

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ  
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Formální jazyky a překladače  
Překladač imperativního jazyka IFJ19  
Tým 052, varianta II

Vedoucí:	Jaroslav Hort	(xhortj04)	25%
	Filip Dráber	(xdrabe09)	25%
	Iveta Strnadová	(xstrna14)	25%
	Norbert Pócs	(xpocsn00)	25%

8. prosince 2019

# 1 Generátor výsledného kódu

Po úspěšné analýze je inicializován generátor výsledného kódu. Součástí inicializace je vytvoření tabulek s rozptýlenými položkami pro ukládání názvu vytvořených proměnných a labelů pro funkce. Vlastní spuštění generátoru spočívá ve volání funkce `generate_code`.

## 1.1 Tří adresný kód

Tří adresný kód je generován na úrovni parseru a precedenční analýzy. Jeho struktura obsahuje v typ kódu, `ac_type`, a tři operandy typu `Token`, `op1`, `op2`, `res`. Výsledná struktura kódu je uložena do seznamu kódu, který je využit v generátoru. Hlavními funkcemi seznamu jsou funkce `setACAct`, která nastaví aktivitu seznamu na první položku, `actAC`, pro nastavení aktivity na následující položku a `isACActive` pro zjištění, zda je v seznamu aktivní prvek.

## 1.2 Průběh generování

Ze seznamu tří adresných kódu jsou postupně brány položky a podle typu kódu je rozhodnuto jaká instrukce bude zapsána do výsledného kódu. Výsledný kód je v průběhu generování rozdělen na dva bloky, kód hlavního programu (`main`) a kód funkcí a ukládán do dynamického řetězce pro tyto bloky.

Tento způsob implementace byl zvolen z důvodu, že volání funkcí je realizováno pomocí skoku na návěští, pokud by byly funkce zapsány uvnitř hlavního kódu, bylo by nutné je přeskakovat, dokud nebudou volány, což by bylo horší na implementaci a výsledný kód by byl méně přehledný.

Po přečtení všech kódu a vygenerování instrukcí je výsledek vypsán na standardní výstup, nejprve kód funkce a poté kód hlavního bloku programu.

## 1.3 Vestavěné funkce

Vestavěné funkce jsou definovány v hlavičkovém souboru generátoru. V případě, že program poprvé volá jednu z těchto funkcí, jsou zapsány do bloku funkcí a funkce je volána jako obyčejné uživatelské funkce. Výsledný kód tedy obsahuje pouze ty funkce, které jsou využívány, tímto řešením udržíme kód relativně čistý.

## 1.4 Generování identifikátorů

Při generování instrukcí pracujících s identifikátory je nejprve prohledána tabulka, zda nebyl identifikátor generován již dříve. Identifikátory z hlavního bloku programu, takzvané globální identifikátory, mají přednost před identifikátory uvnitř funkcí, nebo-li lokálními identifikátory. Pokud nebyl nalezen, je zapsána instrukce do kódu a do příslušné tabulky je vloženo jméno identifikátoru

## 1.5 Problém generování cyklu

U generování cyklu `while` byl zjištěn problém možné definice identifikátoru. Tuto možnost jsme ošetřili tak, že parser vygeneruje tří adresný kód `WHILE_START` pro označení začátku cyklu a `WHILE_END` pro konec cyklu. Když generátor narazí na začátek cyklu, projde všechny kódy uvnitř cyklu, provede kontrolu definice identifikátorů a případně identifikátory nadefinuje předem, než cyklus začne.

## 2 Pomocné knihovny

### 2.1 Knihovna chyb `errors`

Knihovna obsahuje výčet kódů všech možných chyb, které mohou nastat při překladu programu, globální proměnnou `global_error_code`, kterou jednotlivé moduly nastavují na jeden z typu chyby v případě, že někde nastala. Na této proměnné závisí spuštění generátoru. Dále máme dvě funkce pro výpis chyby na standardní chybový výstup `stderr`, první pro výpis interní chyby a druhá pro kompilační chyby, která tiskne navíc název souboru a číslo řádku zdrojového programu na které se chyba vyskytuje.

### 2.2 Dynamický řetězec

Pro jednodušší práci s řetězcí v programovacím jazyce C jsme vytvořili knihovnu `dynamic_string`. Knihovna nám umožňuje vytvořit nový řetězec, rozšířit ho připsáním nového řetězce na konec stávajícího, nebo ho skrátit na požadovanou délku.

## 3 Práce v týmu

Pro jednoduchou práci v týmu jsme zvolili verzovací systém Git a náš repozitář umístili na server GitHub. Každý z nás měl vlastní větev, ve které psal svůj kód a dle potřeby ji sloučil s hlavní větví. Komunikace probíhala osobně v podobě schůzek v prostrách fakulty, nebo online před komunikační kanál Discord.

### 3.1 Rozdělení

Student	Práce	%
Jaroslav Hort	Generátor kódu, rozhraní, knihovny pro dynamický řetězec, chyby, tabulky rámců, dokumentace, prezentace	25%
Filip Dráber	Precedenční analýza, dokumentace	25%
Iveta Strnadová	Parser, LL tabulka, tabulka symbolů, tří adresný kód, knihovna pro práci s <code>indent_stack</code> , dokumentace	25%
Norbert Pócs	Lexikální analyzátor, knihovna pro token, dokumentace	25%