# 중간 레벨 과제

## 간단한 메모리 관리자 - 설계 보고서


| 제출일: 2022.11.09 | 전공 | 컴퓨터공학과 | | 10조 |
|---|---|---|---|---|
| 어드벤처디자인_02 | 학번 | 20190602 | 20190622 | 20190788 |
| 담당 교수: 김경수 교수님 | 이름 | 설진영 | 손현락 | 이근탁 |

# 목 차

# Ⅰ. 사용자 요구사항 분석 및 명세

### 현재 개발하는 프로그램에 대한 사용자의 요구사항, 기능별 분류

c언어의 malloc() free() 함수와 동일하지는 않지만 비슷하게 작동하는 mymalloc(), myfree()함수를 만들어 main 함수에서 주어진 예와 같이 함수를 호출하여 수행되는 결과를 출력하고, 주어진 예시 이외의 다른 시나리오 하나를 더 만들어 결과를 출력하라.

주어진 요구사항에서 필요한 기능은 다음과 같다.

- 메모리 초기화, 메모리 할당, 메모리 해제, 결과 출력.

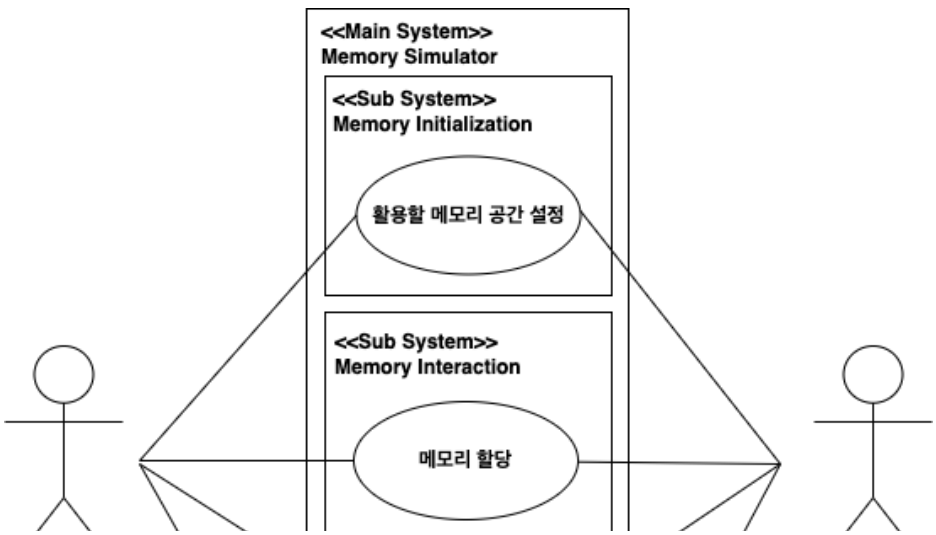
### 각 기능별로 분류한 사용자 요구사항

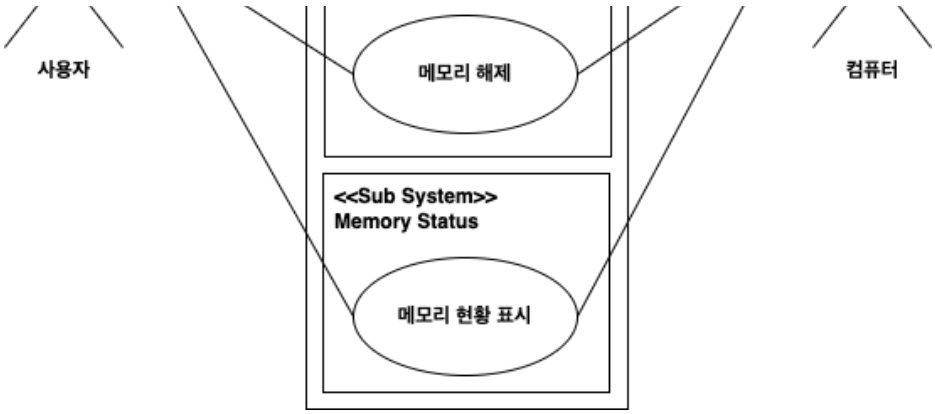
1. 메모리 초기화: 해당 프로그램에서 사용될 가상 힙 영역의 크기를 입력받아 초기화한다.
2. 메모리 할당: mymalloc() 함수를 실행하면 할당할 공간이 있는지를 검사한다. 만약 할당할 수 있는 공간이 존재한다면 덩어리를 잘라 내어 시작 주소를 반환한다.
3. 메모리 해제: myfree() 함수를 실행하면 할당된 공간을 반환한다. 반환된 공간이 다른 공간과 연속되는 경우이면 하나의 큰 공간으로 합친다.
4. 결과 출력: 변수 available 덩어리 들의 전체적인 모습을 순차적으로 출력한다. available 리스트는 할당하고 남은 덩어리(chunk), 즉 가용 공간에 대한 정보를 저장한다. 가용 공간은 크기와 위치가 제각각이고 할당과 반환이 반복됨에 따라 정보 내용이 변하므로 연결리스트로 저장한다.

### 각 사용자 요구사항에 대한 입/출력 및 요구사항들 간의 관계

1. 메모리 초기화: 프로그램에서 사용할 힙영역의 크기를 입력받음.
2. 메모리 할당: int MyMalloc(char allocated_name, int allocated_size) - 할당할 메모리 용량과 메모리 공간의 이름을 입력으로 받아 메모리 할당 작업 후 할당된 공간에 대한 시작 주소를 반환
3. 메모리 해제: void MyFree(char free_name) - 해제할 메모리의 이름을 입력받아 메모리 해제 작업을 진행, 반환 값은 없음.
4. 결과 출력: 저장공간을 입력으로 받아 해당 저장공간 내의 각종 정보를 시각적으로 표현한다.

### 사용자 요구사항과 대응되는 핵심 기능 도식화






# Ⅱ. 프로그램 전체 구조 설계

### 프로그램의 전체 기능과 목표

본 프로그램에서는 C언어에서의 메모리 동적할당 및 해제함수인 malloc과 free를 모방한 MyMalloc 및 MyFree 함수를 구현한다. 그에 대한 결과 및 할당되어 있는 메모리 상태를 보여주어 malloc과 free함수가 어떤 식으로 동작하는지 보여주는 것을 목표로 한다.

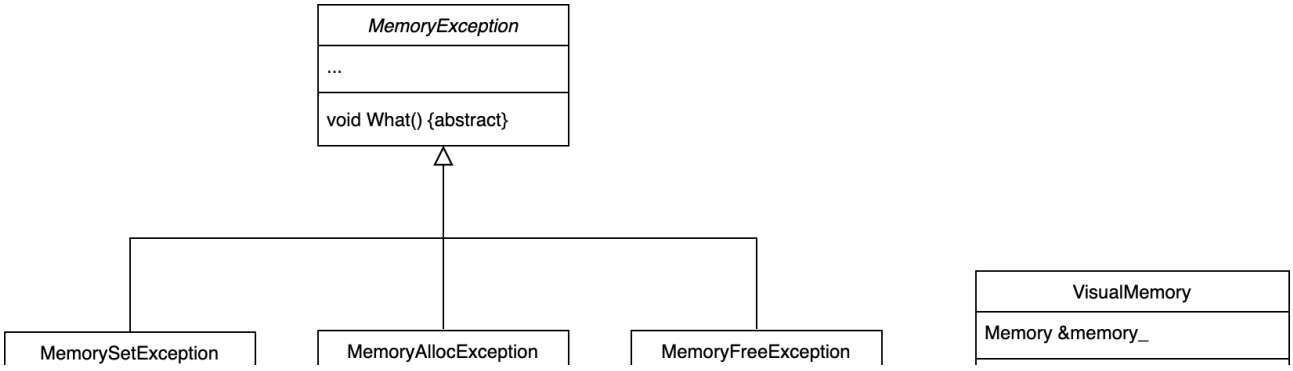
### 프로그램의 실행 및 제약조건

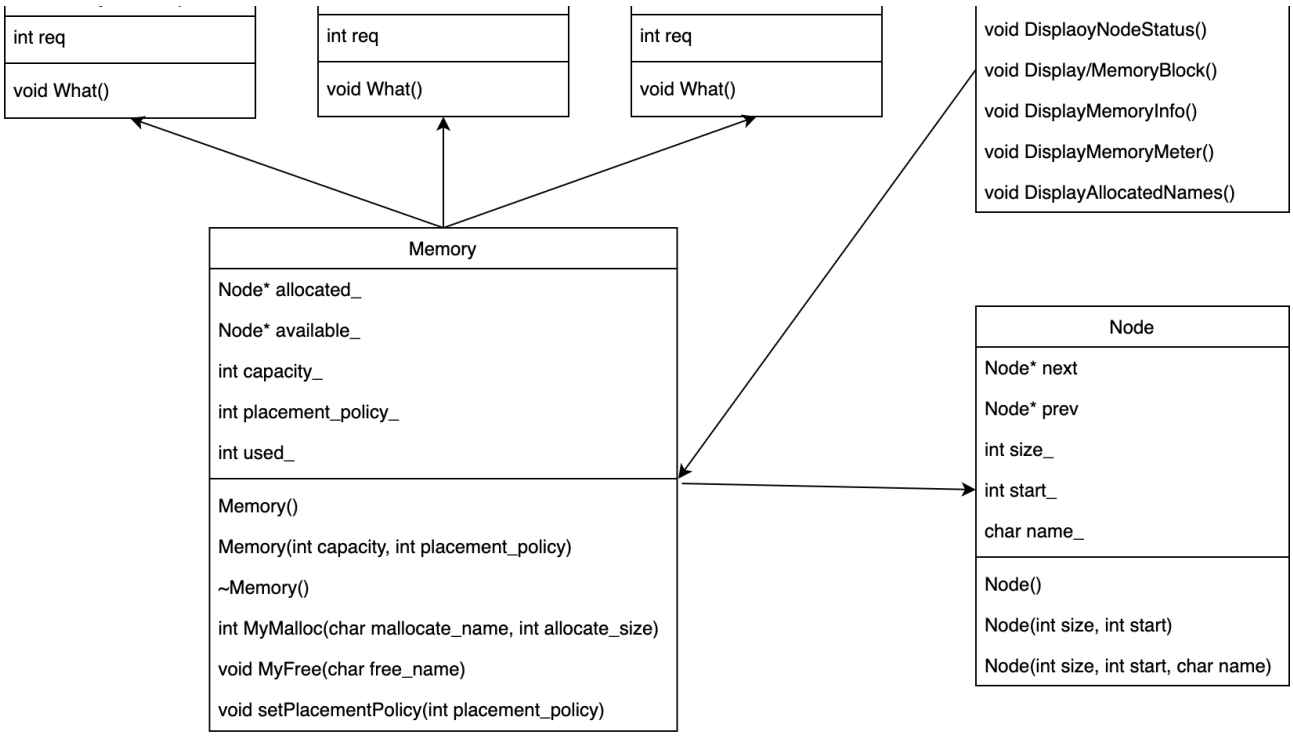
C언어에서 동적할당 함수인 malloc을 사용하게 되면 실제로는 우리가 정적으로 선언한 변수 들과 달리 프로그램 실행 중에 가변적으로 할당하게 된다. 이러한 정적 변수 들은 메모리의 Stack 영역에, 동적할당된 변수 들은 Heap 영역에 할당하게 되는데 ASLR이라는 메모리 보호 기법에 의해 프로그램이 실행될 때마다 Stack 영역과 Heap 영역의 주소가 바뀌어 실제로 Heap 영역에 접근하는 것은 불가능하다. 그러므로 해당 프로그램에서는 정적으로 메모리 클래스를 만들고 이를 Heap 영역으로 간주하여 이 안에서 메모리가 할당되고 해제되는 상황을 시뮬레이션을 통해 보여준다.

### 프로그램 구성하는 서브루틴 또는 모듈의 리스트

본 프로그램을 구현하기 위해서 객체지향 프로그래밍을 채택하였기에 크게는 Memory 클래스, Node 클래스, MemoryVisual 클래스, MemoryException의 4개 클래스, 더 세부적으로 나누어 MemoryException에서 상속을 받은 MemorySetException, MemoryAllocException, MemoryFreeException 3개의 클래스로 도합 7개의 클래스가 있다.

### 프로그램의 통합 구조를 보여주는 다이어그램






# Ⅲ. 모듈 설계 및 모듈 간 제어흐름 설계

## 모듈의 사용 목적과 기능

앞서 밝힌 바와 같이 본 프로그램은 크게는 4개, 세부적으로는 7개의 모듈로 구성하였다.

1. Memory 클래스 : 사용자에게서 메모리 총량을 입력받아 이를 Heap 영역의 크기로 보며 모방하는 클래스
2. Node 구조체: Memory모듈 내의 연결리스트를 구성하는데 사용되는 구조체
3. VisualMemory 클래스 : 현재 메모리의 상태를 나타내는 여러 정보들을 시각적으로 보여주는 메소드들을 가지는 클래스. 사용자 인터페이스 구성에 사용
4. MemoryException 클래스 : 해당 프로그램 실행 중에 나타날 수 있는 각종 예외를 처리하는 클래스
   - MemorySetException 클래스 : Heap 영역 메모리 초기화 시 발생 예외 처리 클래스
   - MemoryAllocException 클래스 : 메모리를 할당할 때 나타나는 예외 처리 클래스
   - MemoryFreeException 클래스 : 할당된 메모리를 해제하려 할 때 발생하는 예외 처리 클래스

## 모듈의 필드 및 메소드 상세

다음은 위에 나열된 각 모듈 들의 상세 내용이다.

1. Memory 클래스

   관리할 메모리를 나타내는 클래스이며, 멤버변수로는 할당된 chunk 들의 Head인 allocated_, 가용한 hole 들의 head인 available_, 변하지 않는 메모리의 총량인 capacity_, 할당되어 있는 메모리의 양을 나타내는 used_, 각 배치 정책을 나타내는 플래그 변수인 placement_policy_가 있다. 또한 메소드로는 사용자에게서 입력받아 메모리를 초기화하는 생성자, 각각 malloc과 free를 모방한 MyMalloc과 MyFree, 그리고 원하는 배치 정책을 변경시키는 setPlacementPolicy가 있다.

2. Node 구조체

   Memory 클래스의 allocated_와 available_에서 할당되었거나 할당이 가능한 시작점과 그로부터의 사이즈를 저장할 수 있는 구조체이다. 멤버 변수로는 해당 Chunk 혹은 Hole의 시작점을 가르키는 start_, 그로부터의 크기를 나타내는 size_, 그리고 메모리가 할당되었을 때 각 메모리의 구분을 위한 name_ 변수가 있다. 즉, 할당할 수 있는 공간을 나타내는 available_에 붙어있는 Node 클래스에서의 name_은 없어도 되기에 Null로 설정한다. 마지막으로 양방향 연결리스트를 구현하기 위한 Node 포인터인 prev, next가 있다. 메소드로는 Chunk Node를 생성하기 위한(메모리 간 구분이 필요한) 생성자 하나와, Hole Node를 생성하기 위한 생성자 하나가 있다.

3. MemoryVisual 클래스

   사용자의 입력으로 초기화한 Memory 클래스에서 할당, 해제가 이루어지는 사이사이에서 메모리의 현 상황을 보여주는 클래스이다. Memory 객체의 getter를 통해 해당 객체의 맴버 변수에 접근한다. 멤버 변수로는 보여줄 메모리의 참조를 받아올 Memory_가 있다. 메소드로는 Chunk와 Hole 연결리스트의 상황을 보여주는 DisplayNodeStatus, 메모리의 단편화 상황을 보여주는 DisplayMemoryBlock, 사용자가 초기화한 메모리 총량, 지금까지 할당된 총량 등 여러 가지 메모리 정보를 Text로 보여주는 DisplayMemoryInfo, 할당된 메모리를 게이지로 확인할 수 있는 DisplayMemoryMeter, 마지막으로 메모리 해제를 하기 위해서 이름이 필요하므로 이때까지 할당된 메모리들의 이름을 모두 보여주는 DisplayAllocatedNames가 있다.

4. MemoryException 클래스

   메모리를 각각 초기화, 할당, 해제하는 과정에서 발생하는 예외를 정의하기 위한 추상 클래스이며, 각 상황마다 MemorySetException, MemoryMallocException, MemoryFreeException라는 객체로 정의된다. 메모리를 초기화하는 상황에서는 양수가 아닌 숫자를 입력받았을 때, 할당하는 상황에서는 단편화된 상황에서 가용한 최대 메모리보다 큰 메모리를 할당하려 할 때나 크기가 양수가 아닐 때, 해제하는 상황에서는 해제하려는 이름의 메모리가 할당된 메모리에 없을 때의 예외를 처리해준다.

**모듈의 실행 및 제약조건**

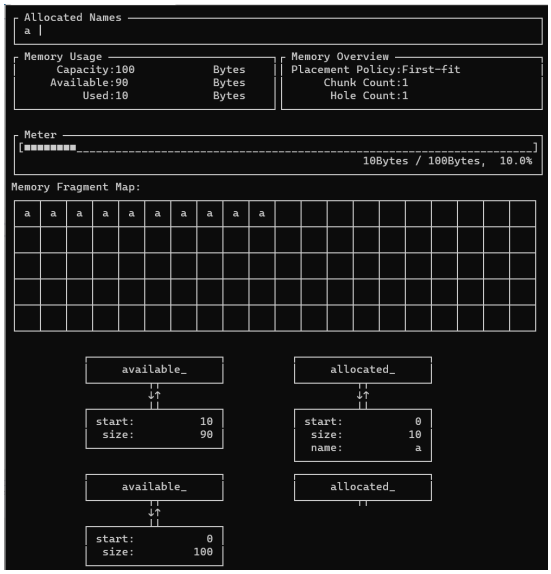
본 프로그램의 주목적은 메모리가 동적으로 할당되거나 해제될 때의 동작을 모방하고 메모리의 상황을 보여주는 프로그램이라 했었다. 이 중에서 메모리의 상황을 콘솔 환경에서 보여주기 위해서 따로 Visual Memory라는 클래스를 작성하였다. 이러한 콘솔 창에서 정렬된 모습으로 보이기 위해 적절히 출력문을 작성하였는데 이 출력문이 메모리의 이름이 한 글자일 때를 가정하여 작성하였기에 메모리를 할당할 때의 이름을 한 글자로 입력받도록 설정해두었다. 또한 이 프로그램은 같은 이름으로 메모리를 다시 할당받을 경우, 별다른 오류가 발생하지 않는다. 프로그램 작동 과정상 문제가 없고, 이 경우 가장 마지막에 할당받은 메모리가 해제 요청 시 가장 먼저 해제된다. 하지만 메모리 공간을 구분하기 어렵고 오해의 여지가 있으므로 같은 이름으로 할당하는 것을 권장하지 않는다.

### 모듈의 실행 예제

```
#include "Memory.h"
#include "VisualMemory.h"
#include <iostream>

int main() {
    Memory memory(100);
    VisualMemory view(memory);
    memory.MyMalloc('a', 10);
    view.DisplayAllocatedNames();
    view.DisplayMemoryInfo();
    view.DisplayMemoryMeter();
    view.DisplayMemoryBlock();
    view.DisplayNodeStatus();
    memory.MyFree('a');
    view.DisplayNodeStatus();
}
```

< 사용 예>

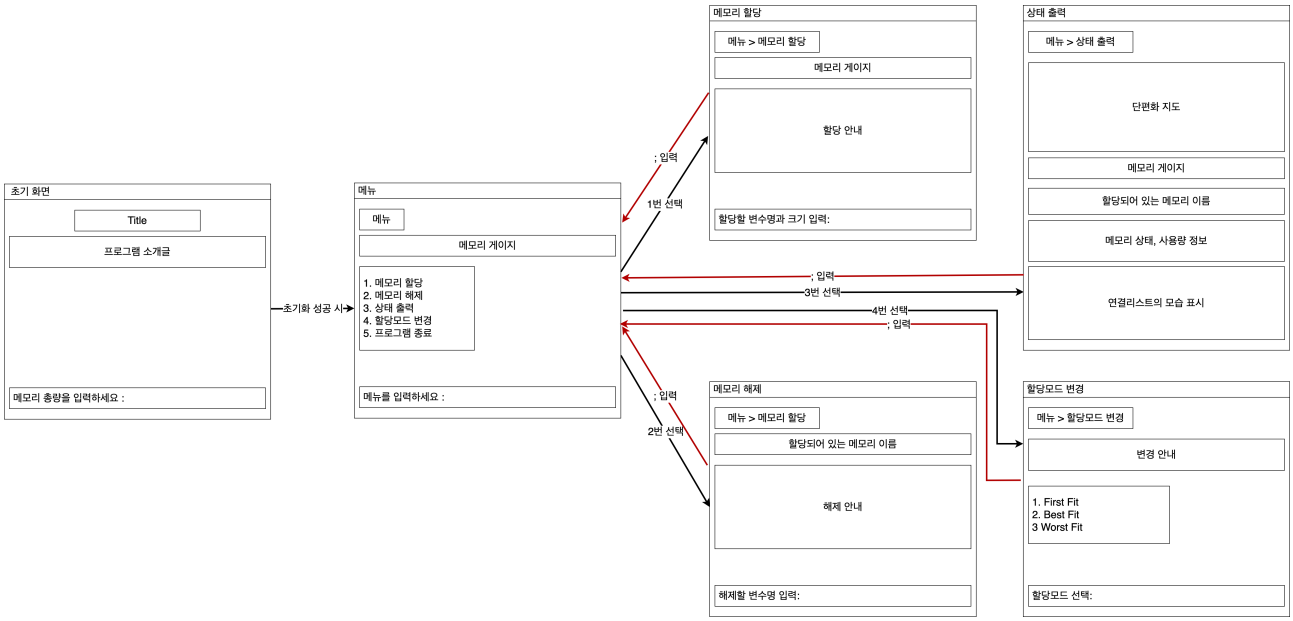



< 결과 화면 >

모듈의 사용 방법과 그 결과는 다음과 같다. VisualMemory에 속해있는 맴버함수 이외의 것들은 별도의 출력이 없다.

## Ⅳ. 사용자 인터페이스 설계

### 인터페이스의 전체적인 구조 도식화 및 설명




기본적으로 사용자와 windows 터미널에서 단순 텍스트로 상호작용 하나, 사용자가 큰 불편 없이 해당 프로그램을 사용할 수 있도록 사용성을 최대한 개선하였다.

- 처음 시작했을 때의 UI

  (이름, 제작 목적)

  - 시뮬레이션에 사용할 총 공간의 크기를 입력받음

- 메뉴화면
  (메모리 현황)
  1. 메모리 할당
  2. 메모리 해제
  3. 현황 출력
  4. 할당모드 변경
  5. 프로그램 종료
- 메모리 할당 메뉴 화면
  (메모리 현황)
  - 메모리 할당할 곳의 이름, 크기 입력받음
- 메모리 해제 메뉴 화면
  (할당되어있는 메모리의 이름)
  - 해제할 메모리의 이름을 입력받음
- 현황 출력
  (전체적인 메모리 이용 현황)
  (실제로 프로그램이 동작할 때 available, allocated 리스트의 현황)

**사용자 인터페이스의 세부 루틴**

처음에 시작했을 때 제목, 프로그램의 제작 목적을 출력하여 밝힌다. 그다음 시뮬레이션에서 사용할 가상의 '힙 영역'의 총량을 입력하라는 문구를 출력한다.

사용자로부터 총량을 입력받은 이후에 본격적인 메인화면이 나타난다. 우선 메인화면이라는 것을 내비게이션으로 명시한 후, 현재 메모리 현황을 출력하고, 메모 할당, 메모리 해제, 메모리 현황 출력, 할당모드 변경, 프로그램 종료 메뉴를 출력한 이후, 메뉴를 입력하라는 문구를 출력한다.

메모리 할당 메뉴 ; 우선 '메뉴의 메모리 할당'라고 내비게이션으로 명시한 후, 현재 메모리 현황을 출력하여 사용자에게 하여금 앞으로 할당할 수 있는 용량의 크기를 확인하게 한다. 그 후, c언어에서의 malloc 함수의 원형을 보여주고, 그중에서 변수명과 크기를 지정해달라는 문구를 출력한다.

메모리 해제 메뉴 : 우선 '메뉴의 메모리 해제'라고 내비게이션으로 명시한 후, c언어에서의 free 함수의 원형을 보여주고, 그중에서 해제할 메모리의 변수명을 지정하라는 문구를 출력한다.

현황 출력 메뉴 : 우선 '메뉴의 상태 출력'이라고 내비게이션으로 명시한 후, 전체적인 공간에서 할당된 메모리 들의 크기와 위치를 일종의 GUI로 표시한다. 그 후 메모리 공간을 미터기로 표시하고, 할당된 메모리 들의 변수명 들을 출력하고, 메모리의 사용량에 대한 정보, 현재 할당모드에 대한 정보, 실제로 이 프로그램에서의 node 상태에 대한 정보를 출력한다.

할당모드 변경 메뉴 : 우선 '메뉴의 할당모드 변경'이라고 내비게이션으로 명시한 후, 현재 할당모드의 상황을 출력하고, 변경할 수 있는 모드를 출력한 이후, 선택하라는 문구를 출력한다.

**사용자 인터페이스와 직접적으로 통신하는 내부 모듈**

전반적인 메뉴의 큰 틀은 main 함수에서 진행된다. main 함수에서 시뮬레이션에 사용할 전체적인 메모리 크기를 입력받게 하고, 메뉴를 출력하고, 선택된 메뉴에 대해 추가적인 사항을 출력한다.

메모리의 정보와 관련된 출력은 VisualMemory 클래스가 관장한다. VisualMemroy 클래스 객체는 메모리의 총 정보를 담고 있는 Memory 클래스 객체에서 남은 메모리와 할당된 메모리를 가리키는 available, allocated 리스트를 가져와서 출력 작업을 진행한다. 출력 작업은 메모리 공간을 게이지 형태로 출력, 메모

리 단편화 지도 출력, 할당된 메모리 들의 변수명 출력, 메모리 사용량과 할당모드에 대한 정보를 출력, 실제 available, allocated 연결리스트 현황을 출력하는 메소드가 정의되어있다. main 함수는 상황에 맞게 VisualMemory 클래스 객체의 메소드를 호출하여 필요한 정보를 출력하도록 되어있다.

## 참고 문헌

UML 강의자료

어드벤쳐디자인 중간레벨 문제