Implementační dokumentace k projektu do IPP 2017/2018
Jméno a příjmení: Patrik Holop
Login: xholop01

1 Parse.php

1.1 Kontrola parametrů

Kontrola parametrů probíhala pomocí postupného procházení seznamu parametrů a nastavovaním značek (proměnné typu bool pro každý povolený parametr), když sa daný parametr už našel. Jestli se narazí na parametr, který není povolen nebo už byl nalezen, skript končí chybou. Stejně tak se stane po doparsování parametrů porovnáváním nalezených parametrů. Jestli nastala určitá nepovolená kombinace (např. help s jinými parametrami).

1.2 Kontrola vstupního programu

Vstupní program je postupne parsován řádek po řádku. Po načtení řádku ze vstupu se nejdříve odstraní všechny komentáře tak, že se najde první výskyt znaku # a zbytek retězce počínaje daným znakem se smaže. Jestli po tomto kroku řetězec obsahuje pouze bílé znaky, pokračuje sa dalším řádkem. Interní dvoustavový konečný automat určuje, jestli se načítá hlavička vstupního programu nebo instrukce. V prvním případě se řetězec převede na malé znaky a porovná se s ".ippcode18". Jestli je validní, dále se již předpokládají instrukce.

Instrukce je spracována tak, že se nadbytečné bílé znaky odstrání a následně je rozdělena na pole podřetězců s mezerou jako oddělovačem. První prvek, instrukční kód, je převeden na velké znaky a porovnán v interní strukturou instrukcí. Ta je reprezentována asociativním polem tvaru ("Název instrukce": pole parametrů instrukce). Jestli se instrukce v poli nenachází, nastane chyba. V opačném případe se kontrolují parametry instrukce pomocí získaného pole. Toto pole parametrů různých délek obsahuje prvky reprezentující typy možných parametrů, např. `VARIABLE`, `SYMB`, `LABEL`, pokud instrukce parametry nemá, je prázdné. Podle délky tohoto pole ze zjistí, jestli instrukce obsahuje správný počet zadaných parametrů a pak dojde ke kontrole parametrů podle očekávaného typu v daném poli.

Na kontrolu jednotlivých typů byly doporučeny regulární výrazy, ale abych byl schopen vypsat co nejpřesnejší chybovou hlášku v případě chyby, kontrola se provádí manuálne. V případe proměnné se název rozdělí pomocí znaku `@` na rámec (ten je porovnán s `TF`, `GF` a `LF`) a jméno. Skript obsahuje množinu speciálních znaků, které může jméno obsahovat. První znak jména je kontrolován samostatně, kvůli speciálním podmínkám. Dále se iteruje přes znaky řetězce a kontroluje se jejich validita. V případě `LABEL` se volá stejná funkce, která kontroluje jméno proměnné. `SYMB` se kontroluje tak, že zavolá funkce kontrolující proměnnou a funkce kontrolující konstantu (funguje podobně jako kontrola proměnné, pouze s jinými pravidly určujícími kontrétní typ konstanty), pokračuje se podle toho, která z nich skončila úspěchem. `TYPE` je vyhledán v poli retězců obsahujícím jednotlivé typy. Řetězce jsou kontrolovány konečným automatem, který v případě, že narazí na znak '\', načte následující 3 znaky (jestli je řetězec kratší, program končí chybou) a zjistí, jestli je jedná o číslice.

V případe, že se zkontrolovala celá instrukce a její parametry a všechny kontrolní funkce skončili úspěchem, vygeneruje se element `instruction` a přída se do aktuální XML struktury. Po kontrole celého vstupu se výsledný XML dokument vypíše.

1.3 Rozšíření

`STATP`: Pomocí značek při kontrole parametrů se zjistí, které statistiky se mají kontrolovat a v jakém pořadí. Pomocí značek se také zjistí případná duplicita a nepovolená kombinace. Počet komentářů se zjistí z počtu řádků obsahujúcích znak `#`, počítadlo počtu instrukcí se zvyšuje po přidání vygenerované instrukce do XML struktury. Skript také kontroluje, jestli byl zadaný název souboru validní.

2 Interpret.py

Kontrola parametrů u tohoto skriptu probíha stejně jako u skriptu `parse.php`.

2.1 Kontrola vstupního XML souboru

Všechny chyby v tomhle skriptu jsou spravovány pomocí třídy `ErrorManagement`, která obsahuje interní slovník převádející popis chyby na příslušný návratový kód a dvě statické metody na výpis použití programu a ukončení s chybou a popisem.

Knihovna `xml.etree` obsahuje metodu parse, která načítá vstupní soubor a v případě, že nebyl v korektním tvaru, vrátí vyjimku. Jestli nastala, program se ukončí s chybou. V opačném případě, je tedy vstupní xml soubor validní, je kontrolována validita vzhledem k jazyku ippcode18. Každý element v načteném jazyce má svůj tag, atributy a případné podelementy. Samostatně je kontrolávána hlavička kořenového elementu, ktorá musí obsahovat tag `program`. Povolené atributy `language`, `name` a `description` jsou uloženy v pomocním poli a jestli se nalezený atribut v tomto poli nenachází, případně je toto pole příliš dlouhé, končí program s chybou. Povinnost atributu `language` se kontroluje dodatečne.

Dále jsou všechny instrukce (podelementy kořenového elementu) načteny do interní struktury interpretu a je kontrolován jejich operační kód a pořadí. Pak jsou z interní struktury načteny návěští před samotným interpretováním programu spolu s pořadím instrukce, kde se nacházeli, aby se zapezpečilo správné provádění skokových instrukcí. Ukazatel na index provádené instrukce a nastaví na 0 (první instrukce) a spustí se provádení programu.

Každá instrukce jazyku IppCode18 je reprezentována jednou funkcí interpretu. Interpret pak obsahuje asociativní pole mapující název operačního kódu přímo na ukazatel na danou funkci, která se rozezná z interní struktury a následně se zavolá správná funkce pro danou instrukci. Na počátku provádění každé instrukce se zavolá funkce na kontrolu parametrů, která dostane samotnou instrukci a seznam očekávaných typů ("var", "symb"..). Ta kromě počtu parametrů kontroluje také všechny proměnné, konstanty atd. stejným způsobem, jako tomu bylo v skriptu `parse.php`. Nalezené řetězce se uchovávají v paměti v původním stavu a přílušné escape sekvence jsou převádeny na znaky až v příslušných funkcích, které to vyžadují (např. pri výpisu). Jestli má instrukce speciální požadavky na typ parametrů, jsou kontrolovány dodatečne ve funkci instrukce.

Rámce jsou řešeny pomocí slovníku obsahujícího jejich názvy, kterého hodnoty jsou seznamy slovníku. Seznamy jsou použity pro snadný přístup k poslednímu prvku (použito při lokálním rámci), jednotlivé slovníky pak mapují názvy proměnných na jejich hodnoty. Proměnné jsou inicializovány na hodnotu `None`, aby případná práce s nimi vyhodila výjimku, která bude odchycena jako práce s neinicializovanou proměnnou.

Skokové instrukce v případe skoku jednoduše zjistí pozici návěští z interního slovníku podle jeho názvu, jestli se tam takové návěstí nenachází, dojde k chybě.

2.2 Rozšíření

`FLOAT` - Podpora pro typ float se zajistila přidáním tohoto typu do všech kontrolních a aritmetických funkcí. Hodnota při přirazení je získána pomocí funkce `float.fromhex()`, která při špatném tvaru hodnoty vyhodí výjimku. Stejně se pro výstup použila funkce `float.hex()`. Typové funkce pracují na úrovni přetypování v rámci jazyka Python.

3 Test.php

Kotrola parametrů v testovacím skriptu probíhá stejně jako v předešlých skriptech. Testovací skript pomocí funkce `scandir` rekurzívne projde zadaný případně výchozí adresář. Tahle funkce nepodporuje jména a cesty, které nekončí znakem `'/'`, proto se musí v případe potřeby upravit. Jestli narazí na soubor, zjistí, jestli má příponu `.src`. Jestli ano, uloží název souboru bez přípony a dále hledá v daném adresáři zbylé soubory s daným názvem. Pak zkontroluje, které z typů testovacích souborů se našli a zbylé dogeneruje. Jestli se jednalo o adresář, přidá se jeho název do aktuální prohledávané cesty a funkce se rekurzivně zavolá pro nalezený adresář.

Pro každou takhle nalezenou testovací sadu se spustí vykonávávaní testu pomocí systémových volání funkcí exec. Chybové hlášky a výstupy jednotlivých volání jsou přesměrovány do dočasných souborů, které jsou po vykonání testu smazány. Všechny informace se ukládají do interní struktury, která obsahuje informace o cestách k souborům, návratové kódy, výsledky porovnání pomocí příkazu diff a jeho shoda s očekávaným návratovým kódem. Z téhle informativní struktury se následně vygeneruje celá výstupní HTML stránka. Pro generování je použita stejná knihovna, která generuje XML dokument v skriptu `parse.php`, jen s pozměněnými parametry.