Dokumentace projektu

Implementace překladače jazyka IFJ25

Tým xmetelm00, varianta vv-BVS

Martin Metelka	**xmetelm00**	25 %
Vojtěch Kabelka	xkabelv00	25 %
Filip Kachyňa	xkachyf00	25 %
Jakub Gono	xgonoja00	25 %

3. prosince 2025

1 Úvod

Cílem projektu bylo implementovat překladač jazyka IFJ25, který je inspirovaný jazykem Wren, v jazyce C.

2 Návrh a implementace

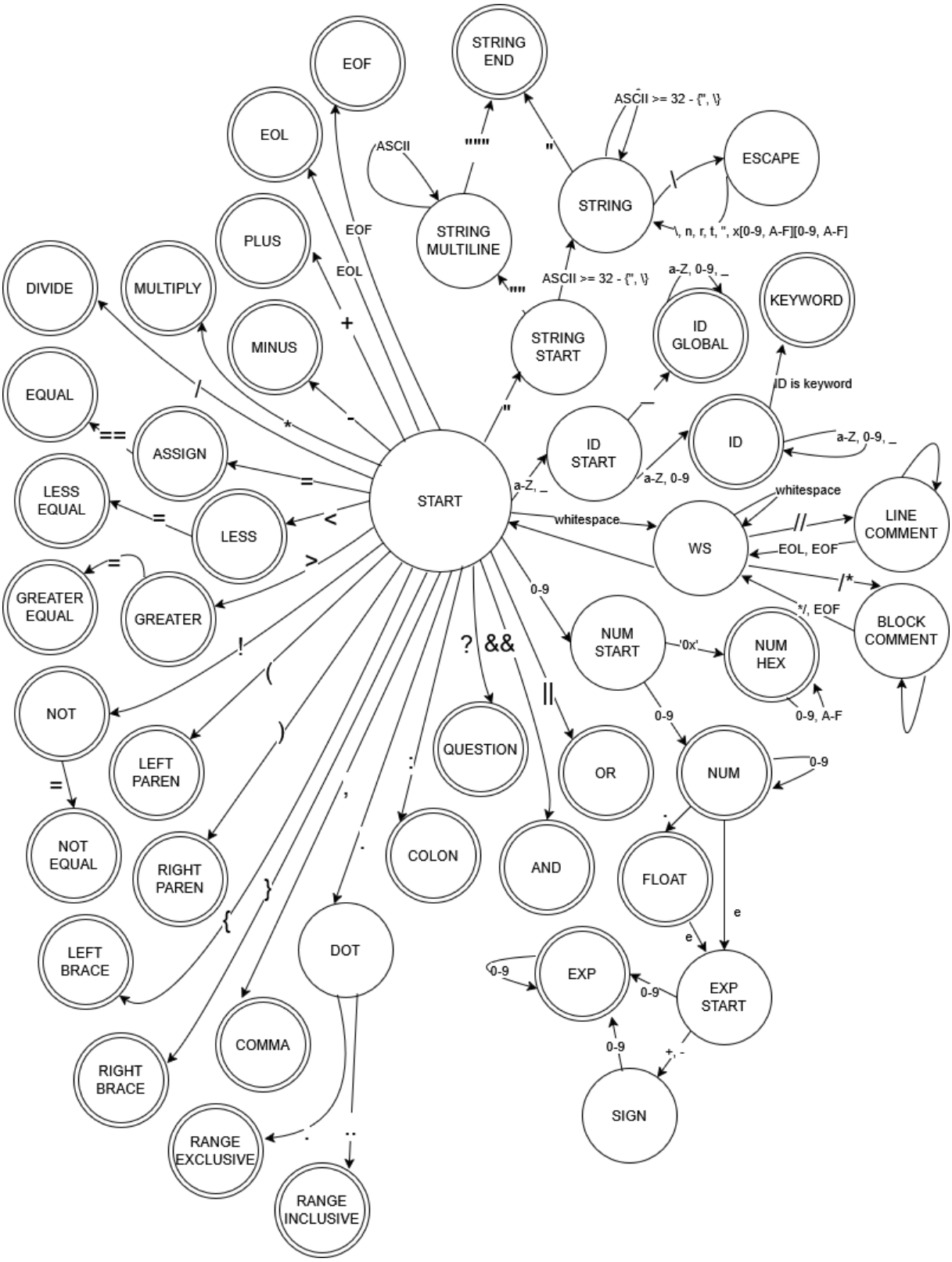
2.1 Lexikální analýza

Lexikální analýzu jsme implementovali ve zdrojovém souboru *scanner.c* a hlavičkovém souboru *scanner.h*, ve kterém jsou podpůrné struktury.

Hlavní funkcí lexikálního analyzátoru je funkce *get_next_token*, volaná parserem. Většinu funkce tvoří switch, pomocí kterého je implementován konečný stavový automat. Funkce pro čtení řetězců, identifikátorů nebo čísel stojí samostatně mimo switch. Podle přečteného znaku analyzátor rozhodne, do jakého stavu přejít. Stavy identifikují lexém a parseru vrací strukturu *Token* uchovávající informace jako je typ lexému, hodnota, číslo řádku a pozici na řádku. V případě, že má lexém chybný tvar, je vrácen token typu *TOKEN_ERROR*. Analyzátor si uchovává v struktuře *Scanner* informaci o vstupním souboru, poslední přečtený znak, číslo řádku a pozici na něm, a informaci, jestli narazil na konec řádku.

Funkce *get_next_token* používá pomocnou funkci *skip_whitespace*, která přeskočí bílé znaky a případné komentáře. Dalšími důležitými funkcemi jsou *peek* a *peek2*, které přečtou následující jeden nebo dva znaky a vrátí se zpět na původní pozici.

Za uvolnění tokenů zodpovídá volající, tedy parser.



Konečný stavový automat lexikálního analyzátoru

2.2 Parser

Parser je hlavní částí implementace překladače. Pracuje jako syntaktický analyzátor, sémantický analyzátor a generátor kódu. Parser pracuje formou rekurzivního sestupu a interaguje s lexikálním analyzátorem a scannerem.

	+	-	*	/	(	)	<	>	<=	>=	==	!=	is	id	num	string	type	$
+	>	>	<	<	<	>	>	>	>	>	>	>	>	<	<	<		>
-	>	>	<	<	<	>	>	>	>	>	>	>	>	<	<			>
*	>	>	>	>	<	>	>	>	>	>	>	>	>	<	<			>
/	>	>	>	>	<	>	>	>	>	>	>	>	>	<	<			>
(	<	<	<	<	<	=	<	<	<	<	<	<	<	<	<	<		
)	>	>	>	>		>	>	>	>	>	>	>	>					>
<	<	<	<	<	<	>	>	>	>	>	>	>	>	<	<			>
>	<	<	<	<	<	>	>	>	>	>	>	>	>	<	<			>
<=	<	<	<	<	<	>	>	>	>	>	>	>	>	<	<			>
>=	<	<	<	<	<	>	>	>	>	>	>	>	>	<	<			>
==	<	<	<	<	<	>	<	<	<	<	>	>	<	<	<	<		>
!=	<	<	<	<	<	>	<	<	<	<	>	>	<	<	<	<		>
is	<	<	<	<		>	<	<	<	<	>	>	>				<	>
id	>	>	>	>		>	>	>	>	>	>	>						>
num	>	>	>	>		>	>	>	>	>	>	>						>
string	>		>			>					>	>						>
type						>												>
$	<	<	<	<	<		<	<	<	<	<	<	<	<	<	<		

Precedenční tabulka

2.3 Tabulka symbolů

Tabulku symbolů jsme implementovali jako výškově vyvážený binární vyhledávací strom. Pro proměnné má typy nulová hodnota, číslo, řetězec, bool a nedefinovaná. Pro funkce, gettery a settery ukládá parametry a jejich počet. Strom je po každé úpravě kontrolován a balancován.

3 Práce v týmu

Spolupráce na projektu probíhala přes github a Discord.

3.1 Rozdělení práce v týmu

Martin Metelka	-	Scanner a parser
Vojtěch Kabelka	-	Tabulka symbolů
Filip Kachyňa	-	Dokumentace
Jakub Gono	-	Debuggování