Βελτιώσεις και εξηγήσεις πάνω στον κώδικα Τμήμα Β3

Διονύσης Νικολόπουλος

6 Ιουνίου 2021

Περιεχόμενα

Ομάδα 15

Αναλυτικά τα μέλη:

Διονύσης Νικολόπουλος AM : 18390126

Θανάσης Αναγνωστόπουλος AM : 18390043

Αριστείδης Αναγνωστόπουλος AM : 16124

Σπυρίδων Φλώρος AM : 141084

Αναλυτικά οι ρόλοι:

Γενικός Συντονιστής: Διονύσης Νικολόπουλος

Υπεύθυνος Τμήματος Εργασίας Β3: Διονύσης Νικολόπουλος

Usernames στο Github

Διονύσης Νικολόπουλος : dnnis

Θανάσης Αναγνωστόπουλος : ThanasisAnagno

Αριστείδης Αναγνωστόπουλος : Aris-Anag

Σπυρίδων Φλώρος : spirosfl

Η εργασία αυτή πραγματοποιήθηκε με χρήση LaTeX

1 Εισαγωγή

Σε αυτό το έγγραφο θα σας ενημερώσω για τις αλλαγές πάνω στον κώδικά μας μέχρι το μέρος B3.
Θα εξηγηθεί ο κώδικας τμηματικά, θα δωθούν πρόσθετες εξηγήσεις, μαζί με αυτές που δίνονται απο τα σχόλια στον κώδικα τον ίδιο.

1.1 Σημειώσεις για τους φακέλους

- **bison-part**: Σε αυτόν τον κατάλογο βρίσκονται τα αρχεία πηγαίου κώδικα του bison.
- **flex-part**: Σε αυτόν τον κατάλογο βρίσκονται τα αρχεία πηγαίου κώδικα του flex.
- **compile-room**: Σε αυτόν τον κατάλογο βρίσκεται το makefile, το οποίο αντιγράφει τα απαραίτητα αρχεία από τους προαναφερόμενους δύο καταλόγους στον τρέχοντα (compile-room), και με αυτά τα αρχεία κάνει compile στο τελικό μας πρόγραμμα, που ονομάζεται uni-c-analyser.

2 Τμηματικά ο κώδικας - Εξηγήσεις

2.1 FLEX

2.1.1 Includes και μεταβλητές

```
/* Kwdikas C gia orismo twn apaitoumenwn header files kai twn
   metablhtwn.
   Otidhpote anamesa sta %{ kai %} metaferetai autousio sto arxeio C
   pou
   tha dhmiourghsei to Flex. */

%{

#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include "bison-SA.tab.h"

// Για να μετράμε τις κολώνες
int columns = 1;
// Αρχικοποιούμε τις μεταβλητές για το άθροισμα των σωστών και λάθος
   λέξεων
int cor_words = 0;
int inc_words = 0;
// Για να καταφέρνουμε να δίνουμε στον χρήστη σωστό output.
char panic_cause_char[100];

%}
```

Εδώ βλέπουμε τα include μας, που είναι απαραίτητα τόσο για την λειτουργεία του προγράμματος (πχ. '#include <stdlib>' κτλ) όσο και για την σύνδεση του flex με του bison (πχ. '#include "bison-SA.tab.h"')
Επίσης, βλέπουμε τις μεταβλητές που ορίζουμε για την ομαλή διεπαφή του χρήστη με το πρόγραμμα.
Πλέον το πρόγραμμά μας μετρά κολώνες και μπορεί να κατευθύνει τον χρήστη στο ακριβές σημείο που το πρόβλημα δημιουργήθηκε (με κάποιο άγνωστο token).
Επίσης, σε σύγκριση με το μέρος B2, τώρα το πρόγραμμα μετρά σωστά

τις σωστές και λάθος εκφράσεις.
Τις λάθος λέξεις τις μετρά ο λεκτικός αναλυτής, ο FLEX. Τις λάθος εκφράσεις ο συντατικός αναλυτής, το BISON.
Επίσης, στην μεταβλητή panic_cause_char αποθηκεύουμε την άγνωστη λέξη, και την αναφέρουμε στον χρήστη μετέπειτα.

2.1.2 TOKENS, Κανονικές Εκφράσεις, Καταστάσεις

```
/* Onomata kai antistoixoi orismoi (ypo morfh kanonikhs ekfrashs).
   Meta apo auto, mporei na ginei xrhsh twn onomatwn (aristera) anti
    twn,
   synhthws idiaiterws makroskelwn kai dysnohtwn, kanonikwn ekfrasewn
    */

SEMI                ;
HASH                #
TILDE               ~
NEQ                 !=
MOD                 \%
POW                 \^
DOT                 \.
COMMA               \,
COLON               \:
AMPER               \&
PAR_END             \)
PAR_START           \(
BRACE_END           \}
BRACE_START         \{
BRACKET_END         \]
BRACKET_START       \[
LOGICAL_OR          \|\|
TYPE_EQ             ==?
TYPE_DIV            \/=?
TYPE_MULTI          \*=?
TYPE_EXCLA          \!=?
TYPE_AMPER          \&\&
TYPE_LESSER         \<=?
TYPE_GREATER        \>=?
WHITESPACE          [ \t]
TYPE_PLUS           \+[\+=]?
TYPE_MINUS          \-[\-=]?
STRING              '.*'|\".*\"
```

```
33 INTCONST             0|[1-9]+[0-9]*
34 COMMENT              \/\*(.|\n)*?\*\/|\/\/.*
35 IDENTIFIER           [a-zA-Z_]([0-9_a-zA-Z]*)
36 FLOAT                [0-9]+\.[0-9]+|[0-9]+\.[0-9]+e[0-9]+
```

Βλέπουμε τις κανονικές εκφράσεις με τις οποίες ο λεκτικός αναλυτής μας ανιχνεύει τις λέξεις του αναλυόμενου κώδικα, και τις συσχετίζει με ένα *μοναδικό* token.
Στο τέλος έχουμε και τους ορισμούς για τις 3 καταστάσεις τις οποίες ορίσαμε για να λειτουργεί ορθά ο λεκτικός αναλυτής.

```
1 %x REALLYEND
2 %x PREPANIC
3 %x PANIC
```

Αυτές είναι:

- **Κατάσταση REALLYEND** είναι η κατάσταση στην οποία ο λεκτικός αναλυτής έχει ήδη απο τον συντακτικό αναλυτής το μύνημα ότι ο πρώτος έχει λάβει τις τελευταίες δράσεις πριν τον τερματισμό του προγράμματος, και αρχίζει να κλείνει "πραγματικά" το πρόγραμμα. (Πριν την κατάσταση αυτή στέλνει μήνυμα EOF στον BISON, που παίρνει δράσεις που θα αναλύσουμε παρακάτω. Μετά από τις δράσεις αυτές, ο FLEX μπαίνει στην κατάσταση REALLYEND)

- **Κατάσταση PANIC** είναι η κατάσταση στην οποία ο λεκτικός αναλυτής έχει συναντίσει ένα άγνωστο token (UNKNOWN TOKEN) και προσπαθεί να κάνει επανάκτιση κανονικής λειτουργίας.

- **Κατάσταση PREPANIC** είναι η κατάσταση στην οποία ο λεξικός μας αναλυτής ειδοποιεί τον συντακτικό αναλυτή ότι πρόκειται να βρεθει (ο δεύτερος) σε κατάσταση πανικού.

2.1.3 Κώδικας κατά την ανίχνευση λεκτικών μονάδων

```
%%
{MOD}           {cor_words++; columns++; return MOD;}
{POW}           {cor_words++; columns++; return POW;}
{DOT}           {cor_words++; columns++; return DOT;}
{SEMI}          {cor_words++; columns++; return SEMI;}
{HASH}          {cor_words++; columns++; return HASH;}
{COMMA}         {cor_words++; columns++; return COMMA;}
{PAR_END}       {cor_words++; columns++; return PAR_END;}
{PAR_START}     {cor_words++; columns++; return PAR_START;}
{BRACE_END}     {cor_words++; columns++; return BRACE_END;}
{LOGICAL_OR}    {cor_words++; columns++; return LOGICAL_OR;}
{BRACE_START}   {cor_words++; columns++; return BRACE_START;}
{BRACKET_END}   {cor_words++; columns++; return BRACKET_END;}
{BRACKET_START} {cor_words++; columns++; return BRACKET_START;}
```

Συνεχίζοντας, παρατηρούμε πώς αυξάνουμε τον αριθμό των σωστών λέξεων για κάθε λέξη που ανιχνεύεται σωστά, αυξάνοντας και το αριθμό των κολωνών κατάλληλα επίσης.
Συνεχίζοντας σε λεκτικές μονάδες με περισσότερα γράμματα απο 1:

```
{FLOAT}    {cor_words++; columns += strlen(yytext); return FLOAT;}
{STRING}   {cor_words++; columns += strlen(yytext); return STRING;}
{INTCONST} {cor_words++; columns += strlen(yytext); return INTCONST;}
```

Εδώ είναι ξεκάθαρο ότι χρησιμοποιούμε την συνάρτηση strlen η οποία μετρά τις λεκτικές μονάδες "απρόβλεπτου" μήκους και προσθέτει τον αριθμό γραμμάτων τους στην μεταβλητή μέτρησης κολωνών columns. Ακολουθούν τα ονόματα και τα keywords.

```
{IDENTIFIER}    {
if      ( !strcmp(yytext,"do"      ))
  {cor_words++; columns += strlen(yytext); return KEYWORD;}
else if ( !strcmp(yytext,"while"   ))
  {cor_words++; columns += strlen(yytext); return KEYWORD;}
else if ( !strcmp(yytext,"break"   ))
  {cor_words++; columns += strlen(yytext); return KEYWORD;}
else if ( !strcmp(yytext,"if"      ))
  {cor_words++; columns += strlen(yytext); return KEYWORD_IF;}
else if ( !strcmp(yytext,"struct"  ))
  {cor_words++; columns += strlen(yytext); return KEYWORD_STR;}
else if ( !strcmp(yytext,"for"     ))
```

```
  {cor_words++; columns += strlen(yytext); return KEYWORD_FOR;}
else if ( !strcmp(yytext,"return"  ))
  {cor_words++; columns += strlen(yytext); return KEYWORD_RET;}
else if ( !strcmp(yytext,"case"    ))
  {cor_words++; columns += strlen(yytext); return KEYWORD_CASE;}
else if ( !strcmp(yytext,"else"    ))
  {cor_words++; columns += strlen(yytext); return KEYWORD_ELSE;}
else if ( !strcmp(yytext,"func"    ))
  {cor_words++; columns += strlen(yytext); return KEYWORD_FUNC;}
else if ( !strcmp(yytext,"void"    ))
  {cor_words++; columns += strlen(yytext); return KEYWORD_VOID;}
else if ( !strcmp(yytext,"sizeof"  ))
  {cor_words++; columns += strlen(yytext); return KEYWORD_SIZE;}
else if ( !strcmp(yytext,"include" ))
  {cor_words++; columns += strlen(yytext); return KEYWORD_INCL;}
else if ( !strcmp(yytext,"continue"))
  {cor_words++; columns += strlen(yytext); return KEYWORD_CONT;}
else if ( !strcmp(yytext,"switch"  ))
  {cor_words++; columns += strlen(yytext); return KEYWORD_SWITCH;}
else if ( !strcmp(yytext,"int"     ))
  {cor_words++; columns += strlen(yytext); return KEYWORD_VAR_TYPE;}
else if ( !strcmp(yytext,"char"    ))
  {cor_words++; columns += strlen(yytext); return KEYWORD_VAR_TYPE;}
else if ( !strcmp(yytext,"long"    ))
  {cor_words++; columns += strlen(yytext); return KEYWORD_VAR_TYPE;}
else if ( !strcmp(yytext,"short"   ))
  {cor_words++; columns += strlen(yytext); return KEYWORD_VAR_TYPE;}
else if ( !strcmp(yytext,"float"   ))
  {cor_words++; columns += strlen(yytext); return KEYWORD_VAR_TYPE;}
else if ( !strcmp(yytext,"const"   ))
  {cor_words++; columns += strlen(yytext); return KEYWORD_VAR_TYPE;}
else if ( !strcmp(yytext,"double"  ))
  {cor_words++; columns += strlen(yytext); return KEYWORD_VAR_TYPE;}
else
  {cor_words++; columns += strlen(yytext); return IDENTIFIER;}
}
```

Εδώ έχουμε το ενδεχόμενο κάποια έγκυρη λεκτική μονάδα που ανιχνεύσαμε να είναι *δευσμευμένη λέξη* της Uni-C.
Αν είναι, επιστρέφουμε το κατάλληλο token.
Αν όχι τότε επιστρέφουμε απλά token IDENTIFIER.

Πηγαίνοντας σε πιο πολύπλοκες λεκτικές μονάδες όπως οι τελεστές:

```
{TYPE_EXCLA}    { if (!strcmp(yytext, "!=")) {cor_words++; columns +=
    2; return NEQ;        } else    { columns++; return EXCLA;  }}
{TYPE_EQ}       { if (!strcmp(yytext, "==")) {cor_words++; columns +=
    2; return EQQ;        } else    { columns++; return EQ;     }}
{TYPE_DIV}      { if (!strcmp(yytext, "/=")) {cor_words++; columns +=
    2; return EQ_DIV;     } else    { columns++; return DIV;    }}
{TYPE_MULTI}    { if (!strcmp(yytext, "*=")) {cor_words++; columns +=
    2; return EQ_MULTI;   } else    { columns++; return MULTI;  }}
{TYPE_LESSER}   { if (!strcmp(yytext, "<=")) {cor_words++; columns +=
    2; return LESSER_EQ;  } else    { columns++; return LESSER; }}
{TYPE_GREATER}  { if (!strcmp(yytext, ">=")) {cor_words++; columns +=
    2; return GREATER_EQ; } else    { columns++; return GREATER;}}
{TYPE_AMPER}    { if (!strcmp(yytext, "&&")) {cor_words++; columns +=
    2; return LOGICAL_AND;} else    { columns++; return AMPER;  }}
{TYPE_MINUS}    { if (!strcmp(yytext, "--")) {cor_words++; columns +=
    2; return MINUSMINUS; } else if (!strcmp(yytext, "-=")) { columns
    +=2; return EQ_MINUS; } else { columns++; return MINUS;}}
{TYPE_PLUS}     { if (!strcmp(yytext, "++")) {cor_words++; columns +=
    2; return PLUSPLUS;   } else if (!strcmp(yytext, "+=")) { columns
    +=2; return EQ_PLUS;  } else { columns++; return PLUS; }}
```

Έχουμε ιδιαίτερη διαχείρηση των τελεστών για να είμαστε σίγουροι ότι ο λεκτικός μας αναλυτής δεν "μπερδεύει" για παράδειγμα, τον τελεστή "++" από τον "+", καθώς συντατικά είναι πολύ διαφορετική η συμπεριφορά τους.
Στην συνέχεια δίνονται οδηγίες στον λεκτικό αναλυτή για την διαχείρηση του κενού, της κανούργιας γραμμής και των σχολίων του κώδικα.

```
{WHITESPACE} { columns++; }
{COMMENT}    { /*Do nothing, comment*/ }
\n           { columns=1; /*Start from zero cols again*/ return NEWLINE; }
```

Για τα κενά ο λεκτικός αναλυτής απλά αυξάνει τις κολώνες, για τα σχόλια δεν κάνει τίποτα, ενώ για τις καινούργιες γραμμές επαναφέρει την μεταβλητή μέτρησης κολωνών columns στην αρχική τιμή 1.
Τελειώνοντας απο τον κώδικα του λεκτικού αναλυτή, βλέπουμε το πιο πολύπλοκο κομμάτι του.

2.1.4 Καταστάσεις (Πανικού!)

Μπαίνοντας στο πιό πολύπλοκο κομμάτι του λεκτικού μας αναλυτή, έχουμε την διαχείρηση λανθασμένων λέξεων, οι οποίες ρίχνουν τον αναλυτή μας σε μία κατάσταση "πανικού".
Απο αυτή την κατάσταση προσπαθεί μετά να "ξεφύγει" με το να αναφέρει το λάθος στον συντακτικό αναλυτή και να επανέλθει στην αρχική κατάσταση <INITIAL>, επανακκινόντας την λεκτική ανάλυση του υπόλοιπου αρχείου.

```
/*Εδώ το flex πιάνει"" οποιονδήποτε άλλο χαρακτήρα που
 δεν περιγράφεται απο τις παραπάνω κανονικές εκφράσεις.*/

<INITIAL>.  { BEGIN(PREPANIC); strcpy(panic_cause_char,yytext);
    inc_words++; return UNKNOWN;}
<PREPANIC>. { BEGIN(PANIC); printf("Column: %d Unknown word: '%s%s",
    columns,panic_cause_char,yytext);}
<PANIC>{WHITESPACE} { columns++; printf("'\n"); BEGIN(INITIAL);}
<PANIC>\n           { columns=1; printf("'\n"); BEGIN(INITIAL);}
<PANIC>.            { ECHO; }

/*Εδώ καλούμε ένα τμήμα κώδικα που μας βοηθά να δώσουμε ένα token
  στον bison για να δηλώσουμε το τέλος του αρχείου, αποτρέποντας
  όμως τον bison να τερματίζει άμεσα την εκτέλεση. Έτσι,
  καταφέρνουμε να εκτελούμε την συνάρτηση print_report() στο
  bison-SA.y, για να ανεφέρουμε τον αριθμό των σωστών και
  λανθασμένων λέξεων και εκφράσεων.*/

<INITIAL><<EOF>>  { BEGIN(REALLYEND);
                    printf("*---- RUN REPORT: --------------------*\n"
                           "|- Words:\n"
                           "| Number of  correct  words       : %d\n"
                           "| Number of incorrect words       : %d\n"
                           "*-----------------------------------*\n"
                           ,cor_words, inc_words);
                  return EOP; }
 /*Εδώ, μετά την πάροδο των προηγούμενων, πραγματικά""
  τερματίζουμε την εκτέλεση του flex, έχουμε ήδη τυπώσει την αναφορά
  με την print_report() με το bison, και αρχίζουμε να τερματίζουμε
  το πρόγραμμα συνολικά.*/
<REALLYEND><<EOF>> {yyterminate();}
%%

/* Το πρόγραμμα αυτό δεν έχει main(), καθώς δεν τρέχει αυτόνομα,είναι
```

```
33 απλά ο λεκτικός αναλυτής, η συντακτική ανάλυση γίνεταιαπό
34 τον bison. */
```