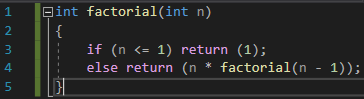
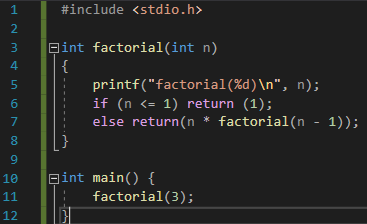
2.1 순환의 소개

순환 어떤 알고리즘이나 함수가 자기 자신을 호출하여 문제를 해결하는 프로그래밍 기법

순환의 예 순환문제, 순환자료구조를 다루는 프로그램에 적합

ex) 팩토리얼

프로그램 2.1 

프로그램 2.2 

순환호출의 내부적인 구현 순환 = 다른 함수를 호출하는 것과 동일

복귀 주소가 시스템 스택에 저장, 호출되는 함수의 매개변수∙지역변수를 스택으로부터 할당

활성 레코드 함수를 위한 시스템스택의 공간

호출된 함수가 끝나게 되면 복귀주소를 추출해 호출한 함수로 돌아감

ex) factorial(3) 🡪 factorial(3), factorial(2), factorial(1)의 활성레코드 생성

3 🡪 3,2 🡪 3,2,1 🡪 3,2 🡪 3

순환 알고리즘의 구조 자기자신을 순환적으로 호출하는 부분

순환호출을 멈추는 부분 순환호출을 멈추는 부분이 없다면? 무한한 순환호출 🡪 오류발생(스택 오버플로우)

순환 🡨🡪 반복 반복 for, while등의 반복구조로 되풀이

간단, 효율적

순환 주어진 문제를 해결하기위해 자신을 다시 호출해 작업 수행

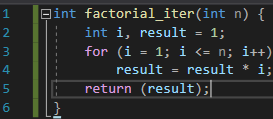
순환적인 문제∙자료구조

수행속도 면에서 떨어짐

꼬리순환 순환호출이 끝에서 이뤄지는 경우

반복알고리즘으로 쉽게 바꿀 수 있음

반복, 순환은 문제해결능력이 같음 🡪 반복, 순환 변환 가능

프로그램 2.3 

바람직한 형태 그때마다 다름

순환 이해하기 쉬움, 간단한 코딩

느린 실행시간

순환을 쓰지 않으면 작성할 수 없는 프로그램도 존재 🡪 순환은 필수

순환의 원리 문제의 일부를 해결한 다음, 나머지 문제에 대해 순환호출 🡪 문제가 점점 작아짐 = 풀기 쉬워짐

ex) else return (n\*/\*해결된 부분\*/factorial(n-1)/\*남아있는 부분\*/)

분할 정복 주어진 문제를 더 작은 동일한 문제로 분할하여 해결하는 방법

순환 알고리즘의 성능 반복을 통한 팩토리얼 n번 반복 = O(n)

순환을 통한 팩토리얼 n번 순환 = O(n)

여분의 저장공간, 사전작업 필요 🡪 반복보다 더 긴 시간 필요

Quiz 1. 22

2. int sub\_iter(int n) {

int res = 0;

for (int i = n; i > 0; i -= 3) {

res += i;

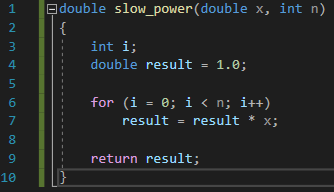
}

return res;

}

2.2 거듭제곱값 계산

순환이 반복보다 빠른 문제

반복 알고리즘 프로그램 2.4 

순환알고리즘 알고리즘2.1 power(x, n) :

if n == 0

then return 1;

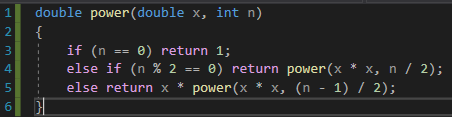
else if n이 짝수

then return power(x \* x, n / 2);

else if n이 홀수

then return x \* power(x \* x, (n - 1) / 2);

xn=(x2)n/2임을 이용

프로그램 2.5 

수행시간 차이 반복 O(n)

순환 O(log2n)

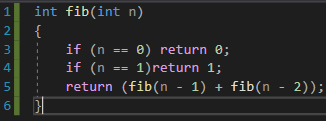
Quiz 1. 6 🡪 3 🡪 1

2. O(n)🡪O(log n)줄어든다

2.3 피보나치 수열의 계산

반복이 순환보다 빠른 문제

피보나치 수열 앞의 두개의 숫자를 더해 뒤의 숫자를 만드는 것

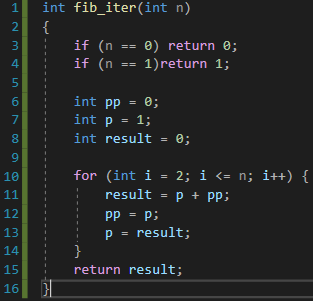
순환 프로그램 2.6 

매우 비효율적 이전 항을 구하기 위해 계속 중복 계산함

ex) fib(6)을 구하기 위해, fib(4)는 2번, fib(3)은 3번, fib(2)는 5번, fib(1)은 8번 호출됨

시간복잡도 T(n) = T(n-1)+T(n-2)+C

O(2n)

반복 프로그램 2.7 

Quiz 1. 3번

2. O(n)

3. 하나의 호출이 두개로 나누어지니 n 🡪 2n

2.4 하노이탑 문제

문제 막대A에 쌓여있는 원판 3개를 막대C로 옮기는 것

조건 한번에 하나의 원판만 이동할 수 있다

맨 위에 있는 원판만 이동할 수 있다

크기가 작은 원판 위에 큰 원판이 쌓일 수 없다

중간의 막대를 임시적으로 이용할 수 있으나 앞의 조건들을 지켜야 한다.

의사코드 void hanoi\_tower(int n, char from, char tmp, char to)

{

if(n==1) from에 있는 한개의 원판을 to로 옮긴다.

else

{

from의 맨 밑의 원판을 제외한 나머지 원판들을 tmp로 옮긴다 //n-1개의 원판을 to를 사용해 from에서 tmp로 옮기는 것

from에 있는 한개의 원판을 to로 옮긴다 //쉬움

tmp의 원판들을 to로 옮긴다 //n-1개의 원판을 tmp를 사용해 from에서 to로 옮기는 것

}

}

🡪

void hanoi\_tower(int n, char from, char tmp, char to)

{

if(n==1) from에 있는 한개의 원판을 to로 옮긴다.

else

{

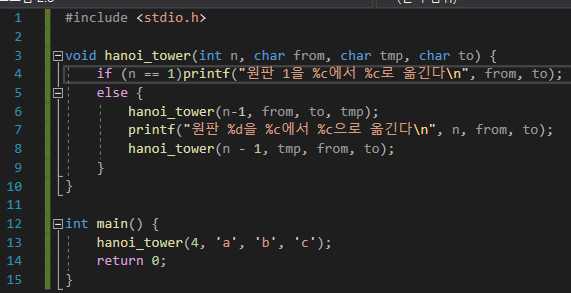
hanoi\_tower(n-1, from, to, tmp); //n-1개의 원판을 to를 사용해 from에서 tmp로 옮기는 것

from에 있는 한개의 원판을 to로 옮긴다 //쉬움

hanoi\_tower(n-1, tmp, from, to); //n-1개의 원판을 from를 사용해 tmp에서 to로 옮기는 것

}

}

프로그램 2.8 

반복적인 형태로 바꾸기 어려운 순환 꼬리순환 순환호출이 순환함수의 맨 끝에서 이뤄지는 경우

쉽게 반복형태로 변환가능

머리순환 여러군데에서 순환호출이 이뤄지는 경우

쉽게 반복형태로 변환 불가능

꼬리순환, 머리순환 모두 양쪽으로 표현할 수 있다면, 꼬리순환으로 해야함

Quiz 1. 2

연습문제

1. 5

2. 4

3. 4

4. 3

5. int recursive(int n)

{

if (n == 1) return 0;

return n \* recursive(n); //return n\*recursive(n-1);

}

6. int recursive(int n)

{

printf("recursice(%d)\n", n);

//if (n == 1) return 0;

return n \* recursive(n-1);

}

7. 5

4

3

2

1

0

반환값: 16

8. 5

4

3

2

1

0

반환값: 95

9. 10

7

4

1

-2

반환값: 3

10. 1

2

3

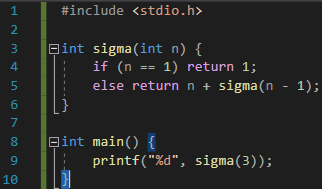
4

5

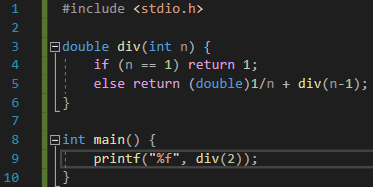
11. 7

12. evisrucer

13.



14.



15.

fib(6) is called

fib(5) is called

fib(4) is called

fib(3) is called

fib(2) is called

fib(1) is called

fib(0) is called

fib(1) is called

fib(2) is called

fib(1) is called

fib(0) is called

fib(3) is called

fib(2) is called

fib(1) is called

fib(0) is called

fib(1) is called

fib(4) is called

fib(3) is called

fib(2) is called

fib(1) is called

fib(0) is called

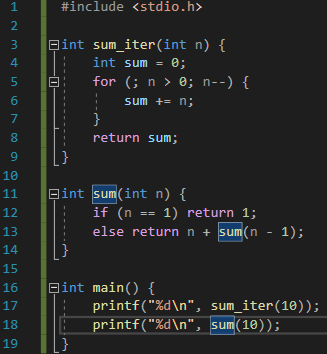
fib(1) is called

fib(2) is called

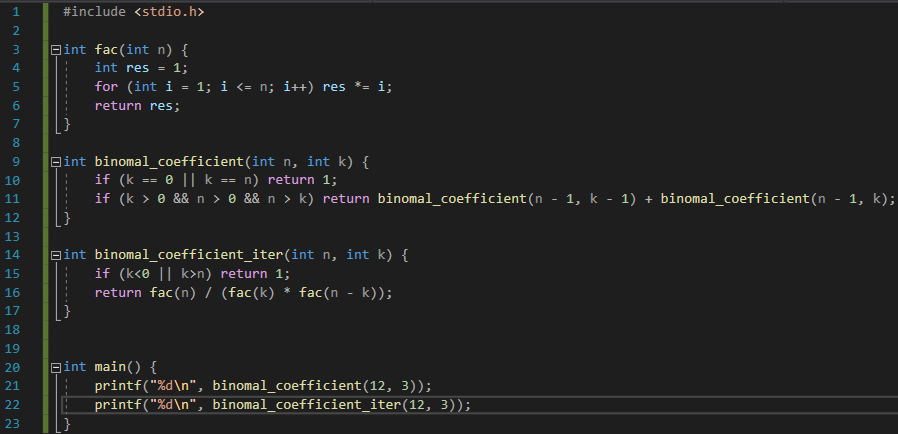
fib(1) is called

fib(0) is called

16.

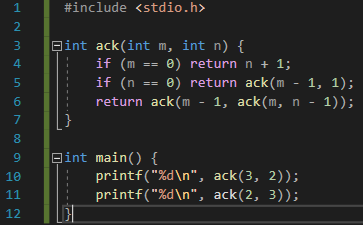


17.

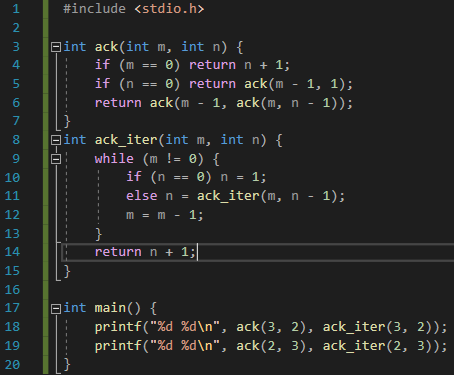


18. (a) 29, 9

18. (b)



18. (c)



19.

반복적인 피보나치 수열이 더 빠르다.

순환 O(n2)

반복 O(n)

20. (1) 문제가 return n\*fac(n-1)로 1씩 작은 함수를 호출해가며, 문제가 작아지고 있다

20. (2) 문제가 원판을 하나씩 목적지로 옮기고 있으므로 문제가 작아지고 있다.

21.

