객체프로그래밍

202204103 ict융합공학부 이승훈
202204023 ict융합공학부 김보민

[5주차 과제(교재 p156 12번 문제)]

풀이)
ㄴ class 가 들어간 헤더파일 생성하기
ㄴ main 함수가 들어간 main.cpp 생성하기
ㄴ public에 있는 생성자로 Ram.cpp 생성하기
ㄴ Ram.cpp에서 public의 선언부 보고 구현부 작성하기
1. size 멤버 변수에 mem 배열의 크기 할당
2. mem 배열 초기화
3. 메시지 출력
4. adress에 있는 mem 배열 값 반환
5. adress에 있는 value값 mem에 저장

//보민 , ->승훈
[Ram.h(헤더파일 만들기)]

```
#ifndef RAM_H
#define RAM_H

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

#endif
```

[main.cpp(main함수 들어간 main.cpp생성하기)]

```
#include<iostream>
using namespace std;

#include "Ram.h"

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 << "102번지의 값=" << (int)ram.read(102) << endl;
}
```

[Ram.cpp(수정 전)]

```
#include<iostream>
using namespace std;

#include "Ram.h"

Ram::Ram() {
	size = 100 * 1024;
	mem[] = 0; -> 여기서 왜 오류가 나지? 배열 초기화를 어떻게 시켜야할까?
			// C++에서는 배열을 초기화 하려면 대괄호 안에 크기를 지정하고
				중괄호를 사용해서 초기값을 할당해줘.
			-> 그럼 mem[100 * 1024]={ }; 로 쓰면 되나 ?
}			// 맞아. 근데 여기서 데이터 타입이 정의되어 있지 않으니까
				mem의 데이터 타입인 char까지 추가해주면 될 거 같아.
Ram::~Ram() {
	cout << "메모리 제거됨"<< endl;

}

Ram::read(int address) {	->여기서는 왜 오류가 나지?
				// 여기도 데이터 타입이 없어서 그래 데이터 타입을 추가
				해주면 될 거 같아.
	return mem[address];
}
void Ram::write(int address, char value) {
	mem[address] = value;

}
```

[Ram.cpp(수정 1회)]

```
#include<iostream>
using namespace std;

#include "Ram.h"

Ram::Ram() {
	size = 100 * 1024;
	char mem[100*1024] = { };

}
Ram::~Ram() {
	cout << "메모리 제거됨"<< endl;

}
char Ram::read(int address) {
	return mem[address];
}
void Ram::write(int address, char value) {
	mem[address] = value;

}

// 실행 결과는 책 그대로 나오긴 하는데 char mem[100*1024] = { }; 이 부분이 신경쓰여
   여기서 mem은 Ram의 로컬 변수로 선언되어 있으니까 배열이 생성자를 빠져나가면 메모리에서 해제 돼. 그럼 다른 멤버 함수에서 우리는 이 배열을 못 쓰게 되지.
-> 그럼 mem을 Ram 클래스의 멤버 변수로 선언해서 초기화 시켜주면 되지 않을까 ?
//응 그걸 어떻게 해야 할지 잘 모르겠어.
-> c++에서 배열을 초기화 시키는 방법 중에 for문을 사용하는 방법이 있어.
   for (int i = 0; i < size; i++) {
	mem[i] = 0;
    }
   이 for문 또한 mem의 배열을 초기화 시켜주는 방법이야. 0부터 size-1까지 배열의 각 요소를 0으로 초기화 시켜주고 각각을 초기화 시켜줘. 그리고 마지막 mem의 배열의 i번째 요소에도 0을 할당해주지
// for (int i = 0; i <= size; i++) {
    }  그럼 이렇게 써도 같은 의미 아니야?
-> 이렇게 쓰게 되면 size값도 루프의 마지막 반복에 포함 돼. 이렇게 되면 배열의 인덱스 범위를 벗어난 size 번째 요소에 접근을 시도하게 되니까 오류를 일으킬 수 있어. 그래서 그런 방법은 안 쓰는게 좋아.
```

[Ram.cpp(수정 2회, 최종)]

```
#include<iostream>
using namespace std;

#include "Ram.h"

Ram::Ram() {
	size = 100 * 1024;
	for (int i = 0; i < size; ++i)
		mem[i] = 0;
}
Ram::~Ram() {
	cout << "메모리 제거됨"<<endl;
}
char Ram::read(int address) {
	return mem[address];
}
void Ram::write(int address, char value) {
	mem[address] = value;
}
```