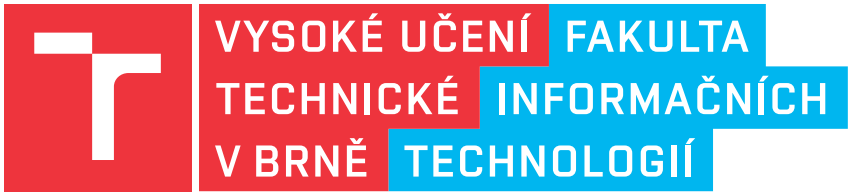

Projektová dokumentace

Implementace překladače imperativního jazyka IFJ20

Tým 109, varianta I

Jan Kleisl	**(xkleis00)**	50 %
Jan Frühauf	(xfruha00)	50 %
Otakar Sedlák	(xsedla1r)	0 %
Vladyslav Tverdokhlib	(xtverd01)	0 %

9. prosince 2020

Obsah

1 Úvod

V projektu jsme se měli naučit naprogramovat překladač v jazyce C. Tento překladač měl načítat jako zdrojový jazyk IFJ20 (podmnožina programovacího jazyka Go) a překládat ho do cílového jazyka IFJcode20 (mezikód). Překladač načítá ze standardního datového vstupu a vypisuje na standardní datový výstup.

2 Návrh, implementace a jednotlivé části

Překladač se skládá z několika částí, na kterých jsme v průběhu semestru pracovali. Níže budou popsány jednotlivé části podrobněji.

2.1 Lexikální analýza

Lexikální analýza je tvořena souborem `scanner.c` a jeho hlavičkovým souborem `scanner.h`. Scanner načítá znaky a postupně z nich vytváří tokeny a ty ukládá do datové struktury `Token` která obsahuje `Token_types` a `Token_attribute`. Typy tokenu jsou datového typu `enum` a atribut je datového typu `union`. Rozlišujeme několik typů tokenů a to, datové typy, klíčová slova, operátory, relační znaménka, symboly a stavy `ENF_OF_LINE`, `END_OF_FILE` a `ERROR`. Dále pak přidělujeme jeden z následujících atributů: `float64`, `integer`, `keyword` a `string`.

2.2 Syntaktická analýza

Syntaktická analýza tvoří hlavní část našeho překladače. Přijímá tokeny ze scanneru, řeší korektní syntaxi kódu a volá jeho generování. Skládá se ze souborů `parser.c` a `parser.h`. Syntaktická analýza se řídí LL – gramatikou a pravidly v LL – tabulce. Zpracování výrazů provádíme v odděleném modulu - soubory `expression.c` a `expression.h`. Takzvaná precedenční analýza vyhodnocuje výrazy pomocí precedenční tabulky. Precedenční tabulka se skládá z operátorů + a -, které mají stejnou prioritu, stejně tak jako další operátory * a / proto jsou sloučeny do 2 sloupců/řádků místo 4. Dále v precedenční tabulce nalezneme levou závorku (, kterou následuje hned pravá závorka) . Jako další v pořadí jsou relační operátory =, !=, >=, <=, > a < , které jsou taky sloučené do jednoho sloupce/řádku. Předposlední se v tabulce nachází i , které zastupuje všechny povolené literály. Všechny nepovolené symboly zastupuje znak dalaru $.

2.3 Sémantická analýza

Tabulku symbolů jsme zpracovávali pomocí binárního vyhledávacího stromu, protože jsme měli tuto variantu projektu. V tabulce můžeme vyhledávat jestli vyhledávaný identifikátor existuje a zda odpovídají i ostatní náležitosti.

2.4 Generování výsledného kódu

Cílovým jazykem pro vygenerování kódu, jak již bylo dříve zmíněno, je IFJcode20. Kód je generován z parseru, který si však jen volá funkce ze souboru `codegenerator.c` jehož hlavičkovým souborem je `codegenerator.h`.

2.4.1 Průběh generování

První věc, která je vygenerována je hlavička, definice globálních proměnnýchv glogálním rámci `GF` a definice vestavěných funkcí. Dále už pak záleží na parseru jaké funkce na vygenerování jakých částí kódů volá.

2.4.2 Generování úseků

Generátor pomocí implementovaných funkcí umožňuje generovat jednotlivé části a úseky kódu. Generování funkce je rozděleno do 8 funkcí pro vygenerování různých částí. Mezi základní patří například `gen_func_start`, `gen_func_end` nebo `gen_func_call`. Generování proměnných je rozděleno na 3 podfunkce - inicializaci, deklaraci a přiřazení základní hodnoty (nastavení na 0, prázdný string nebo false v závislosti na typu). Dále generátor obsahuje funkce na generování úseku `IF`, `ELSE` a matematické operace.

2.5 Překlad projektu

Projekt je překládán pomocí souboru `Makefile` příkazem `make`, jak bylo řečeno v zadání.

3 Vlastní datové struktury

3.1 Binární vyhledávací strom

Pro realizaci tabulky symbolů jsme využili datové struktury `tBSTNode`, nacházející se v souboru `symtable.c` a `symtable.h`, kterou jsme znali z předmětu IAL a také jsme měli zadanou. Každý uzel stromu obsahuje klíč, datovou strukturu `tData` a ukazatel na levý a pravý podstrom. Struktura `tData` v sobě dále nese `tType`, `tAttributes` a počítadlo. Struktura `tType` rozlišuje mezi proměnnou a funkcí a struktura `tAttributes` mezi integerem, floatem a stringem. Binární strom má dále implementované funkce pro správnou funkčnost.

3.2 Dynamický řetězec

Dynamický řetězec je implementovaný v souboru `string_dyn.c` s hlavičkou v souboru `string_dyn.h`. Struktura nese název `string_dyn` a jak již z názvu vyplývá, jedná se o dynamický řetězec. Tuto strukturu jsme vytvořili z toho důvodu, že při generování nevíme, jak dlouhý bude výsledný řetězec dlouhý. Nad touto strukturou je implementováno několik funkcí potřebných pro správné fungování podle naší potřeby.

3.3 Zásobník symbolů

Jako další z vlastních datových struktur jsme implementovali zásobník symbolů, který využíváme během vyhodnocování výrazů v precedenční analýze. Tento zásobník nalezneme v souboru `symstack.c` a v jeho hlavičkovém souboru `symstack.h`. Znovu jsme implementovali funkce potřebné k správnému chodu.

4 Týmová spolupráce a práce v týmu

4.1 Verzovací systém

Hned na začátku jsme se dohodli, že jako verzovací systém budeme využívat GitHub. Proto jsme si tam vytvořili soukromý repozitář, do kterého jsme si udělili přístup. Systém nám umožnil psát překladač ve větvích, proto jsme si vždy na nějakou část programu (scanner, parser, codegen, atp.) větev vytvořili a v ní pracovali.

4.2 Komunikace

Jako hlavní komunikační kanál jsme využívali Discord. Měli jsme vytvořenou místnost pro vývoj a hlasový kanál. Bohužel, jak se později ukázalo, jsme hlasový kanál nevyužívali a komunikovali jsme jen písemnou formou. Komunikace velmi vázla a dospěla do stádia, kdy jsme na projektu pracovali pouze ve 2, čemuž odpovídá i výsledné rozdělení.

4.3 Rozdělení práce

Jak již bylo zmíněno projekt jsme dělali pouze ve 2 lidech. Proto tedy následující rozdělení.

Jan Kleisl: LL-gramatika, precedenční analýza, generování kódu, dokumentace
Jan Frühau: lexikální analýza, syntaktická analýza, sémantická analýza, LL-gramatika
Otakar Sedlák: -
Vladyslav Tverdokhlib: -

5 Závěr

Na projektu jsme začali pracovat později než bylo záhodno a tudíž jsme lehce nestíhali. Na tomto dodává i tvrzení, že jsme při práci postrádali 2 lidi což je půl týmu a to se také projeví. Velmi nám byly nápomocné přednášky a democvičení z předmětů IFJ a IAL. V průběhu jsme se setkávali s různými problémy, které jsme alespoň částečně zvládali odbavit v nejbližších následujících dnech. Tento projekt nám dal hodně zkušeností do budoucího profesního života.

6 Přílohy

6.1 LL-gramatika

6.2 LL-tabulka

6.3 Precedenční tabulka