재귀함수

Recursion Function

201807027 이태양

개요

재귀함수

하나의 함수가 자기 자신(함수)을 다시 호출하는 방식을 재귀함수라 한다.

```
void HI()
{
    cout << "HI"<<endl;
    HI();
}
```

재귀함수

재귀 결과 (무한 루프)

특징

장점

이해하기 쉽다.

코드를 간단하게 작성할 수 있다.

변수의 사용을 줄여준다.

단점

반복문에 비해 메모리 사용과 실행시간이 길어질 수 있다.

경우에 따라 스택 오버플로우가 발생할 수 있다.

스택 오버플로우

메모리에서 지역변수를 저장하는 스택의 크기가 꽉 찼을 때 발생하는 에러이다.

```
void HI()
{
    cout << "HI"<<endl;
    HI();
}
```

다음 코드에서 재귀함수를 끝내는 조건을 설정하지 않으면

무한루프와 같이 HI라는 함수가 끝없이 호출되어 스택에 적재되다가

할당된 스택의 크기를 벗어나 에러가 발생한다.

사용법

처음 코드의 스택 오버플로우를 해결하기 위해 다음과 같이 수정해야 한다.

```
int main()
{
    HI( 1 );
}

void HI(int num)
{
    if (num == 6) return;

    cout << num << " : HI"<<endl;
    HI(num + 1);

}
```

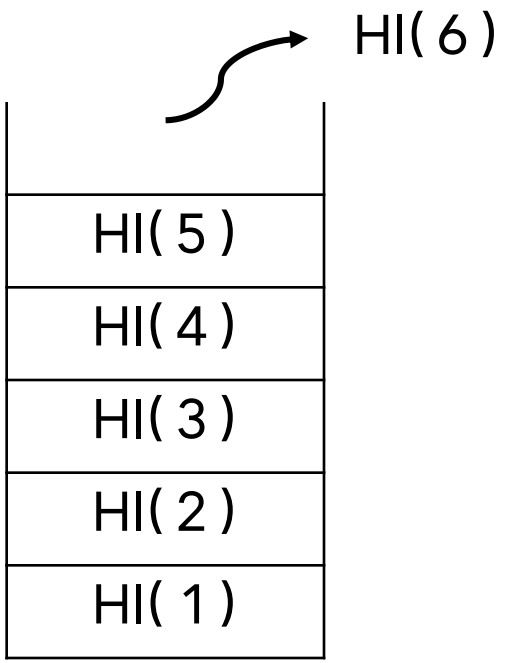
HI 함수가 재귀 호출될 때 마다 매개변수 num을 1씩 증가 시키다

num이 6되었을 때 함수를 종료 시킨다.

사용법

이때 함수가 리턴 되어도 바로 그 다음 스택의 TOP인 함수가

이어서 실행된다.




```
void HI(int num)
{
    if (num == 6)
    {
        return;
    }

    cout << num << " : HI"<<endl;

    HI(num + 1);

    // 코드 추가
    cout << num << " : 리턴" << endl;
}
```

```
1 : HI
2 : HI
3 : HI
4 : HI
5 : HI
5 : 리턴
4 : 리턴
3 : 리턴
2 : 리턴
1 : 리턴
```

사용 예

상황에 따라 개발자가 재귀함수 혹은 반복문으로 작성할지

결정하면 된다.

```cpp
int main()
{
    cout << fact(5);
}

// 재귀함수로 구현한 팩토리얼
int fact(int num)
{
    if (num == 1) return 1;

    return num * fact(num - 1);
}
```

재귀함수

```cpp
int main()
{
    cout << fact2(5);
}

// 반복문으로 구현한 팩토리얼
int fact2(int num)
{
    int result = 1;
    for (int i = 1; i <= num; i++)
    {
        result *= i;
    }

    return result;
}
```

반복문