

Αρχές Γλωσσών Προγραμματισμού και Μεταφραστών

Απόστολος Ζεκυριάς (1100554) – Σπυρίδων Μανταδάκης (1100613) – Παναγιώτης Παπανικολάου (1104804) – Αλέξανδρος Γεώργιος Χαλαμπάκης (1100754)

# **Γενικές Πληροφορίες**

Απαντήσεις Project

c

Στις επόμενες σελίδες παρουσιάζονται οι απαντήσεις της ομάδας μας στο Project του μαθήματος "Αρχές Γλωσσών Προγραμματισμού και Μεταφραστών". Σε αυτήν τη σελίδα παρέχονται πληροφορίες σχετικά με τα μέλη της ομάδας.

Η ομάδα αποτελέιται από τους εξής φοιτητές:

Απόστολος Ζεκυριάς

Σπυρίδων Μανταδάκης

Παναγιώτης Παπανικολάου

Αλέξανδρος Γεώργιος Χαλαμπάκης

**Αναλυτικότερες Πληροφορίες:**

[A person wearing glasses and a black shirt

Description automatically generated](https://www.youtube.com/watch?v=dRW2LBKzM7A)[Απόστολος](https://youtu.be/v3a6Fz9fWe4?si=934JN9VypLPD8Lft) [Ζεκυριάς](https://youtu.be/YgC3SpOWrvw?si=TTqcnArgTyh0i8dv) 1100554

[up1100554@ac.upatras.gr](mailto:up1100554@ac.upatras.gr)

[Φοιτητής 2ου έτους](https://docs.google.com/document/d/10-blbL7yAtm5ahq3mtDn46sf4O0fPu424H8AR08jXIU/edit#heading=h.yx8l8bvprakc)

[A person sitting in a parka

Description automatically generated](https://www.pornhub.com/view_video.php?viewkey=66b718dd87ad3#1)Σπυρίδων Μανταδάκης 1100613

[up1100613@ac.upatras.gr](mailto:up1100613@ac.upatras.gr)

Φοιτητής 2ου έτους

A person sitting at a table with his hand on his chin

Description automatically generatedΠαναγιώτης Παπανικολάου 1104804

[up1104804@ac.upatras.gr](mailto:up1104804@ac.upatras.gr)

Φοιτητής 2ου έτους

A person drinking from a cup

Description automatically generatedΑλέξανδρος Γεώργιος Χαλαμπάκης 1100754

[up1100754@ac.upatras.gr](mailto:up1100754@ac.upatras.gr)

Φοιτητής 2ου έτους

**Περιεχόμενα**

1. Αρχική μορφή της περιγραφής της γραμματικής της γλώσσας σε BNF.……………………………………………………………………………………….  **3**
2. Τελική μορφή της περιγραφής του λεξικού αναλυτή μας……………………**6**
3. Τελική μορφή της περιγραφής του συντακτικού αναλυτή μας……………**10**
4. Screenshots παραδειγμάτων εφαρμογής……………………………………..**22**
5. Σχόλια……………………………………………………………………………………..**28**

**Αρχική μορφή της περιγραφής της γραμματικής της γλώσσας σε BNF.**

**Παρακάτω φαίνεται η περιγραφή της γραμματικής της γλώσσας μας σε BNF με την οποία αρχίσαμε την υλοποίηση της εργασίας και τροποποιήσαμε αργότερα για την ορθή λειτουργία της.**

%%

<PROGRAM> : := START <STATEMENTS> END /\* ΟΡΙΣΜΟΣ ΠΡΟΓΡΑΜΜΑΤΟΣ\*/

<STATEMENTS>: := <STATEMENT> [<STATEMENTS>] /\*ΟΡΙΣΜΟΣ ΔΗΛΩΣΕΩΝ \*/

<STATEMENTS> : := <STATEMENT\_IF> /\* ΟΡΙΣΜΟΣ ΔΗΛΩΣΗΣ\*/

| <STATEMENT\_WHILE>

| <STATEMENT\_ASSIGN>

| <STATEMENT\_SWITCH>

| <STATEMENT\_RETURN>

| <STATEMENT\_CLASS>

| <VARIABLE\_DECLARATION>

| < METHOD\_DECLARATION>

| <STATEMENT\_DO\_WHILE>

| <STATEMENT\_FOR>

| <STATEMENT\_PRINT>

| <CREATE\_CLASS\_OBJECT>

|<STATEMENT\_BREAK>

|<COMMENTS>

| <PROGRAM>

<STATEMENT\_IF> : := “IF” ”(“ <CONDITION>”)” “{“<STATEMENTS>”}” /\* ΟΡΙΣΜΟΣ ΔΗΛΩΣΗΣ IF \*/

| “IF” ”(“ <CONDITION>”)” “{“<STATEMENTS>”}” <ELSE\_IF> “ELSE ”

“{“<STATEMENTS>”}”

<ELSE\_IF> ::=”ELSE IF” ”(“ <CONDITION>”)” “{“<STATEMENTS>”}” /\* ΟΡΙΣΜΟΣ ELSE IF\*/  
 | ”(“ <CONDITION>”)” “{“<STATEMENTS>”}” <ELSE\_IF>

<STATEMENT\_WHILE> : := ”WHILE” “(“ <CONDITION>”)” “{“ <STATEMENTS> ”}”

/\* ΟΡΙΣΜΟΣ ΔΗΛΩΣΗΣ WHILE \*/

<STATEMENT\_BREAK> ::= “BREAK” “;” /\*ΟΡΙΣΜΟΣ ΔΗΛΩΣΗΣ BREAK\*/

<STATEMENT\_ASSIGN> ::= <VARIABLE\_DECLARATION> “=” <EXPRESSION> “;”

/\* ΟΡΙΣΜΟΣ ΔΗΛΩΣΗΣ «=»\*/

<STATEMENT\_SWITCH> ::= “SWITCH” <EXPRESSION> <SWITCH\_BODY> /\*ΟΡΙΣΜΟΣ ΔΗΛΩΣΗΣ SWITCH\*/

<COMMENTS> ::= "// " <CHARACTERS> /\*ΟΡΙΣΜΟΣ ΣΧΟΛΙΩΝ\*/

| " /\* " <CHARACTERS> " \*/ "

<STATEMENT\_PRINT> ::="OUT.PRINT” “(“ < STRING\_LITERAL > [ "," <EXPRESSION>] ”)” “;”

/\*ΟΡΙΣΜΟΣ ΔΗΛΩΣΗΣ PRINT\*/

<VARIABLE\_DECLARATION>::= [<ACCESS\_MODIFIER> ] <VARIABLE\_TYPE> <IDENTIFIER> {

/\* ΟΡΙΣΜΟΣ ΔΗΛΩΣΗΣ ΜΕΤΑΒΛΗΤΗΣ\*/

< METHOD\_DECLARATION>::=<ACCESS\_MODIFIER> <RETURN\_TYPE> <IDENTIFIER> “(“ [<PARAMETER\_LIST> ] “)” "{" [<VARIABLE\_DECLARATION>] <STATEMENTS> “}”

/\*ΟΡΙΣΜΟΣ ΔΗΛΩΣΗΣ ΜΕΘΟΔΟΥ\*/

<RETURN\_TYPE> ::=<VARIABLE\_TYPE> /\*ΟΡΙΣΜΟΣ ΤΥΠΟΥ RETURN\*/  
 |”VOID”

<ACCESS\_MODIFIER> ::= “PUBLIC” /\* ΟΡΙΣΜΟΣ ΤΡΟΠΟΠΟΙΗΤΗΣ ΠΡΟΣΒΑΣΗΣ\*/  
 |“PRIVATE”

<VARIABLE\_TYPE> ::= “INT” /\*ΟΡΙΣΜΟΣ ΤΥΠΟΥ ΜΕΤΑΒΛΗΤΗΣ\*/  
 | “CHAR”  
 | “DOUBLE”   
 | “BOOLEAN”  
 | “STRING”

<PARAMETER\_LIST> ::=<VARIABLE\_TYPE> <IDENTIFIER> [“,” <PARAMETER\_LIST> ]

/\*ΟΡΙΣΜΟΣ ΛΙΣΤΑΣ ΠΑΡΑΜΕΤΡΩΝ\*/

<CONDITION> ::= <EXPRESSION > < COMPARISON> <EXPRESSION> /\* ΟΡΙΣΜΟΣ ΣΥΝΘΗΚΗΣ\*/

| <EXPRESSION > < COMPARISON> <EXPRESSION> “&&” <CONDITION>

| <EXPRESSION > < COMPARISON> <EXPRESSION> “||” <CONDITION>

<SWITCH\_BODY> ::= “CASE” <EXPRESSION> “:” <STATEMENTS> [ “DEFAULT” “:” <STATEMENTS> ]

| “CASE” <EXPRESSION>“:” <STATEMENTS> <SWITCH\_BODY>

/\*ΟΡΙΣΜΟΣ ΣΩΜΑΤΟΣ SWITCH\*/

<STATEMENT\_RETURN> ::= “RETURN” <EXPRESSION> “;” /\*ΟΡΙΜΣΟΣ RETURN\*/

<STATEMENT\_CLASS>::= <ACCESS\_MODIFIER> “CLASS” <CLASS\_IDENTIFIER>”{“ <VARIABLE\_DECLARATION> <METHOD\_DECLARATION>”}”

/\*ΟΡΙΣΜΟΣ ΚΛΑΣΗΣ\*/

<CLASS\_IDENTIFIER> ::= <UPPERCASE> [<IDENTIFIER>] /\*ΟΡΙΣΜΟΣ ΑΝΑΓΝΩΡΙΣΤΙΚΟΥ ΚΛΑΣΗΣ\*/

<CREATE\_CLASS\_OBJECT> ::= <CLASS\_IDENTIFIER> <IDENTIFIER> “=” <STATEMENT\_NEW> ”();”

/\*ΟΡΙΣΜΟΣ ΔΗΜΙΟΥΡΓΙΑΣ ΑΝΤΙΚΕΙΜΕΝΟΥ ΚΛΑΣΗΣ\*/

<STATEMENT\_DO\_WHILE> ::= “DO” “{“ <STATEMENTS> “}” ”WHILE” “(“ CONDITION “)”

/\*ΟΡΙΣΜΟΣ ΔΗΛΩΣΗΣ DO WHILE\*/

<ACCESS\_TO\_CLASS\_MEMBERS> ::= <IDENTIFIER >“.”, IDENTIFIER>”;”

/\*ΟΡΙΣΜΟΣ ΠΡΟΣΒΑΣΗΣ ΣΤΑ ΜΕΛΗ ΚΛΑΣΗΣ \*/

<STATEMENT\_FOR> ::= ”FOR” “(“ <STATEMENT\_ASSIGN> “;” <CONDITION> “;” <STATEMENT\_ASSIGN> “)” ”{“

<STATEMENTS> ”}”

/\*ΟΡΙΣΜΟΣ ΔΗΛΩΣΗΣ FOR\*/

<OPERATION\_CONTINUE> ::= “+” “(“ <OPERATION> “)” /\*ΟΡΙΣΜΟΣ ΣΥΝΕΧΕΙΑΣ ΠΡΑΞΗΣ\*/

|”-“ “(“ <OPERATION> “)”

| “\*” “(“ <OPERATION> “)”

| “/” “(“ <OPERATION> “)”

<ADDITION> ::= <VALUE> “+” <VALUE> [<OPERATION\_CONTINUE> ] /\* ΟΡΙΣΜΟΣ ΑΘΡΟΙΣΗΣ\*/

< MULTIPLICATION> ::= <VALUE> “\*” <VALUE> [<OPERATION\_CONTINUE> ]

/\*ΟΡΙΣΜΟΣ ΠΟΛΛΑΠΛΑΣΙΑΣΜΟΥ\*/

<SUBTRACTION> ::= <VALUE> "-" <VALUE> [<OPERATION\_CONTINUE>] /\*ΟΡΙΣΜΟΣ ΑΦΑΙΡΕΣΗΣ\*/

<DIVISION> ::=<VALUE> "/" <VALUE> [<OPERATION\_CONTINUE> ] /\* ΟΡΙΣΜΟΣ ΔΙΑΙΡΕΣΗΣ\*/

<IDENTIFIER> ::= < LETTER> /\* ΟΡΙΣΜΟΣ ΑΝΑΓΝΩΡΙΣΤΙΚΟΥ \*/

| <IDENTIFIER> <LETTER>

| <IDENTIFIER> <NUMBER>

| <IDENTIFIER> “\_”

<NUMBER> ::= <DIGIT> [<NUMBER>] /\* ΟΡΙΣΜΟΣ ΑΡΙΘΜΟΥ\*/

<COMPARISON> ::= “>” /\*ΟΡΙΣΜΟΣ ΣΥΓΚΡΙΣΗΣ\*/

| “<”

| «<=»

| «>»

| «==»

| «=!»

<EXPRESSION> ::= <VALUE> /\* ΟΡΙΣΜΟΣ ΕΚΦΡΑΣΗΣ\*/

| <OPERATION>

| <STATEMENT\_ NEW>

| <BOOLEAN>

<BOOLEAN> ::= “TRUE” /\*ΟΡΙΣΜΟΣ BOOLEAN \*/

| “FALSE”

<STATEMENT\_NEW> ::= "NEW" <VARIABLE\_TYPE>

| “NEW” <CLASS\_IDENTIFIER> /\* ΟΡΙΣΜΟΣ NEW\*/

<VALUE> ::= < NUMBER> /\* ΟΡΙΣΜΟΣ ΤΙΜΗΣ \*/

| <IDENTIFIER>

<OPERATION> ::= <ADDITION> /\* ΟΡΙΣΜΟΣ ΠΡΑΞΗΣ\*/

| < MULTIPLICATION>

| <SUBTRACTION>

| <DIVISION>

<DIGITIT> ::=” 0”|”1”|”2”|”3”|”4”|”5”|”6”|”7”|”8”|”9” /\*ΟΡΙΣΜΟΣ ΨΗΦΙΟΥ\*/

<LETTER> ::= <UPPERCASE> /\*ΟΡΙΣΜΟΣ ΓΡΑΜΜΑΤΟΣ\*/

| <LOWERCASE>

<UPPERCASE> ::= “A”|”B”|”C”|”D”|”E”|”F”|”G”|”H”|”I”|”J”|”K”|”L” /\*ΟΡΙΣΜΟΣ ΚΕΦΑΛΑΙΩΝ ΓΡΑΜΜΑΤΩΝ\*/

|”M”|”N”|”O”|”P”|”Q”|”R”|”S”|”T”|”U”|”V”|”W”|”X”|”Y”|”Z”

<LOWERCASE> :: = “a”|”b”|”c”|”d”|”e”|”f”|”g”|”h”|”I”|”j”|”k”|”l” /\*ΟΡΙΣΜΟΣ ΜΙΚΡΩΝ ΓΡΑΜΜΑΤΩΝ\*/

|”m”|”n”|”o”|”p”|”q”|”r”|”s”|”t”|”u”|”v”|”w”|”x”|”y”|”z”

<STRING\_LITERAL> ::= ’ ” ’ <CHARACTERS> ’ ” ’ /\*ΟΡΙΣΜΟΣ ΔΗΛΩΣΗΣ ΚΕΙΜΕΝΟΥ ΤΟΥ PRINT \*/

<CHARACTERS> ::= < <CHARACTER> [<CHARACTERS>] /\*ΟΡΙΣΜΟΣ ΧΑΡΑΚΤΗΡΩΝ\*/

<CHARACTER> ::=<DIGIT> /\*ΟΡΙΣΜΟΣ ΧΑΡΑΚΤΗΡΑ\*/  
 |<LETTER>  
 |<SPECIAL\_CHARACTERS>

<SPECIAL\_CHARACTERS> ::= "\_" | "+" | "-" | "\*" | "/" | "(" | ")" | "{" | "}" | "[" | "]" | "<" | ">" | "=" | "!" | "&" | "|" | ";" | "," | "." |”%”|”#”|”@”|”^”|”?”| "'" | '"' | " " | "\t" | "\n" | "\r" |”\n”

\*ΟΡΙΣΜΟΣ ΕΙΔΙΚΩΝ ΧΑΡΑΚΤΗΡΩΝ\*/

%%

**Τελική μορφή της περιγραφής του λεξικού αναλυτή μας:**

%{

/\* DEFINITIONS \*/

#include <stdio.h> /\* Standard input/output library \*/

#include <stdlib.h> /\* Standard library for memory allocation \*/

#include <string.h> /\* String manipulation functions \*/

#include <ctype.h> /\* Character classification functions \*/

#include "bison\_Program.tab.h" /\* Include bison header file \*/

/\* External references to Bison parser \*/

extern FILE \*yyin;

extern FILE \*yyout;

/\* Function for error handling \*/

extern void yyerror(char \*s);

/\* Buffer for storing strings \*/

char string\_buffer[1000];

char\* string\_buffer\_pointer;

int line\_in=-1;

%}

/\* Flex options \*/

%option noyywrap

%option yylineno

/\* Define states for multi-line, single-line comments, and string contents \*/

%x COMMENT\_CONTENTS

%x ONELINE\_COMMENT\_CONTENTS

%x STRING\_CONTENTS

/\* Regular expression definitions for easier reading \*/

DIGIT [0-9]

LETTER [a-zA-Z]

WHITESPACE [ \t]+

NEWLINE [\n]+

NUMBER [1-9]{DIGIT}\*|"0"

DOUBLE {DIGIT}+\.{DIGIT}+d

IDENTIFIER [a-zA-Z\_][a-zA-Z0-9\_]\*

%%

"if"                        { yylval.strvalue = strdup(yytext); return TOKEN\_IF; }

"else"                      { yylval.strvalue = strdup(yytext); return TOKEN\_ELSE; }

"while"                     { yylval.strvalue = strdup(yytext); return TOKEN\_WHILE; }

"do"                        { yylval.strvalue = strdup(yytext); return TOKEN\_DO; }

"for"                       { yylval.strvalue = strdup(yytext); return TOKEN\_FOR; }

"switch"                    { yylval.strvalue = strdup(yytext); return TOKEN\_SWITCH; }

"case"                      { yylval.strvalue = strdup(yytext); return TOKEN\_CASE; }

"default"                   { yylval.strvalue = strdup(yytext); return TOKEN\_DEFAULT; }

"break"                     { yylval.strvalue = strdup(yytext); return TOKEN\_BREAK; }

"return"                    { yylval.strvalue = strdup(yytext); return TOKEN\_RETURN; }

"class"                     { yylval.strvalue = strdup(yytext); return TOKEN\_CLASS; }

"public"                    { yylval.strvalue = strdup(yytext); return TOKEN\_PUBLIC; }

"private"                   { yylval.strvalue = strdup(yytext); return TOKEN\_PRIVATE; }

"int"                       { yylval.strvalue = strdup(yytext); return TOKEN\_INT; }

"char"                      { yylval.strvalue = strdup(yytext); return TOKEN\_CHAR; }

"void"                      { yylval.strvalue = strdup(yytext); return TOKEN\_VOID; }

"double"                    { yylval.strvalue = strdup(yytext); return TOKEN\_DOUBLE; }

"bool"                      { yylval.strvalue = strdup(yytext); return TOKEN\_BOOLEAN; }

"string"                    { yylval.strvalue = strdup(yytext); return TOKEN\_STRING; }

"true"                      { yylval.strvalue = strdup(yytext); return TOKEN\_TRUE; }

"false"                     { yylval.strvalue = strdup(yytext); return TOKEN\_FALSE; }

"new"                       { yylval.strvalue = strdup(yytext); return TOKEN\_NEW; }

"out.print"                 { yylval.strvalue = strdup(yytext); return TOKEN\_OUT\_PRINT; }

";"                         { return TOKEN\_SEMICOLON; }

"{"                         { return TOKEN\_LBRACE; }

"}"                         { return TOKEN\_RBRACE; }

"("                         { return TOKEN\_LPAREN; }

")"                         { return TOKEN\_RPAREN; }

"["                         { return TOKEN\_LBRACKET; }

"]"                         { return TOKEN\_RBRACKET; }

"="                         { return TOKEN\_ASSIGN; }

","                         { return TOKEN\_COMMA; }

"+"                         { return TOKEN\_PLUS; }

"-"                         { return TOKEN\_MINUS; }

"\*"                         { return TOKEN\_MULT; }

"/"                         { return TOKEN\_DIV; }

"<"                         { return TOKEN\_LESS\_THAN; }

">"                         { return TOKEN\_GREATER\_THAN; }

"=="                        { return TOKEN\_EQUAL; }

"!="                        { return TOKEN\_NOT\_EQUAL; }

"<="                        { return TOKEN\_LESS\_THAN\_EQUAL; }

">="                        { return TOKEN\_GREATER\_THAN\_EQUAL; }

"&&"                        { return TOKEN\_AND; }

"||"                        { return TOKEN\_OR; }

"%"                         { return TOKEN\_MODULO; }

"#"                         { return TOKEN\_HASH; }

"@"                         { return TOKEN\_AT; }

"^"                         { return TOKEN\_CARET; }

"?"                         { return TOKEN\_QUESTION\_MARK; }

"."                         { return TOKEN\_DOT; }

"!"                         { return TOKEN\_EXCLAMATION\_POINT; }

"|"                         { return TOKEN\_PIPE; }

":"                         { return TOKEN\_COLON; }

{DOUBLE}                    { yylval.dvalue = atof(yytext); return DOUBLE\_NUMBER;}

{NUMBER}                    { yylval.intvalue = atoi(yytext); return NUMBER; }

\'[^\']\'                   { yylval.charvalue = yytext[1]; return CHARACTER; }

{IDENTIFIER}                { yylval.strvalue = strdup(yytext); return IDENTIFIER;}

"/\*"                                     {BEGIN(COMMENT\_CONTENTS); string\_buffer\_pointer = string\_buffer;line\_in=0;}

<COMMENT\_CONTENTS>"\*"+"/"                {BEGIN(INITIAL); \*string\_buffer\_pointer ='\0';

                                              line\_in=-1; printf("Multiple Line Comment\n");}

<COMMENT\_CONTENTS>"\n"                   {line\_in++;}

<COMMENT\_CONTENTS>[^\*\n]\*                {char \*ypointer = yytext;

                                              while (\*ypointer ) \*string\_buffer\_pointer++ = \*ypointer++; }

<COMMENT\_CONTENTS>[\*]\*                   {}

<COMMENT\_CONTENTS><<EOF>>                {yyerror("Comment not terminated"); return 0;}

"//"                                     {BEGIN(ONELINE\_COMMENT\_CONTENTS); string\_buffer\_pointer = string\_buffer;}

<ONELINE\_COMMENT\_CONTENTS>.              {}

<ONELINE\_COMMENT\_CONTENTS>[^\\\n\"]+     {char \*ypointer = yytext;

                                              while( \*ypointer) \*string\_buffer\_pointer++ = \*ypointer++; }

<ONELINE\_COMMENT\_CONTENTS>"\n"           {BEGIN(INITIAL); \*string\_buffer\_pointer = '\0'; printf("One Line Comment\n");}

\"                                       {BEGIN(STRING\_CONTENTS); string\_buffer\_pointer= string\_buffer; }

<STRING\_CONTENTS>\"                      {BEGIN(INITIAL); \*string\_buffer\_pointer = '\0';yylval.strvalue = strdup(string\_buffer);

                                               return STRING\_LITERAL; }

<STRING\_CONTENTS>\\n                     {\*string\_buffer\_pointer++ = '\n';}

<STRING\_CONTENTS>\\t                     {\*string\_buffer\_pointer++ = '\t';}

<STRING\_CONTENTS>\\\"                    {\*string\_buffer\_pointer++ = '"';}

<STRING\_CONTENTS>[^\\\n\"]+              {char \*ypointer = yytext;

                                               while ( \*ypointer ) \*string\_buffer\_pointer++ = \*ypointer++;}

<STRING\_CONTENTS><<EOF>>                    {yyerror("String not terminated"); return 0;}

{WHITESPACE}                             {}

{NEWLINE}                                {}

. {yyerror("Invalid character");}

%%

**Τελική μορφή της περιγραφής του συντακτικού αναλυτή μας**

**Screenshots παραδειγμάτων εφαρμογής.**

Αρχικά εκτελουμε στο terminal τις εντολές:

flex flex\_Program.l

**A black background with yellow and green text

Description automatically generated**

και bison bison\_Program.y

**A black background with yellow and green text

Description automatically generated**

Μετά κάνουμε compile τα δύο αρχεία εκτελώντας την εντολή:

gcc bison\_Program.tab.c lex.yy.c -o myParser.exe -lfl**A black background with yellow and green text

Description automatically generated**

και τρέχουμε το αρχείο εισόδου (input\_file.txt) με την εντολή:

./myParser intput\_file.txt

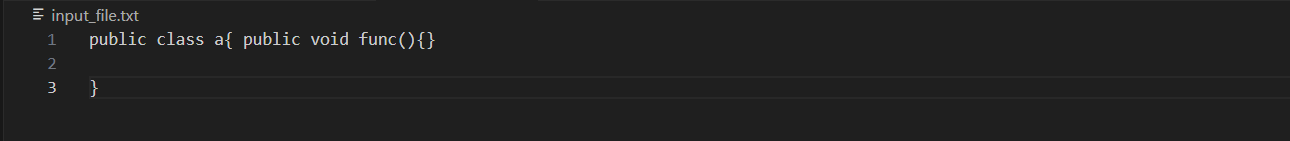
**A black background with yellow and green text

Description automatically generated**

**ΕΡΩΤΗΜΑ 1**

ΠΑΡΑΔΕΙΓΜΑ 1

Το αρχέιο εισόδου περίεχει τον ψευδοκώδικα:



και με την εκτέλεση της εντολής:

./myParser intput\_file.txt

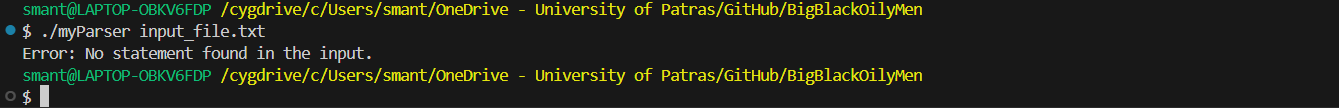
Το terminal εκτυπώνει :

A black screen with yellow and white text

Description automatically generated

ΠΑΡΑΔΕΙΓΜΑ 2

Όταν το αρχείο εισόδου είναι κένο, το terminal εκτυπώνει:



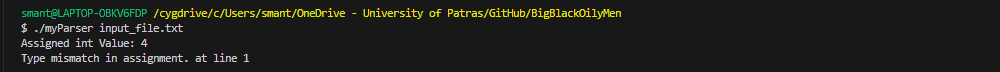
ΠΑΡΑΔΕΙΓΜΑ 3

Όταν το αρχείο εισόδου περιέχει τον ψευδοκώδικα:

A black rectangular object with white lines

Description automatically generated

Το terminal εκτυπώνει:

****

**ΕΡΩΤΗΜΑ 2**

ΠΑΡΑΔΕΙΓΜΑ 1

Το αρχέιο εισόδου περίεχει τον ψευδοκώδικα:

A black background with a black line

Description automatically generated

Το terminal εκτυπώνει :

A black screen with yellow text

Description automatically generated

ΠΑΡΑΔΕΙΓΜΑ 1

Το αρχέιο εισόδου περίεχει τον ψευδοκώδικα:

A screen shot of a computer

Description automatically generated

Το terminal εκτυπώνει :A computer screen with text on it

Description automatically generated

Η επικαιροποιημένη BNF για την υλοποίηση του ερωτήματος 2 περιέχει το παρακάτω κομμάτι κώδικα: A screenshot of a computer program

Description automatically generated

**ΕΡΩΤΗΜΑ 3**

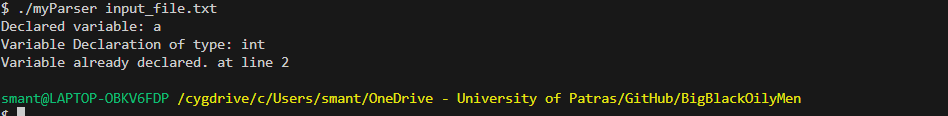
ΠΑΡΑΔΕΙΓΜΑ 1

Το αρχέιο εισόδου περίεχει τον ψευδοκώδικα:

A black rectangular object with a black line

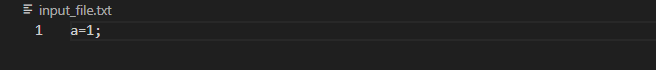
Description automatically generated

Το terminal εκτυπώνει:



ΠΑΡΑΔΕΙΓΜΑ 2

Το αρχέιο εισόδου περίεχει τον ψευδοκώδικα:



Το terminal εκτυπώνει: A black screen with yellow text

Description automatically generated

ΠΑΡΑΔΕΙΓΜΑ 3

Το αρχέιο εισόδου περίεχει τον ψευδοκώδικα:

A screen shot of a computer

Description automatically generated

Το terminal εκτυπώνει:

A screen shot of a computer

Description automatically generated

ΠΑΡΑΔΕΙΓΜΑ 4

Το αρχέιο εισόδου περίεχει τον ψευδοκώδικα:

A black screen with white text

Description automatically generated

Το terminal εκτυπώνει:

A computer screen with text on it

Description automatically generated

**ΣΧΟΛΙΑ**

Για την ανάπτυξη της εργασίας ξεκινήσαμε από την αρχική μορφή της BNF γραμματικής και σταδιακά την τροποποιήσαμε ώστε να μπορεί σε συνδυασμό με το αρχείο flex να υλοποιεί τα ζητούμενα.

Για το Ερώτημα 1, συγκεκριμένα το error handling, χρησιμοποιούμε το yyerror και άλλα print messages για να αντιμετωπίζουμε syntax errors, σωστή μορφή class identifier, error για κενό αρχείο εισόδου, error για απουσία δήλωσης class στο εκτελέσιμο αρχείο και πολλά άλλα διαφορετικά errors.

Ένα πρόβλημα που αντιμετωπίσαμε ήταν η αναφορά στη σωστή γραμμή σε περίπτωση που λείπει το τελευταίο token πριν την ολοκλήρωση δεξιού μέλους του κανόνα. Μετά από αρκετή προσπάθεια δεν βρήκαμε κάποιο τρόπο λύσης, αλλά κατανοήσαμε το λόγο που προκαλείται αυτό το πρόβλημα. Ο Parser λειτουργεί σαν stack στο οποίο κάνουμε shift tokens, και όταν αναγνωρίζεται ένα δεξιό μέλος ενός κανόνα, κάνουμε reduce. Ουσιαστικά όταν λείπει το τελευταίο token στον κανόνα, ο Parser συνεχίζει στις επόμενες γραμμές μέχρι να βρει ένα token. Αν βρει token αλλά δεν είναι εκείνο για την ολοκλήρωση του κανόνα, σταματάει εκεί και αναφέρει το syntax error αλλά στη γραμμή “yylineno” που βρήκε το επόμενο token. Επιπλέον, οι εντολές συνθήκης και εντολές loop μόνο αναγνωρίζονται από τον Parser ως κανόνες και δεν υλοποιούν τις λειτουργίες που έχει μια εντολή Loop και συνθήκης.

Στο Ερώτημα 2 τροποποιήσαμε τον κανόνα “VARIABLE\_DECLARATION” για να μπορεί να διαχειρίζεται δηλώσεις πολλών μεταβλητών καθώς και δηλώσεις με ανάθεση τιμής.

Στο Ερώτημα 3 προσθέσαμε Symbol Table για τον έλεγχο και την αποθήκευση μεταβλητών και μεθόδων. Σε συνδυασμό με τις βοηθητικές μεθόδους που έχουν οριστεί στην αρχή του bison αρχείου, σε κάθε δήλωση μεταβλητής και μεθόδου γίνεται έλεγχος αν υπάρχει ήδη, αν δεν υπάρχει τότε προστίθεται στην αντίστοιχη συνδεδεμένη λίστα.

Το β ερώτημα το προσπαθήσαμε αλλά δεν καταφέραμε να ενσωματώσουμε τις αλλαγές για scope handling στην τελική έκδοση του αρχείου Bison διότι δημιουργήθηκαν αρκετά προβλήματα και errors.

Στο γ ερώτημα καταφέραμε να υλοποιήσουμε την εκτέλεση αριθμητικών πράξεων καθώς και την ανάθεση αριθμητικού αποτελέσματος πράξης σε μεταβλητή. Δεν καταφέραμε ωστόσο να αποθηκεύσουμε τιμές σε μεταβλητές οι οποίες να μπορούν μετά να χρησιμοποιούνται από το υπόλοιπο πρόγραμμα καθώς και να αλλάξουν. Επιπλέον, ένα άλλο πρόβλημα που δεν προλάβαμε να επιλύσουμε είναι ο έλεγχος τύπου δεδομένων σε αριθμητικές πράξεις, π.χ. δεν βγαίνει error στον κώδικα: “int a= “String” +5;” ενώ θα έπρεπε.