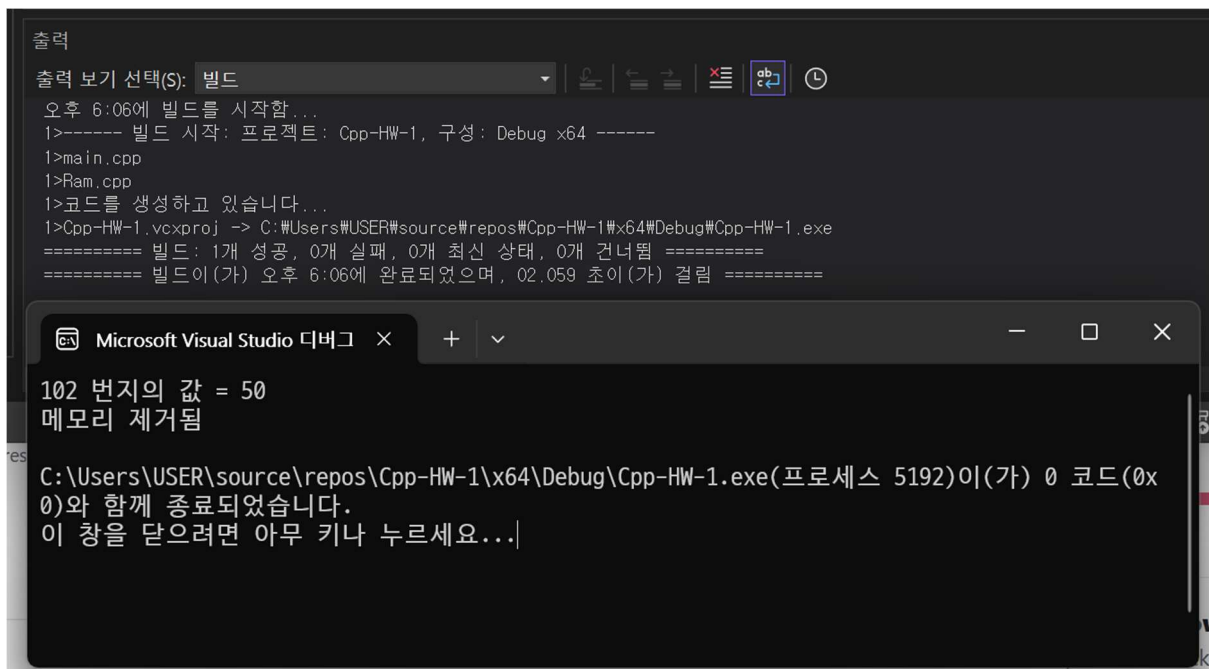


# 객체지향프로그래밍 과제#1

소프트웨어전공 202004006 고창석

소프트웨어전공 202284011 김연재

- 소스 수행 결과 화면 캡처



The screenshot shows two windows from Visual Studio. The top window is the 'Output' window, set to 'Build' (출력 보기 선택(S): 빌드). It displays the following text:

```
출력
출력 보기 선택(S): 빌드
오류 6:06에 빌드를 시작함...
1>----- 빌드 시작: 프로젝트: Cpp-HW-1, 구성: Debug x64 -----
1>main.cpp
1>Ram.cpp
1>코드를 생성하고 있습니다...
1>Cpp-HW-1.vcxproj -> C:\Users\USER\source\repos\Cpp-HW-1\x64\Debug\Cpp-HW-1.exe
===== 빌드: 1개 성공, 0개 실패, 0개 최신 상태, 0개 건너뛸 =====
===== 빌드가 (가) 오류 6:06에 완료되었으며, 02.059 초이 (가) 걸림 =====
```

The bottom window is the 'Microsoft Visual Studio 디버그' (Debug Console) window. It shows the following output:

```
102 번지의 값 = 50
메모리 제거됨

C:\Users\USER\source\repos\Cpp-HW-1\x64\Debug\Cpp-HW-1.exe(프로세스 5192)이(가) 0 코드(0x0)와 함께 종료되었습니다.
이 창을 닫으려면 아무 키나 누르세요...
```

- 소스 구현 설명

- 문제 정의

Ram 클래스는 데이터가 기록될 메모리 공간과 크기 정보를 가지고, 주어진 주소에 데이터를 기록하고, 주어진 주소로부터 데이터를 읽어온다.

main 함수는 Ram을 활용하여 address를 읽고 value를 저장한다.

주어진 코드를 활용해 헤더파일과 cpp 파일을 분리하고 정상 작동하는 프로그램을 완성하는 것이 문제의 목적이다.

■ 문제 해결 방법, 문제를 해결한 키 아이디어 또는 알고리즘 설명

Ram 클래스를 Ram.h로 분리하고, 코드가 작동하기 위한 로직을 Ram.cpp에 구현했다.

- Ram.h

```
1  #ifndef RAM_H
2  #define RAM_H
3
4  class Ram {
5      char mem[100 * 1024];
6      int size;
7  public:
8      Ram();
9      ~Ram();
10     char read(int address);
11     void write(int address, char value);
12 };
13
14 #endif
```

클래스에 read, write 메서드가 있으며 write 메서드는 주소와 데이터 값을 저장하는 기능을 하고 read 메서드로 데이터 값을 불러올 수 있다.

- Ram.cpp

```
1  #include "Ram.h"
2  #include <iostream>
3  using namespace std;
4
5  Ram::Ram() {
6      size = 100 * 1024;
7      for (int i = 0; i < size; i++) {
8          mem[i] = 0;
9      }
10 }
11
12 Ram::~Ram() {
13     cout << "메모리 제거됨" << endl;
14 }
15
16 char Ram::read(int address) {
17     if (address >= 0 && address < size) {
18         return mem[address];
19     }
20     else {
21         cout << "잘못된 주소 접근" << endl;
22         return 0;
23     }
24 }
25
26 void Ram::write(int address, char value) {
27     if (address >= 0 && address < size) {
28         mem[address] = value;
29     }
30     else {
31         cout << "잘못된 주소 접근" << endl;
32     }
33 }
```

mem 배열을 초기화 하기 위해서 Ram 생성자를 만들고 for 반복문을 사용하여 배열 속 모든 값을 초기화 하였다.

read, write 메서드에서 주소에 접근할때 잘못된 주소로 접근 할 경우를 확인하기 위해 조건문을 따로 구성하였다.

- Main.cpp

```
1  ✓ #include "Ram.h"
2  | #include <iostream>
3  | using namespace std;
4
5  ✓ int main() {
6  |     Ram ram;
7  |     ram.write(100, 20);
8  |     ram.write(101, 30);
9  |     char res = ram.read(100) + ram.read(101);
10 |     ram.write(102, res);
11 |     cout << "102 번지의 값 = " << (int)ram.read(102) << endl;
12 |
13 |     return 0;
14 | }
```

외부에 정의된 Ram 클래스를 불러와 Ram 클래스의 객체를 생성하고 활용한다.

write와 read 메서드를 사용하여 작동하고, 결과물을 출력한다.

#### ■ 아이디어 평가

김연재 - 잘못된 주소 접근을 방지하기 위한 조건문을 추가한 점은 신뢰성을 높이는 중요한 부분이라고 생각한다.

고창석 - 배열 초기화를 위해 생성자에서 반복문을 사용한 것은 효율적이며 적절하다고 생각한다.