컴파일러 구성 과제#1

1605020 박소현

1642041 이경연

목차

[1.Code 3](#_Toc4491779)

[2.No error in the input data file 9](#_Toc4491780)

[3.With error in the input data file 12](#_Toc4491781)

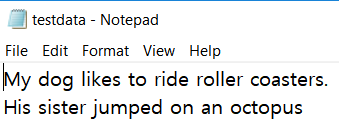
[4.Data file given from professor 15](#_Toc4491782)

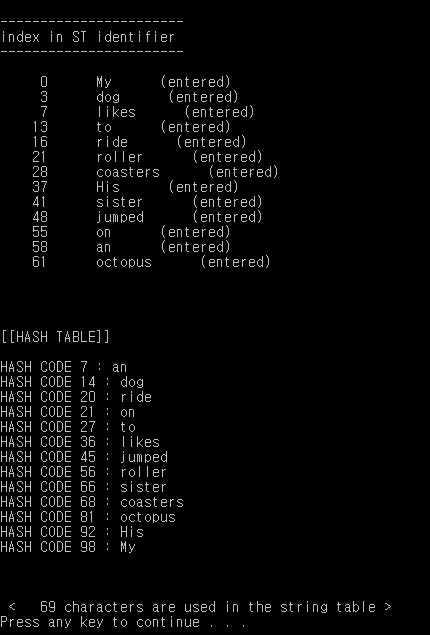
1.Code

|  |
| --- |
| * **#include <stdio.h>** * **#include <stdlib.h>** * **#include <string.h>** * **#define FILE\_NAME "testdata1.txt"** * **#define STsize 30 //size of string table** * **#define HTsize 100 //size of hash table** * **#define FALSE 0** * **#define TRUE 1** * **//letter, digit, 그리고 seperator에 대해 특정 문자가 해당 변수에 속하는지 체크할 때 사용** * **#define isLetter(ch) (((ch) >= 'a' && (ch) <= 'z') || ((ch) >= 'A' && (ch) <= 'Z') || (ch) == '\_')** * **#define isDigit(x) (((x)<='9' && (x)>='0'))** * **#define isSeperator(s) ((s) == ' ' | (s) == '\n' | (s) == '\t' | (s) == ';' | (s) == '.' | (s) == ',' | (s) == '?' | (s) == '!')** * **typedef struct HTentry \*HTpointer;** * **typedef struct HTentry {** * **int index; //index of identifier in ST** * **HTpointer next; //pointer to next identifier** * **}HTentry;** * **//나올 수 있는 여러 종류의 에러타입들을 미리 지정해둔다.** * **enum errorTypes { noerror, illsp, illid, overst, toolong };** * **typedef enum errorTypes ERRORtypes;** * **HTpointer HT[HTsize];** * **char ST[STsize];** * **int nextid = 0;** * **int nextfree = 0;** * **int hashcode = 0;** * **int sameid = 0;** * **int found;** * **ERRORtypes err;** * **FILE \*fp;** * **char input;** * **//file을 읽기 모드로 열어준 뒤, 해당 파일에서 문자 한 개를 읽어오는 함수** * **void initialize() {** * **fp = fopen(FILE\_NAME, "r");** * **input = fgetc(fp);** * **}** * **void PrintHeading() {** * **printf("\n\n");** * **printf("-----------------------\n");** * **printf("index in ST identifier\n");** * **printf("-----------------------\n");** * **printf("\n");** * **}** * **//Hash Table에 저장되어진 값들을 출력해주는 함수** * **void PrintHStable() {** * **int i, j;** * **HTpointer here;** * **printf("\n\n\n\n[[HASH TABLE]]\n\n");** * **//Hash Table의 크기만큼 반복** * **for (i = 0; i < HTsize; i++)** * **if (HT[i] != NULL) {** * **printf("HASH CODE %d : ", i);** * **//Hash Table에서 here가 NULL이 될 때까지 here->next를 통해 다음으로 계속 반복** * **for (here = HT[i]; here != NULL; here = here->next) {** * **j = here->index;** * **//String table에서 빈 칸이 나올 때까지 저장되어있는 문자값들을 출력함.** * **while (j < STsize && ST[j] != '\0') {** * **printf("%c", ST[j++]);** * **}** * **printf(" ");** * **}** * **printf("\n");** * **}** * **printf("\n\n\n <%5d characters are used in the string table > \n", nextfree);** * **}** * **//err에 저장된 값에 따라 다른 에러문이 출력되도록 해주는 함수** * **void PrintError(ERRORtypes err) {** * **//String table의 크기보다 더 많은 값들이 저장된 경우** * **if (err == overst) {** * **printf("...Error... OVERFLOW");** * **PrintHStable();** * **}** * **else if (err == illsp) //지정해준 seperator가 아닌 다른 문자가 입력될 경우** * **printf("...Error... %c is illegal seperator \n", input);** * **else if (err == toolong) //변수의 길이가 너무 긴 경우** * **printf("...Error... ");** * **//변수 이름이 숫자로 시작되는 경우** * **else if (err == illid) {** * **printf("...Error... ");** * **while (input != EOF && (isLetter(input) || isDigit(input)))** * **{** * **printf("%c", input);** * **input = fgetc(fp);** * **}** * **printf("\tStart with digit\n");** * **}** * **}** * **//seperator 값이 입력되는 경우 다음 문자를 읽어들이는 함수** * **void SkipSeperators() {** * **while (input != EOF && !(isLetter(input) || isDigit(input)))** * **{** * **//지정된 seperator가 아닐 경우에는 에러 출력 함수를 호출** * **if (!isSeperator(input)) {** * **err = illsp;** * **PrintError(err);** * **}** * **input = fgetc(fp);** * **}** * **}** * **//file로부터 문자들을 한 개씩 읽어와 string table에 저장해주는 함수** * **void ReadID() {** * **nextid = nextfree;** * **//읽어들인 input값이 숫자로 시작되는 경우 에러 출력 함수를 호출** * **if (isDigit(input)) {** * **err = illid;** * **PrintError(err);** * **}** * **else {** * **//input이 문자 또는 숫자일 경우 string table에 해당 값을 저장해준 뒤 nextfree를 1 증가시키고 file로부터 새로 읽어온다.(이를 반복)** * **while ((isLetter(input) || isDigit(input))) {** * **//nextfree가 STsize값과 똑같을 경우 string table이 가득 찼다는 얘기이므로 overflow 에러를 출력시키도록 에러 출력 함수를 호출한다.** * **if (nextfree == STsize) {** * **err = overst;** * **PrintError(err);** * **}** * **ST[nextfree] = input;** * **nextfree++;** * **input = fgetc(fp);** * **}** * **}** * **}** * **//hash code를 계산해주는 함수** * **void ComputeHS(int nid, int nfree) {** * **int code, i;** * **code = 0;** * **//string table에 저장되어있는 변수 길이만큼 for문을 반복하며 값을 더해준다.(이 때, 저장되어있는 값은 문자이므로 정수값으로 바꿔줘야 한다.)** * **for (i = nid; i < nfree - 1; i++)** * **code += (int)ST[i];** * **//hash code값이 hash table의 크기를 벗어나면 안되므로 HTsize를 나눈 나머지를 hash code로 넣는다.** * **hashcode = code % HTsize;** * **}** * **//hash table에 해당 hash code값이 있는지를 확인해주는 함수** * **void LookupHS(int nid, int hscode) {** * **HTpointer here;** * **int i, j;** * **found = FALSE; //found를 먼저 FALSE로 초기화시켜준다.** * **//hash table에서 hash code 값이 있을 경우** * **if (HT[hscode] != NULL) {** * **//hash table에 here 포인터 배치** * **here = HT[hscode];** * **while (here != NULL && found == FALSE) {** * **found = TRUE;** * **i = here->index;** * **j = nid;** * **sameid = i;** * **//다음 identifier까지 index 조정** * **while (ST[i] != '\0' && ST[j] != '\0' && found == TRUE) {** * **if (ST[i] != ST[j])** * **found = FALSE;** * **else {** * **i++;** * **j++;** * **}** * **}** * **//hash table 다음 칸으로 조정** * **here = here->next;** * **}** * **}** * **}** * **//계산된 hash code값을 hash table에 저장해주는 함수** * **void ADDHT(int hscode) {** * **HTpointer ptr;** * **ptr = (HTpointer)malloc(sizeof(ptr)); //HTpointer의 크기만큼 메모리 공간을 할당해준다.** * **ptr->index = nextid;** * **ptr->next = HT[hscode];** * **HT[hscode] = ptr;** * **}** * **int main() {** * **int i;** * **PrintHeading();** * **initialize();** * **//파일의 끝에 도달하기 전까지 while문 반복** * **while (input != EOF)** * **{** * **err = noerror;** * **SkipSeperators();** * **ReadID();** * **//잘못된 변수가 아닐 경우** * **if (err != illid) {** * **if (nextfree == STsize) {** * **err = overst;** * **PrintError(err);** * **}** * **ST[nextfree++] = '\0';** * **ComputeHS(nextid, nextfree); //읽어온 값들에 대해 hash code를 계산** * **LookupHS(nextid, hashcode); //계산된 hash code값이 hash table에 이미 있는지 확인** * **//hash table에 해당 hash code값이 없을 경우** * **if (!found&&ST[nextid] != NULL) {** * **//변수의 길이가 너무 길지 않을 경우** * **if ((nextfree - nextid - 1) <= 10) {** * **printf("%6d ", nextid);** * **for (i = nextid; i < nextfree - 1; i++) //string table에 저장되어있는 변수를 출력해준다.** * **printf("%c", ST[i]);** * **printf(" (entered)\n");** * **ADDHT(hashcode); //해당 hash code를 hash table에 저장해준다.** * **}** * **//변수의 길이가 너무 길 경우** * **else if ((nextfree - nextid - 1) > 10) {** * **err = toolong;** * **PrintError(err); //변수 길이가 긴 것에 대한 에러를 출력해준다.** * **for (i = nextid; i < nextfree - 1; i++)** * **printf("%c", ST[i]);** * **nextfree = nextid; //string table에 변수가 저장되지 않도록 nextid값을 다시 nextfree에 넣어준다.** * **printf("\t too long identifier\n");** * **}** * **}** * **//hash table에 해당 hash code값이 있는 경우** * **else if (found) {** * **printf("%6d ", sameid);** * **for (i = nextid; i < nextfree - 1; i++)** * **printf("%c", ST[i]);** * **printf(" (already existed)\n");** * **nextfree = nextid; //string table에 변수가 저장되지 않도록 nextid값을 다시 nextfree에 넣어준다.** * **}** * **}** * **}** * **PrintHStable(); //모든 것이 완료된 후 hash table을 출력해준다.** * **}** |

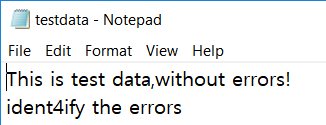
2.No error in the input data file

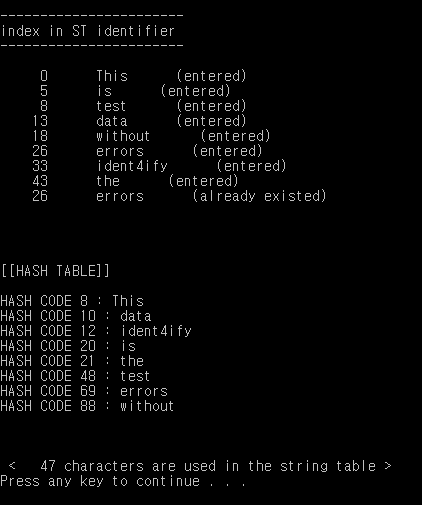
2.1 Input data\_1



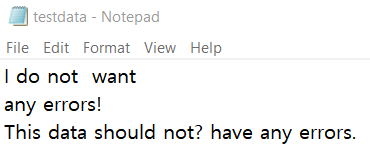


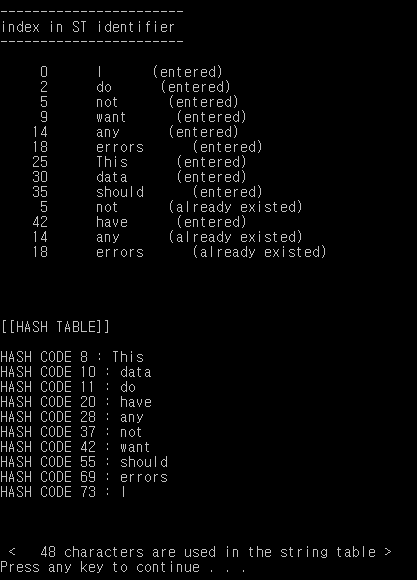
2.2 Input data\_2





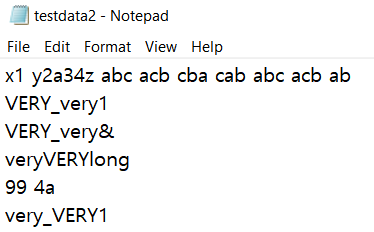
2.3 Input data\_3

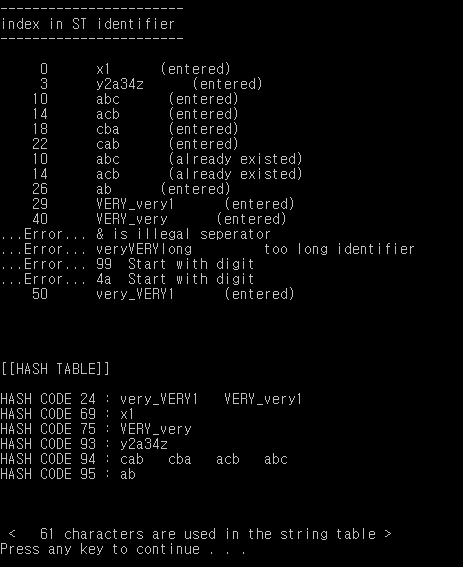




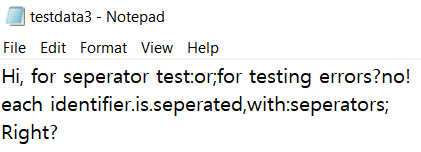
3.With error in the input data file

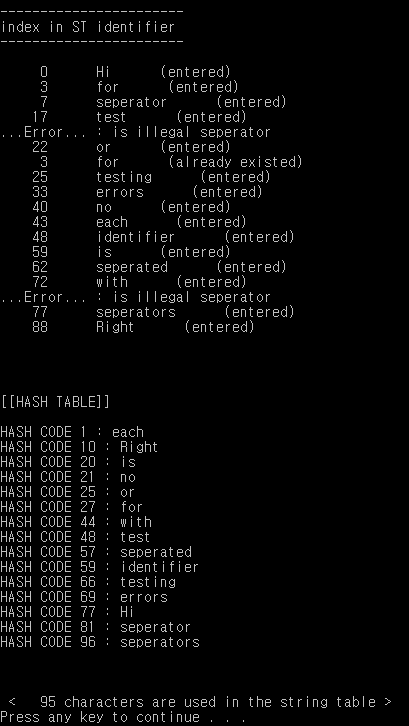
3.1 Input data\_1



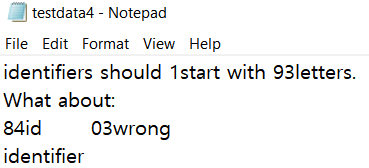


3.2 Input data\_2





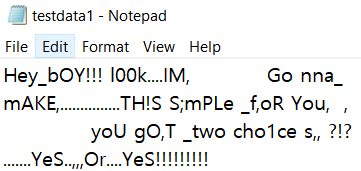
3.3 Input data\_3



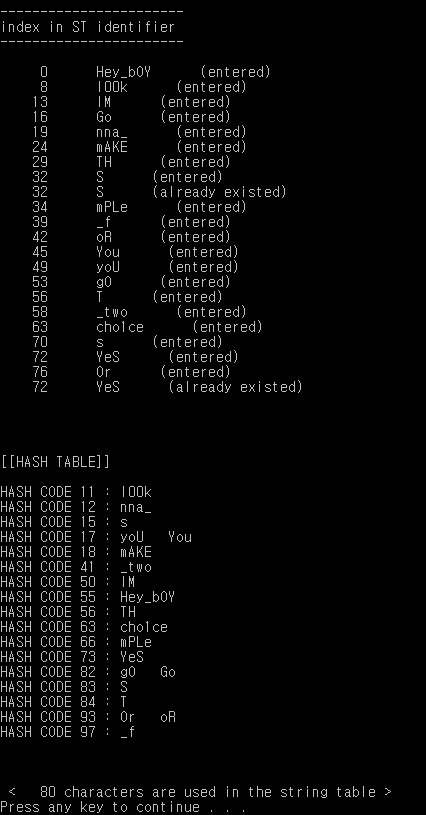


4.Data file given from professor

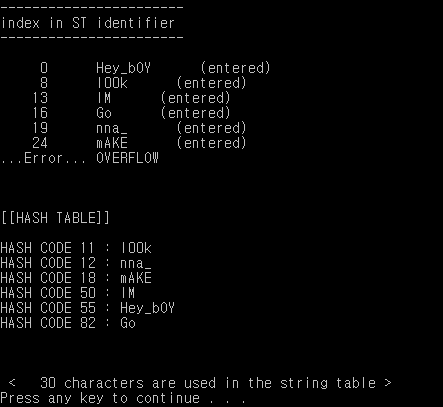
4.1 Input data\_1



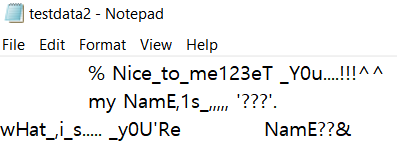
1) ST = 1000일 때

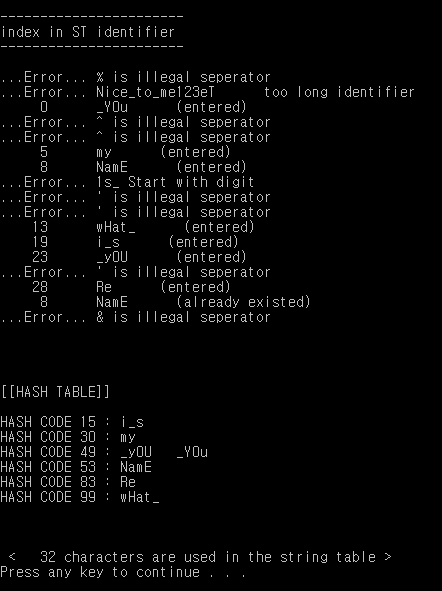


2) ST = 30일 때

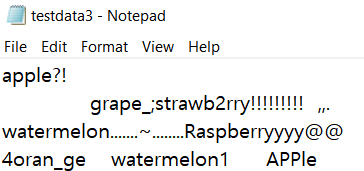


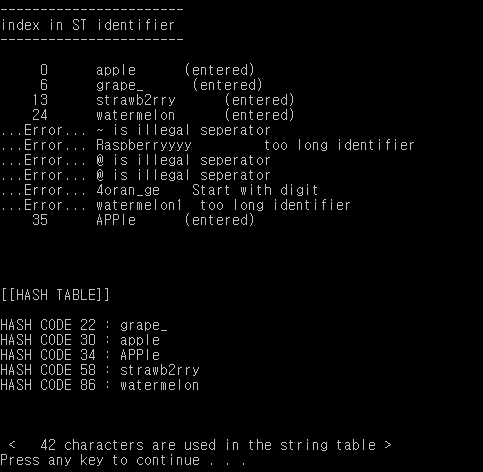
4.2 Input data\_2





4.3 Input data\_3





4.4 Input data\_4

