DAY2
과제 1번
보고서

2023741024
로봇학부 박건후

목차:

(1) 헤더 파일

```
#pragma once
#include <iostream>
namespace hw1{
class manager{
    int cnt;
    int* ptr;
    int index;


    public:
    void start();
    void get_number(int a);
    void repeat();
    void print();
    manager();
    ~manager();
};
}
```

헤더파일에서, hw1이라는 네임스페이스를 만들고 그 안에 manager라는 클래스를 만들었습니다. 멤버변수로, cnt는 입력받을 수의 개수, ptr은 그 안에 멤버변수로 사용자로부터 입력 받은 정수값을 저장할 배열을 가리킬 포인터, 그리고 index는 정수값들을 입력받을 때 그 때 몇 번째 인덱스의 값을 받고 있는지 저장하는 변수입니다. index는 get_number메서드를 사용하면서 필요하게 되어 추가하게 되었습니다. 문제에서는 모든 변수와 함수를 클래스 내부에서 선언하고 사용하라고 되어 있었기에 처음에 몇 개의 원소를 할당받을 것인지 값을 받고, 그 만큼 동적할당을 하는 메서드입니다. 그 후에 정수형 데이터를 연속으로 입력받는데, 그것을 수행하는 메서드가 repeat입니다. 그 repeat을 할 때 get_number라는 메서드를 입력받을 때마다 호출하게 했습니다. 그렇게 만든 이유는 각각의 메서드의 기능을 철저히 분리하여 나중에 다른 메서드에서도 쉽게 바로 기능을 사용할 수 있게 하기 위해 이처럼 설계한 것입니다. 이에 따라 get_number를 호출할 때마다 몇 번째 인덱스인지 저장하는 변수가 필요했고 이를 index라는 내부 멤버변수를 만들게 되었습니다. 그리고 print()라는 함수는 받은 값을 처리하고 출력하는 역할을 하게 만들었습니다.

(2) 구현 파일

```cpp
#include "hw1.hpp"
using namespace hw1;
void manager::start(){
        std::cout<<"몇 개의 원소를 할당하겠습니까? : ";
        std::cin>>  cnt;
        if(!std::cin || cnt<1){
            std::cout<<"잘못된 입력입니다. 다시 입력하세요"<<std::endl;
            std::cin.clear(); //오류 상태 초기화
            std::cin.ignore(1000, '\n'); //버퍼에 들어간 문자 무시
            manager::start();
        }
        else ptr = new int[cnt];
    }
void manager:: get_number(int a){
       ptr[index] = a;
       this->index++;
}
```

위의 헤더 파일에서 선언을 하고 이를 구현하는 것을 다른 cpp파일을 만들어 구현하였습니다. 클래스도 어떻게 보면 네임스페이스처럼 경계 역할을 하고 그 상태에서 hw1 네임스페이스로 감쌌기 때문에 hw1::을 또 써줘야 할 필요성이 있기에 using namespace hw1;을 사용하여 구현코드를 작성할 때 편의성을 갖추려 했습니다.

start메서드에서 할당할 원소의 개수를 물어보고 값을 멤버변수인 cnt에 저장을 하였습니다. 이 때 원소의 개수가 1 미만이면 메모리를 할당하지 않게 되며 문자를 입력하거나 다른 자료형을 넣는 것을 대비하여 !std::cin을 사용하여 다른 문자 입력에 대한 예외처리를 하였습니다. cin객체에는 입력이 제대로 되지 않았을 때 내부의 플래그 멤버변수가 바뀌게 되고, 바뀌었을 때 더 이상 입력을 받지 않습니다. 또한 플래그 변수값이 바뀌었을 때 !연산자를 cin에 사용하면 값이 true가 되며 if문으로 예외처리를 할 수 있습니다. 하지만 잘못된 입력이 되었을 경우 다시 입력을 받아야 하기에 cin의 상태를 정상 상태로 돌려줄 clear()메서드를 사용하였고, 또한 cin의 입력스트림 버퍼에 입력하면서 엔터 문자열이 들어가기에 ignore메서드를 사용하여 입력 버퍼를 비워주었습니다. 그리고 처음부터 다시 입력을 받아야 하므로 처음에는 while문을 사용하여 입력을 다시 받게 할까 생각했었지만 이번엔 재귀함수를 사용하였습니다. 절차적으로 line by line으로 들어가기에 재귀함수를 호출하기 전에 cin 버퍼를 정상상태로 만들고 재귀함수를 호출하고 다시 cin으로 입력을 받기에 정상적으로 진행되고 while문보다 더욱 코드가 간단해지기 때문에 재귀함수를 사용하였습니다. 여기서 예외가 발생했을 때 동적할당을 하는 것을 막기 위해 else 문으로 철저히 분리하였습니다.

```
void manager:: get_number(int a){
        ptr[index] = a;
        this->index++;
}
```

```
void manager:: repeat(){
        int input=0;
      for(int i=0; i< cnt; i++){
        std::cout<<"정수형 데이터 입력: ";
        std::cin>>  input;
        if(!std::cin){
            std::cout<<"잘못된 입력입니다. 다시 입력하세요"<<std::endl;
            std::cin.clear(); //오류 상태 초기화
            std::cin.ignore(1000, '\n'); //버퍼에 들어간 문자 무시
            i--;
            continue;
        }
        get_number(input);
      }
}
```

get_number는 repeat메서드에서 내부적으로 호출합니다. 원래는 get_number메서드를 따로 만들지 않고 repeat메서드에서 처리할 수 있지만 코드의 재활용을 위해 기능을 철저히 분리하여 만들었습니다. repeat메서드는 정수형 데이터 입력을 연속적으로 받는 것을 담당하도록 만들었습니다. 우선 get_number라는 값을 받는 메서드를 만들고 그 이후에 받는 것을 반복하는 repeat메서드를 만든 것입니다. 이 때도 사용자로부터 값을 받기 때문에 이에 대한 예외처리를 진행하였습니다. 그래서 if문으로 !std::cin로 문자입력, 엔터입력 같은 예외를 막으려고 했습니다. 이 때 다시 입력받게 하기 위해 while문을 따로 쓸 수도 있겠지만 for문으로 반복적으로 받는 환경을 활용하고자 값을 실제로 저장하는 get_number메서드 앞에 if문을 사용하고, if문 내부에서 continue를 하여 잘못된 입력일 시 값을 저장하지 못하게 했습니다. 잘못된 입력을 받았을 때 continue를 하면 해당 i값에서 입력값을 저장하지 않고 넘어가는 것이기에(증감식 i++) 사용자가 할당하고자 하는 개수만큼 정확히 저장하게 하기 위해 continue앞에 i--를 하여 그 i번째에 값을 저장하는 것을 넘기지 않도록 만든 것입니다.

```
void manager:: print(){
        int min= ptr[0];
        int max= ptr[0];
        int sum =0;
        for(int i=0; i< cnt; i++){
            sum+= ptr[i];
            if(min>ptr[i]){
                min=ptr[i];
            }
            if(max < ptr[i]){
                max= ptr[i];
            }
        }
        double avg= (double)sum/ cnt;
        std::cout<<"최대값: "<< max <<std::endl<<"최소값: "<< min<< std::
}
manager::manager():cnt(0), ptr(NULL), index(0){};
manager::~manager(){
        delete[] ptr;
}
```

repeat을 통해 정수값을 입력받고 저장하는 것을 완료했다면 최대 최소를 구하고 값을 가공하고 출력해야 합니다. 이를 담당하는 메서드를 print()라는 메서드로 구현하였습니다. 최대, 최소를 기존에 하던 방식으로 반복문을 통해 max, min값을 업데이트하였고 전체합, 평균을 구하고 이를 출력하는 함수입니다.

print()아래의 메서드는 생성자와 소멸자입니다. 생성자를 만들 때는 멤버변수에 쓰레기값이 들어가지 않도록 초기화하는 정도로만 만들었습니다. start메서드, repeat메서드를 통해 cnt, ptr, index변수값이 설정되기 때문에 그렇습니다. 초기화할 때는 멤버 초기화 리스트를 사용하여 멤버변수를 생성하고 초기화했습니다. 소멸자에서는 메모리 해제를 하는 코드를 작성했습니다. 메모리를 클래스 내부에서 start메서드에서 ptr멤버변수를 통해 할당하므로 이에 대한 메모리 해제를 객체가 소멸할 때 메모리를 해제하고 소멸해야 하기 때문입니다.

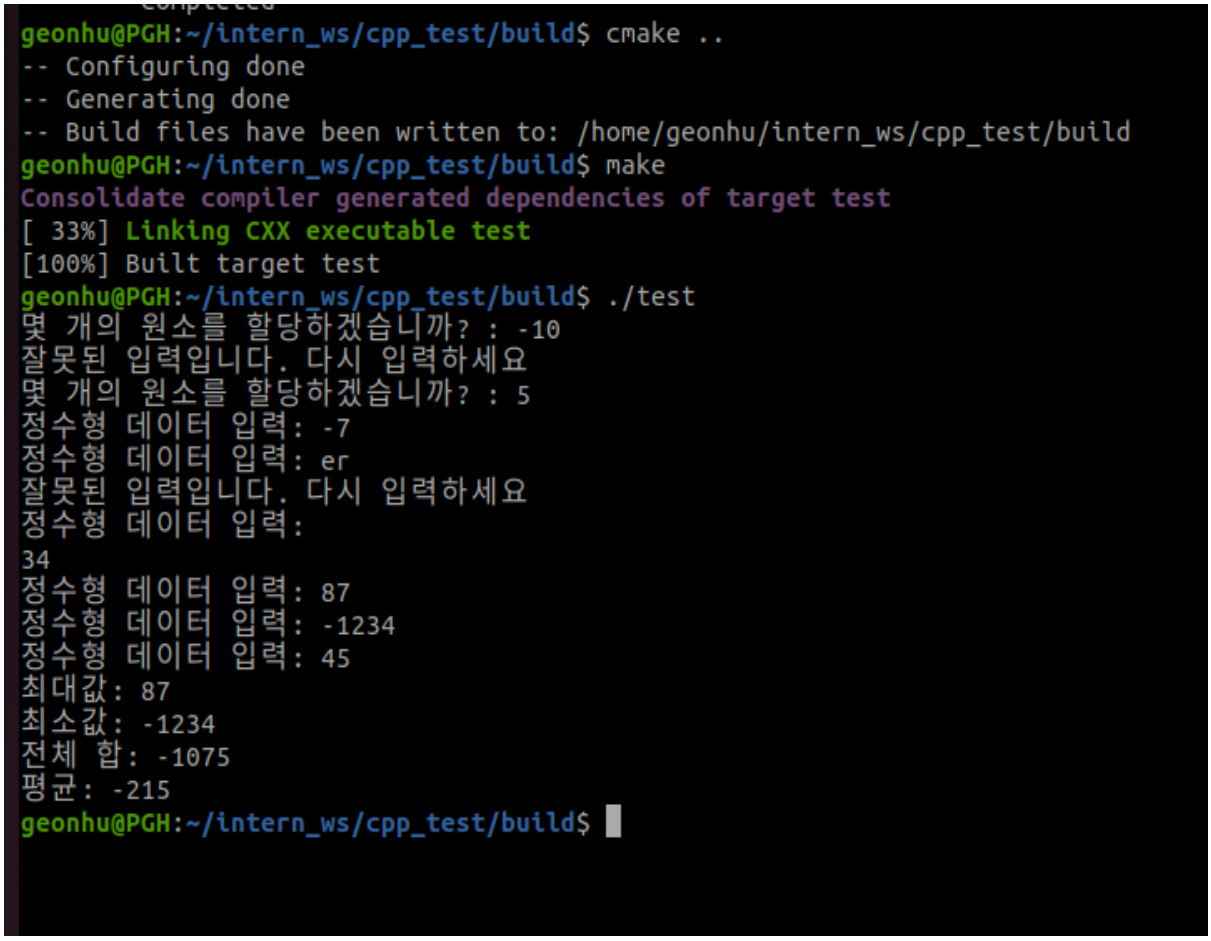
(3) main 파일

```
#include "hw1.hpp"

using namespace hw1;
int main(){
    manager key;
    key.start();
    key.repeat();
    key.print();


    return 0;
}
```

main부입니다. hw1.hpp파일을 인클루드하고 hw1 네임스페이스 안에 있기에 hw1::을 사용하여 클래스에 접근해야 하지만 코드를 쉽게 간단히 하기 위해 using namespace를 사용하였습니다.

객체를 생성하고 순서에 따라 메서드를 호출하여 프로그램이 작동되게 하였습니다.

(4) 실행 사진


geonhu@PGH:~/intern_ws/cpp_test/build$ cmake ..
-- Configuring done
-- Generating done
-- Build files have been written to: /home/geonhu/intern_ws/cpp_test/build
geonhu@PGH:~/intern_ws/cpp_test/build$ make
Consolidate compiler generated dependencies of target test
[33%] Linking CXX executable test
[100%] Built target test
geonhu@PGH:~/intern_ws/cpp_test/build$ ./test
몇 개의 원소를 할당하겠습니까? : -10
잘못된 입력입니다. 다시 입력하세요
몇 개의 원소를 할당하겠습니까? : 5
정수형 데이터 입력: -7
정수형 데이터 입력: er
잘못된 입력입니다. 다시 입력하세요
정수형 데이터 입력:
34
정수형 데이터 입력: 87
정수형 데이터 입력: -1234
정수형 데이터 입력: 45
최대값: 87
최소값: -1234
전체 합: -1075
평균: -215
geonhu@PGH:~/intern_ws/cpp_test/build$