|  |
| --- |
|  |
| Εργαστηριακή Άσκηση Flex & Bison |
| Αρχές Γλωσσών Προγραμματισμού & Μεταφραστών |

|  |
| --- |
| ΔΕΛΗΜΠΑΛΤΑΔΑΚΗΣ ΓΡΗΓΟΡΙΟΣ  ΑΜ: 1084647  MAIL: up1084647@upnet.gr  ΔΑΣΚΑΛΑΚΗΣ ΑΛΚΙΒΙΑΔΗΣ  ΑΜ:  MAIL: up@upnet.gr  ΡΑΙΚΟΣ ΙΑΣΩΝ  ΑΜ:  MAIL: up@upnet.gr |

Περιεχόμενα

[Περιγραφή της γραμματικής της γλώσσας σε BNF 2](#_Toc137074053)

[Περιγραφή της υλοποίησης του λεξικού και συντακτικού αναλυτή 3](#_Toc137074054)

[Λεκτικός Αναλυτής (Flex) 3](#_Toc137074055)

[Συντακτικός Αναλυτής (Bison) 3](#_Toc137074056)

[Παραδείγματα Εφαρμογής 4](#_Toc137074057)

[Σχόλια - Παραδοχές 5](#_Toc137074058)

[Τελικός κώδικας περιγραφής της γλώσσας 6](#_Toc137074059)

[lexer.l: 6](#_Toc137074060)

[parser.y: 9](#_Toc137074061)

# Περιγραφή της γραμματικής της γλώσσας σε BNF

# Περιγραφή της υλοποίησης του λεξικού και συντακτικού αναλυτή

Μια μικρή εισαγωγή

## Λεκτικός Αναλυτής (Flex)

## Συντακτικός Αναλυτής (Bison)

# Παραδείγματα Εφαρμογής

Screenshots με ένα παράδειγμα για κάθε ερώτημα

# Σχόλια - Παραδοχές

# Τελικός κώδικας περιγραφής της γλώσσας

## lexer.l:

%{

#include <stdio.h>

#include "parser.tab.h"

void yyerror(const char \*);

extern int rb\_number;

int rb\_counter = 0;

void check\_rb\_num();

void check\_radio\_button\_id();

extern int flag\_checked;

extern char checked\_button\_id[100];

%}

%option yylineno

**%%**

"<LinearLayout" { printf("%s", yytext); return LIN\_LAYOUT\_OPEN\_TAG; }

">" { printf("%s", yytext); return GT; }

"/>" { printf("%s", yytext); return CLOSE\_TAG; }

"</LinearLayout>" { printf("%s", yytext); return LIN\_LAYOUT\_CLOSE\_TAG; }

"android:layout\_width" { printf("%s", yytext); return LAYOUT\_WIDTH; }

"android:layout\_height" { printf("%s", yytext); return LAYOUT\_HEIGHT; }

"android:id" { printf("%s", yytext); return ID; }

"android:orientation" { printf("%s", yytext); return ORIENTATION; }

"<RelativeLayout" { printf("%s", yytext); return RE\_LAYOUT\_OPEN\_TAG; }

"</RelativeLayout>" { printf("%s", yytext); return RE\_LAYOUT\_CLOSE\_TAG; }

"<TextView" { printf("%s", yytext); return TEXT\_OPEN\_TAG; }

"android:text" { printf("%s", yytext); return TEXT; }

"android:textColor" { printf("%s", yytext); return TEXT\_COLOR; }

"<ImageView" { printf("%s", yytext); return IMAGE\_OPEN\_TAG; }

"android:src" { printf("%s", yytext); return SRC; }

"android:padding" { printf("%s", yytext); return PADDING; }

"<Button" { printf("%s", yytext); return BUTTON\_OPEN\_TAG; }

"<RadioGroup" { printf("%s", yytext); return R\_GROUP\_OPEN\_TAG; }

"</RadioGroup>" { check\_rb\_num(); check\_radio\_button\_id(); printf("%s", yytext); return R\_GROUP\_CLOSE\_TAG; }

"<RadioButton" { rb\_counter++; printf("%s", yytext); return R\_BUTTON\_OPEN\_TAG; }

"android:checkedButton" { printf("%s", yytext); return CHECKED\_BUTTON; }

"android:rb\_number" { printf("%s", yytext); return RB\_NUMBER; }

"<ProgressBar" { printf("%s", yytext); return PRO\_BAR\_OPEN\_TAG; }

"android:max" { printf("%s", yytext); return MAX; }

"android:progress" { printf("%s", yytext); return PROGRESS; }

"<!--"[^-]\*(-[^-]+)\*"-->" { printf("%s", yytext); /\* Ignore comments \*/ }

"=" { printf("%s", yytext); return EQUAL; }

\"[^\"\n]\*\" { printf("%s", yytext); sscanf(yytext, "%s", yylval.str); return STRING; }

[ \t\n\r\f]+ { printf("%s", yytext); /\* Ignore whitespace \*/ }

. { yyerror("Invalid character"); }

**%%**

void yyerror(const char \*msg) {

printf("\n");

fprintf(stderr, "\nError: %s.\nAt line: %d\n", msg,yylineno);

exit(1);

}

void check\_rb\_num(){

if(rb\_counter!=rb\_number)

{

char err\_msg[] = "Invalid number of RadioButton elements. They should be ";

char str\_rb\_num[5];

snprintf(str\_rb\_num, sizeof(str\_rb\_num), "%d", rb\_number);

strcat(err\_msg, str\_rb\_num);

yyerror(err\_msg);

}

rb\_counter = 0;

}

void check\_radio\_button\_id()

{

if(flag\_checked == 0 && strcmp(checked\_button\_id, "") != 0)

{

char err\_msg[] = "Invalid id in RadioButton elements. Exactly one id should match the id of android:checkedButton = ";

strcat(err\_msg, checked\_button\_id);

yyerror(err\_msg);

}

flag\_checked = 0;

}

int yywrap(void) {

return 1;

}

## parser.y:

%{

#include <stdio.h>

#include <stdlib.h>

#include <stdbool.h>

#include <string.h>

int yylex();

void yyerror(const char \*);

extern FILE \*yyin;

bool is\_numeric(char\* str);

void remove\_quotes(char\* str);

int string\_to\_int(char\* str);

void insert\_id(char \*str);

bool check\_id(char \*str);

typedef struct node {

char \*id;

struct node \*next;

} Node;

Node \*head = NULL;

int rb\_number;

char checked\_button\_id[100];

int flag\_checked = 0;

bool flag = false;

void checkPositiveInt(int integ, char attribute[]);

void checkRestrictions(int integ1, int integ2, char\* less, char\* more);

%}

%token LIN\_LAYOUT\_OPEN\_TAG GT LIN\_LAYOUT\_CLOSE\_TAG

%token RE\_LAYOUT\_OPEN\_TAG RE\_LAYOUT\_CLOSE\_TAG

%token LAYOUT\_WIDTH LAYOUT\_HEIGHT ID ORIENTATION EQUAL

%token STRING

%token TEXT\_OPEN\_TAG CLOSE\_TAG TEXT TEXT\_COLOR

%token IMAGE\_OPEN\_TAG SRC PADDING

%token BUTTON\_OPEN\_TAG R\_GROUP\_OPEN\_TAG R\_GROUP\_CLOSE\_TAG R\_BUTTON\_OPEN\_TAG CHECKED\_BUTTON RB\_NUMBER

%token PRO\_BAR\_OPEN\_TAG MAX PROGRESS

%token COMMENT

%union{

char str[20];

char str2[20];

}

%type <str> STRING

%locations

%start layout

**%%**

layout : lin\_layout

| re\_layout

;

lin\_layout : LIN\_LAYOUT\_OPEN\_TAG lin\_layout\_attr GT lin\_layout\_content LIN\_LAYOUT\_CLOSE\_TAG

;

lin\_layout\_attr : mandatory\_attr lin\_layout\_opt\_attr

;

mandatory\_attr : layout\_width\_attr layout\_height\_attr

| layout\_height\_attr layout\_width\_attr

;

lin\_layout\_opt\_attr : id\_attr ORIENTATION EQUAL STRING

| ORIENTATION EQUAL STRING id\_attr

| id\_attr

| ORIENTATION EQUAL STRING

| /\* empty \*/

;

id\_attr : ID EQUAL STRING

{

insert\_id($3);

if(flag == true && strcmp(checked\_button\_id, "") != 0)

{

if(strcmp($3, checked\_button\_id) == 0)

{

flag\_checked = 1;

}

}

flag = false;

}

;

layout\_width\_attr: LAYOUT\_WIDTH EQUAL STRING {

if(is\_numeric($3))

{

int pos\_int = string\_to\_int($3);

checkPositiveInt(pos\_int, "layout\_width");

}

else

{

if(!((strcmp($3, "\"match\_parent\"") == 0) || (strcmp($3, "\"wrap\_content\"") == 0))){

char err\_msg[] = "Invalid android:layout\_width = ";

strcat(err\_msg, $3);

yyerror(err\_msg);

}

}

}

;

layout\_height\_attr: LAYOUT\_HEIGHT EQUAL STRING{

if(is\_numeric($3))

{

int pos\_int = string\_to\_int($3);

checkPositiveInt(pos\_int, "layout\_height");

}

else

{

if(!((strcmp($3, "\"match\_parent\"") == 0) || (strcmp($3, "\"wrap\_content\"") == 0))){

char err\_msg[] = "Invalid android:layout\_height = ";

strcat(err\_msg, $3);

yyerror(err\_msg);

}

}

}

;

lin\_layout\_content : element

| lin\_layout\_content element

| /\*empty\*/

;

re\_layout : RE\_LAYOUT\_OPEN\_TAG re\_layout\_attr GT re\_layout\_content RE\_LAYOUT\_CLOSE\_TAG

;

re\_layout\_attr : mandatory\_attr re\_layout\_opt\_attr

;

re\_layout\_opt\_attr : id\_attr

| /\* empty \*/

;

re\_layout\_content : element

| re\_layout\_content element

| /\* empty \*/

;

element : lin\_layout

| re\_layout

| text\_view

| image\_view

| button

| radio\_group

| pro\_bar

;

text\_view : TEXT\_OPEN\_TAG text\_attr CLOSE\_TAG

;

text\_attr : text\_mandatory\_attr text\_opt\_attr

;

text\_mandatory\_attr : mandatory\_attr TEXT EQUAL STRING

| TEXT EQUAL STRING mandatory\_attr

| layout\_height\_attr TEXT EQUAL STRING layout\_width\_attr

| layout\_width\_attr STRING TEXT EQUAL STRING layout\_height\_attr

;

text\_opt\_attr : id\_attr TEXT\_COLOR EQUAL STRING

| TEXT\_COLOR EQUAL STRING id\_attr

| id\_attr

| TEXT\_COLOR EQUAL STRING

| /\* empty \*/

;

image\_view : IMAGE\_OPEN\_TAG image\_attr CLOSE\_TAG

;

image\_attr : image\_mandatory\_attr image\_and\_button\_opt\_attr

;

image\_mandatory\_attr : mandatory\_attr SRC EQUAL STRING

| SRC EQUAL STRING mandatory\_attr

| layout\_height\_attr SRC EQUAL STRING layout\_width\_attr

| layout\_width\_attr SRC EQUAL STRING layout\_height\_attr

;

image\_and\_button\_opt\_attr : id\_attr padding\_attr

| padding\_attr id\_attr

| id\_attr

| padding\_attr

| /\* empty \*/

;

padding\_attr : PADDING EQUAL STRING

{

int pos\_int = string\_to\_int($3);

checkPositiveInt(pos\_int, "padding");

}

;

button : BUTTON\_OPEN\_TAG button\_attr CLOSE\_TAG

;

button\_attr : button\_mandatory\_attr image\_and\_button\_opt\_attr

;

button\_mandatory\_attr : mandatory\_attr TEXT EQUAL STRING

| TEXT EQUAL STRING mandatory\_attr

| layout\_width\_attr TEXT EQUAL STRING layout\_height\_attr

| layout\_height\_attr TEXT EQUAL STRING layout\_width\_attr

;

radio\_group : R\_GROUP\_OPEN\_TAG r\_group\_attr GT r\_group\_content R\_GROUP\_CLOSE\_TAG

;

r\_group\_attr : r\_group\_mandatory\_attr r\_group\_opt\_attr

;

r\_group\_mandatory\_attr : mandatory\_attr rb\_number\_attr

| rb\_number\_attr mandatory\_attr

| layout\_height\_attr STRING rb\_number\_attr layout\_width\_attr

| layout\_width\_attr STRING rb\_number\_attr layout\_height\_attr

;

rb\_number\_attr : RB\_NUMBER EQUAL STRING

{

int pos\_int = string\_to\_int($3);

checkPositiveInt(pos\_int, "rb\_number");

rb\_number = pos\_int;

}

;

r\_group\_opt\_attr : id\_attr checked\_button\_attr

| checked\_button\_attr id\_attr

| id\_attr

| checked\_button\_attr

| /\* empty \*/

;

checked\_button\_attr : CHECKED\_BUTTON EQUAL STRING

{

strcpy(checked\_button\_id, $3);

}

r\_group\_content : radio\_button

| r\_group\_content radio\_button

| COMMENT

;

radio\_button : R\_BUTTON\_OPEN\_TAG radio\_button\_attr CLOSE\_TAG

;

radio\_button\_attr : radio\_button\_mandatory\_attr radio\_button\_opt\_attr

;

radio\_button\_mandatory\_attr : mandatory\_attr TEXT EQUAL STRING

| TEXT EQUAL STRING mandatory\_attr

| layout\_height\_attr TEXT EQUAL STRING layout\_width\_attr

| layout\_width\_attr TEXT EQUAL STRING layout\_height\_attr

;

radio\_button\_opt\_attr : { flag = true; } id\_attr

| /\* empty \*/

;

pro\_bar : PRO\_BAR\_OPEN\_TAG pro\_bar\_attr CLOSE\_TAG;

pro\_bar\_attr: mandatory\_attr pro\_bar\_opt\_attr

;

pro\_bar\_opt\_attr: id\_attr

| max\_attr

| progress\_attr

| id\_attr max\_attr

| max\_id\_attr

| id\_attr progress\_attr

| progress\_id\_attr

| max\_progress\_attr

| progress\_max\_attr

| all\_three\_ProgressBar\_attr

| /\* empty \*/

;

max\_attr: MAX EQUAL STRING { checkPositiveInt(string\_to\_int($3), "max"); }

;

progress\_attr: PROGRESS EQUAL STRING{ checkPositiveInt(string\_to\_int($3), "progress"); }

;

max\_id\_attr: MAX EQUAL STRING id\_attr{ checkPositiveInt(string\_to\_int($3), "max"); }

;

progress\_id\_attr: PROGRESS EQUAL STRING id\_attr{ checkPositiveInt(string\_to\_int($3), "progress"); }

max\_progress\_attr: MAX EQUAL STRING PROGRESS EQUAL STRING{

checkPositiveInt(string\_to\_int($3), "max");

checkPositiveInt(string\_to\_int($6), "progress");

checkRestrictions(string\_to\_int($3),string\_to\_int($6), "max", "progress");

}

;

progress\_max\_attr: PROGRESS EQUAL STRING MAX EQUAL STRING{

checkPositiveInt(string\_to\_int($3), "progress");

checkPositiveInt(string\_to\_int($6), "max");

checkRestrictions(string\_to\_int($6),string\_to\_int($3), "max", "progress");

}

;

all\_three\_ProgressBar\_attr: id\_attr max\_progress\_attr

| id\_attr progress\_max\_attr

| max\_progress\_attr id\_attr

| progress\_max\_attr id\_attr

| MAX EQUAL STRING id\_attr PROGRESS EQUAL STRING{

checkPositiveInt(string\_to\_int($3), "max");

checkPositiveInt(string\_to\_int($7), "progress");

checkRestrictions(string\_to\_int($3),string\_to\_int($7), "max", "progress");

}

| PROGRESS EQUAL STRING id\_attr MAX EQUAL STRING{

checkPositiveInt(string\_to\_int($7), "max");

checkPositiveInt(string\_to\_int($3), "progress");

checkRestrictions(string\_to\_int($7),string\_to\_int($3), "max", "progress");

}

;

**%%**

bool is\_numeric(char\* str) {

char local\_str[20];

strcpy(local\_str, str);

remove\_quotes(local\_str);

// Check the first character

int i = 0;

if (local\_str[0] == '-') {

// If the first character is a minus sign, move to the next character

i = 1;

}

// Check the remaining characters

while (local\_str[i] != '\0') {

// If any character is not a digit, return false

if (local\_str[i] < '0' || local\_str[i] > '9') {

return false;

}

i++;

}

// All characters are digits

return true;

}

void remove\_quotes(char\* str) {

int length = strlen(str);

// Check if the string is long enough to have quotes

if (length >= 2 && str[0] == '"' && str[length - 1] == '"') {

// Afairoume to arxiko quote kanontas shift mia thesi aristera olous tous xaraktires

for (int i = 0; i < length - 1; i++) {

str[i] = str[i + 1];

}

// Afairesi tou teliko quote thetontas stin thesi tou ton termatiko xaraktira

str[length - 2] = '\0';

}

}

int string\_to\_int(char\* str)

{

remove\_quotes(str);

return atoi(str);

}

void insert\_id(char \*str)

{

//Eisagvgi you id string stin lista

Node \*newNode = malloc(sizeof(Node));

newNode->id = malloc((strlen(str) + 1) \* sizeof(char));

strcpy(newNode->id, str);

newNode->next = head;

head = newNode;

bool id\_exists = false;

id\_exists = check\_id(str);

if(id\_exists)

{

char err\_msg[] = "Duplicate android:id ";

strcat(err\_msg, str);

yyerror(err\_msg);

}

}

bool check\_id(char \*str)

{

bool id\_exists = false;

Node \*current = head->next;

// diatrexw tin lista kai elegxw an to new id yphrxe

while (current != NULL) {

if(strcmp(current->id, str)==0)

id\_exists = true;

current = current->next;

}

return id\_exists;

}

void checkPositiveInt(int integ, char attribute[]){

if(integ<0){

char err\_msg[50] = "Invalid android:";

char str1[20];

sprintf(str1, "%d", integ); // Convert pos\_int into string

strcat(err\_msg, attribute);

strcat(err\_msg, " = \"");

strcat(err\_msg, str1);

strcat(err\_msg, "\"");

yyerror(err\_msg);

}

}

void checkRestrictions(int integ1, int integ2, char\* less, char\* more){

if(integ1<integ2){

char str[20];

char str2[20];

sprintf(str, "%d", integ1); // Convert pos\_int into string

sprintf(str2, "%d", integ2); // Convert pos\_int into string

char err\_msg[50] = "Invalid android:" ;

strcat(err\_msg, less);

strcat(err\_msg, " = \"");

strcat(err\_msg, str);

strcat(err\_msg, "\"");

strcat(err\_msg, " is less than android:");

strcat(err\_msg, more);

strcat(err\_msg, " = \"");

strcat(err\_msg, str2);

strcat(err\_msg, "\"");

yyerror(err\_msg);

}

}

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

if (argc < 2) {

fprintf(stderr, "Usage: %s <inpufile>\n", argv[0]);

return 1;

}

FILE \*input\_file = fopen(argv[1], "r");

char line[100];

if (!input\_file) {

perror("Failed to open input file");

return 1;

}

yyin = input\_file;

yyparse();

printf("\n\nThe file was succesfully parsed\n");

fclose(input\_file);

return 0;

}