**컴파일러 과제3**

1976393 최윤지

2076278 윤지윤

2076285 이강희

**1) 전체 소스코드**

|  |
| --- |
| main.c |
| /\*  \* main.c - 각 token에 대한 출력  \* programmer – 최윤지, 윤지윤, 이강희  \* date – 06 / 01 / 2023  \*/  #include <stdio.h>  #include <stdlib.h>  #include "glob.h"  #include "tn.h"  extern int yyparse();  extern void PrintHStable();  void main() {  printf("\n[Error Report : Error Information]\n");  printf("==================================================================================\n");  // 각 변수 초기화  cLine = 1;  nextid = 0;  nextfree = 0;  // 파싱 및 에러 타입 출력  yyparse();  printf("==================================================================================\n");  printf("Parsing ends.\*\*\*");  printf("\t %d error(s) detected\n\n\n\n", cErrors);  // HS테이블 출력  PrintHStable();  } |
| parser.y |
| %{  /\*  \* parser.y - yacc source for the MiniC  \* Programmer - 최윤지 윤지윤 이강희  \* date - 06 / 01 / 2023  \*/  #include <stdio.h>  #include <stdlib.h>  #include <ctype.h>  #include <malloc.h>  #include "glob.h"  int type\_int=0;  int type\_void=0;  int param\_int=0;  extern reporterror();  extern yylex();  extern yyerror(s);  %}  %token TIDEN TNUMBER TCONST TELSE TIF TINT TRETURN TVOID TWHILE  %token TADDASSIGN TSUBASSIGN TMULASSIGN TDIVASSIGN TMODASSIGN  %token TOR TAND TEQUAL TNOTEQU TGREAT TLESS TGREATE TLESSE TINC TDEC  %token TPLUS TMINUS TMULTIPLY TDIVIDE TMOD TNOT TASSIGN TLPAREN TRPAREN TCOMMA TSEMICOLON  %token TLBRACKET TRBRACKET TLBRACE TRBRACE  %token TILLIDENT TLONG TOVER  %nonassoc LOWER\_THEN\_ELSE  %nonassoc TELSE  %%  mini\_c : translation\_unit  ;  translation\_unit : external\_dcl  | translation\_unit external\_dcl  ;  external\_dcl : function\_def  | declaration  | TIDEN TSEMICOLON  | TIDEN error  {  yyerrok;  reporterror(wrong\_st); /\* error - wrong statement \*/  }  ;  function\_def : function\_header compound\_st  |function\_header TSEMICOLON  |function\_header error /\* 비정상적인 함수 정의 \*/  {  /\* 에러 발생시 tuple 수정을 위해 default값 0세팅 \*/  /\* identifier about parse error \*/  look\_tmp->type=0;  yyerrok;  /\* error - wrong function definition \*/  reporterror(wrong\_funcdef);  }  ;  function\_header : dcl\_spec function\_name formal\_param  ;  dcl\_spec : dcl\_specifiers  ;  dcl\_specifiers : dcl\_specifier  | dcl\_specifiers dcl\_specifier  ;  dcl\_specifier : type\_qualifier  | type\_specifier  ;  type\_qualifier : TCONST  ;  /\* 해당 하는 type 변수를 1로 설정\*/  type\_specifier : TINT {type\_int=1;} /\* type : integer \*/  | TVOID {type\_void=1;} /\* type : void \*/  ;  function\_name : TIDEN  {  /\* identifier about parse error or not defined identifier/function\*/  if(look\_id->type==0||look\_id->type==5){  if(type\_void==1){  look\_id->type=4; /\* function, return type=void \*/  }else if(type\_int==1){  look\_id->type=6; /\* function, return type=int \*/  }  type\_int=0;  type\_void=0;  look\_tmp=look\_id;  }  }  /\* ill identifier, too long identifier, overflow 인 경우, scanner에서 에러 발생시킴\*/  |TILLIDENT  |TLONG  |TOVER  ;  formal\_param : TLPAREN opt\_formal\_param TRPAREN  ;  opt\_formal\_param : formal\_param\_list  |  ;  formal\_param\_list : param\_dcl  | formal\_param\_list TCOMMA param\_dcl  ;  /\* 파라미터인 경우 \*/  param\_dcl : dcl\_spec declarator{  if(type\_int==1){ /\* int의 경우 \*/  param\_int=1;  look\_id->type=7; /\* integer scalar parameter \*/  }  type\_int=0;  type\_void=0;  param\_int=0;  }  ;  compound\_st : TLBRACE opt\_dcl\_list opt\_stat\_list TRBRACE  | TLBRACE opt\_dcl\_list opt\_stat\_list error  {  yyerrok;  reporterror(nobrace); /\* no brace error \*/  }  ;  opt\_dcl\_list : declaration\_list  |  ;  declaration\_list : declaration  | declaration\_list declaration  ;  declaration : dcl\_spec init\_dcl\_list TSEMICOLON{  type\_int=0;  type\_void=0;  }  | dcl\_spec init\_dcl\_list error{  look\_tmp->type=0; /\* 예상치 못한 에러 발생 \*/  yyerrok;  type\_int=0;  type\_void=0;  reporterror(nosemi); /\* no semicolon error \*/  }  ;  init\_dcl\_list : init\_declarator  | init\_dcl\_list TCOMMA init\_declarator  ;  init\_declarator : declarator  | declarator TASSIGN TNUMBER  ;  declarator : TIDEN{  if(look\_id->type==0){  if(type\_int==1){  look\_id->type=1; /\* integer scalar variable \*/  }  else if(type\_void==1)  look\_id->type=2; /\* void scalar variable \*/  }  look\_tmp=look\_id;  }  | TIDEN TLBRACKET opt\_number TRBRACKET  {  if(look\_id->type==0){  if(type\_int==1)look\_id->type=3; /\* integer array variable \*/  }  look\_tmp=look\_id;  }  |TIDEN TLBRACKET opt\_number error{  yyerrok;  reporterror(nobracket); /\* no bracket error \*/  }  |TILLIDENT  |TLONG;  opt\_number : TNUMBER  ;  opt\_stat\_list : statement\_list  |  ;  statement\_list : statement  | statement\_list statement  | statement\_list declaration  ;  statement : compound\_st  | expression\_st  | if\_st  | while\_st  | return\_st  ;  expression\_st : opt\_expression TSEMICOLON  | expression error  {  yyerrok;  reporterror(nosemi); /\* no semicolon error \*/  }  ;  opt\_expression : expression  |  ;  if\_st : TIF TLPAREN expression TRPAREN statement %prec LOWER\_THEN\_ELSE  | TIF TLPAREN expression TRPAREN statement TELSE statement  ;  while\_st : TWHILE TLPAREN expression TRPAREN statement  ;  return\_st : TRETURN opt\_expression TSEMICOLON  ;  expression : assignment\_exp  ;  assignment\_exp : logical\_or\_exp  | unary\_exp TASSIGN assignment\_exp  | unary\_exp TADDASSIGN assignment\_exp  | unary\_exp TSUBASSIGN assignment\_exp  | unary\_exp TMULASSIGN assignment\_exp  | unary\_exp TDIVASSIGN assignment\_exp  | unary\_exp TMODASSIGN assignment\_exp  ;  logical\_or\_exp : logical\_and\_exp  | logical\_or\_exp TOR logical\_and\_exp  ;  logical\_and\_exp : equality\_exp  | logical\_and\_exp TAND equality\_exp  ;  equality\_exp : relational\_exp  | equality\_exp TEQUAL relational\_exp  | equality\_exp TNOTEQU relational\_exp  ;  relational\_exp : additive\_exp  | relational\_exp TGREAT additive\_exp  | relational\_exp TLESS additive\_exp  | relational\_exp TGREATE additive\_exp  | relational\_exp TLESSE additive\_exp  ;  additive\_exp : multiplicative\_exp  | additive\_exp TPLUS multiplicative\_exp  | additive\_exp TMINUS multiplicative\_exp  ;  multiplicative\_exp : unary\_exp  | multiplicative\_exp TMULTIPLY unary\_exp  | multiplicative\_exp TDIVIDE unary\_exp  | multiplicative\_exp TMOD unary\_exp  ;  unary\_exp : postfix\_exp  | TMINUS unary\_exp  | TNOT unary\_exp  | TINC unary\_exp  | TDEC unary\_exp  ;  postfix\_exp : primary\_exp  | postfix\_exp TLBRACKET expression TRBRACKET  | postfix\_exp TLPAREN opt\_actual\_param TRPAREN  | postfix\_exp TINC  | postfix\_exp TDEC  ;  opt\_actual\_param : actual\_param  |  ;  actual\_param : actual\_param\_list  ;  actual\_param\_list : assignment\_exp  | actual\_param\_list TCOMMA assignment\_exp  ;  primary\_exp : TIDEN  {  if(look\_id->type==0)  look\_id->type=5; /\* not defined identifier/function \*/  }  | TNUMBER  | TLPAREN expression TRPAREN  ;  %% |

|  |
| --- |
| scanner.l |
| %{  /\* scanner.l - lexical analyzer for the MiniC  \* programmer – 최윤지, 윤지윤, 이강희  \* date – 06/01/2023  \*/  #include <stdio.h>  #include <stdlib.h>  #include "tn.h" /\* token name definition \*/  #include "glob.h" /\* global variation \*/  extern ComputeHS(int nid, int nfree);  extern LookupHS(int nid, int nfree);  extern SymbolTable();  extern ADDHT(int hscode);  extern PrintHStable();  extern yyerror(s);  int i;  %}  %%  /\*  \* Keyword(예약어)  \*/  "const" {return(TCONST);}  "else" return(TELSE);  "if" return(TIF);  "int" return(TINT);  "return" return(TRETURN);  "void" {return(TVOID);}  "while" return(TWHILE);  /\*  \* ILL Identifier : 숫자로 시작하는 경우  \*/  [0-9]+[a-zA-Z\_][a-zA-Z0-9\_]\* {  yyerror("scanner error : start with digit identifier");  return (TILLIDENT);  }  /\*  \* 주석문  \*/  \/\/.\* ;  /\*  \* Operator (연산자)  \*/  "+" return(TPLUS);  "-" return(TMINUS);  "\*" return(TMULTIPLY);  "/" return(TDIVIDE);  "%" return(TMOD);  "=" {return(TASSIGN);}  "+=" return(TADDASSIGN);  "-=" return(TSUBASSIGN);  "\*=" return(TMULASSIGN);  "/=" return(TDIVASSIGN);  "%=" return(TMODASSIGN);  "!" return(TNOT);  "&&" return(TAND);  "||" return(TOR);  "==" return(TEQUAL);  "!=" return(TNOTEQU);  "<" return(TLESS);  ">" return(TGREAT);  "<=" return(TLESSE);  ">=" return(TGREATE);  "++" return(TINC);  "--" return(TDEC);  /\*  \* Special Symbols  \*/  "(" {return(TLPAREN);}  ")" { return(TRPAREN);}  "," {return(TCOMMA);}  "{" return(TLBRACE);  "}" return(TRBRACE);  "[" return(TLBRACKET);  "]" return(TRBRACKET);  ";" {return(TSEMICOLON);}  /\*  \* White spaces  \*/  [ \t] ;  [\n] {  cLine++;  }  /\*  Identifier (명칭)  \* 11자 이상일 경우, 에러 호출  \* 아닌 경우, SymbolTable 호출 (ST삽입, hashcode계산, HT삽입 수행)  \*/    [a-zA-Z\_][a-zA-Z0-9\_]\* {  if(yyleng>MaxIdentLen){  yyerror("scanner error :identifier is too long");  return (TLONG);  }  else{  SymbolTable();  return (TIDEN);  }  }  /\*  \* Constant (상수)  \*/  [0-9]+ {return(TNUMBER);}  /\*  \* illegal Symbol  \*/  . {yyerror("scanner error : illegal separator");}  %%  int yywrap(){  return 1;  } |

|  |
| --- |
| reporterror.c |
| /\*  \* reporterror.c – error에 대한 report를 따로 처리  \* progrmmer – 최윤지, 윤지윤, 이강희  \* date – 06 / 01 / 2023  \*/  #include <stdio.h>  #include <stdlib.h>  #include <string.h>  #include "tn.h"  #include "glob.h"  extern char \*yytext;  extern yylex();  /\*  \* yyerror() - parse에러 외의 에러 출력  \*/  yyerror(s)  char \*s;  {  if (s != "parse error") { //parser\_tab.c에서 발생한 parse에러가 아닌경우  printf("\t%-5d %-10s %-50s\n", cLine, yytext, s);//에러 메세지 출력  cErrors++;//에러수 1 증가  }  }  /\* reporterror  \*에러문 출력함수  \*cErrors = 에러 갯수 세는 변수, 에러가 있을 때마다 +1  \*/  void reporterror(ERRORtypes err) {  switch (err) {  case 0: //잘못된 statement  printf("%-5d %-15s %-50s\n", cLine, "< Error > =>", " Wrong Statement");  cErrors++;  break;  case 1: //올바르지 않은 함수 선언  printf("%-5d %-15s %-50s\n", cLine, "< Error > =>", " Wrong function definition");  cErrors++;  break;  case 2://semicolon 생략됨  printf("%-5d %-15s %-50s\n", cLine, "< Error > =>", " Missing semicolon");  cErrors++;  break;  case 3://중괄호 오류  printf("%-5d %-15s %-50s\n", cLine, "< Error > =>", " Missing brace");  cErrors++;  break;  case 4: //대괄호 오류  printf("%-5d %-15s %-50s\n", cLine, "< Error > =>", " Missing bracket");  cErrors++;  break;  }  } |

|  |
| --- |
| symtable.c |
| /\*  \* symtable.c – identifier들의 hashcode를 계산하고 st에 삽입  \* progrmmer – 최윤지, 윤지윤, 이강희  \* date – 06 / 01 / 2023  \*/  #include <stdio.h>  #include <string.h>  #include <stdlib.h>  #include "glob.h"  #include "tn.h"  /\* PrintHStable -  \* hash table을 형식에 맞추어 출력  \*/  void PrintHStable()  {  HTpointer here;  int i, j;  printf("\n[[HASH TABLE]]\n");  printf("==================================================================================\n");  for (i = 0; i < HTsize; i++) {  if (HT[i] != NULL) {  here = HT[i];  do {  printf("Hash Code%4d : (", i);  for (j = here->index; ST[j] != '\0'; j++) printf("%c", ST[j]);  printf(" : ");  switch (here->type) {  //int variable  case 1: printf("integer scalar variable, line%d)\n", here->cLine); break;  //void variable  case 2: printf("void scalar variable, line%d)\n", here->cLine); break;  //integer array variable  case 3: printf("integer array variable, line%d)\n", here->cLine); break;  //void function  case 4: printf("function, return type=void, line%d)\n", here->cLine); break;  //not defined  case 5: printf("not defined identifier/function, line%d)\n", here->cLine); break;  //int function  case 6: printf("function, return type=int, line%d)\n", here->cLine); break;  //int scalar parameter  case 7: printf("integer scalar parameter, line%d)\n", here->cLine); break;  default: printf("identifier about parse error, line%d)\n", here->cLine); break;  }  here = here->next;  } while (here != NULL);  }  }  printf("==================================================================================\n");  }  //identifier의 hash code들을 모두 더하여 modulo연산을 수행  void ComputeHS(int nid, int nfree) {  int code, i;  code = 0;  for (i = nid; i < nfree - 1; i++) {  code += (int)ST[i]; //hash code더하기  }  hashcode = code % HTsize; //modulo  }  //identifier가 이전에 삽입된 적 있는지 살펴봄 이미 존재한다면 ST에 이미 존재하는 시작 index를 불러옴  void LookupHS(int nid, int hscode) {  HTpointer here;  int i, j;  found = FALSE;  if (HT[hscode] != NULL) {  here = HT[hscode];  while (here != NULL && found == FALSE) {  found = TRUE;  i = here->index;  j = nid;  sameid = i;  while (ST[i] != '\0' && found == TRUE) {  if (ST[i] != ST[j]) {  found = FALSE;  }  else {  i++;  j++;  }  }  here = here->next;  }  }  }  void ADDHT(int hscode){  //hash table에 새로운 identifier를 추가  //null이었으면 바로 추가하고, 이미 존재한다면 linked list로 연결  HTpointer tmp;  if (HT[hscode] == NULL) {  tmp = (HTpointer)malloc(sizeof(struct HTentry));  tmp->type = 0;  tmp->next = NULL;  tmp->cLine = cLine;  HT[hscode] = tmp;  tmp->index = nextid;  }  else {  tmp = (HTpointer)malloc(sizeof(struct HTentry));  tmp->type = 0;  tmp->index = nextid;  tmp->next = HT[hscode];  tmp->cLine = cLine;  HT[hscode] = tmp;  }  look\_id = tmp;  }  int SymbolTable()  {  nextid = nextfree;  // yytext 한글자씩 ST에 넣음  for (int i = 0; i < yyleng; i++) {  ST[nextfree++] = yytext[i];  }  // identifier를 다 넣은 후  ST[nextfree++] = '\0';  // HS계산, 각 identifer에 대해, hastable에 이미 있는지 확인  ComputeHS(nextid, nextfree);  LookupHS(nextid, hashcode);  //처음 등장한 identifier일 경우  if (!found) {  ADDHT(hashcode); // HT 삽입  }  //identifier가 이미 존재하는 경우  else {  nextfree = nextid; //이번에 ST에 읽은 identifier를 다음에 덮어씀  }  //ST overflow 발생 시  if (nextfree == STsize) {  yyerror("scanner error : Symbol table overflow");  return (TOVER);  PrintHStable();  exit(1);  }  /\*ST출력  printf("===ST===\n");  for (int i = 0; i < nextfree; i++) {  if (ST[i] == '\0') printf("\n");  else printf("%c", ST[i]);  }  printf("\n\n\n");  \*/  } |
| glob.h |
| /\*  \* glob.h - global variable for the project  \* progrmmer – 최윤지, 윤지윤, 이강희  \* date – 06 / 01 / 2023  \*/  #define STsize 1000 // ST 크기  #define HTsize 100 // HT 크기  #define FALSE 0  #define TRUE 1  #define MaxIdentLen 10 // identifier의 최대 길이  typedef struct HTentry\* HTpointer; // HTentry를 가리키는 HTpointer  typedef struct HTentry {  int index; // ST 읽기 위해 사용  int type; // case 구분을 위해 사용  int cLine; // 코드 line  HTpointer next;  };  HTpointer HT[HTsize];  HTpointer look\_id;  HTpointer look\_tmp;  char ST[STsize];  int nextid; // 현재 identifier의 시작 index  int nextfree; // ST에서 다음 character가 삽입될 곳의 index  int hashcode;  int sameid;  int yyleng;  char\* yytext;  /\*  \* errorTypes  \* - wrong\_st: 잘못된 statement인 경우  \* - wrong\_funcdef: 잘못된 function 정의  \* - nosemi: semicolon 생략된 경우  \* - nobrace : 중괄호 생략된 경우  \* - nobracket: 대괄호 생략된 경우  \* - tlong: 11자 이상의 identifier인 경우  \* - toverflow: ST overflow가 발생한 경우  \*/  enum errorTypes { wrong\_st, wrong\_funcdef, nosemi, nobrace, nobracket, tlong, toverflow };  typedef enum errorTypes ERRORtypes;  ERRORtypes err;  /\*  \* cErrors: 에러 개수 세기  \* cLine: 코드 줄 수 세기  \* found: symtable.c 에서 사용하는 검색용 flag  \*/  int cErrors; // 에러 개수  int cLine; // 코드 line  int found; // symtable.c에서 검색하기 위해 사용 |
| tn.h |
| #ifndef YYSTYPE  #define YYSTYPE int  #endif  #define TIDEN 258  #define TNUMBER 259  #define TCONST 260  #define TELSE 261  #define TIF 262  #define TINT 263  #define TRETURN 264  #define TVOID 265  #define TWHILE 266  #define TADDASSIGN 267  #define TSUBASSIGN 268  #define TMULASSIGN 269  #define TDIVASSIGN 270  #define TMODASSIGN 271  #define TOR 272  #define TAND 273  #define TEQUAL 274  #define TNOTEQU 275  #define TGREAT 276  #define TLESS 277  #define TGREATE 278  #define TLESSE 279  #define TINC 280  #define TDEC 281  #define TPLUS 282  #define TMINUS 283  #define TMULTIPLY 284  #define TDIVIDE 285  #define TMOD 286  #define TNOT 287  #define TASSIGN 288  #define TLPAREN 289  #define TRPAREN 290  #define TCOMMA 291  #define TSEMICOLON 292  #define TLBRACKET 293  #define TRBRACKET 294  #define TLBRACE 295  #define TRBRACE 296  #define TILLIDENT 297  #define TLONG 298  #define TOVER 299  #define LOWER\_THEN\_ELSE 300  extern YYSTYPE yylval; |

**3) 직접 만든 에러 있는 테스트데이터 4개에 대한 input, output 캡쳐본**

|  |
| --- |
| 직접 만든 테스트데이터 1 (Error) |
|  |

|  |
| --- |
| 직접 만든 테스트데이터 2 (Error) |
|  |

|  |
| --- |
| 직접 만든 테스트데이터 3 (Error) |
|  |

|  |
| --- |
| 직접 만든 테스트데이터 4 (Error) |
|  |

**4) 직접 만든 에러 없는 테스트데이터 4개에 대한 input, output 캡쳐본**

|  |
| --- |
| 직접 만든 테스트데이터 1 (noError) |
|  |

|  |
| --- |
| 직접 만든 테스트데이터 2 (noError) |
|  |

|  |
| --- |
| 직접 만든 테스트데이터 3(noError) |
|  |

|  |
| --- |
| 직접 만든 테스트데이터 4 (noError) |
|  |

**5) 과제에 대한 설명**

|  |
| --- |
| 1. 아래 5가지 경우에 대해 reporterror   * wrong\_st: 잘못된 statement * wrong\_funcdef: 잘못된 function 정의 * nosemi: semicolon 생략된 경우 * nobrace : 중괄호 생략된 경우 * nobracket: 대괄호 생략된 경우   2. scanner error는 yyerror를 통한 에러 출력(ill identifier, too long identifier, overflow 인 경우, scanner에서 에러 발생시킴)   * TILLIDENT(ill identifier) : 지정된 규칙으로 생성되지않은 잘못된 identifier * TLONG(too long identifier) : 11자 이상의 identifier * TOVER (overflow) : symbol table이 overflow 된 경우 |