VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Dokumentace k projektu

do předmětu IFJ a IAL

Implementace interpretu imperativního jazyka IFJ16

Tým 029, varianta b/3/I

Jiruška Adam, xjirus01, 50% - vedoucí

Janeček David, xjanec28, 50%

Kuba Michal, xkubam02, 0%

Karpíšek Miroslav, xkarpi05, 0%

Žádná rozšíření

Obsah

[1. Úvod 3](#_Toc468718230)

[2. Implementace 3](#_Toc468718231)

[2.1 Lexikální analyzátor 3](#_Toc468718232)

[2.1.1 Deterministický koneční automat lexikálního analyzátoru: 4](#_Toc468718233)

# 1. Úvod

# 2. Implementace

## 2.1 Lexikální analyzátor

Lexikální analyzátor (scanner) je založen na deterministickém konečném automatu a je to jediná část interpretu pracující přímo se vstupním souborem. Jeho úkolem je ze vstupního textu odstranit nepotřebné části (bílé znaky, komentáře) a zbylé lexémy reprezentovat pomocí tokenů.

Scanner je ovládán syntaktickým analyzátorem, který si žádá o tokeny, které slouží jako jeho vstup. Při implementaci jsem vzorovou knihovnu str.c z ukázky jednoduchého interpretu.

## 2.2 Syntaktický analyzátor

Syntaktický analyzátor funguje na principu rekurzívního sestupu. Podle tokenu, který obdrží od lexikálního analyzátoru, se rozhoduje, které z pravidel LL gramatiky má použít. Při tvorbě gramatiky (viz obr. 1) bylo potřeba zohlednit, aby šlo pravidlo vybrat vždy jednoznačně.

Další částí syntaktického analyzátoru je precedenční analýza, používaná ke zpracování výrazů. Ta pracuje se zásobníkem neterminálů a terminálů, a také s precedenční tabulkou (viz obr. 2). Podle této tabulky se provádějí jednotlivá pravidla a tvoří kód pro provedení výrazu.

## 2.3 Interpret

Interpret je část programu vykonávající kód, který byl vygenerován v průběhu syntaktické analýzy. V naší implementaci je interní kód rozdělen na pásky pro jednotlivé funkce, mezi kterými se interpret přepíná při provádění programu. Při volání funkcí je také důležité vytvořit si rámec funkce, kde jsou uložené hodnoty lokálních proměnných pro danou funkci, také je tam návratová adresa a místo, kam se vrací hodnota. Tento rámec se vkládá na zásobník rámců, kde vždy na vrcholu je rámec aktuálně prováděné funkce. K dispozici jsou také rámce poslední prováděné funkce a funkce, která bude zavolána (připravovaný rámec).

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

LL gramatika

<prog> -> CLASS <class> <classes>

<classes> -> CLASS <class> <classes>

<classes> -> EPS

<class> -> ID { <def\_list> }

<def\_list> -> STATIC <definition> <def\_list>

<def\_list> -> EPS

<definition> -> DATA\_TYPE ID <definition\_rest>

<definition\_rest> -> ;

<definition\_rest> -> = <expression> ;

<definition\_rest> -> ( <param\_list> ) <body>

<param\_list> -> DATA\_TYPE ID <param\_rest>

<param\_rest> -> , DATA\_TYPE ID <param\_rest>

<param\_rest> -> EPS

<body> -> { <stat\_list> }

<stat\_list> -> <stat> <stat\_list>

<stat\_list> -> EPS

<stat> -> EPS

<stat> -> DATA\_TYPE ID <definition\_rest>

<stat> -> ID <stat\_rest>

<stat> -> { <stat\_list> }

<stat> -> IF ( <expression> ) <body> ELSE <body>

<stat> -> WHILE ( <expression> ) { <stat\_list> }

<stat> -> RETURN <expression> ;

<arguments> -> <arg> <arguments\_rest>

<arguments\_rest> -> , <arg> <arguments\_rest>

<arguments\_rest> -> EPS

<arg> -> TERM

### 2.1.1 Deterministický koneční automat lexikálního analyzátoru:

