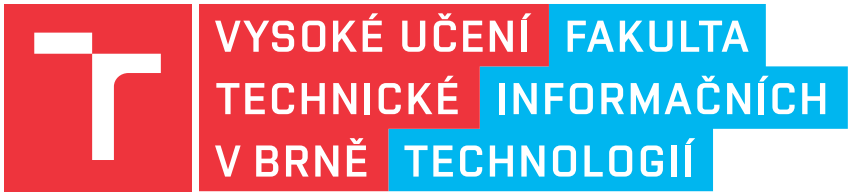


Projektová dokumentace

IPP

Interpret nestrukturovaného imperativního jazyka IPPcode23

Nikita Kotvitskiy (xkotvi01)

Obsah

1 Návrh a implementace I. části projektu **i**

2 Konečné automaty použité v parseru **i**

3 Návrh a implementace II. části projektu **i**

3.1 Třída Program i

3.2 Třída Interpret ii

3.3 Pomocné třídy ii

3.4 Diagram tříd ii

1 Návrh a implementace I. části projektu

Hlavní částí implementace je třída ”Parser”, jejíž rozhraní se skládá z jediné metody ParseProgram s jedním parametrem $source (obsahuje zdroj vstupu). Tato třída je schopná provést lexikální a syntaktickou analýzu kódu a zároveň vytvořit XML reprezentaci celého programu.

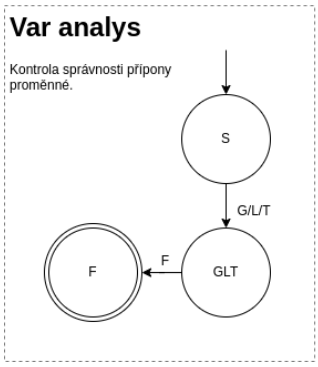
Analýza programu je kontextově závislá. Kód se čte řádek po řádku. Před analýzou řádek se upravuje: odstraňují se sekvence bílých znaků a komentáře.

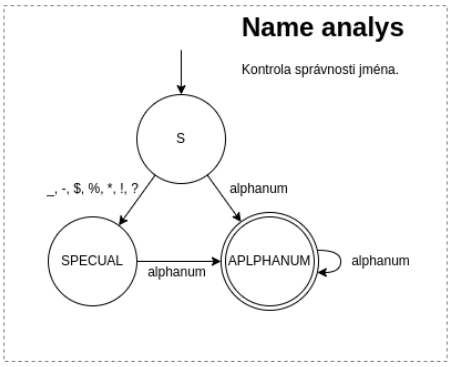
Samotná analýza se skládá ze dvou části: zaprvé se provádí kontrola hlavičky. Špatná hlavička způsobuje výskyt chyby 21. Pak se kontrolují příkazy. Při zpracování každého příkazu se vytváří nový XML elementy ”instruction”s přislušnými argumenty. Neznámé příkazy vedou na chybu 22. Každý příkaz se rozděluje na pole lexémů. První prvek pole je identifikátor příkazu, ostatní jsou jeho argumenty. Podle identifikátoru lze rozpoznat typ. Typ reprezentuje počet a typ argumentu příkazu. Například typ ”var_symb_symb”znamená, že příkaz má tři argumenty: jeden identifikátor proměnné a dva symboly. Špatný počet argumentu vede na chybu 23. Pak se kontrolují jednotlivé argumenty (a zároveň se vytváří příslušné XML elementy). Kontrola správnosti argumentů je implementovaná v podobě konečných automatu. Špatný formát argumentu vede na chybu 23.

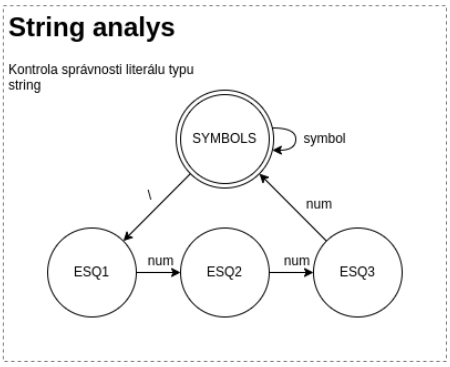
Jakmile všechny řádky programu jsou zpracované, XML dokument se převádí na řetězec a odesílá se na standardní vystup STDOUT.

2 Konečné automaty použité v parseru

V parseru se používají tři různé konečné automaty: Var analys, Name analys a String analys.







3 Návrh a implementace II. části projektu

Implementace je založena na dvou velkých třídách Program a Interpret a a dvou menších, pomocných třídách Value a Frame. Kvůli takovému přístupu je možná relativně snadná realizace algoritmů pro zpracování jednotlivých instrukcí a pro ukládání dat. Zpracování argumentů je implementováno pomocí knihovny ’argparse’.

3.1 Třída Program

Třída Program je určena pro zpracování XML reprezentace programů a pro kontrolu správnosti této reprezentace. Ona také obsahuje několik důležitých metod, které vytváří pomocné struktury. Například, ona vytváří slovník typu ’order: instruction’ a seznam s uspořádaným pořadím. To povoluje snadno vyhledávat samotné instrukce podle jejich pořadí. Kromě toho, metoda této třídy skenuje celý program, vyhledává všechny návěští a vytváří slovník typu ’label: order’, což usnadňuje práci se skoky. Ještě jednou důležitou funkcí třídy Program

je poskytování vhodného rozhraní pro manipulace atributy instrukcí a jejích argumenty. Například, metoda get_argument_type vrátí typ určitého argumentu určité instrukce podle její pořadí.

3.2 Třída Interpret

Třída Interpret je nejdůležitější část celého programu. Ona obsahuje metody pro spravování jednotlivých instrukcí včetně ošetřování všech chyb. Kromě toho, ona obsahuje argumenty pro ukládání dat programu: globální rámec, zásobník lokálních rámců, případný dočasný rámec, zásobník volání, iterátory pro pohyb programem a statistiky interpretu. Součástí třídy také jsou metody pro manipulace s těmito daty. Například, metoda defin_frame určuje rámec, ve kterém se nachází proměnná podle její jména, metoda check_var_init ověřuje, zda už byla proměnná inicializovaná, set_var_value ustanoví novou hodnotu proměnné a tak dál. Hlavní metodou je process_program, která iterativně prochází všemi instrukcí a provádí je. Pro každou instrukci existuje odpovídající metoda, která obsahuje speciální algoritmus. Například, metoda MOVE ověřuje, jestli proměnná var byla definována, pak ověřuje symbol (v případě že je také proměnná, ověřuje její definování a inicializování) a provádí přesun hodnoty.

3.3 Pomocné třídy

Třída Value obashuje dva atributy: type a value. Type reprezentuje typ hodnoty, value - samotnou hodnoty. Pokud jeden z těchto atributu je None, pak hodnota není inicializována. Třída také obsahuje rozhraní pro manipulaci s hodnotou. Objekty třídy Value jsou základem pro ukládání proměnných a také jsou prvky na zásobníku.

Třída Frame obsahuje jenom jeden atribute vars, který je slovníkem pro ukládání proměnných. Tento slovník je typu 'name: Value'. Třída obsahuje pohodlné rozhraní: umožňuje ukládán nové proměnné, vrátí informaci o proměnných a ověřuje jejích existenci.

3.4 Diagram tříd