

C++프로그래밍및실습

# 전투력측정기

진척 보고서 #2

제출 일자 : 12/03

이름 : 한승한

학번 : 213996

## 프로젝트 목표

### 배경 및 필요성

코딩에 대한 배경지식이 거의 없어서 지금까지 배운 내용들을 기반으로 프로젝트를 진행하려고 했다.  
그러다 보니 단순하면서 실용성 있는 프로젝트를 생각하게 되었다.

운동을 시작하였는데 실제로 근성장이 이루어지고 있는지 궁금하여 확인이 필요하였고,  
이에 근성장이 제대로 이루어지는지 판단하는 프로그램이 필요하여 직접 만들어보기로 하였다.

### 프로젝트 목표

데이터가 주어졌을때 그것을 기반으로 얼마나 근력이 상승했는가에 대한 수치(단위 %)  
를 나타내는 프로그램을 만드는 것이 목표이다.

### 차별점

물론 흔히 일지를 작성하는 프로그램이 존재한다. 하지만 거의 일지만 작성할 수 있는 프로그램이며  
대부분 VER.Pro와 같은 유료화 된 버전을 사용해야만 수치가 나타난다.  
금전적 부분에서 차별이 있다고 생각한다.

## 기능 계획

### STEP 1. 시작 단계

사용자가 접근하기 쉽도록 길잡이가 필요하였고 이에 메뉴의 기능이 필요하다.

세부 기능 1)

선택하는 메뉴창을 띄운다. ( 1. 종합적으로 계산 / 2. 분할적으로 계산 )

선택 1) 바로 **STEP 2)**로 넘어간다.

선택 2) 밑의 세부 기능 2)로 넘어간다.

세부 기능 2)

메뉴창을 출력한다. ( 1. 등 / 2. 가슴 / 3. 어깨 / 4. 하체 / 5. 이두 / 6. 삼두 / 7. 복근 )

사용자가 올바른 번호 입력을 마치면 **STEP 2)**로 넘어간다.

세부 기능 2-1)

추가 기능을 생성한다. ( 입력 : /추가 )

만일 메뉴의 목록 중 자신이 추가하고자 하는 목록이 있는 경우 추가를 할 수 있도록 도와준다.

ex) 전완근을 추가하고자 한다. ( 입력: /추가 -> 입력 : 전완근 )

( 1. 등 / 2. 가슴 / 3. 어깨 / 4. 하체 / 5. 이두 / 6. 삼두 / 7. 복근 / 8. 전완근 )

세부 기능 3)

그 외의 입력일 경우 경고문을 보내고 다시 입력으로 돌아온다.

### STEP 2. 단위 선택 단계

사용자의 입력을 받기 전에 단위를 선택한다. ( 1. 달 / 2. 주 )

요일마다 받는 것은 수치가 굉장히 많아진다.

또한 근성장은 곧바로 일어나지 않기 때문에 큰 단위를 사용하여 입력받기 위함이다.

세부 기능 1)

단위를 입력받는다.

세부 기능 2)

그 외의 입력일 경우 경고문을 보내고 다시 입력으로 돌아온다.

### STEP 3. 사용자 입력 저장

사용자가 입력한 수치를 저장한다.

하지만 사용자가 입력법을 모르기 때문에 안내문을 출력한다.

세부 기능 1)  
안내문 출력

세부 기능 2)  
경로를 입력받고 출력하기 ( 경로를 입력받고 해당 파일을 출력한다. )

세부 기능 3)  
만일 경로를 읽을 수 없는 경우에는 경고문을 보내고 다시 입력으로 돌아온다.

### STEP 4. 사용자 DATA 분석

사용자의 데이터 값을 분석한다. 대단한 분석법을 쓰는 것이 아니고 그냥 상승하는 정도치만 계산한다.

세부 기능 1)  
입력받은 자료의 차이를 백분율로 바꿔서 계산시킬 것 같다.

세부 기능 2)  
cout를 사용해서 결과를 출력한다.

### STEP 5. 결과

분석값에 대한 결과를 그래프로 표기해준다.

세부 기능 1) Graph\_Cal  
그래프를 그리기 위해서는 결과값을 토대로 비율을 측정해야 한다. 그것을 구현한 함수다.

세부 기능 2) graph  
색이 채워진 블록과 그렇지 않은 블록을 사용해 그래프를 만들어볼 예정이다.

고려해야 할 점.

- \* 음수값이 발생 할 경우도 대비를 해야 한다. -> 오히려 근손실이 발생했을 경우
- \* 1달을 기준으로 측정하기에 측정 결과값이 매우 미미할 수 있다.
- \* 시험기간이나 과제로 인한 시간적 여유가 없을 경우를 대비해 넉넉한 일정으로 계획을 세운다.
- \* 자료값은 실제 나의 운동값을 사용할 것이므로 운동을 마친 후 일지를 꼭 작성해야 한다.

## 진척사항

### STEP 4. 데이터 분석

사용자의 데이터 값을 분석한다. 대단한 분석법을 쓰는 것이 아니고 그냥 상승하는 정도치만 계산한다.

세부 기능 1)

입력받은 자료의 차이를 백분율로 바꿔서 계산시킬 것 같다.

※ 데이터를 백분율로 바꾸는 계산은 다음과 같다.

$$rate = \frac{current - first}{first} \times 100(\%)$$

- \* rate는 결과값의 비율이다.
- \* current는 현재 가리키는 Data이다.
- \* first는 처음 설정해둔 초기 Data이다.

week1 ~ weekX까지 주가 진행될 때마다 비율은 업데이트 된다.

week1의 Data를 초기 first로 저장하고, current는 week2 ~ weekX까지 업데이트 된다.

```
void Calculator::Cal() { // 계산기 함수
    growth_rate.clear(); // 호출 될 때마다 결과값 초기화
    float rate; // 계산 결과값이 저장될 float 변수
    for (auto& row : current_value) {
        for (int j = 0; j < row.size(); j++) {
            // 증가한 비율 계산 : 현재주 - 이전주 / 이전주 * 100
            if (first_value[j] != 0) // 분모가 0이 아닐 경우 : 계산 진행
                rate = ((row[j]) - first_value[j]) / first_value[j] * 100;
            else
                rate = 0; // 분모가 0일 경우 : 0
            growth_rate.push_back(rate); // 결과값 push_back
        }
    }
}
```

< Cal() 함수 >

\* `growth_rate.clear()` : 여러번의 호출을 대비하여 계산한 성장률을 초기화하는 구문이다.

- 코드 상 여러번 호출되는 일이 없기에 사실상 필요는 없다. (삭제 예정)

\* `float rate` : 계산 결과값이 저장될 변수이다. 바로 벡터에 넣지 왜 변수를 통하느냐?

분모가 0인 경우와 그렇지 않은 경우를 대비하기 위해 변수를 통하여 구분하고 벡터에 넣는다.

\* `for문` : `current_value`는 밑의 함수에서 사용된다. 벡터 간 행과 열의 사이즈가 동일하기에

참조해도 상관이 없어서 넣었다. 행과 열을 반복하는 반복문이다.

\* **if문** : 분모가 0인 경우와 그렇지 않은 경우를 구분하여 계산 진행 후 결과 값 push\_back  
위 계산을 진행하기 위해선 각 변수에 값이 할당되어야 한다. 아래의 코드는 값을 할당하는 코드다.

```
// STEP 4.
void Calculator::Data_Analysis(const string& path) { // 한 줄씩 출력하는 함수
    string line; // 한줄 저장할 string 변수
    ifstream file(path); // 파일 경로 열기
    while (getline(file, line)) { // 한줄씩 반복
        stringstream ss(line);
        string value; // 구분한 값을 저장할 변수
        vector<string> data_row; // 구분한 열 데이터를 저장할 1차원 벡터
        while (getline(ss, value, ',')) {
            data_row.push_back(value); // 쉼표를 통해 구분하여 데이터에 push back
        }
        data.push_back(data_row); // 구분한 열 데이터를 2차원 벡터에 추가
    }

    file.close(); // 파일 닫기

    // 이름 저장
    for (int i = 1; i < data[0].size(); i++) {
        name.push_back(data[0][i]);
    }

    // 주 저장
    for (int i = 1; i < data.size(); i++) {
        week.push_back(data[i][0]);
    }

    // current_value 데이터 추가
    for (int i = 1; i < data.size(); i++) {
        current_value.push_back({});
        for (int j = 1; j < data[0].size(); j++) {
            current_value.back().push_back(atof(data[i][j].c_str()));
        }
    }

    // first_value 데이터 추가
    for (int i = 0; i < current_value[0].size(); i++) {
        first_value.push_back(current_value[0][i]);
    }

    Cal(); // 계산 진행
    Display(); // 결과 표기
}
```

※ 각 변수 설명은 주석을 통해 설명을 진행하고 있으니 생략한다.

\* **ifstream file(path)** : fstream을 통해 파일을 path에 저장된 경로를 통해 연다.

※ **include <fstream>**을 포함해야 한다.

\* **while문** : getline()을 통해 한 줄씩 읽는 과정을 반복한다.

\* **while문** : getline(ss, value, ',') 한 줄씩 읽어들이는 데이터를 쉼표로 구분하여 value에 저장한다.

※ **include <sstream>**을 포함해야 한다.

- 처음에는 Split()함수를 만들어 구현했으나 쉽고 간결한 방법을 찾아 채택했다.

\* **data\_row.push\_back** : 먼저 value를 행에 넣고, 행을 열에 넣는 과정이다.

\* **for문** : 각 데이터에 해당하는 인덱스를 참조하여 반복문을 통해 벡터에 저장하는 과정이다.

- 각 index가 1부터 시작하는 이유는 읽어온 data vector의 (1,1)부터 숫자 데이터가 있기 때문이다.

- atof는 **#include <cstdlib>**를 포함해야 사용할 수 있으며 데이터형을 변환해준다.

atof는 **character형만 변환가능하여** c\_str()함수를 사용하여 string -> char로 캐스팅한다.

- current\_value는 <float> 2차원 벡터이고, data는 <string> 2차원 벡터이므로 back()을 사용하여 2차원 벡터의 마지막 행에 push\_back을 해주는 형태이다.

- \* Cal() : 위에 설명한 함수다. 위 데이터를 할당한 것을 참조하여 계산을 진행한다.
- \* Display() : 결과를 표기해 주는 함수다. 아래에서 설명한다.

```
void Calculator::Display() { // 결과를 출력하는 함수
    cout << endl << endl;
    cout << "< 성장률 계산 결과 >" << endl << endl;
    auto growth_rate_start = growth_rate.begin(); // 벡터의 시작값을 변수에 저장 (auto로 타입 자동 선정)
    for (auto& i : week) {
        cout << "[ " << i << " ]" << endl; // 몇 주차인지 나타내는 반복문
        for (int j = 0; j < name.size(); ++j) {
            // 종목과 성장률을 나타내는 반복문 : fixed << setprecision(2) : 소수 2번째 자리까지 표기
            cout << name[j] << ": " << fixed << setprecision(2) << *growth_rate_start << "%" << endl;
            ++ growth_rate_start; // 값 증가
        }
        cout << endl;
    }
    Graph_Cal(growth_rate); // 그래프 비율을 계산하고
    Graph_Print(growth_rate, graph_rate); // 그 비율을 가지고 그래프 출력
}
```

< Display() 함수 >

원래 계획상 [STEP 5.]에서 결과를 구현하기로 했지만 계산값이 잘 이루어져 있는지 확인하는 과정이 필요하여 먼저 구현하게 되었다.

\* auto growth\_rate\_start = growth\_rate.begin()

수업시간에 배운 vector 中 begin()을 사용하여 벡터의 시작값을 초기화했다.

이유는 모르겠지만 결과값을 참조할 때 배열을 참조하듯 사용하면 값이 0으로 나오는 현상이 있다. 제대로 된 주소값을 알려주지 못해서인가? 생각되어 주소를 가리키는 방법으로 진행하기로 했다.

auto 키워드는 데이터의 형을 알아서 지정해주는 아주 편리한 친구라서 사용하였다.

\* for문 : week벡터의 데이터 주소값을 auto& (변수 i)에 가져와 반복한다.

밑의 for문은 name벡터의 사이즈만큼 반복한다.

name벡터는 배열을 참조하듯 사용해도 값이 잘 출력되기에 이렇게 사용했다.

\* fixed << setprecision(2)

#include <iomanip>을 포함해야 사용할 수 있다.

결과값을 소수 두 번째 자리까지 표기한다. (정돈된 모습을 위해 두 번째 자리까지 표기하로 하였다.)

\* ++ growth\_rate\_start

변수가 가리키는 주소를 업데이트 해주어 다음 값을 가리키게 한다.

\* Graph\_Cal(), Graph\_Print() 함수는 [STEP 5.]에서 설명한다.

## STEP 5. 결과 그래프 출력

분석값에 대한 결과를 그래프로 표기해준다.

세부 기능 1) Graph\_Cal

그래프를 그리기 위해서는 결과값을 토대로 비율을 측정해야 한다. 그것을 구현한 함수다.

세부 기능 2) Graph\_Print

색이 채워진 블록과 그렇지 않은 블록을 사용해 그래프를 만들어볼 예정이다.

세부 기능 1)부터 살펴본다.

```
void Calculator::Graph_Cal(vector<float> growth_rate) { // 그래프 비율 계산 함수
    float max_rate = 0; // 초기 최대 비율 0

    cout << endl << endl;
    cout << "< 그래프 >" << endl << endl;
    for (int i = 0; i < growth_rate.size(); i++) {
        if(fabs(growth_rate[i]) > max_rate) { // ||결과값|| 중 제일 큰 것을 저장
            max_rate = fabs(growth_rate[i]); // fabs는 float 절대값 처리
        }
    }
    for (int i = 0; i < growth_rate.size(); i++) {
        float temp_graph_rate = (growth_rate[i] / max_rate) * 20; // 그래프 비율 : (비율 / 가장 큰 비율) * 20(칸)
        graph_rate.push_back(static_cast<int>(round(temp_graph_rate))); // round()함수로 반올림 후 int형으로 배열에 저장
    }
}
```

< Graph\_Cal() 함수 >

\* float max\_rate = 0

초기화를 먼저 진행해 준다.

\* for문 : 성장비율(growth\_rate vector)의 사이즈만큼 반복한다.

- if문 : fabs (cmath) 성장비율을 절대값으로 바꿔 큰 값을 비교한다. (음수일 경우도 대비)
- 비교 후 큰 값을 절대값으로 바꿔 저장한다.

### ※ 계산식

$$\text{그래프 비율} = \frac{\text{성장비율}}{\text{성장비율 중 가장 큰 값}} \times \text{그래프 칸 개수}$$

- 마지막으로 계산된 그래프 비율을 벡터에 push back해준다.

\* graph\_rate (vector) = <int>형

\* temp\_graph\_rate = <float>형

\* 서로의 데이터형이 달라 **형변환 후** push back을 진행한다.

**round() 함수** : 반올림을 진행하는 함수다. **형변환 전에** float값을 반올림하여 정수로 만들어 준다. (올림, 내림도 있지만 오차를 최소화 하기 위해 반올림 사용) # cmath에 포함된 기능이다.

\* 수업시간에 잠깐 언급한 static\_cast<int>를 사용해 형변환을 한다.

컴파일 전에 실행하기 때문에 dynamic보다는 위험성이 있지만 결과가 잘 나오므로 진행한다.



## 세부 기능 2)

```
void Calculator::Graph_Print(vector<float> growth_rate, vector<int> graph_rate) { // 그래프 출력 함수
    int all_bar = 40; // 총 표기할 칸 수 40
    int right_bar = all_bar / 2; // 오른쪽 칸 20
    int left_bar = all_bar / 2; // 왼쪽 칸 20
    auto graph_rate_start = graph_rate.begin(); // 그래프 비율 벡터의 시작값 저장
    auto growth_rate_start = growth_rate.begin(); // 성장 비율 벡터의 시작값 저장

    for (auto& i : week) {
        cout << "[ " << i << " ]" << endl; // 몇 주차인지 나타내는 반복문
        for (int j = 0; j < name.size(); ++j) {
            cout << name[j] << " "; // 종목 이름 출력
            if (*graph_rate_start < 0) { // 성장률이 음수일 때 그래프 그리기
                for (int k = 0; k < left_bar; k++) { // 왼쪽 칸 수만큼 반복 20
                    if (k < left_bar - abs(*graph_rate_start)) // 성장률이 음수이므로 절대값 처리
                        cout << " ";
                    else
                        cout << "■";
                }
            }
            else {
                for (int k = 0; k < left_bar; k++)
                    cout << " ";
            }
            cout << "|"; // 칸 나누기 : 음수 | 양수
            for (int k = 0; k < right_bar; k++) {
                if (k < *graph_rate_start && *graph_rate_start > 0) // 성장률이 양수일 때 그래프 그리기
                    cout << "■";
                else
                    cout << " ";
            }
            cout << " (" << fixed << setprecision(2) << *growth_rate_start << "%)" << endl; // 성장 비율을 소수 2번째자리까지 출력
            growth_rate_start++; // 다음 성장률로 업데이트
            graph_rate_start++; // 다음 그래프 비율로 업데이트
        }
        cout << endl; // 줄바꿈
    }
}
```

< Graph\_Print함수 >

**auto graph\_rate\_start, growth\_rate\_start = begin();**

Display()함수에서 사용한 방법과 동일한 방법이다. 벡터의 시작값을 미리 넣어놓고 반복문이 끝날 때 업데이트 시켜주며 벡터 값을 사용하는 방법이다.

**for(auto& i : week)** : week벡터의 주소값을 주어 출력한다.

**for(int j = 0; j < name.size(); ++j)** : name vector의 크기만큼 반복하며 배열처럼 참조한다.

**if (\*graph\_rate\_start < 0)** : 그래프 성장률이 음수일 경우에 그리는 그래프다.

※ vector의 begin()은 **반복자**라서 값을 저장하지 않고 가리키는 포인터와 비슷한 역할이다.  
따라서 (**역참조 \***)를 필수로 사용해야 값을 참조할 수 있다.

**if (k < left\_bar - abs(\*graph\_rate\_start):**

(20칸 - 그래프 비율)만큼 빈 네모를 먼저 출력하는데 그 이유는 **왼쪽 그래프**는 비율의 **음수 부분**으로 가운데 "|" 분기선을 시작점으로 하여 출력돼야 하기 때문에 빈 네모를 먼저 출력시킨다.

**if (k < \*graph\_rate\_start && \*graph\_rate\_start > 0):**

그래프 비율이 양수일 때 그래프 비율만큼 k번 색이 찬 네모를 출력한다.

**fixed << setprecision(2)** : 결과값을 소수 두 번째 자리까지 출력한다.

**growth\_rate\_start ++, graph\_rate\_start ++** : 두 변수가 가리키는 주소를 업데이트한다.

## 헤더 파일

```
// Caculator Class - is-a관계로 사용 : 부모클래스
class Calculator {
protected:
    vector<vector<string>> data; // 데이터 원본을 저장할 벡터
    vector<string> name; // 종목:이름을 저장할 벡터
    vector<string> week; // 주차를 저장할 벡터
    vector<vector<float>> current_value; // 현재 주차 값을 저장할 2차원 벡터
    vector<float> first_value; // 첫 주차 값을 저장할 벡터
    vector<float> growth_rate; // 성장률을 저장할 벡터
    vector<int> graph_rate; // 그래프 비율을 저장할 벡터

public:
    // STEP 4.
    void Data_Analysis(const string& path); // 데이터 파일 분석 함수
    void Cal(); // 계산기 함수
    void Display(); // 결과를 출력하는 함수

    // STEP 5.
    void Graph_Cal(vector<float> growth_rate); // 그래프 비율 계산 함수
    void Graph_Print(vector<float> growth_rate, vector<int> graph_rate); // 그래프 출력 함수
};

// Menu 클래스 : 10/30 Class : 자식클래스
class Menu : public Calculator {
```

\* 설명

1) 계산기라는 클래스를 하나 더 만들었다.

메뉴에 넣기엔 기능적으로 차이가 있는 것 같았고, 수업시간에 배운 내용을 하나라도 더 써보자는 마음으로 만들어보았다. (부모-자식 간 상속(is-a)을 사용해 구현했다.)

2) 각 변수들의 생성 목적은 주석을 참고하면 될 것 같다.

3) 각 함수들의 내용은 위 서술과 같다.

## 추가) 입력 오류 발견 및 수정

[중간보고서#1]의 내용은 메뉴에서 제공되는 선지에 맞춰 사용자 입력을 받아 저장하는 내용이다. 이 과정에서 사용자의 입력을 저장할 때 int형 데이터 변수에 저장 받기로 되어있었다. int형 데이터 변수를 통해 if문을 구성하면 잘 돌아가긴 하지만 약간의 오류가 존재했다.

기존 if문 ex) if (number(int) == 1)

사용자의 입력 중 첫 글자에 '1'이 있으면 모두 조건문이 성립된다는 것이다. ex) 1sdjfljz

때문에 int형 데이터 변수가 아닌 string형 데이터 변수인 'input'을 사용하여 입력을 저장하는 것으로 변경하였다.

현재 if문 ex) if (input(string) == "1")

## 테스트 결과

### STEP 4. 사용자 DATA 분석

사용자의 데이터 값을 분석한다. 대단한 분석법을 쓰는 것이 아니고 그냥 상승하는 정도치만 계산한다.

세부 기능 1)

입력받은 자료의 차이를 백분율로 바꿔서 계산시킬 것 같다.

세부 기능 2)

cout를 사용해서 결과를 출력한다.

#### < 성장률 계산 결과 >

[ week1 ]

Leg: 0.00%

Shoulder: 0.00%

Back: 0.00%

Chest: 0.00%

[ week2 ]

Leg: 5.48%

Shoulder: 24.78%

Back: 23.06%

Chest: 25.58%

[ week3 ]

Leg: -29.68%

Shoulder: 22.20%

Back: -16.38%

Chest: 1.72%

첫 주는 계산법에 따라 (week1 data - week1 data)가 되기 때문에 0으로 나온다. 비교군이 없으니 0으로 나오는 것이 당연하다. 세부 기능 1, 2를 한번에 보여주는 그림이다.

※ 파일은 test\_read.csv 파일로 읽어왔다. 엑셀 파일은 글자가 깨지는 현상이 발생한다.

## STEP 5. 결과

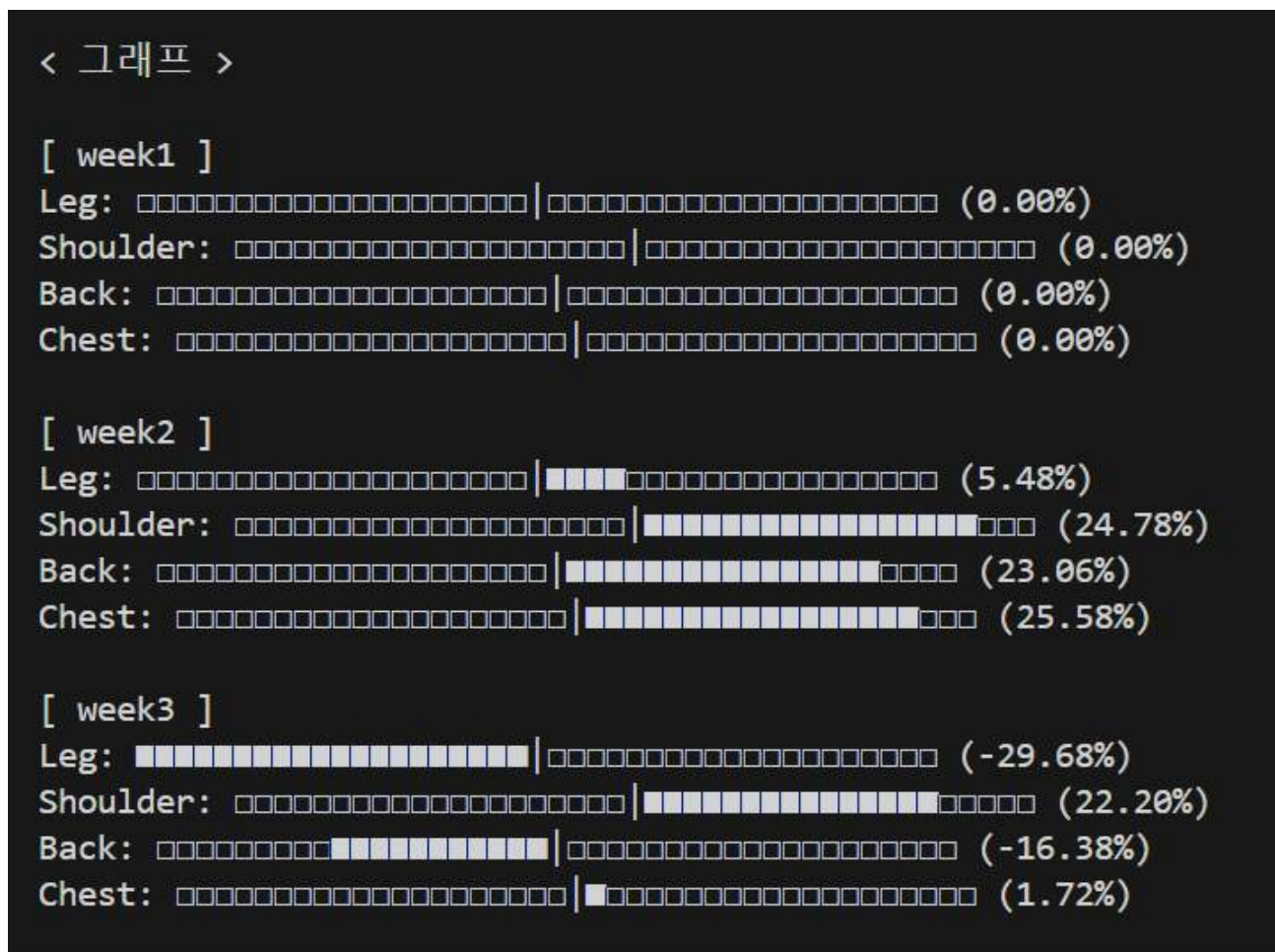
분석값에 대한 결과를 그래프로 표기해준다.

세부 기능 1) Graph\_Cal

그래프를 그리기 위해서는 결과값을 토대로 비율을 측정해야 한다. 그것을 구현한 함수다.

### 세부 기능 2) graph

색이 채워진 블록과 그렇지 않은 블록을 사용해 그래프를 만들어볼 예정이다.



- 1) 성장률 계산 결과를 가지고 그래프 비율로 계산하여 그 수만큼 블록을 출력하여 만든다.
- 2) 그래프 비율 계산법에 맞게 가장 높은 절대값을 가진 (-29.68)은 전부 칸이 가득 차 있는 모습을 확인할 수 있다.
- 3) 음수와 양수를 가르는 | 표기를 시작점으로 하여 그래프가 출력되는 것도 역시 확인 가능하다.
- 4) 그래프 칸 비율 결과, 주, 부위, 성장 수치 역시 계산대로 정확하게 출력되고 있는 모습이다.

## 계획 대비 변경 사항

### - 변경 前 : STEP 3. 사용자 입력 저장 中

세부 기능 2)

사용자로부터 입력 받기

세부 기능 3)

만일 숫자가 아닌 다른 값이 입력된다면 경고문을 출력하고 다시 입력으로 돌아온다.

### - 변경 後 : STEP 3. 사용자 입력 저장 中

세부 기능 2)

경로를 입력받고 출력하기 ( 경로를 입력받고 해당 파일을 출력한다. )

세부 기능 3)

만일 경로를 읽을 수 없는 경우에는 경고문을 보내고 다시 입력으로 돌아온다.

### - 변경 사항과 사유

직접 입력하여 수치를 저장하는 방식에서 파일을 읽어 수치를 저장하는 방식으로 변경.

우선 사용자의 직접입력은 상당한 시간 소요가 발생하므로 변경하기로 했다. (단점 보완)

### - 변경 前 : STEP 4. 데이터 분석 中

세부 기능 1)

입력받은 자료의 차이를 백분율로 바꿔서 계산시킬 것 같다.

### - 변경 後 : STEP 4. 데이터 분석 中

세부 기능 1)

입력받은 자료의 차이를 백분율로 바꿔서 계산시킬 것 같다.

세부 기능 2) - ADD

cout 기능을 사용해서 출력한다. (새로이 추가된 항목 : STEP 5.에서 STEP 4.로 이동)

### - 변경 사항과 사유

계산 결과값이 잘 나오는지 확인을 위해 미리 구현하였다.

### - 변경 前 : STEP 5. 결과 中

세부 기능 1) cout

그냥 cout 기능을 사용해서 출력한다.

세부 기능 2) graph

색이 채워진 블록과 그렇지 않은 블록을 사용해 그래프를 만들어볼 예정이다.

### - 변경 後 : STEP 5. 결과 中

세부 기능 1) Graph\_Cal

그래프를 그리기 위해서는 결과값을 토대로 비율을 측정해야 한다. 그것을 구현한 함수다.

세부 기능 2) graph

색이 채워진 블록과 그렇지 않은 블록을 사용해 그래프를 만들어볼 예정이다.

### - 변경 사항과 사유

세부 기능 1)은 STEP 4.로 옮겨갔고, 그 빈자리를 비율 계산 함수로 덮어씌웠다.

세부 기능으로 나눌 정도인가 싶어서 안 만들었는데 빈 자리가 생겨 만들었다.

## 프로젝트 일정

(표 참고)

업무	11/3	11/10	11/13	11/16	11/23	11/30	12/2
제안서 작성	完						
STEP 1		完					
STEP 2			完				
STEP 3				完			
STEP 4					完		
STEP 5						完	
최종 점검							