알고리즘의 제어

알고리즘은 4가지 제어구조로 명령어를 구성한다.

1. 순차적 제어

2. 선택적 제어

   조건 테스트에 의한 선택적 문장 실행

3. 반복적 제어

   반복제어변수(LCV)에 의한 정해진 수의 명령어들의 반복

   LCV의 초기화와 유한한 반복 후에 반복제어를 exit할 수 있는지 검토

4. 함수(부프로그램)의 호출과 복귀

   main() ⇄ f1() ⇄ f2() ⇄ f3()

Recursive Mechanism

재귀기법(Recursive Mechanism)

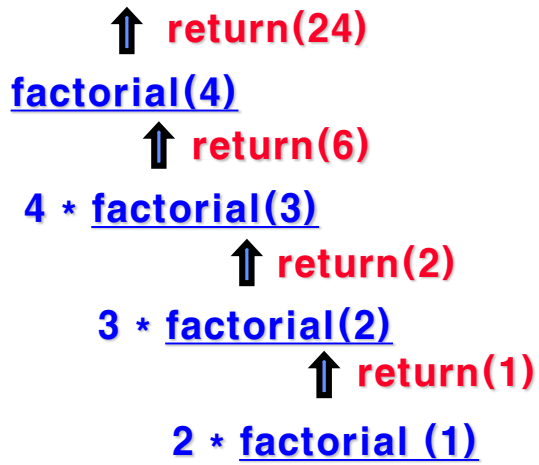
1. 재귀적 정의 (Recursive Definition)

   어떤 것을 정의 할 때 자신의 부분으로 재 정의 하는 것

   [예 1] computing factorials

$$n! = \begin{cases} 1 & \text{,if } n <= 1 \\ n * (n-1)! & \text{,otherwise} \end{cases}$$

```
Int factorial(int n)
{
    if (n <= 1)
        return 1;
    else
        return n * factorial(n-1);
}
```



재귀함수를 사용한 경우 반복으로 구현가능하다

재귀기법(Recursive Mechanism)

[예 2] 트리구조

이진트리의 정의 : 이진 트리는 공집합이거나 루트와 왼쪽 서브트리, 오른쪽 서브트리 두 개의 분리된 이진 트리로 구성된 노드의 유한집합이다.
또한 왼쪽 서브트리와 오른쪽 서브트리도 이진트리이다.

이진트리 운행(binary tree traversal)

```
void inorder(TNODETYPE *tptr)
{
    if (tptr)  {
        inorder(tptr->left);
        printf("%d", tptr->data);
        inorder(tptr->right);
    }
}
```

스택을 이용하여
반복으로 구현할 수 있지만
코드가 무척 복잡하다.

유클리드알고리즘

[정의] 두 수의 최대공약수(Great Common Divisor) 구하기

gcd(x, y) = x ,if y=0

= gcd(y, x%y) ,otherwise

예) gcd(24, 18) = gcd(18, 6) = gcd(6, 0)

1. 재귀적 정의를 이용한 알고리즘

```
START_GCD(x,y)
    if (y == 0)
       return(x)
    else
        call GCD(y, x%y)
    endif
END_GCD
```

2. 반복을 이용한 알고리즘

```
START_GCD(x,y)
    while (y != 0) do
        tmp = y
        y = x % y
        x = tmp
     endwhile
     return x
END_GCD
```

유클리드알고리즘

1. 재귀적 정의를 이용한 알고리즘 C함수 구현

```
int gcd(int x, int y)
{
    if (y == 0)
        return x;
    else
        return gcd(y, x%y);
}
```

2. 반복을 이용한 알고리즘 C함수 구현

```
int gcd(int x, int y)
{
        int tmp;
        while (y != 0) {
                tmp = y;
                y = x % y;
                x = tmp;
         }
         return x;
}
```

프로그래밍 실습과 정리

1. 다음의 main()을 이용하여 유클리드 알고리즘의 gcd함수를 실행시켜 결과를 확인해 보시오

```
main()
{
    int gvalue;
    int a, b, again=1;

    while (again) {
        printf("최대공약수를 구할 두 수를 입력 하세요. ");
        scanf("%d %d", &a, &b);
        gvalue = gcd(a, b);
        printf("gcd(%d, %d) = %d\n\n", a, b, gvalue);
        printf("계속하실래요?(1/0)");
        scanf("%d", &again);
    }
}
```

```
F:\aprog\class4\gcd_test.d
56 14
356 28
90 25
56 16
55 15
75 15
82 34
72 32
564 36
750 150
48 11
48 18
55 22
84 24
35 15
288 140
```

2. 최대공약수를 구하는 문제를 파일로부터 읽어서 그 문제를 제공하면 사용자가 풀고 맞으면 “Correct…”를 출력하고 틀리면 답을 출력해 준다. 또한 문제 내고 맞추는 것이 끝나면 전체 몇 문제 중에 몇 개 맞혔는지 출력하는 프로그램을 작성하시오

프로그래밍 실습과 정리

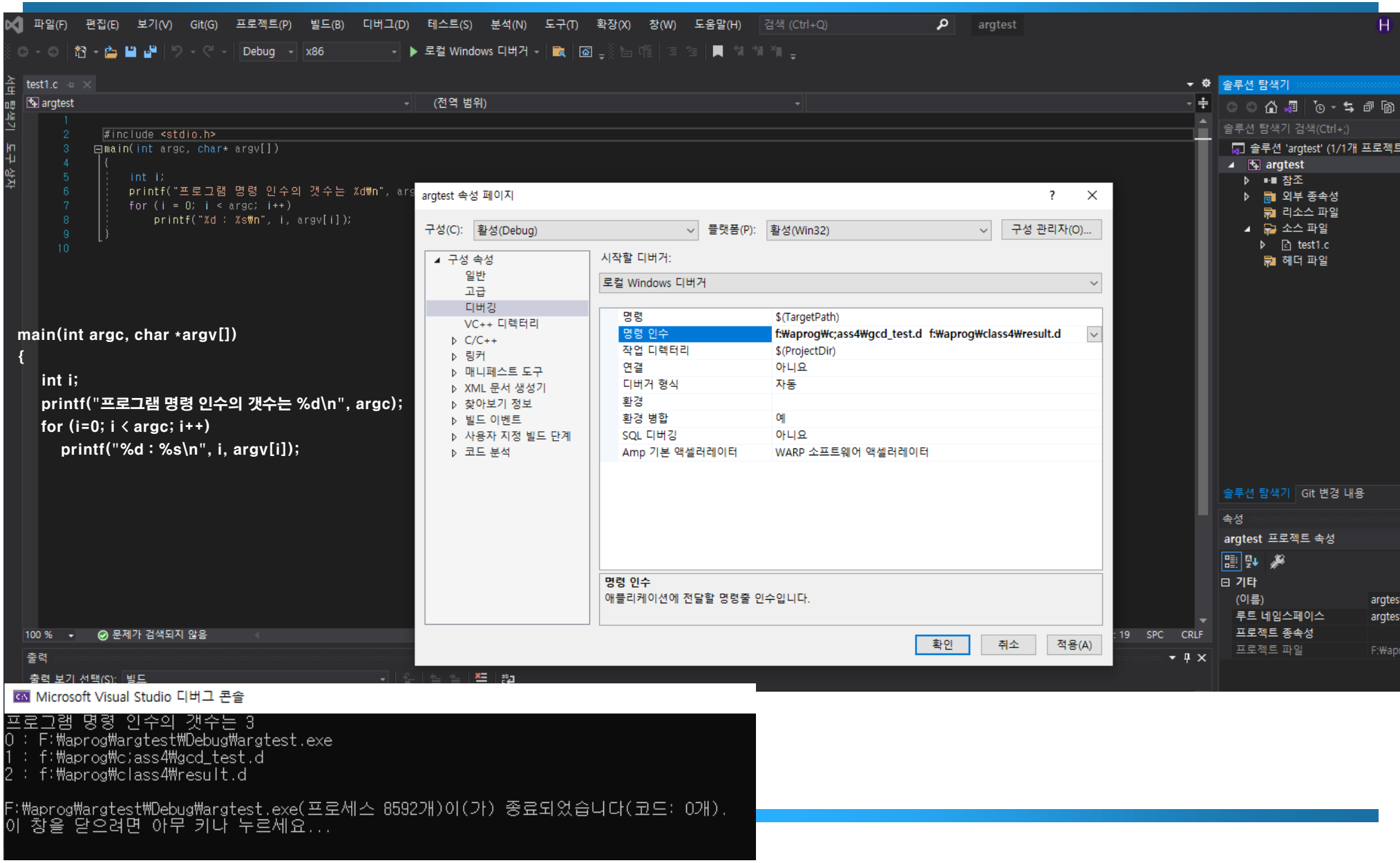
main()의 명령인수 받아들이기 = 컴파일러마다 다르다

visual studio : 프로젝트명에서 오른쪽 버튼..속성..디버깅..명령인수로 준다
Dev-C++ : 실행 탭 하위메뉴의 매개변수들
리눅스 : 명령어 라인 # cp f1 f2

```
main(int argc, char *argv[])
{
    int i;
    printf("프로그램 명령 인수의 갯수는 %d\n", argc);
    for (i=0; i < argc; i++)
        printf("%d : %s\n", i, argv[i]);
}
```

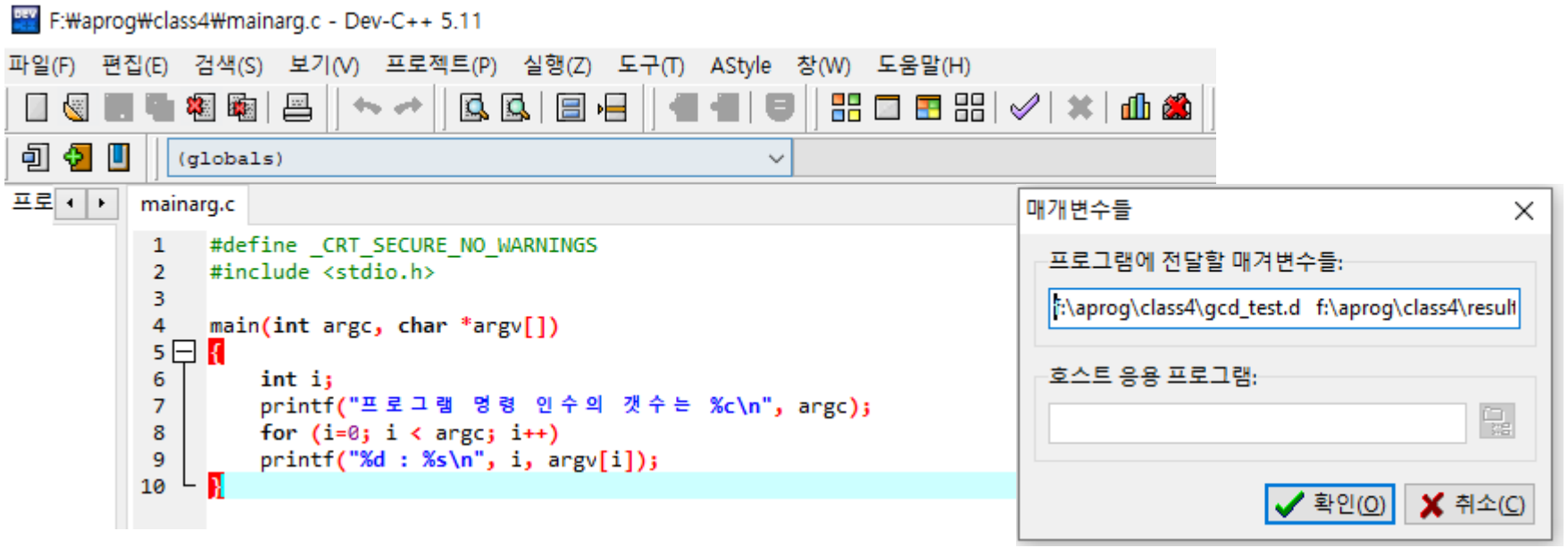
주로 main 함수의 인수는 그 프로그램의 실행을 위한 입출력파일을 지정할 때 많이 사용한다

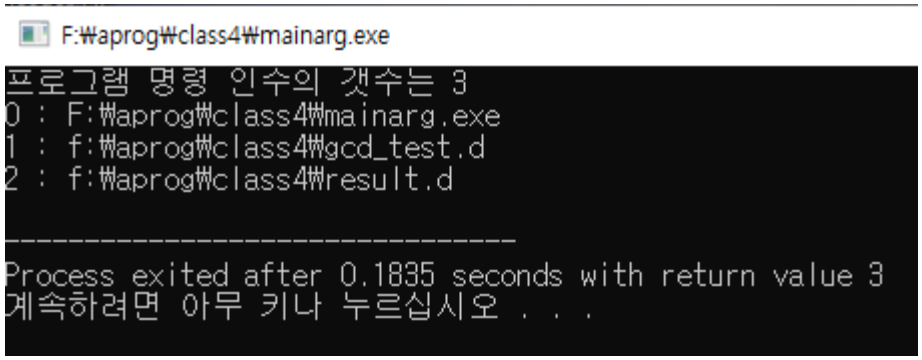
Visual Studio 사용




```
main(int argc, char *argv[])
{
    int i;
    printf("프로그램 명령 인수의 갯수는 %d\n", argc);
    for (i=0; i < argc; i++)
        printf("%d : %s\n", i, argv[i]);
```

Dev-C++사용


F:₩aprog₩class4₩mainarg.c - Dev-C++ 5.11
파일(F) 편집(E) 검색(S) 보기(V) 프로젝트(P) 실행(Z) 도구(T) AStyle 창(W) 도움말(H)
(globals)
mainarg.c
1 #define _CRT_SECURE_NO_WARNINGS
2 #include <stdio.h>
3
4 main(int argc, char *argv[])
5 {
6 int i;
7 printf("프로그램 명령 인수의 갯수는 %c\n", argc);
8 for (i=0; i < argc; i++)
9 printf("%d : %s\n", i, argv[i]);
10 }
매개변수들
프로그램에 전달할 매개변수들:
f:\aprog\class4\gcd_test.d f:\aprog\class4\result
호스트 응용 프로그램:
확인(O)
취소(C)


F:₩aprog₩class4₩mainarg.exe
프로그램 명령 인수의 갯수는 3
0 : F:₩aprog₩class4₩mainarg.exe
1 : f:₩aprog₩class4₩gcd_test.d
2 : f:₩aprog₩class4₩result.d
Process exited after 0.1835 seconds with return value 3
계속하려면 아무 키나 누르십시오 . . .

정리하며 실습하기

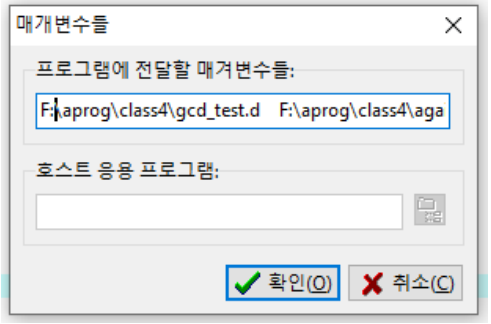
```
main(int argc, char *argv[])
{
    int a, b, result, answer;
    int correct=0, i=0;
    FILE *infile;
    if (argc != 2) {
       printf("실행인수를 잘못 주었습니다...\n");
       exit(1);
    }

    if ((infile = fopen(argv[1], "r")) == NULL) {
        printf("입력 파일을 열 수 없습니다. \n");
        exit(1);
    }
```

```
  while (fscanf(infile, "%d %d", &a, &b) != EOF ) {
      printf("%d : gcd(%d, %d) = ", i+1, a, b);
      scanf("%d", &answer);
      result = gcd(a, b);
      if (answer != result)
         printf("Answer : gcd(%d, %d) = %d \n", a, b, result);
      else {
          printf("Correct....\n");
          correct++;
       }
       i++;
  }

  printf("%d 문제 중에 %d 문제 통과하셨습니다....\n", i, correct);

}
```

정리하며 실습하기

3. 앞의 gcd 프로그램에서 틀린 문제를 출력파일 again.d에 출력하도록 프로그램을 수정하시오

```
main(int argc, char *argv[])
{
    int a, b, result, answer;
    int correct=0, i=0;
    FILE *infile, *out;
    if (argc != 3) {
       printf("실행인수를 잘못 주었습니다...\n");
       exit(1);
    }

    if ((infile = fopen(argv[1], "r")) == NULL) {
       printf("입력 파일을 열 수 없습니다. \n");
       exit(1);
    }

   if ((out = fopen(argv[2], "w")) == NULL) {
        printf("출력파일을 열수없습니다\n");
        exit(1);
}
```




```
F:\aprog\class4\gcd_test.d
56 14
356 28
90 25
56 16
55 15
75 15
82 34
72 32
564 36
750 150
48 11
48 18
55 22
84 24
35 15
288 140
```

정리하며 실습하기

```
  while (fscanf(infile, "%d %d", &a, &b) != EOF ) {
      printf("%d : gcd(%d, %d) = ", i+1, a, b);
      scanf("%d", &answer);
      result = gcd(a, b);
      if (answer != result)   {
         fprintf(out,  "%d %d\n" , a, b);
         printf("Answer : gcd(%d, %d) = %d \n", a, b, result);
      }
      else {
          printf("Correct....\n");
          correct++;
       }
       i++;
  }
  printf("%d 문제 중에 %d 문제 통과하셨습니다....\n", i, correct);

}
```

F:\aprog\class4\again.d
56 14
356 28
82 34
564 36
84 24
288 140

```
1 : gcd(56, 14) = 7
Answer : gcd(56, 14) = 14
2 : gcd(356, 28) = 14
Answer : gcd(356, 28) = 4
3 : gcd(90, 25) = 5
Correct....
4 : gcd(56, 16) = 8
Correct....
5 : gcd(55, 15) = 5
Correct....
6 : gcd(75, 15) = 15
Correct....
7 : gcd(82, 34) = 17
Answer : gcd(82, 34) = 2
8 : gcd(72, 32) = 8
Correct....
9 : gcd(564, 36) = 6
Answer : gcd(564, 36) = 12
10 : gcd(750, 150) = 150
Correct....
11 : gcd(48, 11) = 1
Correct....
12 : gcd(48, 18) = 6
Correct....
13 : gcd(55, 22) = 11
Correct....
14 : gcd(84, 24) = 6
Answer : gcd(84, 24) = 12
15 : gcd(35, 15) = 5
Correct....
16 : gcd(288, 140) = 7
Answer : gcd(288, 140) = 4
16 문제 중에 10 문제 통과하셨습니다.
```

정리하며 실습하기

3. 앞의 gcd 프로그램에서 틀린 문제를 출력파일 again.d에 출력하도록 프로그램을 수정하시오

```
main(int argc, char *argv[])
{
    int a, b, result, answer;
    int correct=0, i=0;
    FILE *infile, *out;
    if (argc != 3) {
       printf("실행인수를 잘못 주었습니다...\n");
       exit(1);
    }

    if ((infile = fopen(argv[1], "r")) == NULL) {
       printf("입력 파일을 열 수 없습니다. \n");
       exit(1);
    }

   if ((out = fopen(argv[2], "w")) == NULL) {
        printf("출력파일을열수없습니다\n");
        exit(1);
}
```

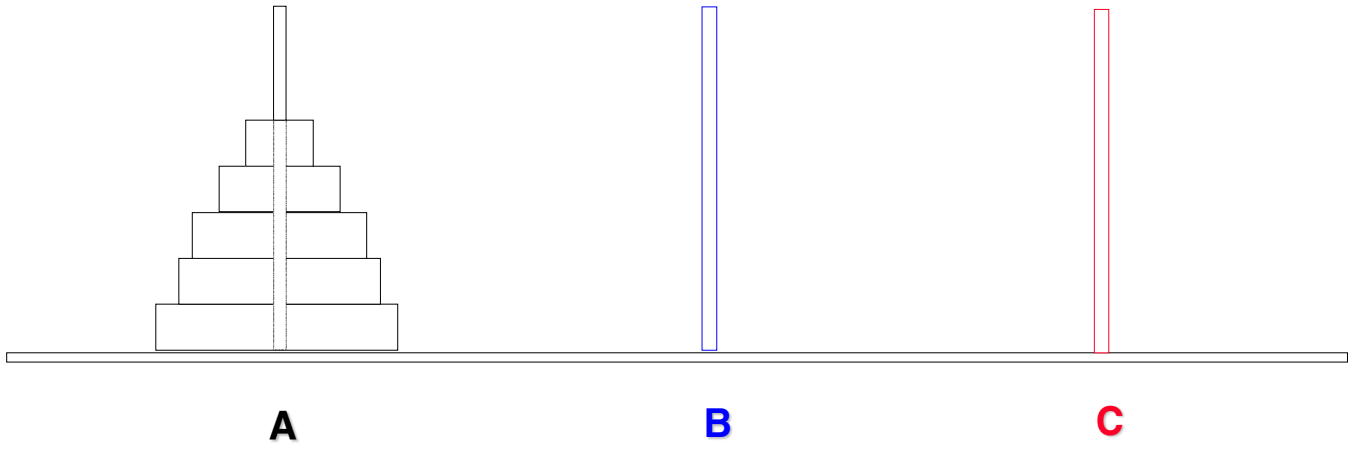
```
  while (fscanf(infile, "%d %d", &a, &b) != EOF ) {
      printf("%d : gcd(%d, %d) = ", i+1, a, b);
      scanf("%d", &answer);
      result = gcd(a, b);
      if (answer != result)   {
         fprintf(out,  "%d %d\n" , a, b);
         printf("Answer : gcd(%d, %d) = %d \n", a, b, result);
      }
      else {
          printf("Correct....\n");
          correct++;
       }
       i++;
  }
  printf("%d 문제 중에 %d 문제 통과하셨습니다....\n", i, correct);

}
```

하노이 탑

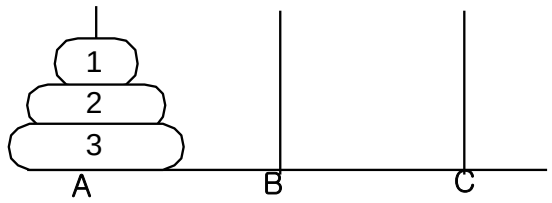
[정의] [막대 A]의 크기 순으로 쌓여 있는 디스크를 다음의 조건에 따라 [막대C]로 옮기기

(1) 디스크는 한번에 하나밖에 움직이지 못한다.

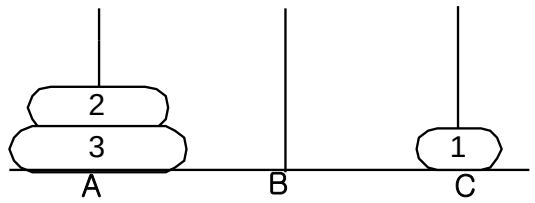
(2) 항상 큰 디스크 위에 작은 디스크를 놓아야 한다.



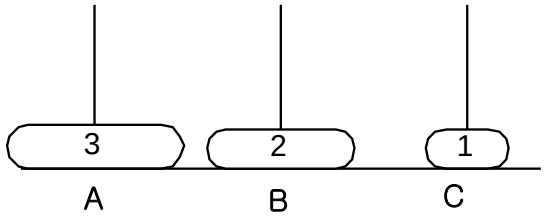
하노이 탑



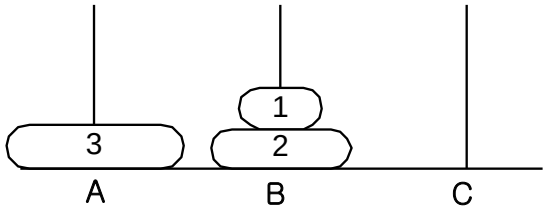
1



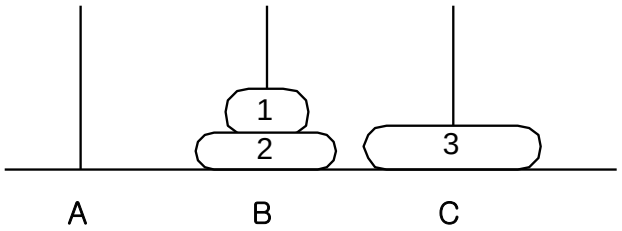
2



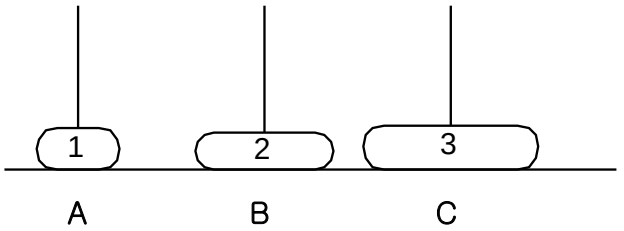
3



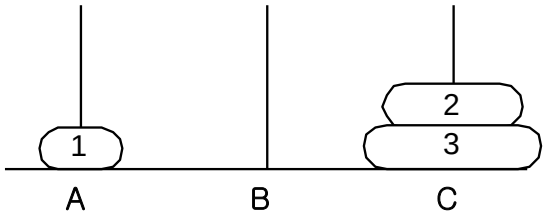
4



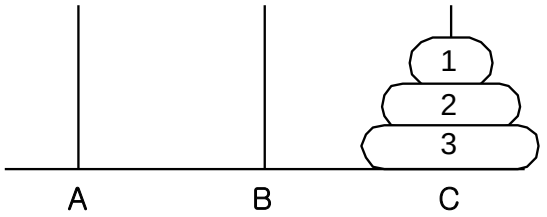
5



6



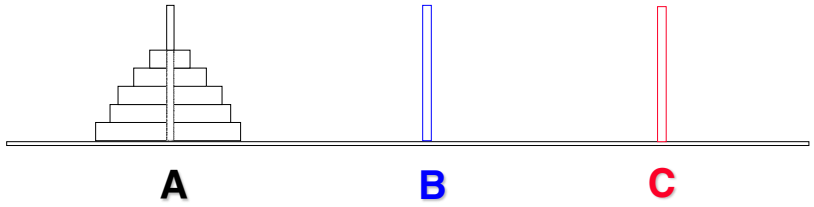
7



하노이 탑

1,2,3 세 개의 디스크를 막대 A로부터 막대 C로 옮기려면
(1) 디스크 1과2를 막대 B로 옮긴다.
(2) 막대 A의 가장 큰 디스크 3을 목적지 막대 C로 옮긴다.
(3) 막대 B의 디스크 1과2를 막대 C로 옮긴다.

n개의 디스크에 대하여 확장하면




```
if (n==1)
        디스크를 직접 A로부터 C로 옮긴다.
else  {
        가장 큰 디스크를 제외한 n-1개의 디스크를
        A로부터 B로 옮긴다.
        A에 남아있는 가장 큰 디스크를 C로 옮긴다.
        B에 있는 n-1 개의 디스크를 목적지 C로 옮긴다.
 }
```

하노이 탑

[하노이 탑 문제의 C함수 구현]
입력 : 디스크의 개수, 출발 막대, 목적 막대, 중간 막대
출력 : 각 단계에서 디스크가 어느 막대로부터 어느 막대로 움직이는지 출력

hanoi(3, 'A', 'C', 'B')

```
void hanoi(int n, char a, char c, char b)
{
        if (n == 1)
            printf("Move disk from %c to %c\n", a, c);
        else {
            hanoi(n-1, a, b, c);
            hanoi(1, a, c, b);
            hanoi(n-1, b, c, a);
        }
}
```

하노이 탑 실습하기

디스크의 개수 n을 입력 받고
함수 void hanoi(int n, char a, char, c, char b) 를 호출하는 main()을 작성하여 프로그램을 실행시켜 보자.

```
void hanoi(int n, char a, char c, char b);
long count=0;
void main()
{
    int n;
    printf("Input the number of disk: ");
    scanf("%d", &n);
    if (n <= 0)
        printf("\n No disk!!\n");
    else
       hanoi(n, 'A', 'C', 'B');
   printf("Moving count = %ld\n", count);
}

void hanoi(int n, char a, char c, char b)
{
   if (n == 1) {
      count++;
      printf("Move disk from %c to %c\n", a, c);
   }
   else {
      hanoi(n-1, a, b, c);
      hanoi(1, a, c, b);
      hanoi(n-1, b, c, a);
   }
}
```

```
Input the number of disk: 4
Move disk from A to B
Move disk from A to C
Move disk from B to C
Move disk from A to B
Move disk from C to A
Move disk from C to B
Move disk from A to B
Move disk from A to C
Move disk from B to C
Move disk from B to A
Move disk from C to A
Move disk from B to C
Move disk from A to B
Move disk from A to C
Move disk from B to C
Moving count = 15
```