C++ 프로그래밍 및 실습

바이오차 혼입 시멘트 실험 도우미

최종 보고서

제출일자: 12/24
제출자명: 박경준
제출자학번: 192195

1. 프로젝트 목표

1) 배경 및 필요성

현재 조경학과에 속한 조경공학연구실 학부연구생으로 '바이오차 혼입 시멘트(콘크리트) 개발을 위한 실험'을 진행 중이다. 해당 실험은 탄소 절감을 목적으로 기존 (포장)콘크리트에 바이오차를 혼입하여 최적의 대체율을 찾는 실험이다. 이에 따른 재료들의 혼합을 하기 전 바이오차 대체율에 따른 각각의 재료들의 양 설정이 중요하다. 또한, 대체율에 따른 테스트 값들을 통해 분석을 진행할 수 있다.

2) 프로젝트 목표

프로젝트를 통해 직접 계산하던 방식에서 벗어나 준비과정부터 테스트 값들을 통한 유추를 자동화하여 실험 시간 단축을 목표로 한다.

3) 차별점

해당 프로젝트는 실험을 진행하면서 시간 소요가 있던 과정을 경험하고 필요에 의해서 만든 프로젝트로 최종적으로 단순히 값을 출력하는 것이 아닌 값의 범위를 통해서 대체율의 적합성 결과를 출력하여 차별성이 존재한다.

2. 기능 계획

1) 대체율에 따른 재료의 양 설정

- 바이오차 대체율(0~6%)에 따른 물, 시멘트, 잔골재의 양 설정

(1) 대체율에 따른 실험체(공시체)명 설정

- 바이오차는 시멘트를 대체, 즉 바이오차와 시멘트는 값이 유동적으로 변함

- 반면, 물과 잔골재의 양은 고정값

2) 시간대 별 Flow Test 값

- 0 ~ 40min, 각 10분 간격으로 Flow Test 진행하며 나온 값을 입력 후 150mm 기준으로 적합/부적합 결정

3) 강도실험 값

- 압축강도, 쪼갬인장강도, 휨강도 실험을 통해서 얻은 값들을 입력하여 평균값(KN)을 얻은 뒤 Mpa로 변환 후 설계 기준 강도를 통한 적합/부적합 결정

4) 적합성 파악

- Flow-Test 및 3개의 강도 실험을 통해 얻은 값을 입력 후 기준에 맞게 적합성 파악을 하여 적합/부적합 문자열 출력

5) 데이터들 최종 출력

- 대체율, 실험체 명, Flow Test, 강도 적합성를 표 형식으로 출력

6) 표 형식 텍스트 파일 저장

- 표 형식의 값들과 키워드를 experiment_data.txt 파일에 저장

- 실행할 경우 키워드는 고정으로 유지되고 값만 갱신된다.

3. 기능 구현

(1) 바이오차 대체율 입력에 따른 실험체명 및 재료양 출력

- 대체율 입력, 재료양 출력

- main.cpp에서의 getBioRateFromUser() 함수를 통해 사용자에게 대체율(0~6%)를 입력 받은 뒤 이를 기반으로 experiment.cpp에서의 재료양과 관련된 함수들이 실행되어 실험체명과 재료양이 출력된다.

- 클래스 멤버 함수, 함수 정의, 입출력, 연산자, 반복문, 조건문

- 코드 스크린샷

```
// 사용자에게 바이오차 대체율을 입력받는 함수
int getBioRateFromUser() {
    int bioRate;
    while (true) {
        cout << "바이오차 대체율(%)을 입력하시오 : ";
        cin >> bioRate;
        if (bioRate < 0 or bioRate > 6) {
            cout << "잘못된 입력입니다. 바이오차 대체율을 0 ~ 6% 사이의 값으로 입력해주세요." << endl;
            continue;
        }
        break;
    }
    return bioRate;
}
```

```cpp
// 바이오차 대체율에 따른 시멘트양 계산 함수
void ExperimentData::calculateCement()
{
    double bioAsh = bioRate * 48.5; // 바이오차 1%당 바이오차 양(g)

        // 바이오차에 따른 시멘트 양 계산
    if (bioRate == 0.01) {
        cement = 4.80 - bioAsh / 1000.0;
    }       You, 43 minutes ago • refactor: 최종 코드 리팩토링
    else if (bioRate == 0.02) {
        cement = 4.75 - bioAsh / 1000.0;
    }
    else if (bioRate == 0.03) {
        cement = 4.71 - bioAsh / 1000.0;
    }
    else if (bioRate == 0.04) {
        cement = 4.66 - bioAsh / 1000.0;
    }
    else if (bioRate == 0.05) {
        cement = 4.56 - bioAsh / 1000.0;
    }
    else if (bioRate == 0.06) {
        cement = 4.56 - bioAsh / 1000.0;
    }
}
```

```cpp
// 사용자에게 바이오차 대체율을 입력받는 함수의 유효성 판별
// 바이오차는 0 ~ 6%까지 사용
int ExperimentData::getBioRate() const
{
    int bioRate;
    while (true) {
        cout << "바이오차 대체율을(%)을 입력하시오 : ";
        cin >> bioRate;
        if (bioRate < 0 or bioRate > 6) {
            cout << "잘못된 입력입니다. 바이오차 대체율을 0 ~ 6% 사이의 값으로 입력해주세요." << endl;
            continue;
        }
        break;
    }
    return bioRate;
}

void ExperimentData::printSpecimenName()
{
    cout << "실험체(공시체)명 : " << getSpecimenName() << endl;
}


// 바이오차 대체율에 따라 시멘트 양이 결정
// 물과 잔골재양은 고정값
void ExperimentData::printMaterials()
{
    double bioAsh = getBioAsh();
    double cement = getCement(); // 시멘트 양을 그대로 가져옴

    cout << "바이오차 양 : " << bioAsh << "g" << endl;
    cout << "시멘트 양 : " << fixed << setprecision(2) << cement << "kg" << endl; // 시멘트 양 반올
    cout << "물 양 : 2.29Kg" << endl; // 고정값
    cout << "잔골재 양 : 11.04kg" << endl; // 고정값
}
```

(2) 시간대 별 Flow-Test 값 입력 및 유효성 검사

- 5개의 Flow-Test값 입력, 유효성 검사결과 반환

- 입력한 0~40분의 10분 간격 Flow-Test 값을 기반으로 5개 중 하나의 값이라도 150mm 미만의 값이 있다면 부적합을 얻는다.

- 문자열, string stream, 배열, 반복문, 조건문, 멤버 함수

- 코드 스크린샷

```
// 실험 결과의 적합성을 알기 위해 플로우 테스트 값을 확인
// 0 ~ 40분 까지의 5개의 값 입력 및 유효성 검사 실시
// 플로우 테스트 값 중 하나라도 150mm 이하의 값을 가지면 적합하지 않는 코드 작성
void ExperimentData::saveFlowTest()
{
    string input;
    double flowTestValues[5];
    // Flow-Test 시간은 고정값, 0은 직후 Flow-Test 값
    int times[5] = { 0, 10, 20, 30, 40 };

    // 유효성 검사를 위한 while문
    while (true) {
        cout << "10분 간격의 시간대 별 (0분 ~ 40분) Flow-Test 값을 공백으로 구분하여 mm단위로 입력하세요 :";
        cin.ignore();
        // 입력 한 줄로 받기
        getline(cin, input);

        // 입력 stringstream에 저장
        stringstream ss(input);

        // validInput 변수를 통한 유효성 검사
        // 시간 대 별 5개의 입력을 작성하지 않았을 경우
        bool validInput = true;
        for (int i = 0; i < 5; i++) {
            if (!(ss >> flowTestValues[i])) {
                validInput = false;
                cout << "유효하지 않은 입력입니다. 숫자를 공백으로 구분하여 5개 입력하세요." << endl;
                break;
            }
        }

        if (validInput) {
            break;
        }
    }

    bool isFlowTestSuitable = true; // 기본적으로 적합으로 설정
```

```
    // 150mm 이상이면 적합으로 설정
    for (int i = 0; i < 5; i++) {
        if (flowTestValues[i] >= 150) {
            isFlowTestSuitable = false;
            break;
        }
    }

    // 적합 여부를 ExperimentData 클래스에 저장
    setFlowTestSuitability(isFlowTestSuitable);
}
```

(3) 세 가지 강도 값 입력 및 유효성 검사

- 세 가지 강도의 값(3개) 입력, 강도 적합성 반환

- 압축강도, 쪼갬인장강도, 휨강도 각각의 3개의 실험 결과를 입력하면 수식을 통한 평균값을 계산 한다. 이후 KN단위의 값을 Mpa 단위로 바꾸는 수식을 적용하여 이 값이 사전에 설정한 설계 기준 강도에 못 미치면 부적합을 출력한다.

- 조건문, 반복문, 문자열, 연산자, 멤버 변수

- 코드 스크린샷

```
// 실험 결과의 적합성을 알기 위해 3개의 강도 값을 확인
// 압축, 쪼갬인장, 휨강도 각각의 3개의 값을 입력 및 유효성 검사 실시
void ExperimentData::saveStrength(const string& strengthType)
{
    string input;
    // 강도 입력 리스트
    double strengthValues[3];

    // 유효성 검사를 위한 while문
    while (true) {
        cout << "실험체의 " << strengthType << "를 공백으로 구분하여 KN 단위(실수)로 3개의 값을 입력하세요 :
        cin.ignore();
        getline(cin, input);

        stringstream ss(input);

        // 유효성 검사를 위한 vaildInput 설정
        bool validInput = true;
        for (int i = 0; i < 3; i++) {
            if (!(ss >> strengthValues[i])) {
                validInput = false;
                cout << "유효하지 않은 입력입니다. 숫자(실수)를 공백으로 구분하여 입력하세요." << endl;
                break;
            }
        }

        if (validInput) {
            break;
        }
    }
```

```
    double sum = 0;
    for (int i = 0; i < 3; i++) {
        sum += strengthValues[i];
    }

    double average = sum / 3;

    // 적합성 파악을 위한 평균값 변환
    double transformedValue = (average * 1000) / (50 * 50);

    // 강도에 따라서 setter 함수에 평균값 저장
    if (strengthType == "압축강도") {
        setCompressiveStrength(transformedValue);
    }
    else if (strengthType == "쪼갬인장강도") {
        setTensileStrength(transformedValue);
    }
    else if (strengthType == "휨강도") {
        setFlexuralStrength(transformedValue);
    }
}
```

```
// 입력받은 강도 값이 적합한지 파악하는 함수
// 설계 기준 강도 이상의 값을 가진다면 적합
void ExperimentData::checkStrengthSuitability()
{
    // 설계 기준 강도 설정(Mpa)
    double standardCompressiveStrength = 17.89; // 기준 압축강도
    double standardTensileStrength = 2.5;      // 기준 쪼갬인장강도
    double standardFlexuralStrength = 4.5;     // 기준 휨강도

    // 실험 결과
    double compressiveStrength = getCompressiveStrength();
    double tensileStrength = getTensileStrength();
    double flexuralStrength = getFlexuralStrength();

    // 각각의 적합 여부 저장
    // 강도가 설계 기준 강도값 이상이면 적합
    bool isCompressiveSuitable = (compressiveStrength >= standardCompressiveStrength);
    bool isTensileSuitable = (tensileStrength >= standardTensileStrength);
    bool isFlexuralSuitable = (flexuralStrength >= standardFlexuralStrength);

    // 적합 여부를 ExperimentData 클래스에 저장
    setCompressiveSuitability(isCompressiveSuitable);
    setTensileSuitability(isTensileSuitable);
    setFlexuralSuitability(isFlexuralSuitable);
}
```

(4) 데이터 및 적합성 표 형식으로 출력

- 위의 기능들로 얻어진 재료양 및 적합성 데이터를 출력

- 재료양과 적합성 데이터를 최종적으로 표 형식으로 출력한다. Iomanip 라이브러리를 통해 공백을 조절하였다.

- 문자열 출력, 멤버 변수, iomanip의 setw()함수

- 코드 스크린샷

```
// 최종적으로 표 형태로 데이터들을 출력
void ExperimentData::printExperimentData()
{
    // 컬럼명(keyword) 출력
    cout << left << setw(18) << "실험체명"
        << left << setw(33) << "바이오차 대체율(%)"
        << left << setw(23) << "시멘트 양(kg)"
        << left << setw(18) << "물 양(kg)"
        << left << setw(23) << "잔골재 양(kg)"
        << left << setw(23) << "플로우 테스트"
        << left << setw(17) << "압축강도"
        << left << setw(23) << "쪼갬인장강도"
        << left << setw(18) << "휨강도" << endl;

    // 구분선 출력
    for (int i = 0; i < 150; i++) cout << '-';
    cout << '\n';

    // 값 출력
    cout << left << setw(21) << getSpecimenName()
        << left << setw(22) << static_cast<int>(getBioRate())
        << left << setw(18) << getCement()
        << left << setw(16) << getWater()
        << left << setw(21) << getFineAggregate()
        << left << setw(16) << getFlowTestSuitabilityString()
        << left << setw(18) << (getCompressiveSuitability() ? "적합" : "부적합")
        << left << setw(17) << (getTensileSuitability() ? "적합" : "부적합")
        << left << setw(16) << (getFlexuralSuitability() ? "적합" : "부적합") << endl;
}
```

(5) 표 데이터 값 텍스트 파일에 저장

- 텍스트 파일에 표 데이터가 저장

- 표 형식의 데이터를 텍스트 파일에 저장한다. 키워드 값이 만약에 텍스트 파일에 존재하면 값만 갱신되도록 설정하였다.

- fstream라이브러리의 파일 입/출력, 문자열 출력, 멤버 변수, iomanip의 setw()함수

- 코드 스크린샷

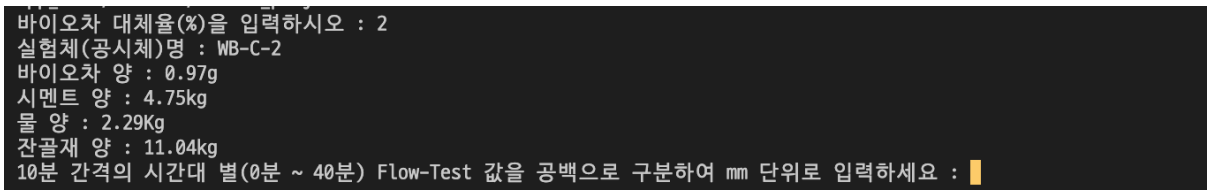
```cpp
// 표로 출력한 내용을 텍스트 파일에 저장
void ExperimentData::saveExperimentDataToFile()
{
    // 파일에 추가 모드로 열기
    ofstream outputFile("experiment_data.txt", ios_base::app);

    if (outputFile.is_open()) {
        // 이전에 표 헤더가 파일에 쓰여졌는지 확인 후 추가하지 않는다
        if (outputFile.tellp() == 0) {
            outputFile << left << setw(18) << "실험체명"
                << left << setw(33) << "바이오차 대체율(%)"
                << left << setw(23) << "시멘트 양(kg)"
                << left << setw(18) << "물 양(kg)"
                << left << setw(23) << "잔골재 양(kg)"
                << left << setw(23) << "플로우 테스트"
                << left << setw(17) << "압축강도"
                << left << setw(23) << "쪼갬인장강도"
                << left << setw(18) << "휨강도" << endl;

            for (int i = 0; i < 130; i++) outputFile << '-';
            outputFile << '\n';
        }

        outputFile << left << setw(17) << getSpecimenName()
            << left << setw(20) << static_cast<int>(getBioRate())
            << fixed << setprecision(2) << left << setw(15) << getCement()
            << left << setw(16) << getWater()
            << left << setw(17) << getFineAggregate()
            << left << setw(15) << getFlowTestSuitabilityString()
            << left << setw(17) << (getCompressiveSuitability() ? "적합" : "부적합")
            << left << setw(16) << (getTensileSuitability() ? "적합" : "부적합")
            << left << setw(16) << (getFlexuralSuitability() ? "적합" : "부적합") << endl;

        outputFile.close();
        cout << "데이터가 파일에 추가되었습니다." << endl;
    }
    else {
        cout << "파일을 열 수 없습니다." << endl;
    }
```

4. 테스트 결과

(1) 바이오차 대체율 입력에 따른 실험체명 및 재료양 출력

- 사용자가 입력한 값에 따라 출력 설정, '2'를 입력하여 실험체명이 WB-C-2가 되었다.

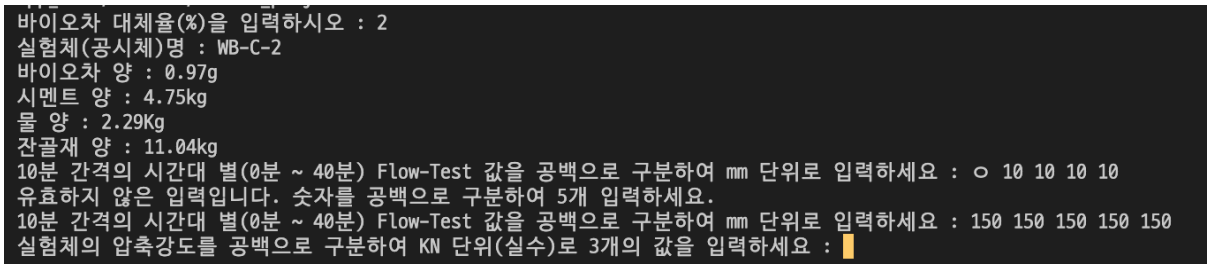
- 테스트 결과 스크린샷



(2) 시간대 별 Flow-Test 값 입력 및 유효성 검사

- 공백으로 시간대 별 값들을 입력한 후 적합성이 저장된다. 또한 올바른 형식으로 입력하지 않았을 경우 유효성 검사에 걸려 설정된 문자열이 출력된다.

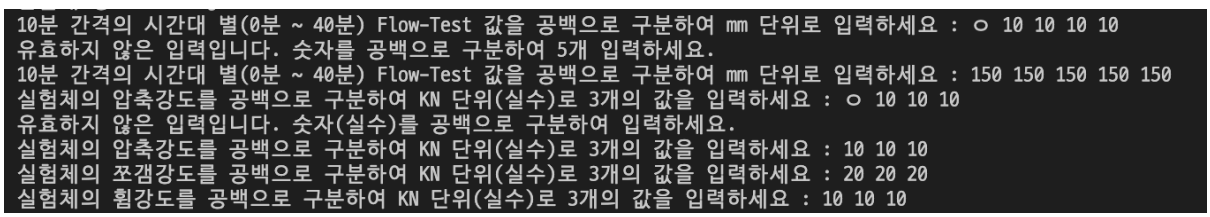
- 테스트 결과 스크린샷



(3) 세 가지 강도 값 입력 및 유효성 검사

- 압축강도, 쪼갬인장강도, 휨강도 각각의 3개의 값을 입력한 후 적합성이 저장된다. 또한 올바른 형식으로 입력하지 않았을 경우 유효성 검사에 걸려 설정된 문자열이 출력된다.

- 테스트 결과 스크린샷



(4) 데이터 및 적합성 표 형식으로 출력

- 최종적으로 표 형식으로 데이터를 출력한다.

- 테스트 결과 스크린샷

```
실험체의 압축강도를 공백으로 구분하여 KN 단위(실수)로 3개의 값을 입력하세요 : ㅇ 10 10 10
유효하지 않은 입력입니다. 숫자(실수)를 공백으로 구분하여 입력하세요.
실험체의 압축강도를 공백으로 구분하여 KN 단위(실수)로 3개의 값을 입력하세요 : 10 10 10
실험체의 쪼갬강도를 공백으로 구분하여 KN 단위(실수)로 3개의 값을 입력하세요 : 20 20 20
실험체의 휨강도를 공백으로 구분하여 KN 단위(실수)로 3개의 값을 입력하세요 : 10 10 10
실험체명      바이오차 대체율(%)      시멘트 양(kg)      물 양(kg)      잔골재 양(kg)      플로우 테스트      압축강도      쪼갬인장강도      휨강도
-------------------------------------------------------------------------------------------------------------------------------------
WB-C-2              2                 4.75              2.29              11.04              적합              부적합              부적합              부적합
```

(5) 표 데이터 값 텍스트 파일에 저장

- 표 출력 이후에 '데이터가 파일에 추가되었습니다' 문장과 함께 텍스트 파일에 표 형식의 데이터가 저장된다.

- 테스트 결과 스크린샷

```
실험체명      바이오차 대체율(%)      시멘트 양(kg)      물 양(kg)      잔골재 양(kg)      플로우 테스트      압축강도      쪼갬인장강도      휨강도
-------------------------------------------------------------------------------------------------------------------------------------
WB-C-2              2                 4.75              2.29              11.04              적합              부적합              부적합              부적합
데이터가 파일에 추가되었습니다.
```

experiment_data.txt

```
실험체명      바이오차 대체율(%)      시멘트 양(kg)      물 양(kg)      잔골재 양(kg)      플로우 테스트      압축강도      쪼갬인장강도      휨강도
-------------------------------------------------------------------------------------------------------------------------------------
WB-C-5              5                 4.56              2.29              11.04              적합              부적합              부적합              부적합
WB-C-3              3                 4.71              2.29              11.04              적합              부적합              부적합              부적합
WB-C-4              4                 4.66              2.29              11.04              적합              부적합              부적합              부적합
WB-C-2              2                 4.75              2.29              11.04              적합              부적합              부적합              부적합
```

5. 계획 대비 변경 사항

1) 플로우 테스트 유효성 검사

- 입력받은 Flow-Test값을 평균내어 평균값만 출력

- 기준값(mm)을 설정하여 값이 기준값이 넘는다면 부적합을 설정하여 적합성을 출력

- 연구실 실험 결과 방식이 바뀜에 따라 적용

2) 표 형식 텍스트 파일 저장

- 표 형식 데이터 terminal에서 출력

- experiment_data.txt에 저장되도록 설정

- 실행하고 난 뒤 결과값을 저장하는게 실험하는 입장에서 편리하기 때문이다.

6. 느낀점

프로젝트를 진행하며 프로그래밍 언어의 습득에 있어 실전 경험이 중요함을 알게 되었습니다. 강의를 통해 이론을 배우는 것도 중요하지만, 실제 문제에 부딪혀보며 시행착오를 겪는 과정에서 이해력이 크게 향상되었습니다. 또한 프로그래밍을 통해 실험 결과를 자동화하는 경험을 통해, 앞으로 경험할 다양한 실험뿐만 아니라 일상생활에서도 자동화가 가능하다는 것을 인지하였습니다.

프로젝트를 진행하는 과정에서 가장 어려웠던 부분은 코드 리팩토링이었습니다. 단일 파일에서 모든 작업을 처리하던 것을 최종 제출을 위해 여러 파일로 분리하는 과정에서 다양한 오류에 직면했습니다. 많이 시도 끝에 세 개의 파일로 나누어 main.cpp를 실행시켜 원활하게 프로그램이 작동하게 만든 순간은 기뻤습니다.

이 경험이 다음 프로젝트에서 발생할 수 있는 에러 해결에 큰 도움이 될 것이라도 생각합니다.