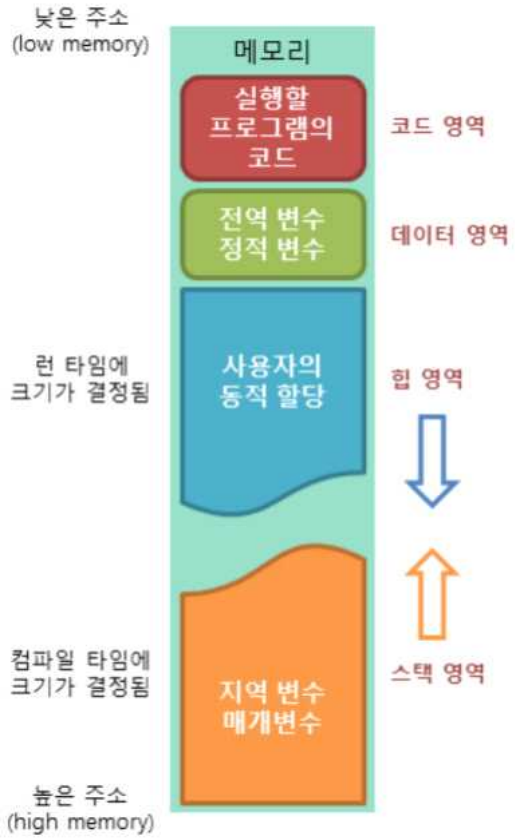
<제2강 console 게임 제작>

1. 메모리



① 코드(code) 영역
- 상수 등

② 데이터(data) 영역
- 전역 변수, 정적 변수
- 프로그램 시작 ~ 종료

③ 스택(stack) 영역
- 지역, 매개 변수 등
- 함수의 시작 ~ 종료

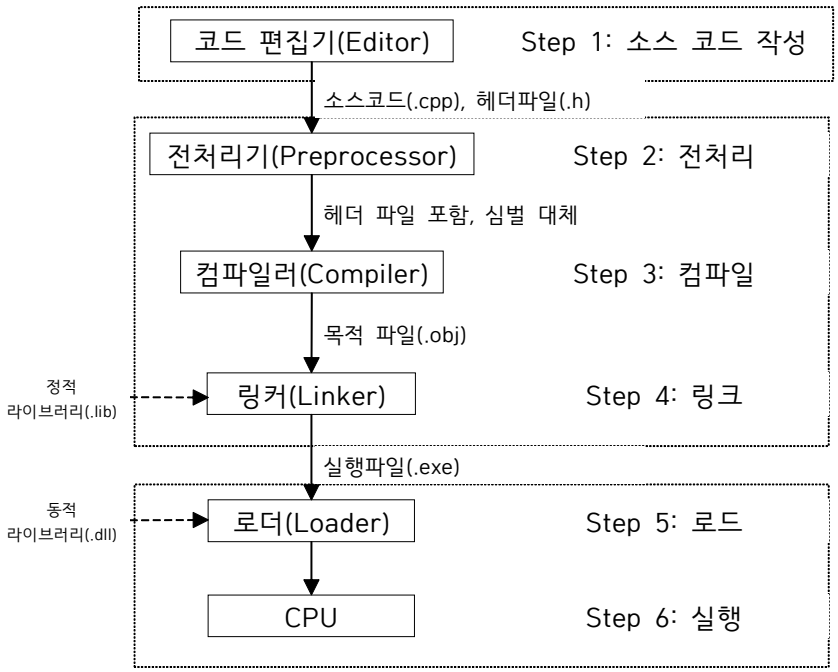
※ Stack Overflow

④ 힙(heap) 영역
- 동적할당
- 사용자에 의해 설정

★ 사용자가 반드시 관리해야 하는 영역

※ Heap Overflow

2. 빌드 과정



※ 빌드(build) 과정(= 컴파일 + 링킹)

① 전처리
- 주석 제거: 컴퓨터는 필요 없음.
- 헤더 파일 삽입: #include 복사

=> cpp에 있는 함수 원형은 링킹할 때 오브젝트 파일과 결합함.
- 매크로 치환: #define 매크로 적용

② 컴파일: 컴파일러는 최종 소스 파일을 기계어(어셈블리어)로 변환하여 오브젝트 파일(object)을 만든다.
- 문법 검사
- Data영역 메모리 할당

③ 링크:
- obj파일 묶어서 실행 파일 제작
- 라이브러리 연결

※ 유의사항
- cpp 파일들은 각각 독립적으로 컴파일 되며, 컴파일이 완료된 cpp파일들을 링킹한다.
- #include하는 헤더파일은 컴파일이 아니라 cpp파일로 복사될 뿐이다.

3. 헤더(.h)와 소스파일(.cpp)을 분리하자.

1) #include ???

< >: 표준으로 사용되는 것 " ": 정의한 것.(경로에서 찾는다.)

=> 헤더 파일 분리 방법: 선언은 헤더파일에, 정의는 cpp파일로 정리한다.

Q) 헤더파일을 사용 및 분리하는 이유

- 파일 하나에 길게 코딩하면 복잡한 프로그램을 제작할때 힘들기 때문에 분리한다.
- 다른 소스 파일들에 포함시킬 목적으로 재사용하기 위해 분리한다.

2) 헤더 가드

- 헤더 가드 종류

① #ifndef, #endif

```
#ifndef MY_HEADER
#define MY_HEADER
int TestAdd(int a, int b)
{
    return a+b;
}
#endif // !MY_HEADER
```

② #pragma once: c++ 공식 언어는 아님, MS에서 지원

```
#pragma once
int TestAdd(int a, int b)
{
    return a+b;
}
```

- 헤더가드(#pragma once)가 필요한 이유

=> 정의가 중복되면 안되기 때문에!

ex.

[Add.h]

```
#pragma once
int TestAdd(int a, int b)
{
        return a+ b;
}
```

[Test.h]

```
#pragma once
#include "Add.h"
int PlusAndMul()
{
  int temp = TestAdd(10, 20);
  temp *= 10;
  return temp;
}
```

[main.cpp]

```
#include<iostream>
#include "Add.h"
#include "AddMul.h"
using namespace std;
int main(){
  cout << TestAdd(100, 200) << endl;
  cout << PlusAndMul() << endl;
}
```