2021-1 컴파일러 01분반

최병주 교수님

과제 1 - Symbol Table

팀 6

1876375정하늘

1971039이진경

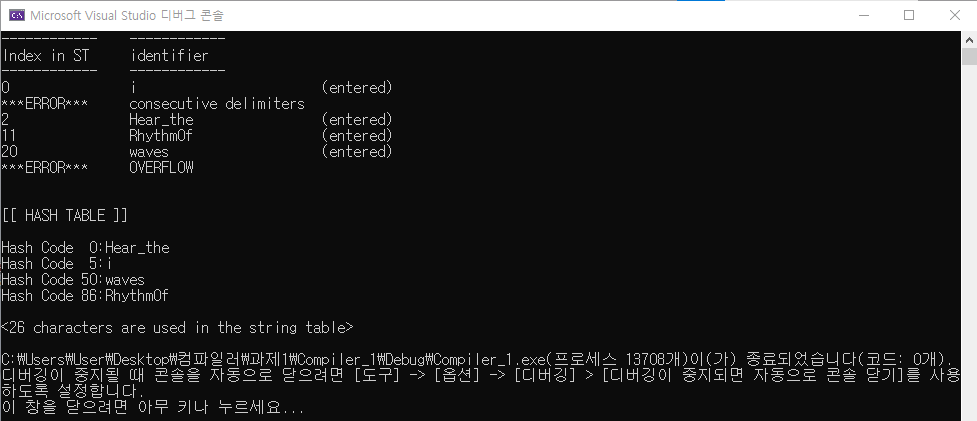
1971051최수정

1. SymbolTable.c

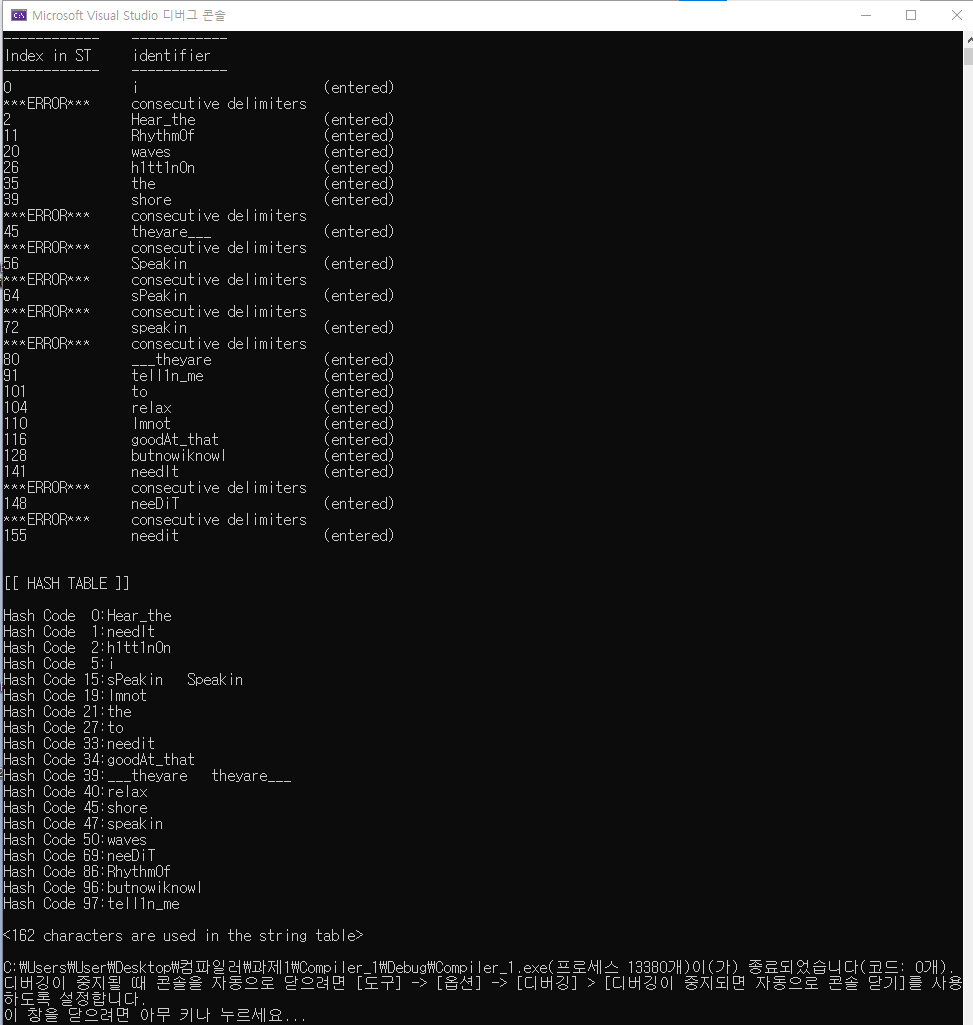
|  |
| --- |
| /\*  컴파일러(01) 과제 1  팀 6: 1876375정하늘, 1971039이진경, 1971051최수정  과제 제출일: 2021/03/30  \*/  #define \_CRT\_SECURE\_NO\_WARNINGS  #include <stdio.h>  #include <stdlib.h>  #include <string.h>  #define FILE\_NAME "testdata.txt"  #define STsize 1000  #define HTsize 100  /\* HT 관련 구조체 \*/  typedef struct HTentry\* HTpointer;  typedef struct HTentry {  int index; // ST내 시작 인덱스  HTpointer next; // 다음 id 포인터  } HTentry;  /\* 에러 정의 열거형 \*/  typedef enum errorTypes {  //문제없음, 구분자연속, id숫자시작, id허락되지않은문자, id길이초과, 오버플로우  noerror, illsp, swdigit, illid, longid, overst  }errorTypes;  char ST[STsize]; // 스트링 테이블  HTpointer HT[HTsize]; // 해쉬 테이블  FILE\* rfp; // 파일 포인터  char seperator[9] = { ' ', '\t', '.', ',', ';', ':', '?', '!', '\n' }; // 구분자 목록  char input; // 현재 읽고 있는 character  int start = 0, end = 0; // 현재 identifier의 ST 내 시작, 끝 위치  errorTypes err; // 현재 에러를 담고 있는 변수  /\* 에러 출력 \*/  void printError() {  int i;  if (err == noerror) return;  printf("\*\*\*ERROR\*\*\*\t");  if (err == illsp) { // 구분자만 연속하여 들어오는 에러  printf("%-20s\t\n", "consecutive delimiters");  }  else if (err == longid) { // id 길이 초과 에러  printf("%-20s\t%s\n", ST + start, "too long identifier");  }  else if (err == swdigit) { // id 숫자로 시작하는 에러  printf("%-20s\t%s\n", ST + start, "start with digit");  }  else if (err == illid) { // id에 잘못된 character 에러  printf("%-20s\t", ST + start);  for (i = start; i < end - 1; i++) { //잘못된 character들 출력  if (!isLetter(ST[i]) && !isNumber(ST[i])) {  printf("%c ", ST[i]);  }  }  printf("is not allowed\n");  }  else if (err == overst) { // 오버플로우 에러  printf("OVERFLOW\n");  }  return;  }  /\* 파일 열기 \*/  void initialize() {  fopen\_s(&rfp, FILE\_NAME, "r");  if (rfp == NULL) {  printf("파일 열기 실패\n");  exit(1); //강제 종료  }  input = fgetc(rfp);  }  /\* 구분자 여부 확인 \*/  int isSeperator(char c) {  int i;  for (i = 0; i < sizeof(seperator); i++) {  if (c == seperator[i]) return 1;  }  return 0;  }  /\* 알파벳/\_ 여부 확인 \*/  int isLetter(char c) {  if ('a' <= c && c <= 'z') return 1;  if ('A' <= c && c <= 'Z') return 1;  if (c == '\_') return 1;  return 0;  }  /\* 숫자 여부 확인 \*/  int isNumber(char c) {  if ('0' <= c && c <= '9') return 1;  else return 0;  }  /\* 구분자들은 스킵하고 다음 identifier 시작위치까지 이동 \*/  void SkipSeperators() {  int cnt = 0;  while (input != EOF && isSeperator(input)) {  cnt++;  input = fgetc(rfp);  }  // illsp(구분자연속) 에러 확인  if (cnt > 1) {  err = illsp;  printError();  }  }  /\* identifier 읽기 \*/  void ReadID() {  err = noerror;  // 숫자로 시작하는 에러 체크  if (isNumber(input)) err = swdigit;  // identifier 끝까지 읽기  while (input != EOF && !isSeperator(input)) {  // 오버플로우 체크  if (end >= STsize) {  err = overst;  break;  }  // 올바르지 않은 character있는지 확인  if (!isLetter(input) && !isNumber(input))  err = illid;  ST[end++] = input;  input = fgetc(rfp);  }  // identifier 마지막에 널문자 추가  if (end >= STsize)  err = overst;  else  ST[end++] = '\0';  // 길이 초과 에러 체크  if (err != overst && (end - start - 1) > 12) err = longid;  // id 관련 에러 출력  printError();  }  /\* ST에 받아두었던 id 삭제 \*/  void deleteID() {  end = start;  }  /\* ST에 받아두었던 id 저장 확정 \*/  void insertID() {  start = end;  }  /\* 해시함수 \*/  int ComputeHS(int start, int end) {  int asciisum = 0, i;  for (i = start; i < end; i++) {  asciisum += ST[i];  }  return asciisum % HTsize;  }  /\* HT 내 존재 여부 확인 \*/  int LookupHS(int hscode, int start, int end) {  HTpointer p = HT[hscode];  char str[22];  strncpy(str, ST + start, end - start);  for (; p != NULL; p = p->next) {  if (!strcmp(ST + p->index, str)) return p->index; //존재하는 경우  }  return -1; //존재하지 않는 경우  }  /\* HT에 추가 \*/  void ADDHT(int hscode, int start) {  HTentry\* hte = (HTentry\*)malloc(sizeof(HTentry));  if (hte == NULL) {  fprintf(stderr, "malloc error\n");  exit(1);  }  hte->index = start;  hte->next = NULL;  HTpointer p = HT[hscode];  if (p == NULL) {  HT[hscode] = hte;  }  else {  hte->next = p;  HT[hscode] = hte;  }//chain의 head에 추가해야함  }  /\* ST의 헤딩부분 출력 \*/  void printHeading() {  printf("------------\t------------\n");  printf("%s", "Index in ST");  printf("\t");  printf("%s", "identifier\n");  printf("------------\t------------\n");  }  /\* 해시 테이블 출력 \*/  void PrintHStable() {  printf("\n\n[[ HASH TABLE ]]\n\n");  int i, cnt = 0;  HTpointer p;  for (i = 0; i < HTsize; i++) {  if (HT[i] == NULL) continue;  printf("Hash Code %2d:", i);  for (p = HT[i]; p != NULL; p = p->next) {  printf("%s ", (ST + p->index));  cnt++;  }  printf("\n");  }  printf("\n<%d characters are used in the string table>\n", end);  }  int main() {  initialize(); //파일열기  printHeading();  while (input != EOF) {  // identifier를 읽어 ST에 넣기  SkipSeperators();  if (input == EOF) break;  ReadID();  // ST에 오버플로우가 발생한 경우 종료  if (err == overst) {  deleteID();  break;  }  // 오류 없는 경우: identifier 중복 체크 후 HT에 추가  if (err == noerror) {  int hscode = ComputeHS(start, end);  int exist = LookupHS(hscode, start, end);  // 새로 삽입  if (exist == -1) {  printf("%d\t\t", start);  printf("%-20s", ST + start);  printf("\t%s", "(entered)");  ADDHT(hscode, start);  insertID();  }  // 이미 존재  else {  printf("%d\t\t", exist);  printf("%-20s", ST + start);  printf("\t%s", "(already entered)");  deleteID();  }  printf("\n");  }  // 오류가 발생한 경우  else {  deleteID();  }  err = noerror;  }  // 해쉬테이블 출력  PrintHStable();  fclose(rfp);  } |

2. 채점용 테스트데이터

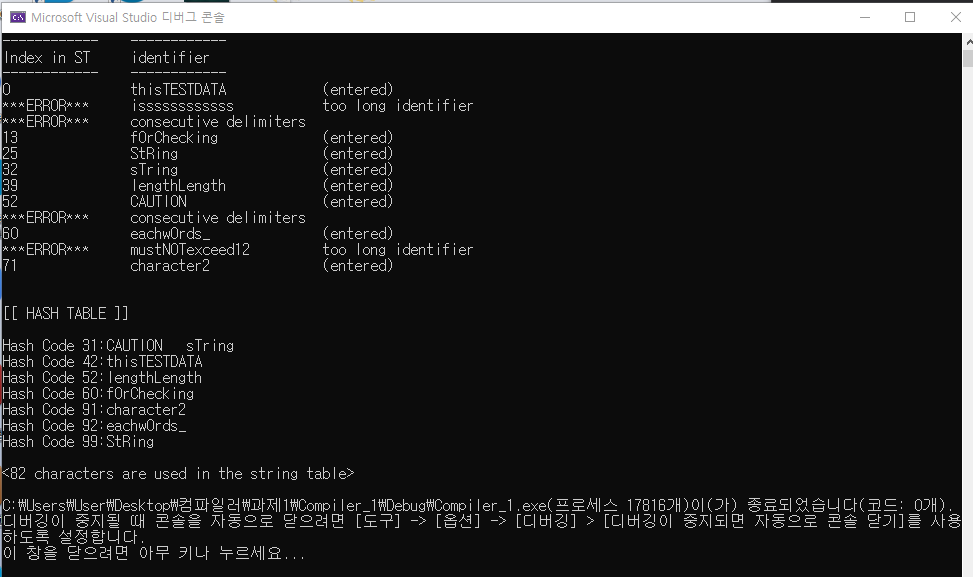
testdata1 (ST size 30)



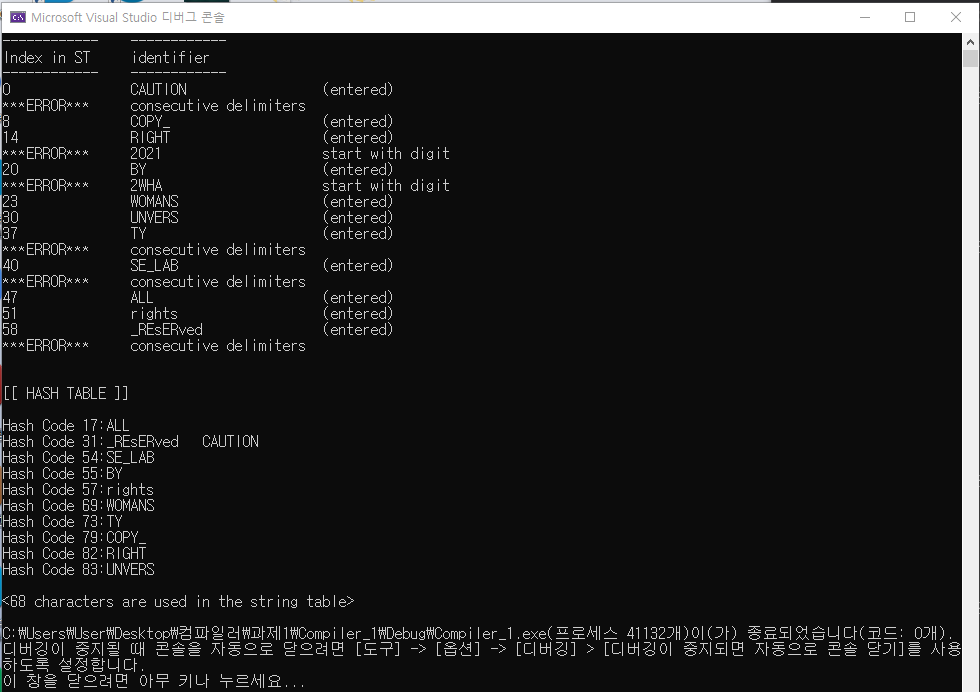
testdata1 (ST size 1000)



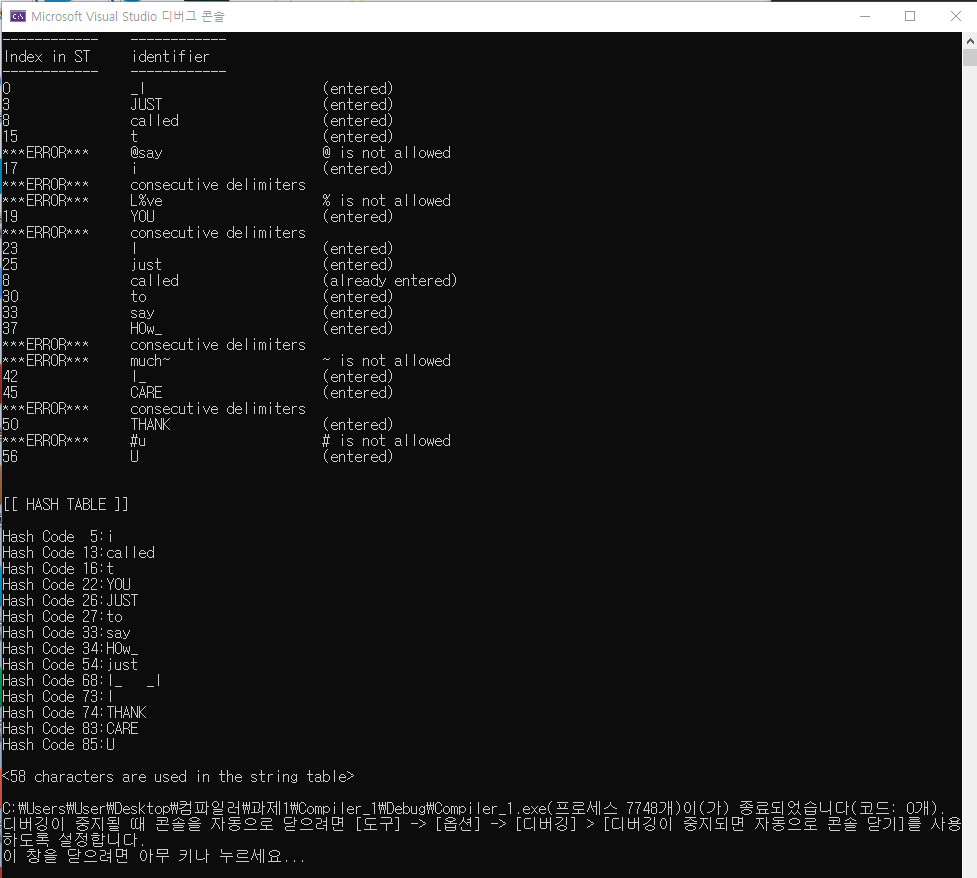
testdata2



testdata3



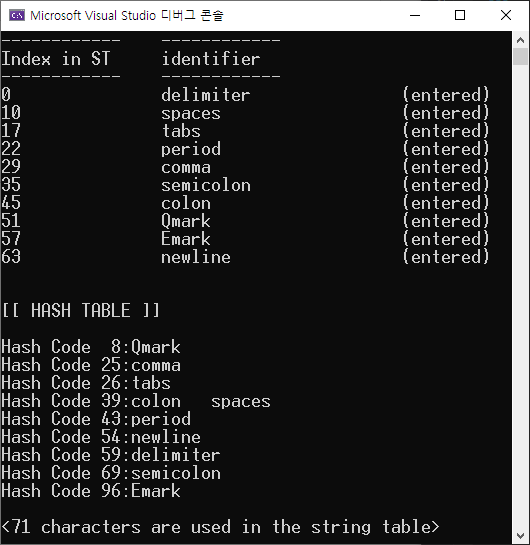
testdata4



3. 직접 작성한 ‘에러 없는’ 입력데이터 3개와 각각에 대한 실행 결과(ST size 1000)

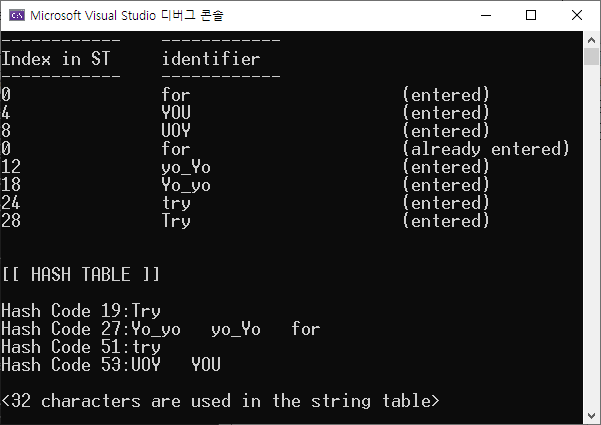
properdata1.txt

|  |
| --- |
| delimiter spaces tabs period.comma,semicolon;colon:Qmark?Emark!newline |



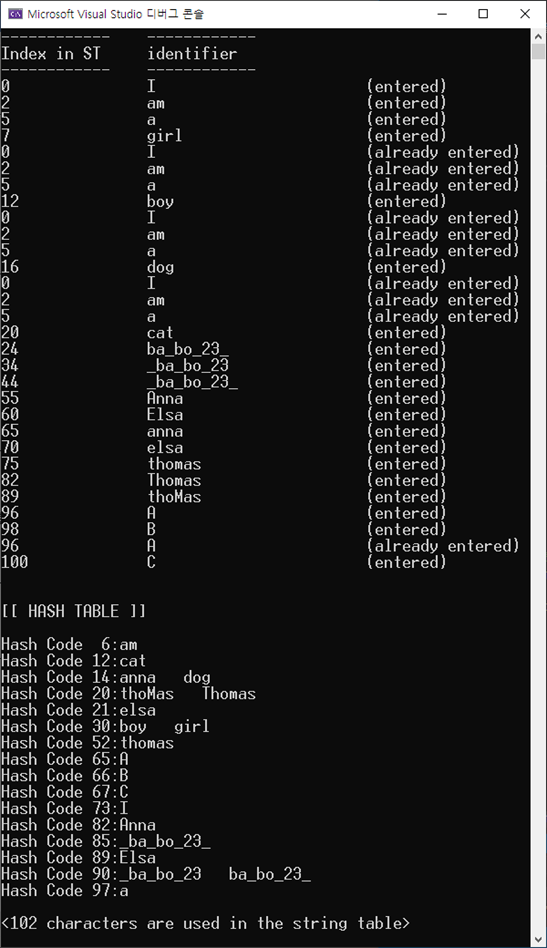
properdata2.txt

|  |
| --- |
| for:YOU  UOY for yo\_Yo  Yo\_yo.try?Try |



properdata3.txt

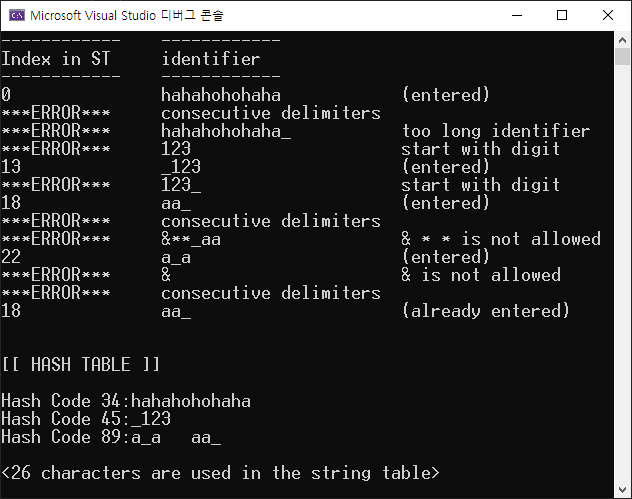
|  |
| --- |
| I am a girl  I am a boy  I am a dog  I am a cat  ba\_bo\_23\_ \_ba\_bo\_23 \_ba\_bo\_23\_  Anna,Elsa,anna,elsa,thomas,Thomas,thoMas  A!B?A.C |



4. 직접 작성한 ‘에러 있는’ 입력데이터 3개와 각각에 대한 실행 결과 (ST size 1000)

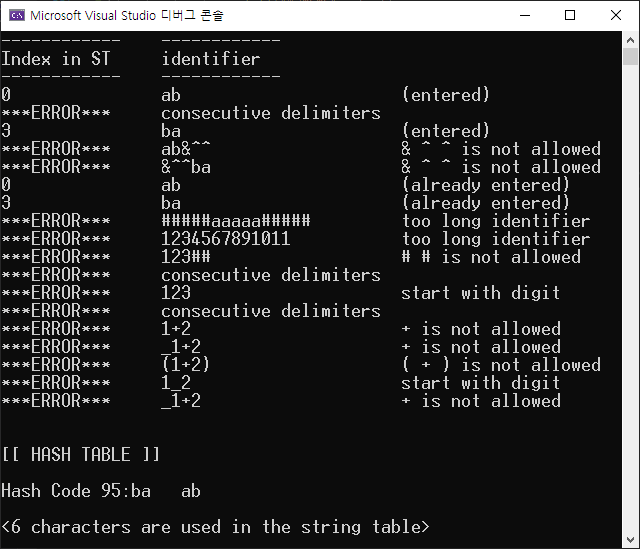
errordata1.txt

|  |
| --- |
| hahahohohaha::::: hahahohohaha\_  123;\_123 123\_  aa\_!!!!!!&\*\*\_aa a\_a!&???aa\_ |



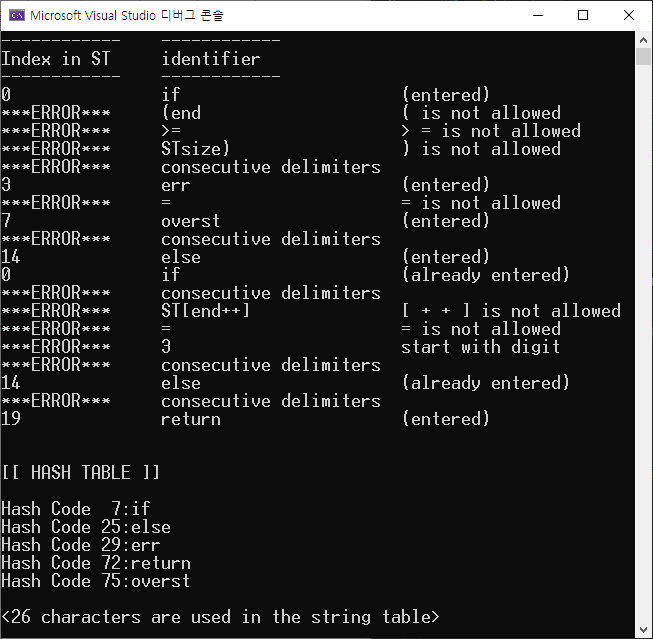
errordata2.txt

|  |
| --- |
| ab ? ba ab&^^ &^^ba ab ba  #####aaaaa#####  1234567891011  123##?????123  1+2 \_1+2 (1+2) 1\_2 \_1+2 |



errordata3.txt

|  |
| --- |
| if (end >= STsize)  err = overst;  else if  ST[end++] = 3;  else  return; |



5. 팀원간 기여도

|  |  |  |
| --- | --- | --- |
| 1876375정하늘 | 1971039 이진경 | 1971051최수정 |
| 1/3 | 1/3 | 1/3 |

6. 팀 내에서 자체적으로 정의한 부분

1) 허락되지 않은 character가 들어온 경우 모두 illid error로 처리

ex) aa&aa: 잘못된 문자 &를 포함한 잘못된 identifier, 저장하지 않음

2) illid error에서 잘못된 문자가 여러 개 들어올 경우, 잘못된 문자 모두 출력함.

ex) &\*\*\_aa: 잘못된 문자는 총 3개로, “& \* \* is not allowed” 라고 출력

3) illsp error는 구분자가 연속하여 들어오는 경우라고 정의하였음

ex) ab?!?cd: ?!?에서 illsp error 출력, ab와 cd는 정상 id로 저장

4) identifier에서 발생할 수 있는 에러는 3가지(longid, swdigit, illid), 에러가 중복하여 발생하여도 에러 우선순위가 높은 한 가지만을 출력 -> 우선순위: **longid > illid > swdigit**

ex) aaaaaaaaaaaaaaaaaa#: longid error

ex) 1111111111111111aaa: longid error

ex) 111#: illid error

5) character 개수는 널문자를 포함하여 카운트하였음