FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Dokumentácia – IFJ 2018

Tím 40, varianta II

Adam Hostin	xhosti02	25 %
Sabína Gregušová	xgregu02	25 %
Dominik Peza	**xpezad00**	**25 %**
Adrián Tulušák	xtulus00	25 %

Obsah

1 Úvod

Naším cieľom je implementovať prekladač imperatívneho jazyka IFJ18 do predmetov IFJ a IAL v jazyku C. Hlavnou náplňou práce bola implementácia: lexikálneho analyzátora, parsera (syntaktická a sématická analýza) a generátora inštrukcií.

2 Lexikálna analýza

Na začiatku sme implementovali lexikálny analyzátor v súbore `lexer.c`, ktorého základom je deterministický konečný automat (ďalej iba DKA). Hlavnou funkciou v tomto súbore je `get_next_token`, ktorá číta jednotlivé znaky a pomocou príkazu `switch` prechádza do nasledujúcich stavov podľa DKA až kým nevyhodnotí lexikálne správny token, inak vracia `ER_LEX`. Lexikálny analyzátor musí mať nadstavený vstupný súbor, ktorý obsahuje program napísaný v jazyku IFJ18. Pri úspešnom vyhodnotení tokenu sa správne uvoľní všetká alokovaná pamäť. Matematické a relačné operátory sú vyhodnotené vcelku rýchlo a jednoducho, identifikátory, reťazce a čísla vyžadujú viacej prechodov a používajú dynamický reťazec, o ktorom ďalej pojednáva sekcia Spracovanie reťazcov. Pri identifikátore sa kontrolujú povolené znaky na základe pozície v reťazci a na záver sa identifikátor porovná so všetkými kľúčovými slovami, ak sa nájde zhoda, je to kľúčové slovo, inak je to identifikátor. Reťazce sú ohraničené dvojitými úvodzovkami (") a môžu obsahovať escape sequence. Pre tento prípad existuje špeciálny stav `STATE_BACKSLASH_LITERAL`, do ktorého sa prechádza pri prečítaní znaku \ a čaká sa na skratku escape sequence, ako napríklad t, s alebo n.

2.1 Štruktúra Token_t

Pre jednoduchšiu prácu s tokenmi sme použili štruktúru `Token_t`, ktorá obsahovala:

- **union Token_attr**
- **struct Token_type**

Union Token_attr obsahuje možné atribúty tokenu, konkrétne to sú: string, integer, flt a keyword.
Struct Token_type obsahuje typy tokenov, konkrétne to sú: `EOF`, `EOL`, identifikátor, kľúčové slovo, relačné a matematické operátory, ľava a pravá zátvorka, čiarka, komentár, int, float a string.

2.2 Spracovanie reťazcov

Pre jednoduchšie spracovanie reťazcov sme sa rozhodli implementovať súbor `dynamic_string.c`. Jeho súčasťou je aj štruktúra `string_t`, ktorá obsahuje samotný ukazateľ na dynamický reťazec, súčastnú veľkosť reťazca a celkovú veľkosť bufferu. Na začiatku je alokovaný reťazec s veľkosťou 10 a pri každom pridaní znaku sa kontroluje, či je ešte v reťazci miesto. Keď sa blížime k zaplneniu reťazca, funkcia `check_empty_bites` zväčší veľkosť buffera o 5, čím zaistí adekvátnu veľkosť pre reťazec. Všetky alokácie pamäte sú kontrolované a ich zlyhanie je adekvátne ošetrené vrátením internej chyby `ER_INTERNAL`.

3 Syntaktická analýza

3.1 Sématická analýza

3.2 Tabuľka symbolov

4 Generátor inštrukcií

5 Práca v tíme

Náš tím sme si zostavili pomerne skoro.

5.1 Komunikácia

Už na začiatku sme sa dohodli na pravidelných týždenných stretnutiach, kde sme diskutovali o našej ďalšej práci na nadchádzajúci týždeň.

5.2 Verzovanie

5.3 Hodnotenie

6 Prílohy

6.1 Deterministický konečný automat

6.2 LL-gramatika

6.3 Precedenčná tabuľka

	+	**-**	*****	**/**	**(**	**)**	**i**	**R**	**$**
+	>	<	<	<	<	>	<	>	>
-	>	>	<	<	<	>	<	>	>
*****	>	>	>	>	<	>	<	>	>
/	<	>	>	>	<	>	<	>	>
(	<	<	<	<	<	=	<	<	
)	>	>	>	>		>		>	>
i	>	>	>	>		>		>	>
R	<	<	<	<	<	>	<		>
$	<	<	<	<	<		<	<	