

# 컴파일러 과제1

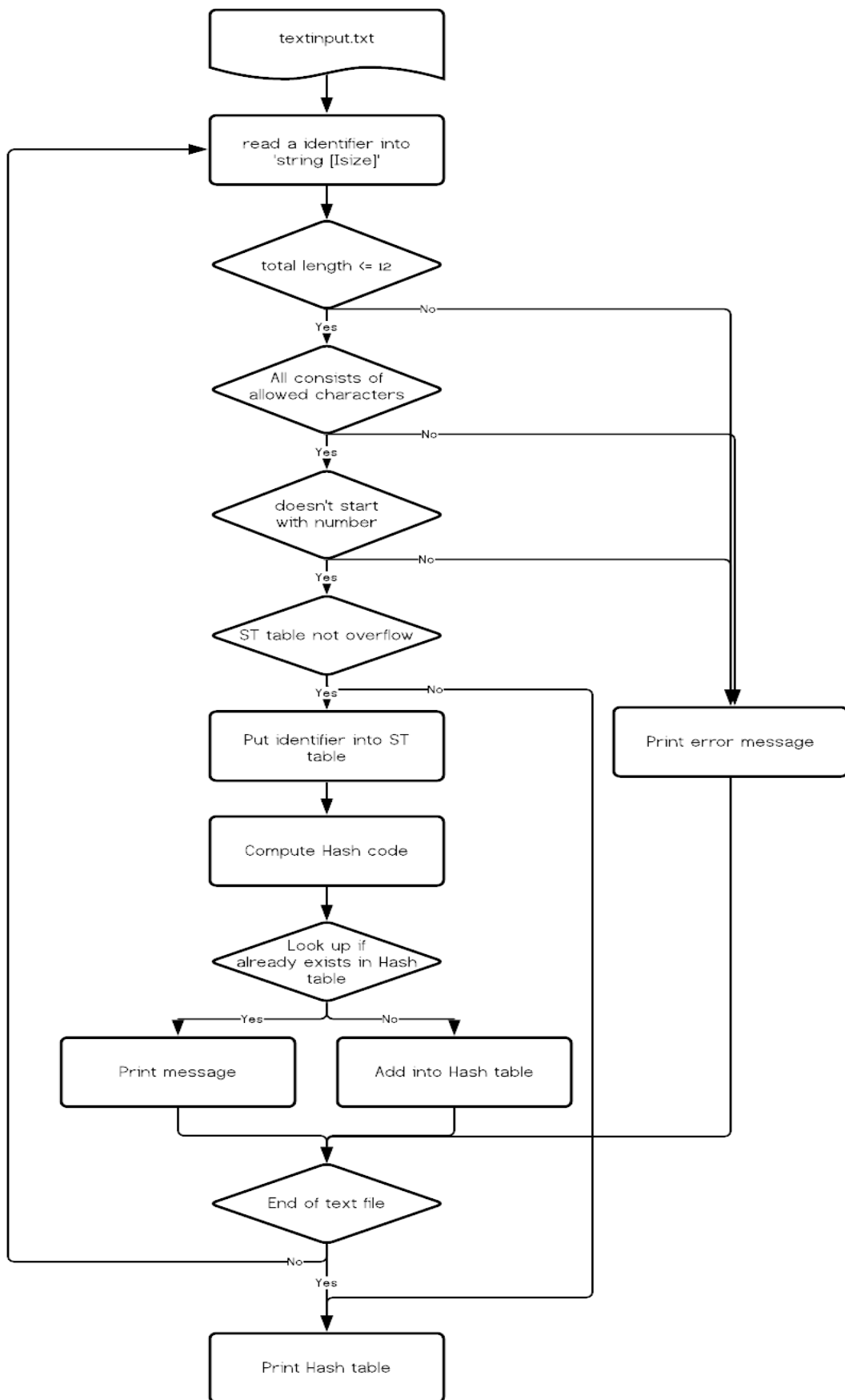
## Report

-Hash Table 구현하기 -

Team1 (박기은, 이혜인, 한지수)

## 1. 구현

### - 플로우 차트



## - 소스코드 (C Program)

```

/*****
*****
* SymbolTable Implementation (STsize = 1000)
Contributors : Team 01(박기은, 이혜인, 한지수)
Date: 2021-03-27
입력 : " .,:;?!\\t\\n" 로 구분된 식별자들로 이루어진 파일(FILE_NAME)
출력 : 식별자들과 stringtable(ST)에서의 index, 존재 정보
Global 변수 :
    ST: String Table 배열
    HT: Hash Table 로 ST 의 index 로 표현된다.
    letters: A..Z, a..z, _
    digits: 0~9
    seperators: null , . ; : ? ! \\t \\n \\r
*****
*****/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define FILE_NAME "testdata1.txt"
#define STsize 1000 //String Table 크기
#define HTsize 100 //Hash Table 크기
#define Isize 25 //입력 식별자의 최대 크기
#define isWord(x) (((x>='a'&&(x)<='z') || ((x)>='A'&&(x)<='Z')) || (x=='_'))
#define isNum(x) ((x) >= '0' && (x) <= '9')
#define idlen 12 //identifier 최대 길이 12
    typedef struct HTentry* Htpointer;
typedef struct HTentry
{
    int index; //ST 에서의 식별자 index(head index)
    Htpointer next; //다음 식별자의 포인터
} HTentry;
//에러 분류에 도움을 주는 errorTypes
enum errorTypes
{
    noerror,
    illsp,
    illid,
    overst,
    overfl,
    noidentifier
};
typedef enum errorTypes ERRORtypes;
char separators[] = " .,:;?!\\t\\r\\n";
Htpointer HT[HTsize];
char ST[STsize];
int nid = 0; //현재 identifier 위치 index
int nfree = 0; //ST 에서 다음으로 빈 index
int sameid = 0; //identifier 의 첫번째 index
int flag = 0; //0 이면 중복 없음, 1 이면 중복 있음
int EOFflag = 1;

```

```

int initalize_done = 0;
ERRORtypes errr;
FILE* fp; //파일 포인터
char input;
char string[Isize];
int hashcode; //ComputeHT 결과 저장 변수
void initialize() {
    fp = fopen(FILE_NAME, "r");
    input = fgetc(fp); initalize_done = 1;
}
//isSeperator 함수: char c 가 seperator 에 해당여부(0/1) 출력
int isSeperator(char c)
{
    for (int i = 0; i < strlen(separators); i++)
    {
        if (c == separators[i] || c == '\r')
            return 1;
    }
    return 0;
}
//isWordNum 함수: char c 가 word/Num 에 해당여부(0/1) 출력
int isWordNum(char c)
{
    if (isWord(c) || isNum(c))
        return 1;
    else
        return 0;
}
// PrintHStable 함수: HashTable 결괏값 출력 - hashcode, identifier list 와 전체 character 출력
void PrintHStable()
{
    printf("[[ HASH TABLE ]]\n\n");
    int i = 0;
    while (i < HTsize) {
        if (HT[i] != NULL) {
            printf("Hash code %-3d: ", i); //hashcode 출력
            HTpointer pt = HT[i];
            while (pt != NULL) {
                //□ hashcode 의 identifier list 출력
                int st_index = pt->index;
                while (ST[st_index] != '\0' && st_index < STsize) printf("%c", ST[st_index++]);
                printf(" ");
                pt = pt->next;
            }
            printf("\n");
        }
        i++;
    }
    printf("\n<%d characters are used in the string table>\n", nfree); //nfree 로 전체 character
출력
}
//PrintError 함수: ERRORtypes 객체 내용을 바탕으로 ERROR 정보를 출력

```

```

// error: illsp(허용되지 않는 문자 사용한 식별자 에러, SkipSeperators 함수)
//      illid(숫자와 함께 시작하는 식별자 에러, SkipStartWithNumber 함수)
//      overst(식별자길이 idlen 을 넘는 식별자 에러, get_identifier 함수)
//      overfl(파일 크기로 overflow 발생, ReadIO 함수)
void PrintError(ERRORtypes err)
{
    printf("***ERROR***\t\t");
    if (err == illsp)
    {
        for (int i = 0; string[i] != '\0'; i++) printf("%c", string[i]);
        printf(" is not allowed\n");
    }
    else if (err == illid)
    {
        for (int i = 0; string[i] != '\0'; i++) printf("%c", string[i]);
        printf(" starts with digit\n");
    }
    else if (err == overst)
    {
        for (int i = 0; string[i] != '\0'; i++) printf("%c", string[i]);
        printf(" is more than 12 words\n");
    }
    else if (err == overfl)
    {
        printf("\tOVERFLOW\n");
        printf("\n\n===== \n");
        PrintHStable();
        exit(0);
    }
}

//get_identifier 함수: identifier 단위로 접근하여 overst 에러 확인
void get_identifier()
{
    int i = 0; char temp;
    if (italize_done == 1) { temp = input; italize_done = 0; }
    else temp = fgetc(fp);
    while (1) {
        if (!isSeparator(temp)) break;
        temp = fgetc(fp);
    }
    for (i = 0; i < Isize; i++) {
        if (temp == EOF || isSeparator(temp))
        {
            string[i] = '\0';
            if (temp == EOF) EOFflag = 0;
            break;
        }
        else
        {
            string[i] = temp;
        }
        temp = fgetc(fp);
    }
}

```

```

    if (i == 0) {
        //separator 뒤에 EOF 인 경우 string 에 '\0'만 들어가있음
        errr = noidentifier;
    }
    if (i > idlen) {
        errr = overst;
        PrintError(errr);
    }
}
//SkipSeparators 함수: 허용하지 않은 문자 있는지 확인(illsp 에러 체크)
void SkipSeparators()
{
    int i = 0;
    while (string[i] != '\0')
    {
        if (!isWordNum(string[i]))
        {
            errr = illsp;
            PrintError(errr);
            break;
        }
        i++;
    }
}
//SkipStartWithNumber 함수: 숫자로 시작하는지 확인(illid 에러 체크)
void SkipStartWithNumber()
{
    if (isNum(string[0]))
    {
        errr = illid;
        PrintError(errr);
    }
}
//ReadIO 함수: FILE_NAME 에 해당하는 fp 로 파일 읽어서 ST 에 넣기(overfl 에러 체크)
void ReadID()
{
    nid = nfree;
    for (int i = 0; string[i] != '\0'; i++)
    {
        if (nfree >= STsize) //STSize overflow
        {
            errr = overfl;
            PrintError(errr);
            break;
        }
        else {
            ST[nfree++] = string[i];
        }
    }
}
//ComputeHS 함수: ST 에 존재하는 [nid~(nfree-2)]까지의 character 를 이용한 해시함수 구현
//          H(x) = (f(x) mod m)+1
void ComputeHS(int nid, int nfree)

```

```

{
    int tot_ascii = 0;
    for (int i = nid; i < nfree - 1; i++)
        tot_ascii += (int)ST[i];
    hashcode = tot_ascii % HTsize;
}

// LookupHS: identifier 의 해시 결과 중복 발생 여부에 따라 flag 값 조정
void LookupHS(int nid, int hscore)
{
    HTpointer temp;
    int a, b; //a: HT 의 index, b: 현재 identifier 의 index
    flag = 0;
    if (HT[hscore] != NULL) {
        temp = HT[hscore];
        while (temp != NULL && flag == 0) {
            flag = 1;
            a = temp->index;
            b = nid;
            sameid = a;
            while (ST[a] != '\0' && flag == 1) {
                //중복 발생
                if (ST[a] == ST[b]) {
                    a++;
                    b++;
                }
                //중복되지 않을 경우
                else flag = 0;
            }
            temp = temp->next;
        }
    }
}

// ADDHT 함수: HTpointer 할당 받아서 HS 의 hscore 에 identifier 삽입
void ADDHT(int hscore)
{
    HTpointer pt;
    pt = (HTpointer)malloc(sizeof(pt));
    pt->index = nid;
    pt->next = HT[hscore];
    HT[hscore] = pt;
}

void PrintHeading() {
    printf("=====\n");
    printf("컴파일러 과제 1\n");
    printf("1 조 - 박기은, 이해인, 한지수\n");
    printf("=====\n\n");
}

// main 함수
// FILE_NAME 에 해당하는 파일 열기
// while{
//     identifier 의 에러(overst, illsp, illid) 확인
//     1. ReadID: ST 에 identifier 저장, overfl 확인

```

```

//      2. ComputeHS: identifier 에 대한 해시 함수 결과 설정
//      3. LookupHS: 해시 함수 결과 바탕으로 HS 에 접근, flag 값 설정
//      4. ADDHT: HT 에 identifier 추가
// }
// Hash Table 결과 출력하기
int main()
{
    PrintHeading();
    int i;
    initialize();
    printf("-----\t-----\n");
    printf("Index in ST\tIdentifier\n");
    printf("-----\t-----\n");
    while (EOFflag) {
        errr = noerror;
        get_identifier(); //overst, noidentifier 확인
        SkipSeparators(); //illsp 확인
        SkipStartWithNumber(); //illid 확인
        if (errr == noerror)
        {
            ReadID(); //overfl 확인
            ST[nfree++] = '\0';
            ComputeHS(nid, nfree);
            LookupHS(nid, hashcode);
            if (!flag) {
                printf("%-11d\t\t", nid);
                i = nid;
                while (i < nfree - 1) printf("%c", ST[i++]);
                printf("...(entered)\n");
                ADDHT(hashcode);
            }
            else {
                printf("%-11d\t\t", sameid);
                i = nid;
                while (i < nfree - 1) printf("%c", ST[i++]);
                printf("...(already exists)\n");
                nfree = nid;
            }
        }
    }
    printf("\n\n=====\n");
    PrintHStable();
}

```



## 2. 채점용 테스트 데이터 실행결과

- testdata1.txt

```
=====
컴파일러 과제 1
1조 - 박기은, 이혜인, 한지수
=====

-----
Index in ST          Identifier
-----
0                    i...(entered)
2                    Hear_the...(entered)
11                   RhythmOf...(entered)
20                   waves...(entered)
26                   h1tt1n0n...(entered)
35                   the...(entered)
39                   shore...(entered)
45                   theyare____...(entered)
56                   Speakin...(entered)
64                   sPeakin...(entered)
72                   speakin...(entered)
80                   ____theyare...(entered)
91                   tell1n_me...(entered)
101                  to...(entered)
104                  relax...(entered)
110                  Imnot...(entered)
116                  goodAt_that...(entered)
128                  butnowiknowI...(entered)
141                  needIt...(entered)
148                  neeDiT...(entered)
155                  needit...(entered)

=====
[[ HASH TABLE ]]

Hash code 0 : Hear_the
Hash code 1 : needIt
Hash code 2 : h1tt1n0n
Hash code 5 : i
Hash code 15 : sPeakin Speakin
Hash code 19 : Imnot
Hash code 21 : the
Hash code 27 : to
Hash code 33 : needit
Hash code 34 : goodAt_that
Hash code 39 : ____theyare theyare____
Hash code 40 : relax
Hash code 45 : shore
Hash code 47 : speakin
Hash code 50 : waves
Hash code 69 : neeDiT
Hash code 86 : RhythmOf
Hash code 96 : butnowiknowI
Hash code 97 : tell1n_me

<162 characters are used in the string table>
```

STSize를 1000으로 놓았을 때 실행결과

```
=====
컴파일러 과제 1
1조 - 박기은, 이혜인, 한지수
=====

-----
Index in ST      Identifier
-----
0               i...(entered)
2               Hear_the...(entered)
11              RhythmOf...(entered)
20              waves   (entered)
***ERROR***      OVERFLOW

=====

[[ HASH TABLE ]]

Hash code 0 : Hear_the
Hash code 5 : i
Hash code 50 : waves
Hash code 86 : RhythmOf

<30 characters are used in the string table>
```

STSize를 30으로 놓았을 때 실행결과 (Overflow 발생 및 프로그램 종료)

- testdata2.txt

```
=====
컴파일러 과제 1
1조 - 박기은, 이혜인, 한지수
=====

-----
Index in ST      Identifier
-----
0               thisTESTDATA...(entered)
***ERROR***      isssssssssss is more than 12 words
13              fOrChecking...(entered)
25              StRing...(entered)
32              sTring...(entered)
39              lengthLength...(entered)
52              CAUTION...(entered)
60              eachwOrds...(entered)
***ERROR***      mustNOTexceed12 is more than 12 words
71              character2...(entered)

=====

[[ HASH TABLE ]]

Hash code 31 : CAUTION sTring
Hash code 42 : thisTESTDATA
Hash code 52 : lengthLength
Hash code 60 : fOrChecking
Hash code 91 : character2
Hash code 92 : eachwOrds_
Hash code 99 : StRing

<82 characters are used in the string table>
```

```
=====
컴파일러 과제 1
1조 - 박기은, 이혜인, 한지수
=====
```

```
-----
Index in ST          Identifier
-----
0                    CAUTION...(entered)
8                    COPY...(entered)
14                   RIGHT...(entered)
***ERROR***         2021 starts with digit
20                   BY...(entered)
***ERROR***         2WHA starts with digit
23                   WOMANS...(entered)
30                   UNVERS...(entered)
37                   TY...(entered)
40                   SE_LAB...(entered)
47                   ALL...(entered)
51                   rights...(entered)
58                   _REsERved...(entered)

=====
```

```
[[ HASH TABLE ]]
```

```
Hash code 17 : ALL
Hash code 31 : _REsERved CAUTION
Hash code 54 : SE_LAB
Hash code 55 : BY
Hash code 57 : rights
Hash code 69 : WOMANS
Hash code 73 : TY
Hash code 79 : COPY_
Hash code 82 : RIGHT_
Hash code 83 : UNVERS
```

```
<68 characters are used in the string table>
```

```
=====
컴파일러 과제 1
1조 - 박기은, 이혜인, 한지수
=====
```

Index in ST	Identifier
0	_l...(entered)
3	JUST...(entered)
8	called...(entered)
15	t...(entered)
***ERROR***	@say is not allowed
17	i...(entered)
***ERROR***	L%ve is not allowed
19	YOU...(entered)
23	l...(entered)
25	just...(entered)
8	called...(already exists)
30	to...(entered)
33	say...(entered)
37	HOw...(entered)
***ERROR***	much~ is not allowed
42	l...(entered)
45	CARE...(entered)
50	THANK...(entered)
***ERROR***	#u is not allowed
56	U...(entered)


```
=====
[[ HASH TABLE ]]
```

```
Hash code 5 : i
Hash code 13 : called
Hash code 16 : t
Hash code 22 : YOU
Hash code 26 : JUST
Hash code 27 : to
Hash code 33 : say
Hash code 34 : HOw_
Hash code 54 : just
Hash code 68 : l _ l
Hash code 73 : l
Hash code 74 : THANK
Hash code 83 : CARE
Hash code 85 : U
```

```
<58 characters are used in the string table>
```

### 3. 직접 작성한, '에러 없는' 입력데이터 3 개와 각각에 대한 실행 결과 (ST size=1000)

- noerr\_input1.txt

 noerr\_input1 - Windows 메모장

파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

```
compiling project assignment number1 with three team members  
team_1 gPdlS wltN rldms  
A A A
```

```
=====
컴파일러 과제 1
1조 - 박기은, 이혜인, 한지수
=====

-----
Index in ST          Identifier
-----
0                    compiling...(entered)
10                   project...(entered)
18                   assignment...(entered)
29                   number1...(entered)
37                   with...(entered)
42                   three...(entered)
48                   team...(entered)
53                   members...(entered)
61                   team_1...(entered)
68                   gPdlS...(entered)
74                   wltN...(entered)
79                   rldms...(entered)
85                   A...(entered)
85                   A...(already exists)
85                   A...(already exists)
=====

[[ HASH TABLE ]]

Hash code 6  : gPdlS
Hash code 23 : team
Hash code 36 : three
Hash code 44 : with
Hash code 46 : rldms
Hash code 47 : members
Hash code 53 : wltN
Hash code 59 : project
Hash code 62 : compiling
Hash code 65 : A
Hash code 67 : team_1
Hash code 81 : assignment
Hash code 98 : number1

<87 characters are used in the string table>
```

- noerr\_input2.txt

noerr\_input2 - Windows 메모장

파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

party ypart tpary

x y z

x\_1 x\_2 x\_3 y\_1 y\_2 y\_3 z\_1 z\_2 z\_3

up down abc cba aaa

```
=====
컴파일러 과제 1
1조 - 박기은, 이혜인, 한지수
=====
```

```
-----
Index in ST      Identifier
-----
0      party...(entered)
6      ypart...(entered)
12     tpary...(entered)
18     x...(entered)
20     y...(entered)
22     z...(entered)
24     x_1...(entered)
28     x_2...(entered)
32     x_3...(entered)
36     y_1...(entered)
40     y_2...(entered)
44     y_3...(entered)
48     z_1...(entered)
52     z_2...(entered)
56     z_3...(entered)
60     up...(entered)
63     down...(entered)
68     abc...(entered)
72     cba...(entered)
76     aaa...(entered)
=====
```

```
[[ HASH TABLE ]]
```

```
Hash code 20 : x
Hash code 21 : y
Hash code 22 : z
Hash code 29 : up
Hash code 40 : down
Hash code 60 : tpary ypart party
Hash code 64 : x_1
Hash code 65 : y_1 x_2
Hash code 66 : z_1 y_2 x_3
Hash code 67 : z_2 y_3
Hash code 68 : z_3
Hash code 91 : aaa
Hash code 94 : cba abc
```

```
<80 characters are used in the string table>
```

- noerr\_input2.txt

### noerr\_input3 - Windows 메모장

파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

gazell wolf

empty emty mpty mty pemty

n\_input n\_output n\_tot

A B C D E F G H

aaa aa1 aa2

컴파일러 과제 1  
1조 - 박기은, 이혜인, 한지수

Index in ST	Identifier
0	gazell...(entered)
7	wolf...(entered)
12	empty...(entered)
18	emty...(entered)
23	mpty...(entered)
28	mtty...(entered)
32	pemty...(entered)
38	n_input...(entered)
46	n_output...(entered)
55	n_tot...(entered)
61	A...(entered)
63	B...(entered)
65	C...(entered)
67	D...(entered)
69	E...(entered)
71	F...(entered)
73	G...(entered)
75	H...(entered)
77	aaa...(entered)
81	aa1...(entered)
85	aa2...(entered)

[ [ HASH TABLE ] ]

Hash code 39 : gazell  
Hash code 40 : wolf  
Hash code 43 : aa1  
Hash code 44 : aa2  
Hash code 46 : mty  
Hash code 47 : emty  
Hash code 48 : n\_tot  
Hash code 58 : mpty  
Hash code 59 : pemty empty  
Hash code 65 : A n\_input  
Hash code 66 : B  
Hash code 67 : C  
Hash code 68 : D  
Hash code 69 : E  
Hash code 70 : F  
Hash code 71 : G  
Hash code 72 : H  
Hash code 91 : aaa  
Hash code 94 : n\_output

<89 characters are used in the string table>

4. 직접 작성한, '에러 있는' 입력데이터 3 개와 각각에 대한 실행 결과 (ST size=1000)

- witherr\_input1.txt

witherr\_input1 - Windows 메모장

파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

abc compilingproject asng

1b1g X Y Z A+

abc cba bca

```
=====
컴파일러 과제 1
1조 - 박기은, 이혜인, 한지수
=====

-----
Index in ST      Identifier
-----
0               abc...(entered)
***ERROR***     compilingproject is more than 12 words
4               asng...(entered)
***ERROR***     1b1g starts with digit
9               X...(entered)
11              Y...(entered)
13              Z...(entered)
***ERROR***     A+ is not allowed
0               abc...(already exists)
15              cba...(entered)
19              bca...(entered)

=====

[[ HASH TABLE ]]

Hash code 25 : asng
Hash code 88 : X
Hash code 89 : Y
Hash code 90 : Z
Hash code 94 : bca cba abc

<23 characters are used in the string table>
```



- witherr\_input2.txt

witherr\_input2 - Windows 메모장

파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

x1 y2a34z hi hello  
hi\_hello\_nice\_to\_meet\_you very much  
m&m a23b c44 dc45

컴파일러 과제 1

1조 - 박기은, 이혜인, 한지수

Index in ST

Identifier

0	x1...(entered)
3	y2a34z...(entered)
10	hi...(entered)
13	hello...(entered)
***ERROR***	hi_hello_nice_to_meet_you is more than 12 words
19	very...(entered)
24	much...(entered)
***ERROR***	m&m is not allowed
29	a23b...(entered)
34	c44...(entered)
38	dc45...(entered)

[ [ HASH TABLE ] ]

Hash code 3 : c44  
Hash code 4 : dc45  
Hash code 9 : hi  
Hash code 29 : much  
Hash code 32 : hello  
Hash code 54 : very  
Hash code 69 : x1  
Hash code 93 : y2a34z  
Hash code 96 : a23b

<43 characters are used in the string table>

- witherr\_input3.txt

witherr\_input3 - Windows 메모장

파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

way ywq yay

1234 full x\_1 x\_2

lf lf\_ yay

hashtable test\_for\_the\_long\_string

x 123 t a(1)

=====

컴파일러 과제 1

1조 - 박기은, 이해인, 한지수

=====

-----

Index in ST	Identifier
-------------	------------

-----

0	way...(entered)
---	-----------------

4	ywq...(entered)
---	-----------------

8	yay...(entered)
---	-----------------

***ERROR***	1234 starts with digit
-------------	------------------------

12	full...(entered)
----	------------------

17	x_1...(entered)
----	-----------------

21	x_2...(entered)
----	-----------------

25	lf...(entered)
----	----------------

28	lf_...(entered)
----	-----------------

8	yay...(already exists)
---	------------------------

32	hashtable...(entered)
----	-----------------------

***ERROR***	test_for_the_long_string is more than 12 words
-------------	--

42	x...(entered)
----	---------------

***ERROR***	123 starts with digit
-------------	-----------------------

44	t...(entered)
----	---------------

***ERROR***	a(1) is not allowed
-------------	---------------------

=====

[ [ HASH TABLE ] ]

Hash code 5 : lf\_

Hash code 10 : lf

Hash code 16 : t

Hash code 20 : x

Hash code 35 : full

Hash code 37 : way

Hash code 39 : yay

Hash code 40 : hashtable

Hash code 53 : ywq

Hash code 64 : x\_1

Hash code 65 : x\_2

<46 characters are used in the string table>

## 5. 팀원간 기여도

모든 팀원이 각각 1/3 씩 기여

## 6. main 함수 정의 및 추가 함수

- main 함수

FILE\_NAME 에 해당하는 파일 열기

While {

1. get\_identifier, SkipSeparators, SkipStartWithNumber: identifier 의 예러(overst, illsp, illid) 확인
2. ReadID: ST 에 identifier 저장, overfl 확인
3. ComputeHS: identifier 에 대한 해시 함수 결과 설정
4. LookupHS: 해시 함수 결과 바탕으로 HS 에 접근, flag 값 설정
5. ADDHT: HT 에 identifier 추가 //

}

Hash Table 결과 출력하기

- 추가된 함수

- 1) isNum(): 입력값 char 형 데이터가 digit number(0-9)인지 체크하는 함수
- 2) isWord(): 입력값 char 형 데이터가 문자(a-z, A-Z, \_)인지 체크하는 함수
- 3) isWordNum(): 입력값 char 형 데이터가 digit number 이거나 문자인지 체크하는 함수
- 4) isSeperator(): 입력값 char 형 데이터가 미리 지정한 구분자(separators 배열)에 속하는지 체크하는 함수
- 5) SkipStartWithNumber(): 숫자로 시작하는 identifier 의 경우 예러 메시지를 출력하게 하는 함수

- 추가된 변수/상수

- 1) string 변수: 텍스트 파일에서 separator 단위로 먼저 identifier 를 읽어들이는 문자열 변수
- 2) Isize 상수: string 문자열 할당을 위한 최대 identifier 길이 상수
- 3) errorTypes noidentifier: separator 만 넘기다가 EOF 를 만났을 경우 string 에 'W0' 한 개만 읽어들이 identifier 는 없는 경우
- 4) string 변수: 텍스트 파일에서 separator 단위로 먼저 identifier 를 읽어들이는 문자열 변수
- 5) Isize 상수: string 문자열 할당을 위한 최대 identifier 길이 상수
- 6) errorTypes noidentifier: separator 만 넘기다가 EOF 를 만났을 경우 string 에 'W0' 한 개만 읽어들이 identifier 는 없는 경우
- 7) initialize\_done 변수: 초기 initialize 확인 여부를 나타내는 변수
- 8) flag 변수: 해쉬테이블에 존재하는 지 확인하는 플래그
- 9) EOFflag 변수: 텍스트 파일을 모두 읽었는지 아닌지 확인하는 플래그