HW1 보고서

로빛 20기 인턴 2025407006 모시온

목차

hw1 프로젝트 개요

본 프로젝트는 C++ 개발 환경에서 Class 개념을 적극 활용하여 모든 사용 변수 및 함수를 Class 와 접목하여 Class 의 멤버 변수 및 멤버 함수로 정의하여 작성하는 데의 개발 배경을 통해 사용자의 지정 크기에 해당하는 배열을 동적 할당한 후 각각의 요소에 정수 값을 입력한 후 해당 동적 배열의 모든 요소를 비교한 후 최대값, 최소값, 전체 합과 평균을 구하는 코드입니다.

1. 알고리즘 설계

2-1. .hpp 설명

개요에서 설명한 바와 동일하게 해당 프로젝트는 C++의 구분되는 점인 Class 활용을 적극적으로 활용하여 코드에서 활용되는 모든 사용 변수 및 함수들을 Class 의 멤버 변수 및 멤버 함수로 정의하였습니다.

```
#include <iostream>
```

우선 헤더파일 내부에 멤버 변수가 있으며 이를 입출력하는 방식으로 코드가 동작되기 때문에 기본적으로 제공되는 변수 입출력을 cout, cin 을 활용해 구현할 수 있도록 내부에서 iostream 을 선언하였습니다.

```
namespace HW1{
```

또한 본격적으로 시작될 Git 의 활용에 대비할 수 있도록 여러 유저의 관여로 인해 변수명의 충돌에 대비할 수 있도록 namespace 를 해당 과제의 제목(순서)인 HW1 로 지정하여 사용하였습니다. 이는 현재 위 과제와 관련된 인원들이 관여를 시도할 시 현재 과제의 내용을 미리 사전에 숙지되어있는 상태가 전제되었기 때문에 모두가 한번에 이해할 수 있도록 HW1 으로 명명하였습니다.

```
class hw1 {
private:
    int n;
    double check_n;
    int* arr;
    int max;
    int min;
    int sum;
    double avg;

public:
    //생성자
    hw1() {
        n = 0;
        check_n = 0.0;
        arr = 0;
        max = 0;
        min = 0;
        sum = 0;
        avg = 0.0;
    }
```

왼쪽의 사진은 위 코드에서 사용되는 Class입니다.

코드 실행 시 본 코드에서는 사용자의 지정 원소 개수를 받은 후 동적 할당 및 각 원소 값을 직접적으로 입력할 수 있도록 하였습니다.

n와 check_n은 사용자의 입력이 정수가 아니거나 올바른 입력이 아닐 때를 감지할 수 있도록 변수를 이중으로 구성하였습니다.

arr은 원소 개수에 따라 동적으로 길이가 변하는 배열을 뜻하는 변수입니다.

각각의 변수 max, min, sum, avg는 최대값, 최소값,

전체 합, 평균값을 뜻합니다.

이때 사용자의 입력에 따라 연산과정 및 동적 할당 과정을 거치게 되는 변수들은 private 안에 종속될 수 있도록 구성하여 버그가 예방되도록 유도하였습니다.

```
//생성자
hw1() {
    n = 0;
    check_n = 0.0
    arr = 0;
    max = 0;
    min = 0;
    sum = 0;
    avg = 0.0;
}
//소멸자
~hw1() {
    delete arr;
}
```

Class 선언 시 사용자의 입력에 따라 일련의 지정 과정을 거치는 각각의 Class 멤버 변수들을 초기화할 수 있도록 생성자 및 소멸자 함수를 거쳐 초기 시작 시 초기화 및 완료 시 삭제될 수 있도록 하였습니다.

```
//원소 개수
void count() {
    std::cout << "몇 개의 원소를 할당하시겠습니까? : " << std::endl;
    std::cin >> check_n;
}
//입력 확인용 예외처리(일반 정수입력, 실수 입력, 음수 입력, 문자 입력)
bool check_cin() {
    if (std::cin.fail()) {
        return false;
    }
    if (check_n <= 1 || !check_n) {
        return false;
    }
    n = (int)(check_n);
    return true;
}
```

사용자의 입력이 올바른지 확인하는 함수입니다.
사용자의 원소 개수 할당을 물어본 후 사용자의 입력되는 값을 실수형으로 선언된 check_n 에 저장, 이후 해당 check_n 이 올바른 입력인지 판별하는 check_cin 함수를 구동하여 판별할 수 있도록 하였습니다.

우선 자료형부터 살필 수 있도록 하였습니다.
실수형으로 선언된 check_n 에 사용자가 임의로 특수문자를 포함한 문자를 입력할 시 C++ 내부에서 제공하는 .fail 을 활용하여 cin 을 통한 입력이 올바르지 않은 사실을 확인하며 참일시, 불리언으로 선언된 함수에서 불을 반환하여 해당 사실을 알 수 있도록 처리하였습니다.
하지만 아직 여러 문제가 남아있습니다.
바로 사용자가 1 이하의 실수 및 음수 또는 1 이상이나 정수가 아닌 양수를 입력했을 때의 상황입니다.

이를 예외처리할 수 있도록 하기 위하여 check_n 이 우선 "개수"로서 표현할 수 있는 수 이상인지 판별합니다. (1 개 -> O / 0.5 개 -> X / -1 개 -> X)
또한 제대로 check_ n 이 저장되지 않은 경우을 !check_n 을 판별함으로서 처리할 수 있도록 하였습니다.

이때, check_n 값이 4.1 또는 5.6 과 같이 1 이상의 정수가 아닌 양수인 경우 정수형으로 변환한 값을 n 에 저장하여 활용되도록 유도하였습니다.

(본래, 정수형으로 변환하여 저장된 n 과 check_n 을 비교하여 둘이 다를 시 n 으로 판별을 이행하려고 하였습니다. 하지만 본 과정은 n 을 다른 판별없이 정수형으로 저장하여 사용한다면 문제가 발생하지 않기 때문에 이를 생략하였습니다.)

따라서 사용자의 배열 길이 입력이 올바른 경우 참을 반환하여 이후 동적 할당을 행하도록 하였습니다.

```
//동적할당
void mallocing() {
    arr = new int[n];
}
//순서대로 값 대입
void insert_element() {
    for (int i = 0; i < n; i++) {
        std::cout << "정수형 데이터 입력: ";
        std::cin >> arr[i];
    }
}
```

동적 할당 부분입니다.
앞서 정수형 포인터 배열로 선언한 arr 을 new 를 통해 n 만큼 동적 할당하였으며, 배열의 길이는 n 이므로 for 반복문을 0 부터 n-1 번째까지 반복하여 동적 할당 배열 arr 의 요소에 각각의 사용자의 입력 값을 저장할 수 있도록 하였습니다.

하지만 문제가 있었습니다.
이렇게 되었을 경우 사용자의 입력이 정수형이 아닐 수도 있기 때문입니다.
실제로 실행 시 다음과 같은 오류가 발생합니다.

```
sion@sion-Laptop:~/intern_ws/hw1/build$ ./test
몇 개의 원소를 할당하시겠습니까? :
5
정수형 데이터 입력: a
잘못된 입력
정수형 데이터 입력: 잘못된 입력
정수형 데이터 입력: 잘못된 입력
정수형 데이터 입력: 잘못된 입력
정수형 데이터 입력: 잘못된 입력
최대값: 0
최소값: 0
전체합: 0
평균: 0
```

이를 해결하기 위하여 위 코드를 다음과 같이 수정하였습니다.

```cpp
void insert_element() {
    for (int i = 0; i < n; i) {
        std::cout << "정수형 데이터 입력: ";
        double check_arr;
        std::cin >> check_arr;
        if(std::cin.fail() || check_arr != (int)check_arr){
            std::cout << "잘못된 입력" << std::endl;
            std::cin.clear();
            std::cin.ignore(1000, '\n');
            continue;

        }
        else{
            arr[i] = (int)check_arr;
            i++;
        }
    }
}
```

앞서 사용한 방법과 동일하게 입력을 임시적으로 저장하는 변수 check_arr 을 활용하여 사용자가 check_arr 과 다른 자료형으로 입력하였을 때와 실수형으로 입력하게 되었을 때, 안내 메시지와 함께 입력되는 버퍼를 전부 초기화한후 덮어쓰기를 통해 쓰레기값이 관여할 수 없도록 하였습니다. 사용자의 정수형 입력이 정상인 경우 arr 배열에 정수형 형태로 저장할 수 있도록 하였으며, 이후 진행되는 i++를 통해 다음 인덱스로 옮기게 하였습니다.

실행 결과는 다음과 같습니다.

```
sion@sion-Laptop:~/intern_ws/hw1/build$ ./test
몇 개의 원소를 할당하시겠습니까? :
s
종료합니다
sion@sion-Laptop:~/intern_ws/hw1/build$ ./test
몇 개의 원소를 할당하시겠습니까? :
-2
종료합니다
sion@sion-Laptop:~/intern_ws/hw1/build$ ./test
몇 개의 원소를 할당하시겠습니까? :
4
정수형 데이터 입력: d
잘못된 입력
정수형 데이터 입력: 1
정수형 데이터 입력: 2
정수형 데이터 입력: 3
정수형 데이터 입력: 4
최대값: 4
최소값: 1
전체합: 10
평균: 2.5
```

2-2. .cpp 설명

```cpp
#include "hw1.hpp"
```

헤더파일을 불러옵니다.

main.cpp 에서 또한 기본 입출력을 사용하나, 앞서 불러온 hw1.hpp 내부에 포함된 iostream 을 통해 추가적인 호출없이도 사용이 가능합니다.

```
using namespace HW1;
```

hw1.hpp 의 충돌을 예방하기 위한 namespace 를 불러오며,

```
int main() {
    //단일 객체 생성
    hw1 element;

    element.count();
    //예외처리
    if (!element.check_cin()) {
        std::cout << "종료합니다" << std::endl;
        return 0;
    }
    //동적할당 -> 요소 삽입 -> 연산 -> 출력
    element.mallocing();
    element.insert_element();
    element.calculate();
    element.result();

    return 0;
}
```

이후 main 함수 문을 동작함으로서 예외처리 안내문구를 제외한 나머지 동작을 hw1.hpp 에 정의된 Class 의 멤버 함수를 호출하여 동작을 이룰 수 있도록 하였습니다.

2. 실행 결과

```
sion@sion-Laptop:~/intern_ws/hw1/build$ ./test
몇 개의 원소를 할당하시겠습니까? :
4
정수형 데이터 입력: d
잘못된 입력
정수형 데이터 입력: 1
정수형 데이터 입력: 2
정수형 데이터 입력: 3
정수형 데이터 입력: 4
최대값: 4
최소값: 1
전체합: 10
평균: 2.5
```