VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Projekt z předmětů IFJ & IAL
Překladač jazyka IFJ17

Švub Daniel (xsvubd00)

Poisl Daniel (xpoisl00)
Weigel Filip (xweige01)
Zwierz Jan (xzwier00)

5. prosince 2017 Tým 88, varianta II

Obsah

1 Úvod

Tato dokumentace popisuje návrh a způsob implementace projektu z projektů IFJ a IAL. Jedná se o analyzátor zdrojových kódů jazyka IFJ17, zjednodušené varianty jazyka FreeBasic. Druhá část projektu, interpret daného jazyka do mezikódu IFJcode17, není implementována. Program pouze ověří lexikální, syntaktickou a sémantickou správnost zdrojového kódu.

2 Práce v týmu

2.1 Rozdělení práce

- **Daniel Švub** – vedoucí týmu, gramatika, parser, zásobník, úpravy
- **Daniel Poisl** – scanner, tabulka symbolů
- **Filip Weigel** – scanner, GIT repozitář
- **Jan Zwierz** – gramatika, dokumentace, makefile

2.2 Komunikace mezi členy

Naše schůzky probíhaly na kolejích, kde bydlí 3 ze 4 členů týmu. Na první schůzce byly rozděleny úkoly a části projektu jednotlivým členům. Členové na této koleji se scházeli jak uznali za vhodné. Čtvrtý člen docházel, kdykoli to bylo potřeba, popř. jsme komunikovali přes internet.

2.3 Sdílení výsledků práce

Ke sdílení práce jednotlivců byl členem týmu Filipem Weiglem vytvořen repozitář na GitHubu. Tento repozitář byl využíván ke sdílení aktuálních verzí souborů, na kterých jednotliví členové provedli jakékoli úpravy. Jako první na repozitář byly umístěny soubory scanner.c a parser.c společně s hlavičkovými soubory. Následovaly další zdrojové soubory s tabulkami vypracovanými v excelu a nakonec i dokumentace.

2.4 Problémy

Jelikož jsme špatně odhadli rozsah projektu a měli jsme problém se zahájením implementace, nedokázali jsme dokončit celý projekt. Problém spočíval v přílišném odkládání zahájení prací na projektu s tím, že při samotné implementaci jsme naráželi na víc problémů než jsme

očekávali. Například: problémy s kompatibilitou zdrojových souborů tvořených jednotlivými členy, přičemž každý očekával něco jiného od zdrojového kódu druhé strany. To způsobovalo nečekané chyby při překladu programu a vyžádalo si zdlouhavé debuggování.

3 Implementace

3.1 Scanner

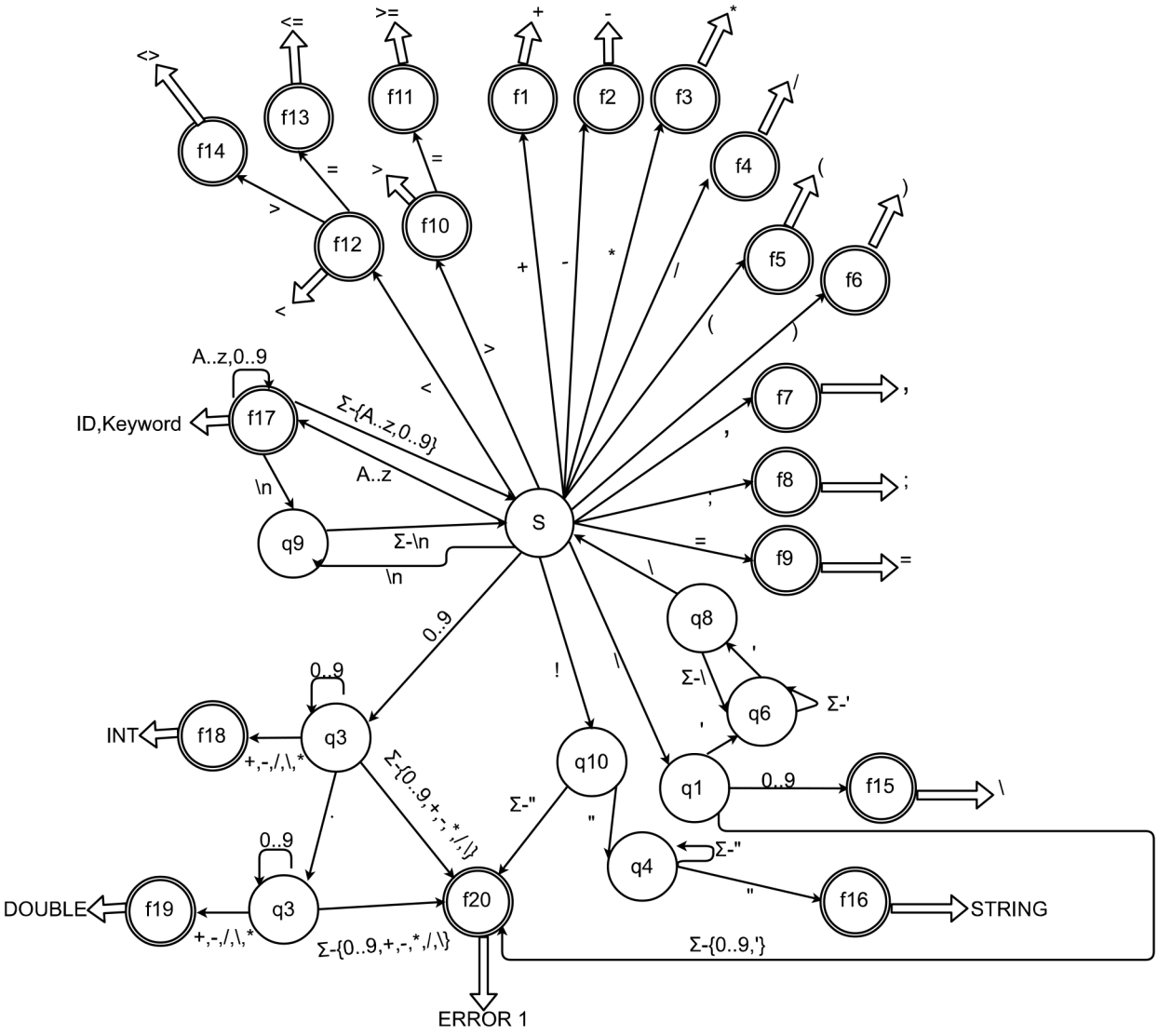
Scanner je v programu využíván na čtení symbolů ze vstupu a následným předáním dat parseru, ale provádí i lexikální analýzu. Realizován je pomocí podmínek, které rozpoznají jednoduché znaky (+, -, . . .) a pro složitější výrazy jako např. klíčová slova je využito jednotlivých case. Celkem je ve scanneru použito 10x case. Každý case slouží k jinému rozpoznávání, ale některý case se skládá i z více case.

Scanner je volán Parserem. Scanner následně začne číst vstupní kód a zapisuje data na pole charu. Pokud je zjištěna lexikální chyba v zadávaném kódu, je Scanner ukončen a posílá Parseru hodnotu 1(chyba v programu v rámci lexikální analýzy). Data se poté posílají zpět Parseru na syntaktickou a sémantickou analýzu.

- **Case 1**: Spustí se pokud je zadán *backslash* a určí zda se jedná o celočíselné dělení, nebo o komentář
- **Case 2**: Rozpoznávání zda se jedná o klíčové slovo jazyka, nebo zda je zadáván identifikátor proměnné, funkce.
- **Case 3**: Zapisování a určení zda se jedná o celé číslo. Pokud je zadána tečka tak se spouští *Case 7*.
- **Case 4**: zapisování vstupního řetězce. Zapisování se ukončí pokud je zadána *uvozovka*.
- **Case 5**: Čte řádkový komentář a hledá jeho ukončení pomocí odřádkování v kódu.
- **Case 6**: Čte blokový komentář. Po přečtení apostrofu je spuštěn *Case 8*.

- **Case 7**: Čte čísla a zapisuje si jejich hodnotu. Po zadání operátoru je příslušná hodnota čísla odeslána Parseru.
- **Case 8**: Kontroluje zda se po *apostrofu* nachází *backslash*. Pokud ano tak je ukončen blokový komentář, pokud ne tak se opět spouští *Case 6*.
- **Case 9**: Ošetření vícenásobného odřádkování v zadávaném kódu.
- **Case 10**: Kontrola jestli se po vykřičníku zadává vstupní řetězec. Pokud je za vykřičníkem cokoliv jiného než uvozovka tak je vrácena lexikální chyba kódu.

Scanner byl vyvíjen postupným přidáváním podmínek pro rozpoznávání jednotlivých znaků. Začínalo se jednoduchým rozpoznáváním operátorů (+, -, *, /). Další fáze bylo rozpoznávání klíčových slov a identifikátorů. Závěrečným krokem bylo oddělení komentářů ze vstupu, aby se překládal pouze kód a zapisování vstupních řetězců a čísel.


<>
<=
>=
+
-
*
/
(
)
f14
f13
f11
f1
f2
f3
f4
f5
f6
f12
f10
=
>
<
A..z,0..9
ID,Keyword
f17
Σ-{A..z,0..9}
A..z
\n
S
f7
,
f8
;
f9
=
q9
Σ-\n
\n
q8
\
0..9
!
Σ-\
q6
Σ-'
'
INT
f18
q3
+,-,/,\,*
Σ-{0..9,+,-,*,/,\}
q10
q1
0..9
f15
\
Σ-"
"
q4
f16
STRING
DOUBLE
f19
f20
Σ-{0..9,+,-,*,/,\}
Σ-{0..9,'}
ERROR 1