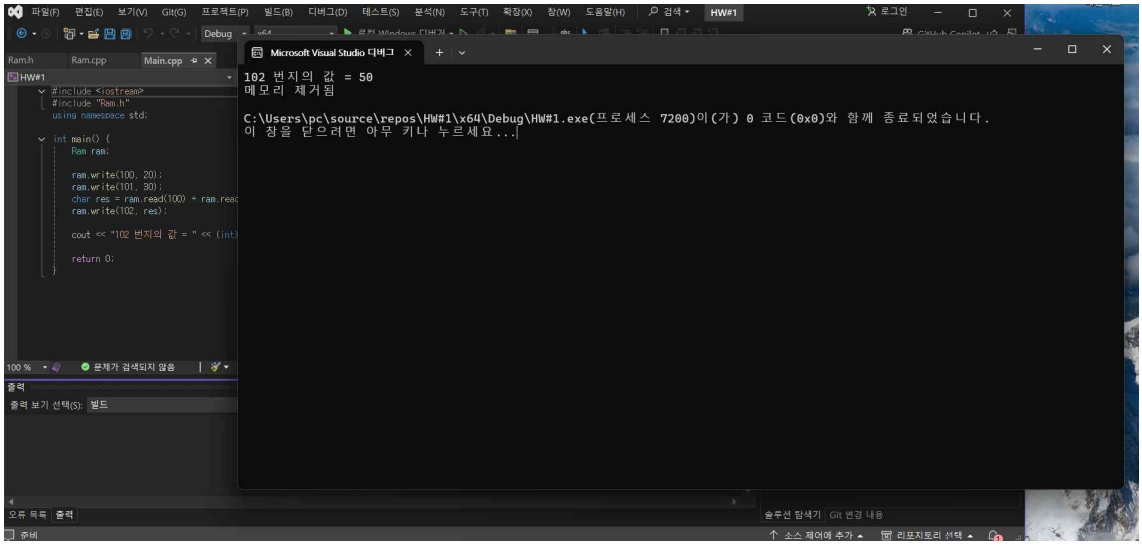
1) 소스코드 수행결과 화면 캡쳐



2) 소스 구현 설명

2-1) 문제 정의:
메모리의 특정 주소에 값을 쓰고, 그 값을 읽어오는 메모리 관리 시스템을 구현하는 것이 목표이다. 이 프로그램은 메모리에서 데이터를 저장하고, 불러오며, 이를 조작할 수 있도록 해야 한다.

2-2) 문제 해결 방법:

해결 방법은 다음 세 가지 파일로 구성된다.

Ram.h: Ram 클래스를 선언하며, 100KB 크기의 메모리를 갖는다. 이 클래스는 메모리 주소에서 데이터를 읽고 쓸 수 있는 메서드를 제공한다.

Ram.cpp:

Ram 클래스의 메서드를 구현한 파일이다.
생성자: 메모리의 모든 주소를 0으로 초기화한다.
소멸자: 메모리가 해제될 때 메시지를 출력한다.
read() 메서드: 메모리 주소를 받아 그 위치에 저장된 값을 반환하며, 주소가 유효하지 않으면 0을 반환한다.
write() 메서드: 특정 주소에 값을 저장하며, 주소가 유효한 범위일 때만 동작한다.

Main.cpp: Ram 클래스의 사용 예시를 보여주는 파일이다. 두 개의 값을 메모리에 저장하고, 그 값을 더한 후 새로운 주소에 저장한 뒤 결과를 출력한다.

3) 아이디어 평가:

메모리를 배열로 시뮬레이션하는 아이디어는 작은 규모의 메모리 관리 작업에 적합하다. 기본적인 읽기 및 쓰기 작업을 처리하며, 주소의 유효성 검사로 메모리 접근의 안정성을 보장한다. 이러한 구현은 간단하면서도 메모리 조작을 이해하는 데 유용하다.

4) 문제를 해결한 키 아이디어 또는 알고리즘 설명:
핵심 아이디어는 고정 크기의 배열을 사용해 RAM을 시뮬레이션하는 것이다. Ram 클래스는 메모리 관리 논리를 캡슐화하며, 사용자는 간단한 읽기 및 쓰기 작업을 통해 메모리와 상호작용할 수 있다. 이 클래스는 메모리 접근의 경계를 제어해 불필요한 에러를 방지한다. 프로그램은 두 값을 더해 새로운 메모리 주소에 결과를 저장하고 이를 출력하는 간단한 계산을 수행한다.