**화이트 박스 테스트 케이스 설계**

**[ 목적 ]**

* Exercise 1부터 2까지는 화이트 박스 테스트 케이스 설계 방법에 익숙해지기 위한 실습입니다. 먼저 실습 1에서는 논리 흐름에 근거한 테스트 케이스를 찾아내기 위하여 원시 코드에서 논리흐름을 어떻게 찾아낼 수 있는가를 느껴본 후에 실습 2에서는 각 커버리지에 맞는 테스트 케이스를 찾아보는 경험을 합니다. 실습 3은 조금 큰 시스템에 대한 테스트 케이스를 찾아 실행해 보고 그 커버리지를 측정하는 과정을 실습합니다.
* Exercise 1 은 논리 흐름을 파악하고 테스트할 경로들을 찾고 이를 구동시키는 테스트 데이터를 찾아봅니다.
* Exercise 2는 테스트 케이스 설계를 바탕으로 프로그램을 실제 테스팅하는 것이다.

**[ 실습 요령 ]**

1. 모든 실습은 각 페이지에 있는 프로그램의 알고리즘에 대한 설명을 기준으로 한다. 요구 사항이나 기본 지식을 이해하기 위하여 먼저 의사 코드를 잘 읽어보고 실시한다.
2. 수강자들은 각자 개인의 능력으로 각 문제의 테스트 케이스를 설계하여야 한다.
3. 각 문제에 대한 테스트 케이스를 표로 작성한다.
4. 설계한 테스트 케이스를 실제 실행해 보고 그 커버리지를 측정하여 제출한다.
5. 모든 설계 실습은 시간 내에 완성하여 e-Class 사이트에 제출한다.

**설계 #4-1: 테스트 경로와 테스트 케이스 찾기**

**[ 목적 ]**

* 커버리지에 대한 개념이 이해와 테스트 경로 찾기 연습

**[ 수행 방법 ]**

* 다음 논리 흐름을 가진 비만도 측정 프로그램에 대하여 논리 흐름을 찾아 그래프로 그린다.
* Statement coverage를 만족하는 경로를 찾고 이를 구동하는 테스트 데이터를 찾는다.
* Branch coverage 를 만족하는 경로를 찾고 이를 구동하는 테스트 데이터를 찾는다.

<BMI값을 이용한 비만도 측정>

* BMI 값은 자신의 몸무게(Kg) 값을 키(Meter)의 제곱으로 나눈 값을 말한다.
* BMI 값은 18세 이상인 사람에게 맞도록 설계되어 있다.
* 건강한 사람의 BMI 값은 20에서 25 사이에 있다.
* BMI 지수가 20 미만인 사람은 저체중이다.
* BMI 지수가 25를 넘는 사람은 과체중(비만)이다.

<BMI 지수를 이용한 비만도 판단 프로그램>

1. *Read Weight (w)*
2. *Read Height (h)*
3. *IF w > 400 THEN*
4. *Print “invalid weight”*
5. *ENDIF*
6. *If h > 3 THEN*
7. *Print “invalid height”*
8. *ENDIF*
9. *BMI=w/(h\*h)*
10. *Print ( “BMI = “ BMI)*
11. *IF BMI < 20 THEN*
12. *Print “underweight”*
13. *ELSE IF BMI >= 20 AND BMI <=25 THEN*
14. *Print “ideal weight”*
15. *ELSE IF BMI > 25 THEN – note this statement is now redundant*
16. *Print “overweight”*
17. *ENDIF*

**문제 1**

위 비만도 측정 프로그램의 논리 흐름을 그래프로 그려라.

|  |
| --- |
| **Worksheet #4-1: 논리 흐름 그래프 찾기(BMI 프로그램)** |
|  |

**.**

**문제 2**

위 비만도 측정을 위한 프로그램의 논리를 테스트하기 위하여 Statement coverage를 만족시키는 최소의 테스트 케이스를 구하여 표에 기입하라.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Worksheet #4-2: 테스트 케이스(BMI 프로그램)** | | | | | | |
| Weight | Height | BMI | 예상 결과 | 수행된 문장 개수 | 문장 커버리지(%) | 수행된 문장 번호 |
| 500 | 4 | 31.25 | invalid weight  invalid height  overweight | 12 | 85.71 | 1 2 3 4 6 7 9 10 11 12 14 15 17 18 19 |
| 500 | 5 | 20 | invalid weight  invalid height  ideal weight | 11 | 78.57 | 1 2 3 4 6 7 9 10 11 12 14 16 19 |
| 500 | 6 | 13.88 | invalid weight  invalid height  underweight | 10 | 71.42 | 1 2 3 4 6 7 9 10 11 13 19 |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

**문제 3**

위 비만도 측정을 위한 프로그램의 논리를 테스트하기 위하여 Branch coverage를 만족시키는 최소의 테스트 케이스를 구하여 표에 기입하라.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Worksheet #4-3: 테스트 케이스(BMI 프로그램)** | | | | | | | | |
| Weight | Height | BMI | 예상 결과 | Weight  > 400 | Height >  3 | BMI < 20 | 20<BMI <=25 | BMI > 25 |
| 500 | 4 | 31.25 | invalid weight  invalid height  overweight | T | T | F | F | T |
| 80 | 2 | 20 | ideal weight | F | F | F | T | F |
| 80 | 4 | 5 | invalid height  underweight | F | F | T | F | F |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |

**설계 #4-2: 화이트 박스 테스팅**

### 목적 : 화이트 박스 테스팅을 위한 테스트 케이스 설계와 실행

**수행 방법**

* 다음은 계량 단위를 변환하기 위한 프로그램이다. Branch coverage를 만족시키는 테스트 케이스를 설계하고 이를 이용하여 테스팅 하라.

/\*

\* Converts measurements given in one unit to any other unit of the same

\* category that is listed in the database file, units.dat.

\* Handles both names and abbreviations of units.

\*/

#include <stdio.h>

#include <string.h>

#define NAME\_LEN 30 /\* storage allocated for a unit name \*/

#define ABBREV\_LEN 15 /\* storage allocated for a unit abbreviation \*/

#define CLASS\_LEN 20 /\* storage allocated for a measurement class \*/

#define NOT\_FOUND -1 /\* value indicating unit not found \*/

#define MAX\_UNITS 20 /\* maximum number of different units handled \*/

typedef struct { /\* unit of measurement type \*/

char name[NAME\_LEN]; /\* character string such as "milligrams" \*/

char abbrev[ABBREV\_LEN];/\* shorter character string such as "mg" \*/

char class[CLASS\_LEN]; /\* character string such as "pressure", "distance", "mass" \*/

double standard; /\* number of standard units equivalent to this unit \*/

} unit\_t;

int fscan\_unit(FILE \*filep, unit\_t \*unitp);

void load\_units(int unit\_max, unit\_t units[], int \*unit\_sizep);

int search(const unit\_t units[], const char \*target, int n);

double convert(double quantity, double old\_stand, double new\_stand);

int

main(void)

{

unit\_t units[MAX\_UNITS]; /\* units classes and conversion factors\*/

int num\_units; /\* number of elements of units in use \*/

char old\_units[NAME\_LEN], /\* units to convert (name or abbrev) \*/

new\_units[NAME\_LEN]; /\* units to convert to (name or abbrev)\*/

int status; /\* input status \*/

double quantity; /\* value to convert \*/

int old\_index, /\* index of units element where

old\_units found \*/

new\_index; /\* index where new\_units found \*/

/\* Load units of measurement database \*/

load\_units(MAX\_UNITS, units, &num\_units);

/\* Convert quantities to desired units until data format error

(including error code returned when q is entered to quit) \*/

printf("Enter a conversion problem or q to quit.\n");

printf("To convert 25 kilometers to miles, you would enter\n");

printf("> 25 kilometers miles\n");

printf(" or, alternatively,\n");

printf("> 25 km mi\n> ");

for (status = scanf("%lf%s%s", &quantity, old\_units, new\_units);

status == 3;

status = scanf("%lf%s%s", &quantity, old\_units, new\_units)) {

printf("Attempting conversion of %.4f %s to %s . . .\n",

quantity, old\_units, new\_units);

old\_index = search(units, old\_units, num\_units);

new\_index = search(units, new\_units, num\_units);

if (old\_index == NOT\_FOUND)

printf("Unit %s not in database\n", old\_units);

else if (new\_index == NOT\_FOUND)

printf("Unit %s not in database\n", new\_units);

else if (strcmp(units[old\_index].class,

units[new\_index].class) != 0)

printf("Cannot convert %s (%s) to %s (%s)\n",

old\_units, units[old\_index].class,

new\_units, units[new\_index].class);

else

printf("%.4f%s = %.4f %s\n", quantity, old\_units,

convert(quantity, units[old\_index].standard,

units[new\_index].standard),

new\_units);

printf("\nEnter a conversion problem or q to quit.\n> ");

}

return (0);

}

/\*

\* Gets data from a file to fill output argument

\* Returns standard error code: 1 => successful input, 0 => error,

\* negative EOF value => end of file

\*/

int

fscan\_unit(FILE \*filep, /\* input - input file pointer \*/

unit\_t \*unitp) /\* output - unit\_t structure to fill \*/

{

int status;

status = fscanf(filep, "%s%s%s%lf", unitp->name,

unitp->abbrev,

unitp->class,

&unitp->standard);

if (status == 4)

status = 1;

else if (status != EOF)

status = 0;

return (status);

}

/\*

\* Opens database file units.dat and gets data to place in units until end

\* of file is encountered. Stops input prematurely if there are more than

\* unit\_max data values in the file or if invalid data is encountered.

\*/

void

load\_units(int unit\_max, /\* input - declared size of units \*/

unit\_t units[], /\* output - array of data \*/

int \*unit\_sizep) /\* output - number of data values

stored in units \*/

{

FILE \*inp;

unit\_t data;

int i, status;

/\* Gets database of units from file \*/

inp = fopen("units.dat", "r");

i = 0;

for (status = fscan\_unit(inp, &data);

status == 1 && i < unit\_max;

status = fscan\_unit(inp, &data)) {

units[i++] = data;

}

fclose(inp);

/\* Issue error message on premature exit \*/

if (status == 0) {

printf("\n\*\*\* Error in data format \*\*\*\n");

printf("\*\*\* Using first %d data values \*\*\*\n", i);

} else if (status != EOF) {

printf("\n\*\*\* Error: too much data in file \*\*\*\n");

printf("\*\*\* Using first %d data values \*\*\*\n", i);

}

/\* Send back size of used portion of array \*/

\*unit\_sizep = i;

}

/\*

\* Searches for target key in name and abbrev components of first n

\* elements of array units

\* Returns index of structure containing target or NOT\_FOUND

\*/

int

search(const unit\_t units[], /\* array of unit\_t structures to search \*/

const char \*target, /\* key searched for in name and abbrev

components \*/

int n) /\* number of array elements to search \*/

{

int i,

found = 0, /\* whether or not target has been found \*/

where; /\* index where target found or NOT\_FOUND \*/

/\* Compare name and abbrev components of each element to target \*/

i = 0;

while (!found && i < n) {

if (strcmp(units[i].name, target) == 0 ||

strcmp(units[i].abbrev, target) == 0)

found = 1;

else

++i;

}

/\* Return index of element containing target or NOT\_FOUND \*/

if (found)

where = i;

else

where = NOT\_FOUND;

return (where);

}

/\*

\* Converts one measurement to another given the representation of both

\* in a standard unit. For example, to convert 24 feet to yards given a

\* standard unit of inches: quantity = 24, old\_stand = 12 (there are 12

\* inches in a foot), new\_stand = 36 (there are 36 inches in a yard),

\* result is 24 \* 12 / 36 which equals 8

\*/

double

convert(double quantity, /\* value to convert \*/

double old\_stand, /\* number of standard units in one of quantity's original units \*/

double new\_stand) /\* number of standard units in 1 new unit \*/

{

return (quantity \* old\_stand / new\_stand);

}

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | fscan\_unit | | load\_unit | |  | | | | | | search | | main | | | |
|  | status == 4 | status != EOF | status == 0 | status != EOF | quantity | old-units | new\_units | old\_index | new\_index |  | strcmp(units[i].name, target) == 0 || strcmp(units[i].abbrev, target) == 0 | found | old\_index == NOT\_FOUND | new\_index == NOT\_FOUND | strcmp(units[old\_index].cls, units[new\_index].cls) != 0 | (else 문)  strcmp(units[old\_index].cls, units[new\_index].cls) == 0 |
| [1] | T | F | F | T | 25 | nm | km | NOT\_FOUND | 10 | old\_index | F | F | T | F | F | F |
| new\_index | T | T |
| [2] | F | T | T | F | 25 | km | nm | 10 | NOT\_FOUND | old\_index | T | T | F | T | F | F |
| new\_index | F | F |
| [3] | T | F | F | F | 25 | km | gram | 10 | 1 | old\_index | T | T | F | F | T | F |
| new\_index | T | T |
| [4] | T | F | F | F | 25 | km | m | 10 | 9 | old\_index | T | T | F | F | F | T |
| new\_index | T | T |

1. 사용한 units.dat 파일

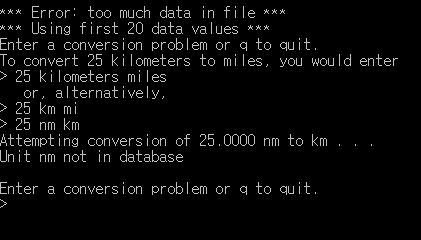
|  |  |  |
| --- | --- | --- |
| [1]에서 사용한 units.dat 내용 | [2]에서 사용한 units.dat 내용 | [3], [4]에서 사용한 units.dat 내용 |
| milligram mg mass 0.000001  gram g mass 0.001  kilogram kg mass 1  ton t mass 1000  kiloton ton amss 1000000  second s time 1  minute min time 60  hour hr time 3600  day d time 86400  meter m length 1  kilometer km length 1000  centimeter cm length 0.01  millimeter mm length 0.001  inch in length 39.370079  feet ft length 3.28084  mile mi length 0.000621  ampere amp electric 1  abampere abamp electric 10  kelvin k temperature 1  mole mol substance 1  candela cd luminous 1 | milligram mg mass 0.000001  gram g mass 0.001  kilogram kg mass 1  ton t mass 1000  kiloton ton amss 1000000  second s time 1  minute min time 60  hour hr time 3600  day d time 86400  meter m length 1  kilometer km length 1000  centimeter cm length 0.01  millimeter mm length 0.001  inch in length 39.370079  feet ft length 3.28084  mile mi length 0.000621  ampere amp electric 1  abampere abamp electric 10  kelvin k temperature 1 4  mole mol substance 1 | milligram mg mass 0.000001  gram g mass 0.001  kilogram kg mass 1  ton t mass 1000  kiloton ton amss 1000000  second s time 1  minute min time 60  hour hr time 3600  day d time 86400  meter m length 1  kilometer km length 1000  centimeter cm length 0.01  millimeter mm length 0.001  inch in length 39.370079  feet ft length 3.28084  mile mi length 0.000621  ampere amp electric 1  abampere abamp electric 10  kelvin k temperature 1  mole mol substance 1 |

1. 예상되는 테스트 수행 결과

|  |  |  |  |
| --- | --- | --- | --- |
| [1]번 결과 | [2]번 결과 | [3]번 결과 | [4]번 결과 |
| (1)  \*\*\* Error: too much data in file \*\*\*  \*\*\* Using first 20 data values \*\*\* 출력  (2)  Unit nm not in database 출력 | (1)  kelvin k temperature 1 4에서 fscan\_unit 함수 중지  (2)  \*\*\* Error in data format \*\*\*  \*\*\* Using first 19 data values \*\*\* 출력  (3) Unit nm not in database 출력 | (1) Cannot convert km (length) to gram (mass) 출력 | (1) 25.0000km = 25000.0000m 출력 |

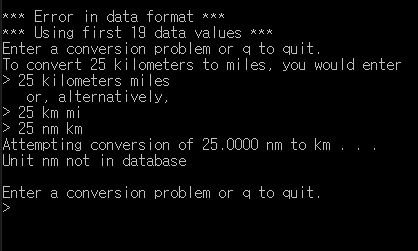
1. 테스트 수행 결과

[1]번 수행 결과



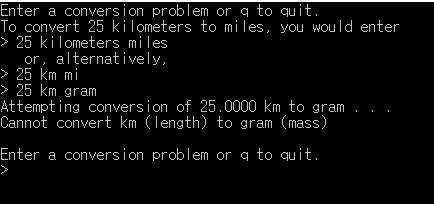
- unit의 개수가 최대 개수인 20개를 넘었기 때문에 데이터가 많다는 오류 메시지가 출력되었고, old\_units 변수에 units.dat에 없는 unit인 nm을 저장했기 때문에 database에 nm이 존재하지 않는다는 메시지가 출력되었습니다.

[2]번 수행 결과



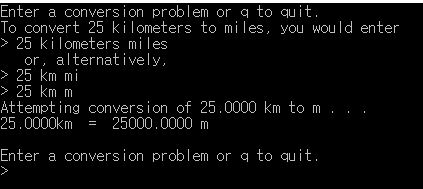
- units.dat 파일에 저장된 데이터 중 하나가 지정된 형식과 다르게 저장되었기 때문에 데이터 형식에 오류가 있다고 출력되었고, new\_units 변수에 units.dat에 없는 unit인 nm을 저장했기 때문에 database에 nm이 존재하지 않는다는 메시지가 출력되었습니다.

[3]번 수행 결과



- units.dat을 입력 받는 부분에서는 오류가 발생하지 않았으며, old\_units와 new\_units에 저장된 unit들의 계통이 서로 다르기 때문에 단위를 변환할 수 없다는 메시지가 출력되었습니다.

[4]번 수행 결과



- units.dat을 입력 받는 부분에서는 오류가 발생하지 않았으며, old\_units와 new\_units에 저장된 unit들의 계통이 같기 때문에 단위 변환이 정상적으로 작동된 모습을 볼 수 있습니다.

1. 분기 커버리지 ((수행된 조건의 수/ 전체 조건의 수) \* 100)

[1]번 : (5 / 10) \* 100 = 50%

[2]번 : (5 / 10) \* 100 = 50%

[3]번 : (4 / 10) \* 100 = 40%

[4]번 : (4 / 10) \* 100 = 40%