PPJ - Pitanja vezana za ispit

github.com/kotur95/Programming/tree/master/PPJ
08.02.2019.

E-Mail sa odgovorima na pitanja upućenim asistentu Nemanji Mićoviću.

1. Pitanje (Početni simbol za Program)

Interesuje me da li mogu koristiti sledeće pravilo `gramatike` za definisanje `programa` kao niza naredbi, gde je ';' separator (mogao je biti i '\n') ...

Primer 1 (*Moj predlog*):

```
Program : Naredba ';' Program {}
        | Naredba ';'         {}
        ;
```

NAPOMENA: Dakle ako dobro zaključujem, ovde se radi o desnoj rekurziji umesto levoj kao što je rađeno na primerima sa vežbi koje sam bio u prilici da pogledam.

VAŽNO: Kod ovog načina, pravilo mora biti u ovom redosledu. Program -> Naredba ';' Program .. Drugačinji redosled Program -> Program ';' Naredba ne bi radio, pokušaj da izvedeš i videćeš!!

Navešću još neke načine zadavanja programa.

Primer 2 (*Prazno pravilo*):

```
Program : Naredba ';' Program {}
        |                     {}
        ;
```

Primer 3 (*Levo rekurzivno*):

```
Program : Niz_naredbi ';' {}
        ;

Niz_naredbi : Niz_naredbi ';' Naredba {}
            | Naredba                 {}
            ;
```

Primer 4 (*by Blagoje Mirković*):

```
Program : Program Naredba ';' {}
        | Naredba ';'         {}
```

NAPOMENA: Između ostalog pokušavam separator da izbacim iz definicije naredbe, da ga ne bih iznova i iznova pisao kod definisanja svake vrste naredbe (dodela/print ..) pojedinačno.

PITANJE: Da li postoje skrivene mane nekog od ovih načina, u prvi mah sve deluje da radi ok, i da li vi imate neki predlog šta je najbolje ili je svejedno?

SLEDEĆE PITANJE NA SLEDEĆOJ STRANI

2. Pitanje (Garbage collector)

- Da se napravi na vrlo prost način oslobađanja od promenljivih alociranih na hip-u, dat je primer.
- Posle izvršavanja svake linije, ako smo alocirali `tmp` objekte, oslobodimo se njih.
- Oslobađanje memorije se vrši liniju po liniju.

MOTIVACIJA: Automatski `delete`, ne moramo misliti gde treba osloboditi memoriju.

Praktičan (pseudo) primer

```
%{     std::vector <MojTip*> tmps; // Vektori tmp-ova
/* Funkcija koja prima objekat tipa MojTip po vrednosti, alocira ga na
 hip-u, smesta njegovu adresu u vektor tmps i return-uje je*/
MojTip * new_tmp(MojTip kopiraj = MojTip()) {
    MojTip * ptr = new MojTip(kopiraj);
    tmps.push_back(ptr);
    return ptr;
}
/* Funkcija za ciscenje vektora tmp objekata */
void clear_tmps() {
    for (auto& e : tmps) delete e;
    tmps.resize(0);
}
/* Tablica simbola cuva po vrednosti */
std::map <std::string, MojTip> tablica_simbola;       %}
/* Tipovi */
%token <int_type> broj_token
%type <MojTip_type> E
%%
/* ------------------------------------------------------------ *
 *  U Gramatici jos uradimo sledece, sa clear_tmps()           *
 *  nakon svake linije oslobadjamo nepotrebne tmp objekte      *
 * ------------------------------------------------------------ */
Program : Naredba ';' Program { clear_tmps(); }
        | Naredba ';' { clear_tmps(); }
        ;

Naredba : print_token '(' E ')' { std::cout << *$3 << std::endl; }
        ;
/* ------------------------------------------------------------ *
 *  Za din. alokaciju koristimo:  new_tmp umesto new           *
 *  Operacije (+,-) vracaju rezultat po vrednosti!                  *
 * ------------------------------------------------------------ */
E : E '+' E { $$ = new_tmp(*$1 + *$3); }
  | E '-' E { $$ = new_tmp(*$1 - *$3); }
  ;

E : E ',' broj_token { $1->dodaj_na_kraj($3); $$ = $1; }
  | broj_token { $$ = new_tmp(); $$->dodaj_na_kraj($1); }
  ;
```

Primer sa primenjenom idejom kada je `MojTip` tip `List`
https://github.com/kotur95/Programming/blob/master/PPJ/brojevi_jan_2019/parser.ypp

PITANJE: Da li je ok uraditi ovo na ispitu, i vaše mišljenje?

Odgovori na pitanja

Odgovor na pitanje broj 1 (Početni simbol za Program)

Da, mozete dodati karaktere u niz naredbi da napravite separator ili kraj naredbe, ali budite pazljivi sa tim da ne napravite npr da je ';' IZMEDJU naredbi (kao u Pascalu) u odnosu na to da je na KRAJU naredbe.

Sto se tice ispita savetujem da uradite onako kako je Vama najlakse ali i najbrze. Najcesce se ispit sastoji iz dva zadatka pa se potrudite da uradite oba :) Ako krenete ovako to je sasvim u redu, ali najbolje da resite jos neki rok na ovaj nacin za svaki slucaj da bi vam uslo u prste i da se ne desi da smo i Vi i ja prevideli neki problem koji se nije manifestovao ovde, a naravno desava se na ispitu..

Odgovor na pitanje broj 2 (Garbage collector)

Kada cistite temps iz vektora lose je raditi resize nakon toga jer time smanjite nazad alociranu memoriju u vektoru, a posle je ponovo (povremeno) alocirate koristeci push_back. Ono sto Vama treba je clear koji zadrzi memorijski kapacitet, a poizbacuje pokazivace koji su pre toga dealocirani. Naravno, i dalje mozete koristit push_back (on alocira i prosiri sam ako je potrebno). Na taj nacin cete otprilike imati konstantno zauzece memorije tokom duzeg rada programa (konvergirace duzina vektora nekom maksimalnom + eps zivih promenljivih pre nego sto se pozove brisanje...) a smanjite broj alokacija/dealokacija dosta.

Mali savet za kasniji rad, ovo sto zelite da radite (iako je malo pipavo da se koristi u bison-u) zove se shared_ptr. U pitanju je pametni pokazivac na objekat koji vrsi automatsku dealokaciju slicno kao sto to funkcionise u javi koriscenjem brojanja referenci.
https://thispointer.com/learning-shared_ptr-part-1-usage-details/

Pristup koji ste koristili u ovom kontekstu je sasvim ok.

Srdacan pozdrav, Nemanja