VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Dokumentace ke společnému projektu pro předměty IFJ a IAL

Implementace interpretu imperativního jazyka IFJ16.

Tým 086, varianta a/3/II

ZS 2016

Řešitelé:

Richtarik Lukáš	xricht25
Čechák Jiří	xcecha04
Mlýnek Přemysl	xmlyne04
Mynarčík Petr	xmynar05
Molitoris Miloš	xmolit00

Rozšíření: FUNEXP, SIMPLE, (BOOLOP), (UNARY)

11. říjen 2016

1. Úvod

Tato dokumentace popisuje implementaci interpretu imperativního jazyka IFJ16, který je zjednodušenou podmnožinou jazyka Java SE 8. Implementován je v jazyce C.

Interpret načte zdrojový soubor v jazyce IFJ16 a následně kontroluje, zdali v něm nejsou lexikální, syntaktické nebo sémantické chyby a během interpretace se kontroluje, zdali nenastanou běhové chyby. V případě nalezení chyby nebo projevení nějaké interní chyby interpretu, vypíše na standardní chybový výstup chybovou hlášku a ukončí se s návratovou hodnotou dané chyby.

Jsou zde popsány implementace modulů, použité algoritmy a také diagram konečného automatu použitého v lexikální analýze, a také LL-gramatika a precedenční tabulka, které byly použity v syntaktické analýze.

2. Diagram konečného automatu

3. LL-gramatika

4. Precedenční tabulka

5. Implementace modulů a algoritmů

5.1 Lexikální analýza

Lexikální analýza je implementována pomocí konečného automatu. Začíná se ve stavu state_Start a v závislosti na načteném znaku se konečný automat posune do dalšího stavu, kde se již kontroluje na základě načtení následujících znaků, zdali je konkrétní lexém napsán správně.

Pokud je načten znak, který do daného lexému nepatří nebo se jedná o neočekávaný znak, jde o chybu a program je ukončen s návratovou hodnotou 1.

U komentářů se kontroluje, zdali jsou korektně zapsány, u víceřádkových i ukončeny.

Po přečtení a zpracování celého lexému, teď již tokenu, je tento token předán syntaktickému analyzátoru k syntaktické a sémantické analýze.

5.2 Syntaktická a sémantická analýza

5.4 Interpret

5.5 Implementace vyhledávání podřetězce v řetězci

Pro vyhledávání podřetězce v řetězci byl podle zadání použit Knuth-Morris-Prattův algoritmus. Tento algoritmus pracuje na základě konečného automatu a byl implementován takto:

Pokud je hledaný podřetězec prázdný (má nulovou délku), nachází se tento podřetězec na nulté pozici řetězce a je tedy funkcí vrácena 0. Jinak je na základě řetězce, ve kterém vyhledáváme vytvořeno pole (vektor), kde hodnoty jednotlivých prvků představují hrany z uzlů konečného automatu. Poté konečný automat postupuje po jednotlivých znacích řetězce, při shodě se posune na další znak, jinak se vrátí na znak určený dříve vytvořeným polem.

5.6 Implementace řazení

K implementaci funkce na řazení řetězce byl použit dle zadání algoritmus Shell sort, který pracuje na principu vkládání.

Nejdříve se získá délka kroku, což je polovina délky řetězce. Následně se v cyklu řadí znaky řetězce, které jsou od sebe v řetězci vzdáleny o velikost kroku a s každou další iterací se délka kroku zmenší o polovinu. Jakmile má krok velikost jedna, jsou řazeny prvky vedle sebe.

5.7 Implementace tabulky symbolů

Tabulka symbolů je implementována jako tabulka s rozptýlenými položkami s explicitním zřetězením synonym. K vyhledávání v tabulce symbolů a k ukládání položek do tabulky symbolů slouží klíč, kterým je identifikátor proměnné nebo funkce.

V globální tabulce symbolů jsou uloženy funkce s jejich parametry a globální proměnné. Do lokální tabulky symbolů se ukládají lokální proměnné daných funkcí.

6. Práce na projektu

Nejdříve se rozdělili moduly, na kterých se začalo pracovat. Jakmile byly tyto moduly hotovy a otestovány, začalo se pracovat na dalších modulech. V průběhu práce na jednotlivých modulech se náš tým několikrát sešel k prodiskutování průběhu vývoje a potřebných změn v implementaci již implementovaných modulů.

Pokud nebyla možnost se sejít nebo bylo potřeba něco rychleji sdělit ostatním členům týmu, případně se na něco zeptat, využíval se ke komunikaci společný chat a soukromá skupina na sociální síti Facebook nebo email.

Po implementování všech modulů a sestavení projektu proběhlo závěrečné testování interpretu jako celku za použití námi vytvořených zdrojových kódů v IFJ16.

7. Závěr

Prací na tomto projektu, jsme získali nové zkušenosti v programování v jazyce C, a také velkou zkušenost práce v malém týmu a komunikace s ostatními členy týmu.