Dokumentace úlohy JMP: Jednoduchý makroprocesor v Python 3 do IPP 2013/2014
Jméno a příjmení: David Kovařík
Login: xkovar66

Návrh a analýza požadavků

Jelikož jazyk Python 3 podporuje různá programovací paradigmata, bylo při návrhu celého programu možno využít těch nejlepších aspektů některých z nich. Přístupový bod programu (spoštěný soubor) je vytvořen jako skript určený ke zpracování argumentů příkazové řádky, nastavení a inicializaci kontrolních struktur a řízení jednotlivých fází převodu vstupního souboru na požadovaný výstup.

Skript ovšem využívá několika modulů, které obsahují jak definici funkcí, tak tříd. Skutečně tedy dochází ke kombinaci několika paradigmat.

Zpracování argumentů příkazové řádky

Aby nebylo nutné provádět kontrolu zadaných argumentů „hluboko“ v programu, je při jeho spuštění inicializována konfigurační struktura, která zapouzdřuje tyto argumenty a další informace z nich odvozené. Při zpracování argumentů se vychází z předpokladu, že argument pasivně zasahuje do konfigurace programu, ikdyž nebyl explicitně zadán při spouštění. V takovém případě je mu přiřazena implicitní hodnota, díky čemuž není nutné provádět opakované kontroly jejich hodnoty.

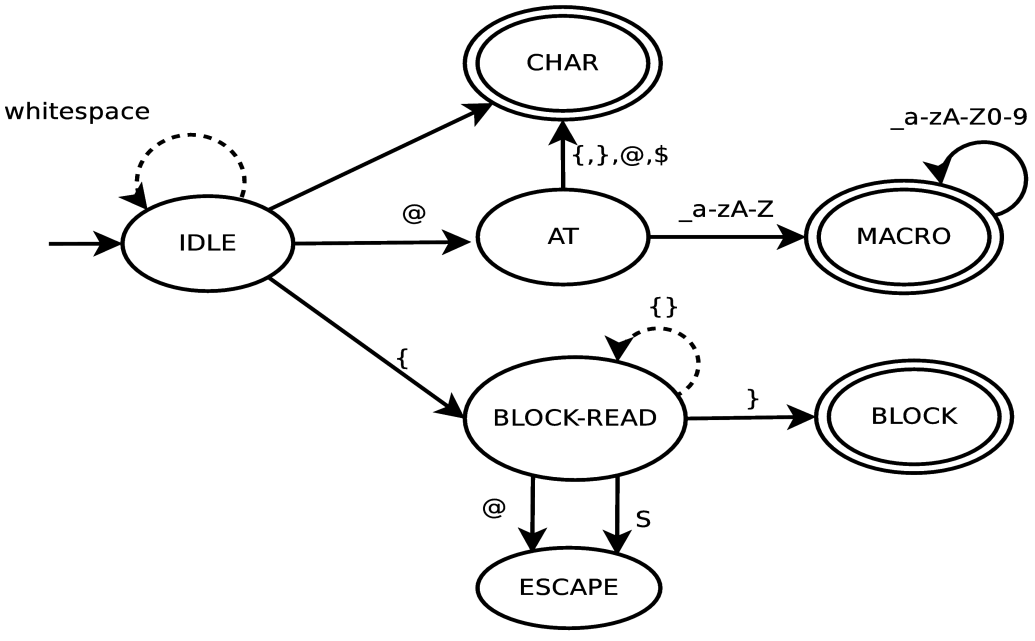
Zpracování zdrojového souboru

Pro reprezentaci zdrojového souboru slouží třída *ExtendableString*, která představuje řetězec, ve kterém je možné určitou část textu nahradit jinou. Každá instance obsahuje čtecí hlavu, která ukazuje na znak, který má být přečten jako další. Po každém čtení je hlava posunuta na následující znak. Po vytvoření ukazuje čtecí hlava na první znak. Jakmile hlava dosáhne konce souboru, vrací prázdnou hodnotu (v jazyce Python hodnotu *None*). Je tedy možné si všimnout jakési podobnosti se standardní funkcí *fgetc* jazyka C.

Lexikální analýza

Instance třídy *Scanner* poskytuje rozhraní lexikální analýzy zdrojového souboru. Majoritním úkolem je schopnost načíst jednu lexikální jednotu (token) ze souboru. Lexikální analyzátor rozlišuje tři druhy lexikálních jednotek. Jedná se o jeden znak, jméno makra či blok. Kvůli specifikaci vstupního jazyka (především pak pravidel pro načítání bloků) nebylo možné použít deterministický konečný automat. Použitým modelem je tedy určitým způsobem degradovaný zásobníkový automat, kdy místo zásobníku vystupuje počítadlo, které se podle pravidel načítání bloku inkrementuje nebo dekrementuje.

Schéma automatu je zobrazeno na přiloženém obrázku. Hrany znázorněné přerušovanou čarou jsou podmíněny argumenty programu, nebo stavem počítadla.



Poznámka: symbol 'S' značí vstupní abecedu

Pokud je načtenou lexikální jednotkou znak, je opsán na výstup. Stejně tak je opsán celý načtený blok, ovšem uvnitř bloku platí jiná pravidla pro speciální znaky. Postup při načtení názvu makra je popsán níže.

Reprazentace maker

Pro každý typ makra existuje samostatná třída, která jej reprezentuje. Instance maker jsou pod svým názvem uloženy v tabulce maker. Všechna makra mají společnou bázovou třídu, která deklaruje jejich rozhraní. Samotná expanze makra je provedena vyvoláním metody *expand(...)* s požadovanými argumenty. Každý typ makra je v tabulce maker reprezentováno pomocí právě jedné instance odpovídající třídy. Při vícenásobném výskytu jednoho typu makra je pak v tabulce maker kopírována pouze jejich reference (např. při vytváření aliasů). Přístup k položkám tabulky maker je obdobný jako při přístupu k položkám slovníků.

Tabulka maker je primárně využívána pro nalezení volaného makra a provedení expanze. Obsah a strukturu tabulky mohou měnit pouze některá (vestavěná) makra.

Pokud je na vstupu rozpoznán název makra, je z tabulky načteno. Podle toho, kolik má makro formálních parametrů, tolik je ze vstupu přečteno lexikálních jednotek, které jsou za tyto parametry dosazeny. Následně dojde k samotné expanzi, tedy nahrazení jmen parametrů v těle makra za jejich hodnoty. Výstupem expanze je tedy řetězec. Ten ovšem není opsán na výstup, ale vrácen zpět na vstup. Expandované makro pak na vstupu nahradí celou sekvence jména volaného makra a všech jeho argumentů. Procesor si totiž pamatuje, na jaké pozici čtecí hlava narazila na volání makra a kde skončil jeho poslední parametr.

Převod na výstup

Na základě výše zmíněných postupů je vstup převeden na výstup. Jelikož je expandované makro vloženo zpět na vstup je možné dosáhnout až (teoreticky) nekonečného generování / vyvolávání maker. Pochopitelně by bylo možné omezit počet vnořených volání maker, například z omezením velikosti paměti. Tím bychom ovšem kriticky omezili „vyjadřovací schopnosti“ makroprocesoru a jazyku, který zpracovává.