**C언어 메모리 구조**



10514나정휘

프로그래밍 과목

이정태 선생님

2018.06.25

**-목차-**

**제1장. 포인터**

가. 주소 값

나. 포인터

**제2장. 전역 변수와 지역 변수**

가. 지역 변수

나. 전역 변수

다. 정적 변수

**제3장. 메모리의 구조**

가. 메모리 영역의 구성 요소

**제4장. 코드(Code) 영역**

가. 코드 영역

**제5장. 데이터(Data) 영역**

가. 데이터 영역의 시작과 끝

나. 전역 변수, 정적 변수

다. BSS영역

**제6장. 자료구조 스택(Stack)**

가. LIFO 구조

나. overflow 오류

**제7장. 스택(Stack) 영역**

가. 지역 변수

나. 스택 프레임

**제8장. 힙(Heap) 영역**

가. 힙 영역의 용도

나. C언어의 메모리 동적 할당

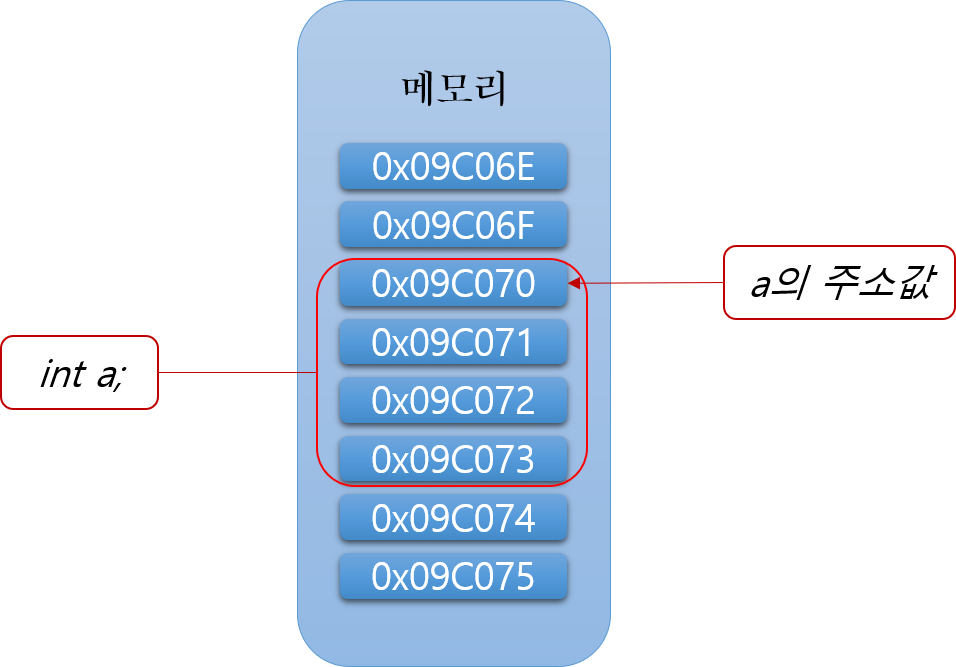
**서론**

이 보고서를 작성하는 과정을 통해 컴퓨터의 메모리 구조, 특히 C언어에서 다룰 수 있는 영역에 대해 더욱 깊이 연구하고, 더욱 자세히 알아가기를 바란다.

**제1장. 포인터**

**가. 주소 값**

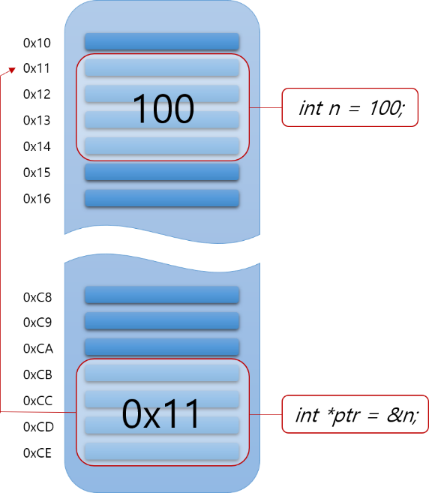
데이터의 주소는 해당 데이터가 저장된 메모리 주소의 첫 번째 바이트를 나타낸다.



**나. 포인터**

C언어에서 포인터는 메모리의 주소 값을 저장하는 변수이다.

|  |
| --- |
| *int n = 100;*  *int \*ptr = &n;* |



**제2장. 전역 변수와 지역 변수**

**가. 지역 변수**

C언어에서 변수는 크게 지역 변수와 전역 변수 2가지로 나뉜다.

지역 변수는 특정 블록 내부에서 선언된 변수를 의미한다. 지역 변수는 선언된 블록 스코프 내에서만 접근이 가능하고, 해당 함수가 끝날 때 메모리에서 해제된다. 지역 변수는 후술할 스택(Stack) 영역에 저장된다.

**나. 전역 변수**

전역 변수는 블록 외부에서 선언된 변수를 의미한다. 전역 변수는 어디에 서든 접근이 가능하고, 프로그램 시작 전에 할당되며, 프로그램이 끝날 때 메모리에서 해제된다. 전역 변수는 후술할 데이터(Data) 영역에 저장된다.

**다. 정적 변수**

정적 변수는 static 키워드를 사용해 선언한다. 선언이 이루어진 블록 스코프 내에서만 접근이 가능하지만, 함수가 끝날 때 메모리에서 해제되는 지역 변수와 다르게, 프로그램이 끝날 때까지 현재 자리에 그대로 남아서 값을 유지한다. 전역 변수와 마찬가지로 프로그램 시작 전에 할당되며, 후술할 데이터(Data) 영역에 저장된다.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **지역 변수** | **전역 변수** | **정적 변수** |
| **할당 시기** | 함수 실행 시 | 프로그램 시작 시 | 프로그램 시작 시 |
| **해제 시기** | 함수 종료 시 | 프로그램 종료 시 | 프로그램 종료 시 |
| **접근 가능 위치** | 특정 블록 스코프 | 전체 | 특정 블록 스코프 |
| **저장 영역** | 스택 영역 | 데이터 영역 | 데이터 영역 |

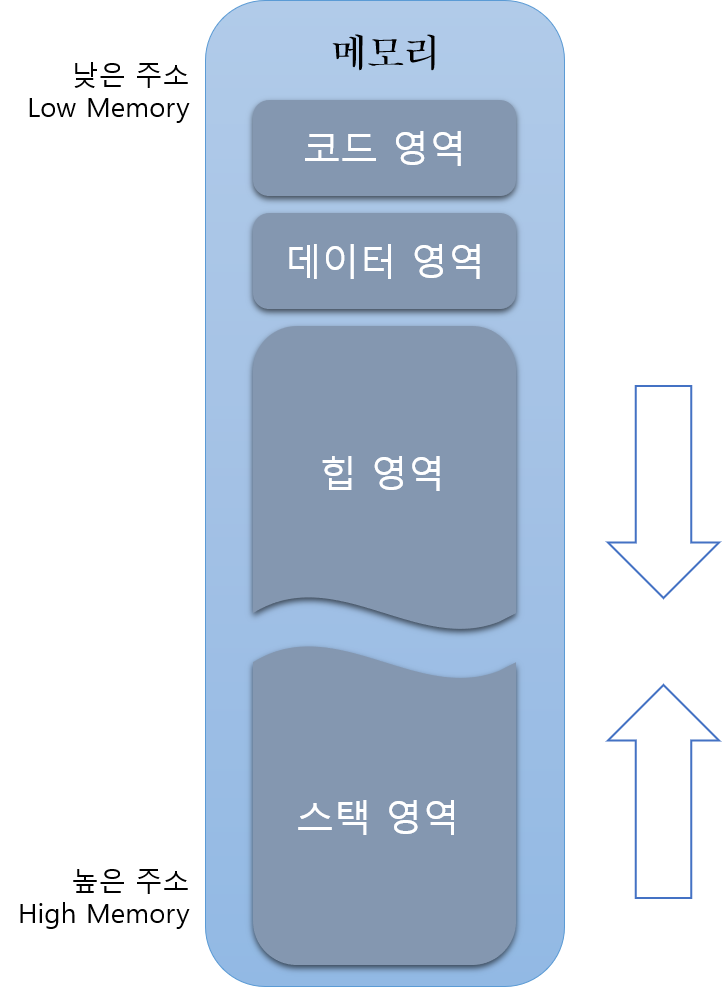
**제3장. 메모리 구조**

**가. 메모리 영역의 구성 요소**

메모리에는 여러 가지 영역이 있다. 그 중, 프로그램이 OS로부터 할당 받는 대표적인 메모리 영역은 약 4가지 가 있다.

메모리의 낮은 주소(low memory)부터 높은 주소(high memory)까지 차례로

코드(Code) 영역, 데이터(Data) 영역, 스택(Stack) 영역, 힙(Heap) 영역이 존재한다.



**제4장. 코드(Code) 영역**

**가. 코드 영역**

코드 영역에는 여러 데이터가 들어간다. 대표적인 것 들을 살펴보자면,

우리가 작성한 소스코드가 코드 영역에 저장이 된다. 물론 컴파일 된 기계어 형태로 저장이 된다. 또한, 상수 도 코드 영역에 저장이 된다.

상수 란, 한 번 값을 지정해주면 더 이상 값이 바뀌지 않는 데이터를 말한다. 보통 const 키워드를 사용해 선언한다.

**제5장. 데이터(Data) 영역**

**가. 데이터 영역의 시작과 끝**

데이터 영역은 프로그램 시작과 함께 메모리에 적재되며, 프로그램 종료 시 메모리에서 해제가 된다. 이러한 특성을 가지고 있는 전역 변수와 정적 변수가 데이터 영역에 저장이 된다. 또한, 구조체도 데이터 영역에 저장이 된다.

참고로, 초기화 되지 않은 데이터는 BSS(Block Stated Symbol) 영역에 저장이 되고, 초기화 된 데이터만 데이터 영역에 저장된다.

**나. 전역 변수, 정적 변수**

위 내용과 같이 전역, 정적 변수는 데이터 영역에 저장이 되고, 프로그램 시작 시 할당, 종료 시 해제된다. 정적 변수는 프로그램이 시작될 때에는 공간만 할당이 되고, 선언이 된 함수 실행 시에 초기화가 된다.

**다. BSS 영역**

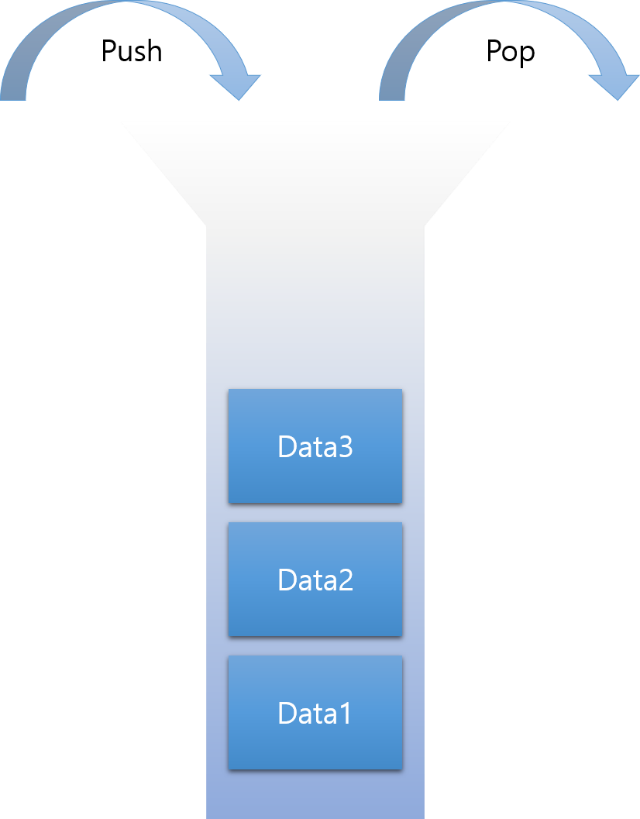
앞에서 서술한 대로 BSS(Block Stated Symbol) 영역에는 초기화가 되지 않은 데이터 들이 저장이 된다.

BSS영역에는 실제 변수가 차치할 공간이 할당되지 않아 그 만큼 binary size가 작아진다.

**제6장. 자료구조 스택(Stack)**

**가. LIFO 구조**

“스택” 이라는 자료구조는 후입 선출, 즉 Last-In First-Out(LIFO) 형태를 갖는 자료구조이다. 스택은 한 쪽 구멍에서 데이터의 입출력이 모두 이루어진다. 그러므로, 먼저 들어간 것이 아래로 내려가기 때문에 나중에 나오게 된다. 스택에 데이터를 집어넣는 연산을 push, 데이터를 제거하는 연산을 pop이라 한다.



**나. overflow 오류**

스택 구조를 다룰 때에는 overflow오류를 조심해야 한다. Overflow란, 정해진 스택의 용량을 초과해서 데이터가 들어올 때 데이터가 넘치는 것을 말한다. 재귀 호출을 여러 번 하다 보면, 후술할 콜 스택(Call Stack)이 많이 쌓여 스택 오버 플로우(Stack Overflow) 오류가 나서 런타임 에러를 발생시키며 프로그램이 종료된다.

**제7장. 스택(Stack) 영역**

**가. 지역 변수**

지역 변수는 스택 영역에 저장되는 데이터 중 하나이다. 함수가 호출되면 함수의 매개변수 등과 함께 스택 영역에 저장이 되고, 함수가 종료되면 스택 영역에서 빠져나온다.

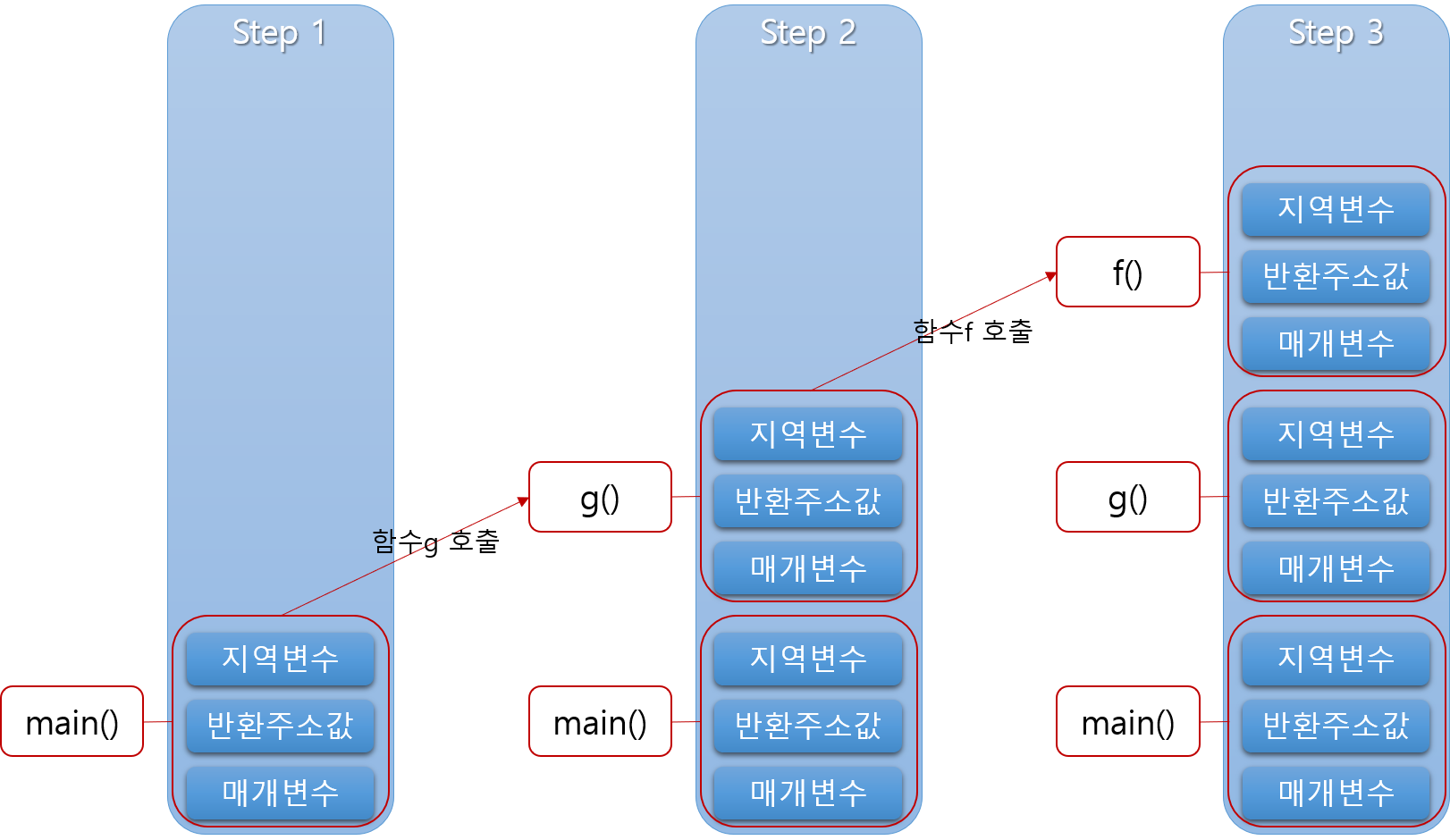
스택 영역에서는 메모리의 높은 주소에서 낮은 주소의 방향으로 데이터가 저장이 된다.

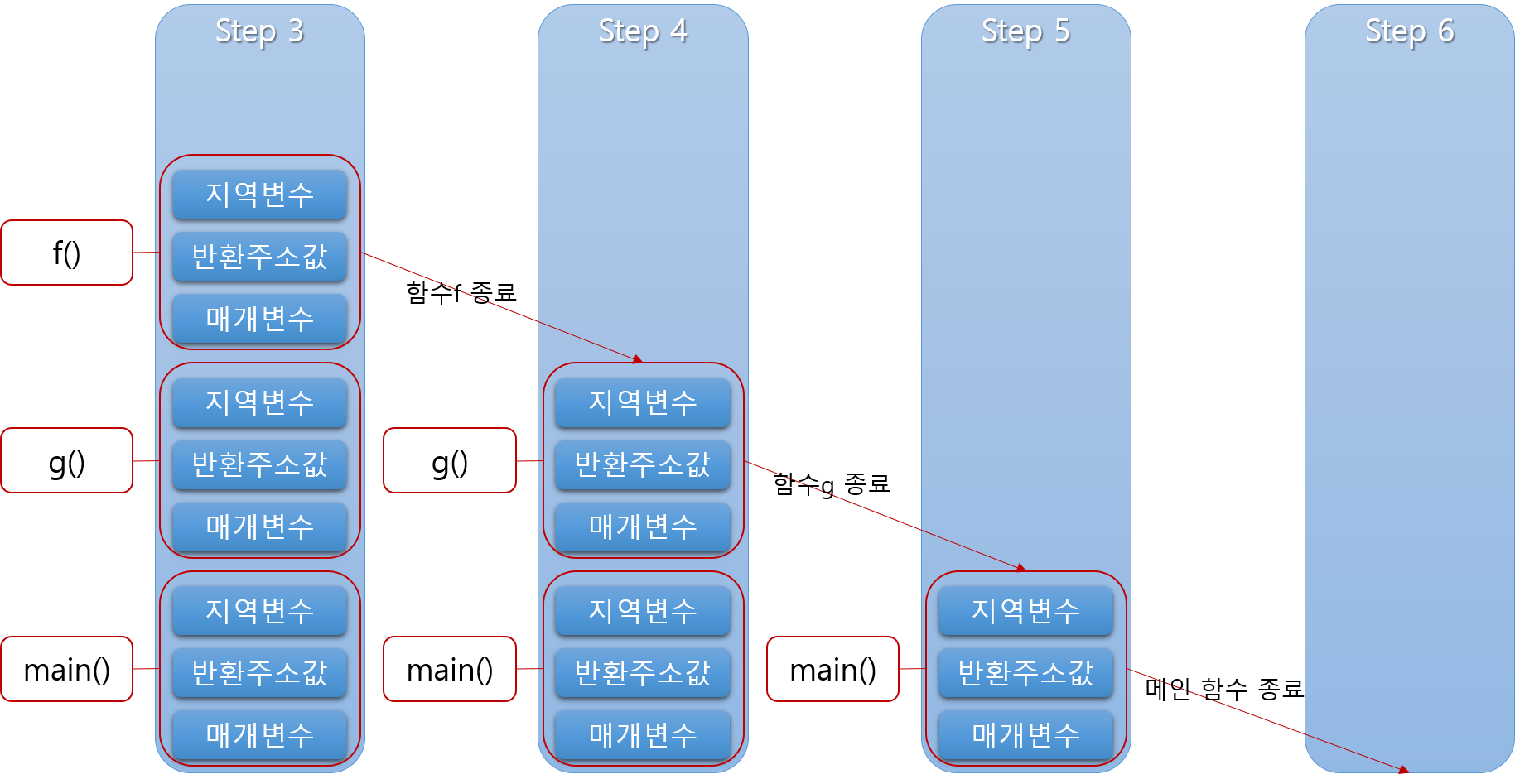
**나. 스택 프레임(Stack Frame)**

스택 영역은 함수 호출과 관계가 있는 각종 변수 가 저장되는 영역이다. 함수가 호출될 때 스택에 함수의 매개변수, 호출이 끝난 뒤 돌아갈 반환 주소 값, 해당 함수에서 선언이 된 지역 변수가 저장이 된다. 이렇게 스택 영역에 저장되는 함수의 호출 정보를 스택 프레임이라 한다.

|  |
| --- |
| void f () {  //something  }  void g () {  f ();  //something  }  int main () {  g ();  return 0;  } |

아래 그림은 위 예제 코드에 의한 스택 프레임의 변화이다.





**제8장. 힙(Heap) 영역**

**가. 힙 영역의 용도**

메모리의 힙 영역은 사용자가 직접 관리하는 영역이다. 힙 영역은 사용자가 필요할 때 할당해서 쓰고 다 쓰면 해제된다. 힙 영역은 스택 영역과 다르게 낮은 주소에서 높은 주소로 메모리가 할당된다.

**나. C언어의 메모리 동적 할당**

c언어에서 메모리를 동적으로 할당할 때에는 malloc 함수를 사용하고, 원형은 다음과 같이 생겼다.

|  |
| --- |
| #include <stdlib.h>  void \*malloc (size\_t size); |

이 함수는 할당 받을 바이트 수를 인수로 받은 뒤, 힙 영역에서 메모리 크기에 맞고, 아직 할당되지 않은 적당한 블록을 찾아서 할당을 한다. 그렇게 찾은 블록의 첫 번째 바이트 주소를 반환한다. 반환 값이 주소 값이기 때문에 해당 공간에 직접 접근할 때에는 포인터를 사용해야 한다.

**결론**

이 보고서를 작성하면서 기존에 알고 있던 내용들을 더욱 확실하고 자세히 알게 되었고, BSS 영역 같은 알지 못했던 것을 알게 되어서 뿌듯함을 느꼈다.

참고 문헌

<http://tcpschool.com/c/c_pointer_intro>

<http://tcpschool.com/c/c_memory_structure>

<http://tcpschool.com/c/c_memory_stackframe>

<https://blog.perfectacle.com/2017/01/25/c-ref-003/>

<http://sfixer.tistory.com/entry/%EB%A9%94%EB%AA%A8%EB%A6%AC-%EC%98%81%EC%97%ADcode-data-stack-heap>

<https://kldp.org/node/122255>