Dokumentace projektu z předmětů IFJ a IAL

Implementace překladače imperativního jazyka IFJ19

Tým 057, varianta I

11. 12. 2019

Martin Koči (xkocim05) 30 %
Magdaléna Ondrušková (xondru14) 20 %
Michal Koval (xkoval17) 30 %
Zuzana Hradilova (xhradi16) 20 %

Obsah

1 Úvod **2**

2 Implementace **2**
2.1 Lexikální analýza 2
2.2 Tabulka symbolů 2
2.3 Syntaktická analýza 2
2.3.1 Zpracování výrazů 2
2.4 Sémantická analýza 2
2.5 Generování kódu 2

3 Práce v týmu **2**
3.1 Deadliny a pokusná odevzdání 3
3.2 Rozdělení práce 3

4 Diagram konečného automatu pro lexikální analýzu **4**

5 LL Tabulka **5**

6 Precedenční tabulka **6**

1 Úvod

Cílem projektu bylo vytvořit překladač ze zdrojového jazyka IFJ19, založeného na programovacím jazyce Python, do jazyka IFJcode19. Program byl implementován v jazyce C.

2 Implementace

Projekt je rozdělen do několika částí, které byli implementovány samostatně.

2.1 Lexikální analýza

2.2 Tabulka symbolů

Tabulka symbolů slouží k zaznamenávání dat o proměnných, funkcích a jejich parametrech. Byla implementována jako binární vyhledávací strom, o kterém jsme se učili v předmětu IAL.

Proměnné v IFJ19 jsou rozdělené na globální, definované v hlavním těle programu a lokální, definované ve funkci. Vzhledem k potřebě uchovávat různá data o globálních a lokálních symbolech byli funkce pro práci s tabulkami rozděleny.

K prvkům v tabulce se přistupovalo pomocí unikátního klíče (id), pomocí nějž bylo možné symboly vyhledávat a měnit obsah uložených dat o dané proměnné.

Při vložení globálního symbolu, který byl funkcí, byli taktéž uloženy data o jejích parametrech a ukazatel na seznam parametrů, který byl implementován pomocí dvousměrně vázaného lineárního seznamu, taktéž probraného v předmětu IAL.

Funkce pro práci se symboly, definované v tabulce symbolů, byly následně použity při syntaktické analýze a generovaní kódu, především pro kontrolu datových typů a vlastností funkcí (např byla-li použitá funkce někde v programu definována, ověření počtu parametru, atd.)

2.3 Syntaktická analýza

2.3.1 Zpracování výrazů

Výrazy zpracováváme odděleně od ostatcích částí syntaktické analýzy, v souboru `expression_parser.c`. Analýza výrazů voláme z hlavní části syntaktické analýzy pomocí funkce `callExpression`. Tato funkce dostane jako jediný parametr 1. token výrazu. V případě chyby funkce navrátí chybový kód, pokud je kontrolovaný výraz správný je zpátky do syntaktické analýzy vráceno OK.

Samotné zpracování výrazu probíhá pomocí precedenční analýzy. Je načten token, který je podle pravidel precedenční tabulky (viz. Tabulka 3 na str. 6) vložen na zásobník.

2.4 Sémantická analýza

2.5 Generování kódu

3 Práce v týmu

Na projektu jsme pracovali jsme ve čtyřčlenném týmu. Po zveřejnění zadaní jsme se sešli na 1. týmové schůzce, ujasnili si, co všechno bude potřeba udělat a rozdělili si úkoly.
Na většině částech projektu jsme pracovali samostatně nebo ve dvojicích, ale často jsme se potkávali a společně konzultovali řešení. Komunikace probíhala také přes Discord a Messenger.
Zdrojové soubory jsme sdíleli pomocí GitHubu, což nám umožnilo jejich rychlé sdílení a složení výsledného projektu.

3.1 Deadliny a pokusná odevzdání

Pro každý úkol jsme si určili termín, do kterého jsme se snažili o jeho dokončení. Dodržet tyto termíny nám velmi pomohli i možnosti vyzkoušet si odevzdání projektu pomocí pokusného odevzdání. Před prvním pokusném odevzdání jsme zvládli dokončit lexikální, syntaktickou a většinu sémantické analýzy a začít s generováním kódu. Mezi prvním a druhým odevzdáním jsme se zaměřili na generování kódu, odchycení běhových chyb a testování. Později jsme náš projekt dolaďovali a opravovali chyby. Tyto možnosti vyzkoušet si projekty odevzdat „nanečisto“ pro nás byli velmi užitečné a „donutili“ nás nenechat řešení projektu na poslední chvíli.

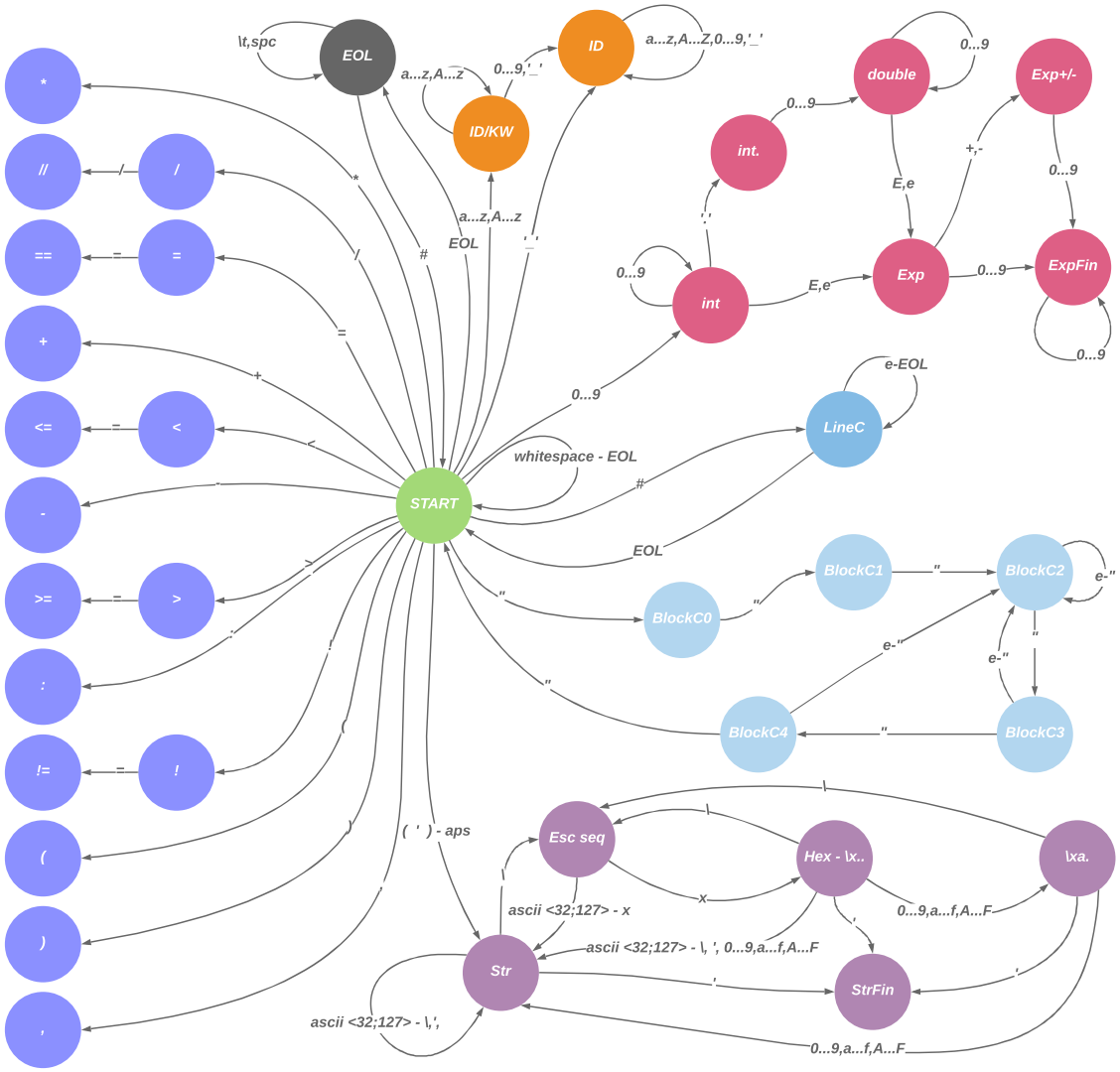
3.2 Rozdělení práce

Jednotlivé části projektu byly rozděleny podle následující tabulky. Rozhodli jsme se rozdělit procenta podle vykonané práce jednotlivých členů týmu, rozsahu napsaného kódu a času stráveného řešením projektu.

Martin Koči	Vedoucí týmu, syntaktická analýza, sémantická analýza	30%
Magdaléna Ondrušková	Syntaktická analýza, zpracování výrazů, testování	20%
Zuzana Hradilová	Tabulka symbolu, testování, dokumentace	20%
Michal Koval	Lexikální analýza, generování kódu	30%

Tabulka 1: Rozdělení práce v týmu

4 Diagram konečného automatu pro lexikální analýzu



Obrázek 1: Návrh konečného automatu

5 LL Tabulka

	eof	eol	def	id	(	:	indent	=	return	pass	while	if	else	,	)	none	float	string	int	dedent	expr
<prog>	1		1	1						1	1	1									1
<st-list>	3		2	2						2	2	2									2
<stat>			4	5						7	8	9									6
<params>				11											10						
<nested-st-list>				32						32	32	32									32
<assign>				43																	42
<next-param>														12	13						
<arg-params>				15											14	15	15	15	15		
<value>				22												18	19	20	21		
<arg-next-params>														16	17						
<nested-stat>				37						33	36	34									35
<func-nested-st-list>				23					23	23	23	23									23
<func-nested-stat>				28					29	24	27	25									26
<def-id>		45			44																
<after-id>		41			41				40												
<next-func-nested-st-list>				31					31	31	31	31								30	31
<nex-nested-st-list>				39						39	39	39								38	39
<after-return>		47																			46
<eof-or-eol>	49	48																			

Tabulka 2: LL tabulka

6 Precedenční tabulka

	+	-	*	/	//	<	<=	>	>=	==	!=	(	)	id	int	float	string	none	$
+	>	>	<	<	<	>	>	>	>	>	>	<	>	<	<	<	<		>
-	>	>	<	<	<	>	>	>	>	>	>	<	>	<	<	<			>
*	>	>	>	>	>	>	>	>	>	>	>	<	>	<	<	<			>
/	>	>	>	>	>	>	>	>	>	>	>	<	>	<	<	<			>
//	>	>	>	>	>	>	>	>	>	>	>	<	>	<	<	<			>
<	<	<	<	<	<							<	>	<	<	<	<		>
<=	<	<	<	<	<							<	>	<	<	<	<		>
>	<	<	<	<	<							<	>	<	<	<	<		>
>=	<	<	<	<	<							<	>	<	<	<	<		>
==	<	<	<	<	<							<	>	<	<	<	<	<	>
!=	<	<	<	<	<							<	>	<	<	<	<	<	>
(	<	<	<	<	<	<	<	<	<	<	<	<	=	<	<	<	<	<	
)	>	>	>	>	>	>	>	>	>	>	>		>						>
id	>	>	>	>	>	>	>	>	>	>	>		>						>
int	>	>	>	>	>	>	>	>	>	>	>		>						>
float	>	>	>	>	>	>	>	>	>	>	>		>						>
string	>					>	>	>	>	>	>		>						>
none										>	>		>						>
$	<	<	<	<	<	<	<	<	<	<	<	<		<	<	<	<	<	

Tabulka 3: Precedenční tabulka k analýze výrazů