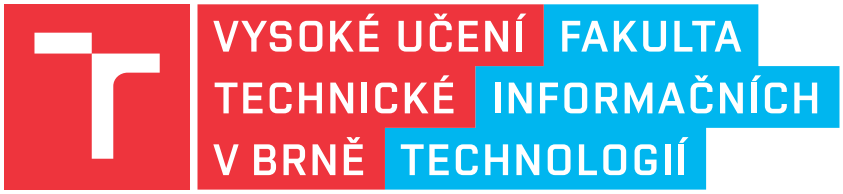

Projektová dokumentace

Implementace překladače imperativního jazyka IFJ17

Tým 104, varianta II

Dominik Harmim	**(xharmi00)**	25 %
Vojtěch Hertl	(xhertl04)	25 %
Timotej Halás	(xhalas10)	25 %
Matej Karas	(xkaras34)	25 %

28. října 2017

Obsah

1 Úvod

Cílem projektu bylo vytvořit program v jazyce C, který načte zdrojový kód zapsaný ve zdrojovém jazyce IFJ17, jenž je zjednodušenou podmnožinou jazyka FreeBASIC a přeloží jej do cílového jazyka IFJcode17 (mezikód).

2 Návrh a implementace

2.1 Lexikální analýza

TODO

2.2 Syntaktická analýza

TODO

2.3 Sémantická analýza

TODO

2.4 Optimalizace

TODO

2.5 Generování cílového kódu

TODO

2.6 Překladový systém

Projekt je možné přeložit dvěma způsoby, buď nástrojem CMake nebo nástrojem GNU Make. Oba způsoby vytvoří spustitelný soubor `ifj17_compiler`.

2.6.1 CMake

Při vývoji, ladění a testování projketu někteří z nás používaly CMake, protože překlad tímto nástrojem je integrován ve vývojovém prostředí CLion, které většina členů týmu používala a protože nastavení pravidel pro překlad tímto nástrojem je jednoduché.

V souboru `CMakeLists.txt` jsou nastavena pravidla pro překlad. Je zde nastaven překladač `gcc` a všechny potřebné parametry pro překlad. Dále je zde uvedeno, že se pro překlad mají používít všechny soubory s příponou `.c` a `.h` a taky je zde uveden název spustitelného souboru.

Soubory pro překlad je možné vygenerovat příkazem `cmake .` a překlad provést příkazem `cmake --build .` nebo pro toto využít nástroje vývojového prostředí.

2.6.2 GNU Make

Jedním z požadavdků pro odevzdání projektu bylo přiložit k odevzdanému projektu i soubor `Makefile` a přeložení projektu příkazem `make`. Z tohoto důvodu jsme vytvořili překladový systém i pomocí nástroje GNU Make.

V souboru `Makefile` jsou nastavena pravidla pro překlad. Je zde nastaven překladač `gcc` a všechny potřebné parametry pro překlad. Vytvořili jsme pravidla, které nejdříve vytvoří objektové soubory ze všech souborů s příponou `.c` v daném adresáři za pomoci vytvořených automatických pravidel a překladače `gcc`,

který dokáže automaticky generovat tyto pravidla, více viz *Dependecy Generation* v kapitole *C and C++* knihy [1]. Následně je ze všech objektových souborů vytvořen jeden spustitelný soubor.

Nástroj GNU Make jsme mimo překlad využili i pro automatické zabalení projektu pro odezvdání, generování dokumentace, spouštění automatických testů a čištění dočasných souborů.

3 Speciální algoritmy a datové struktury

3.1 Tabulka s rozptýlenými položkami

TODO

3.2 Dynamický řetězec

Vytvořili jsme strukturu `struct dynamic_string` pro práci s řezězci dynamické délky, kterou používáme např. pro ukládání řezězce nebo identifikátoru u atributu tokenu v lexikální analýze, protože dopředu nevíme, jak bude tento řetězec dlouhý.

Daná struktura a rozhraní operací nad ní je popsáno v souboru `dynamic_string.h` a operace jsou implementovány v souboru `dynamic_string.c`.

Struktura v sobě uchovává ukazatel na řezězec, velikost řetězce a informaci o tom, kolik paměti je pro řetězec vyhrazeno. Implementovali jsme operace pro inicializaci struktury, uvolnění vyhrazených dat, přidání znaku na konec řezězce, porovnávání řetězců, kopie celých řezězců a další. Při inicializaci struktury se vyhradí paměťový prostor pouze pro určitý počet znaků (určeno konstantou), až při nedostatku vyhrazené paměti se velikost paměti zvýší.

4 Práce v týmu

4.1 Způsob práce v týmu

Na projektu jsme začali pracovat začátkem října. Práci jsme si dělili postupně, tj. neměli jsme od začátku stanovený kompletní plán rozdělení práce. Na dílčích částech projektu pracovali většinou jednotlivci nebo dvojice členů týmu. Nejprve jsme si stanovili strukturu projektu a vytvořili překladový systém.

4.1.1 Verzovací systém

Pro správu souborů projektu jsme používali verzovací systém Git. Jako vzdálený repositář jsme používali GitHub.

Git nám umožnil pracovat na více úkolech na projektu současně v tzv. větvích. Většinu úkolů jsme nejdříve připravili do větve a až po otestování a schválení úprav ostatními členy týmu jsme tyto úpravy začlenili do hlavní vývojové větve.

4.1.2 Komunikace

Komunikace mezi členy týmů probíhala převážně osobně nebo prostřednictvím aplikace Slack, kde jsme si buď psali přímo mezi sebou nebo si vytvářeli různé skupinové konverzace podle tématu.

V průběhu řešení projektu jsme měli i několik osobních setkání, kde jsem probírali a řešili problémy týkající se různých částí projektu.

4.2 Rozdělení práce mezi členy týmu

Práci na projektu jsme si rozdělili rovnoměrně s ohledem na její složitost a časovou náročnost. Každý tedy dostal procentuální hodnocení 25 %. Tabulka 1 shrnuje rozdělení práce v týmu mezi jednotlivými členy.

Člen týmu	Přidělená práce
Dominik Harmim	vedení týmu, organizace práce, dohlížení na provádění práce, konzultace, kontrola, testování, dokumentace, struktura projektu
Vojtěch Hertl	lexikální analýza
Timotej Halás	syntaktická analýza, sémantická analýza
Matej Karas	syntaktická analýza, sémantická analýza

Tabulka 1: Rozdělení práce v týmu mezi jednotlivými členy

5 Závěr

TODO

Literatura

[1] Mecklenburg, R. W.; Oram, A.: *Managing projects with GNU make*. Cambridge [Mass.]: O'Reilly, třetí vydání, 2005, ISBN 05-9600-610-1.

A Diagram konečného automatu specifikující lexikální analyzátor

TODO

B LL-gramatika

TODO

C LL-tabulka

TODO

D Precedenční tabulka

TODO