CTIP for C
Coverage Test

200511305 김성규
200511306 김성훈
200614164 김효석
200611124 유성배
200518036 곡진화

Coverage Tool
gcov

GCOV

- gcov
  - 코드 커버리지 테스트 프로그램
  - GNU CC(gcc)와 함께 사용
  - http://gcc.gnu.org/onlinedocs/gcc/Gcov.html

  - 옵션 --coverage 로 컴파일 시 gcno 파일(메타 파일) 생성
    - 컴파일 시 최적화 옵션은 사용하지 않아야 함
    - 최적화 과정에서 여러 개의 행들이 하나의 함수로 통합 가능성 존재

  - 옵션 --coverage 로 링커 수행 후 생성된 파일 실행 시
    - gcda 파일 (데이터 파일) 생성
    - 실행된 파일이 수행한 횟수 기록
    - 파일 반복 실행 시 기존의 횟수에 누적됨

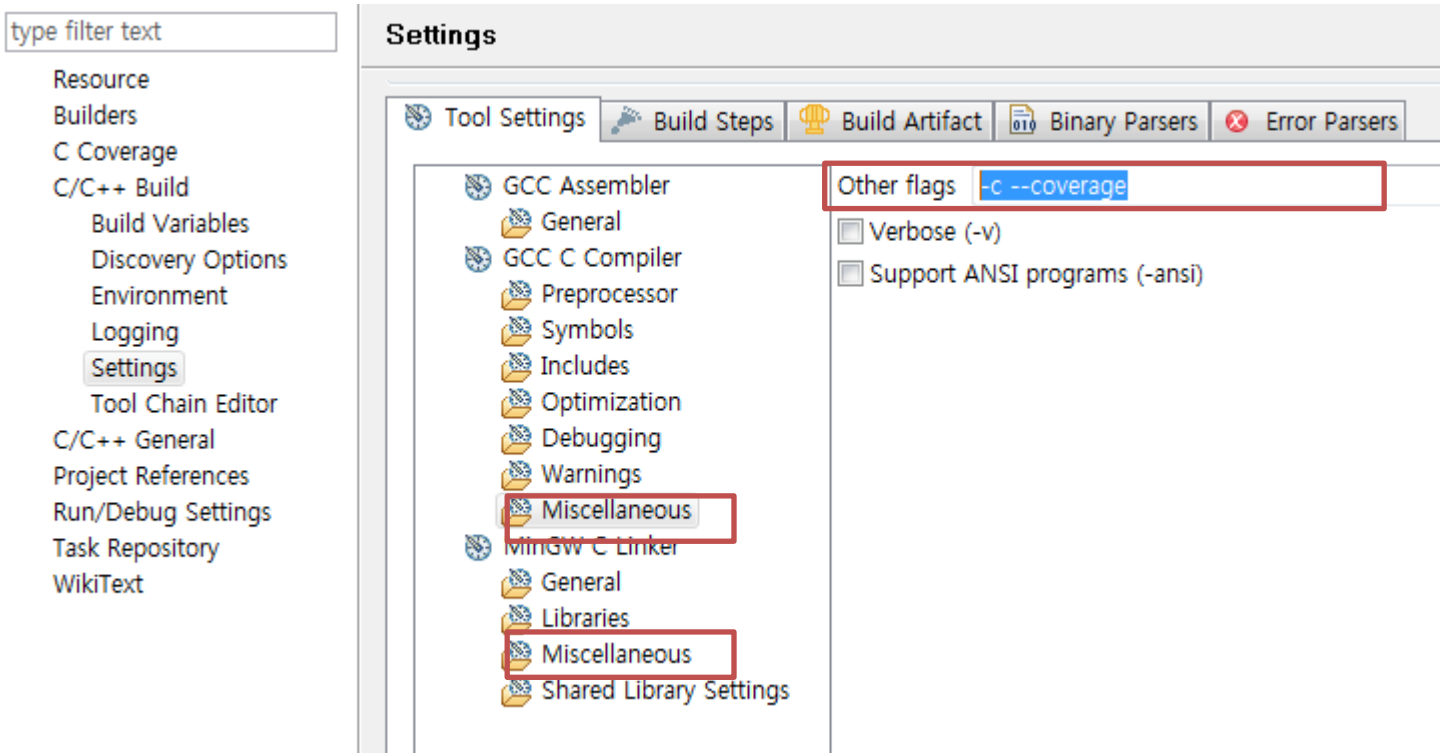
  - gcno파일과 gcda파일을 이용하여 gcov 파일 생성
    - gcov 파일 : 기존 c 파일에 커버리지 정보를 기록

GCOV

- Mingw(Minimalist GNU for Windows)
  - http://sourceforge.net/projects/mingw/
  - gcc와 더불어 gcov도 함께 설치
  - 환경 변수 추가
    - C:₩MinGW₩bin
- 이클립스 플러그인 설치
  - http://sourceforge.jp/projects/ginkgo/releases/
  - org.ginkgo.gcov.feature_0.2.2.zip 다운로드
  - Zip 파일 내 org.ginkgo.gcov_0.2.2.jar를 이클립스 plugin에 복사

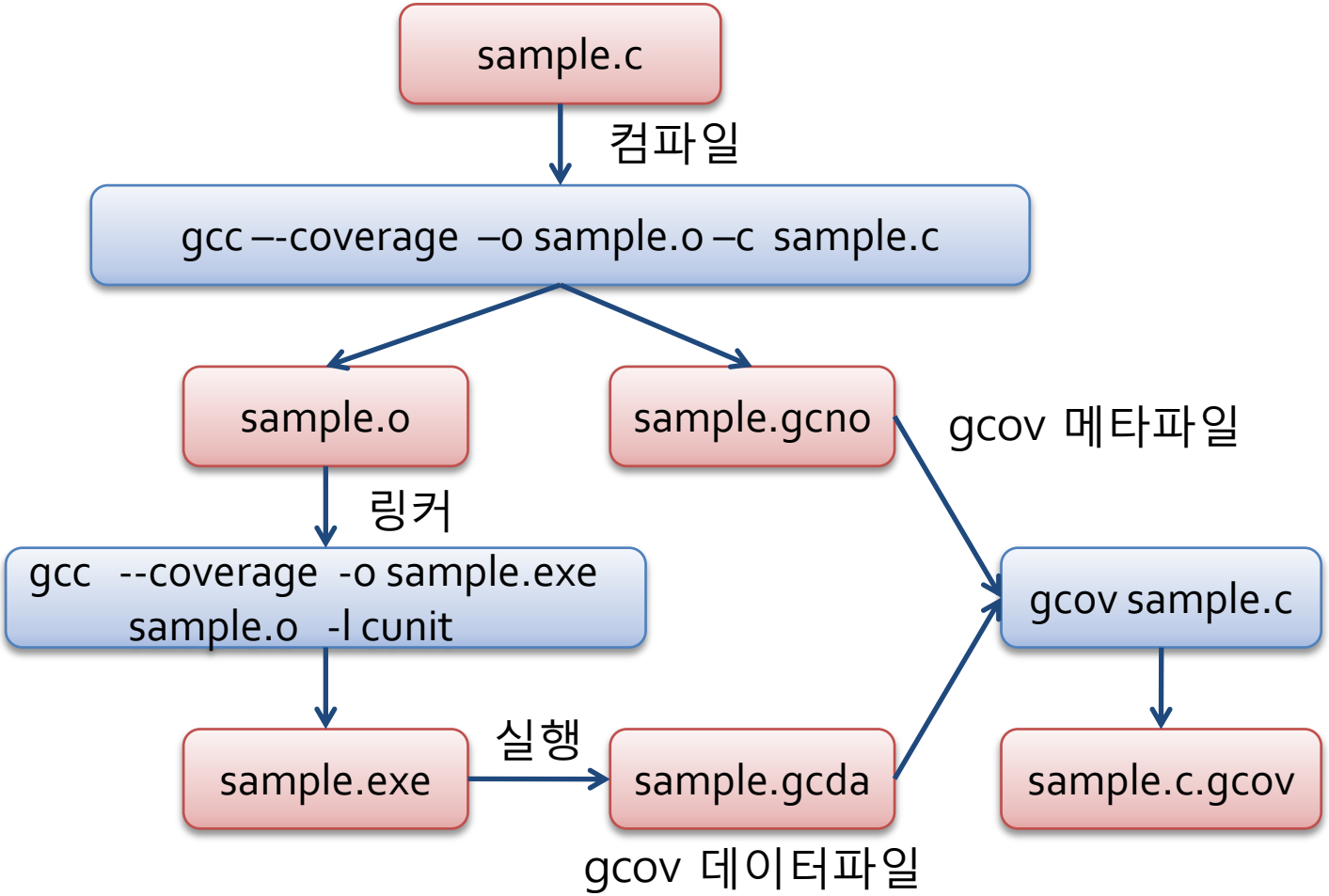
GCOV

- Compiler / Linker Setting
  - Option : --coverage
  - Compiler > Miscellaneous
  - Linker > Miscellaneous



GCOV

❖ 동작 과정


sample.c
컴파일
gcc –-coverage –o sample.o –c sample.c
sample.o
sample.gcno
gcov 메타파일
링커
gcc --coverage -o sample.exe
sample.o -l cunit
gcov sample.c
sample.exe
실행
sample.gcda
sample.c.gcov
gcov 데이터파일

GCOV

❖ gcov option

```
C:₩workspace cdt₩Busfee>gcov
Usage: gcov [OPTION]... SOURCEFILE...

Print code coverage information.

  -h, --help                      Print this help, then exit
  -v, --version                   Print version number, then exit
  -a, --all-blocks                Show information for every basic block
  -b, --branch-probabilities      Include branch probabilities in output
  -c, --branch-counts             Given counts of branches taken
                                    rather than percentages
  -n, --no-output                 Do not create an output file
  -l, --long-file-names           Use long output file names for included
                                    source files
  -f, --function-summaries        Output summaries for each function
  -o, --object-directory DIR|FILE Search for object files in DIR or called FILE
  -p, --preserve-paths            Preserve all pathname components
  -u, --unconditional-branches    Show unconditional branch counts too
```

◆ 주요 옵션

- -b : 출력 파일에 분기가 일어난 횟수 기록과 요약 정보를 출력
- -f : 각각의 함수에 대한 요약 정보 출력
- -a : 출력 파일에 block 정보 기록

GCOV

- gcov –b –f [c file]

```
C:₩workspace cdt₩Busfee>gcov -b -f testbusfee.c
Function 'test_take_menu_015'
Lines executed:89.47% of 19
Branches executed:100.00% of 2
Taken at least once:50.00% of 2
Calls executed:80.00% of 10

Function 'main'
Lines executed:100.00% of 8
No branches
Calls executed:100.00% of 6

File 'testBusfee.c'
Lines executed:92.59% of 27
Branches executed:100.00% of 2
Taken at least once:50.00% of 2
Calls executed:87.50% of 16
testBusfee.c:creating 'testBusfee.c.gcov'
```

- Statement Coverage, Branch Coverage 출력
- [c file].gcov 파일 생성

GCOV

❖ gcov 파일

```
function test_take_menu_015 called 2 returned 100% blocks executed 83%
        2:    7:void test_take_menu_015(void)
        -:    8:{
        2:    9:    int money=600;
        2:   10:    int select = 2;
        2:   11:    e = 1000;
        2:   12:    1;
        2:   13:    int transfer=3;
        -:   14:
        2:   15:    int trans_fee=0;
        2:   16:    int total_fee=0;
        -:   17:
        2:   18:    CU_ASSERT(select_menu(select, &money) == true);
call    0 returned 100%
call    1 returned 100%
        -:   19:
        2:   20:    if(select==1){
branch  0 taken 0% (fallthrough)
branch  1 taken 100%
    #####:   21:        charge_card(&money, charge);
call    0 never executed
    #####:   22:        CU_ASSERT(money == 7500);
call    0 never executed
        -:   23:    }
        -:   24:
        2:   25:    trans_fee=calc_transfer(transfer);
call    0 returned 100%
        2:   26:    CU_ASSERT(trans_fee == 300);
```

함수 분석

라인 실행 횟수

라인 수

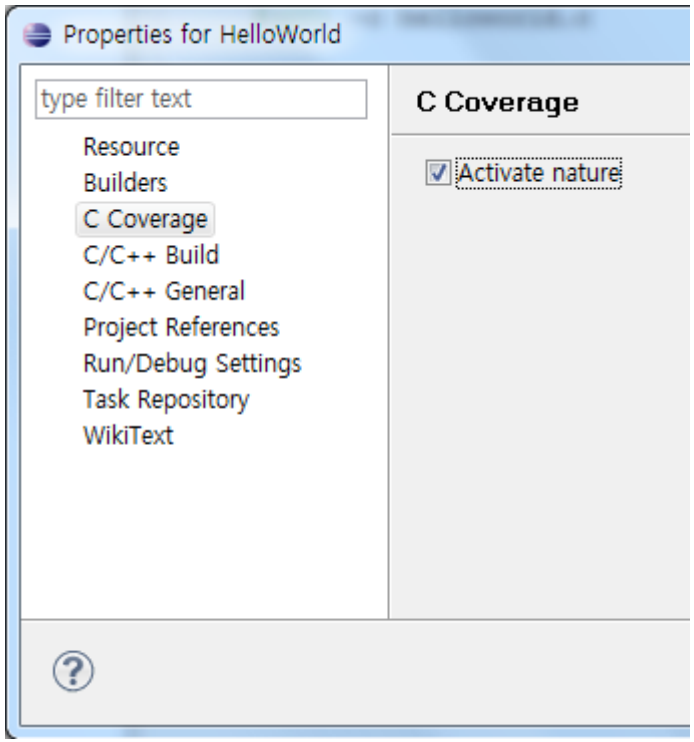
분기 분석

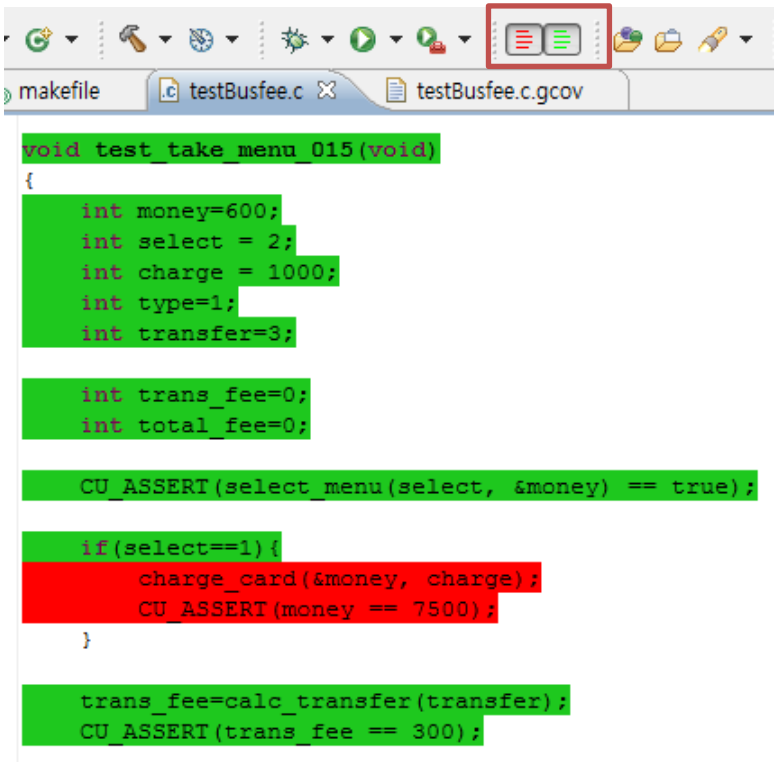
미실행 표시

GCOV

- gcov 플러그인 설정
  - Project>Properties>C Coverage>Check Activate nature
  - 소스 파일에서 각 행의 실행 여부 나타냄

실행 / 미실행 부분을 표시





GCOV

- makefile 작성
  - CTIP 환경에 연동을 위한 makefile

```
all : compile link run move gcov
compile : testBusfee.c
    gcc -I C:\CUnit\include -o testcase.o -c testcase.c
    gcc -I C:\CUnit\include --coverage -o testBusfee.o -c testBusfee.c
link : testBusfee.o
    gcc -L C:\CUnit\lib  --coverage -o testBusfee.exe testBusfee.o testcase.o -l cunit
run : testBusfee.exe
    testBusfee.exe
move : CUnitAutomated-Listing.xml CUnitAutomated-Results.xml
    mv CUnitAutomated-Results.xml ./XML/Result.xml
    mv CUnitAutomated-Listing.xml ./XML/Listing.xml
gcov : testBusfee.gcno testBusfee.gcda
    gcov -f -b testBusfee.c
```

Pairwise Test

Pairwise Test

- 프로젝트 설명
  - T-Money Card를 사용하여
    금액 충전 및 버스 탑승을 이용할 수 있는 프로그램

    - Case 1 : Card 충전 메뉴
      - 입력값
        - 충전요금
      - 결과값
        - 충전 후 소지금

    - Case 2 : 버스 탑승 메뉴
      - 입력값
        - 버스종류
        - 환승 횟수
      - 결과값
        - 환승 횟수
        - 총 요금
        - 남은 소지금

Pairwise Test

- Specification
  - 메뉴 선택 값을 받는다.
  - 메뉴에 따른 입력 값을 받는다.
  - Card 충전 메뉴일 경우,
    - 버스카드 충전의 경우 충전 금액은 1000원 단위이며,
      최대 10000원까지 충전 가능하다.
    - 충전 금액이 10000원이 넘어갈 경우, 충전 전의 소지금이 출력된다.
  - 버스 탑승 메뉴일 경우,
    - 탑승 가능한 버스는 3종류이며, 각 버스별로 요금이 다르다.
      - 마을버스 : 500원, 일반버스 : 900원, 광역버스 : 1700원
    - 버스를 환승 할 수 있는 횟수는 최대 3번이다.
      한번 환승 할 때마다 100원의 추가 금액이 발생한다.
    - 버스 총 요금이 마이너스일 경우, 소지금은 0으로 출력된다.

Pairwise Test

- Representative values
  - 메뉴 선택 여부 : [Select]
    - 1 / 2
  - 소지 금액 : [Money]
    - 0 / 1000 / 5000
  - 충전 금액 : [Charge]
    - Not Use / 1000 / 5000 / 10000 / 11000
  - 환승 횟수 : [Transfer]
    - Not Use / 0 / 1 / 2 / 3
  - 버스 종류 : [Type]
    - Not Use / 1 / 2 / 3

Pairwise Test

❖ Pairwise Test

example - 메모장

파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

```
Money :          0, 1000, 5000
Select:          1, 2
Charge:          Not Use, 1000, 5000,10000,11000
Transfer:        Not Use, 0, 1, 2, 3
Type:            Not Use, 1, 2, 3

IF [Select] = 1
        THEN [Charge] <> "Not Use"
        AND  [Transfer] = "Not Use"
        AND  [Type] = "Not Use";
IF [select] = 2
        THEN [charge] = "Not Use"
        AND  [transfer] <> "Not Use"
        AND  [type] <> "Not Use";
```

PICT 이용

	A	B	C	D	E
1	Select	Money	Charge	Transfer	Type
2	1	0	10000	Not Use	Not Use
3	2	1000	Not Use	0	1
4	1	1000	1000	Not Use	Not Use
5	1	0	1000	Not Use	Not Use
6	2	5000	Not Use	1	2
7	2	0	Not Use	2	3
8	2	5000	Not Use	2	1
9	1	1000	11000	Not Use	Not Use
10	1	5000	5000	Not Use	Not Use
11	2	0	Not Use	1	1
12	2	0	Not Use	0	2
13	1	1000	10000	Not Use	Not Use
14	2	1000	Not Use	1	3
15	1	5000	1000	Not Use	Not Use
16	2	5000	Not Use	0	3
17	1	5000	11000	Not Use	Not Use
18	2	0	Not Use	3	3
19	1	1000	5000	Not Use	Not Use
20	1	0	11000	Not Use	Not Use
21	2	1000	Not Use	3	2
22	2	5000	Not Use	3	1
23	1	5000	10000	Not Use	Not Use
24	1	0	5000	Not Use	Not Use
25	2	1000	Not Use	2	2

Pairwise Test

❖ Generate Testcase

Testcase	Select	Money	Charge	Transfer	Type
1	1	0	10000	Not Use	Not Use
2	2	1000	Not Use	0	1
3	1	1000	1000	Not Use	Not Use
4	1	0	1000	Not Use	Not Use
5	2	5000	Not Use	1	2
6	2	0	Not Use	2	3
7	2	5000	Not Use	2	1
8	1	1000	11000	Not Use	Not Use
9	1	5000	5000	Not Use	Not Use
10	2	0	Not Use	1	1
11	2	0	Not Use	0	2
12	1	1000	10000	Not Use	Not Use

Pairwise Test

❖ Generate Testcase

Testcase	Select	Money	Charge	Transfer	Type
13	2	1000	Not Use	1	3
14	1	5000	1000	Not Use	Not Use
15	2	5000	Not Use	0	3
16	1	5000	11000	Not Use	Not Use
17	2	0	Not Use	3	3
18	1	1000	5000	Not Use	Not Use
19	1	0	11000	Not Use	Not Use
20	2	1000	Not Use	3	2
21	2	5000	Not Use	3	1
22	1	5000	10000	Not Use	Not Use
23	1	0	5000	Not Use	Not Use
24	2	1000	Not Use	2	2

Pairwise Test

❖ 테스트 결과 - XML 파일

Automated Test Run Results

Running Suite T-Money Card test

Running test execute_test_001 ...	Passed
Running test execute_test_002 ...	Passed
Running test execute_test_003 ...	Passed
Running test execute_test_004 ...	Passed
Running test execute_test_005 ...	Passed
Running test execute_test_006 ...	Passed
Running test execute_test_007 ...	Passed
Running test execute_test_008 ...	Failed

File Name	testBusfee.c	**Line Number**	237
Condition	money == 12000		

Running test execute_test_009 ...	Passed
Running test execute_test_010 ...	Passed
Running test execute_test_011 ...	Passed
Running test execute_test_012 ...	Passed
Running test execute_test_013 ...	Passed
Running test execute_test_014 ...	Passed
Running test execute_test_015 ...	Passed
Running test execute_test_016 ...	Failed

File Name	testBusfee.c	**Line Number**	349
Condition	money == 16000		

Pairwise Test

❖ 테스트 결과 - XML 파일

Running test execute_test_017 ...	Passed
Running test execute_test_018 ...	Passed
Running test execute_test_019 ...	Failed
File Name testBusfee.c	**Line Number** 388
Condition money == 11000	
Running test execute_test_020 ...	Passed
Running test execute_test_021 ...	Passed
Running test execute_test_022 ...	Passed
Running test execute_test_023 ...	Passed
Running test execute_test_024 ...	Passed

Cumulative Summary for Run

Type	Total	Run	Succeeded	Failed	Inactive
Suites	1	1	- NA -	0	0
Test Cases	24	24	21	3	0
Assertions	72	72	69	3	n/a

Pass - 21
Failed - 3

Pairwise Test

- 테스트 결과 분석
  - Fail된 테스트 케이스
    - execute_test_008, execute_test_016, execute_test_019
    - 모두 명세서 범위를 초과한 11000원으로 테스트케이스 수행
      [Card 충전 금액이 명세서에서는 최대 10000원]
    - 기대되는 소지금은 **"기존 소지금+충전 금액"**이지만
      코드상 충전 금액을 0으로 인식하며 충전이 되지 않아 발생하는 Fail
    - Fail을 의도한 입력값 외에는 모두 Pass

그렇다면 문제없는 소스코드인가?
테스트케이스에 문제는 없는가?

Coverage Test

Coverage 측정

❖ Pairwise Test에 대한 Coverage 측정

makefile | testBusfee.c | testcase.h | tes

```
}
int calc_transfer(int transfer)
{
    switch(transfer){
    case 0:
    case 1:
    case 2:
    case 3:
        return transfer*100;
    default:
        return 0;
    }
}
int calc_fee(int trans_fee, int type)
{
    switch(type){
    case 1:
        return (500+trans_fee);
    case 2:
        return (900+trans_fee);
    case 3:
        return (1700+trans_fee);
    default:
        return 0;
    }
```

Problems | Tasks | Console | Properties

C-Build [Busfee]

```
File 'testBusfee.c'
Lines executed:78.08% of 73
Branches executed:88.89% of 18
Taken at least once:72.22% of 18
Calls executed:81.08% of 37
testBusfee.c:creating 'testBusfee.c.gcov'
```

Lines Executed : 78.08%
Branches Executed : 88.89%
Taken at least once : 72.22%
Calls Executed : 81.08%

Pairwise Test가 적어도 한번 실행하지 못 한 부분도 존재

Coverage 측정

- 추가 Testcase 생성
  - select_menu 함수

```
Function 'select_menu'
Lines executed:62.50% of 8
Branches executed:66.67% of 6
Taken at least once:50.00% of 6
No calls
```

```
        return true;
    }
    else if(num==0)
    {
        return true;
    }
    else{
        return false;
    }
}
```

select = 0 : 프로그램 종료
select = 1 : 충전
select = 2 : 요금 계산
select != 0,1,2 : 예외 상황

Testcase	Select	Money	Charge	Transfer	Type
25	0	1000	Not Use	Not Use	Not Use
26	5	1000	Not Use	Not Use	Not Use

Coverage 측정

- 추가 Testcase 생성
  - calc_transfer 함수

```
Function 'calc_transfer'
Lines executed:75.00% of 4
Branches executed:100.00% of 2
Taken at least once:50.00% of 2
No calls
```

```
int calc_transfer(int transfer)
{
    switch(transfer){
    case 0:
    case 1:
    case 2:
    case 3:
        return transfer*100;
    default:
        return 0;
    }
}
```

transfer = 0,1,2,3 : 정상적 처리
transfer = 4 : 예외 상황

Testcase	Select	Money	Charge	Transfer	Type
27	2	1000	Not Use	4	2

Coverage 측정

- 추가 Testcase 생성
  - calc_fee 함수

```
Function 'calc_fee'
Lines executed:83.33% of 6
Branches executed:100.00% of 4
Taken at least once:75.00% of 4
No calls
```

```
int calc_fee(int trans_fee, int type)
{
    switch(type){
    case 1:
        return (500+trans_fee);
    case 2:
        return (900+trans_fee);
    case 3:
        return (1700+trans_fee);
    default:
        return 0;
    }
}
```

type = 1,2,3 : 정상적 처리
type = 4 : 예외 상황

Testcase	Select	Money	Charge	Transfer	Type
28	2	5000	Not Use	1	4

Coverage 측정

❖ 추가된 Testcase 결과 및 Coverage

	Running test execute_test_025 ...	Passed
	Running test execute_test_026 ...	Passed
	Running test execute_test_027 ...	Passed
	Running test execute_test_028 ...	Failed
File Name	testcase.c	**Line Number** 387
Condition	total_fee == 100	
	Running test execute_test_028 ...	Failed
File Name	testcase.c	**Line Number** 389
Condition	money == 4900	

Cumulative Summary for Run

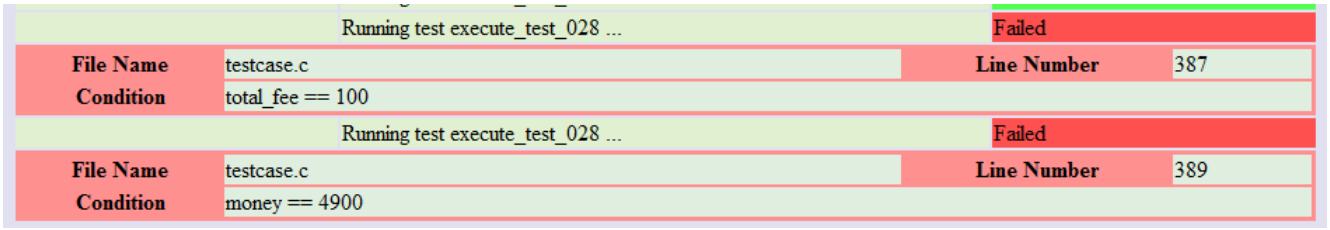
Type	Total	Run	Succeeded	Failed	Inactive
Suites	1	1	- NA -	0	0
Test Cases	28	28	24	4	0
Assertions	82	82	77	5	n/a

```
File 'testBusfee.c'
Lines executed:85.71% of 77
Branches executed:100.00% of 18
Taken at least once:100.00% of 18
Calls executed:82.93% of 41
testBusfee.c:creating 'testBusfee.c.gcov'
```

Coverage 측정

- 테스트 결과 분석
  - execute_test_28() : Fail

Testcase	Select	Money	Charge	Transfer	Type
28	2	5000	Not Use	1	4



  - Type=4 라는 예외상황 발생 시
    total_fee 값은 환승 비용뿐인 100을 예상하였으나
    type 처리 과정에서 환승 비용도 제거된 0으로 처리

Coverage 측정

❖ Coverage 결과 분석

```
File 'testBusfee.c'
Lines executed:85.71% of 77
Branches executed:100.00% of 18
Taken at least once:100.00% of 18
Calls executed:82.93% of 41
testBusfee.c:creating 'testBusfee.c.gcov'
```

- 추가된 테스트케이스를 통해
  branch / condition coverage 100% 만족하는 테스트를 수행

- 하지만 실행되지 않은 라인(함수 호출하는 라인) 존재
  - 테스트하지 않은 UI 관련 함수에 관한 결과

Coverage 측정

❖ Coverage 결과 분석

```
function calc_fee called 14 returned 100% blocks executed 100%
       14:  114:int calc_fee(int trans_fee, int type)
        -:  115:{
       14:  116:    switch(type){
branch  0 taken 29%
branch  1 taken 36%
branch  2 taken 29%
branch  3 taken 7%
        -:  117:    case 1:
        4:  118:        return (500+trans_fee);
        -:  119:    case 2:
        5:  120:        return (900+trans_fee);
        -:  121:    case 3:
        4:  122:        return (1700+trans_fee);
        -:  123:    default:
        1:  124:        return 0;
        -:  125:    }
        -:  126:}
```

- gcov 파일 이용 시 개별적이고 상세한 정보 획득 가능

END