C++ 1일차 보고서

20기 인턴 송수민

1. hw1

- 실행 결과

```
ssm@miki:~/intern_work_space/day1/hw1/build$ ./test
몇 개의 원소를 할당하시겠습니까? : 5
정수형 데이터 입력:34
정수형 데이터 입력:2
정수형 데이터 입력:343
정수형 데이터 입력:5
정수형 데이터 입력:1
최댓값: 343
최솟값: 1
전체합: 385
평 균: 77
ssm@miki:~/intern_work_space/day1/hw1/build$ ./test
몇 개의 원소를 할당하시겠습니까? : f
잘못된 입력입니다. 정수를 입력해주세요.
몇 개의 원소를 할당하시겠습니까? : 2.4
정수를 입력해주세요.
몇 개의 원소를 할당하시겠습니까? : 2.f
정수형 데이터 입력:잘못된 입력입니다. 정수를 입력해주세요.
```

- 코드 분석

1) 헤더 파일

```
namespace hw1 {
    class calculation {
    private:
        int sum;
        float ave;
        int max;
        int min;
        int size;
        int *a;

    public:
        calculation(int size);
        ~calculation();

        void input();
        void compute();
        void display();
    };
}

void inputInt(const char* prompt, int* value);
```

계산을 하는 클래스와 정수형으로 숫자를 받는 함수를 선언하였고, 각 값들을 저장할 변수를 프라이빗으로 선언하였다.

2) 함수 정의 파일

```cpp
#include "hw1.hpp"
#include <iostream>
using namespace std;

namespace hw1 {
    calculation::calculation(int sz) : size(sz) {
        a = new int[size];
        sum = 0;
        ave = 0;
        max = 0;
        min = 0;
    }

    calculation::~calculation() {
        delete[] a;
    }

    void calculation::input() {
        for(int i=0; i<size; i++) {
            inputInt("정수형 데이터 입력:", &a[i]);
        }
    }

    void calculation::compute() {
        sum = 0;
        max = a[0];
        min = a[0];
        for(int i=0; i<size; i++) {
            sum += a[i];
            if(a[i] > max) max = a[i];
            if(a[i] < min) min = a[i];
        }
        ave = (float)sum / size;
    }
```

```
    void calculation::display() {
        cout << "최댓값: " << max << endl;
        cout << "최솟값: " << min << endl;
        cout << "전체합: " << sum << endl;
        cout << "평 균: " << ave << endl;
    }
}

void inputInt(const char* prompt, int* value) {
    while (true) {
        cout << prompt;
        double temp;
        cin >> temp;

        if (cin.fail()) {
            cin.clear();
            char c;
            while (cin.get(c) && c != '₩n');
            cout << "잘못된 입력입니다. 정수를 입력해주세요." << endl;
            continue;
        }

        // 소수 체크
        if (temp != static_cast<int>(temp)) {
            cout << "정수를 입력해주세요." << endl;
            continue;
        }

        *value = static_cast<int>(temp);
        break;
    }
}
```

헤더 파일에 있는 함수들을 정의하는 파일이다.
먼저 사용자가 입력한 개수만큼 정수를 받아오기 위해 동적할당을 통해서 배열을 만들었고 종료 시에 동적할당이 해제되도록 하였다.
input()에서는 for 문을 통해서 값을 사용자가 원하는 만큼 입력할 수 있도록 하였다.
compute()에서는 각 값들을 더하고 최댓값과 최솟값을 계산하였다.
display()에서는 계산한 결과를 출력할 수 있도록 하였다.

inputInt()는 입력받을 때 출력할 문장과 저장할 변수를 포인터로 받는 함수로 예외처리를 위해 만든 함수이다.
문장 출력 이후 값을 double로 받아오고 저장이 안될 시 문자 데이터 이므로 다시 입력버퍼 지운 뒤 재입력을 유도하고 int로 형 변환 후 값이 달라진 경우 소수이므로 재입력을 유도하였다. 정수일 시 int로 형 변환 후 저장하였다.

3) main

```cpp
#include <iostream>
#include "hw1.hpp"
using namespace std;

int main() {
    int s;
    inputInt("몇 개의 원소를 할당하시겠습니까? : ", &s);
    if (s < 1) {
        cout << "입력한 숫자가 너무 작습니다." << endl;
        return 0;
    }
    hw1::calculation calc(s);
    calc.input();
    calc.compute();
    calc.display();
    return 0;
}
```

정수를 입력 받고 1 미만일 시 오류 메시지 출력 후 리턴 시키고 아닐 시 각각의 함수를 호출하였다.

2. hw2

- 실행 결과

```
ssm@miki:~/intern_work_space/day1/hw2/build$ ./test
좌표의 최솟값을 입력하세요 : 0
좌표의 최댓값을 입력하세요 : 10
생성할 점의 개수를 입력하세요 : 5
0번 점: (5, 6)
1번 점: (10, 3)
2번 점: (3, 8)
3번 점: (7, 4)
4번 점: (0, 7)

=== 거리 계산 결과 ===
최소 거리: 2.82843 (0번 점 (5, 6) ↔2번 점 (3, 8))
최대 거리: 10.7703 (1번 점 (10, 3) ↔4번 점 (0, 7))
```

- 코드 분석

1) 헤더파일

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

struct Point {
    int x;
    int y;
};

struct Arr {
    int range_minimum;
    int range_maximum;
    Point* arr;
    int size;

    Arr(int n, int min, int max);
    ~Arr();
};

class Dot {
private:
    int num;
    Arr points;
    float max_dist;
    float min_dist;
    pair<int,int> min_pair;
    pair<int,int> max_pair;

public:
    Dot(int n, int min, int max);
    ~Dot();
    void create_rand_dot();
    void print();
    double distance(int i, int j);
    void calculation_length();
};

double mySqrt(double x);

void inputInt(const char* prompt, int* value);
```

점의 좌표를 저장할 구조체와 계산을 할 클래스를 만들었고, 최대길이에 위치한 두 점과 최소길이에 위치한 두 점을 저장할 변수를 pair로 선언하였다.

2) 함수 정의 파일

```cpp
#include "hw2.hpp"
#include <random>

Arr::Arr(int n, int min, int max) {
    range_minimum = min;
    range_maximum = max;
    size = n;
    arr = new Point[size];
}

Arr::~Arr() {
    delete[] arr;
}

Dot::Dot(int n, int min, int max)
    : num(n), points(n, min, max), max_dist(0), min_dist(1e9) {}

Dot::~Dot() {}

void Dot::create_rand_dot() {
    random_device rd;
    mt19937 gen(rd());
    uniform_int_distribution<> dist(points.range_minimum, points.range_maximum);

    int placed = 0;
    while (placed < num) {
        int x = dist(gen);
        int y = dist(gen);
        bool duplicate = false;
        for (int j = 0; j < placed; j++) {
            if (points.arr[j].x == x && points.arr[j].y == y) {
                duplicate = true;
                break;
            }
        }
        if (!duplicate) {
            points.arr[placed].x = x;
            points.arr[placed].y = y;
            placed++;
        }
    }
```

```
void Dot::print() {
    for (int i = 0; i < num; i++) {
        cout << i << "번 점: (" << points.arr[i].x << ", " << points.arr[i].y << ")₩n";
    }
}

double Dot::distance(int i, int j) {
    int dx = points.arr[i].x - points.arr[j].x;
    int dy = points.arr[i].y - points.arr[j].y;
    return mySqrt((double)(dx*dx + dy*dy));
}

void Dot::calculation_length() {
    for (int i = 0; i < num; i++) {
        for (int j = i+1; j < num; j++) {
            double d = distance(i, j);
            if (d > max_dist) {
                max_dist = d;
                max_pair = {i, j};
            }
            if (d < min_dist) {
                min_dist = d;
                min_pair = {i, j};
            }
        }
    }

    cout << "₩n=== 거리 계산 결과 ===₩n";
    cout << "최소 거리: " << min_dist << " ("
         << min_pair.first << "번 점 " << "(" << points.arr[min_pair.first].x << ", " << points.arr[min_pair.first].y << ")"
         << " ↔ "
         << min_pair.second << "번 점 " << "(" << points.arr[min_pair.second].x << ", " << points.arr[min_pair.second].y << "))₩n";

    cout << "최대 거리: " << max_dist << " ("
         << max_pair.first << "번 점 " << "(" << points.arr[max_pair.first].x << ", " << points.arr[max_pair.first].y << ")"
         << " ↔ "
         << max_pair.second << "번 점 " << "(" << points.arr[max_pair.second].x << ", " << points.arr[max_pair.second].y << "))₩n";
}
```

```
double mySqrt(double x) {
    if (x == 0) return 0;
    double guess = x;
    for (int i = 0; i < 20; i++) {
        guess = 0.5 * (guess + x / guess);
    }
    return guess;
}

void inputInt(const char* prompt, int* value) {
    while (true) {
        cout << prompt;
        double temp;
        cin >> temp;

        if (cin.fail()) {
            cin.clear();
            char c;
            while (cin.get(c) && c != '₩n');
            cout << "잘못된 입력입니다. 정수를 입력해주세요." << endl;
            continue;
        }

        if (temp != static_cast<int>(temp)) {
            cout << "정수를 입력해주세요." << endl;
            continue;
        }

        *value = static_cast<int>(temp);
        break;
    }
}
```

클래스, 구조체 선언 시 변수들의 초기화와 사용자가 원하는 점의 개수만큼 동적할당 배열 선언함

Dot::create_rand_dot() 함수는 지정된 범위 안에서 중복되지 않는 무작위 좌표를 배열에 저장하는 기능을 한다. 난수 발생기를 초기화한 뒤, 반복문을 돌면서 무작위로 x와 y 좌표를 생성하고, 이미 저장된 좌표와 비교해 중복이 없으면 배열에 추가한다. 이 과정을 원하는 개수만큼 반복하면 결과적으로 서로 다른 좌표들이 무작위로 저장된다.

print() 함수는 배열에 저장된 모든 점의 번호와 좌표를 순서대로 출력한다.

distance() 함수는 두 점의 인덱스를 받아 좌표 차이를 계산한 뒤 피타고라스 정리를 이용해 두 점 사이의 거리를 구한다. 여기서 제곱근은 mySqrt()라는 별도의 함수를 사용하여 뉴턴 방법으로 근사 계산한다.

calculation_length() 함수는 모든 점 쌍을 비교하면서 거리를 계산하고, 그중 가장 가까운

두 점과 가장 먼 두 점을 찾아 각각의 거리와 좌표를 출력한다.
inputInt() 함수는 사용자가 입력한 값을 정수로 받을 때 사용되며, 잘못된 입력이나 실수가 들어온 경우 오류 메시지를 출력하고 다시 입력을 받도록 처리되어 있다.

3) main

```cpp
#include "hw2.hpp"
#include <iostream>
using namespace std;

int main() {
    int max, min, num;

    inputInt("좌표의 최솟값을 입력하세요 : ", &min);
    inputInt("좌표의 최댓값을 입력하세요 : ", &max);
    inputInt("생성할 점의 개수를 입력하세요 : ", &num);

    if ((max - min + 1) * (max - min + 1) < num) {
        cout << "점의 갯수가 너무 많습니다." << endl;
        return 0;
    }
    if (num < 2) {
        cout << "점의 갯수가 너무 적습니다." << endl;
        return 0;
    }
    if (max < min) {
        cout << "최솟값과 최댓값이 옳바르지 않습니다." << endl;
        return 0;
    }

    Dot d(num, min, max);
    d.create_rand_dot();
    d.print();
    d.calculation_length();

    return 0;
}
```

숫자를 입력받고 각 숫자들의 예외처리 이후 각각의 함수를 호출한다.

3. hw3

– 실행 결과

```
Type Command(A/U/D/R/L/S)
U
Y Position 1moved!
Type Command(A/U/D/R/L/S)
D
Y Position -1moved!
Type Command(A/U/D/R/L/S)
R
X Position 1moved!
Type Command(A/U/D/R/L/S)
L
X Position -1moved!
Type Command(A/U/D/R/L/S)
A
공격 실패!
Type Command(A/U/D/R/L/S)
S
HP:50
MP:9
Position:0,0
Type Command(A/U/D/R/L/S)
U
Y Position 1moved!
```

```
Type Command(A/U/D/R/L/S)
U
Y Position 1moved!
Type Command(A/U/D/R/L/S)
U
Y Position 1moved!
Type Command(A/U/D/R/L/S)
U
Y Position 1moved!
Type Command(A/U/D/R/L/S)
R
X Position 1moved!
Type Command(A/U/D/R/L/S)
R
X Position 1moved!
Type Command(A/U/D/R/L/S)
R
X Position 1moved!
Type Command(A/U/D/R/L/S)
R
X Position 1moved!
Type Command(A/U/D/R/L/S)
R
```

```
X Position 1moved!
Type Command(A/U/D/R/L/S)
A
공격 성공!
남은 체력:40
Type Command(A/U/D/R/L/S)
A
공격 성공!
남은 체력:30
Type Command(A/U/D/R/L/S)
A
공격 성공!
남은 체력:20
Type Command(A/U/D/R/L/S)
A
공격 성공!
남은 체력:10
Type Command(A/U/D/R/L/S)
A
공격 성공!
남은 체력:0
Monster Die!!
```

```
Type Command(A/U/D/R/L/S)
C
command error
Type Command(A/U/D/R/L/S)
A
공격 실패!
Type Command(A/U/D/R/L/S)
A
공격 실패!
Type Command(A/U/D/R/L/S)
A
공격 실패!
Type Command(A/U/D/R/L/S)
A
공격 실패!
Type Command(A/U/D/R/L/S)
A
공격 실패!
Type Command(A/U/D/R/L/S)
A
공격 실패!
Type Command(A/U/D/R/L/S)
A
공격 실패!
```


Type Command(A/U/D/R/L/S)
A
공격 실패!
Type Command(A/U/D/R/L/S)
A
공격 실패!
Type Command(A/U/D/R/L/S)
A
공격 실패!
Type Command(A/U/D/R/L/S)
A
MP 부족!

- 코드 분석

1) 헤더파일

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

class Monster {
public:
    int HP, x, y;
    Monster();
    Monster(int x, int y, int HP);
    int Be_Attacked();
};

class Player {
public:
    int HP, MP, x, y;
    Player();
    Player(int x, int y);
    void Attack(Monster &target);
    void Show_Status();
    void X_move(int move);
    void Y_move(int move);
};
```

과제에서 선언하라고 한 클래스이다.

2) 함수 정의 파일

```cpp
#include "hw2.hpp"
#include <iostream>
using namespace std;

Player::Player() {
    HP = 50;
    MP = 10;
}

Player::Player(int x_val, int y_val) {
    HP = 50;
    MP = 10;
    x = x_val;
    y = y_val;
}

Monster::Monster() {
    HP = 50;
    x = 0;
    y = 0;
}

Monster::Monster(int x_val, int y_val, int hp_val) {
    x = x_val;
    y = y_val;
    HP = hp_val;
}

// Player 멤버 함수
void Player::Attack(Monster &target) {
    MP--;
    if ((target.x == x) && (target.y == y)) {
        cout << "공격 성공!" << endl;
        HP = target.Be_Attacked();
    }
    else {
        cout << "공격 실패!" << endl;
    }
}
```

```cpp
void Player::Show_Status() {
    cout << "HP:" << HP << endl;
    cout << "MP:" << MP << endl;
    cout << "Position:" << x << ',' << y << endl;
}

void Player::X_move(int move) {
    x += move;
    cout << "X Position " << move << " moved!" << endl;
}

void Player::Y_move(int move) {
    y += move;
    cout << "Y Position " << move << " moved!" << endl;
}

// Monster 멤버 함수
int Monster::Be_Attacked() {
    HP -= 10;
    cout << "남은 체력:" << HP << endl;
    return HP;
}
```

Player 클래스는 기본적으로 체력(HP) 50과 마나(MP) 10을 가진 상태로 생성되며, 좌표를 지정하지 않으면 위치 값은 초기화되지 않고, 다른 생성자를 통해 좌표를 설정할 수도 있다.
Monster 클래스 역시 체력 50과 좌표 (0,0)으로 기본 생성되거나, 원하는 좌표와 체력 값을 입력받아 생성할 수 있다.
Attack() 함수는 공격 시 마나를 1 소모하고, 플레이어와 몬스터의 좌표가 같을 경우 공격이 성공하여 몬스터의 체력이 10 줄어든다. 이때 Monster::Be_Attacked() 함수가 호출되며, 몬스터의 남은 체력이 출력된다. 반대로 위치가 다르면 공격은 실패한다.
Show_Status() 함수는 현재 자신의 체력, 마나, 위치를 출력한다.
X_move()와 Y_move() 함수를 통해서 플레이어의 좌표를 이동시켜 플레이어의 이동을 구현한다.
Monster 클래스의 Be_Attacked() 함수는 체력을 10줄이고 현재 체력을 출력하며, 그 값을 반환한다.
즉 플레이어가 이동하면서 몬스터와 같은 위치에 도달하면 공격할 수 있고, 그때마다 몬스터의 체력이 줄어들게 된다.

3) main

```
#include "hw2.hpp"
#include <iostream>
using namespace std;

int main() {
    Player player(0, 0);
    Monster monster(5, 4, 50);

    while (1) {
        if (player.HP == 0) {
            cout << "Monster Die!!" << endl;
            return 0;
        }

        char ans;
        cout << "Type Command(A/U/D/R/L/S)" << endl;
        cin >> ans;

        char c;
        while (cin.get(c) && c != '₩n');

        switch (ans) {
        case 'A':
            if (player.MP == 0) {
                cout << "MP 부족!" << endl;
                return 0;
            }
            player.Attack(monster);
            break;

        case 'U': player.Y_move(1); break;
        case 'D': player.Y_move(-1); break;
        case 'R': player.X_move(1); break;
        case 'L': player.X_move(-1); break;
        case 'S': player.Show_Status(); break;
        default: cout << "command error" << endl; break;
        }
    }
}
```

각 객체 선언 이후 명령어를 받아와서 스위치문으로 각각의 함수를 실행하여서 동작하는데 이때 문자열이 들어올 시 맨 첫 문자만 저장하고 나머지는 버퍼에서 제거한다.