Obsah obrázku text

Popis byl vytvořen automaticky

**Implementácia prekladača imperatívneho jazyka IFJ22**

**Tým xkniaz00, varianta BVS**

Daniil Kniazkin (xkniaz00) - vedúci

Kotvitskiy Nikita (xkotvi01)

Hrubý Erik (xhruby30)

Hricovová Alžbeta (xhrico00) 7.12.2022

Obsah

[1. Úvod 3](#_Toc121249151)

[2. Implementačné riešenie 3](#_Toc121249152)

[2.1. Lexikálna analýza 3](#_Toc121249153)

[2.2. Syntaktická analýza 3](#_Toc121249154)

[2.3. Sémantická analýza 3](#_Toc121249155)

[2.4. Generovanie jazyka 4](#_Toc121249156)

[2.5. Tabuľka symbolov – varianta BVS 4](#_Toc121249157)

[3. Rozdelenie práce medzi členmi tímu 4](#_Toc121249158)

[4. Zdroje 5](#_Toc121249159)

[5. Diagram deterministického konečného automatu pre lexikálnu analýzu 6](#_Toc121249160)

[6. LL gramatika 7](#_Toc121249161)

# Úvod

Cieľom projektu bolo vytvoriť program v jazyku C, ktorý načíta zdrojový kód zapísaný v zdrojovom jazyku IFJ22 a preloží ho do cieľového jazyka IFJcode22. Jazyk IFJ22 je zjednodušenou podmnožinou jazyka PHP. Zdrojový kód je načítavaný zo štandardného vstupu a výsledný kód sa generuje na štandardný výstup.

# Implementačné riešenie

V tejto kapitole sú uvedené stručné popisy jednotlivých častí implementácie a ich vzájomná spolupráca.

# Lexikálna analýza

Prvou časťou prekladača bolo vytvorenie skenera, ktorý má za úlohu rozpoznávať jednotlivé lexikálne jednotky a prevádzať ich na príslušné tokeny. Implementácia sa nachádza v súboroch *scanner.c* a *scanner.h* a je realizovaná deterministickým konečným automatom.

Hlavnou funkciou skeneru je *readProgram*, ktorá opakovane volá funkciu *getToken*, pokiaľ nenarazí na token značiaci koniec súboru alebo token uzatváracej značky ?>. *getToken* ďalej opakovane volá funkciu *processChar,* ktorá obsahuje konštrukciu switch, kde každý case predstavuje stav automatu. Pre odlíšenie kľúčového slova od identifikátorov hľadáme jeho výskyt v tabuľke kľúčových slov realizovanej ako binárny vyhľadávací strom.

Štruktúra tokenu je nasledovná:

type - určuje druh tokenu

textData - rôzne textové údaje o tokene(napr. meno premennej, identifikátor funkcie...)

numericData - int/float hodnota tokenu

Ďalšie časti prekladača pracujú so štruktúrou *program\_t,* ktorá obsahuje zoznam a počet tokenov naplnených v *readProgram*.

## Syntaktická analýza

## Sémantická analýza

Sémantická analýza sa nachádza v súboroch sematic.c a semantic.h a je riešená dvomi priechodmi. Každý priechod je realizovaný ako deterministický konečný automat cez zoznam tokenov. Funkcia *getFunTable* vykonáva prvý priechod, ktorý naplní tabuľku symbolov *funcTable* pre hlavné telo programu symbolmi pre deklarované funkcie.

Funkcia *semanticControl* vykonáva druhý priechod. Pre telo definovaných funkcií sa vytvára nová lokálna tabuľka symbolov *localTable.*

Funkcia *funCallToken* kontroluje volanie funkcie – či existuje funkcia v tabuľke symbolov *funcTable*, správny typ a počet parametrov.

Funkcia *varToken* vyhodnotí výraz priradenia a aktualizuje/uloží premennú do tabuľky symbolov.

## Generovanie jazyka

## Tabuľka symbolov – varianta BVS

Tabuľka symbolov je implementovaná v súboroch *symtable.c* a *symtable.h* a je riešená ako binárny vyhľadávací strom, kde kľúče prvkov sú identifikátori funkcií alebo premenných zahashované pomocou *Jenkins hash function* [[1]](#_Zdroje).

Štruktúra symbolu tabuľky je nasledovná:

type - určuje či je to symbol premennej alebo funkcie

dtype - určuje typ pre symbol premennej a informácie o návratovom type a parametroch pre symbol funkcie

# Rozdelenie práce medzi členmi tímu

* Daniil Kniazkin :
* Kotvitskiy Nikita :
* Hrubý Erik :
* Hricovová Alžbeta :

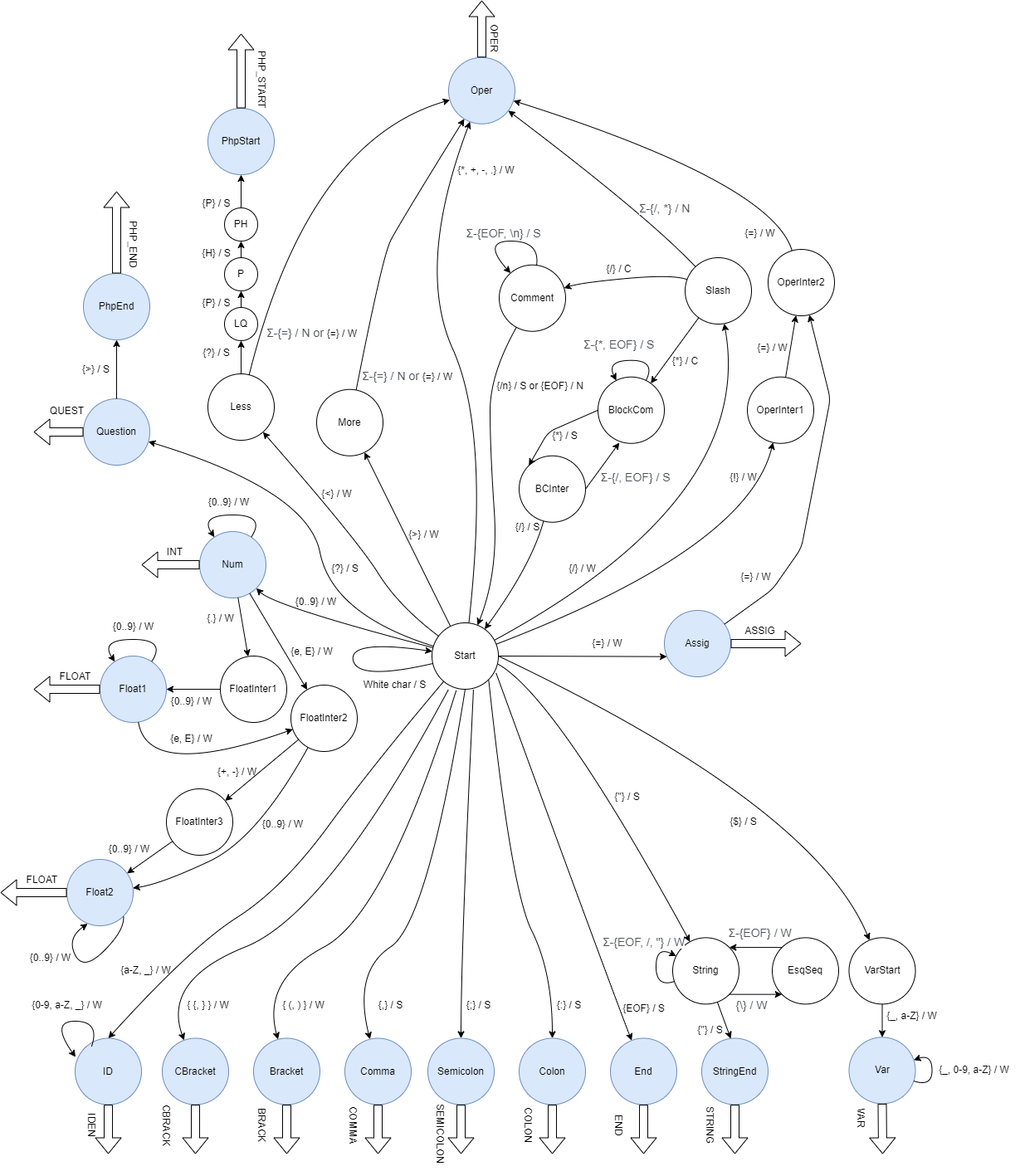
# Zdroje

[1] Jenkins hash function, <https://en.wikipedia.org/wiki/Jenkins_hash_function>

[2] Formální jazyky a překladače, <http://www.fit.vutbr.cz/study/courses/IFJ/public/materials/>

[3] PHP manuál, <https://www.php.net/manual/en/index.php>

# Diagram deterministického konečného automatu pre lexikálnu analýzu



# LL gramatika

Obsah obrázku stůl

Popis byl vytvořen automaticky