의 개념

- if 문(if-statement)
 - 조건 수식을 만족하면 문장을 한 번 수행한다.

If-statement:

**if (expression)
statement**

```
#include <stdio.h>
```

```
int main(void)  
{
```

```
    int a;
```

```
    scanf("%d", &a);
```

a statement
(if-statement)

```
    if(a == 1)
```

```
        printf("a == one\n");
```

a statement

```
    return 0;
```

```
}
```

If 문과 블록

- if 문(if-statement)
 - if는 한 개의 문장에만 관여한다.
그 뒤에 따라오는 두 번째 문장에는 관여하지 않는다.

입출력 결과

```
1
a == one
a == one
계속하려면 아무 키나 누르십시오 . . .
```

입출력 결과

```
2
a == one
계속하려면 아무 키나 누르십시오 . . .
```


```
#include <stdio.h>

int main(void)
{
    int a;

    scanf("%d", &a);

    if(a == 1)
        printf("a == one\n");
        printf("a == one\n");

    return 0;
}
```

 **always executed!!!**

If 문과 블록

```
#include <stdio.h>


int main(void)
{
    int a;

    scanf("%d", &a);

    if(a == 1)
        printf("a == one\n");

    printf("after if-stat\n");

    return 0;
}
```

 **always executed**

입출력 결과

```
1
a == one
after if-stat
계속하려면 아무 키나 누르십시오 . . .
```

입출력 결과

```
2
after if-stat
계속하려면 아무 키나 누르십시오 . . .
```

If 문과 블록

```
#include <stdio.h>

int main(void)
{
    int a;

    scanf("%d", &a);

    if(a == 1)
        printf("a == one\n");
        printf("a == one\n");

    printf("after if-stat\n");

    return 0;
}
```

always executed

입출력 결과

```
1
a == one
a == one
after if-stat
계속하려면 아무 키나 누르십시오 . . .
```

입출력 결과

```
2
a == one
after if-stat
계속하려면 아무 키나 누르십시오 . . .
```

Logical Error

If 문과 블록

```
#include <stdio.h>

int main(void)
{
    int a;


    scanf("%d", &a);

    if(a == 1) {
        printf("a == one\n");
        printf("a == one\n");
    }

    printf("after if-stat\n");

    return 0;
}
```

a block == a statement

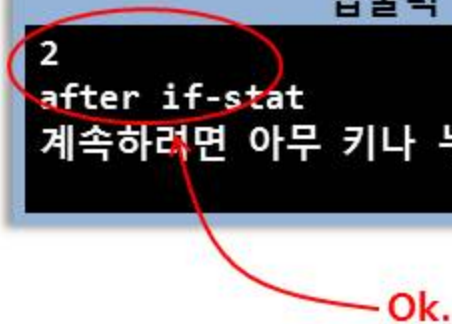


입출력 결과

```
1
a == one
a == one
after if-stat
계속하려면 아무 키나 누르십시오 . . .
```

입출력 결과

```
2
after if-stat
계속하려면 아무 키나 누르십시오 . . .
```



Ok.

연산자

■ 관계(relational) 연산자

Relational Operation	Operator
Equal	==
Not Equal	!=
Less than	<
Equal or Less than	<=
Greater than	>
Equal or Greater than	>=

입출력 결과

```
a+b==c
a!=b+c
a<b
a<=b
c>b
c>=b
계속하려면 아무 키나 누르십시오 . . .
```

```
#include <stdio.h>

int main(void)
{
    int a=1,b=2,c=3;

    if(a+b == c) printf("a+b==c\n");
    if(a != b+c) printf("a!=b+c\n");
    if(a < b) printf("a<b\n");
    if(a <= b) printf("a<=b\n");
    if(c > b) printf("c>b\n");
    if(c >= b) printf("c>=b\n");

    return 0;
}
```

연산자

■ 논리(logical) 연산자

Logical Operation	Operator
Logical And	&&
Logical Or	
Logical Not	!

입출력 결과

```
a<b && c>b
a<b || c<b
!(b>c)
계속하려면 아무 키나 누르십시오 . . .
```

입출력 결과

```
a<b && c>b
a<b || c<b
계속하려면 아무 키나 누르십시오 . . .
```

(Notice)

! 연산자는 뒤에는 괄호를 함께 써주는 것이 논리오류를 막는데 도움이 된다.

```
#include <stdio.h>

int main(void)
{
    int a=1,b=2,c=3;

    if(a<b && c>b) printf("a<b && c>b\n");
    if(a<b || c<b) printf("a<b || c<b\n");
    if( !(b>c) ) printf("!(b>c)\n");

    return 0;
}
```

```
#include <stdio.h>

int main(void)
{
    int a=1,b=2,c=3;

    if(a<b && c>b) printf("a<b && c>b\n");
    if(a<b || c<b) printf("a<b || c<b\n");
    if( ! b>c ) printf("!(b>c)\n");

    return 0;
}
```

수행 안함.
Logical Error

if-else 문의 개념

■ if-else 문(if-else-statement)

- 조건 수식을 만족하면 문장_1을 한 번 수행하고 만족하지 않으면 문장_2를 수행한다.

If-else statement:

```
if ( expression )  
    statement_1  
else  
    statement_2
```

입출력 결과

```
1  
a == one  
계속하려면
```

입출력 결과

```
2  
a == other  
계속하려면 아무 키나 누르십시오 . . .
```

```
#include <stdio.h>  
  
int main(void)  
{  
    int a;  
  
    scanf("%d", &a);  
  
    if(a == 1)  
        printf("a == one\n");  
    else  
        printf("a == other\n");  
  
    return 0;  
}
```

if-else 문과 블록

복합문장을 블록으로 묶음

```
#include <stdio.h>

int main(void)
{
    int a;

    scanf("%d", &a);

    if(a == 1) {
        printf("a == one\n");
        printf("a == one\n");
    }
    else
        printf("a == other\n");

    printf("after if-stat\n");
    return 0;
}
```

a if-else-statement

블록을 사용하지 않음

```
#include <stdio.h>

int main(void)
{
    int a;

    scanf("%d", &a);

    if(a == 1)
        printf("a == one\n");
        printf("a == one\n");

    else
        printf("a == other\n");

    printf("after if-stat\n");
    return 0;
}
```

a if-statement

a statement

else statement ???


Syntax Error !!!

중첩 if문

```
#include <stdio.h>

int main(void)
{
    int a, b;

    scanf("%d", &a);
    scanf("%d", &b);
    if( a == 1 )
        if( b == 1 )
            printf("a == 1, b == 1\n");
        else
            printf("a == 1, b != 1\n");
    else
        printf("a != 1\n");
    return 0;
}
```

 a if-else-statement

입출력 결과

```
1
1
a == 1, b == 1
계속하려면 아무 키
```

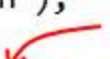
입출력 결과

```
1
2
a == 1, b != 1
계속하려면 아무 키나 누르십시오 . . .
```

```
#include <stdio.h>

int main(void)
{
    int a, b;

    scanf("%d", &a);
    scanf("%d", &b);
    if( a == 1 )
        printf("a == 1\n");
    else
        if( b == 1 )
            printf("a != 1, b == 1\n");
        else
            printf("a != 1, b != 1\n");
    return 0;
}
```

 a if-else-statement

입출력 결과

```
2
1
a != 1, b == 1
계속하려면 아무 키
```

입출력 결과

```
2
2
a != 1, b != 1
계속하려면 아무 키나 누르십시오 . . .
```


while 문의 개념

if에서 문장 수행 횟수

```
#include <stdio.h>

int main(void)
{
    int a;

    scanf("%d", &a);
    if(a < 5) {
        printf("%d < 5\n", a);
        a = a + 1;
    }
    return 0;
}
```

입출력 결과

```
2
2 < 5
계속하려면 아무
```

입출력 결과

```
5
계속하려면 아무 키나 누르십시오 . . .
```

while에서 문장 수행 횟수

```
#include <stdio.h>

int main(void)
{
    int a;

    scanf("%d", &a);
    while(a < 5) {
        printf("%d < 5\n", a);
        a = a + 1;
    }
    return 0;
}
```

입출력 결과

```
2
2 < 5
3 < 5
4 < 5
계속하려면 아무
```

입출력 결과

```
5
계속하려면 아무 키나 누르십시오 . . .
```

while 문의 개념

- While 문(while statement)
 - 조건 수식을 만족할 동안에 statement를 계속 수행한다.

While-statement:

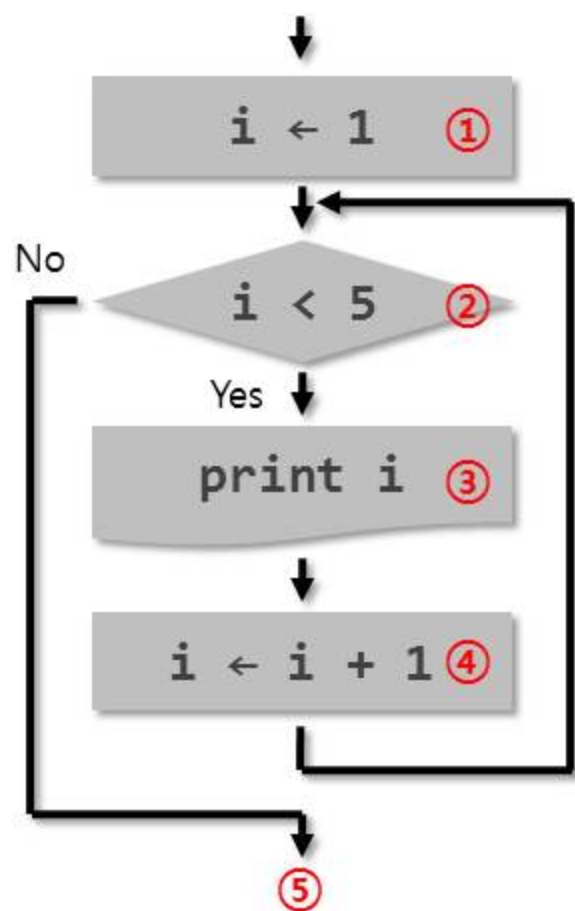
```
while( expression )  
    statement
```

입출력 결과

```
i = 1  
i = 2  
i = 3  
i = 4  
계속하려면 아무 키나 누르십시오 . . .
```

```
#include <stdio.h>  
  
int main(void)  
{  
    int i;  
  
    i=1;  
    while(i<5) {  
        printf("i = %d\n", i);  
        i=i+1;  
    }  
    return 0;  
}
```

while 문의 동작 순서



```
#include <stdio.h>
```

```
int main(void)
{
```

```
    int i;
```

```
    ① i=1;    ②
```

```
    while(i<5) {
```

```
        ③ printf("i = %d\n", i);
```

```
        ④ i=i+1;
```

```
    }
```

```
    ⑤ return 0;
```

```
}
```

```
i==1: ① → ② → ③ →
```

```
i==2: ④ → ② → ③ →
```

```
i==3: ④ → ② → ③ →
```

```
i==4: ④ → ② → ③ →
```

```
i==5: ④ → ② → ⑤ →
```


for 문의 개념

- for 문(for statement)
 - 조건 수식(expr_2)을 만족할 동안에 statement를 계속 수행한다.

For-statement:

```
for( expr_1; expr_2; expr_3 )  
    statement
```

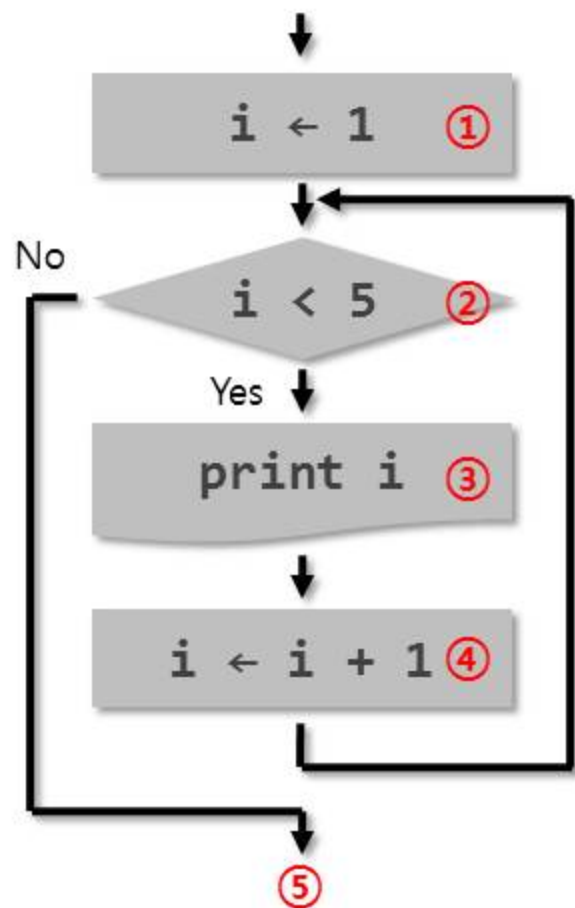
- expr1: 초기화 수식
- expr2: 조건 수식
- expr3: 증감 수식

입출력 결과

```
i = 1  
i = 2  
i = 3  
i = 4  
계속하려면 아무 키나 누르십시오 . . .
```

```
#include <stdio.h>  
  
int main(void)  
{  
    int i;  
  
    for(i=1; i<5; i=i+1) {  
        printf("i = %d\n", i);  
    }  
    return 0;  
}
```

for 문의 동작 순서



```
#include <stdio.h>
```

```
int main(void)
```

```
{
```

```
    int i;
```

```
    for(①i=1; ②i<5; ④i=i+1) {
```

```
        ③printf("i = %d\n", i);
```

```
    }
```

```
    ⑤return 0;
```

```
}
```

$i=1$:	①	→	②	→	③	→
$i=2$:	④	→	②	→	③	→
$i=3$:	④	→	②	→	③	→
$i=4$:	④	→	②	→	③	→
$i=5$:	④	→	②	→	⑤	→

배열의 개념

많은 변수의 선언

```
#include <stdio.h>
```

```
int main(void)
{
    int i;
    int a0, a1, a2;
```

```
    a0 = 0*0;
    a1 = 1*1;
    a2 = 2*2;
```

```
    printf("%d\n", a0);
    printf("%d\n", a1);
    printf("%d\n", a2);
    return 0;
```

```
}
```

변수를 100개 사용할 경우

```
#include <stdio.h>
```

```
int main(void)
{
    int i;
    int a0, a1, a2 ..., a99;
```

```
    a0 = 0*0;
    a1 = 1*1;
    a2 = 2*2;
```

```
    ...
```

```
    a99 = 99*99;
```

```
    printf("%d\n", a0);
    printf("%d\n", a1);
    printf("%d\n", a2);
```

```
    ...
```

```
    printf("%d\n", a99);
    return 0;
```

```
}
```



배열의 개념

반복문을 사용할 수 있다면

```
#include <stdio.h>
```

```
int main(void)  
{
```

```
    int i;  
    int a0, a1, a2 ..., a99;
```

```
    a0 = 0*0;  
    a1 = 1*1;  
    a2 = 2*2;  
    ...  
    a99 = 99*99;
```

```
    printf("%d\n", a0);  
    printf("%d\n", a1);  
    printf("%d\n", a2);  
    ...  
    printf("%d\n", a99);  
    return 0;
```

```
}
```

```
#include <stdio.h>
```

```
int main(void)  
{
```

```
    int i;  
    int a0~99;
```

```
    for(i=0; i<=99; i=i+1)  
        ai = i*i;
```

```
    for(i=0; i<=99; i=i+1)  
        printf("%d\n", ai);
```

```
    return 0;
```

```
}
```

error

배열의 개념

반복문을 사용할 수 있다면

```
#include <stdio.h>
```

```
int main(void)  
{
```

```
    int i;  
    int a[100];
```

```
    for(i=0; i<=99; i=i+1)  
        a[i] = i*i;
```

```
    for(i=0; i<=99; i=i+1)  
        printf("%d\n", a[i]);
```

```
    return 0;
```

```
}
```

```
#include <stdio.h>
```

```
int main(void)  
{
```

```
    int i;  
    int a[100]; // a[0] ~ a[99]
```

```
    for(i=0; i<=99; i=i+1)  
        a[i] = i*i;
```

```
    for(i=0; i<=99; i=i+1)  
        printf("%d\n", a[i]);
```

```
    return 0;
```

```
}
```

배열의 개념

- 배열(array)
 - 같은 타입의 연속된 변수들로 구성된 복합 변수
 - 길이는 정수(n) 값
 - 인덱스의 범위는 $0 \sim n-1$
 - 인덱스 범위를 벗어나면 실행 오류가 발생할 수 있다.

입출력 결과

```
2
9
8
계속하려면 아무 키나 누르십시오 . . .
```

```
#include <stdio.h>

int main(void)
{
    int i;
    int a[3]; // 배열의 길이

    a[0]=2;
    a[1]=9;
    a[2]=8;

    for(i=0; i<3; i=i+1) {
        printf("%d\n", a[i]);
    }
    return 0;
}
```


배열 메모리 구조

```
#include <stdio.h>

int main(void)
{
    int i;
    int a[3];

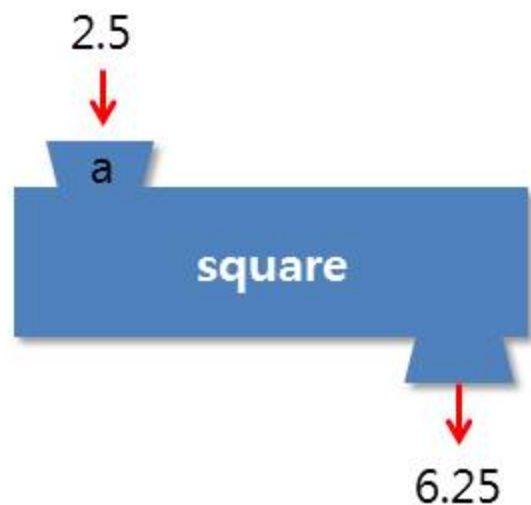
    a[0]=2;
    a[1]=9;
    a[2]=8;

    for(i=0; i<3; i=i+1) {
        printf("%d\n", a[i]);
    }
    return 0;
}
```



함수의 개념

- 함수(function)
 - 명령의 목록(instruction sequence)이 저장되어 있는 메모리 공간
 - 입력 값을 받으면 출력 값을 돌려주는 기능 수행



```
#include <stdio.h>

double square(double a)
{
    return a * a;
}

int main(void)
{
    double x = 2.5;
    double z;

    z = square( x );
    printf("%f\n", z );
    return 0;
}
```


함수의 개념

- 함수의 호출
 - 함수의 **실행 위치**
 - 함수가 실행될 때 필요한 **입력 값**
 - 함수 **반환 값의 활용 방법**
- 함수의 정의
 - 함수가 어떻게 동작하는지 그 방법이 기술되어 있다.

function:

```
ret_type  funct_name( form_argu )  
{  
    ...  
    return value;  
}
```

```
#include <stdio.h>  
  
double square(double a) } 함수 정의  
{  
    return a * a;  
}  
  
int main(void)  
{  
    double x = 2.5;  
    double z;  
  
    z = square( x ); } 함수 호출  
    printf("%f\n", z );  
    return 0;  
}
```

함수의 개념

■ 함수 호출

- 입력 방법: 인자(변수)에 값을 전달
- 반환 방법: 값을 반환

입출력 결과

6.250000

계속하려면 아무 키나 누르십시오 . . .

```
#include <stdio.h>

double square(double a)
{
    return a * a;
}

int main(void)
{
    double x = 2.5;
    double z;

    z = square( x );
    printf("%f\n", z );
    return 0;
}
```

함수의 호출 구조

여러 개의 인자 전달

```
#include <stdio.h>

int add(int a, int b)
{
    return a + b;
}

int main(void)
{
    int x = 2;
    int y = 3;
    int z;

    z = add( x , y );
    printf("%d\n", z );
    return 0;
}
```

여러 함수의 호출

```
#include <stdio.h>

double square(double a)
{
    return a * a;
}

double circle(double radius)
{
    return 3.14 * square(radius);
}

int main(void)
{
    printf("%f\n", circle( 2.5 ) );
    return 0;
}
```

함수의 호출 구조

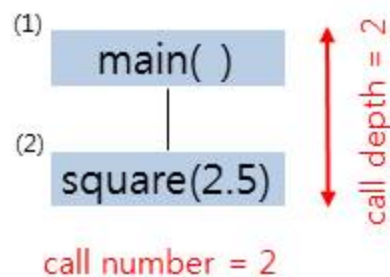
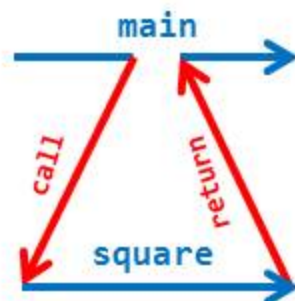
단일 호출

```
#include <stdio.h>

double square(double a)
{
    return a * a;
}

int main(void)
{
    double sq;

    sq = square( 2.5 );
    return 0;
}
```



- Call Depth: 함수의 호출 깊이.
- Call Number: 함수의 호출 횟수.

함수의 호출 구조

중첩 호출

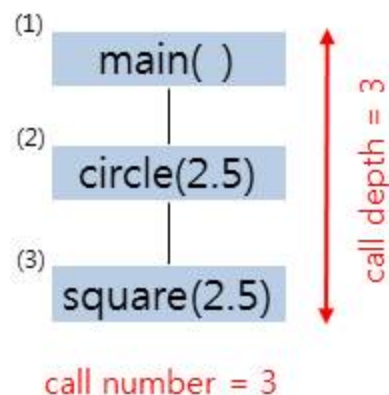
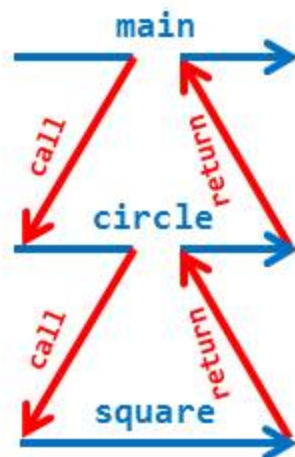
```
#include <stdio.h>

double square(double a)
{
    return a * a;
}

double circle(double r)
{
    return 3.14 * square(r);
}

int main(void)
{
    double area;

    area = circle( 2.5 );
    return 0;
}
```



함수의 호출 구조

연속 호출: 인자 밖 호출

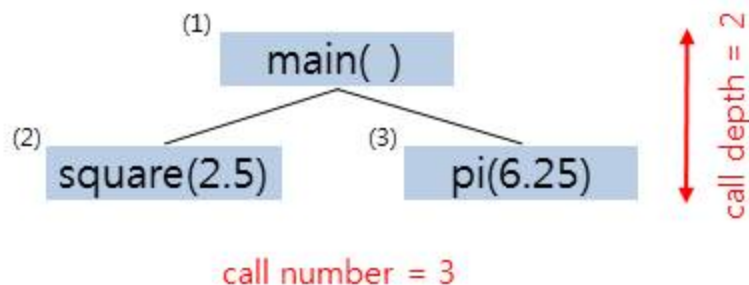
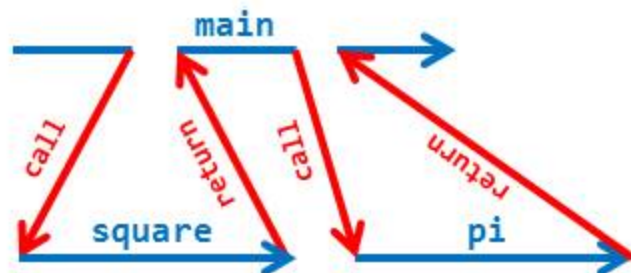
```
#include <stdio.h>

double square(double a)
{
    return a * a;
}

double pi(double b)
{
    return 3.14 * b;
}

int main(void)
{
    double sq;
    double area;

    sq = square( 2.5 );
    area = pi( sq );
    return 0;
}
```



함수의 호출 구조

연속 호출: 인자 안 호출

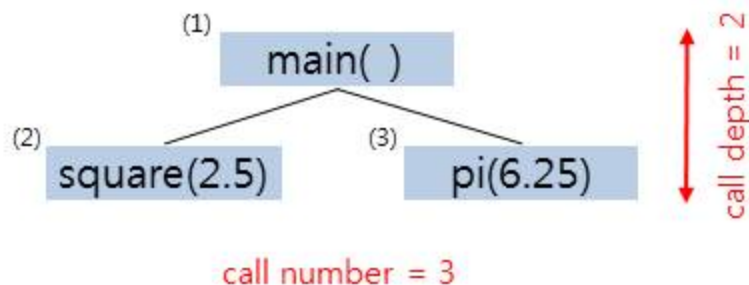
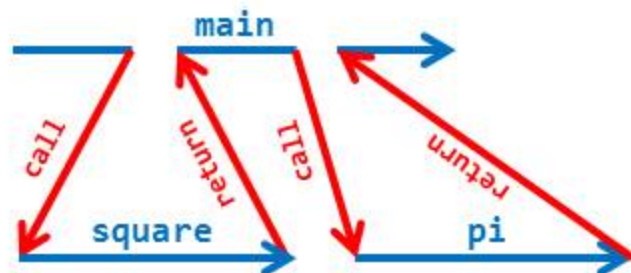
```
#include <stdio.h>

double square(double a)
{
    return a * a;
}

double pi(double b)
{
    return 3.14 * b;
}

int main(void)
{
    double area;

    area = pi( square( 2.5 ) );
    return 0;
}
```



함수의 호출 구조

연속 호출: 인자 밖 호출

```
#include <stdio.h>

double square(double a)
{
    return a * a;
}

double pi(double b)
{
    return 3.14 * b;
}

int main(void)
{
    double sq;
    double area;

    sq = square( 2.5 );
    area = pi( sq );
    return 0;
}
```

같은 호출 구조

연속 호출: 인자 안 호출

```
#include <stdio.h>

double square(double a)
{
    return a * a;
}

double pi(double b)
{
    return 3.14 * b;
}

int main(void)
{
    double area;

    area = pi( square( 2.5 ) );
    return 0;
}
```


함수의 호출

연속 호출: 인자 안 호출

```
#include <stdio.h>

double square(double x)
{
    return x * x;
}

double circle(double r)
{
    return 3.14 * square(r);
}

double cone(double base, double height)
{
    return base * height / 3.0;
}

int main(void)
{
    double volume;

    volume = cone( circle( 2.5 ), 3.5 );
    return 0;
}
```

