

c++ 프로그래밍 및 실습

# 맞왜틀?

진척 보고서 #3

제출일자:2024-12-15

제출자명:최영현

제출자학번:213255

## **1. 프로젝트 목표 (16 pt)**

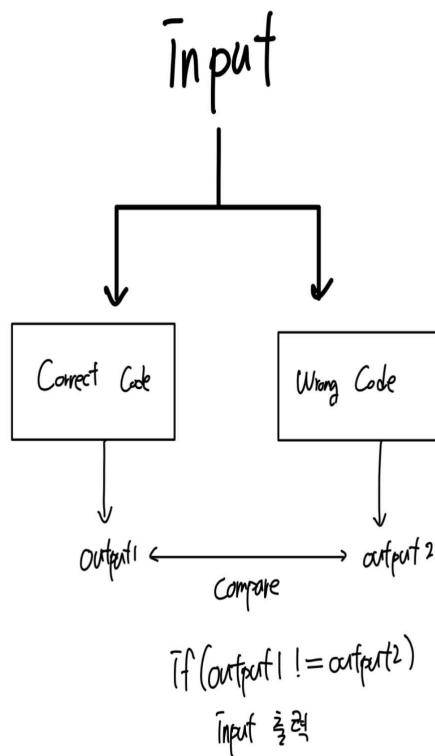
### **1) 배경 및 필요성 (14 pt)**

c++수업을 들으며 내가 남들보다 코딩실력이 뒤떨어진다는 생각이 들어 프로그래밍 문제 사이트인 백준에 들어가 문제들을 조금씩 풀기 시작했다. 그렇게 여러 문제를 풀다보면 분명히 맞게 했다고 생각하고 제출했는데 틀리는 경우가 많았다. 아 진짜.... 맞는데 왜 틀리다는거야??????!!!!!(맞왜틀)

문제에 있는 예제도 다 넣어보고 출력값이 정상인 것을 확인하고 입력의 최소값, 최대값을 넣어보고 정상인 것을 확인하고 코드를 처음부터 쭉읽어봤는데 도저히 뭐가 문제인지 모르겠을 때 반례를 찾아주는 프로그램이 있으면 좋지 않을까 생각하게 되었다.

### **2) 프로젝트 목표**

인터넷에 있는 정답 코드와 내 코드를 입력하고 다른 출력을 만드는 입력값을 찾아낸다.



### 3) 차별점

인터넷에 백준 반례 사이트라고 치면 맨 위에 Testcase AC라고 하나가 있다.

하지만 이 사이트는 백준의 200여개의 문제에 대한 반례만 제공한다.

그래서 내가 만들려고 하는것은 정답 코드만 있다면 어떤 프로그래밍 문제 사이트 출처인지 상관없이 반례를 찾아주는 프로그램이다.

## 2. 기능 계획

### 1) 기능 1 wrong code,correct code() 출력 비교

두 코드에 동일한 입력을 넣고 다른 출력이 나오면(반례) 그 입력값을 출력하는

기능.

세부기능1: wrong code와 correct code를 입력받아 파일에 저장하고 실행 명령을 내리면 컴파일하고 출력을 반환하는 기능

세부기능2: 두 코드에 들어갈 입력값을 만들어주는 코드를 통해 입력->둘의 출력값 비교 무한반복(찾을때까지)기능

### 3. 진척사항

#### 1) 기능 구현

##### (1) wrong code,correct code(입력이 1개인) 출력 비교

```
// 사용자로부터 올바른 코드 입력 받기
cout << "올바른 코드의 내용을 입력하세요. (종료하려면 'done' 입력): " << endl;
string line;
while (getline(cin, line)) {
    if (line == "done") break; // 'done'을 입력하면 종료
    correctCode += line + "\n"; // 입력받은 각 줄을 코드에 추가
}

// 사용자로부터 잘못된 코드 입력 받기
cout << "잘못된 코드의 내용을 입력하세요. (종료하려면 'done' 입력): " << endl;
while (getline(cin, line)) {
    if (line == "done") break; // 'done'을 입력하면 종료
    wrongCode += line + "\n"; // 입력받은 각 줄을 코드에 추가
}
```

두 코드를 입력받고(done을 입력하면 코드 입력이 종료된다.)

```
// 주어진 코드 내용을 지정된 파일에 작성하는 함수
void writeCodeToFile(const string& code, const string& filename) {
    ofstream outFile(filename);
    if (outFile.is_open()) {
        outFile << code; // 파일에 코드 작성
        outFile.close(); // 파일 닫기
    } else {
        cout << "파일을 열 수 없습니다!" << endl;
    }
}
```

파일 작성 함수를 통해

```
// 입력받은 코드를 파일에 작성
writeCodeToFile(correctCode, "correct_program.cpp");
writeCodeToFile(wrongCode, "wrong_program.cpp");
```

파일로 저장한다.

```
// 두 코드 파일을 컴파일
system("g++ correct_program.cpp -o correct_program.exe"); // 올바른 코드 컴파일
system("g++ wrong_program.cpp -o wrong_program.exe"); // 잘못된 코드 컴파일
```

시스템 명령어를 통해 두 코드를 각각 컴파일하여 실행 파일을 생성한다.

```
// 사용자로부터 테스트할 범위 범위 받기
cout << "테스트할 최소 입력값을 입력하세요: ";
cin >> minInput; // 최소 입력값
cout << "테스트할 최대 입력값을 입력하세요: ";
cin >> maxInput; // 최대 입력값
```

테스트할 범위를 설정한다.

```
// 입력값을 범위 내에서 하나씩 테스트
for (int i = minInput; i <= maxInput; ++i) {
    // 올바른 코드와 잘못된 코드를 실행하고 출력 결과를 가져오기
    string correctOutput = runProgram("correct_program.exe", i);
    string wrongOutput = runProgram("wrong_program.exe", i);

    // 출력값 비교
    if (correctOutput != wrongOutput) {
        // 출력값이 다르면 반례 발견
        cout << "입력에 대한 반례 발견!" << endl;
        cout << "입력값: " << i << endl; // 입력값 출력
        cout << "올바른 코드 출력: " << correctOutput << endl; // 올바른 코드 출력
        cout << "잘못된 코드 출력: " << wrongOutput << endl; // 잘못된 코드 출력
        return 0; // 반례가 발견되면 프로그램 종료
    }
}

// 반례를 찾지 못한 경우
cout << "반례없음" << endl;
return 0;
```

설정한 범위에서 하나씩 테스트한다. runProgram함수를 통해 각 프로그램에 동일한 입력값을 넣고 만약 다른 출력이 나온다면 입력값과 두 출력값을 출력하며 프로그램을 종료한다. 반례를 찾지못하고 반복문이 종료되면 "반례없음"을 출력한다.

```

// 프로그램 실행 함수
// 입력값을 파일로 전달하고 해당 프로그램을 실행한 후 결과를 문자열로 반환
string runProgram(const string& filename, int input) {
    // 입력값을 test_input.txt에 기록
    ofstream inputFile("test_input.txt");
    inputFile << input << endl; // 입력값을 파일에 작성
    inputFile.close();

    // 프로그램 실행 명령어 생성
    string command = filename + " < test_input.txt 2>&1"; // 입력값을 파일에서 읽고, 표준 출력을 캡처
    char buffer[128];
    string result = "";
    FILE* pipe = _popen(command.c_str(), "r"); // 명령어 실행 결과를 읽을 파일 생성
    if (!pipe) {
        return "Error opening program."; // 파일 생성 실패 시 에러 메시지 반환
    }

    // 파일에서 데이터를 읽어서 결과에 추가
    while (fgets(buffer, sizeof(buffer), pipe) != nullptr) {
        result += buffer;
    }

    _pclose(pipe); // 파일 닫기
    return result; // 프로그램 실행 결과 반환
}

```

프로그램 실행함수

### - 적용된 배운 내용

```
string runProgram(const string& filename, int input)
```

```
void writeCodeToFile(const string& code, const string& filename)
```

함수 사용, string 사용, 매개변수에서 참조연산자 사용.

```

for (int i = minInput; i <= maxInput; ++i)
    if (correctOutput != wrongOutput)

```

반복문, 조건문 사용.

## 2) 테스트 결과

### (1) wrong code, correct code(입력이 1개인) 출력 비교

이 프로젝트를 만드는 동기가 되었던 [백준 1193번](#) 문제를 이용하여 테스트 하였다.

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

int main(void){
    long long X;
    cin>>X;
    int temp;
    int a;
    for(int i=0;i<100000;i++){
        if((i+1)*(i+2)/2>=X){
            temp=i+1;
            a=X-i*(i+1)/2;
            break;
        }

    if(X==1)
        cout<<"1/1";
    else if(temp%2==0)
        cout<<a<<"/"<<temp+1-a;
    else
        cout<<temp+1-a<<"/"<<a;
    }
    return 0;
}
```

correct\_program

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

int main(void){
    long long X;
    cin>>X;
    long long temp;
    long long horse;
    long long k;

    for(int i=1;i<100000001;i++){
        if((i*(i+1)/2) > X){
            temp=X-(i-1)*i/2;
            horse=i%2;
            k=i;
            break;
        }
    }
    if(X==1)
        cout<<"1/1";
    else if(X==2)
        cout<<"1/2";
    else if(X==3)
        cout<<"2/1";
    else if(horse==0){//pp 1/ipp
        cout<<temp<<"/"<<k-temp+1;
    }
    else//?? i/1??
        cout<<k-temp+1<<"/"<<temp;

    return 0;
}
```

wrong\_program(내가 실제로 틀렸던 코드이다)

올바른 코드의 내용을 입력하세요. (종료하려면 'done' 입력):

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

int main(void){
    long long X;
    cin>>X;
    int temp;
    int a;
    for(int i=0;i<100000;i++)
        if((i+1)*(i+2)/2>=X){
            temp=i+1;
            a=X-i*(i+1)/2;
            break;
        }

    if(X==1)
        cout<<"1/1";
    else if(temp%2==0)
        cout<<a<<"/"<<temp+1-a;
    else
        cout<<temp+1-a<<"/"<<a;

    return 0;
}
done
```

올바른 코드 입력

잘못된 코드의 내용을 입력하세요. (종료하려면 'done' 입력):

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

int main(void){
    long long X;
    cin>>X;
    long long temp;
    long long horse;
    long long k;

    for(int i=1;i<10000001;i++){
        if((i*(i+1)/2) > X){
            temp=X-(i-1)*i/2;
            horse=i%2;
            k=i;
            break;
        }
    }
    if(X==1)
        cout<<"1/1";
    else if(X==2)
        cout<<"1/2";
    else if(X==3)
        cout<<"2/1";
    else if(horse==0){//pp 1/ipp
        cout<<temp<<"/"<<k-temp+1;
    }
    else//  i/1 //
        cout<<k-temp+1<<"/"<<temp;

    return 0;
}
done
```

## 잘못된 코드 입력

```
테스트할 최소 입력값을 입력하세요: 1
테스트할 최대 입력값을 입력하세요: 10
입력에 대한 반례 발견!
입력값: 6
올바른 코드 출력: 1/3
잘못된 코드 출력: 0/5
```

범위 입력, 성공적으로 반례를 찾아내었다.

반례를 찾지 못했을 때 출력도 확인하기 위해 두 코드 모두 corrcet\_program으로 입력하고 범위를 길게하여 실행 시켜 보았다.

```
테스트할 최소 입력값을 입력하세요 : 1  
테스트할 최대 입력값을 입력하세요 : 1000  
반례없음
```

반례없음 출력

#### 4. 계획 대비 변경 사항

##### 기능 추가-->기능 1보완(세부기능1,2)

프로젝트 기획 할 때 무슨 기능이 더 필요할지 딱히 생각이 안나서 일정에 일단 기능추가로 표기해 놓았는데 진행 하다보니 기능 1에 부족한 부분이 있어서 기능 **1보완**으로 변경했다.

##### 세부 기능 1) 입력이 복잡한 코드 비교 기능 추가

현재 코드는 입력이 한개인 단순한 코드에서만 반례찾기 기능을 수행할 수 있기 때문에

## 입력

---

입력의 첫째 줄에는 현재 승환이의 앞에 서 있는 학생들의 수  $N$ ( $1 \leq N \leq 1,000$ , 자연수)이 주어진다.

다음 줄에는 승환이 앞에 서 있는 모든 학생들의 번호표( $1, 2, \dots, N$ ) 순서가 앞에서부터 뒤 순서로 주어진다.

이와 같이 입력이 여러개이고 어떤 규칙에 의해 입력이 정해지는 문제를 풀 수 없다.

그래서 입력이 복잡한 코드를 비교할 수 있는 기능을 추가할 예정이다.

## 세부 기능 2)

### 전에 사용했던 코드 이어서 비교할건지 선택하는 기능 추가

현재까지 구현된 프로그램은 매번 두개의 코드를 입력 해줘야한다. 만약 같은 코드로 입력만 바꾸어서 실행하고 싶다고 해도 말이다.

그래서 사용자가 전에 실행했던 코드를 사용할 건지 선택하는 기능을 추가할 예정이다.

ex) 이 전 코드를 이어서 비교하시겠습니까? (Y/N) :

### 3. 프로젝트 일정

업무		11/3	11/10	11/17	12/1	12/15	12/22
제안서 작성		완료					
기능1	세부 기능1		완료				
	세부 기능2			완료			
기능1 보완	세부 기능1			----->			
	세부 기능2			----->			
테스트 및 오류수정				----->			
보고서 작성						----->	

# -----진척보고서#2-----

## 1. 진척사항

### 1) 기능구현

#### (1) 세부 기능1) 입력이 복잡한 코드 비교 기능 추가

```
// 선택한 코드 유형에 따라 파일 이름 설정
string filename;
if (choice == 1) {
    filename = "correct_program.cpp";
} else if (choice == 2) {
    filename = "wrong_program.cpp";
} else {
    filename = "input_generator.cpp";
}

// 파일에 코드 작성 및 컴파일
writeCodeToFile(code, filename);
string compileCommand = "g++ " + filename + " -o " + filename.substr(0, filename.find('.')) + ".exe";
system(compileCommand.c_str());
cout << "코드 설정 및 컴파일 완료!" << endl;
```

문제에서 요구하는 입력을 만드는 코드를 사용자가 따로 작성하여

(correct program, wrong program처럼)

```
string generateInput() {
    string command = "input_generator.exe";
    FILE* pipe = _popen(command.c_str(), "r");
    if (!pipe) {
        return "Error generating input.";
    }

    string result;
    char buffer[128];
    while (fgets(buffer, sizeof(buffer), pipe) != nullptr) {
        result += buffer;
    }

    _pclose(pipe);
    return result;
}
```

generateInput함수로

```

void findCounterexample() {
    int testCount;
    cout << "테스트 반복 횟수를 입력하세요: ";
    cin >> testCount;
    cin.ignore(); // 남아 있는 개행 문자 제거

    // 테스트 반복
    for (int i = 0; i < testCount; ++i) {
        // 입력 생성
        string input = generateInput();
        cout << "생성된 입력:\n" << input << endl;

        // 두 프로그램 실행
        string correctOutput = runProgram("correct_program.exe", input);
        string wrongOutput = runProgram("wrong_program.exe", input);

        // 출력값 비교
        if (correctOutput != wrongOutput) {
            cout << "반례 발견!" << endl;
            cout << "입력값:\n" << input << endl;
            cout << "올바른 코드 출력: " << correctOutput << endl;
            cout << "잘못된 코드 출력: " << wrongOutput << endl;
            return;
        }
    }

    cout << "반례 없음" << endl;
}

```

입력값을 생성해서 이 값을 각 코드에 입력으로 넣고 출력을 비교한다.

테스트 반복횟수 만큼 이 작업을 반복한다.

입력생성기 코드는 난수생성기를 활용하여 이와 같이 작성한다.

(사용자가 문제의 입력에 맞춰서 작성해줘야한다.)

ex)

```

int main() {
    srand(time(0));

    int N = rand() % 10 + 1;
    cout << N << endl;

    vector<int> students;
    for (int i = 1; i <= N; ++i) {
        students.push_back(i);
    }

    random_device rd;
    mt19937 gen(rd());
    shuffle(students.begin(), students.end(), gen);

    for (int i = 0; i < N; ++i) {
        cout << students[i] << " ";
    }
    cout << endl;

    return 0;
}

```

## (2) 세부 기능2) 전에 사용했던 코드 이어서 비교할건지 선택하는 기능 추가

사용자가 메뉴 중 원하는 작업을 실행할 수 있도록 메뉴를 추가했다.

```
int main() {
    while (true) {
        cout << "메뉴:" << endl;
        cout << "1. 프로그램 설정" << endl;
        cout << "2. 반례 찾기" << endl;
        cout << "3. 종료" << endl;
        cout << "선택: ";
        int choice;
        cin >> choice;

        if (choice == 1) {
            setupPrograms();
            break;
        } else if (choice == 2) {
            findCounterexample();
        } else if (choice == 3) {
            cout << "프로그램을 종료합니다." << endl;
            break;
        } else {
            cout << "잘못된 선택입니다. 다시 시도하세요." << endl;
        }
    }

    return 0;
}
```

1번을 선택하면

```

// 프로그램 파일(올바른 프로그램, 잘못된 프로그램, 입력 생성기)을 설정 및 컴파일합니다.
void setupPrograms() {
    string code;
    int choice;

    // 사용자에게 입력할 코드 종류 선택
    cout << "어떤 코드를 입력하시겠습니까?" << endl;
    cout << "1. 올바른 코드 입력" << endl;
    cout << "2. 잘못된 코드 입력" << endl;
    cout << "3. 입력 생성기 코드 입력" << endl;
    cout << "선택: ";
    cin >> choice;
    cin.ignore(); // 남아 있는 개행 문자 제거

    // 선택한 코드 유형에 따라 안내 메시지 표시
    if (choice == 1) {
        cout << "올바른 코드의 내용을 입력하세요. (종료하려면 'done' 입력): " << endl;
    } else if (choice == 2) {
        cout << "잘못된 코드의 내용을 입력하세요. (종료하려면 'done' 입력): " << endl;
    } else if (choice == 3) {
        cout << "입력 생성기 코드의 내용을 입력하세요. (종료하려면 'done' 입력): " << endl;
    } else {
        cout << "잘못된 선택입니다." << endl;
        return;
    }

    // 코드 내용 입력
    string line;
    while (getline(cin, line)) {
        if (line == "done") break;
        code += line + "\n";
    }

    // 선택한 코드 유형에 따라 파일 이름 설정
    string filename;
    if (choice == 1) {
        filename = "correct_program.cpp";
    } else if (choice == 2) {
        filename = "wrong_program.cpp";
    } else {
        filename = "input_generator.cpp";
    }

    // 파일에 코드 작성 및 컴파일
    writeCodeToFile(code, filename);
    string compileCommand = "g++ " + filename + " -o " + filename.substr(0, filename.find('.')) + ".exe";
    system(compileCommand.c_str());
    cout << "코드 설정 및 컴파일 완료!" << endl;
}

```

setupPrograms()함수가 실행되어 1.올바른 코드를 입력할지, 2.잘못된 코드를 입력 할지, 3.입력 생성기 코드를 입력할지 선택할 수 있다.

2번을 선택하면

```
// 반례 찾기 함수
// 입력 생성기와 두 프로그램을 실행하여 반례를 찾습니다.
void findCounterexample() {
    int testCount;
    cout << "테스트 반복 횟수를 입력하세요: ";
    cin >> testCount;
    cin.ignore(); // 남아 있는 개행 문자 제거

    // 테스트 반복
    for (int i = 0; i < testCount; ++i) {
        // 입력 생성
        string input = generateInput();
        cout << "생성된 입력:\n" << input << endl;

        // 두 프로그램 실행
        string correctOutput = runProgram("correct_program.exe", input);
        string wrongOutput = runProgram("wrong_program.exe", input);

        // 출력값 비교
        if (correctOutput != wrongOutput) {
            cout << "반례 발견!" << endl;
            cout << "입력값:\n" << input << endl;
            cout << "올바른 코드 출력: " << correctOutput << endl;
            cout << "잘못된 코드 출력: " << wrongOutput << endl;
            return;
        }
    }

    cout << "반례 없음" << endl;
}
```

findCounterexample함수를 실행해 반례를 찾는다.

3번을 입력하면 프로그램을 종료한다.

## 2)테스트 결과

Test 1 -백준 12789번

```
메뉴:  
1. 프로그램 설정  
2. 반례 찾기  
3. 종료  
선택: 1  
어떤 코드를 입력하시겠습니까?  
1. 올바른 코드 입력  
2. 잘못된 코드 입력  
3. 입력 생성기 코드 입력  
선택: 1  
올바른 코드의 내용을 입력하세요. (종료하려면 'done' 입력)  
#include <iostream>  
using namespace std;
```

-----올바른 코드-----

```
    return 0;  
}  
done  
코드 설정 및 컴파일 완료!
```

```
메뉴:  
1. 프로그램 설정  
2. 반례 찾기  
3. 종료  
선택: 1  
어떤 코드를 입력하시겠습니까?  
1. 올바른 코드 입력  
2. 잘못된 코드 입력  
3. 입력 생성기 코드 입력  
선택: 2  
잘못된 코드의 내용을 입력하세요. (종료하려면 'done' 입력):  
#include <iostream>  
using namespace std;
```

-----잘못된 코드-----

```
    return 0;
}
done
코드 설정 및 컴파일 완료!
```

메뉴 :

- 1. 프로그램 설정
- 2. 반례 찾기
- 3. 종료

선택 : 1

어떤 코드를 입력하시겠습니까?

- 1. 올바른 코드 입력
- 2. 잘못된 코드 입력
- 3. 입력 생성기 코드 입력

선택 : 3

입력 생성기 코드의 내용을 입력하세요. (종료하려면 'done' 입력):

```
#include <iostream>
```

-----입력 생성기 코드-----

```
done
코드 설정 및 컴파일 완료!
```

메뉴 :

1. 프로그램 설정
2. 반례 찾기
3. 종료

선택 : 2

테스트 반복 횟수를 입력하세요 : 10

생성된 입력 :

9

1 5 7 4 8 3 9 6 2

생성된 입력 :

9

8 6 4 7 9 1 2 3 5

생성된 입력 :

9

9 4 3 5 7 1 6 8 2

생성된 입력 :

9

5 3 9 2 7 1 8 4 6

생성된 입력 :

9

8 7 6 4 1 2 9 3 5

생성된 입력 :

2

1 2

생성된 입력 :

2

2 1

반례 발견 !

입력 값 :

2

2 1

올바른 코드 출력 : Nice

잘못된 코드 출력 : Sad

반례를 찾아냈다.

## Test 2 -백준1052번

메뉴 :

1. 프로그램 설정

2. 반례 찾기

3. 종료

선택 : 1

어떤 코드를 입력하시겠습니까?

1. 올바른 코드 입력

2. 잘못된 코드 입력

3. 입력 생성기 코드 입력

선택 : 1

올바른 코드의 내용을 입력하세요. (종료하려면 'done' 입력):

```
#include <iostream>
```

-----올바른 코드-----

done

코드 설정 및 컴파일 완료!

메뉴 :

1. 프로그램 설정

2. 반례 찾기

3. 종료

선택 : 1

어떤 코드를 입력하시겠습니까?

1. 올바른 코드 입력

2. 잘못된 코드 입력

3. 입력 생성기 코드 입력

선택 : 2

잘못된 코드의 내용을 입력하세요. (종료하려면 'done' 입력):

```
#include <iostream>
```

-----잘못된 코드-----

done

코드 설정 및 컴파일 완료!

메뉴 :

1. 프로그램 설정
2. 반례 찾기
3. 종료

선택 : 1

어떤 코드를 입력하시겠습니까?

1. 올바른 코드 입력
2. 잘못된 코드 입력
3. 입력 생성기 코드 입력

선택 : 3

입력 생성기 코드의 내용을 입력하세요. (종료하려면 'done' 입력):

-----입력 생성기 코드-----

done

코드 설정 및 컴파일 완료!

메뉴 :

1. 프로그램 설정
2. 반례 찾기
3. 종료

선택 : 2

테스트 반복 횟수를 입력하세요: 1000

생성된 입력 :

21 5

생성된 입력 :

5 5

생성된 입력 :

14 1

반례 발견!

입력값 :

23 3

올바른 코드 출력 : 1

잘못된 코드 출력 : 5

반례를 찾아냈다!

## 2. 추후 계획

아직 백준 문제중 2문제 밖에 테스트를 해보지 않아서 각각 다양한 입력을 가진 문제들을 찾아서 테스트 해보고 오류가 발생하면 수정하고 최대한 다양한 입력을 받아들일 수 있게 보완해 나갈 예정이다. (백준 질문게시판 검색에 “반례”라고 검색해서 문제를 찾는다.)

현재 난수 발생기를 이용해서 입력을 만들고 있는데 중복이 발생하는 경우가 생긴다. 하지만 중복제거 코드를 넣으면 코드 비교 속도가 너무 느려져서 어떻게 할지 고민중이다.

따라서 입력 생성 방법을 보완할 것이다.

## 3. 프로젝트 일정

업무		11/3	11/10	11/17	12/1	12/15	12/22
제안서 작성		완료					
기능1	세부 기능1	완료					
	세부 기능2		완료				
기능1 보완	세부 기능1			완료			
	세부 기능2			완료			
테스트 및 오류수정				----->			
입력 생성기 보완				----->			
보고서 작성						----->	

# -----진척보고서#3-----

## 1. 계획 대비 변경 사항

### 1) 입력 생성 방식 유지

기존에 계획으로는 입력 생성기를 보완 하고자 했는데 내가 생각하기에 현재 난수를 이용해 입력값을 만드는 프로그램을 여러번 실행하는 방식이 가장 범용적으로 쓸 수 있는 방법인 것 같아 그대로 하기로 했다.

### 2) 저장, 불러오기, 목록 보기 기능 추가 + 컴파일 메뉴 추가

대신 문제 파일 저장, 불러오기, 목록 보기 기능을 추가해서 이전에 프로그램을 사용했던 코드들을 다시 테스트 해보고 싶을때 편리하게 할 수 있도록 하였다.

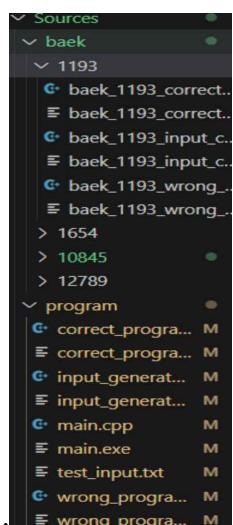
기존에 코드 설정할때 (setupPrograms() 함수) 컴파일이 되게 했었는데 컴파일 과정을 사용자의 지시에 의해 하는게 이번 추가한 기능과 더 잘 맞을 것 같아서 컴파일 메뉴를 추가했다.

## 2. 진척 사항

### 1) 기능 구현

#### (1) 문제 파일 저장 기능

Sources 폴더 안에 program 폴더와 baek 폴더로 나누어 baek폴더에 문제 번호로 폴더를 하나 만들어서 현재 내가 입력한 올바른 코드, 잘못된 코드, 입력생성기, 코드파일이 복사되어 형식에 맞춘 이름으로 저장될 수 있도록하였다



```

// 코드 저장 함수
void saveExistingPrograms() {
    int problemNumber;
    cout << "저장할 문제 번호를 입력하세요: ";
    cin >> problemNumber; // 문제 번호 입력
    cin.ignore(); // 입력 버퍼 비우기

    string basePath = "..\\baek\\" + to_string(problemNumber); // 저장 경로 생성

    // 디렉토리 생성 (존재하지 않을 경우)
    if (!fs::exists(basePath)) {
        if (fs::create_directory(basePath)) { // 디렉토리 생성 시도
            cout << "새 디렉토리를 생성했습니다: " << basePath << endl;
        } else {
            cerr << "디렉토리 생성에 실패했습니다." << endl;
            return;
        }
    }
    else {
        cout << "기존 디렉토리에 저장합니다 : " << basePath << endl;
    }

    // 각 코드 파일 읽기 및 저장
    vector<string> filenames = { // 저장될 파일 이름 목록
        "baek_" + to_string(problemNumber) + "_correct_code.cpp",
        "baek_" + to_string(problemNumber) + "_wrong_code.cpp",
        "baek_" + to_string(problemNumber) + "_input_code.cpp"
    };
    vector<string> sourceFilenames = { // 원본 파일 이름 목록
        "correct_program.cpp",
        "wrong_program.cpp",
        "input_generator.cpp"
    };

    for (size_t i = 0; i < filenames.size(); ++i) {
        string code = readFile(sourceFilenames[i]); // 파일 내용 읽기
        if (code.empty()) {
            cerr << sourceFilenames[i] << " 파일 읽기 실패. 저장을 건너뜁니다." << endl;
            continue; // 파일 읽기 실패 시 다음 파일로 진행
        }

        string fullPath = basePath + "\\\" + filenames[i]; // 전체 경로 생성
        writeCodeToFile(code, fullPath); // 파일에 코드 쓰기
        if (fs::exists(fullPath)) // 파일 저장 성공여부 확인 후 메세지 출력
            cout << filenames[i] << " 저장 완료!" << endl;
        else
            cout << filenames[i] << " 저장 실패!" << endl;
    }
    cout << "모든 코드 저장이 완료되었습니다." << endl;
}

```

## (2) 문제 파일 불러오기 기능

```
while (true) {
    cout << "메뉴:" << endl;
    cout << "1. 프로그램 설정" << endl;
    cout << "2. 기존 파일 불러오기" << endl;
    cout << "3. 반례 찾기" << endl;
    cout << "4. 컴파일" << endl;
    cout << "5. 저장된 문제 목록 보기" << endl;
    cout << "6. 저장하기" << endl;
    cout << "7. 종료" << endl;
```

```
// 파일 복사 함수
void copy(string &sourceFile, string &destinationFile) {
    ifstream source(sourceFile, ios::in); // 원본 파일 읽기 모드로 열기
    if (!source) {
        cerr << "원본 파일을 열 수 없습니다." << endl;
        exit(0);
    }
    ofstream destination(destinationFile, ios::out); // 복사 파일 쓰기 모드로 열기
    if (!destination) {
        cerr << "복사 파일을 열 수 없습니다." << endl;
        exit(0);
    }
    string line;
    while (getline(source, line)) { // 원본 파일에서 한 줄씩 읽어 복사 파일에 쓰기
        destination << line << endl;
    }

    source.close(); // 원본 파일 닫기
    destination.close(); // 복사 파일 닫기
}
```

```
// 기존 파일 불러오기 함수
void loadExistingFiles() {
    int Number;
    cout << "문제 번호를 입력해 주세요: ";
    cin >> Number; // 문제 번호 입력
    string basePath = "..\\baek\\" + to_string(Number); // 문제 파일 경로 생성

    // 문제 번호 디렉토리 확인
    if (!fs::exists(basePath) || !fs::is_directory(basePath)) {
        cout << "저장된 문제가 아닙니다." << endl;
        return;
    }

    // 각 코드 파일 복사
    string sourceFile = "..\\baek\\" + to_string(Number) + "\\baek_" + to_string(Number) + "_wrong_code.cpp";
    string destinationFile = "wrong_program.cpp";
    copy(sourceFile, destinationFile);

    sourceFile = "..\\baek\\" + to_string(Number) + "\\baek_" + to_string(Number) + "_correct_code.cpp";
    destinationFile = "correct_program.cpp";
    copy(sourceFile, destinationFile);

    sourceFile = "..\\baek\\" + to_string(Number) + "\\baek_" + to_string(Number) + "_input_code.cpp";
    destinationFile = "input_generator.cpp";
    copy(sourceFile, destinationFile);

    cout << "불러오기가 완료되었습니다." << endl;
}
```

메뉴에서 2번을 선택하면 문제 번호를 입력하고 해당 번호로 저장된 폴더의 파일들을 불러올 수 있다. 1번기능과 반대로 이번엔 프로그램의 파일들이 변경된다.

### (3) 저장된 문제 목록 보기

5번을 선택하면 baek풀더에 저장된 문제들을 나열할 수 있다. 가시성을 높이기 위해 오름차순으로 정렬하도록 하였다.

```
// 문제 파일 목록을 오름차순으로 정렬하여 나열하는 함수
void listProblemFiles() {
    string path = "..\\baek\\\"; // 문제 파일 경로
    vector<int> problemNumbers; // 문제 번호 저장 벡터

    cout << "문제 파일 목록:" << endl;

    // 디렉토리 탐색
    for (const auto& entry : fs::directory_iterator(path)) {
        if (fs::is_directory(entry)) { // 디렉토리인 경우
            string dirName = entry.path().filename().string(); // 디렉토리 이름 가져오기
            try {
                int problemNumber = stoi(dirName); // 디렉토리 이름을 정수로 변환
                problemNumbers.push_back(problemNumber); // 문제 번호 벡터에 추가
            } catch (const invalid_argument& e) {

                // 디렉토리 이름이 숫자가 아닐 경우 무시
                cout << "숫자로 입력해주세요" << endl;
                continue;
            }
        }
    }

    // 문제 번호 오름차순으로 정렬
    sort(problemNumbers.begin(), problemNumbers.end());

    // 정렬된 문제 번호 출력
    for (const auto& problem : problemNumbers) {
        cout << "문제 번호: " << problem << endl;
    }
}
```

#### (4) 컴파일 메뉴

4번입력시 코드 3개 컴파일

```
} else if (choice == 4) {  
    system("g++ correct_program.cpp -o correct_program.exe"); // 올바른 프로그램 컴파일  
    cout << "1/3 완료" << endl;  
    system("g++ wrong_program.cpp -o wrong_program.exe"); // 잘못된 프로그램 컴파일  
    cout << "2/3 완료" << endl;  
    system("g++ input_generator.cpp -o input_generator.exe"); // 입력 생성기 컴파일  
    cout << "3/3 완료" << endl;  
    cout << "컴파일 완료!" << endl;  
    cout << "프로그램을 종료합니다. 다시 시작해주세요.";  
    break; //컴파일 후 프로그램 종료
```

## 2) 테스트 결과

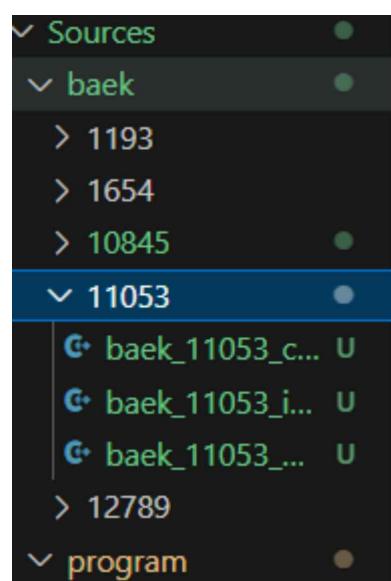
### 1)컴파일 메뉴

```
메뉴 :  
1. 프로그램 설정  
2. 기존 파일 불러오기  
3. 반례 찾기  
4. 컴파일  
5. 저장된 문제 목록 보기  
6. 저장하기  
7. 종료  
선택 : 4  
1/3 완료  
2/3 완료  
3/3 완료  
컴파일 완료!  
프로그램을 종료합니다. 다시 시작해주세요.
```

## 2)저장 기능

----3가지 코드 입력 후----

```
메뉴 :
1. 프로그램 설정
2. 기존 파일 불러오기
3. 반례 찾기
4. 컴파일
5. 저장된 문제 목록 보기
6. 저장하기
7. 종료
선택 : 6
저장할 문제 번호를 입력하세요 : 11053
새 디렉토리를 생성했습니다 : ..\baek\11053
baek_11053_correct_code.cpp 저장 완료!
baek_11053_wrong_code.cpp 저장 완료!
baek_11053_input_code.cpp 저장 완료!
모든 코드 저장이 완료되었습니다.
```



11053 폴더 생성

### 3) 목록 보기 기능

```
메뉴:  
1. 프로그램 설정  
2. 기존 파일 불러오기  
3. 반례 찾기  
4. 컴파일  
5. 저장된 문제 목록 보기  
6. 저장하기  
7. 종료  
선택: 5  
문제 파일 목록:  
문제 번호: 1193  
문제 번호: 1654  
문제 번호: 10845  
문제 번호: 11053  
문제 번호: 12789
```

목록 오름차순으로 출력

### 4) 불러오기 기능

```
메뉴:  
1. 프로그램 설정  
2. 기존 파일 불러오기  
3. 반례 찾기  
4. 컴파일  
5. 저장된 문제 목록 보기  
6. 저장하기  
7. 종료  
선택: 2  
문제 번호를 입력해 주세요: 1193  
불러오기가 완료되었습니다.
```

Sources > program >  correct\_program.cpp >  main(void)

```
1 #include <iostream>
2 using namespace std;
3
4
5 int main(void){
6     int N;
7     cin>>N;
8     int count=1;
9     int num[1000]={0};
10    int temp[1000]={0};
11    int i=0;
12    int t=1;
```

Sources > program >  wrong\_program.cpp > ...

```
1 #include <iostream>
2 using namespace std;
3
4
5
6 int main(void){
7     int N;
8     cin>>N;
9     int count=1;
10    int num[N];
11    int temp[N]={0};
12    int i=0;
13    int t=1;
```

```
Sources > program > C++ input_generator.cpp > main()
1 #include <iostream>
2 #include <cstdlib>
3 #include <ctime>
4 #include <vector>
5 #include <algorithm>
6 #include <random>
7 #include <chrono>
8
9 using namespace std;
10
11 int main() {
12     // 현재 시간의 밀리초를 기반으로 시드 생성
13     unsigned seed = chrono::steady_clock::now().time_since_epoch().count();
14     srand(seed);
15 }
```

파일들을 확인해보면 1193폴더의 파일 내용을 정상적으로 불러온걸 확인할 수 있다.

### 3. 프로젝트 일정

업무		11/ 3	11/10	11/17	12/1	12/15	12/22
제안서 작성		완료					
기능1(출력 비교)	세부기능 1(컴파일,출 력반환)	완료					
	세부기능 2(비교)		완료				
기능1 보완	세부기능1			완료			
	세부기능2			완료			
테스트 및 오류수정				완료			
입력 생성기 보완				취소			
문제 저장,불 러오기,목록 기능				완료			
사용 편의성 증진							----->
보고서 작성						진행중	

### 계획추가)

사용법을 모르는 사용자가 지금 프로그램을 쓴다면 상당히 곤혹스러울 것 같다는 생각이 들어서 사용자가 쉽게 이해하고 사용할 수 있도록 중간 중간 설명을 추가하거나 뒤로가기를 넣는등 편의성을 높일 것이다.

## -----최종 보고서-----

### 1. 진척 사항

#### 사용 편의성 증진

1) 뒤로가기 추가

```
// 코드 설정 함수 (올바른 코드, 잘못된 코드, 입력 생성기 코드 선택)
void setupPrograms() {
    string code;
    int choice;

    cout << "어떤 코드를 입력하시겠습니까?" << endl;
    cout << "1. 올바른 코드 입력" << endl;
    cout << "2. 잘못된 코드 입력" << endl;
    cout << "3. 입력 생성기 코드 입력" << endl;
    cout << "0. 뒤로가기" << endl;
    cout << "선택: ";
    cin >> choice; // 사용자 선택 입력
    cin.ignore(); // 입력 버퍼 비우기

    if (choice == 0) return; // 뒤로가기
```

```
// 반례 찾기 함수 (무작위 입력 생성 및 두 프로그램의 출력 비교)
void findCounterexample() {
    int testCount;
    cout << "테스트 반복 횟수를 입력하세요 (0 입력 시 뒤로가기): ";
    cin >> testCount; // 테스트 횟수 입력
    cin.ignore(); // 입력 버퍼 비우기
    if (testCount == 0) return;
```

```
// 기존 파일 불러오기 함수
void loadExistingFiles() {
    int Number;
    cout << "문제 번호를 입력해 주세요 (0 입력 시 뒤로가기): ";
    cin >> Number; // 문제 번호 입력
    string basePath = "..\\baek\\" + to_string(Number); // 문제 파일 경로 생성

    if (Number == 0) return;
```

```
// 기존 파일 불러오기 함수
void loadExistingFiles() {
    int Number;
    cout << "문제 번호를 입력해 주세요 (0 입력 시 뒤로가기): ";
    cin >> Number; // 문제 번호 입력
    string basePath = "..\\baek\\" + to_string(Number); // 문제 파일 경로 생성

    if (Number == 0) return;
```

각 함수에 0입력시 메뉴로 돌아가도록 코드를 추가했다.

## 2) 도움말 출력 기능 추가

```
else if (choice == 8) {
    showHelp(); // 도움말 출력

while (true) {
    cout << "\n\n메뉴:" << endl;
    cout << "1. 프로그램 설정" << endl;
    cout << "2. 기존 파일 불러오기" << endl;
    cout << "3. 반례 찾기" << endl;
    cout << "4. 컴파일" << endl;
    cout << "5. 저장된 문제 목록 보기" << endl;
    cout << "6. 저장하기" << endl;
    cout << "7. 종료" << endl;
    cout << "8. 도움말" << endl;

    cout << "선택: ";

// 도움말 출력 함수
void showHelp() {
    cout << "\n===== 프로그램 도움말 =====" << endl;
    cout << "1. 프로그램 설정" << endl;
    cout << "    - 올바른 코드, 잘못된 코드, 입력 생성기 코드를 입력합니다." << endl;
    cout << "    - 입력이 끝나면 'done'을 입력하여 종료합니다." << endl;
    cout << "    - 모든 코드를 작성하였다면 컴파일을 해주세요." << endl;

    cout << "\n2. 기존 파일 불러오기" << endl;
    cout << "    - 이미 저장된 문제 번호를 입력하면 해당 문제의 코드가 불러와집니다." << endl;
    cout << "    - 불러오기 후후 컴파일을 해주세요." << endl;

    cout << "\n3. 반례 찾기" << endl;
    cout << "    - 입력 생성기를 통해 무작위 입력을 생성하고" << endl;
    cout << "    - 두 프로그램의 출력을 비교하여 반례를 찾습니다." << endl;

    cout << "\n4. 컴파일" << endl;
    cout << "    - 작성한 코드를 g++ 컴파일러를 통해 실행 파일로 컴파일합니다." << endl;

    cout << "\n5. 저장된 문제 목록 보기" << endl;
    cout << "    - 저장된 문제의 번호를 오름차순으로 나열합니다." << endl;

    cout << "\n6. 저장하기" << endl;
    cout << "    - 현재 작성된 올바른 코드, 잘못된 코드, 입력 생성기 코드를" << endl;
    cout << "    - 특정 문제 번호의 폴더에 저장합니다." << endl;

    cout << "\n7. 종료" << endl;
    cout << "    - 프로그램을 종료합니다." << endl;

    cout << "\n8. 도움말 보기" << endl;
    cout << "    - 이 도움말을 출력합니다." << endl;

    cout << "===== " << endl;
}
```

메뉴에서 8번 입력 시 도움말 출력 함수를 불러오도록 추가했다.

## 2. 테스트

### 1. 뒤로가기 기능 테스트

1번선택후

어떤 코드를 입력하시겠습니까?

- 1. 올바른 코드 입력
- 2. 잘못된 코드 입력
- 3. 입력 생성기 코드 입력
- 4. 뒤로가기

선택: 4

메뉴:

- 1. 프로그램 설정
- 2. 기존 파일 불러오기
- 3. 반례 찾기
- 4. 컴파일
- 5. 저장된 문제 목록 보기
- 6. 저장하기
- 7. 종료
- 8. 도움말

선택: 7

2번선택후

메뉴:

1. 프로그램 설정
2. 기존 파일 불러오기
3. 반례 찾기
4. 컴파일
5. 저장된 문제 목록 보기
6. 저장하기
7. 종료
8. 도움말

선택: 2

문제 번호를 입력해 주세요 (0 입력 시 뒤로가기):

0

메뉴:

1. 프로그램 설정
2. 기존 파일 불러오기
3. 반례 찾기
4. 컴파일
5. 저장된 문제 목록 보기
6. 저장하기
7. 종료
8. 도움말

선택: 1

### 3번선택후

메뉴:

1. 프로그램 설정
2. 기존 파일 불러오기
3. 반례 찾기
4. 컴파일
5. 저장된 문제 목록 보기
6. 저장하기
7. 종료
8. 도움말

선택: 3

테스트 반복 횟수를 입력하세요 (0 입력 시 뒤로가기): 0

메뉴:

1. 프로그램 설정
2. 기존 파일 불러오기
3. 반례 찾기
4. 컴파일
5. 저장된 문제 목록 보기
6. 저장하기
7. 종료
8. 도움말

선택: 1

### 6번 선택후

```
메뉴:
1. 프로그램 설정
2. 기존 파일 불러오기
3. 반례 찾기
4. 컴파일
5. 저장된 문제 목록 보기
6. 저장하기
7. 종료
8. 도움말
선택: 6
저장할 문제 번호를 입력하세요(0 입력 시 뒤로가기): 0

메뉴:
1. 프로그램 설정
2. 기존 파일 불러오기
3. 반례 찾기
4. 컴파일
5. 저장된 문제 목록 보기
6. 저장하기
7. 종료
8. 도움말
선택: ■
```

## 2. 도움말 출력테스트

```
메뉴:
1. 프로그램 설정
2. 기존 파일 불러오기
3. 반례 찾기
4. 컴파일
5. 저장된 문제 목록 보기
6. 저장하기
7. 종료
8. 도움말
선택: 8

===== 프로그램 도움말 =====
1. 프로그램 설정
  - 올바른 코드, 잘못된 코드, 입력 생성기 코드를 입력합니다.
  - 입력이 끝나면 'done'을 입력하여 종료합니다.
  - 모든 코드를 작성하였다면 컴파일을 해주세요.

2. 기존 파일 불러오기
  - 이미 저장된 문제 번호를 입력하면 해당 문제의 코드가 불러와집니다.
  - 불러오기 후 컴파일을 해주세요.

3. 반례 찾기
  - 입력 생성기를 통해 무작위 입력을 생성하고
  - 두 프로그램의 출력을 비교하여 반례를 찾습니다.

4. 컴파일
  - 작성한 코드를 g++ 컴파일러를 통해 실행 파일로 컴파일합니다.

5. 저장된 문제 목록 보기
  - 저장된 문제의 번호를 오름차순으로 나열합니다.

6. 저장하기
  - 현재 작성된 올바른 코드, 잘못된 코드, 입력 생성기 코드를
  특정 문제 번호의 폴더에 저장합니다.

7. 종료
  - 프로그램을 종료합니다.

=====
```

### 3. 계획 대비 달성

업무		11/ 3	11/10	11/17	12/1	12/15	12/22
제안서 작성		완료					
기능1(출력 비교)	세부기능 1(컴파일,출 력반환)	완료					
	세부기능 2(비교)		완료				
기능1 보완	세부기능 1(입력이 복 잡한 코드 비교)			완료			
	세부기능 2(이어서 비교)			완료			
테스트 및 오류수정				완료			
입력 생성기 보완				취소			
문제 저장,불 러오기,목록 기능				완료			
사용 편의성 증진						완료	
보고서 작성						완료	

입력 생성기 보완 빼고는 목표한대로 달성했다.

## 4. 느낀점

프로젝트 기간에 마침 일상 속에서 느낀 불편한 점이 떠올라 그걸 주제로 할 수 있어 좋았다. 프로젝트에서 가장 중요하고 신경썼던 것은 어떻게 프로그램에 들어갈 입력을 만들어내냐는 것이었다. 제안서 낼때 까지만 해도 어찌저찌하면 모든 문제에 적용시킬 수 있는 반례생성기를 만들 수 있을 줄 알았는데 생각보다 쉽지 않았다. 입력이 하나라고 가정했을 때는 굉장히 쉬운데 입력이 여러개 일 때 어떻게 해야 하나 고민을 많이 했다. 그렇게 만들어진 프로그램이 어느정도 제 역할을 하긴하는데 완벽하진 않은 것 같다. 입력생성기도 프로그램마다 귀찮게 만들어야하고 난수를 사용해서 입력을 만들다 보니 반례 입력값만 피해가는 불상사가 발생할 수 도 있다. 그냥 반례가 궁금할 때 혹시나? 하고 돌려보는 정도는 할 수 있을 것 같다. 프로그램을 만들면서 아 이거 추가하면 좋겠다 하는게 많이 떠올라서 중간중간 계획에 추가를 해가며 진행했다. 어쩌면 무계획한 것 같기도 하다.

짧은 기간이지만 대학교 입학후 처음 프로젝트를 해보았는데 좋은 경험이 된 것 같다.