객체지향 프로그래밍 1차 과제

문제

컴퓨터의 주기억장치를 모델링하는 클래스 Ram을 구현하려고 한다. Ram 클래스는 데이터가 기록될 메모리 공간과 크기 정보를 가지고, 주어진 주소에 데이터를 기록하고(write), 주어진 주소로부터 데이터를 읽어 온다(read). ram 클래스는 다음과 같이 선언된다.

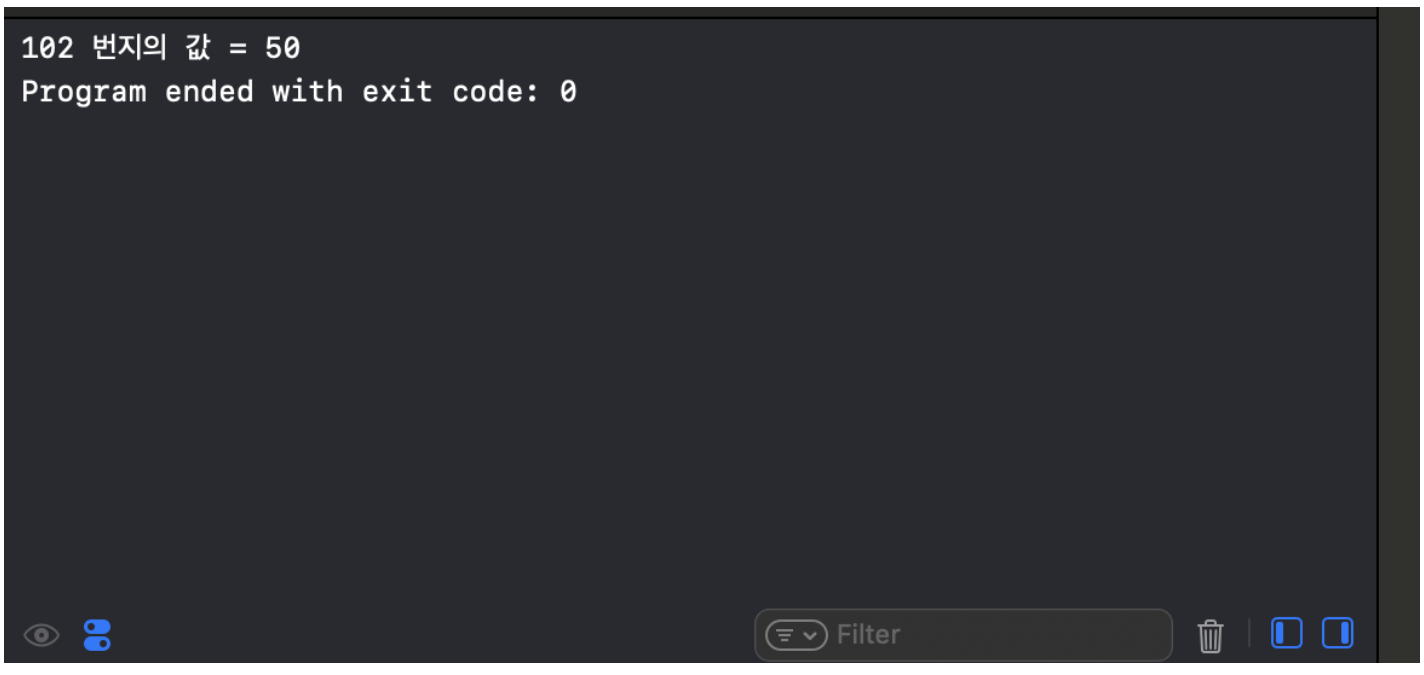
```
class Ram{
    char mem[100*1024]
    int size;
public:
    Ram();
    ~Ram();
    char read(int address);
    void write(int address, char value);
};
```

다음 main() 함수는 100번지에 20을 저장하고, 101번지에 30을 저장한 후, 100번지와 101번지의 값을 읽고 더하여 102번지에 저장하는 코드이다.

```
int main() {
    Ram ram;
    ram.write(100, 20);
    ram.write(101, 30);
    char res = ram.read(100) + ram.read(101)
    ram.write(102, res);
    cout << "102 번지의 값 = " << (int)ram.read(102) << endl;
}
```

실행결과를 참고하여 Ram.h, Ram.cpp, main.cpp로 헤더파일과 cpp 파일을 분리하여 프로그램을 완성해라.

소스 수행 결과 화면 캡쳐



소스 구현 설명

문제 정의

RAM를 모델링하는 클래스를 구현해서, 특정 주소에 데이터를 저장하고 읽어오는 기능을 제공하는 프로그램을 작성하는 것이 이번 과제입니다.

메모리 주소를 기반으로 데이터의 저장과 검색을 수행을 한다. 라고 생각하면 될 것 같습니다.

문제 해결 방법

1. **RAM 클래스 설계**
   - RAM을 모델링하기 위해 `mem` 배열과 `size` 변수를 멤버 변수로 가지는 `Ram` 클래스를 정의합니다. `mem` = 실제 메모리 공간

     `size` = 배열의 크기를 저장
2. **데이터 접근 메서드 구현**
   - `write` method를 구현해서 주어진 주소에 데이터를 저장하는 기능을 추가.
   - `read` method를 구현해서 주어진 주소에서 데이터를 읽어오는 기능을 추가.
3. **예외 처리**
   - 유효하지 않은 주소에 접근하려 할 경우, 예외를 발생시켜 잘못된 접근을 방지합니다.

알고리즘 설명

Ram.h (헤더)

```
#ifndef RAM_H
#define RAM_H

class Ram {
    char mem[100 * 1024]; // 메모리 공간
    int size; // 메모리 크기

public:
    Ram(); // 생성자
    ~Ram(); // 소멸자
    char read(int address); // 주어진 주소에서 데이터 읽기
    void write(int address, char value); // 주어진 주소에 데이터 쓰기
};

#endif // RAM_H
```

클래스의 정의와 메서드 선언을 포함해서 다른 파일에서 이 클래스를 사용할 수 있도록 처리합니다.

Ram.cpp(구현)

```
#include "Ram.h"
#include <stdexcept> // 예외 처리

// 생성자
Ram::Ram() {
    size = sizeof(mem); // mem의 크기로 size 초기화
}
```

```
// 소멸자
Ram::~Ram() {}

// 주어진 주소에서 데이터 읽기
char Ram::read(int address) {
    if (address < 0 || address >= size) {
        // 유효하지 않은 주소 접근
        throw std::out_of_range("Invalid!");
    }
    return mem[address];
}

// 주어진 주소에 데이터 쓰기
void Ram::write(int address, char value) {
    if (address < 0 || address >= size) {
        // 유효하지 않은 주소 접근
        throw std::out_of_range("Invalid!");
    }
    mem[address] = value; // 메모리에 값 쓰기
}
```

클래스의 메서드 구현을 포함하여 메모리 읽기 및 쓰기 로직을 정의했습니다.

main.cpp (메인)

```
#include <iostream>
#include "Ram.h"

using namespace std;

int main() {
    Ram ram; // Ram 객체 생성
    ram.write(100, 20); // 100번지에 20을 저장
    ram.write(101, 30); // 101번지에 30을 저장
    char res = ram.read(100) + ram.read(101); // 100번지와 101번지의 값을 읽고 더하기
    ram.write(102, res); // 결과를 102번지에 저장
    cout << "102 번지의 값 = " << (int)ram.read(102) << endl; // 102번지의 값 출력

    return 0;
}
```

Ram Class를 사용해서 데이터 저장 및 읽기 작업을 수행하는 역할을 합니다.

흐름 설명

`main.cpp`에서 `Ram.h` 헤더 파일을 포함하여 `Ram` 클래스를 정의하고 사용합니다.

`main()` 함수 내에서 `Ram` 클래스의 인스턴스인 `ram` 객체를 생성하게 되고, 이 시점에서 `Ram`의 생성자가 호출되어 메모리 공간을 나타내는 `mem` 배열의 크기가 `size` 변수에 저장됩니다.

이후 `ram.write(100, 20);` 호출을 통해 지정된 주소인 100번지에 20을 기록하게 됩니다. 이어서 `ram.write(101, 30);` 호출로 101번지에 30을 기록합니다.

다음으로 `char res = ram.read(100) + ram.read(101);` 구문을 통해 100번지와 101번지의 값을 읽어와 더한 결과를 `res`에 저장합니다. 이때 `read` 메서드는 해당 주소의 값을 반환합니다.

그 후 `ram.write(102, res);` 호출을 통해 계산된 결과인 50을 102번지에 저장하게 됩니다.

마지막으로 `cout`을 통해 102번지의 값을 출력하고, 이 과정에서도 `read` 메서드는 102번지의 값을 반환하여 출력됩니다.

프로그램은 `main()` 함수가 종료되면서 함께 종료되고, 이 시점에 `Ram` 객체의 소멸자가 호출됩니다.