**Αρχές Γλωσσών Προγραμματισμού & Μεταφραστών**

**Project Flex – Bison**

|  |  |  |  |
| --- | --- | --- | --- |
| **Ονοματεπώνυμο** | **Έτος** | **ΑΜ** | **E-mail** |
| Δρίτσας Ευστάθιος-Ραφαήλ | 2ο / Β’ | 1112100 | up1112100@ac.upatras.gr |
| Μελιτσόπουλος Κωνσταντίνος | 3ο / Γ' | 1100621 | up1100621@ac.upatras.gr |
| Μελιτσόπουλος Φώτιος | 5ο / Ε’ | 1084600 | up1084600@ac.upatras.gr |

**Περιγραφή γραμματικής σε BNF**

<myhtml> ::= 'MYHTML' <head> <body> 'MYHTML\_'

<head> ::= ε

| 'HEAD' <title> <meta> 'HEAD\_'

<title> ::= 'TITLE' <text> 'TITLE\_'

<meta> ::= ε

| <meta> 'META' <name> <content> 'CT'

| <meta> 'META' <charset> 'CT'

<body> ::= 'BODY' <body\_cont> 'BODY\_'

<body\_cont> ::= ε

| <body\_cont> <div\_cont>

| <body\_cont> <div>

<div> ::= 'DIV' <id> <style> 'CT' <div\_cont> 'DIV\_'

<div\_cont> ::= <form>

| <img>

| <a>

| <p>

<p> ::= 'P' <id> <style> 'CT' <text> 'P\_'

<a> ::= 'A' <id> <href> 'CT' <a\_cont> 'A\_'

<a\_cont> ::= <text>

| <img>

<img> ::= 'IMG' <id> <src> <alt> <width> <height> 'CT'

<form> ::= 'FORM' <id> <style> 'CT' <form\_cont> 'FORM\_'

<form\_cont> ::= <form\_cont> <label>

| <form\_cont> <input>

| <label>

| <input>

<input> ::= 'INPUT' <id> <type> <style> <value> 'CT'

<label> ::= 'LABEL' <id> <for> <style> 'CT' <text> 'LABEL\_'

<id> ::= 'ID' 'ASSIGN' 'ALPHA'

<charset> ::= 'CHARSET' 'ASSIGN' 'ALPHA'

<name> ::= 'NAME' 'ASSIGN' 'ALPHA'

<content> ::= 'CONTENT' 'ASSIGN' 'ALPHA'

<style> ::= ε

| 'STYLE' 'ASSIGN' 'ALPHA'

<href> ::= 'HREF' 'ASSIGN' 'ALPHA'

<src> ::= 'SRC' 'ASSIGN' 'ALPHA'

<alt> ::= 'ALT' 'ASSIGN' 'ALPHA'

<width> ::= ε

| 'WIDTH' 'ASSIGN' 'ALPHA'

<height> ::= ε

| 'HEIGHT' 'ASSIGN' 'ALPHA'

<value> ::= ε

| 'VALUE' 'ASSIGN' 'ALPHA'

<type> ::= 'TYPE' 'ASSIGN' 'ALPHA'

<for> ::= 'FOR' 'ASSIGN' 'ALPHA'

<text> ::= ε

| 'TEXT'

*\* εφόσον το bison παράγει LR(1) parsers, στην γραμματική μας θα εμπεριέχεται αριστερή αναδρομή και όχι δεξιά.*

**Παραδοχές Υλοποίησης**

Αρχικά, όπως είναι και εμφανές από τα παραδωθέντα, η υλοποίηση μας φτάνει μέχρι το ερώτημα (1). Δυστυχώς, λόγω εσφαλμένου χρονοπρογραμματισμού, δεν μπορέσαμε να ολοκληρώσουμε κάποιο από τα υπόλοιπα (υπο)ερωτήματα.

Δεύτερον, ενώ στην εκφώνηση ζητείται ρητά η υποστήριξη τοποθέτησης των χαρακτηριστικών της εκάστοτε ετικέτας σε οποιαδήποτε σειρά (ασχέτως αν αυτά είναι υποχρεωτικά ή όχι), στην πράξη η λύση, στην οποία καταλήξαμε δεν ήταν “κομψή” αλλά αναποδοτική, αναλογιζόμενοι την ταχύτητα με την οποία οφείλει να ανταποκρίνεται ο αναλυτής. Διαφορετικά, η επίλυση εκφυλλιζόταν σε έναν απλό κανόνα BNF, ο οποίος όμως επέτρεπε πολλαπλές εμφανίσεις του ίδιου χαρακτηριστικού, που επίσης έπρεπε να διαχειριστούμε με περίσσεια κώδικα C. Συνεπώς, αποφασίσαμε να ορίσουμε εμείς μια αυστήρη σειρά προτεραιότητητας, η οποία είναι η εξής:

**charset > name > content > id > href > src > alt > type > for > style > width > height > value \***

*\* η σειρά προτεραιότητας εφαρμόζεται για όλες τις ετικέτες, ωστόσο δεν αντιστοιχούν όλα τα χαρακτηριστικά σε όλες τις ετικέτες. Η κάθε ετικέτα απαιτεί τα δικά της χαρακτηριστικά στη σειρά που υποδεικνύεται παραπάνω, και δεν δέχεται χαρακτηριστικά που δεν της ανήκουν, ασχέτως σε ποια θέση βρίσκονται.*

**Σχόλια**

Αρχικά, κάναμε χρήση Makefile για διευκόλυνσή μας κατά το debugging. Αυτό επίσης μας επέτρεψε να δημιουργήσουμε τον κανόνα demo εντός του Makefile, ο οποίος απλά τρέχει τα 4 ενδεικτικά αρχεία .myhtml για διαπίστωση της λειτουργικότητας του αναλυτή μας.

Επιπλέον, αρκετή ώρα ξοδεύτηκε στην διάγνωση λανθασμένων εκτυπώσεων, οι οποίες οφείλονταν στον χαρακτήρα \r (carriage return), καθώς δουλεύαμε σε vs code εντός των Windows, το οποίο όμως επικοινωνεί με wsl2 linux, όπου και τρέχαμε την υλοποίησή μας. Τα δύο λειτουργικά συστήματα διαχειρίζονται διαφορετικά τον χαρακτήρα αυτό, κάτι που μας πήρε αρκετή ώρα μέχρι να εντοπίσουμε.

Εντός της main, χρησιμοποιούμε την εντολή ***setvbuf (stdout, NULL, \_IONBF, 0)*** για να απενεργοποιήσουμε το buffer την συνάρτησης printf (), ο οποίος προκαλούσε περίεργη και κατά κύριο λόγο, ακανόνιστη συμπεριφορά ως προς την εκτύπωση των αποτελεσμάτων, γεγονός που δεν έκανε καθόλου εύκολη την διάγνωση σφαλμάτων κατά την εκτέλεση.

**Παραδείγματα**

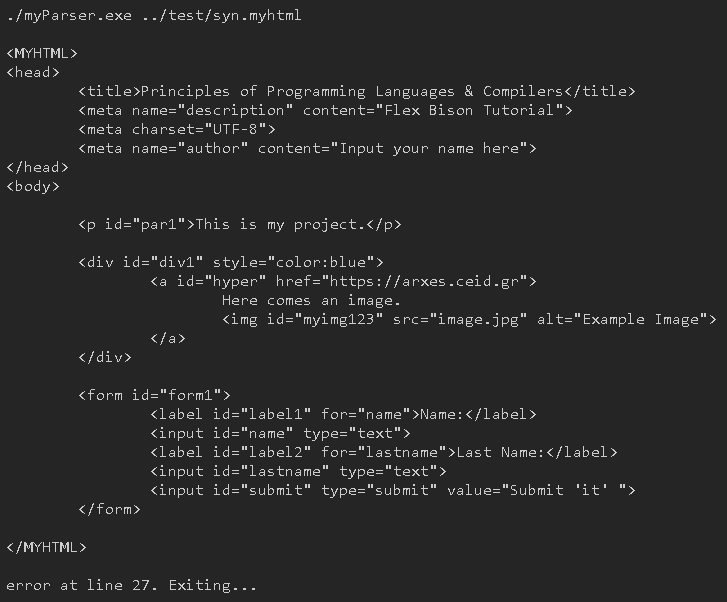
εκτέλεση του αναλυτή myParser.exe με είσοδο ok.myhtml (σωστά μορφοποιημένο αρχείο):



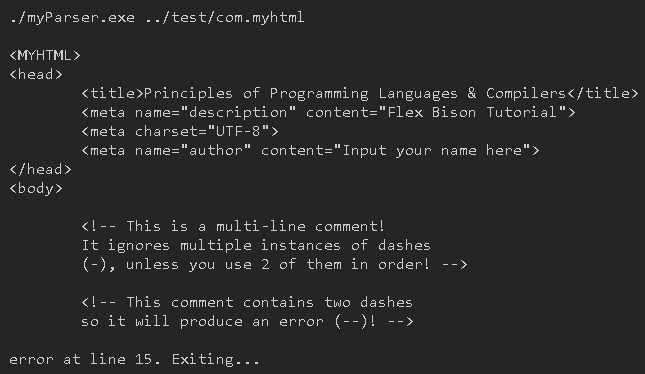
εκτέλεση του αναλυτή myParser.exe με είσοδο lex.myhtml (αρχείο που αποτυγχάνει λόγω της λανθασμένης ετικέτας “</MYHTML”):



εκτέλεση του αναλυτή myParser.exe με είσοδο syn.myhtml (αρχείο που αποτυγχάνει λόγω λανθασμένης σειράς στις ετικέτες “</body>” & “</MYHTML>”):



εκτέλεση του αναλυτή myParser.exe με είσοδο com.myhtml (αρχείο που αποτυγχάνει λόγω κακής μορφοποίησης σχολίων):



**Αρχεία Κώδικα**

**lexer.l:**

%{

#include "parser.tab.h"

#include <stdio.h>

#include <string.h>

void echo (void);

%}

/\* stop executing when you meet EOF \*/

%option noyywrap

/\* automatically update the yylineno variable with the current number of lines \*/

%option yylineno

/\* abstraction for easier use in ALPHA \*/

C [A-Za-z0-9\_ &.!:\-//]

/\* Alplhanumeric handler for nested double or single quotes \*/

ALPHA (\"{C}\*('({C}+)')?{C}\*\")|('{C}\*(\"({C}+)\"){C}\*')

/\* Text handler that doesn't recognize specific characters so that it may match only with text \*/

TEXT [{C}{-}[\n\t<>/=]]+

%%

[ \t\r\n] { echo (); }

"\<!--"[^-]([^-]|-[^-])\*"--\>" { echo (); }

"\<MYHTML\>" { echo (); return MYHTML; }

"\</MYHTML\>" { echo (); return MYHTML\_; }

"\<head\>" { echo (); return HEAD; }

"\</head\>" { echo (); return HEAD\_; }

"\<title\>" { echo (); return TITLE; }

"\</title\>" { echo (); return TITLE\_; }

"\<meta" { echo (); return META; }

"\<body\>" { echo (); return BODY; }

"\</body\>" { echo (); return BODY\_; }

"\<p" { echo (); return P; }

"\</p>" { echo (); return P\_; }

"\<a" { echo (); return A; }

"</a\>" { echo (); return A\_; }

"\<img" { echo (); return IMG; }

"\<form" { echo (); return FORM; }

"\</form\>" { echo (); return FORM\_; }

"\<div" { echo (); return DIV; }

"\</div\>" { echo (); return DIV\_; }

"charset" { echo (); return CHARSET; }

"name" { echo (); return NAME; }

"content" { echo (); return CONTENT; }

"id" { echo (); return ID; }

"style" { echo (); return STYLE; }

"href" { echo (); return HREF; }

"src" { echo (); return SRC; }

"alt" { echo (); return ALT; }

"width" { echo (); return WIDTH; }

"height" { echo (); return HEIGHT; }

"\<input" { echo (); return INPUT; }

"\<label" { echo (); return LABEL; }

"\</label\>" { echo (); return LABEL\_; }

"type" { echo (); return TYPE; }

"value" { echo (); return VALUE; }

"for" { echo (); return FOR; }

"=" { echo (); return ASSIGN; }

"\>" { echo (); return CT; }

{TEXT} { echo (); return TEXT; }

{ALPHA} { echo (); return ALPHA; }

%%

void echo (void) { printf ("%s", yytext); }

**parser.y**

%{

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

extern int yylex (void);

extern int yylineno;

extern FILE \* yyin;

void yyerror (const char \* msg);

%}

%token MYHTML MYHTML\_

%token HEAD BODY HEAD\_ BODY\_

%token TITLE P A FORM LABEL DIV TITLE\_ P\_ A\_ FORM\_ LABEL\_ DIV\_

%token META IMG INPUT CT ASSIGN

%token CHARSET NAME CONTENT ID STYLE HREF SRC ALT WIDTH HEIGHT TYPE VALUE FOR

%token ALPHA TEXT

%%

myhtml : MYHTML head body MYHTML\_;

head : %empty

| HEAD title meta HEAD\_;

title : TITLE text TITLE\_;

meta : %empty

| meta META name content CT

| meta META charset CT;

body : BODY body\_cont BODY\_;

body\_cont : %empty

| body\_cont div\_cont

| body\_cont div;

div : DIV id style CT div\_cont DIV\_;

div\_cont : form

| img

| a

| p;

p : P id style CT text P\_;

a : A id href CT a\_cont A\_;

a\_cont : text

| img;

img : IMG id src alt width height CT;

form : FORM id style CT form\_cont FORM\_;

form\_cont : form\_cont label

| form\_cont input

| label

| input;

input : INPUT id type style value CT;

label : LABEL id for style CT text LABEL\_;

id : ID ASSIGN ALPHA;

charset : CHARSET ASSIGN ALPHA;

name : NAME ASSIGN ALPHA;

content : CONTENT ASSIGN ALPHA;

style : %empty

| STYLE ASSIGN ALPHA;

href : HREF ASSIGN ALPHA;

src : SRC ASSIGN ALPHA;

alt : ALT ASSIGN ALPHA;

width : %empty

| WIDTH ASSIGN ALPHA;

height : %empty

| HEIGHT ASSIGN ALPHA;

value : %empty

| VALUE ASSIGN ALPHA;

type : TYPE ASSIGN ALPHA;

for : FOR ASSIGN ALPHA;

text : %empty

| TEXT;

%%

void yyerror (const char \* msg) {

// exiting stops at the location of error instead

// of continuing to the end of file

fprintf (stderr, "\n\nerror at line %d. Exiting...\n\n", yylineno);

exit (0);

}

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

// used to deactivate buffer in console,

// or the output doesn't make sense

setvbuf (stdout, NULL, \_IONBF, 0);

// guard clause for launch with no args

if (--argc < 1) return 0;

// open file stream

yyin = fopen (\* ++argv, "r");

// handle failure to open file stream

if (!yyin) {

perror ("file not found.\n\n");

return 0;

}

printf ("\n");

yyparse ();

printf ("\n\nfile correctly formatted.\n\n");

return 0;

}