**Zadanie č. 4b**

**Dopredný produkčný systém**

**Zadanie:**

Úlohou je vytvoriť jednoduchý dopredný produkčný systém, s prípadnými rozšíreniami, napríklad o kladenie otázok používateľovi alebo vyhodnocovanie matematických výrazov.

Produkčný systém patrí medzi znalostné systémy, teda medzi systémy, ktoré so svojimi údajmi narábajú ako so znalosťami. Znalosti vyjadrujú nielen informácie o nejakom objekte, ale aj súvislosti medzi objektami, vlastnosti zvolených problémov a spôsoby hľadania ich riešenia. Znalostný systém je teda v najjednoduchšom prípade dvojica – program, ktorý dokáže všeobecne manipulovať so znalosťami a báza znalostí, ktorá opisuje problém a vzťahy, ktoré tam platia. Znalosti majú definovanú nejakú štruktúru a spôsob narábania s touto štruktúrou – to sa nazýva formalizmus reprezentácie znalostí. Program vie pracovať s týmto formalizmom, **ale nesmie byť závislý od toho, aké konkrétne znalosti spracováva**, inak by to už nebol systém, kde riešenie úlohy je dané použitými údajmi.

Produkčný systém na základe odvodzovacieho pravidla *modus ponens* (pravidlo odlúčenia) odvodzuje zo známych faktov a produkčných pravidiel nové fakty. Ak systém nemá dostatok vstupných údajov, môže klásť používateľovi otázky.  
**Produkčný systém ako program nepozná konkrétne pravidlá ani fakty!** Pozná len formalizmus, v tomto prípade štruktúru pravidiel a faktov a spôsob ich spracovania. Pozná akcie (pridaj, vymaž, ...), ktoré sa môžu vykonávať, lebo tie patria do opisu formalizmu.

**Reprezentácia údajov:**

Údaje načítavam zo súboru, ktorý má prednastavenú štruktúru. Na začiatku sa píšu všetky fakty v tvare (……). Čítanie faktov sa ukončí /\*\*\*/. Potom sa čítajú pravidlá. Pravidlá sú zapísané v tvare :

MENO XXX

AK ((…)(…)(…))

POTOM ((XXXX)(XXXXX))

Algoritmus tento súbor rozdelí a rozparsuje do primeraných polí. Algoritmus dokáže taktiež rozlíšiť premenné. Premenné sa zaznačujú ?XXX. Tieto premenné dokážem potom nahradiť vhodnými slovami, údajmi.

**Opis fungovania algoritmu:**

Algoritmus funguje na princípe modus ponens. Z faktov a pravidiel vytváram nové akcie, ktoré potom môže mazať, pridávať nové fakty alebo vypisovať správy na konzolu. Každé pravidlo rozdelím na časti (podľa zátvoriek) a skúšam na každú túto časť fakty. Najskôr skontrolujem, či pravidlo a fakt majú rovnakú dĺžku. Ak nie tak viem s istotou povedať, že to nie je pravdivé. Ak majú rovnakú dĺžku, tak potom porovnávam časť po časti. Ak narazím na premennú, tak zisťujem či pre danú premennú už nemám nejakú hodnotu. Ak nie tak si tam uložím aktuálnu a ak áno tak porovnávam. Ak sú rovnaké tak môžem pokračovať, ak nie sú rovnaké tak pokračujem na ďalší fakt. Tento proces budem opakovať cyklicky až kým sa nedostanem na koniec pravidla. Tento proces budem opakovať pre každé pravidlo a každý fakt. Potom všetky tieto akcie prefiltrujem (odstránim všetky nepoužiteľné akcie) a vykonám prvú akciu ktorá ostala. Ak nezostala žiadna akcia, tak končím a vypíšem správy.

**Opis fungovania algoritmu po krokoch:**

1. Načítam súbor, rozdelím si ho na fakty a pravidlá a rozparsujem

2. V cykle prechádzam postupne pravidlá. Vyberiem nasledujúce pravidlo, rozdelím ho na malé časti a začnem do prvej časti dopĺňať všetky fakty

3. Potom rekurzívne pokračujem až dokiaľ sa nedostanem na koniec pravidla, ak sa úspešne dostanem na koniec, môžem vytvorenú postupnosť uložiť do pomocného poľa

4. Tento proces opakujem až kým neprejdem všetky pravidlá

5. Ak som prešiel všetky pravidlá, tak odfiltrujem nepoužiteľné

6. Ak existuje aspoň jedno pravidlo, tak ho vykonám a skočím na krok 2, inak pokračujem ďalej

7. Vypíšem všetky správy a finálnu pamäť faktov a ukončím

Algoritmus som vytvoril tak, aby nebol viazaný na jedinú bázu pravidiel a faktov. Jednoducho dokážem doplniť informácie do premenných, ktoré sú definované „?X“. Taktiež dokáže algoritmus rozoznať akcie : „**pridaj**“, „**vymaz**“, „**sprava**“ a špeciálne pravidlo „**<>**“. **pridaj** pridá nový záznam do pamäte faktov, **vymaz** vymaže záznam z pamäte faktov a **sprava** vypíše správu na konzolu. Špeciálne pravidlo **<>** znamená že sa dva prvky nesmú rovnať.

**Opis objektov:**

Rules – objekt obsahuje všetky informácie o pravidle. Ukladá sa tam meno, predpoklad a dôsledok pravidla.

**Opis funkcií:**

**checkRec()** – Funkcia rekurzívne kontroluje pravidlo postupne doprava. Vždy dosadí všetky možné fakty do pravidla, a ak nejaké pravidlo platí, tak sa predpokladá je môže byt pravdivé a posunie sa ďalej v rekurzí. Využívam tu predpoklad, že keď predchádzajúca časť pravidla platila a aj toto pravidlo je splnene, tak to platí. Taktiež tu kontrolujem ci sa dane pravidlo nevyhodnocuje ako posledné "<>". Ak áno, tak sa jeho pozícia odloží do poľa LaterUse.

**FilterMemory()** – Funkcia filtruje pomocnú pamäť. Ak je akcia neplatná tak je vymaže. Neplatná je ak :

\* Chcem pridať fakt, ktorý sa tam už nachádza a neobsahuje žiadny iný valídny krok,

\* Chcem odstrániť fakt, ktorý sa tam nenachádza a neobsahuje žiadny iný valídny krok,

\* Chcem len vypísať správu a neobsahuje žiadny iný valídny krok.

**executeFirst()** – V tejto funkcii vykonám akciu. Môj algoritmus vykonáva vždy prvú akciu a zvyšné zahodí. Akciu najskôr rozdelí na dve časti -- {akcia, parametre}. Najskôr sa pozrie, či dokáže danú akciu vykonať (nebudem pridávať duplikát, odstraňovať neexistujúci prvok) a akciu vykoná. Pozná 3 typy akcii

\* pridaj -- pridá fakt do pamäte faktov

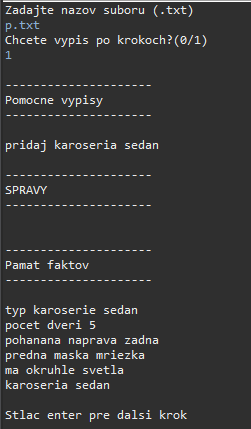
\* vymaz -- výmaze fakt z pamäte faktov

\* sprava -- vypíše správu na konzolu

**checkFacts()** – Funkcia vytvára a vykonáva všetky možne akcie nad pravidlami. Vždy skúsi prvú časť pravidla so všetkými faktami, a ak nejaké "platí", tak ho kontroluje ďalej doprava rekurzívne, pokiaľ sa nedostane až na poslednú časť. Ak každá časť platí, tak sa akcia odloží na vykonanie. Ak sú všetky možne akcie vykonane, tak sa prefiltrujú (odstránia sa všetky nepoužiteľne akcie). Potom ak existuje aspoň jedna použiteľná akcia tak vykoná prvú a zvyšne zahodí. Ak neexistuje žiadna použiteľná akcia, tak sa cyklus aj program ukončia, vypíše sa výpis a pamäť faktov

**returnSplit()** – Funkcia slúži na spracovanie vstupného stringu, rozdelí ich podľa zátvoriek

**Používateľské prostredie:**

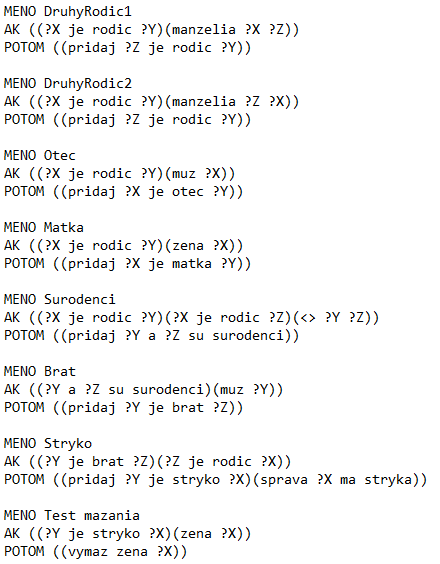


Požívateľské rozhranie jednoducho vypíše všetky dôležité informácie o danom kroku na konzolu. Užívateľ dokáže prejsť na ďalší krok stlačením klávesy enter.

**Testovanie:**

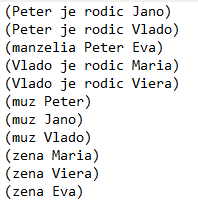
Pri testovaní som skúšal dva rôzne sety pravidiel a skúšal som rôzne fakty. Vždy som si nechal vypisovať priebeh krok po kroku aby som zistil, či program pracuje správne.

**Test – Rodinné vzťahy:**



Pre tieto pravidlá som skúšal pár rôznych začiatočných faktov a kontroloval som výsledky.

1. Ukážkové fakty:



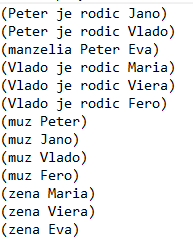
**Výstup:**

Maria ma stryka

Viera ma stryka

Pri týchto faktoch som dostal výpis, ktorý sa zhoduje s ukážkou na stránke.

2. Pridané fakty



**Výstup:**

Maria ma stryka

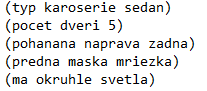
Viera ma stryka

Fero ma stryka

Ďalej som testoval aj inú sadu pravidiel. Testoval som pravidlá pre autá FIAT.

**Test – FIAT**

(nedával som sem pravidlá, pretože ich je veľa)



**Výstup:**

Fiat Croma

**Zhodnotenie:**

Projekt som sa snažil vytvoriť čo najefektívnejšie. Väčšinu dôležitých informácií odkladám do Hashmap aby som urýchlil jej vyhľadávanie. Ukladám si tam dáta, ktoré sú uložené v premenných, polohy premenných, etc. Program je jednoducho rozšíriteľný, kde môžem jednoducho doplniť špeciálne podmienky.