객체지향프로그래밍 정리노트 (Week3_HWork3)

202004029 김정호, 202004040 노성민

Q1. 컴퓨터의 주기억장치를 모델링 하는 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]; // 100KB 메모리. 한 번지는 한바이트이므로 char 타입 사용
        int size;
public:
        Ram(); // mem 배열을 0으로 초기화하고 size를 100*1024로 초기화
        ~Ram(); // "메모리 제거됨" 문자열 출력
        char read(int address); // address 주소의 메모리 바이트 리턴
        void write(int address, char value); // address 주소에 한 바이트로 value 저장
};
```

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

```
102 번지의 값 = 50
메모리 제거됨
```

실행 결과를 참고하여 Ram.h, Ram.cpp, main.cpp로 헤더 파일과 cpp 파일을 분리하여 프로그램을 완성하라.

정호: 우선 Ram 클래스를 보면 char형 배열 mem이 있고 int형 변수 size 그리고 Ram 생성지, 소멸자, char형 read 함수, void형 write 함수가 있어 우선 생성자를 작성해보자.

성민: 생성자에서 mem배열은 0으로 초기화하고 size를 100*1024로 초기화하라고 나와 있는데 배열은 반복문을 통해 모든 값을 0으로 초기화해주면 될 것 같아.

```
Ram::Ram() {
   for (int i = 0; i < 100 * 1024; i++) {
      mem[i] = 0;
```

```
    }
    size = 100 * 1024;
}
```

정호: 소멸자는 문자열만 출력되게 하면 될 것 같아. 그리고 read 함수는 address의 메모리 값을 리턴해주면 되니까 return mem[address]; 를 적어주면 될거야.

```
Ram::~Ram() {
    cout << "메모리 제거됨" << endl;
}
char Ram::read(int address) {
    return mem[address];
}
```

성민: write 함수는 address 주소에 value를 저장해야 하니 mem[address] = value; 해주면 되고 반환값은 없으니 return은 따로 안 해줘도 돼.

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

정호: 그러면 클래스의 선언된 코드들을 다 작성하였으니 문제가 요구한 대로 Ram.h, Ram.cpp, main.cpp로 분리해서 코드를 작성하자. 우선 Ram.h 파일은 우선 헤드 파일에 파일을 하나 만들고 조건 컴파일 문의 상수는 다른 조건 컴파일 상수와의 충돌을 피하고자 클래스 명으로 하고 중복 include 오류를 방지하기 위해 #ifndef , #endif 문도 추가해서 작성해주자.

```
#ifndef RAM_H
#define RAM_H

#endif // !RAM_H

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

성민: 이제 Ram.cpp 파일을 작성해보자. 헤더파일 Ram.h를 include 해서 사용하고 우리가 생각했던 코드를 적으면 될 것 같아.

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

```
#include "Ram.h"

Ram::Ram() {
	for (int i = 0; i < 100 * 1024; i++) {
		mem[i] = 0;
	}
	size = 100 * 1024;
}

char Ram::read(int address) {
	return mem[address];
}

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

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

정호: 이제 main 함수를 실행하면 잘 동작 될 것 같아.