

**Τεχνολογικό Πανεπιστήμιο Κύπρου  
Σχολή Μηχανικής και Τεχνολογίας   
Τμήμα Ηλεκτρολόγων Μηχανικών, Μηχανικών Υπολογιστών και Πληροφορικής**

CEI 222: Αρχές Γλωσσών Προγραμματισμού και Μεταγλωττιστές

Εργαστήριο 1ο   
Φθινοπωρινό Εξάμηνο 2021  
  
  
  
**Όνομα Διδάσκων: Δρ. Ε. Κακουλλή  
Ονόματα Φοιτητών: Σωτήρης Βασιλειάδης,** **19613**

**Μιχαήλ-Άγγελος Δήμου, 19753**

**Κωνσταντίνος Κωνσταντίνου,** **20284**

**Γιώργος Τσοβίλης, 19971**

**Ημερομηνία Παράδοσης: 11/10/21**

Κατανομή ευθυνών:

* Ο Μιχαήλ-Άγγελος Δήμου σύγγραψε τα input samples για το πρόγραμμα, μάζεψε τα screenshots που χρειάστηκαν και βοήθησε με την συγγραφή της αναφοράς.
* Ο Σωτήρης Βασιλειάδης, ο Κωνσταντίνος Κωνσταντίνου, ο Μιχαήλ-Άγγελος Δήμου και ο Γιώργος Τσοβίλης διάβασαν εις βάθος το πρόβλημα, και αφού κατανόησαν σαν ομάδα τί ζητούσε η άσκηση, σκέφτηκαν τη δομή του κώδικα.
* Ο Κωνσταντίνος Κωνσταντίνου και ο Γιώργος Τσοβίλης ετοίμασαν την αναφορά, και διόρθωσαν τα παραπάνω συντακτικά/ορθογραφικά λάθη που εμφανίζονταν κατά τη διάρκεια συγγραφής του κώδικα.
* Ομαδικώς μελετήσαμε το πώς λειτουργεί η γλώσσα C- και το εργαλείο flex και δουλέψαμε μαζί για να γράψουμε τον τελικό κώδικα, κυρίως ο Σωτήρης Βασιλειάδης.

Προβλήματα που αντιμετωπίσαμε:

* Χειρισμός των comments του χρήστη.
* Runtime Error/Crash χρησιμοποιώντας int main(int argc, char \*argv[]) και strcpy για να πιάσουμε το όνομα του input file όταν χρησιμοποιούσαμε a.out < input.txt .

Πως αντιμετωπίσαμε τα προβλήματα αυτά:

* Δημιουργήσαμε μια νέα μεταβλητή int comment όπου αρχικοποιείται με την τιμή 0. Αν ο λεκτικός μας αναλυτής βρει την αρχή ενός comment (/\*) τότε μετατρέπουμε την τιμή του comment σε 1, και αγνοούμε ότι άλλο γράψει ο χρήστης μέχρι να βρεθεί το σύμβολο τερματισμού (\*/) όπου η τιμή της μεταβλητής comment γίνεται 0.
* Όταν χρησιμοποιούσαμε την εντολή a.out < input.txt δημιουργούσε πρόβλημα στο πρόγραμμα μας διότι το διαχειριζόταν ως ένα argument αντί 2, έτσι όταν προσπαθούσαμε να πάρουμε το filename της εισόδου με strcpy δεν παίρναμε τα σωστά αποτελέσματα. Έτσι για να τρέξουμε το πρόγραμμα μας χρησιμοποιούμε main αντί a.out και για είσοδο στο command line γράφουμε main input.txt(ή οτιδήποτε άλλο όνομα αρχείου που έχουμε).

Σύντομη περιγραφή κώδικα :

* Χρησιμοποιώντας τον λεκτικό αναλυτή flex για αναγνώριση λεκτικών εκφράσεων το πρόγραμμα δέχεται σαν input ένα αρχείο του οποίου ο λεκτικός αναλυτής αναγνωρίζει τις λεκτικές εκφράσεις και μετά ανάλογα με τον κανόνα που είναι σύμφωνη αυτή η έκφραση, εμφανίζει το ανάλογο μήνυμα. Στην αρχή έχουμε τα includes και τις μεταβλητές μας μέσα στα %{ %}. Μετά έχουμε τις κανονικές μας εκφράσεις και μετά τα tokens ανάμεσα στα %%. Στο τέλος έχουμε τις συναρτήσεις για τον χειρισμό των tokens και την συνάρτηση main που χειρίζεται την είσοδο και καλεί το εργαλείο flex.

**Τελικός Κώδικας:**

/\* CEI222: Project Step[1] ID: [Sotiris Vasiliadis-ID19613]\_[Michael-Aggelos Demou-ID19753]\_[Konstantinos Konstantinou-ID20284]\_[Giorgos Tsovilis-ID19971] \*/

%{

#include <stdio.h>

#include <string.h>

#include "tokens.h"

void processToken(int);

void printToken(int);

int val=1, comment=0;

char filename[80];

%}

%option yylineno

space [ \t\n]

wspaces {space}\*

letter [A-Za-z]

digit [0-9]

id {letter}{letter}\*

number {digit}{digit}\*

%%

\n {yylineno++;}

{wspaces} {processToken( WSPACES );}

"if" {processToken( IF );}

"else" {processToken( ELSE );}

"int" {processToken( INT );}

"return" {processToken( RETURN );}

"void" {processToken( VOID );}

"while" {processToken( WHILE );}

{id} {processToken( ID );}

{number} {processToken( NUMBER );}

"+" {processToken( PLUS ); }

"-" {processToken( MINUS ); }

"\*" {processToken( TIMES ); }

"/" {processToken( DIVIDE ); }

"<" {processToken( ST ); }

"<=" {processToken( SOE ); }

">" {processToken( GT ); }

">=" {processToken( GOE ); }

"==" {processToken( EQV ); }

"!=" {processToken( NEV ); }

"=" {processToken( EV ); }

";" {processToken( SEMICLN ); }

"," {processToken( CMA ); }

"(" {processToken( LPAR ); }

")" {processToken( RPAR ); }

"[" {processToken( LSQBRA ); }

"]" {processToken( RSQBRA ); }

"{" {processToken( LBRA ); }

"}" {processToken( RBRA ); }

"/\*" {processToken( LCOM ); }

"\*/" {processToken( RCOM ); }

. {printf("Unrecognised token: %s on line %d of file \"%s\".\n", yytext, yylineno, filename);}

%%

/\*Function to print our tokens\*/

void printToken(int t){

/\* IF Statement to catch and post any comments not processed/ignored (t!=26 any white spaces ignored)\*/

if(comment==1 && t!=26){

if(t!=31)

printf("%s",yytext);

else

printf(" ");

}

else

switch(t) {

case IF: printf("Found an IF on line %d of file \"%s\".\n", yylineno, filename);break;

case ELSE: printf("Found an ELSE on line %d of file \"%s\".\n", yylineno, filename);break;

case INT: printf("Found an INT on line %d of file \"%s\".\n", yylineno, filename);break;

case RETURN: printf("Found a RETURN on line %d of file \"%s\".\n", yylineno, filename);break;

case VOID: printf("Found a VOID on line %d of file \"%s\".\n", yylineno, filename);break;

case WHILE: printf("Found a WHILE on line %d of file \"%s\".\n", yylineno, filename);break;

case PLUS: printf("Found a \"+\" on line %d of file \"%s\".\n", yylineno, filename);break;

case MINUS: printf("Found a \"-\" on line %d of file \"%s\".\n", yylineno, filename);break;

case TIMES: printf("Found a \"\*\" on line %d of file \"%s\".\n", yylineno, filename);break;

case DIVIDE: printf("Found a \"/\" on line %d of file \"%s\".\n", yylineno, filename);break;

case ST: printf("Found a \"<\" on line %d of file \"%s\".\n", yylineno, filename);break;

case SOE: printf("Found a \"<=\" on line %d of file \"%s\".\n", yylineno, filename);break;

case GT: printf("Found a \">\" on line %d of file \"%s\".\n", yylineno, filename);break;

case GOE: printf("Found a \">=\" on line %d of file \"%s\".\n", yylineno, filename);break;

case EQV: printf("Found a \"==\" on line %d of file \"%s\".\n", yylineno, filename);break;

case NEV: printf("Found a \"!=\" on line %d of file \"%s\".\n", yylineno, filename);break;

case EV: printf("Found a \"=\" on line %d of file \"%s\".\n", yylineno, filename);break;

case SEMICLN: printf("Found a \";\" on line %d of file \"%s\".\n", yylineno, filename);break;

case CMA: printf("Found a \",\" on line %d of file \"%s\".\n", yylineno, filename);break;

case LPAR: printf("Found a \"(\" on line %d of file \"%s\".\n", yylineno, filename);break;

case RPAR: printf("Found a \")\" on line %d of file \"%s\".\n", yylineno, filename);break;

case LSQBRA: printf("Found a \"[\" on line %d of file \"%s\".\n", yylineno, filename);break;

case RSQBRA: printf("Found a \"]\" on line %d of file \"%s\".\n", yylineno, filename);break;

case LBRA: printf("Found a \"{\" on line %d of file \"%s\".\n", yylineno, filename);break;

case RBRA: printf("Found a \"}\" on line %d of file \"%s\".\n", yylineno, filename);break;

case LCOM: printf("Found (/\*) the start of a comment on line %d of file \"%s\".\nComments ignored: ", yylineno, filename); comment=1; break;

case RCOM: printf("\nFound (\*/) the end of a comment on line %d of file \"%s\".\n", yylineno, filename); comment=0; break;

case ID: printf("Found an ID: \"%s\" on line %d of file \"%s\".\n", yytext, yylineno, filename);break;

case NUMBER: printf("Found a NUMBER: \"%s\" on line %d of file \"%s\".\n", yytext, yylineno, filename);break;

case LETTER: printf("Found a letter: \"%s\" on line %d of file \"%s\".\n", yytext, yylineno, filename);break;

case DIGIT: printf("Found a digit: \"%s\" on line %d of file \"%s\".\n", yytext, yylineno, filename);break;

case WSPACES: break;

};

}

/\*Just a redirection function\*/

void processToken(int t){

printToken(t);

}

/\*Main Function\*/

int main(int argc, char \*argv[]) {

strcpy(filename, argv[1]); //copy file name

/\*If 2 arguments provided (main input.exe etc) then open file, else mistakes were made\*/

if (argc==2)

yyin=fopen(argv[1],"r");

if (argc<2)

{

printf("Too little arguments(files) provided - Exit\n");

exit(1);

}

if (argc>3)

{

printf("Error in opening file, too many arguments(files) provided - Exit\n");

exit(1);

}

yylex();

return 0;

}

**Input Samples:**

**Input 1:**

int a=0,i=0;

while(i<10)

{

if(a>3)

a=b;

else

a=0;

i++;

a++;

}

**Input 2:**

int Sum(int x, int y)

{

int a;

a=x+y;

return a;

}

**Input 3:**

/\* This is a test \*/

if

else

int

return

void

while

+

-

\*

/

<

<=

>

>=

==

!=

=

;

,

(

)

[

]

{

}

.if

asd

5.4

**Screenshots:**

**Output 1:**

**Text

Description automatically generated**

**Output 2:**

**Text

Description automatically generated**

**Output 3:**

Text

Description automatically generated