

Programski prevodioci

Vežbe 5

Sadržaj

1. Uvod.....	1
2. Rešenja zadataka	1
2.1. Zadatak 1: void tip.....	1
2.2. Zadatak 2: return ;.....	2
2.3. Zadatak 3: for iskaz	3
2.4. Zadatak 4: branch.....	4
2.5. Zadatak 5: switch iskaz	5

1. Uvod

U dokumentu su data rešenja zadataka koji su rađeni na petim vežbama.

2. Rešenja zadataka

Svi zadaci se rešavaju sledećim redosledom:

- Dodati nove tokene na vrh `.y` datoteke.
- Definisati regularne izraze u `.l` datoteci za nove tokene.
- Proširiti gramatiku jezika tako da sintaksno podržava novu konstrukciju.
- Dodati semantičke provere.

2.1. Zadatak 1: `void` tip

U skeneru dodati pravilo za `void` ključnu reč (slično kao `int` i `unsigned`):

```
"void"    { yylval.i = VOID; return _TYPE; }
```

Konstantu `VOID` koja služi kao vrednost simbola, definisati (dodati) u enumeraciji `types` u datoteci `defs.h`:

```
enum types { NO_TYPE, INT, UINT, VOID };
```

Ako se desi situacija da je promenljiva tipa `VOID`, treba ispisati (semantičku) grešku:

```
variable  
: type _ID _SEMICOLON  
{
```

```
...  
if($1 == VOID)  
    err("variable cannot be of VOID type");  
}  
;
```

Slično, za pojam `parameter` (ne sme biti `VOID`):

```
parameter
: _TYPE _ID
{
    ...
    if($1 == VOID)
        err("parameter cannot be of VOID type");
}
```

2.2. Zadatak 2: `return ;`

U prvom delu `.y` specifikacije definisati globalnu promenljivu koja broji koliko je u parsiranoj funkciji bilo `return` iskaza:

```
%{
    int return_count = 0;
%}
```

Proširiti gramatiku jezika tako da omogući `return ;` konstrukciju:

```
return_statement
: _RETURN num_exp _SEMICOLON
{
    if(get_type(fun_idx) == VOID) ①
        err("Function cannot return value");
    else if(get_type(fun_idx) != get_type($2))
        err("incompatible types in return");
    return_count++; ③
}

| _RETURN _SEMICOLON
{
    if(get_type(fun_idx) != VOID) ②
        warn("Function should return a value");
    return_count++; ③
}
;
```

- ① Ako je povratni tip funkcije `VOID`, onda se iz nje ne sme vraćati vrednost (greška).
- ② Ako je povratni tip funkcije `INT` ili `UINT`, onda ona treba da vrati vrednost. Ako postoji samo `return ;` treba prijaviti upozorenje. Da bi se prijavilo upozorenje, treba koristiti `warn` makro umesto `err` makroa.
- ③ Svaki put kada se isparsira neka varijanta `return` iskaza, inkrementira se `return_count`.

Na kraju funkcije, ako je `return_count` jednak `0`, znači da u parsiranoj funkciji nije bilo `return` iskaza. Ako je funkcija `INT` ili `UINT`, ovo predstavlja grešku, pa je treba prijaviti:

```

function
: type _ID
{
    ...
}
_LPAREN parameter _RPAREN body
{
    ...

    if( (return_count == 0) && (get_type(fun_idx) != VOID) ) ①
        warn("Function should return a value");
    return_count = 0; ②
}
;

```

- ① Ako u telu funkcije nije postojao nijedan `return` iskaz, a funkcija nije `VOID`, treba prijaviti *warning*.
- ② Nakon provere, brojač obavezno treba resetovati, da bi mogao ponovo da se iskoristi za narednu funkciju.

2.3. Zadatak 3: `for` iskaz

Dodati tokene `_FOR`, `_INC`. Nakon toga izmeniti gramatiku na sledeći način:

```

for_statement
: _FOR _LPAREN _TYPE _ID
{
    int i = lookup_symbol($4, PAR|VAR);
    if(i != -1)
        err("redefinition of variable '%s'", $4);
    else
        $<i>$ = insert_symbol($4, VAR, $3, 1, NO_ATR);
}
_ASSIGN literal
{
    if($3 != get_type($7))
        err("incompatible types in assignment");
}

```

```

    _SEMICOLON rel_exp _SEMICOLON _ID
    {
        $<i>$ = lookup_symbol($12, VAR);
        if($<i>5 != $<i>$)
            err("wrong var for increment");
    }
    _INC _RPAREN statement
    {
        clear_symbols($<i>5); ①
    }
    ;

statement
: compound_statement
| assignment_statement
| if_statement
| return_statement
| for_statement ②
;

```

- ① Nakon kraju fora moramo izbrisati iterator iz tabele simbola kako bi naredni for mogao da definiše iterator sa istim imenom
- ② Ukoliko se novonapravljeni pojam `for_statement` ne ubaci u `statement`, Bison će prijaviti upozorenje: `1 nonterminal useless in grammar`. Ovo upozorenje znači da postoji pojam u `.y` datoteci koji stoji sam za sebe i nema nikakve veze sa ostatkom gramatike, što znači da je njegovo postojanje beskorisno.

2.4. Zadatak 4: `branch`

Dodati tokene `_BRANCH`, `_FIRST`, `_SECOND`, `_THIRD`, `_OTHERWISE`, `_COMMA` Izmeniti gramatiku na sledeći način:

```

branch_statement
: _BRANCH _LPAREN _ID
{
    int idx = lookup_symbol($3, VAR|PAR);
    if(idx == NO_INDEX)
        err("%s' undeclared", $3);
}
_SEMICOLON literal
{
    int idx = lookup_symbol($3, VAR|PAR);
    if(get_type(idx) != get_type($6))
        err("incompatible types...");
}

```

```

_COMMA literal
{
    int idx = lookup_symbol($3, VAR|PAR);
    if(get_type(idx) != get_type($9))
        err("incompatible types...");
}
_COMMA literal
{
    int idx = lookup_symbol($3, VAR|PAR);
    if(get_type(idx) != get_type($12))
        err("incompatible types...");
}
_RPAREN _FIRST statement _SECOND statement _THIRD statement _OTHERWISE statement
;

statement
: compound_statement
| assignment_statement
| if_statement
| return_statement
| branch_statement
;

```

2.5. Zadatak 5: switch iskaz

Dodati tokene _SWITCH, _CASE, _BREAK, _DEFAULT, _COLON a zatim izmeniti gramatiku:

```

%{
    int switch_id_index = 0;
    int case_count = 0;
    int case_array[100];
}%

...

switch_statement
: _SWITCH _LPAREN _ID
{
    if( (switch_id_index = lookup_symbol($3, VAR)) == -1)
        err("'s' undeclared", $3);
}
_RPAREN _LBRACKET case_statements default_statement _RBRACKET
{
    case_count = 0; //ponisti sadrzaj niza
}
;

case_statements
: case_statement

```

```

| case_statements case_statement
;

case_statement
: _CASE literal _COLON
{
    // provera jedinstvenosti konstanti
    int i = 0;
    while(i < case_count) {
        if($2 == case_array[i]) { //ako takva konstanta vec postoji u nizu
            err("duplicated constant in case");
            break;
        }
        i++;
    }
    if(i == case_count) { //ako nije duplikat
        case_array[case_count] = $2; //ubaci konstantu u niz
        case_count++;
    }

    //provera tipa konstante
    if(get_type($2) != get_type(switch_id_index))
        err("wrong type of constant");
}
statement break_statement
;

break_statement
: /* empty */
| _BREAK _SEMICOLON
;

default_statement
: /* empty */
| _DEFAULT _COLON statement
;

```