

# Projektová dokumentácia

## Implementácia prekladača imperatívneho jazyka IFJ21

Tým 082, varianta II

7. decembra 2021

<b>Andrei Shchapaniak</b>	<b>(xshcha00)</b>	25 %
Andrej Binovsky	(xbinov00)	25 %
Zdenek Lapes	(xlapes02)	25 %
Richard Gajdosik	(xgajdo33)	25 %

# Obsah

<b>1</b>	<b>Úvod</b>	<b>1</b>
<b>2</b>	<b>Návrh a implementácia</b>	<b>1</b>
2.1	Lexikálna analýza . . . . .	1
2.2	Syntaktická analýza . . . . .	1
2.2.1	Zpracovanie výrazov pomocou precedenčnej syntaktickej analýzy . . . . .	1
2.3	Sémantická analýza . . . . .	1
2.4	Generovanie cieľového kódu . . . . .	1
2.4.1	Začiatok generovania . . . . .	1
2.4.2	Generovanie – funkcie . . . . .	1
2.4.3	Generovanie – výrazy . . . . .	1
2.4.4	Generovanie – podmienky a cykly . . . . .	1
2.5	Prekladový systém . . . . .	1
2.5.1	CMake . . . . .	1
2.5.2	GNU Make . . . . .	1
<b>3</b>	<b>Speciálne algoritmy a dátové štruktúry</b>	<b>1</b>
3.1	Tabuľka s rozptýlenými položkami . . . . .	1
3.2	Obojsmerný rad . . . . .	1
3.3	Zásobník . . . . .	1
<b>4</b>	<b>Práca v tímu</b>	<b>1</b>
4.1	Zpôsob práce v tímu . . . . .	1
4.1.1	Verzovací systém . . . . .	1
4.1.2	Komunikácia . . . . .	1
4.2	Rozdelenie práce medzi členmi tímu . . . . .	1
<b>5</b>	<b>Záver</b>	<b>1</b>

# **1 Úvod**

## **2 Návrh a implementácia**

### **2.1 Lexikálna analýza**

### **2.2 Syntaktická analýza**

#### **2.2.1 Zpracovanie výrazov pomocou precedenčnej syntaktickej analýzy**

### **2.3 Sémantická analýza**

### **2.4 Generovanie cieľového kódu**

#### **2.4.1 Začiatok generovania**

#### **2.4.2 Generovanie – funkcie**

#### **2.4.3 Generovanie – výrazy**

#### **2.4.4 Generovanie – podmienky a cykly**

### **2.5 Prekladový systém**

#### **2.5.1 CMake**

#### **2.5.2 GNU Make**

## **3 Speciálne algoritmy a dátové štruktúry**

### **3.1 Tabuľka s rozptýlenými položkami**

### **3.2 Obojsmerný rad**

### **3.3 Zásobník**

## **4 Práca v tímu**

### **4.1 Zpôsob práce v tímu**

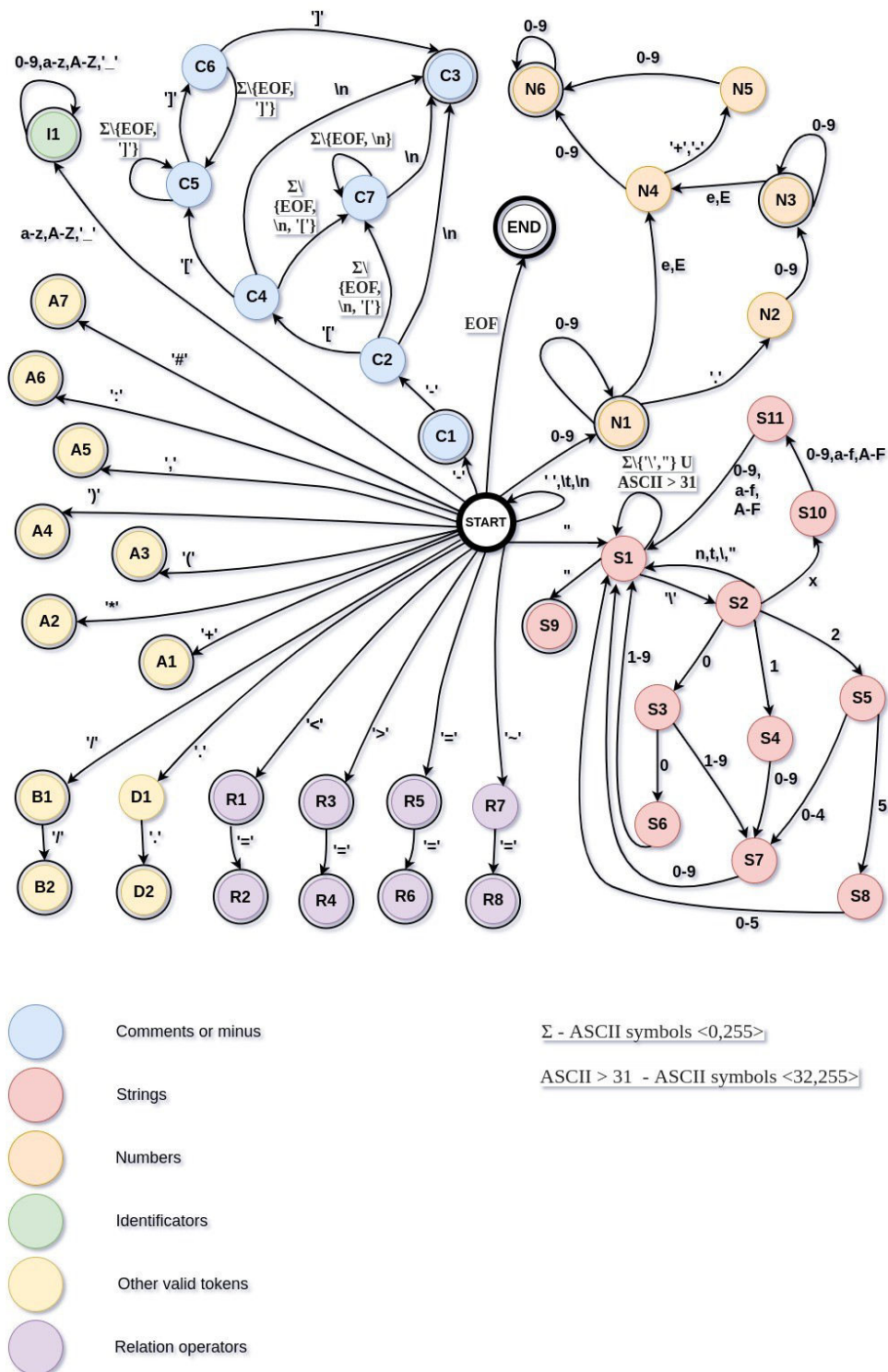
#### **4.1.1 Verzovací systém**

#### **4.1.2 Komunikácia**

### **4.2 Rozdelenie práce medzi členmi tímu**

## **5 Záver**

## Diagram konečného automatu specifikujúceho lexikálny analyzátor



Obr. 1: Diagram konečného automatu specifikujúci lexikálny analyzátor

## LL – gramatika

1.  $\langle \text{prolog} \rangle \rightarrow \text{require t\_string } \langle \text{prog} \rangle$
2.  $\langle \text{prog} \rangle \rightarrow \text{global id : function ( } \langle \text{arg\_T} \rangle \text{ ) } \langle \text{ret\_T} \rangle \langle \text{prog} \rangle$
3.  $\langle \text{prog} \rangle \rightarrow \text{function id ( } \langle \text{arg} \rangle \text{ ) } \langle \text{ret\_T} \rangle \langle \text{stmt} \rangle \text{ end } \langle \text{prog} \rangle$
4.  $\langle \text{prog} \rangle \rightarrow \text{id ( } \langle \text{param} \rangle \text{ ) } \langle \text{prog} \rangle$
5.  $\langle \text{prog} \rangle \rightarrow \text{EOF}$
6.  $\langle \text{arg\_T} \rangle \rightarrow \langle \text{type} \rangle \langle \text{next\_arg\_T} \rangle$
7.  $\langle \text{arg\_T} \rangle \rightarrow \varepsilon$
8.  $\langle \text{next\_arg\_T} \rangle \rightarrow , \langle \text{type} \rangle \langle \text{next\_arg\_T} \rangle$
9.  $\langle \text{next\_arg\_T} \rangle \rightarrow \varepsilon$
10.  $\langle \text{ret\_T} \rangle \rightarrow : \langle \text{type} \rangle \langle \text{next\_ret\_T} \rangle$
11.  $\langle \text{ret\_T} \rangle \rightarrow \varepsilon$
12.  $\langle \text{next\_ret\_T} \rangle \rightarrow , \langle \text{type} \rangle \langle \text{next\_ret\_T} \rangle$
13.  $\langle \text{next\_ret\_T} \rangle \rightarrow \varepsilon$
14.  $\langle \text{arg} \rangle \rightarrow \text{id : } \langle \text{type} \rangle \langle \text{next\_arg} \rangle$
15.  $\langle \text{arg} \rangle \rightarrow \varepsilon$
16.  $\langle \text{next\_arg} \rangle \rightarrow , \text{id : } \langle \text{type} \rangle \langle \text{next\_arg} \rangle$
17.  $\langle \text{next\_arg} \rangle \rightarrow \varepsilon$
18.  $\langle \text{type} \rangle \rightarrow \text{integer}$
19.  $\langle \text{type} \rangle \rightarrow \text{number}$
20.  $\langle \text{type} \rangle \rightarrow \text{string}$
21.  $\langle \text{type} \rangle \rightarrow \text{nil}$
22.  $\langle \text{stmt} \rangle \rightarrow \text{if } \langle \text{expr} \rangle \text{ then } \langle \text{stmt} \rangle \text{ else } \langle \text{stmt} \rangle \text{ end } \langle \text{stmt} \rangle$
23.  $\langle \text{stmt} \rangle \rightarrow \text{while } \langle \text{expr} \rangle \text{ do } \langle \text{stmt} \rangle \text{ end } \langle \text{stmt} \rangle$
24.  $\langle \text{stmt} \rangle \rightarrow \text{local id : } \langle \text{type} \rangle \langle \text{def\_var} \rangle \langle \text{stmt} \rangle$
25.  $\langle \text{stmt} \rangle \rightarrow \text{return } \langle \text{expr} \rangle \langle \text{next\_expr} \rangle \langle \text{stmt} \rangle$
26.  $\langle \text{stmt} \rangle \rightarrow \text{id } \langle \text{fork\_id} \rangle \langle \text{stmt} \rangle$
27.  $\langle \text{stmt} \rangle \rightarrow \varepsilon$
28.  $\langle \text{def\_var} \rangle \rightarrow = \langle \text{one\_assign} \rangle$
29.  $\langle \text{def\_var} \rangle \rightarrow \varepsilon$
30.  $\langle \text{one\_assign} \rangle \rightarrow \text{id ( } \langle \text{param} \rangle \text{ ) }$

31.  $\langle \text{one\_assign} \rangle \rightarrow \langle \text{expr} \rangle$
32.  $\langle \text{param} \rangle \rightarrow \langle \text{param\_val} \rangle \langle \text{next\_param} \rangle$
33.  $\langle \text{param} \rangle \rightarrow \varepsilon$
34.  $\langle \text{param\_val} \rangle \rightarrow \text{id}$
35.  $\langle \text{param\_val} \rangle \rightarrow \langle \text{term} \rangle$
36.  $\langle \text{term} \rangle \rightarrow \text{t\_string}$
37.  $\langle \text{term} \rangle \rightarrow \text{t\_integer}$
38.  $\langle \text{term} \rangle \rightarrow \text{t\_number}$
39.  $\langle \text{term} \rangle \rightarrow \text{nil}$
40.  $\langle \text{next\_param} \rangle \rightarrow , \langle \text{param\_val} \rangle \langle \text{next\_param} \rangle$
41.  $\langle \text{next\_param} \rangle \rightarrow \varepsilon$
42.  $\langle \text{next\_expr} \rangle \rightarrow , \langle \text{expr} \rangle \langle \text{next\_expr} \rangle$
43.  $\langle \text{next\_expr} \rangle \rightarrow \varepsilon$
44.  $\langle \text{fork\_id} \rangle \rightarrow ( \langle \text{param} \rangle )$
45.  $\langle \text{fork\_id} \rangle \rightarrow \langle \text{next\_id} \rangle$
46.  $\langle \text{next\_id} \rangle \rightarrow , \text{id} \langle \text{next\_id} \rangle$
47.  $\langle \text{next\_id} \rangle \rightarrow = \langle \text{mult\_assign} \rangle$
48.  $\langle \text{mult\_assign} \rangle \rightarrow \text{id} ( \langle \text{param} \rangle )$
49.  $\langle \text{mult\_assign} \rangle \rightarrow \langle \text{expr} \rangle \langle \text{next\_expr} \rangle$

## LL – tabulka

	require	global	function	id	integer	string	number	nil	t_integer	t_number	t_string	if	while	local	return	:	=	(	,	EOF	\$
<prolog>	1																				
<prog>		2	3	4																5	
<arg_T>					6	6	6	6													7
<next_arg_T>																			8		9
<ret_T>																10					11
<next_ret_T>																			12		13
<arg>				14																	15
<next_arg>																			16		17
<type>					18	20	19	21													
<stmt>				26								22	23	24	25						27
<def_var>																	28				29
<one_assign>				30																	31
<param>				32				32	32	32	32										33
<param_val>				34				35	35	35	35										
<term>								39	37	38	36										
<next_param>																			40		41
<next_expr>																			42		43
<fork_id>																	45	44	45		
<next_id>																	47		46		
<mult_assign>				48																	49

## Precedenčná tabuľka

1.  $E \rightarrow i$

2.  $E \rightarrow ( E )$

3.  $E \rightarrow \# E$

4.  $E \rightarrow E + E$

5.  $E \rightarrow E - E$

6.  $E \rightarrow E * E$

7.  $E \rightarrow E / E$

8.  $E \rightarrow E // E$

9.  $E \rightarrow E .. E$

10.  $E \rightarrow E > E$

11.  $E \rightarrow E < E$

12.  $E \rightarrow E \geq E$

13.  $E \rightarrow E \leq E$

14.  $E \rightarrow E == E$

15.  $E \rightarrow E \sim E$

	#	*	/	//	+	-	..	<	<=	>	>=	==	~=	(	)	i	\$
#	<	>	>	>	>	>	>	>	>	>	>	>	>	<	>	<	>
*	<	>	>	>	>	>	>	>	>	>	>	>	>	<	>	<	>
/	<	>	>	>	>	>	>	>	>	>	>	>	>	<	>	<	>
//	<	>	>	>	>	>	>	>	>	>	>	>	>	<	>	<	>
+	<	<	<	<	>	>	>	>	>	>	>	>	>	<	>	<	>
-	<	<	<	<	>	>	>	>	>	>	>	>	>	<	>	<	>
..	<	<	<	<	<	<	<	>	>	>	>	>	>	<	>	<	>
<	<	<	<	<	<	<	<	>	>	>	>	>	>	<	>	<	>
<=	<	<	<	<	<	<	<	>	>	>	>	>	>	<	>	<	>
>	<	<	<	<	<	<	<	>	>	>	>	>	>	<	>	<	>
>=	<	<	<	<	<	<	<	>	>	>	>	>	>	<	>	<	>
==	<	<	<	<	<	<	<	>	>	>	>	>	>	<	>	<	>
~=	<	<	<	<	<	<	<	>	>	>	>	>	>	<	>	<	>
(	<	<	<	<	<	<	<	<	<	<	<	<	<	<	=	<	e
)	>	>	>	>	>	>	>	>	>	>	>	>	>	e	>	s	>
i	e	>	>	>	>	>	>	>	>	>	>	>	>	e	>	s	>
\$	<	<	<	<	<	<	<	<	<	<	<	<	<	<	e	<	e

### LEGENDA:

< - insert to stack with shift

> - reduction

= - insert to stack

e - error

s - special case (end of expression)