

**Dokumentace k projektu pro předměty IFJ a IAL**

**Implementace interpretu imperativního jazyka IFJ13**

Tým 005

Varianta a/3/I

13. prosince 2013

Řešitelé: Jan Hanák xhanak27 20% **vedoucí**

Michal Flax xflaxm00 20%

Cuong Duong Tuan xduong00 20%

Jakub Fišer xfiser10 20%

Vladimír Faltýn xfalty06 20%.

Obsah

[1 Úvod 3](#_Toc374710806)

[2 Zadání 3](#_Toc374710807)

[2.1 Varianta zadání 3](#_Toc374710808)

[3 Práce v týmu 3](#_Toc374710809)

[4 Implementace 4](#_Toc374710810)

[4.1 Knuth-Moris-Prattův algoritmus 4](#_Toc374710811)

[4.2 Shell sort 4](#_Toc374710812)

[4.3 Tabulka symbolů 4](#_Toc374710813)

[4.4 Lexikální analýza 4](#_Toc374710814)

[4.5 Syntaktická analýza 4](#_Toc374710815)

[4.6 Sémantická analýza 4](#_Toc374710816)

[4.7 Generátor tříadresného kódu 4](#_Toc374710817)

[5 Závěr 6](#_Toc374710818)

[6 Přílohy 7](#_Toc374710819)

[6.1 Konečný automat lexikálního analyzátoru 7](#_Toc374710820)

[6.2 LL gramatika 8](#_Toc374710821)

1 Úvod

Tato dokumentace popisuje implementaci interpretu pro imperativní jazyk IFJ13. Dokumentace je členěna do několika kapitol popisující konkrétní části intepretu.

Přiložen je graf konečného automatu specifikující lexikální analyzátor a LL-gramatika pro syntaktický analyzátor.

2 Zadání

Jazyk IFJ13 je podmnožinou skriptovacího jazyka PHP. Jazyk je case-sensitive a je jazykem dynamicky typovaným. Zadáním je vytvořit program, který načte zdrojový soubor v jazyce IFJ13 a interpretuje jej, přičemž vrací návratovou hodnotu podle toho, jak proběhla činnost intepretu.

2.1 Varianta zadání

Variantou zadání byla a/3/I, to konkrétně znamená:

- na věstavenou funkci *find\_string()* je použitý Knuth-Moris-Prattův algoritmus

- věstavěná funkce *sort\_string()* je implementována metodou shell sort

- tabulka symbolů je implementována pomocí binárního vyhledávacího stromu

3 Práce v týmu

Docházelo k nepravidelným schůzkám, práce se rozdělovala na několik částí. Jako první se napsal lexikální analyzátor, poté se navrhla LL-gramatika pro syntaktický analyzátor, který byl záhy napsán. Generátor tříadresného kódu byl napsaný před sémantickou analýzou, která byla napsána jako poslední. Časový plán kódování nebyl, spoléhalo se na svědomitý přístup všech členů týmu.

Kód byl psán jak v Linuxu, tak Windows, přičemž pro testování jsme využívali školní server Merlin. Ke sdílení kódu jsme využili repozitáře GIT a pro vzájemnou komunikaci jsme použili skupinový chat v síti Facebook.

4 Implementace

4.1 Knuth-Moris-Prattův algoritmus

Věstavená funkce *find\_string()* využívá Knuth-Moris-Prattův algoritmus, který je založen na úvaze, že časové ztráty triviálního algoritmu jsou způsobeny tím, že v případě neshody vyhledávaného podřetězce v řetězci se podřetězec posune o jednu pozici doprava a znovu se celý zkontroluje, je tedy nutné se často při porovnání vracet zpět k pozicích, jež byly dříve zkontrolovány. Snahou tohohle algoritmu je v případě neshody hledaného podřetězce v řetězci posunout podřetězec tak, aby se nebylo nutné vracet ve vlastním řetězci. Jeho maximální složitost je O(m+n).

4.2 Shell sort

Věstavená řadící funkce *sort\_string()* je založena na Shell sortu, který patří mezi nestabilní řadící metody. Je založený na Bubble sortu. V několika průchodech se řadí prvky ve stejné vzdálenosti od sebe pomocí Bubble sortu, vzdálenost se postupně snižuje, obecně se začíná na n/2, která se nadále snižuje dělením 2. Výhodou je např. to, že narozdíl od Heap sortu není třeba halda a narozdíl od Quick sortu není potřeba zásobník.

4.3 Tabulka symbolů

Tabulka symbolů byla implementována pomocí binárního vyhledávacího stromu.

// doplnit

4.4 Lexikální analýza

// doplnit

4.5 Syntaktická analýza

// doplnit

4.6 Sémantická analýza

// doplnit

4.7 Generátor tříadresného kódu

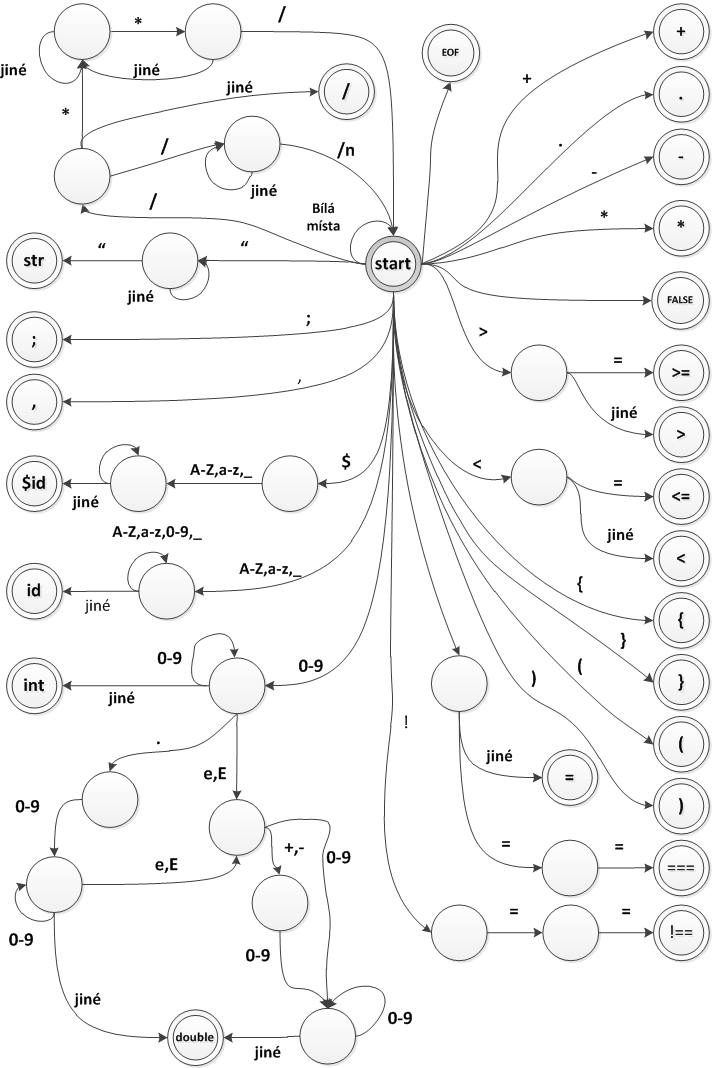
// doplnit

5 Závěr

// doplnit

6 Přílohy

6.1 Konečný automat lexikálního analyzátoru



6.2 LL gramatika

// doplnit