Dokumentace úlohy JMP: Jednoduchý makroprocesor v jazyce Python 3 do IPP 2013/2014
Jméno a příjmení: Michal Kozubík
Login: xkozub03

1 Popis řešení úlohy

Pro řešení registrovaného zadání projektu (implementace jednoduchého makroprocesoru) jsem zvolil řešení založené na objektově orientovaném paradigmatu.

1.1 Kontrola parametrů

Skript nejprve provádí kontrolu načtených parametrů. Vzhledem k počtu přípustných parametrů a odlišné funkčnosti modulů, které jsou pro tuto funkci dostupné přímo v pythonu, než je požadovaná funkčnost jsem se rozhodl kontrolu parametrů realizovat přes jednoduchý cyklus a sadu regulárních výrazů, díky kterým je analýza přepínačů jednoduše realizovatelná. Zároveň probíhá kontrola, zda-li nebyl některý parametr zadán vícekrát nebo v chybném tvaru a jestli nejsou přítomny i parametry, které skript nezná.

1.2 Čtení vstupu

Poté je načten celý obsah zadaného souboru (případně je před vstup umístěn ještě text u přepínače **`cmd`**) do paměti jako jeden řetězec. Následně se předá řízení konečnému automatu, který jej čte znak po znaku a kontroluje, zda-li nenarazil na nepovolený symbol, znak značící počátek bloku a nebo volání makra. Pokud se jedná o povolený znak a nejde o volání makra nebo začátek bloku, uloží se do výstupního bufferu.

Je-li načten znak počátku bloku, přepne se čtečka (třída **`reader`**) do režimu načítání bloku, přičemž je dle specifikace funkčnost řídících znaků ignorována. Po načtení celého bloku (najde se párový ukončovací znak k otvíracímu) je obsah bloku přenesen do výstupního bufferu.

Při načtení názvu makra probíhá kontrola, existuje-li takové makro. Pokud ne, skončí se chybou. V opačném případě je z tabulky maker zjištěn počet argumentů, které přijímá a čtečka se je pokusí načíst a následně zavolá expanzi volaného makra. Tato expanze je poté připojena na aktuální pozici vstupu, takže bude podrobena analýze čtečkou, která se přepne zpět do základní čtecí fáze.

1.3 Makra

Pro realizaci maker je více než vhodný objektově orientovaný přístup. Každé makro je objekt obsahující různé informace (počet přijímaných parametrů atp.). Zároveň v sobě nese i tělo jeho expanze. Při jeho vyvolání jsou v něm nalezeny výskyty názvů přijímaných argumentů a jsou nahrazeny hodnotami, které mu byly čtečkou předány.

Výjimkou jsou vestavěná makra, která mají pevně naprogramovanou funkčnost. Respektive jsou to klasická makra, která si však nesou informaci (bool hodnotu) o tom, že mají některou z těchto vestavěných funkcí a je pro jejich expanzi zavolána jiná funkce než pro uživatelem definovaná makra.

1.4 Tabulka maker

Tabulka maker je implementována jednoduchým slovníkem. Ten je indexován jejich názvy a jako hodnoty jsou přímo makra. Před samotným čtením vstupu jsou do této tabulky uložena vestavěná makra.

Díky této implementaci není problém, aby i uživatelem definovaná makra získaly funkčnost některých vestavěných maker, jelikož vestavěné makro let funguje tak, že položce indexované názvem měněného makra přiřadí hodnotu druhého makra. Výjimkou je samozřejmě redefinice makrem **`null`**, kdy je druhé makro odstraněno z tabulky atp.

1.5 Makro def

Makro pro definování nových maker dostane 3 argumenty – název nového makra, seznam přijímaných parametrů a tělo. Nejdříve vytvoří novou instanci v tabulce maker, následně získává parametry ze seznamu a předává je novému makru pomocí metody, která je zařadí do jeho přijímaných argumentů a jako poslední mu předá samotné tělo určené k expanzi.

1.6 Makro set

Makro přijímá pouze 1 parametr nastavující ignorování bílých znaků a na základě hodnoty tohoto parametru nastaví globální proměnnou, na kterou bere zřetel čtečka.

1.7 Makro let

Funkčnost tohoto makra byla zmíněna už při popisu tabulky maker. Vyjma kontrol povolených přepsání maker atp. pouze nastaví hodnotu na indexu přepisovaného makra hodnotou indexovanou názvem druhého makra.