

프로그래밍 역량 강화 전문기관, 민코딩

ALU KATA



[참고] Refactoring KATA

✓카타 (일본어)

- 무술이 실전에서 바로 사용될 수 있도록,
- 상황을 가정하여 실무를 위한 시뮬레이션

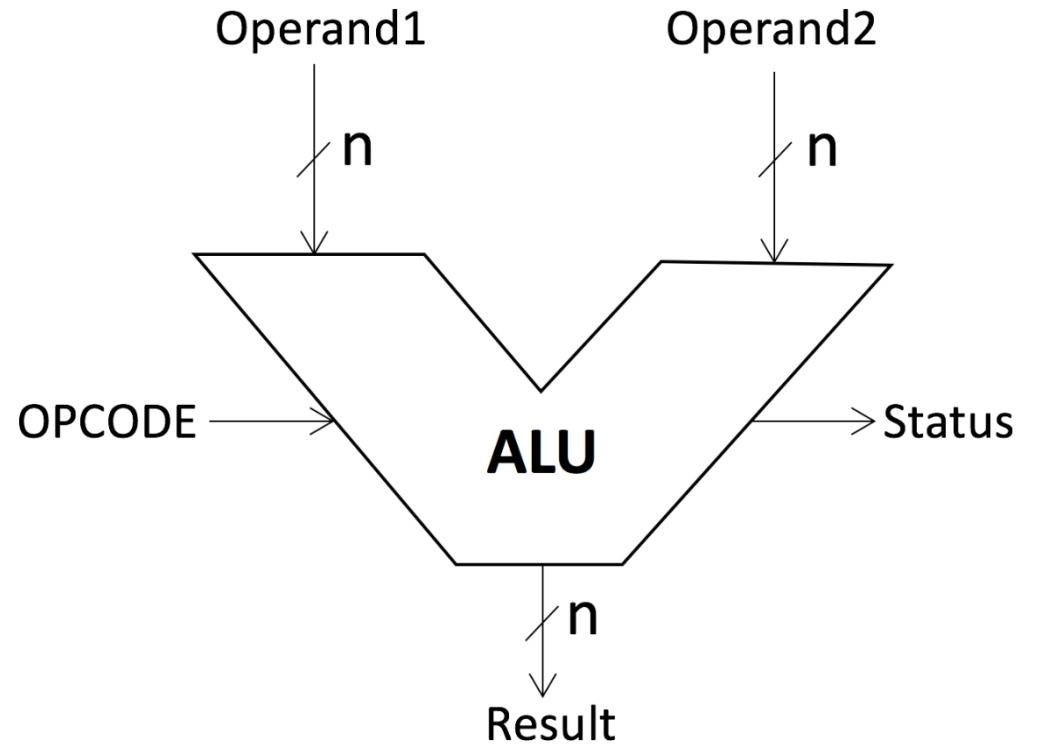
✓리팩토링 카타

- 실무 리팩토링을 할 수 있도록
- 리팩토링 훈련 자료들

ALU 리팩토링

Operand1 과 Operand2에
수를 넣고,
OPCODE를 넣으면

그 결과로
Status / Result 가 나온다.



Lagacy Code

✓ DS 사내 링크

- <https://github.samsungds.net/cra1-sec/ALU>

✓ 사외 링크

- <https://github.com/mincoding1/ALU>
 - #include 의 프로젝트명을 본인에게 맞게 수정 필요.



main ▾ ALU / cpp / ALU_test.cpp

 mincoding1 Create ALU_test.cpp

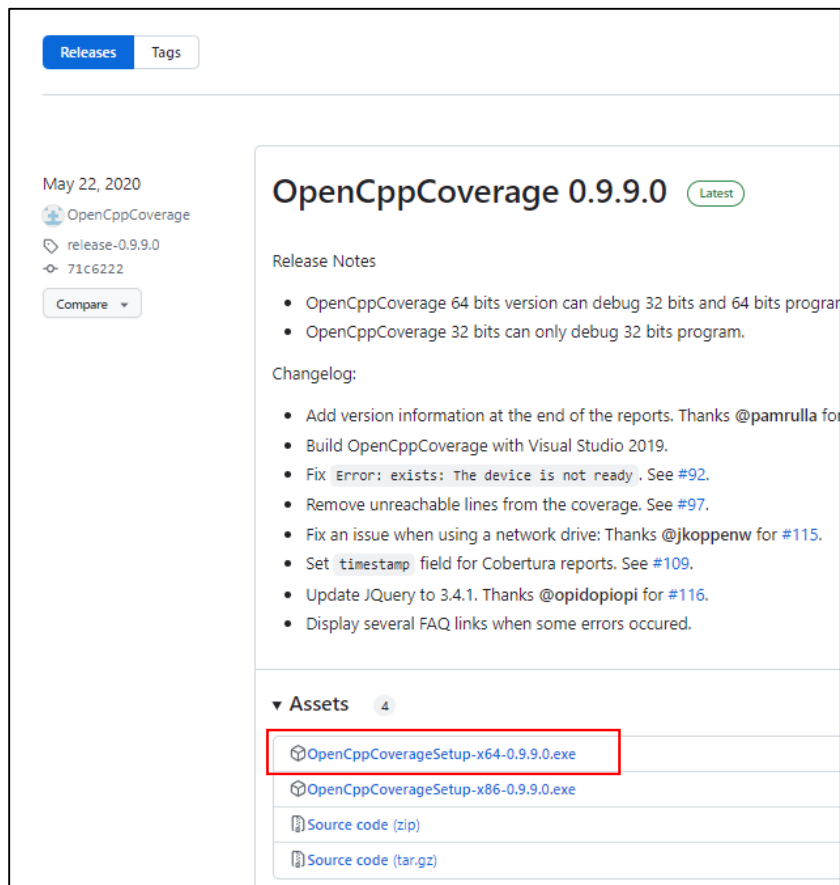
1 contributor

16 lines (13 sloc) | 290 Bytes

```
1  #include "pch.h"
2  #include "../Project6/ALU.cpp";
3  #include "../Project6/Result.cpp";
4
5  TEST(ALU, ADDTest) {
6      ALU alu;
7      alu.setOperand1(10);
8      alu.setOperand2(20);
9      alu.setOPCODE("ADD");
10
11     Result ret;
12     alu.enableSignal(&ret);
13
14     EXPECT_EQ(30, ret.getResult());
15     EXPECT_EQ(0, ret.getStatus());
16 }
```

코드 커버리지 측정을 위한 준비

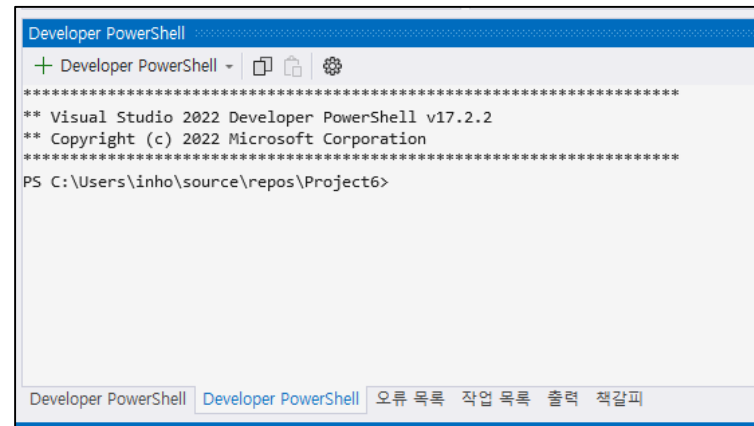
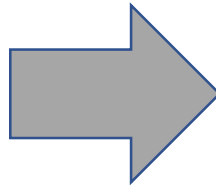
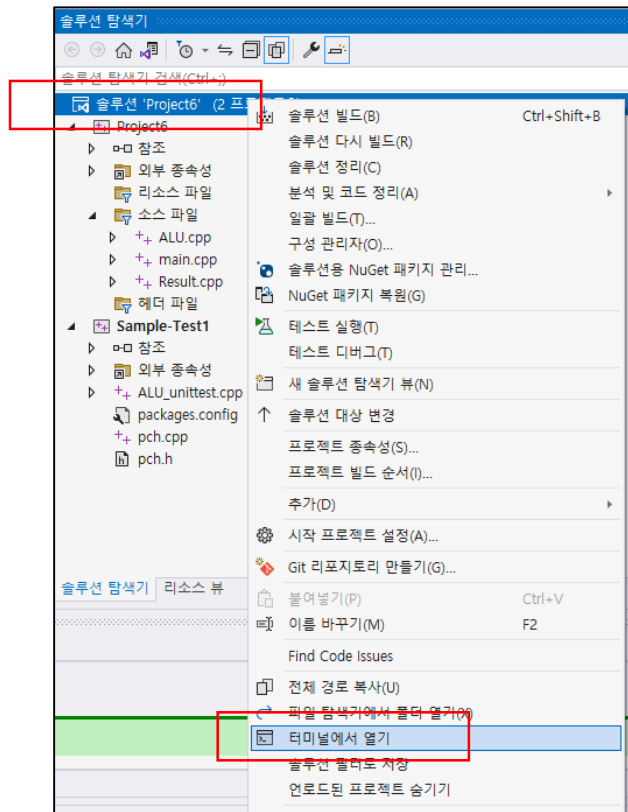
<https://github.com/OpenCppCoverage/OpenCppCoverage/releases>



64bit exe 파일로 다운로드를 받는다.
설치 후, **Visual Studio**를 꺾다가 다시 켜다.

동작 테스트

프로젝트가 아닌 “**솔루션**” 에서 **터미널 열기**



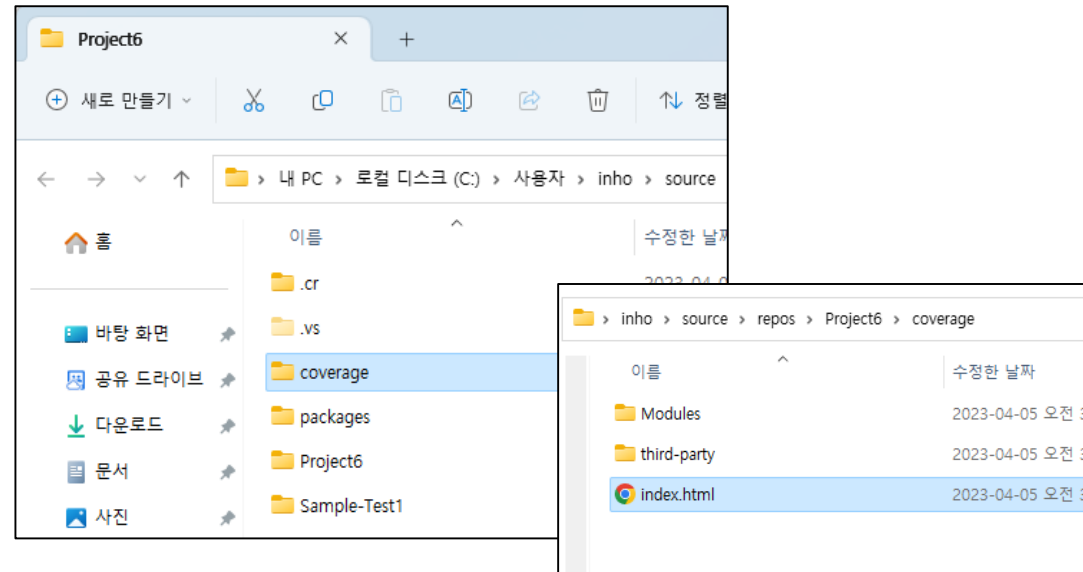
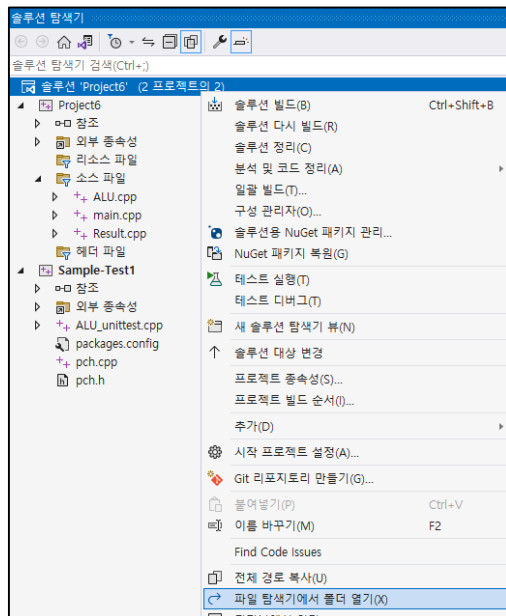
코드 커버리지 확인

OpenCppCoverage.exe --sources C:*.c --export_type=html:coverage -- .\x64\Debug\Sample-Test1.exe

1. 커버리지 측정 시작
2. coverage폴더에서 결과 확인

```
개발자 PowerShell
+ 개발자 PowerShell
[-----] 1 test from GildedRoseTest (3 ms total)

[-----] Global test environment tear-down
[=====] 1 test from 1 test case ran. (5 ms total)
[ PASSED ] 1 test.
[info] Module: C:\Windows\System32\kernel.appcore.dll is selected because it matches selected pattern: *
[info] -----
[info] Coverage generated in Folder C:\Users\minco\source\repos\Project109\coverage
[info] -----
[info] The code coverage report is not what you expect? See the FAQ https://github.com/OpenCppCoverage/OpenCppCoverage/wiki/FAQ.
PS C:\Users\minco\source\repos\Project109> OpenCppCoverage.exe --sources C:*.c --export_type=html:coverage -- .\x64\Debug\Sample-Test1.exe
```



[Trouble Shooting] OpenCppCoverage.exe

```
OpenCppCoverage.exe --sources C:*.c --export_type=html:coverage -- .\x64\Debug\Sample-Test1.exe
```

✓오타율이 매우 높습니다! 다음 항목을 체크해주세요.



1. -- 는 총 3개가 기입이 되었는가?
2. 소스코드가 C:\에 있다면 : --source C:*.c
소스코드가 D:\에 있다면 : --source D:*.c
3. --export_type=html:coverage 가 정확히 입력 되었는가?
4. 띄어쓰기를 정확히 지켰는가? (특히 마지막 -- 부분)
5. 경로에 [GoogleTest 프로젝트명].exe이 정확히 입력 되었는가?
6. 윈도우는 'W' or '\ '를 사용한다. Bash 같은 리눅스CLI는 '/'를 사용한다.

안된다면
"x64\ "를 삭제하여
시도해본다.

Code Coverage 결과

- ✓ 초록색 : 테스트가 닿은 곳
- ✓ 빨간색 : 테스트가 닿지 못한 곳

Sample-Test1.exe

Coverage	Total lines	Items
 Uncover 47% — Cover 53%	77	C:\Users\inho\source\repos\Project6\x64\Debug\Sample-Test1.exe
 Uncover 66% — Cover 34%	55	C:\Users\inho\source\repos\Project6\Project6\ALU.cpp



```
ALU.cpp
1. #pragma once
2.
3. #include <string>
4. #include "Result.cpp"
5.
6. class ALU
7. {
8.     int operand1 = -1;
9.     int operand2 = -1;
10.    std::string OPCODE = "";
11.
12. public:
13.    void setOperand1(int operand1) {
14.        this->operand1 = operand1;
15.    }
16.
17.    void setOperand2(int operand2) {
18.        this->operand2 = operand2;
19.    }
20.
21.    void setOPCODE(std::string OPCODE) {
22.        this->OPCODE = OPCODE;
23.    }
24.
25.    void enableSignal(Result *r) {
26.        if (OPCODE == "ADD" && OPCODE != "MUL" && OPCODE != "SUB") {
27.            if (operand1 != -1 && operand2 != -1) {
28.                int result = operand1 + operand2;
29.                r->setResult(result);
30.                r->setStatus(0);
31.            }
32.            else if (operand1 == -1) {
33.                r->setResult(65535);
34.                r->setStatus(1);
35.            }
36.            else if (operand2 == -1) {
37.                r->setResult(65535);
38.                r->setStatus(2);
39.            }
40.        }
41.        else if (OPCODE != "ADD" && OPCODE == "MUL" && OPCODE != "SUB") {
42.            if (operand1 != -1 && operand2 != -1) {
43.                int result = operand1 * operand2;
44.                r->setResult(result);
45.                r->setStatus(0);
46.            }
47.            else if (operand1 == -1) {
48.                r->setResult(65535);
49.                r->setStatus(1);
50.            }
51.            else if (operand2 == -1) {
52.                r->setResult(65535);
53.            }
54.        }
55.    }
56.
57.    void enableSignal() {
58.        Result *r = new Result();
59.        enableSignal(r);
60.        delete r;
61.    }
62.
63.    void execute() {
64.        enableSignal();
65.    }
66.
67.    void reset() {
68.        operand1 = -1;
69.        operand2 = -1;
70.        OPCODE = "";
71.    }
72.
73.    void print() {
74.        std::cout << "operand1: " << operand1 << " operand2: " << operand2 << " OPCODE: " << OPCODE << "\n";
75.    }
76.
77.    ~ALU() {
78.        reset();
79.    }
80.
81.};
```

리팩토링 하기 전,

테스트 환경을 만들고 진행해야한다.

- 마음껏 리팩토링을 할 수 있다.

[도전] Unit Test 작성하기

Unit Test를 작성한다.

Code Coverage가 100%가 나오는지 확인한다.

만약 안나온다면,
유닛테스트를 추가하여
100%를 만들어낸다.

[도전] 리팩토링 시작!

Rule

- 수정해도 영향을 크게 끼치지 않는 것부터 수정한다.
- 단축키를 사용하면서 리팩토링한다.