

# VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

## FAKULTA INFORMAČNÍCH TECHNOLOGIÍ



Dokumentace projektu do předmětu IFJ a IAL

## **Interpret jazyka IFJ15**

Tým 043, varianta b/2/I

6. prosince 2015

Martin Honza (xhonza03)  
Patrik Jurnečka (xjurne03)  
Hana Slámová (xslamo00)  
Frantisek Šumšal (xsumsa01)  
Adam Švidroň (xsvidr00)

# Obsah

<b>1</b>	<b>Úvod</b>	<b>2</b>
<b>2</b>	<b>Struktura projektu</b>	<b>2</b>
<b>3</b>	<b>Lexikální analyzátor (scanner)</b>	<b>2</b>
3.1	Konečný automat . . . . .	2
<b>4</b>	<b>Syntaktická analyzátor (parser)</b>	<b>3</b>
4.1	LL gramatika . . . . .	3
4.2	Precedenční syntaktická analýza . . . . .	3
<b>5</b>	<b>Sémantický analyzátor</b>	<b>4</b>
<b>6</b>	<b>Interpret</b>	<b>4</b>
<b>7</b>	<b>Chybové stavy</b>	<b>4</b>
<b>8</b>	<b>Algoritmy z předmětu IAL</b>	<b>5</b>
8.1	Heapsort . . . . .	5
8.2	Boyer-Mooreův algoritmus . . . . .	5
8.3	Binární vyhledávací strom (BVS) . . . . .	5
<b>9</b>	<b>Práce v týmu</b>	<b>6</b>
9.1	Rozdělení práce . . . . .	6
<b>10</b>	<b>Závěr</b>	<b>6</b>
10.1	Metriky kódu . . . . .	6



## 4 Syntaktická analýzátor (parser)

Syntaktický analyzátor postupně volá tokeny z lexikálního analyzátoru. Tokeny jsou zpracovávány dvěma způsoby. Pomocí LL gramatiky (Tabulka 1) a precedenční syntaktické analýzy (Tabulka 2). Při úspěšném dokončení syntaktické analýzy se generuje instrukční páska pro interpret.

### 4.1 LL gramatika

LL gramatika postupně ověřuje příchozí tokeny.

1.  $\langle program \rangle \rightarrow \langle declrList \rangle EOF$
2.  $\langle declrList \rangle \rightarrow \langle funcDeclr \rangle \langle declrList \rangle$
3.  $\langle declrList \rangle \rightarrow \langle empty \rangle$
4.  $\langle funcDeclr \rangle \rightarrow \langle typeSpec \rangle ID (\langle params \rangle)$
5.  $\langle typeSpec \rangle \rightarrow INT$
6.  $\langle typeSpec \rangle \rightarrow DOUBLE$
7.  $\langle typeSpec \rangle \rightarrow STRING$
8.  $\langle params \rangle \rightarrow \langle paramItem \rangle$
9.  $\langle params \rangle \rightarrow \langle paramItem \rangle, \langle params \rangle$
10.  $\langle paramItem \rangle \rightarrow \langle typeSpec \rangle ID$
11.  $\langle paramItem \rangle \rightarrow \langle empty \rangle$

Tabulka 1: LL gramatika

### 4.2 Precedenční syntaktická analýza

Syntaktická analýza zdola nahoru. Dle Zadání jsem analýzu implementovali za pomoci precedenční tabulky, která nám pomáhá ošetřit vstupní výrazy.

	+	-	*	/	<	>	<=	>=	==	!=	(	)	id	\$
+	>	>	<	<	>	>	>	>	>	>	<	>	<	>
-	>	>	<	<	>	>	>	>	>	>	<	>	<	>
*	>	>	>	>	>	>	>	>	>	>	<	>	<	>
/	>	>	>	>	>	>	>	>	>	>	<	>	<	>
<	<	<	<	<	<	>	>	>	>	>	<	>	<	>
>	<	<	<	<	<	>	>	>	>	>	<	>	<	>
<=	<	<	<	<	<	>	>	>	>	>	<	>	<	>
>=	<	<	<	<	<	>	>	>	>	>	<	>	<	>
==	<	<	<	<	<	>	>	>	>	>	<	>	<	>
!=	<	<	<	<	>	>	>	>	>	>	<	>	<	>
(	<	<	<	<	<	<	<	<	<	<	<	=	<	
)	>	>	>	>	>	>	>	>	>	>		>		>
id	>	>	>	>	>	>	>	>	>	>		>		>
\$	<	<	<	<	<	<	<	<	<	<	<		<	

Tabulka 2: Precedenční tabulka syntaktické analýzy výrazů

## **5 Sémantický analyzátor**

## **6 Interpret**

Poslední částí Interpretu je vlastní překlad, který nastane pouze pokud zdrojový kód úspěšně projde syntaktickou a sémantickou analýzou. Interpretační část má za úkol zpracovat instrukční sadu založenou na 3AC (tří adresné instrukce).

## **7 Chybové stavy**

Pro chybové stavy jsme implementovali vlastní funkci, která je vždy volána při nalezení chyby v kódu, jak syntaktické, sémantické nebo při interpretaci. Návrátové chybové kódy byly definovány v zadání projektu.

## 8 Algoritmy z předmětu IAL

### 8.1 Heapsort

Heapsort neboli „řazení hromadou“ je řadící metoda s lineární složitostí. Algoritmus patří mezi nestabilní algoritmy, protože přesouvá prvky s příliš velkými skoky a také se nechová přirozeně.

### 8.2 Boyer-Mooreův algoritmus

Dle zadání jsem měli vestavěnou funkci **find**, která by používala pro vyhledání Boyer-Mooreův algoritmus. Boyer-Mooreův algoritmus je jedním velmi efektivním algoritmem pro vyhledávání podřetězců v řetězcích. Algoritmus prochází textem a porovnává vzorek zprava doleva a pro posuv bere ten z výsledků dvou heuristik, který je výhodnější. Jeho efektivita spočívá v časové složitosti. Čím delší je vyhledávaný podřetězec, tím je kratší doba. Funkce je implementována podle skript z předmětu IAL.

- ComputerJumps()

Stanovení hodnot pole *CharJump*, které určují posuv vzorku. Použití pole *CharJump* pro postup vzorku činí tento algoritmus mnohem rychlejší než algoritmus Knuth-Morris-Prattův.

Jednotlivé symboly řetězce mají uloženou svojí ASCII hodnotu.

- ComputerMatchJumps()

Funkce pro samotný výpočet pole *MatchJump*.

- Boyer\_Moor\_Alg()

Stanovení indexu polohy vzorku nalezeného v textu.

### 8.3 Binární vyhledávací strom (BVS)

Nejpoužívanější implementace pro dynamické vyhledávací tabulky. Algoritmus je implementován rekurzivně, dle opory k předmětu IAL (algoritmy). Vyhledávání v BVS je velmi podobné binárnímu vyhledávání v seřazeném poli.

Vytvořili jsme vlastní knihovnu funkcí. Implementace knihovny je v souborech *bst.c* a *bst.h*.

Rozhraní *bst* poskytuje funkčnost specifickou pro tabulku symbolů, která je implementována v souborech *stable.c* a *stable.h*.

## 9 Práce v týmu

Komunikace v týmu probíhala za pomoci IRC Freenode. Pravidelná osobní setkání se uskutečňovalo jednou za čtrnáct dní nebo po domluvě v CVT. Na osobním setkání bylo objasněno, co vše je zapotřebí udělat a také provedeno rovnoměrné rozdělení práce.

Ke sdílení zdrojových kódů jsme využili GitHub s vlastním soukromým kanálem.

### 9.1 Rozdělení práce

- **Martin Honza:** Precedenční analýza výrazu
- **Patrik Jurnečka:** Dokumentace, prezentace, interpret
- **Hana Slámová:** Vestavěné funkce
- **Frantisek Šumšal:** Lexikální analyzátor, syntaktický analyzátor
- **Adam Švidroň:** Heap sort, interpret

## 10 Závěr

Na projektu se pracovalo téměř dva měsíce, i když jsme začali včas, dokončit projekt jsme nestíhali do pokusného odevzdání. V posledním týdnu před odevzdáním jsme dodělávali interpret a snažili se vše zkompletovat, aby fungoval projekt dle zadání. Dokumentace se tvořila průběžně při dokončení některého logického celku.

Projekt byl velmi náročný, nejen ze strany programové, ale také hlavní roli hrála domluva týmu. Nakonec se to zvládlo dokončit. Z projektu jsme si odnesli velké zkušenosti, jak efektivně organizovat a pracovat v týmu.

Navržená implementace byla nakonec otestována v prostředí operačních systémů Linux a MS Windows 10. Dokumentace i prezentace jsme psali v jazyce  $\text{\LaTeX}$ .

### 10.1 Metriky kódu

- Počet souborů: ...
- Počet řádků zdrojového kódu: ...
- Velikost spustitelného souboru: ... (OS Linux 32 bitová architektura)

## Reference

- [1] AHO, A. V. *Compilers: principles, techniques,*. 2nd ed. Boston: Pearson/Addison Wesley, c2007. ISBN 03-214-8681-1.