Αναφορά Project Flex-Bison

**Στοιχεία ομάδας:**

Όλα τα μέλη της ομάδας είναι στο 3ο έτος.

|  |  |  |
| --- | --- | --- |
| Όνομα | AM | email |
| Βασιλάκη Ελένη | 1070939 | vasilaki.eleni2000@gmail.com |
| Γιαννοπούλου Αρχοντία | 1070933 | arhodia30@gmail.com |
| Λιούμη Κυριακή | 1067410 | kyriaki.lm@gmail.com |

**BNF:** (η τελική μορφή μαζί με τα structs και τα σχόλια του ερωτήματος 4)

(ο σχολιασμός γίνεται με το κόκκινο χρώμα )

Όπου «ε» το κενό

Αρχή της γραμματικής :

<start> ::= <program>

Αρχή του προγράμματος:

<program> ::= PROGRAM <name> <lines> <body>

Δυνατότητα αλλαγής μιας γραμμής ή πολλών γραμμών :

<lines> ::= CHANGE\_LINE

| CHANGE\_LINE <lines>

Υπάρχει η επιλογή σε περίπτωση που δεν θέλουμε να βάλουμε συνάρτηση ή σε περίπτωση που δεν θέλουμε να βάλουμε structs.

<body> ::= <main>

|<structs> <main>

| <structs> <functions> <main>

Τα structs είναι προαιρετικά, οπότε υπάρχει η δυνατότητα να βάλουμε από 0 μέχρι όσες φορές θέλουμε:

<structs> ::= ε

| <struct> <structs>

Η περιπτώσεις να έχουμε ή απλό struct ή με typedef:

<struct> ::= STRUCT NAME <lines> <var> ENDSTRUCT <lines>

| TYPEDEF STRUCT NAME <lines> <var> NAME ENDSTRUCT <lines>

Περίπτωση μίας ή πολλών συναρτήσεων. Η περίπτωση που δεν θέλουμε καμία, καλύπτεται πιο πάνω στο body.

<functions> ::= <function>

|<function> <functions>

Υλοποίηση συνάρτησης:

<function> ::= FUNCTION NAME LEFTBRA <list> RIGHTBRA <lines> <var> <commands> RETURN NAME <lines>

Τα ορίσματα της συνάρτησης. Δυνατότητα επιλογής ανάμεσα σε μια μεταβλητή, πολλές μεταβλητές και πίνακα:

<list> ::= NAME

|NAME COMMA <list>

|NAME LEFTSTRBRA INT RIGHTSTRLBRA

|NAME LEFTSTRBRA INT RIGHTSTRLBRA COMMA <list>

Δυνατότητα δήλωσης πολλών μεταβλητών ή και καμίας:

<var> ::= ε

| <var2> <var>

Ορισμός μεταβλητών:

<var2> ::= VARS <lines> <multiple>

Δυνατότητα επιλογής ανάμεσα σε μια μεταβλητή, πολλές μεταβλητές και πίνακα, στην ίδια γραμμή ή σε διαφορετική γραμμή:

<multiple> ::= ε

| <type\_specifier> <list> EROTIMATIKO <lines> <multiple>

Τύποι μεταβλητών. Ο τρίτος τύπος είναι για το struct:

<type\_specifier> ::= CHAR

| INTEGER

|NAME

Δυνατότητα επιλογής πολλών εντολών :

<commands> ::= <command>

| <command> <commands>

Είδη εντολών :

<command> ::= <comment>

|<loop>

|<conditions>

|<print>

|<assignment>

|<big\_comment>

Είδη loop:

<loop> ::= <while>

|<for>

Υλοποίηση της while:

<while> ::= WHILE LEFTBRA <condition> RIGHTBRA <lines> <break> <commands> <break> ENDWHILE <lines>

Υλοποίηση της for :

<for> ::= FOR NAME ANOKATOTEL ISON INT TO INT STEP INT <lines> <break> <commands> <break> ENDFOR <lines>

Είδη condition :

<conditions> ::= <if>

|<switch>

Υλοποίηση της if:

<if> ::= IF LEFTBRA <condition> RIGHTBRA THEN <lines> <commands> <elseif2> <else> ENDIF <lines>

Υλοποίηση της else if. Δυνατότητα επιλογής 0,1 ή πολλές φορές:

<elseif2> ::= ε

| ELSEIF <lines> <commands> <elseif2>

Υλοποίηση της else . Δυνατότητα επιλογής 0 ή 1 φορές

<else> ::= ε

| ElSE <lines> <commands>

Υλοποίηση switch:

<switch> ::= SWITCH LEFTBRA <condition> RIGHTBRA <lines> <case2>

<default> ENDSWITCH <lines>

Δυνατότητα επιλογής πόσες φορές θα χρησιμοποιηθεί το case.

<case2> ::= ε

| <case> <case2>

Υλοποίηση του case.

<case> ::= CASE LEFTBRA <condition> RIGHTBRA ANOKATOTEL <lines> <break> <commands> <break>

Υλοποίηση του default. Επιλογή 0 ή 1 φορά.

<default> ::= ε

| DEFAULT ANOKATOTEL <lines> <break> <commands> <break>

Υλοποίηση του print με και χωρίς εκτύπωση μεταβλητής:

<print> ::= PRINT LEFTBRA <words> RIGHTBRA EROTIMATIKO <lines>

| PRINT LEFTBRA <words> LEFTSTRBRA COMMA NAME RIGHTSTRLBRA RIGHTBRA EROTIMATIKO <lines>

Υλοποίηση συνθηκών :

<condition> ::=NAME <symbol> NAME

| <condition> <logic> <condition>

Σύμβολά σύγκρισης

<symbol> ::= MEGALYTERO

|MIKROTERO

| ISON ISON

|THAYMASTIKO

Λογικές πράξεις:

<logic> ::= <and>

|<or>

Υλοποίηση εντολής ανάθεσης :

<assignment> ::= NAME ISON <expression> EROTIMATIKO <lines>

Επιλογή του δεύτερου μέλους της εξίσωσης. Οι επιλογές είναι : συνάρτηση, μεταβλητή, πράξη μεταξύ μεταβλητών και χρήση παρενθέσεων

<expression> ::= INT

| <func>

| NAME

| <expression> <symbol2> <expression>

| LEFTBRA <expression> RIGHTBRA

Πράξεις :

<symbol2> SYN

|PLIN

|MUL

|DIV

|POW

Κλήση συνάρτησης:

<func> ::= NAME LEFTBRA <list> RIGHTBRA

Υλοποίηση break. Δυνατότητα επιλογής 0 ή 1 φορά

<break> ::= ε

|<break> EROTIMATIKO <lines>

Υλοποίηση σχολίου

<comment> ::= TISEKATO <words> <lines>

Υλοποίηση σχολίου του 4ου ερωτήματος. Επιλογή να είναι σε μία γραμμή ή σε πολλές

<big\_comment> ::= COMSTART <words> COMEND <lines>

| COMSTART CHANGE\_LINE <test> COMEND <lines>

Δυνατότητα επιλογής τι θα βάλουμε μέσα στο σχόλιο των πολλών γραμμών. Οι επιλογές είναι : κείμενο, αλλαγή γραμμών , εντολές

<test> ::= ε

| <words> CHANGE\_LINE <test>

| <lines> <test>

| <command> <test>

Υλοποίηση της main :

<main> ::= STARTMAIN <lines> <var> <commands> ENDMAIN <lines>

Δυνατότητα να γράψουμε κείμενο με κενά ανάμεσα στις λέξεις. Χρησιμοποιείται στα σχόλια και το print.

<words> ::= NAME

| NAME <words>

**Flex:**

%{

Δήλωση βιβλιοθηκών που χρησιμοποιούνται:

#include "y.tab.h"

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

Δήλωση συναρτήσεων που χρειάζονται και αρχείων για το input και το output

extern int yylex();

extern FILE \*yyin;

extern FILE \*yyout;

Δήλωση μεταβλητής που θα μετράει τις γραμμές για να βλέπουμε σε ποια γραμμή βρίσκεται το σφάλμα.

int line\_count=1;

%}

%option noyywrap

Ορισμοί ονομάτων :

digit [0-9]

letter [a-zA-Z]

num {digit}+

%%

Δήλωση δεσμευμένων λέξεων και συμβόλων που χρησιμοποιούμε στο πρόγραμμα. Τα αντιμετωπίζουμε όλα ως string, έτσι ώστε να εκτυπώνονται οι λέξεις. Μετά κάνουμε return την συγκεκριμένη δεσμευμένη λέξη που θέλουμε κάθε φορά, ακριβώς έτσι όπως είναι (ECHO και return).

"PROGRAM" {yylval.string = strdup(yytext); ECHO;return PROGRAM;}

"STARTMAIN" {yylval.string = strdup(yytext); ECHO;return STARTMAIN;}

"VARS" {yylval.string = strdup(yytext); ECHO;return VARS;}

"CHAR" {yylval.string = strdup(yytext); ECHO;return CHAR;}

"INTEGER" {yylval.string = strdup(yytext); ECHO;return INTEGER;}

"STRUCT" {yylval.string = strdup(yytext); ECHO;return STRUCT;}

"TYPEDEF" {yylval.string = strdup(yytext); ECHO;return TYPEDEF;}

"ENDSTRUCT" {yylval.string = strdup(yytext); ECHO;return ENDSTRUCT;}

"FUNCTION" {yylval.string = strdup(yytext); ECHO;return FUNCTION;}

"RETURN" {yylval.string = strdup(yytext); ECHO;return RETURN;}

"END\_FUNCTION" {yylval.string = strdup(yytext); ECHO;return END\_FUNCTION;}

"ENDMAIN" {yylval.string = strdup(yytext); ECHO;return ENDMAIN;}

"WHILE" {yylval.string = strdup(yytext); ECHO;return WHILE;}

"ENDWHILE" {yylval.string = strdup(yytext); ECHO;return ENDWHILE;}

"FOR" {yylval.string = strdup(yytext); ECHO;return FOR;}

"TO" {yylval.string = strdup(yytext); ECHO;return TO;}

"STEP" {yylval.string = strdup(yytext); ECHO;return STEP;}

"ENDFOR" {yylval.string = strdup(yytext); ECHO;return ENDFOR;}

"IF" {yylval.string = strdup(yytext); ECHO;return IF;}

"THEN" {yylval.string = strdup(yytext); ECHO;return THEN;}

"ELSEIF" {yylval.string = strdup(yytext); ECHO;return ELSEIF;}

"ELSE" {yylval.string = strdup(yytext); ECHO;return ELSE;}

"ENDIF" {yylval.string = strdup(yytext); ECHO;return ENDIF;}

"SWITCH" {yylval.string = strdup(yytext); ECHO;return SWITCH;}

"CASE" {yylval.string = strdup(yytext); ECHO;return CASE;}

"DEFAULT" {yylval.string = strdup(yytext); ECHO;return DEFAULT;}

"ENDSWITCH" {yylval.string = strdup(yytext); ECHO;return ENDSWITCH;}

"PRINT" {yylval.string = strdup(yytext); ECHO;return PRINT;}

"BREAK" {yylval.string = strdup(yytext); ECHO;return BREAK;}

"AND" {yylval.string = strdup(yytext); ECHO;return AND;}

"OR" {yylval.string = strdup(yytext); ECHO;return OR;}

"(" {yylval.string = strdup(yytext); ECHO;return LEFTBRA; }

"[" {yylval.string = strdup(yytext); ECHO;return LEFTSTRBRA; }

">" {yylval.string = strdup(yytext); ECHO;return MEGALYTERO ; }

"<" {yylval.string = strdup(yytext); ECHO;return MIKROTERO ; }

"!" {yylval.string = strdup(yytext); ECHO;return THAYMASTIKO ; }

"=" {yylval.string = strdup(yytext); ECHO;return ISON ; }

"," {yylval.string = strdup(yytext); ECHO;return COMMA ; }

"]" {yylval.string = strdup(yytext); ECHO;return RIGHTSTRLBRA ;}

")" {yylval.string = strdup(yytext); ECHO;return RIGHTBRA ; }

":" {yylval.string = strdup(yytext); ECHO;return ANOKATOTEL ; }

";" {yylval.string = strdup(yytext); ECHO;return EROTIMATIKO ; }

"+" {yylval.string = strdup(yytext); ECHO;return SYN ; }

"-" {yylval.string = strdup(yytext); ECHO;return PLIN; }

"\*" {yylval.string = strdup(yytext); ECHO;return MUL ; }

"/" {yylval.string = strdup(yytext); ECHO;return DIV; }

"^" {yylval.string = strdup(yytext); ECHO;return POW; }

"%" {yylval.string = strdup(yytext); ECHO;return TISEKATO ; }

"/\*" {yylval.string = strdup(yytext); ECHO;return COMSTART; }

"\*/" {yylval.string = strdup(yytext); ECHO;return COMEND; }

[\n] {line\_count++ ;yylval.string = strdup(yytext); ECHO; return CHANGE\_LINE ; }

Εδώ δεν έχουμε return γιατί δεν χρειάζεται να μας επιστρέφει τίποτα για τα κενά

[ \t]\* {yylval.string = strdup(yytext); ECHO;}

Δήλωση αριθμών

{num} {yylval.string = strdup(yytext); ECHO; return INT; }

Δήλωση ονομάτων για μεταβλητές κ.α

[a-zA-Z\_][a-zA-Z0-9\_]\* { yylval.string = strdup(yytext); ECHO; return NAME; }

%%

**Bison:**

**Καθώς έχουν εξηγηθεί ήδη οι κανόνες της γραμματικής από το bnf. Εδώ θα αναλυθούν μόνο τα επιπλέον στοιχεία. Ωστόσο έχουμε παραθέσει όλο το bison για να υπάρχει μια συνέχεια.**

%{

Δήλωση βιβλιοθηκών που χρησιμοποιούνται

#include <stdio.h>

#include <string.h>

Δήλωση αρχείων για input και output

extern FILE \*yyin;

extern FILE \*yyout;

Δήλωση συναρτήσεων που χρειάζονται:

extern int yylex();

extern int yyparse();

extern char\* yytext();

extern int line\_count;

void yyerror(const char \*s);

Δήλωση μεταβλητής που θα χρησιμοποιηθεί για τα errors

int error=0;

%}

Δηλώσεις bison :

Δεσμευμένες λέξεις: (σε string για να μπορούν να εκτυπώνονται στο output )

%token<string> PROGRAM NAME CHANGE\_LINE STARTMAIN ENDMAIN FUNCTION LEFTBRA RIGHTBRA

%token<string> RETURN END\_FUNCTION VARS CHAR INTEGER EROTIMATIKO INT BREAK

%token<string> PRINT LEFTSTRBRA COMMA RIGHTSTRLBRA TISEKATO WHILE ENDWHILE

%token<string> FOR ANOKATOTEL ISON TO STEP ENDFOR MEGALYTERO MIKROTERO THAYMASTIKO AND OR

%token<string> IF THEN ELSEIF ELSE ENDIF SWITCH CASE DEFAULT ENDSWITCH SYN PLIN MUL DIV POW

%token<string> STRUCT TYPEDEF ENDSTRUCT TEXT COMSTART COMEND SPACE

%left SYN PLIN

%left MUL DIV

%right POW

Ορισμός που είναι η αρχή των κανόνων μας

%start start

Για υλοποίηση string

%union{

char\* string;

}

Δήλωση ως string τους τύπους

%type<string> start program body function functions main var type\_specifier var2

%type<string> commands command comment loop while for list condition symbol logic

%type<string> print conditions if switch assignment func break symbol2 lines multiple

%type<string> structs struct big\_comment

%%

Το start χρησιμοποιείται για να μπορούμε να εκτυπώσουμε όλο το πρόγραμμα με μία εντολή. Με την εντολή fprintf εκτυπώνουμε σε αρχείο

start: program {fprintf(yyout, $1);};

program: PROGRAM NAME lines body ;

lines: CHANGE\_LINE

| CHANGE\_LINE lines

;

body:main

|structs functions main

;

structs: %empty {}

|struct structs

;

Ο τύπος που έχει χρησιμοποιηθεί για τα structs είναι ο “ kiriaki “. Και με τις if ελέγχουμε αν η δήλωση είναι σωστή. Αν ο τύπος που δηλωθεί έχει άλλο όνομα θα έχουμε σφάλμα. Επίσης αν δημιουργηθούν δύο struct με το ίδιο όνομα θα εμφανιστεί πάλι σφάλμα.

struct: STRUCT NAME lines var ENDSTRUCT lines {

if (strcmp($2,"kiriaki") == 0 && count < 1) {count++;}

else if (strcmp($2,"kiriaki") != 0){ yyerror("Wrong type for the struct\n") ;}

else {yyerror("The struct already exists\n");};}

Ομοίως με παραπάνω η επεξήγηση του κώδικα, μόνο που εδώ έχουμε πιο πολλές περιπτώσεις στα if , αφού θέλουμε να έχουμε το ίδιο όνομα και στην αρχή και στο τέλος του struct. Βάλαμε ένα comment καθώς το όνομα την δεύτερη φορά αναγνωριζόταν ως μεταβλητή και είχαμε error.

| TYPEDEF STRUCT NAME lines var big\_comment NAME ENDSTRUCT lines {

if (strcmp($3,"kiriaki") == 0 && strcmp($7,"kiriaki") == 0 && count < 1) {count++;}

else if (strcmp($3,"kiriaki") != 0){ yyerror("Wrong type for the struct\n") ;}

else if (strcmp($3,"kiriaki") == 0 && strcmp($7,"kiriaki") != 0){ yyerror("The name at the end is not the same with the beginning \n") ;}

else {yyerror("The struct already exists\n");};}

;

functions: function

| function functions

;

function: FUNCTION NAME LEFTBRA list RIGHTBRA lines var commands

RETURN NAME lines END\_FUNCTION lines

;

list: NAME

| NAME COMMA list

| NAME LEFTSTRBRA INT RIGHTSTRLBRA

| NAME LEFTSTRBRA INT RIGHTSTRLBRA COMMA list;

var: %empty {}

| var2 var

;

var2: VARS lines multiple

;

multiple: %empty {}

|type\_specifier list EROTIMATIKO lines multiple

;

Με το κομμάτι κώδικα ελέγχουμε αν έχει δημιουργηθεί struct με τέτοιο όνομα. Αν δεν έχει δημιουργηθεί τότε δεν μπορούμε να δηλώσουμε μεταβλητή με τέτοιο τύπο. Οπότε βγάζει σφάλμα.

type\_specifier: CHAR

| INTEGER

| NAME {if (strcmp($1,"kiriaki")!=0){yyerror("This type does not exist");}

else {};}

;

commands: command

| command commands

;

command:comment

|loop

|conditions

|print

|assignment

|big\_comment

;

loop: while

| for

;

while: WHILE LEFTBRA condition RIGHTBRA lines break commands

break ENDWHILE lines

;

for: FOR NAME ANOKATOTEL ISON INT TO INT STEP INT lines

break commands break ENDFOR lines

;

conditions: if

| switch

;

if: IF LEFTBRA condition RIGHTBRA THEN lines commands elseif2 else ENDIF lines

;

elseif2: %empty {}

| ELSEIF lines commands elseif2

;

else: %empty

| ELSE lines commands

;

switch: SWITCH LEFTBRA condition RIGHTBRA lines case2 default ENDSWITCH lines

;

case2: %empty {}

| case case2

;

case: CASE LEFTBRA condition RIGHTBRA ANOKATOTEL lines break commands break

;

default: %empty {}

| DEFAULT ANOKATOTEL lines break commands break

;

print: PRINT LEFTBRA words RIGHTBRA EROTIMATIKO lines

|PRINT LEFTBRA words LEFTSTRBRA COMMA NAME RIGHTSTRLBRA RIGHTBRA EROTIMATIKO lines

;

condition: NAME symbol NAME

| condition logic condition;

symbol: MEGALYTERO

|MIKROTERO

|ISON ISON

|THAYMASTIKO

;

logic: AND

| OR

;

assignment: NAME ISON expression EROTIMATIKO lines

;

expression: INT

| func

| NAME

| expression symbol2 expression

| LEFTBRA expression RIGHTBRA

;

symbol2: SYN

|PLIN

|MUL

|DIV

|POW

;

func: NAME LEFTBRA list RIGHTBRA

;

break: %empty {}

| BREAK EROTIMATIKO lines

;

comment: TISEKATO NAME lines

;

big\_comment: COMSTART NAME COMEND lines

|COMSTART CHANGE\_LINE test COMEND lines

;

test: %empty {}

|NAME CHANGE\_LINE test

| lines test

|command test

;

main: STARTMAIN lines var commands ENDMAIN lines

;

words: NAME

| NAME words

;

%%

Συνάρτηση για τα errors. Αν βρεθεί error , σταματά η εκτέλεση του προγράμματος και εμφανίζεται στο αρχείο με την έξοδο το κατάλληλο μήνυμα με το error και την γραμμή στην οποία βρίσκεται.

void yyerror(const char \*s) {

printf("error %s\n",s);

fprintf(yyout,"\n%s at line %d",s,line\_count);

error++ ;

}

Συνάρτηση main. Διαβάζεται το αρχείο εισόδου και τα αποτελέσματα γράφονται στο αρχείο εξόδου.

int main ( int argc, char \*\*argv )

{

++argv; --argc;

if ( argc > 0 )

yyin = fopen( argv[0], "r" );

else

yyin = stdin;

yyout = fopen ( "output", "w" );

yyparse ();

printf("errors: %d",error);

return 0;}

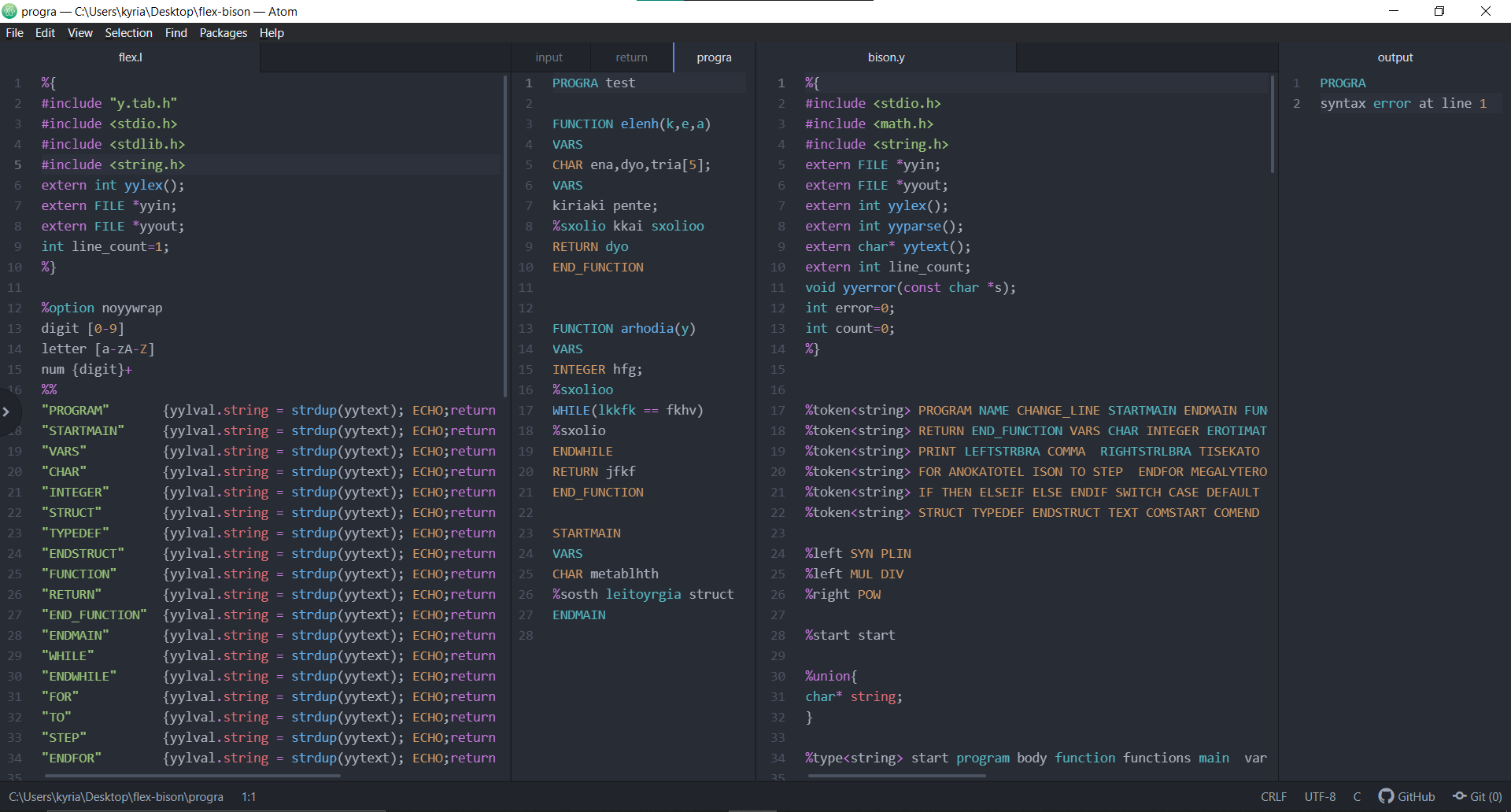
**Screenshots:**

Δεν μπορούμε να βάλουμε όλα τα πιθανά errors , οπότε θα βάλουμε 2-3 παραδείγματα από κάθε ερώτημα.

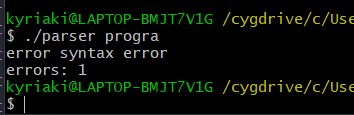
Σε κάθε screenshot βλέπουμε 4 στήλες. Αριστερά είναι το flex, μετά το input, μετά το bison και δεξιά το output.

**Παραδείγματα από Ερώτημα 1 :**

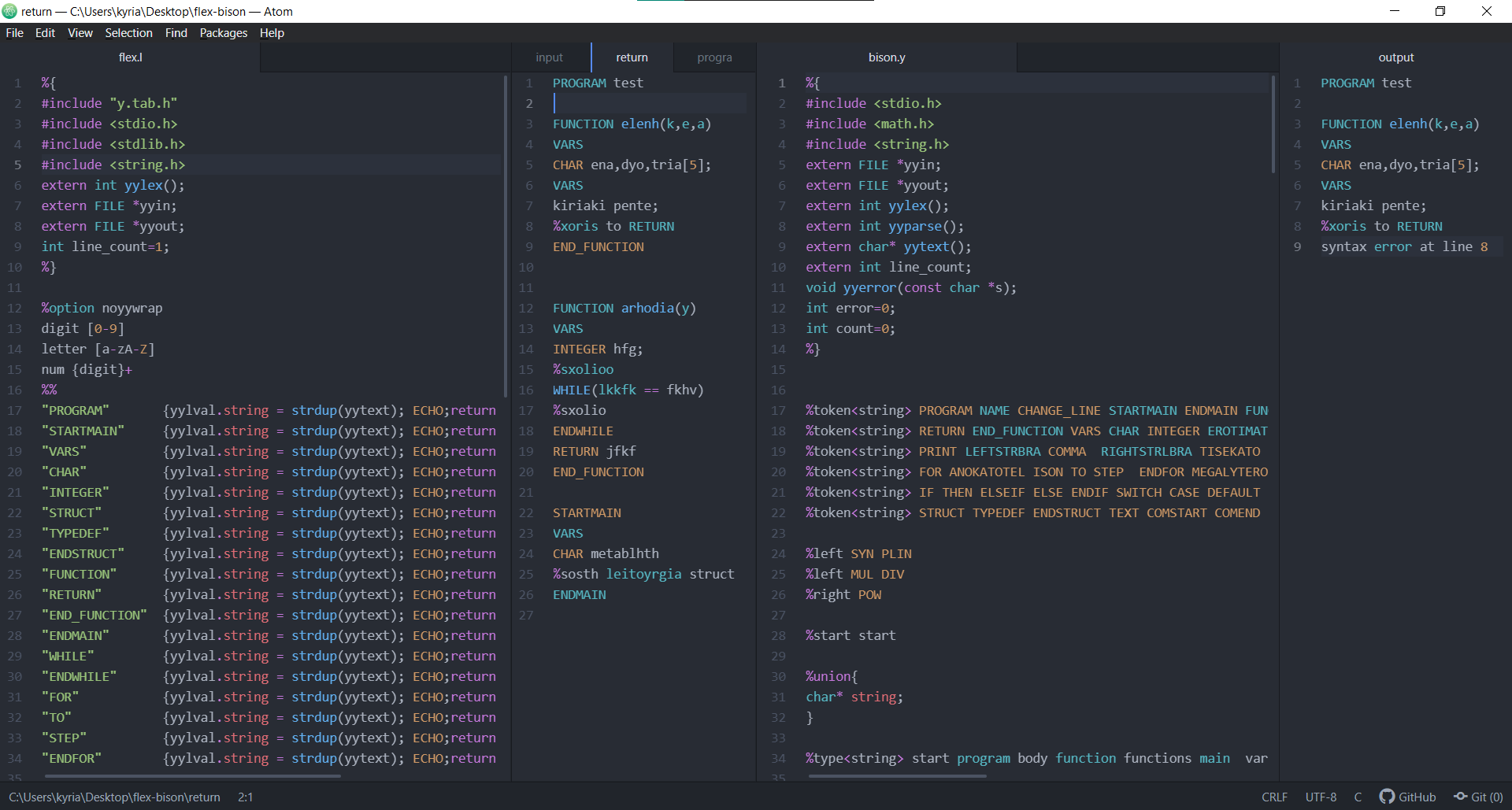
1. Έχοντας γράψει λάθος τη δεσμευμένη λέξη PROGRAM, βλέπουμε ότι στο output δεν εκτυπώνεται όλο το input καθώς προκύπτει syntax error στην γραμμή 1.



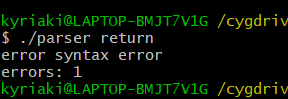
Βλέπουμε και το αποτέλεσμα στο Cygwin. Χωρίς να εκτυπώνεται εκεί το πρόγραμμα, μόνο τα errors. Το πρόγραμμα εκτυπώνεται μόνο στο output.



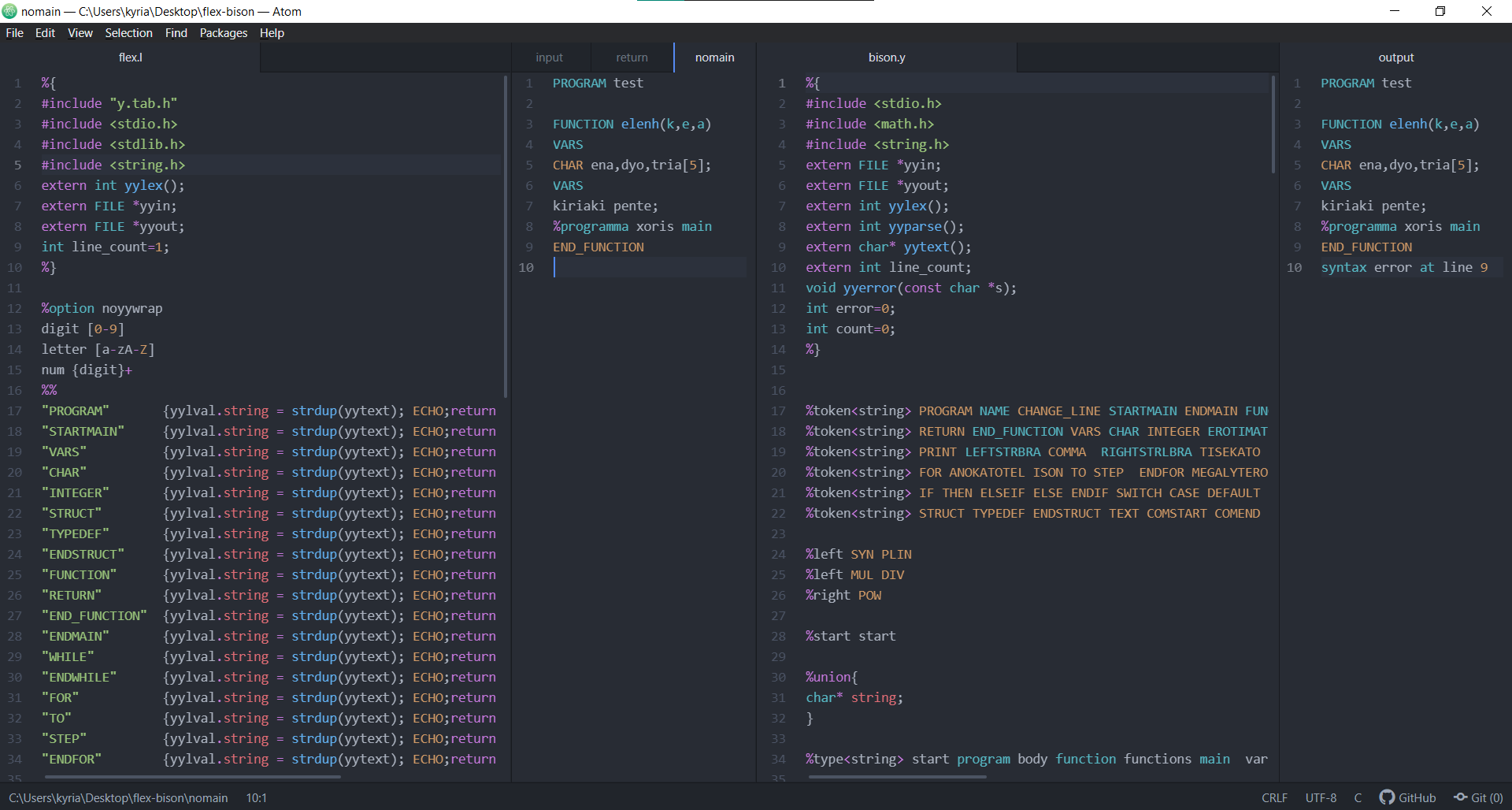
1. Στην είσοδο αυτή έχουμε παραλείψει από την μια συνάρτηση το RETURN. Έτσι βλέπουμε στο output δεξιά ότι υπάρχει syntax error στην γραμμή 8, δηλαδή στην γραμμή που θα έπρεπε να βρίσκεται το RETURN.



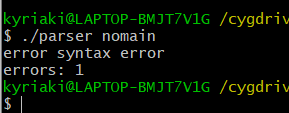
Εκτέλεση στο Cygwin:



1. Στο Input αυτό δεν έχουμε βάλει καθόλου main. Επειδή όμως η main πρέπει να υπάρχει υποχρεωτικά έχουμε error στην γραμμή 9, δηλαδή στην γραμμή που αναμενόταν να υπάρχει η main.

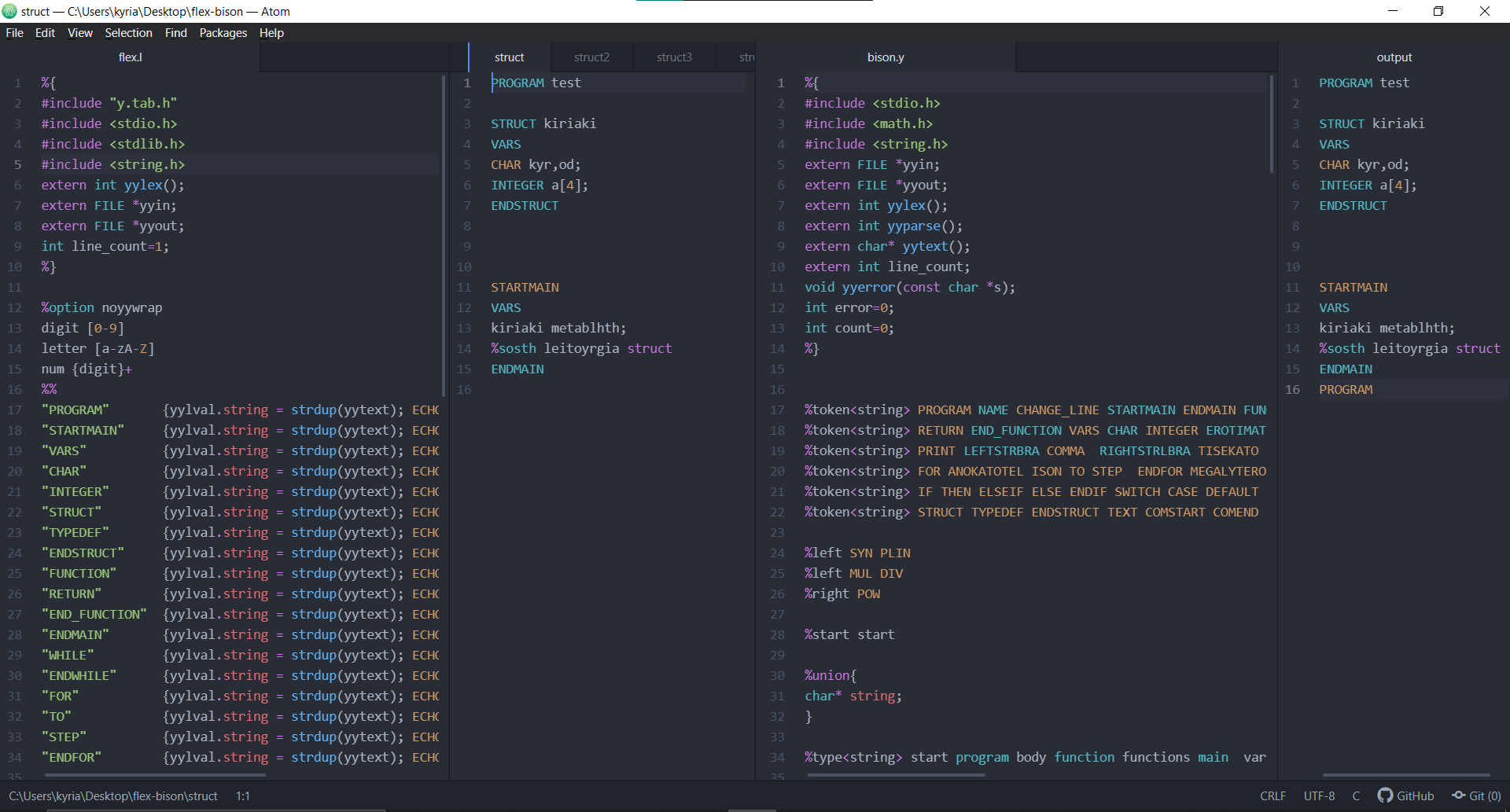


Εκτέλεση στο Cygwin:

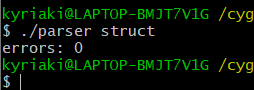


**Παραδείγματα από Ερώτημα 2 :**

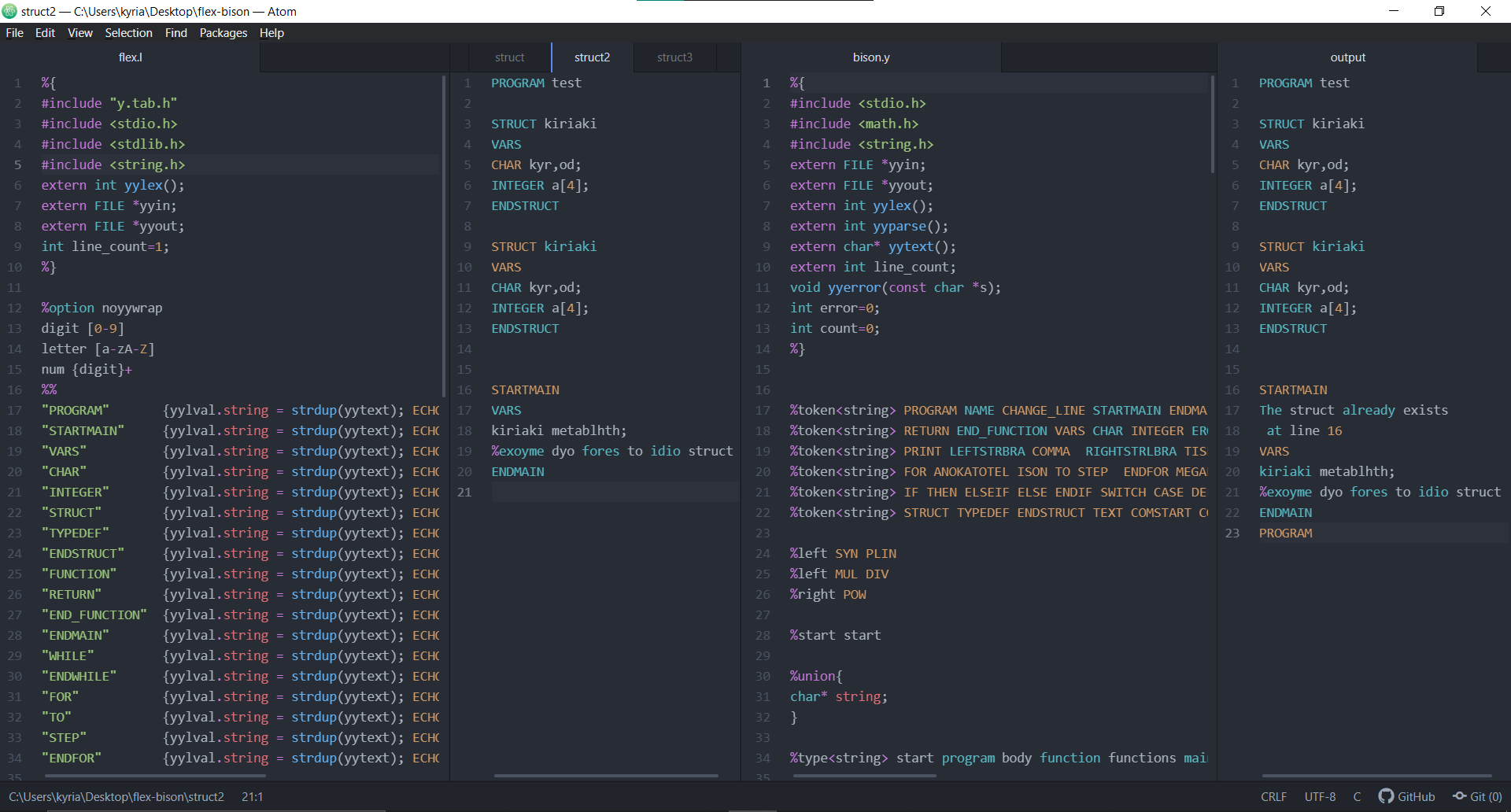
1. Εδώ βλέπουμε την σωστή λειτουργία ενός struct και έτσι στο Output εκτυπώνεται όλη η είσοδος, χωρίς error.



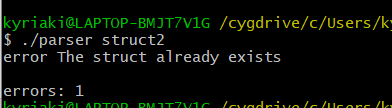
Εκτέλεση στο Cygwin:



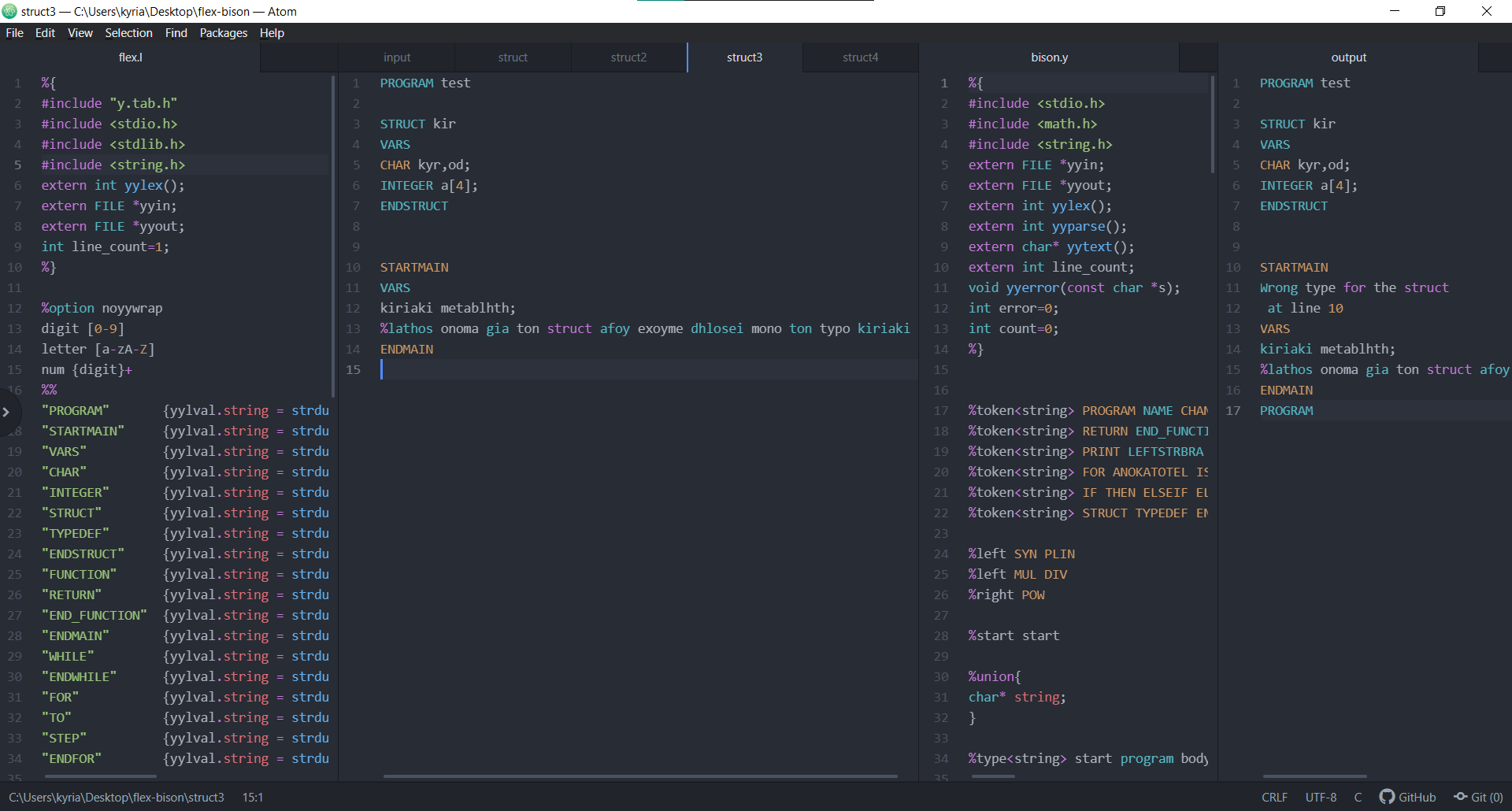
1. Έχοντας βάλει δύο φορές το ίδιο struct, βλέπουμε ότι εμφανίζεται το κατάλληλο error.



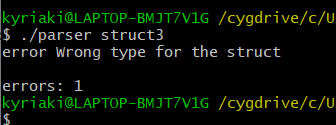
Εκτέλεση στο Cygwin:



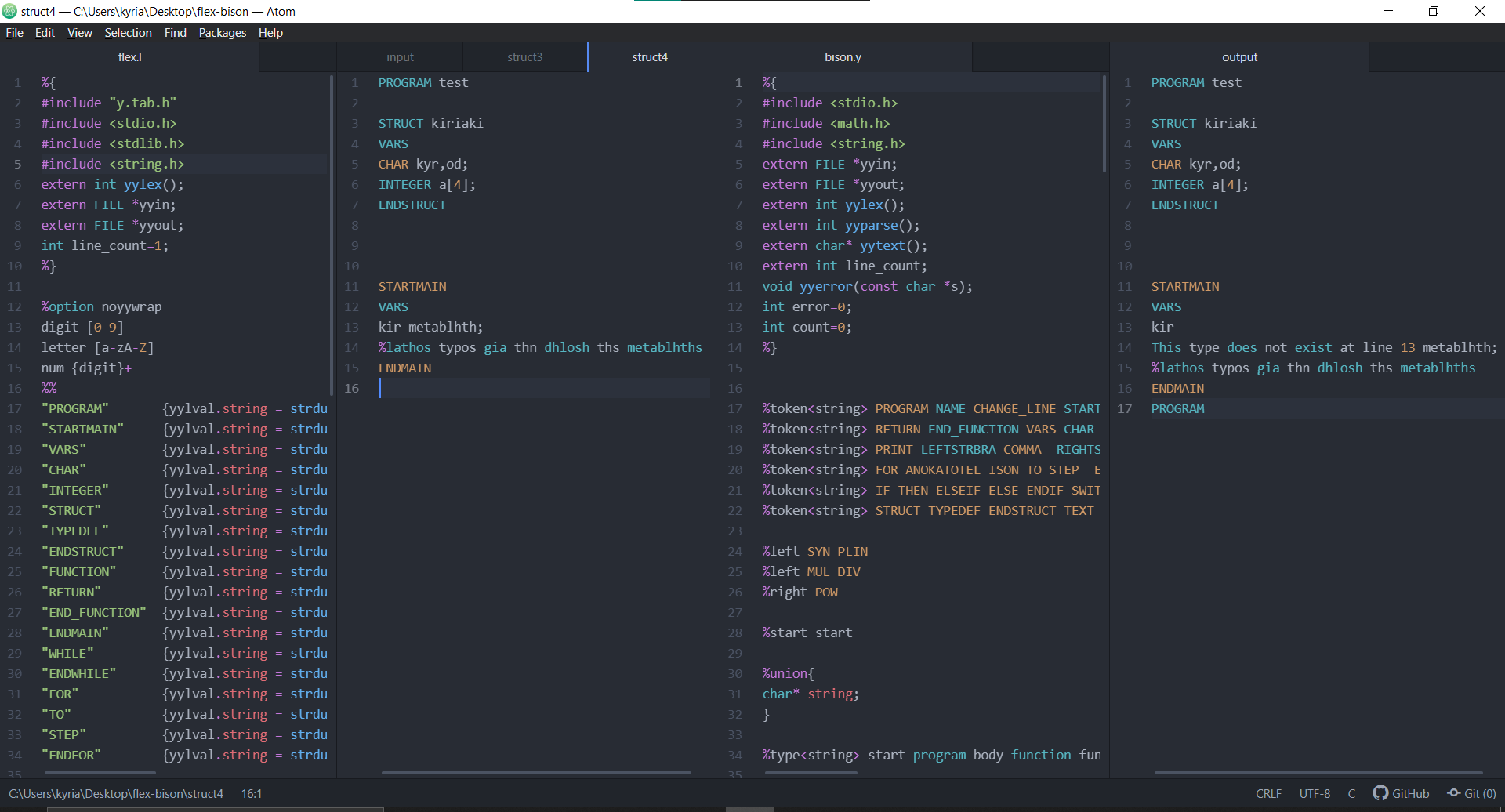
1. Έχουμε βάλει λάθος τύπο στο struct, αφού έχουμε δηλώσει να γίνεται αποδεκτός μόνο ο τύπος “kiriaki”. Έτσι εμφανίζεται το κατάλληλο error στο output.



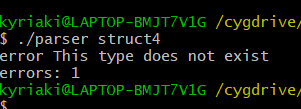
Εκτέλεση στο Cygwin:



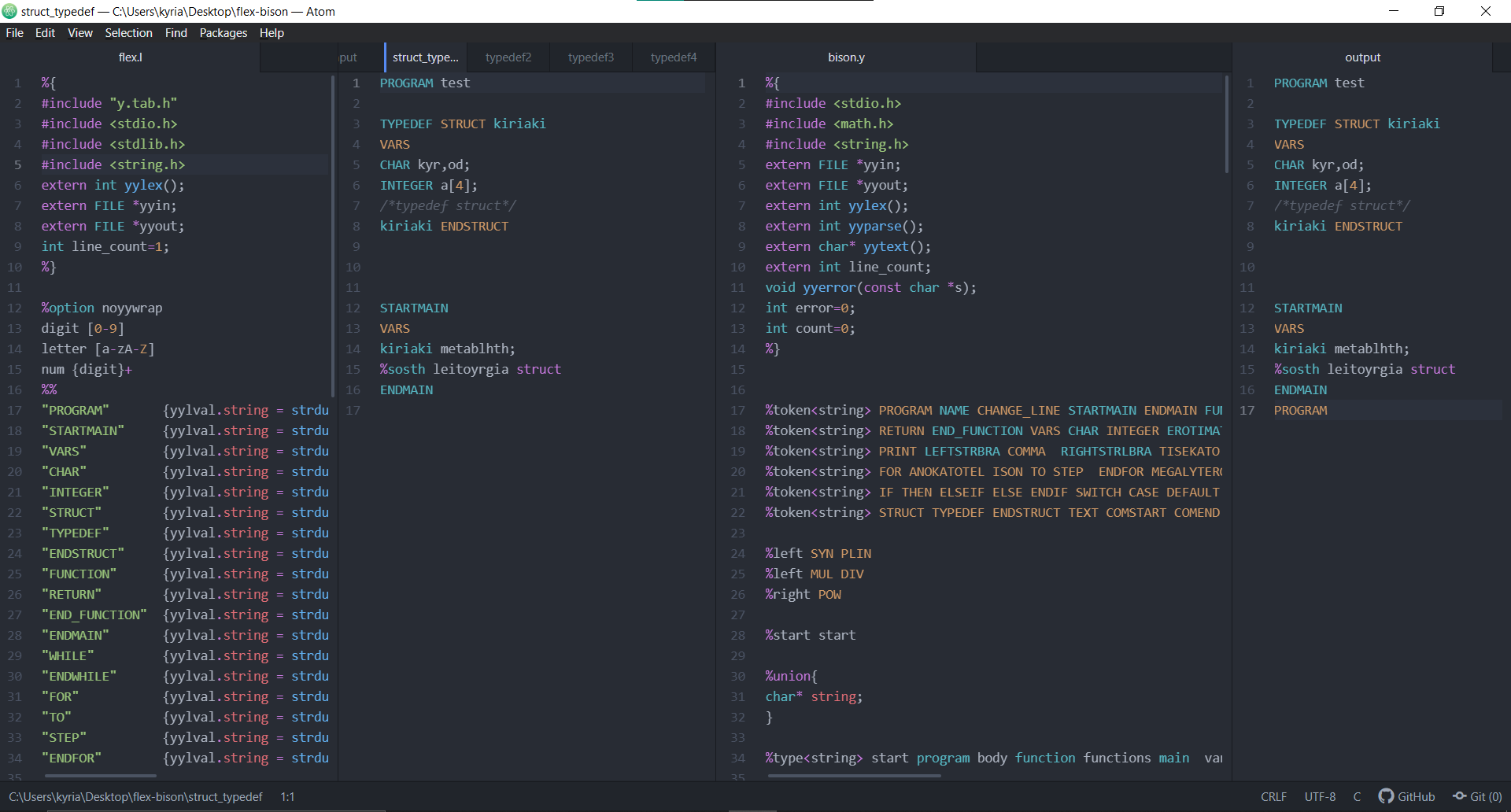
1. Δηλώνουμε μια μεταβλητή τύπου “kir”. Ενώ ο τύπος του struct είναι ο “kiriaki”. Βλέπουμε το κατάλληλο μήνυμα.



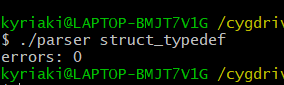
Εκτέλεση στο Cygwin:



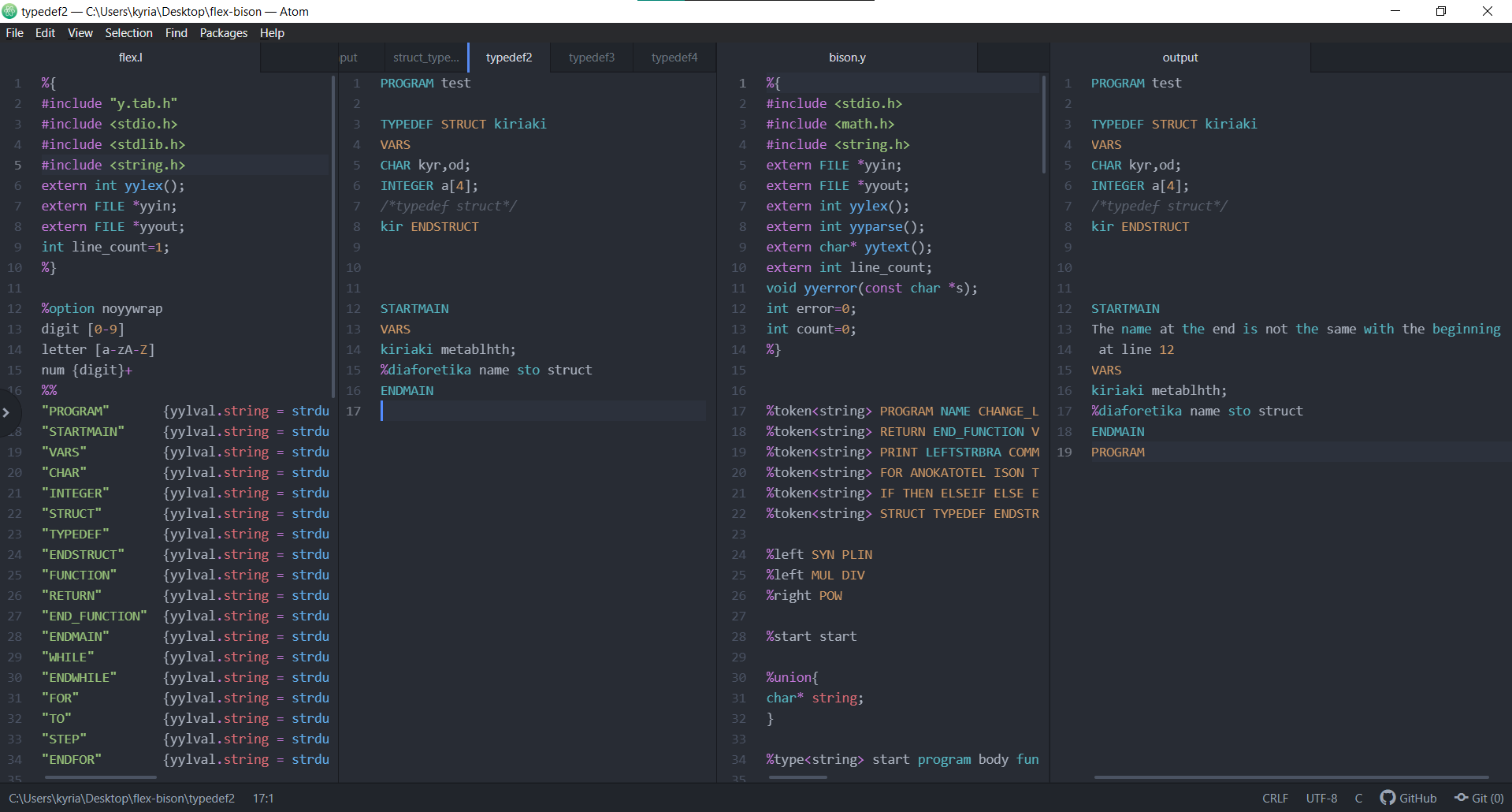
1. Σωστή λειτουργία struct με typedef



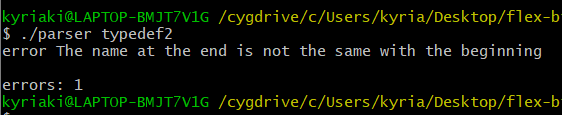
Εκτέλεση στο Cygwin:



1. Έχουμε βάλει διαφορετικό όνομα στην αρχή και το τέλος του struct. Εμφανίζεται το κατάλληλο error.



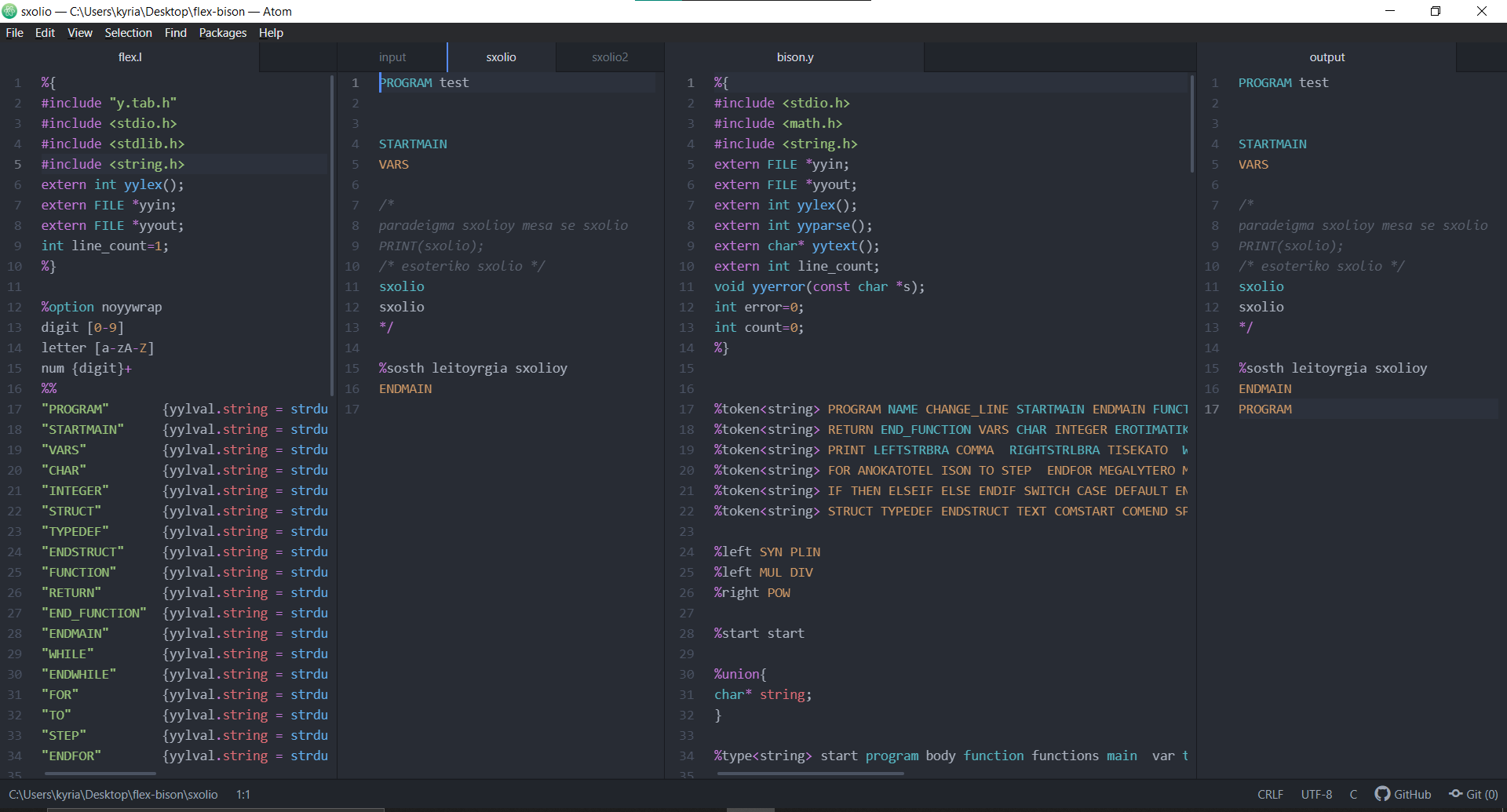
Εκτέλεση στο Cygwin:



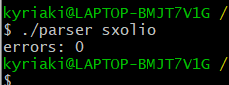
**Οι υπόλοιπες περιπτώσεις με τα errors στα typedef είναι ίδιες με των απλών structs παραπάνω.**

**Παραδείγματα από Ερώτημα 4 :**

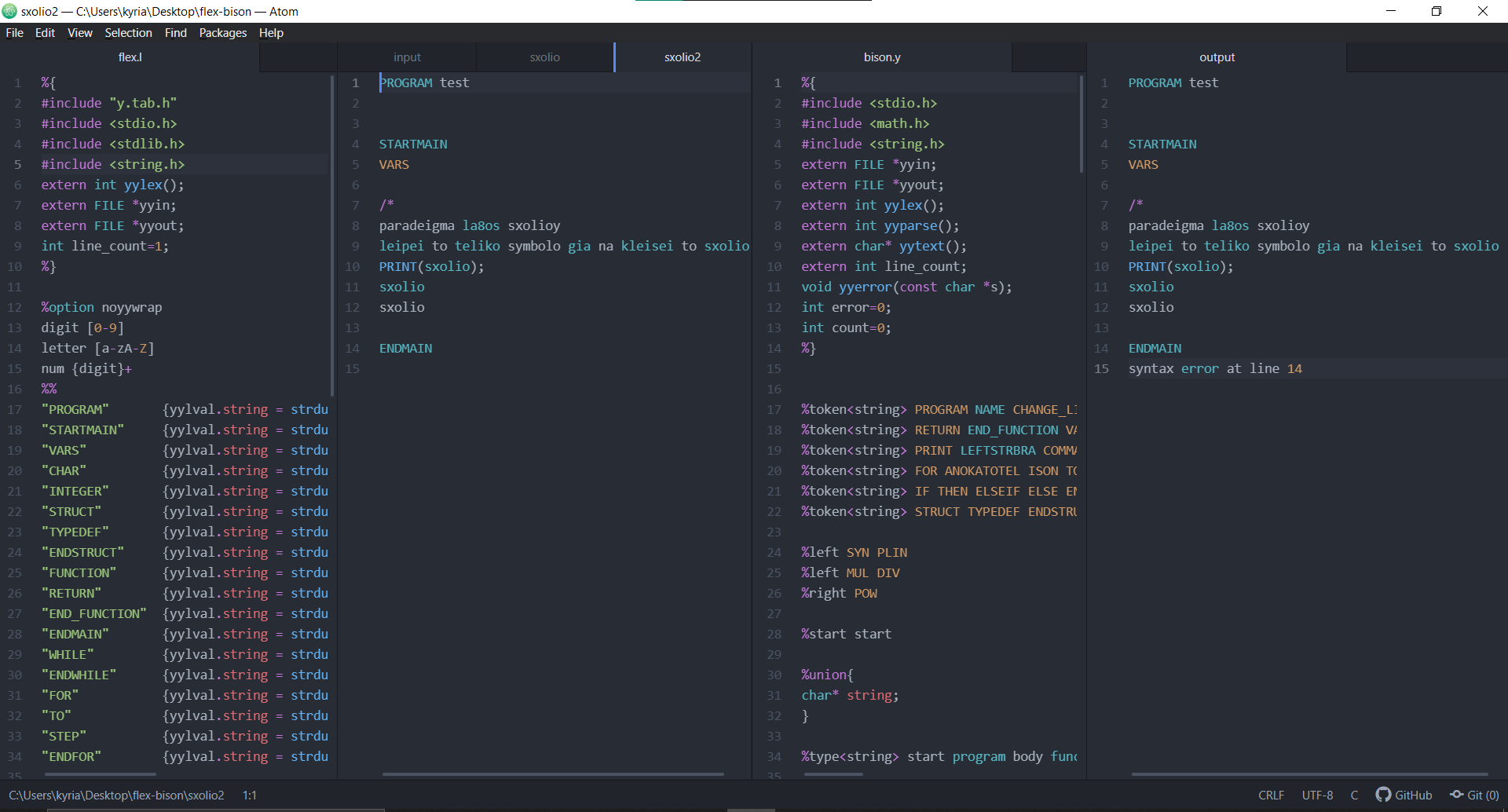
1. Σωστή λειτουργία του σχολίου της μορφής /\* σχόλιο \*/



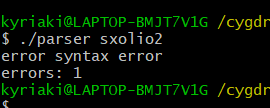
Εκτέλεση στο Cygwin :



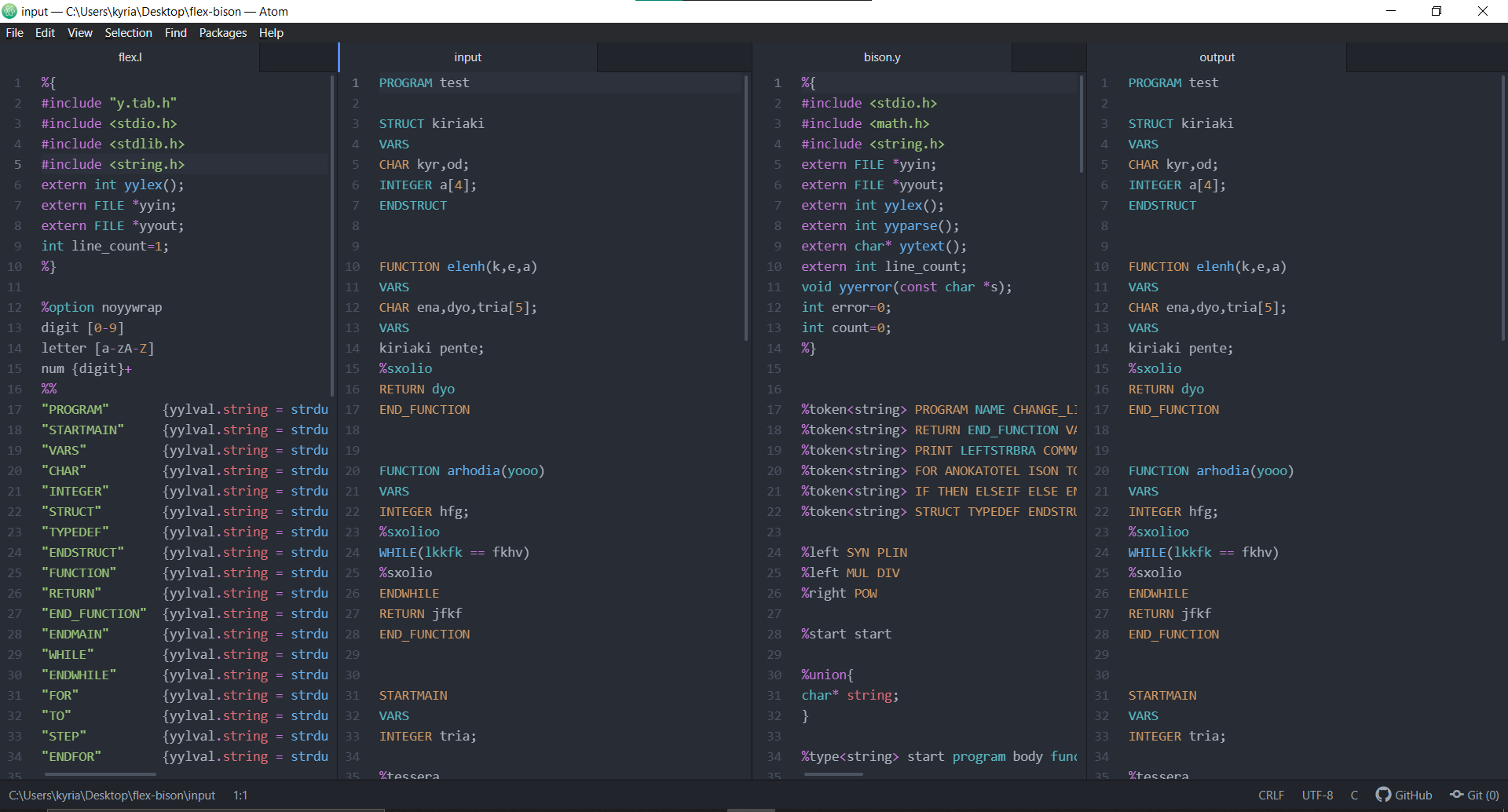
1. Έχουμε αφαιρέσει το σύμβολο \*/ που κλείνει το σχόλιο και βλέπουμε το error στη γραμμή 14, καθώς κάπου εκεί περιμένει να βρει το σύμβολο.



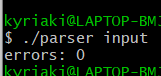
Εκτέλεση στο Cygwin:



* Έχουμε φτιάξει ένα input με όλες τις εντολές που ζητούνται, το οποίο λειτουργεί σωστά χωρίς errors.



Εκτέλεση στο Cygwin:



Το input δεν χωράει ολόκληρο στο screenshot.Το στέλνουμε λοιπόν μαζί με τα υπόλοιπα αρχεία.

Προσπάθεια ερωτήματος 3 για τον έλεγχο σωστής δήλωσης των ονομάτων που χρησιμοποιούνται σε συναρτήσεις.

function: FUNCTION NAME LEFTBRA list RIGHTBRA lines var commands{char\* x; int i; int found;

for(i = 0; i < 4; i++){if(strcmp($2, x) == 0 ){ found=1;}else{};}

if (found=0) {$2 = x;}

else if (found>0) {yyerror("The function already exists");}

;}

Επεξήγηση:

Ορίζουμε έναν πίνακα x από string. Τρέχουμε μια for για τα στοιχεία του πίνακα. Με μια if ελέγχουμε αν το όνομα που δίνουμε υπάρχει ήδη στον πίνακα με την συνάρτηση strcmp() και αν υπάρχει τότε βάζουμε την τιμή 1 στην μεταβλητή found, την οποία χρησιμοποιούμε από κάτω. Στο else δεν γίνεται τίποτα. Με μια if ελέγχουμε αν η μεταβλητή found έχει την τιμή 0. Αν έχει την τιμή 0 σημαίνει ότι δεν βρέθηκε στον πίνακα τέτοιο όνομα, οπότε το εισάγουμε εμείς στον πίνακα. Αλλιώς αν το found >0 , δηλαδή βρέθηκε στον πίνακα τότε καλούμε την yyerror() και βγάζει το κατάλληλο μήνυμα ότι υπάρχει ήδη συνάρτηση με τέτοιο όνομα.

Ομοίως θα γινόταν και ο έλεγχος στον κανόνα με την δήλωση των μεταβλητών στο πρόγραμμα. Θα φτιάχνεται ένας πίνακας που θα αποθηκεύει τα ονόματα όλων των μεταβλητών του προγράμματος και με συγκρίσεις θα καταλήγουμε στο αν έχουν ξαναδηλωθεί οι μεταβλητές και θα παίρνουμε τα κατάλληλα errors.

Με την υλοποίηση αυτή έχουμε segmentation fault και λόγω έλλειψης χρόνου δεν καταφέραμε να το διορθώσουμε. Ωστόσο το παραθέτουμε σαν ιδέα.

**Σχόλια και Παραδοχές:**

* Το τρίτο ερώτημα δεν τρέχει και δεν είναι ολοκληρωμένο, αλλά η προσπάθεια υλοποίησης φαίνεται από πάνω.
* Στο δεύτερο ερώτημα αναγνωρίζεται μόνο ένας τύπος, για ευκολία.
* Η έξοδος εκτυπώνεται σε ξεχωριστό αρχείο με όνομα output.
* Δημιουργήθηκαν πολλές είσοδοι για τα screenshot δείχνοντας error και μη. Τα στέλνουμε όλα σε περίπτωση που θέλετε να τα χρησιμοποιήσετε. Το βασικό input βρίσκεται μόνο του, ενώ όλα τα υπόλοιπα input για τα παραδείγματα βρίσκονται όλα μαζί σε έναν ξεχωριστό φάκελο μέσα στο zip.