

C++_2일차_보고서_20기_인턴_2025402040_컴퓨터정보공학부_노기문

목차

- p.2 - hw_01 : 숫자 할당 및 평균, 최대, 최소 구하기
- p.12 - hw_02 : 여러개의 점 생성 및 각각의 점들사이간의 거리
- p.20 - hw_03 : 게임 구현

hw_01 : 숫자 할당 및 평균, 최대, 최소 구하기

<문제 요지>

이번 문제는 C에서의 기본 적인 것을 C++에서 얼마나 잘 적용하냐의 문제입니다.

중요포인트는

1. 새로운 동적할당을 어떻게 적용할것인가?
2. C에서의 구조체격인 class를 어떻게 할것이고 객체1개를 어떻게 활용 할것인가?

↳ cpp코드의 stats를 객체로 하여 진행했습니다.

입니다.

조건

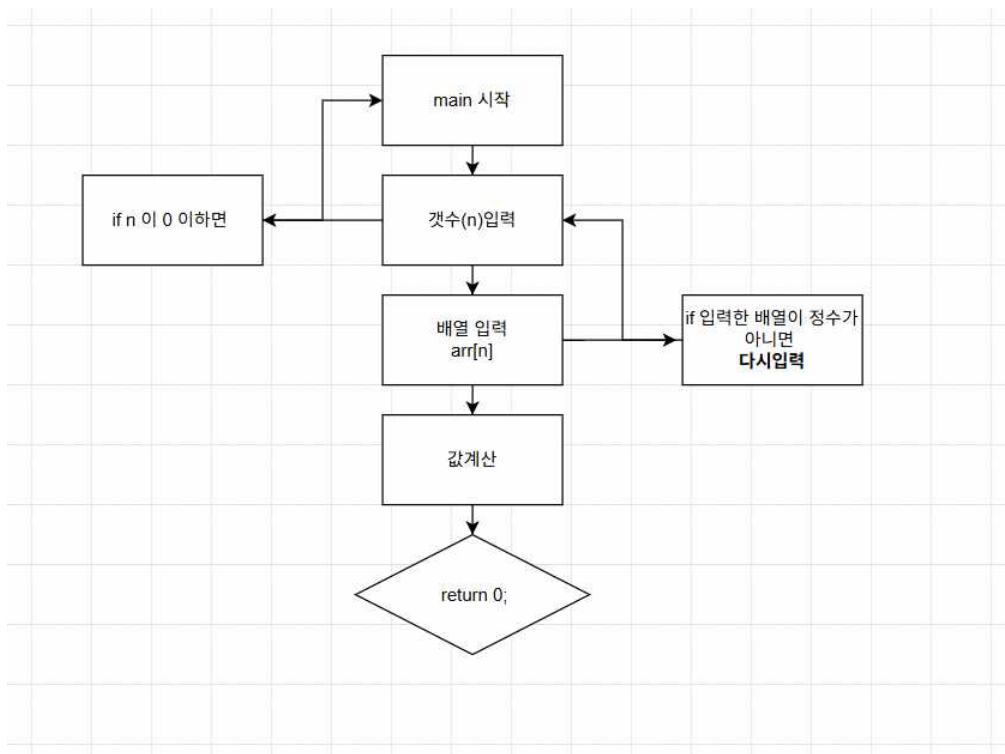
다음 과제를 C++로, Class를 이용하여 작성하시오

모든 변수와 함수를 클래스 멤버변수, 멤버함수로 작성할 것

(코드에서 생성하는 클래스 객체는1개)

<알고리즘 설명 및 code>

1. 알고리즘 설명



1. 배열 크기 입력

사용자에게 몇 개의 정수를 입력할지 묻는다.

0 이하가 들어오면 다시 입력받는다.

양수가 들어오면 그 크기만큼 동적 배열을 생성한다.

2. 배열 값 입력 (검증 포함)

각 원소를 차례대로 입력받는다.

입력은 문자열로 받고, 그 문자열이 정수로만 구성되었는지 검사한다.

맨 앞이 '-'이면 음수로 허용하고 나머지 문자가 숫자인지 확인.

숫자가 아닌 문자가 하나라도 있으면 다시 입력 요구.

검증이 끝나면 직접 정수 변환(문자 '0'~'9'를 숫자 값으로 바꿔서 계산).

변환된 정수를 배열에 저장.

3. 통계 연산

배열이 채워지면 다음 작업을 수행한다.

최대값: 배열 첫 원소를 기준으로 잡고, 순차 비교하면서 더 큰 값으로 갱신.

최소값: 동일하게 최소값으로 갱신.

합계: 모든 원소를 누적합.

평균: 합계를 배열 크기로 나눈 실수 값.

4. 출력

최대값, 최소값, 합계, 평균을 화면에 출력하고 프로그램 종료.

즉, 전체 흐름은 배열 크기 입력 → 유효한 정수만 배열에 채움 → 배열을 순회하며 최대·최소·합·평균 계산 → 결과 출력으로 요약된다.

2. hpp :

```
1  class ArrayStats {
2  private:
3      int* arr;    // 정수 배열로 변경
4      int size;
5  public:
6      ArrayStats(int n);    // 동적 할당
7      ~ArrayStats();        // 메모리 해제
8      void input();         // 입력 (문자열 검사 후 정수 변환)
9      int getMax();         // 최대값
10     int getMin();         // 최소값
11     int getSum();         // 합
12     double getAverage();  // 평균
13 };
```

private

1. *arr : 동적으로 입력한 값을 할당하고=> class 내부에서 public | getMax, getMin, getSum, getAverage|를 통해 계산합니다.
2. size : input()함수에서 배열에 숫자들을 할당하기 위한 정수 = main()에서의 n입니다.

public

1. ArrayStats(int n) : 생성자
2. ~ArrayStats() : 소멸자
3. void input() : 배열을 입력받는 함수
4. int getMax() : 최대 구하는 함수
5. int getMin() : 최소 구하는 함수
6. int getSum() : 배열들의 합을 구하는 함수
7. double getAverage() : 배열들의 평균을 구하는 함수

3. cpp :


main

```
1  int main() {
2      int n;
3      while (true) {
4          cout << "몇 개의 원소를 할당하겠습니까? : ";
5          cin >> n;
6          if (n <= 0) {
7              cout << "개수는 양수입니다. 다시 입력하세요.\n";
8          } else break;
9      }
10
11     ArrayStats stats(n);
12     stats.input();
13
14     cout << "최대값: " << stats.getMax() << endl;
15     cout << "최소값: " << stats.getMin() << endl;
16     cout << "전체합: " << stats.getSum() << endl;
17     cout << "평균: " << stats.getAverage() << endl;
18
19     return 0;
20 }
```

앞의 알고리즘으로 설명 대체.

중요** : 객체인 stats를 1개를 사용해

1-3line



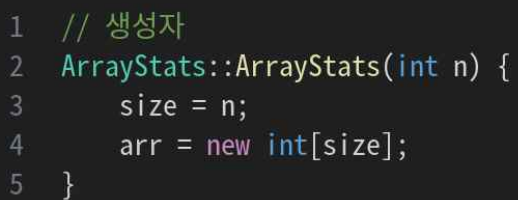
```
1  #include "cpp_test1/hw_01.hpp"
2  #include <iostream>
3  using namespace std;
```

1: 위의 헤더파일

2: C++에 있는 입출력을 위한 헤더 파일

3: std쓰기위한 코드

5-9line



```
1  // 생성자
2  ArrayStats::ArrayStats(int n) {
3      size = n;
4      arr = new int[size];
5  }
```

메인에서 받은 n으로 배열의 개수 할당

11-14line



```
1 // 소멸자
2 ArrayStats::~ArrayStats() {
3     delete[] arr;
4 }
```

소멸자 정의

16-55 line

```
1 // 입력
2 void ArrayStats::input() {
3     for (int i = 0; i < size; i++) {
4         while (true) {
5             std::string temp;
6             std::cout << i + 1 << "번째 정수 입력: ";
7             std::cin >> temp;
8
9             bool is_number = true;
10            int start = 0;
11
12            // 음수 허용 → 맨 앞에 '-' 있으면 건너뛰
13            if (temp[0] == '-') {
14                if (temp.size() == 1) is_number = false; // "-"만 입력된 경우
15                start = 1;
16            }
17
18            for (int j = start; j < (int)temp.size(); j++) {
19                if (temp[j] < '0' || temp[j] > '9') {
20                    is_number = false;
21                    break;
22                }
23            }
24
25            if (is_number) {
26                // 문자열 → 정수 변환 직접 구현
27                int value = 0;
28                for (int j = start; j < (int)temp.size(); j++) {
29                    value = value * 10 + (temp[j] - '0');
30                }
31                if (temp[0] == '-') value = -value;
32
33                arr[i] = value;
34                break;
35            } else {
36                std::cout << "정수가 아닙니다. 다시 입력하세요.\n";
37            }
38        }
39    }
40 }
```

먼저 bool type 으로 True / False로 이것이 숫자인지 아닌지 판별하는 로직을 사용한다.

예외처리를 위해 문자열로 값을 받아준다음

문자열의 첫 번째가 '-'인지 아닌지 판별한다.

if '-'면 그다음것의 여부에 따라 '-'만 입력했는지 아니면 다음것이 존재하는지 판단한다.

다음것이 존재하면 다음것부터 숫자를 판별할수 있게 start =1;로 설정한다.

양수든 음수든 이제 size에 따라 0과 9사이에 없으면 bool type을 false로 바꾸고 다시 입력하게 한다. (소수도 인정X why? => '.'은 숫자가 아니기 때문)

모든 것을 판별했을 때 다 숫자라면 (is_number = True)

```
int value = 0;
for (int j = start; j < (int)temp.size(); j++) {
    value = value * 10 + (temp[j] - '0');
}
```

value를 0으로 초기화.

temp라는 문자열을 한 글자씩 확인하면서 '0'을 빼서 숫자로 변환.

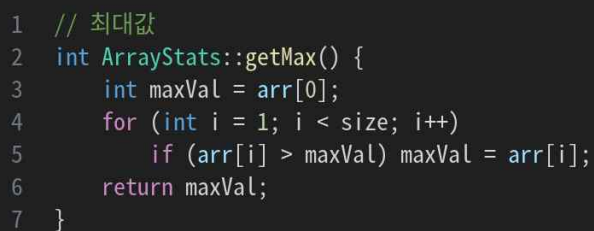
value = value * 10 + digit로 자릿수를 반영.

ex : "123" → (((0*10+1)*10+2)*10+3) = 123

음수 처리

```
if (temp[0] == '-') value = -value;
문자열 첫 글자가 '-'라면 음수로 변환
```


57-63line



```
1 // 최대값
2 int ArrayStats::getMax() {
3     int maxVal = arr[0];
4     for (int i = 1; i < size; i++)
5         if (arr[i] > maxVal) maxVal = arr[i];
6     return maxVal;
7 }
```

배열 1번째를 최대값으로 잡고 나머지 값들과 비교하면 더 크면 최대값에 큰 배열로 저장


65-71line



```
1 // 최소값
2 int ArrayStats::getMin() {
3     int minVal = arr[0];
4     for (int i = 1; i < size; i++)
5         if (arr[i] < minVal) minVal = arr[i];
6     return minVal;
7 }
```

57-63line과 로직 동일


73-78line



```
1 // 합
2 int ArrayStats::getSum() {
3     int sum = 0;
4     for (int i = 0; i < size; i++) sum += arr[i];
5     return sum;
6 }
```

반복하며 배열의 합구함.

80-83line



```
1 // 평균
2 double ArrayStats::getAverage() {
3     return static_cast<double>(getSum()) / size;
4 }
```

73~78line에서 구한 합을 전체의 개수인 size로 나눔.

<result screen>

```
ngm@ngm:~/intern_ws/cpp_test1/build$ ./run
몇 개의 원소를 할당하겠습니까? : -9
개수는 양수입니다. 다시 입력하세요.
몇 개의 원소를 할당하겠습니까? : 0
개수는 양수입니다. 다시 입력하세요.
몇 개의 원소를 할당하겠습니까? : 4
1번째 정수 입력: 1
2번째 정수 입력: 2
3번째 정수 입력: 3
4번째 정수 입력: asasasasa
정수가 아닙니다. 다시 입력하세요.
4번째 정수 입력: 0.33333
정수가 아닙니다. 다시 입력하세요.
4번째 정수 입력: -6
최대값: 3
최소값: -6
전체합: 0
평균: 0
ngm@ngm:~/intern_ws/cpp_test1/build$ ^C
ngm@ngm:~/intern_ws/cpp_test1/build$
```

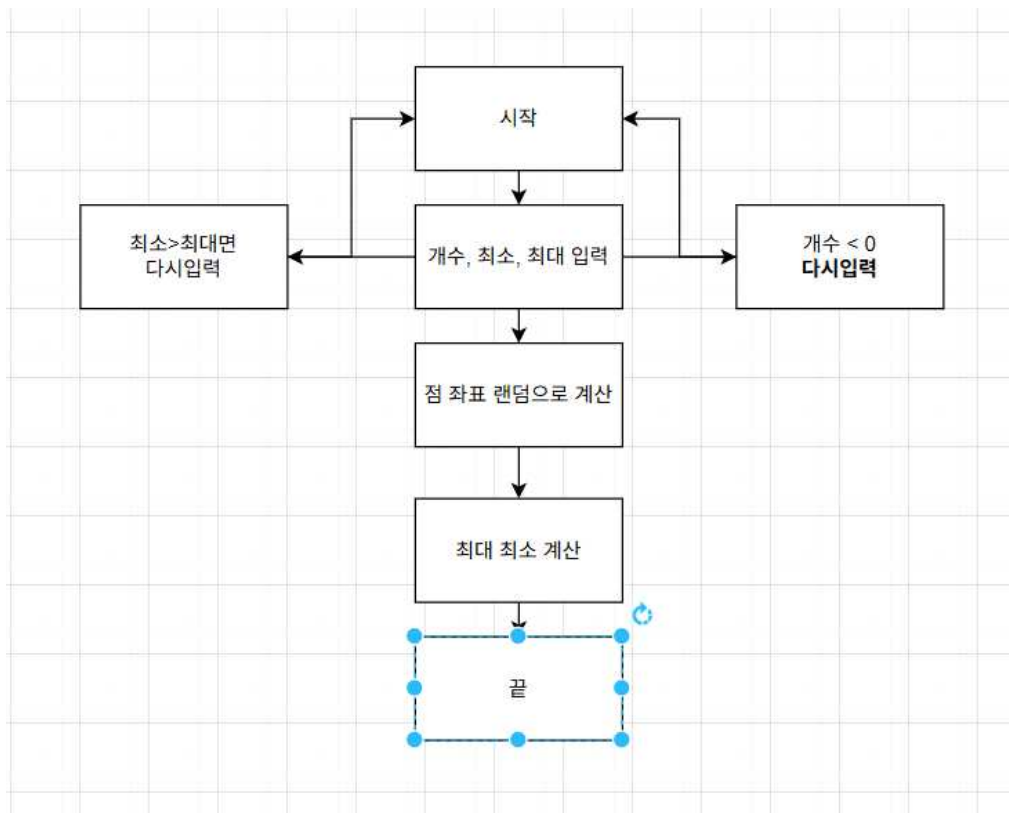
hw_02 : 여러개의 점 생성 및 각각의 점들사이간의 거리

<문제 요지>

1. 이문제는 class내에 구조체를 어떻게 구현 할것인가의 문제이다. => struct Point & class PointClac 를 만들어 해결 했습니다.

<알고리즘 설명 및 code>

1. 알고리즘



1. 입력

사용자에게 점의 개수 n , 좌표 최소값 \min , 최대값 \max 를 입력받는다.

조건 검사:

$n < 1$ 이면 다시 입력 요구.

$\min > \max$ 이면 다시 입력 요구.

2. 점 생성 (PointCalc 생성자)

크기 n 의 동적 배열 `points`를 할당.

각 점의 좌표 (x, y) 를 \min 이상 \max 이하의 정수 난수로 초기화.

`rand() % (max - min + 1)`로 구간 난수 생성.

points[i].x, points[i].y에 저장.

3. 점 출력

printPoints()에서 모든 점을 (x, y) 형태로 출력.

4. 최소 거리 계산 (Pointsdistance_min)

모든 점 쌍 (i, j)에 대해 제곱 거리 $d2 = (x_i - x_j)^2 + (y_i - y_j)^2$ 를 구한다.

가장 작은 제곱거리 min_d2와 대응하는 점 쌍 (A, B)를 저장.

최종적으로 $\sqrt{\text{min_d2}}$ 를 구해 두 점 좌표와 함께 출력.

5. 최대 거리 계산 (Pointsdistance_max)

모든 점 쌍 (i, j)에 대해 같은 방식으로 제곱 거리 d2를 구한다.

이번에는 가장 큰 max_d2를 찾아 점 쌍 (A, B)를 저장.

최종적으로 $\sqrt{\text{max_d2}}$ 를 구해 두 점 좌표와 함께 출력.

6. 프로그램 흐름 요약

점 개수와 범위를 입력받음.

난수로 점들을 생성.

모든 점 출력.

모든 점 쌍의 거리를 계산해 최소·최대 거리와 그 점 쌍 출력.

2. hpp

```
1  struct Point {
2      int x;
3      int y;
4  };
5
6  class PointCalc {
7  private:
8      Point* points;    // 동적 배열
9      int size;
10     int *distance;
11 public:
12     PointCalc(int n, int min, int max);
13     ~PointCalc();
14     void printPoints();
15
16     void Pointsdistance_max();
17     void Pointsdistance_min();
18 };
```

먼저 Point로 구조체를 만들어 x좌표와, y좌표를 설정
또한 class내에 좌표, 거리를 동적할당을 진행

3. cpp

main

```
1  int main() {
2      srand(static_cast<unsigned int>(time(NULL)));
3      int n, min, max;
4      while(1){
5          std::cout << "점 개수 ,최소값, 최대값 입력(소수를 입력해도 정수부분만 인식합니다.): ";
6          std::cin >> n >> min >> max;
7
8          if (n < 1) {
9              std::cout << "N은 1 이상이어야 함\n";
10             continue;
11         }
12
13         else if (min > max){
14             std::cout<<"최소값이 최대값의 값보다 클수 없습니다\n";
15         }else break;
16     }
17
18     PointCalc pc(n, min, max);
19     pc.printPoints();
20     pc.Pointsdistance_min();
21     pc.Pointsdistance_max();
22
23     return 0;
24 }
```

srand()를 통해 난수 초기화 설정

개수, 최소, 최대 입력(예외처리 | 개수<0, 최소>최대면 다시입력)

line 1-4

```
1  #include "cpp_test1/hw_02.hpp"
2  #include <iostream>
3  #include <cmath>
4  using namespace std;
```

헤더파일들

cmath 헤더파일 => 제곱근 계산 가능

line 6-15

```
1 PointCalc::PointCalc(int n, int min, int max) {
2     size = n;
3     points = new Point[size];
4     // 한 번만 호출하면 됨
5     for (int i = 0; i < size; i++) {
6
7         points[i].x = min + rand() % (max - min + 1);
8         points[i].y = min + rand() % (max - min + 1);
9     }
10 }
11 PointCalc::~PointCalc() {
12     delete[] points;
13 }
```

생성자/ 소멸자

main에서 받은 개수, 최소, 최대를 받아 좌표의 동적할당을 진행함.
그런다음 rand()를 사용해 x,y좌표 난수설정

line 19-23

```
1 void PointCalc::printPoints() {
2     for (int i = 0; i < size; i++) {
3         cout << "점" << i+1 << " = (" << points[i].x << ", " << points[i].y << ")\n";
4     }
5 }
```

할당한 좌표들을 cout

line 26-48

```
1 void PointCalc::Pointsdistance_min() {
2     if (size < 2) return;
3     int dx1 = points[0].x - points[1].x;
4     int dy1 = points[0].y - points[1].y;
5     int d_origin = dx1 * dx1 + dy1 * dy1;
6     int min_d2 = d_origin;
7     Point A{}, B{};
8     for (int i = 0; i < size; i++) {
9         for (int j = i + 1; j < size; j++) {
10             int dx = points[i].x - points[j].x;
11             int dy = points[i].y - points[j].y;
12             int d2 = dx * dx + dy * dy;
13             if (d2 < min_d2) {
14                 min_d2 = d2;
15                 A = points[i];
16                 B = points[j];
17             }
18         }
19     }
20     cout << "최소 거리 = " << sqrt((double)min_d2)
21         << " (" << A.x << ", " << A.y << ") - ("
22         << B.x << ", " << B.y << ")\n";
23 }
```

먼저 좌표간의 거리차를 구하기 위해 1번째(0) 와 2번째(1)의 거리를 구하고 그것을 최소로 기준을 잡음 그것을 통해 반복문으로 좌측값을 구함 또한 cout을 통해 최소거리와 좌표를 표시함.

line 50 - 69

```
1 void PointCalc::Pointsdistance_max() {
2     if (size < 2) return;
3     int max_d2 = -1;
4     Point A{}, B{};
5     for (int i = 0; i < size; i++) {
6         for (int j = i + 1; j < size; j++) {
7             int dx = points[i].x - points[j].x;
8             int dy = points[i].y - points[j].y;
9             int d2 = dx * dx + dy * dy;
10            if (d2 > max_d2) {
11                max_d2 = d2;
12                A = points[i];
13                B = points[j];
14            }
15        }
16    }
17    cout << "최대 거리 = " << sqrt((double)max_d2)
18         << " ((" << A.x << ", " << A.y << ") - ("
19         << B.x << ", " << B.y << "))\n";
20 }
```

최소값 로직과 동일 but 길이는 음수가 나올수 없기에 0으로 시작

<result screen>

```
ngm@ngm:~/intern_ws/cpp_test1/build$ ./run
점 개수 ,최소값, 최대값 입력: -9 5 6
N은 1 이상이어야 함
점 개수 ,최소값, 최대값 입력: 5 20 10
최소값이 최대값의 값보다 클수 없습니다
점 개수 ,최소값, 최대값 입력: 5 10 20
점1 = (17,12)
점2 = (11,17)
점3 = (16,12)
점4 = (20,14)
점5 = (12,10)
최소 거리 = 1 ((17,12) - (16,12))
최대 거리 = 9.48683 ((11,17) - (20,14))
```

hw_03 : 게임구현

<문제 요지>

3번 과제는 C++ 객체지향 프로그래밍 연습용으로, 플레이어와 몬스터를 각각 클래스(Player, Monster)로 나누어 구현하는 문제입니다.

핵심 요구사항

1. 클래스 분리

헤더 파일(.hpp)과 소스 파일(.cpp)을 따로 작성해야 합니다.

Player 클래스와 Monster 클래스를 각각 만듭니다.

2. 기능

플레이어는 명령어 입력(A/U/D/R/L/S)을 받아 이동, 공격, 상태 출력 가능.

공격 시 MP 1 소모.

MP가 부족하면 공격 불가능.

몬스터는 플레이어 공격을 받아 HP가 줄어듦.

3. 종료 조건

몬스터 HP가 0 이하가 되면 Monster Die! 출력 후 프로그램 종료.

시뮬레이션 예시 흐름

이동 명령어 입력 → 좌표 변경 출력

상태 명령어 입력 → HP, MP, 좌표 출력

공격 명령어 입력 → 몬스터 체력 감소, MP 감소

몬스터 체력 0 → 게임 종료

즉, 플레이어의 명령 입력을 통해 이동, 상태 확인, 공격을 수행하고, 몬스터가 쓰러질 때까지 진행하는 간단한 텍스트 기반 게임을 만드는 과제입니다.

<알고리즘 설명 및 code>

1. 알고리즘

1. 초기화

Player 객체 생성 (초기 HP=50, MP=10, 위치 (0,0))

Monster 객체 생성 (HP=50, 위치 (5,4))

2. 명령 입력 루프

사용자로부터 한 글자 명령을 입력받는다.

가능한 명령:

A/a: 공격

U/u: 위로 이동 (y-1)

D/d: 아래로 이동 (y+1)

L/l: 왼쪽 이동 (x-1)

R/r: 오른쪽 이동 (x+1)

S/s: 상태 출력

3. 공격 로직 (Player::Attack)

MP가 0 이하면 공격 불가, 프로그램 종료.

공격 시 MP 1 소모.

Monster::Be_Attacked(10) 호출 → 몬스터 HP 10 감소.

결과 출력: 남은 몬스터 HP.

몬스터 HP ≤ 0 이면 처치 성공 메시지 출력 후 프로그램 종료.

4. 이동 로직 (X_move, Y_move)

플레이어의 좌표를 +1 또는 -1 이동.

이동 후 좌표 출력.

5. 상태 확인 (Show_Status)

현재 HP, MP, 좌표(x,y)를 출력.

6. 프로그램 종료 조건

몬스터를 공격해서 HP가 0 이하가 되면 종료.

플레이어가 MP를 모두 소모하면 공격 시도 시 종료.

그 외에는 무한히 명령을 입력받아 진행.

즉, 사용자가 키보드로 명령을 입력해 플레이어를 움직이거나 몬스터를 공격하고, 몬스터를 쓰러뜨리거나 MP가 다 소진될 때까지 반복하는 텍스트 기반 RPG 미니게임 알고리즘이다.

2. hpp

```
#include <iostream>
```

```
class Monster;
```

```
class Player {
```

```
public:
```

```
    int HP, MP, x, y;
```

```
    Player();
```

```
    Player(int x, int y);
```

```
    void Attack(Monster &target);
```

```
    void Show_Status() const;
```

```
    void X_move(int move);
```

```
    void Y_move(int move);
```

```
};
```

```
class Monster {
```

```
public:
```

```
    int HP, x, y;
```

```
    Monster();
```

```
    Monster(int x, int y, int HP);
```

```
    int Be_Attacked(int damage); // 남은 HP 반환
```

```
};
```

3. cpp

```
#include "cpp_test1/hw_03.hpp"
```

```
using namespace std;
```

```
static const int ATTACK_POWER = 10;
```

```
Player::Player() : HP(50), MP(10), x(0), y(0) {}
```

```
Player::Player(int x_, int y_) : HP(50), MP(10), x(x_), y(y_) {}
```

```
void Player::Attack(Monster &target) {
```

```
    if (MP <= 0) {
```

```
        cout << "MP 부족!\n";
```

```
        exit(0);
```

```
    }
```

```
    MP -= 1;
```

```
    int remain = target.Be_Attacked(ATTACK_POWER);
```

```
    cout << "공격 성공! 몬스터 HP:" << remain << "\n";
```

```
    if (remain <= 0) {
```

```
        cout << "몬스터를 처치했습니다. 프로그램 종료.\n";
```

```
        exit(0);
```

```
    }
```

```
}
```

```
void Player::Show_Status() const {
```

```
    cout << "HP:" << HP << "\nMP:" << MP << "\nPosition:" << x << "," << y << "\n";
```

```
}
```

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

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

```
Monster::Monster() : HP(30), x(0), y(0) {}
```

```
Monster::Monster(int x_, int y_, int HP_) : HP(HP_), x(x_), y(y_) {}
```

```
int Monster::Be_Attacked(int damage) {
```

```
    HP -= damage;
```

```
    if (HP < 0) HP = 0;
```

```
    return HP;
```

```
}
```

```

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

    cout << "Type Command(A/U/D/L/R/S)Wn";
    char c;
    while (cin >> c) {
        switch (c) {
            case 'A': case 'a':
                player.Attack(monster);
                break;
            case 'U': case 'u':
                player.Y_move(-1);
                break;
            case 'D': case 'd':
                player.Y_move(1);
                break;
            case 'L': case 'l':
                player.X_move(-1);
                break;
            case 'R': case 'r':
                player.X_move(1);
                break;
            case 'S': case 's':
                player.Show_Status();
                break;
            default:
                cout << "잘못된 명령Wn";
        }
        cout << "Type Command(A/U/D/L/R/S)Wn";
    }
    return 0;
}

```


<resut screen>

```
ngm@ngm:~/intern_ws/cpp_test1/build$ ./run
Type Command(A/U/D/L/R/S)
a
공격 성공! 몬스터 HP:40
Type Command(A/U/D/L/R/S)
u
Y Position moved! Now -1
Type Command(A/U/D/L/R/S)
d
Y Position moved! Now 0
Type Command(A/U/D/L/R/S)
l
X Position moved! Now -1
Type Command(A/U/D/L/R/S)
s
HP:50
MP:9
Position:-1,0
Type Command(A/U/D/L/R/S)
a
공격 성공! 몬스터 HP:30
Type Command(A/U/D/L/R/S)
a
공격 성공! 몬스터 HP:20
Type Command(A/U/D/L/R/S)
a
공격 성공! 몬스터 HP:10
Type Command(A/U/D/L/R/S)
a
공격 성공! 몬스터 HP:0
몬스터를 처치했습니다. 프로그램 종료.
```

-끝-