Part I

miniC jezik

Chapter 1

miniC jezik

miniC je podskup programskog jezika C. Dakle, miniC programe je moguće kompajlirati regularnim C kompajlerom. miniC je nastao odabirom osobina i koncepata C jezika koji su interesantni za kurs implementacije kompajlera. Međutim, ove osobine su uzete u određenoj meri, tako da olakšaju implementaciju jezika. Autori su se uzdržali od mnogih karakteristika C-a koji nepotrebno komplikuju implementaciju jezika, a koji, u edukativnom smislu, ne doprinose značajno.

U ovom delu, dat je opis miniC jezika (njegova sintaksa i semantika) i primer programa. Kompletna gramatika miniC jezika je navedena na kraju prvog dela (deo 1.4).

1.1 Leksika

Leksika se bavi opisivanjem osnovnih gradivnih elemenata jezika. Za prirodne jezike, to su reči, a za programske jezike, to su simboli. String simbola se zove leksema. Skup leksema i pravila njihovog formiranja predstavljaju leksiku jezika.

U miniC jeziku postoji nekoliko vrsta simbola. To su identifikatori i označeni i neoznačeni celobrojni literali.

Identifikator u miniC jeziku je malo pojednostavljen u odnosu na C jezik, pa se sastoji od malih slova, velikih slova i cifara, i ne sme započinjati cifrom. Primeri ispravnih miniC identifikatora su:

```
a, A, abc, number, Number, MyNumber, num2, num2str
```

a neispravnih: `3a`, `_x`, `my_number`.

Celobrojni označeni literal se sastoji od predznaka plus ili minus iza kojeg sledi jedna ili više cifara, dok se neoznačeni celobrojni literal sastoji od jedne ili više cifara iza kojih sledi malo ili veliko slovo "u". Primeri literala su:

```
0, -123, 12345
5u, 1234U
```

1.2 Sintaksa

Sintaksa opisuje skup pravila za kombinovanje simbola u ispravne jezičke konstrukcije. Sintaksa se formalno opisuje gramatikom (vidi deo 1.4). U objašnjenjima sintaksnih konstrukcija, u zagradama je navedeno ime pojma u gramatici koje odgovara toj konstrukciji.

1.2.1 Program

Najmanji miniC program ima bar jednu funkciju.

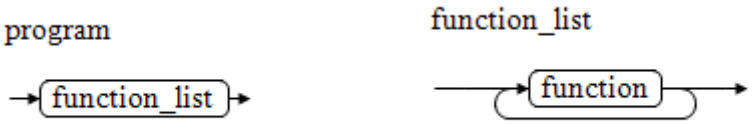

Figure 1.1: Sintagram programa i liste funkcija

Da bi program mogao da započne izvršavanje potrebna mu je `main()` funkcija. Povratna vrednost funkcije `main()` je celobrojnog tipa (`int`).

```
int main() {
  ...
}
```

1.2.2 Tipovi

miniC jezik podržava samo `int` i `unsinged` tipove podatka, tj. označene i neoznačene celobrojne vrednosti.

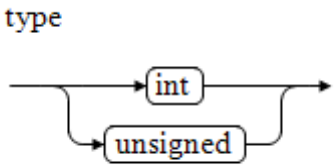

Figure 1.2: Sintagram tipa podatka

1.2.3 Promenljive

Deklaracija promenljive (*variable*) sadrži ime tipa i ime promenljive a završava se separatorom ";".

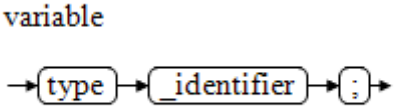

Figure 1.3: Sintagram deklaracije promenljive

U jednoj miniC deklaraciji, moguće je deklarisati samo jednu promenljivu. Da bi se deklarisale dve promenljive, potrebno je napisati dve deklaracije. U primeru, prve tri deklaracije su ispravne, dok poslednja nije:

```
int counter;
int line;
unsigned n;

int counter, line;  //error
```

1.2.4 Funkcija

Zaglavlje miniC funkcije (*function*) se sastoji od tipa povratne vrednosti funkcije, imena funkcije i malih zagrada u kojima se može, a ne mora, navesti (jedan) parametar funkcije.

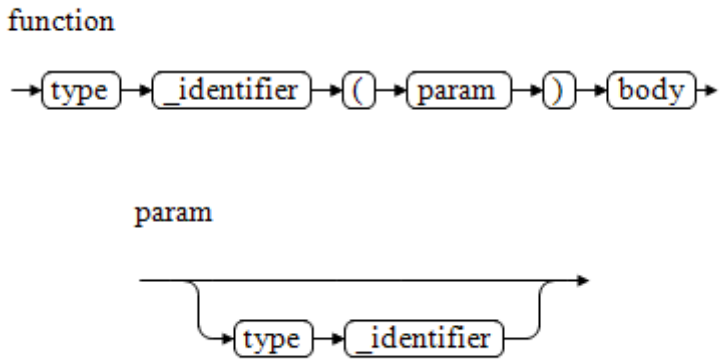

Figure 1.4: Sintagram definicije funkcije i parametra

Primeri sintaksno ispravnih miniC funkcija su dati u primeru:

```
int f() {
  ...
}

int f(unsigned a) {
  ...
}
```

Iza zaglavlja se navode vitičaste zagrade u kojima se piše telo funkcije (*body*). Telo funkcije se sastoji od deklaracija lokalnih promenljivih (*variable_list*) i od iskaza (*stmt_list*), u tom redosledu. Telo može, a ne mora, da sadrži lokalne promenljive. Isto tako, može a ne mora da sadrži iskaze, što znači da funkcija može imati prazno telo. Slede ispravni primeri miniC funkcija:

```
int f() {
}

int f(int a) {
  return a;
}

int f() {
  int x;
}

unsigned f() {
  unsigned x;
  x = 0u;
}
```

Unutar tela funkcije, prvo se navode deklaracije lokalnih promenljivih, pa tek onda iskazi. To znači, da nije ispravno napisati deklaraciju lokalne promenljive posle nekog iskaza:

```
int f() {
  int x;
  x = 0;
  int y;     //error
  y = 0;
}
```

1.2.4.1 Poziv funkcije

Poziv miniC funkcije (*function_call*) se sastoji od imena funkcije i malih zagrada u kojima se može, a ne mora, navesti (jedan) argument.

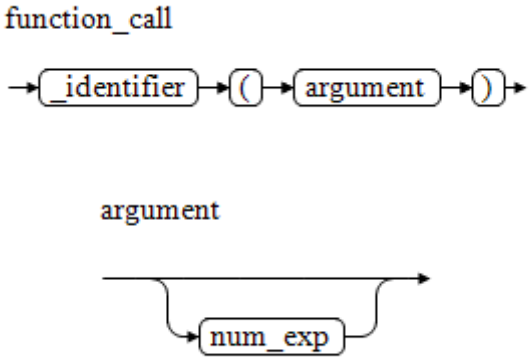

Figure 1.5: Sintagram poziva funkcije

Slede (i ispravni i neispravni) primeri poziva miniC funkcija:

```
unsigned f0() {
  return 0u;
}

int f1(int a) {
  return a + a;
}

int f() {
  unsigned x;
  int y;
  x = f0();
  x = f0(5u);  //error, f0 nema parametar
  y = f1();    //error, f1 ima parametar
  y = f1(3);
}
```

1.2.5 Iskazi

Iskazi koji mogu da se pojave u telu funkcije su: iskaz dodele, `if` iskaz, `return` iskaz i blok iskaza. U iskazima se koriste numerički izrazi.

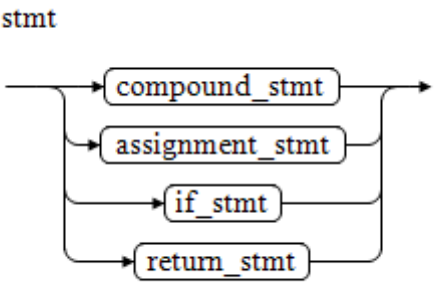

Figure 1.6: Sintagram iskaza

1.2.5.1 Numerički izraz

Numerički izraz (*num_exp*) se gradi od literala, promenljivih i poziva funkcija. Od aritmetičkih operacija, podržano je samo sabiranje i oduzimanje. Numerički izraz se može pisati i u malim zagradama.

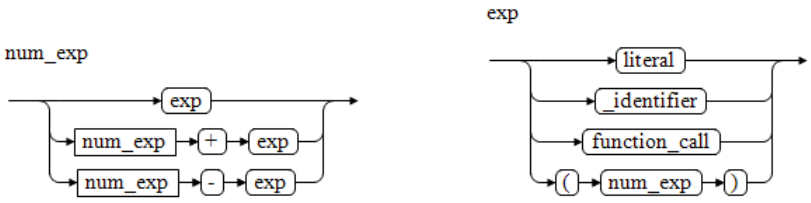

Figure 1.7: Sintagrami izraza

Primeri ispravnih miniC numeričkih izraza su:

```
a
0
line + 1
j + (k - 4)
f()
2 + f()
```

1.2.5.2 Iskaz dodele

Sa leve strane znaka jednako u iskazu dodele (*assignment_stmt*) se može naći promenljiva, a sa desne strane numerički izraz.

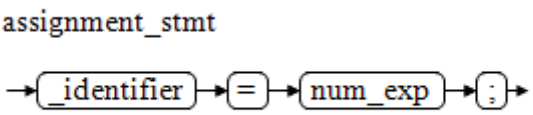

Figure 1.8: Sintagram iskaza dodele

Primeri pravilnih iskaza dodele u miniC jeziku su:

```
a = b;
counter = 0;
line = line + 1;
c = j + (k - 4);
m = f();
```

1.2.5.3 `if` iskaz

U `if` iskazu (*if_stmt*) se prvo navodi ključna reč `if`, zatim uslov u malim zagradama i onda iskaz. Opciono, `if` iskaz može da sadrži i `else` deo koji se sastoji od ključne reči `else` i iskaza. Uslov `if` iskaza, u miniC jeziku, sadrži relacioni izraz. Relacije koje su podržane u miniC jeziku su `<`, `>`, `<=`, `>=`. `==` i `!=`.

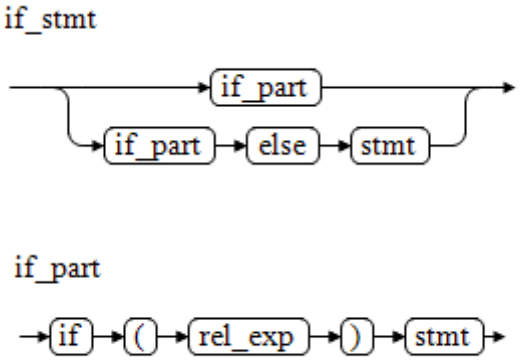

Figure 1.9: Sintagram `if` iskaza i relacionog izraza

Primeri pravilnih `if` iskaza su:

```
if(a < b)
  a = b;

if(a < b) {
  a = b;
}

if(a == b)
  counter = counter + 1;
else
  counter = counter - 1;
```

1.2.5.4 `return` iskaz

`return` iskaz (*return_stmt*) očekuje da se iza ključne reči `return` navede numerički izraz, čija će vrednost biti povratna vrednost funkcije u kojoj se nalazi.

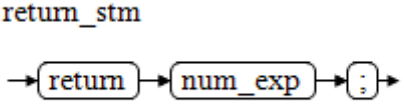

Figure 1.10: Sintagram `return` iskaza

Primeri `return` iskaza u miniC jeziku su dati u sledećem primeru (prva 3 iskaza su ispravna, a poslednji nije):

```
return 0;
return a + 1 - (d + 5);
return f3(a);

return;  //error: return iskaz mora sadrzati izraz
```

1.2.5.5 Blok iskaza

Blok iskaza (*compound_stmt*) čine iskazi napisani u vitičastim zagradama. Blok može biti i prazan.

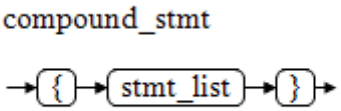

Figure 1.11: Sintagram bloka iskaza

Primeri pravilnih blokova u miniC jeziku su:

```
{ }
{ a = 8; }
{
  if(a == b)
    counter = 0;
  a = 10;
}
```

U bloku se ne može deklarisati promenljiva, pa zato prva linija unutar bloka u sledećem primeru nije nedozvoljena u miniC jeziku:

```
{
  int a;  //error: u bloku se ne moze
          //deklarisati promenljiva
  a = 8;
}
```

1.2.6 Komentari

Komentari u miniC jeziku započinju sa dva znaka *slash* //, a u nastavku, do kraja linije (do znaka \n), sledi tekst komentara. Primer pravilnog komentara:

```
if(a == b)  // tekst komentara
  a = -b;
```

1.2.7 Primer miniC programa

miniC je dosta ograničen programski jezik, ali uprkos svojim ograničenjima može da se koristi za programiranje malih programa. Na primer, miniC program za izračunavanje apsolutne vrednosti označenog broja bi mogao da izgleda ovako:

```
int abs(int i) {
  int res;
  if(i < 0)
    res = 0 - i;
```

```
  else
    res = i;
  return res;
}

int main() {
  return abs(-5);
}
```

1.3 Semantika

Semantika jezika opisuje značenje sintaksno ispravnih konstrukcija jezika. Kod koji je sintaksno ispravan, ne mora biti i semantički ispravan. Na primer, ako se poziva funkcija koja nije prethodno definisana, to predstavlja semantičku grešku. Iako postoje mnogi formalizmi za zapis sematike programskih jezika neretko se za te potrebe u oblasti razvoja kompajlera koriste rečenice prirodnog jezika, tj. neformalni zapis.

1.3.1 Standardni identifikatori

Standardni identifikatori su: rezervisane reči (`int`, `unsigned`, `if`, `else`, `return`) i identifikator `main`. Identifikator `main` je ime funkcije, za koju se podrazumeva da je definisana u izvršivom miniC programu. Izvršavanje miniC programa započinje izvršavanjem `main` funkcije. Definicija ove funkcije izgleda:

```
int main () {
   ...
}
```

Telo `main` funkcije definiše korisnik. Ako telo `main` (kao i svake druge) funkcije ne sadrži `return` iskaz, podrazumeva se da je povratna vrednost funkcije nedefinisana i da do povratka iz funkcije dolazi po izvršavanju poslednjeg iskaza iz njenog tela.

1.3.2 Opseg vidljivosti (*scope*)

Prema opsegu vidljivosti (područje važenja, doseg, vidljivost), identifikatori se razvrstavaju u globalne i lokalne.

Globalni identifikatori su imena funkcija. Oni su definisani na nivou programa (van funkcija). Opseg vidljivosti globalnih identifikatora je od mesta njihove definicije do kraja programskog teksta.

Lokalni identifikatori su imena lokalnih promenljivih i parametara i oni su definisani u okviru funkcija. Opseg vidljivosti lokalnih identifikatora je od mesta njihove definicije do kraja tela funkcije u kojoj su definisani. Znači, svaka funkcija poseduje svoje lokalne promenljive.

Identifikatori mogu biti korišćeni samo iza njihove definicije (to proizlazi iz opsega njihovog važenja).

Može se desiti da se isto ime deklariše u nekoliko ugnježdenih opsega vidljivosti. U tom slučaju, uobičajeno je da se koristi deklaracija imena koja je najbliža datoj upotrebi imena. Primer C iskaza:

```
{
  int x = 1;     //prva promenljiva x
  int y = 2;
  {
    int x = 3;   //druga promenljiva x
    y += x;
  }
  y += x;
}
```

U primeru postoje dve različite promenljive `x`, deklarisane u dva različita opsega vidljivosti. Druga promenljiva `x` je sakrila vidljivost prve. Zato će vrednost promenljive `y` na kraju ovog dela programa biti `6`.

1.3.3 Jednoznačnost identifikatora

Svi globalni identifikatori moraju biti međusobno različiti i svi lokalni identifikatori iste funkcije moraju biti međusobno različiti. Ako postoje identični globalni identifikatori i lokalni identifikatori neke funkcije, tada van te funkcije važe globalni, a unutar nje lokalni identifikatori.

Lokalni identifikatori raznih funkcija mogu biti identični.

Rezervisane reči smeju da se koriste samo u skladu sa svojom ulogom i na globalnom i na lokalnom nivou. Standardni identifikator `main` je rezervisan samo na globalnom nivou.

1.3.4 Provera tipova

Na upotrebu identifikatora utiče njegov tip. Na primer, tip identifikatora s leve strane iskaza pridruživanja određuje tip izraza sa desne strane ovog iskaza (leva i desna strana iskaza pridruživanja moraju imati isti tip).

Tip izraza iz `return` iskaza neke funkcije i tip ove funkcije moraju biti identični.

Tipovi korespondentnih parametara funkcije i argumenata iz njenog poziva moraju biti identični. Argumenti poziva funkcije moraju da se slažu po broju sa parametrima funkcije.

U istom relacionom izrazu smeju biti samo identifikatori istog tipa.

1.4 miniC gramatika

Gramatika miniC jezika je napisana korišćenjem BNF notacije. Vertikalna crta se koristi da ukaže na alternativna pravila i čita se "ili". Koristimo konvenciju da simboli jezika (*terminals*) započinju donjom crtom "_" da bi se razlikovali od pojmova (*nonterminals*).

Simboli (*terminals*)

```
        _letter ::= "a"|"A"|"b"|"B"|"c"|"C"|"d"|"D"
                  |"e"|"E"|"f"|"F"|"g"|"G"|"h"|"H"
                  |"i"|"I"|"j"|"J"|"k"|"K"|"l"|"L"
                  |"m"|"M"|"n"|"N"|"o"|"O"|"p"|"P"
                  |"q"|"Q"|"r"|"R"|"s"|"S"|"t"|"T"
                  |"u"|"U"|"v"|"V"|"w"|"W"|"x"|"X"
                  |"y"|"Y"|"z"|"Z"
         _digit ::= "0"|"1"|"2"|"3"|"4"|"5"|"6"|"7"
                  |"8"|"9"
    _identifier ::= letter ( letter | digit )*
   _int_literal ::= digit +
  _uint_literal ::= digit + ( "u" | "U" )
```

Pojmovi (*nonterminals*)

```
       program ::= function_list
  function_list ::= function
                  | function_list function
      function ::= type _identifier "(" param ")" body
          type ::= "int"
                  | "unsigned"
         param ::=
                  | type _identifier
```

```
          body ::= "{" variable_list stmt_list "}"
  variable_list ::=
                 | variable_list variable
      variable ::= type _identifier ";"
     stmt_list ::=
                 | stmt_list stmt
          stmt ::= compound_stmt
                 | assignment_stmt
                 | if_stmt
                 | return_stmt
  compound_stmt ::= "{" stmt_list "}"
assignment_stmt ::= _identifier "=" num_exp ";"
       num_exp ::= exp
                 | num_exp "+" exp
                 | num_exp "-" exp
           exp ::= literal
                 | _identifier
                 | function_call
                 | "(" num_exp ")"
       literal ::= _int_literal
                 | _uint_literal
  function_call ::= _identifier "(" argument ")"
      argument  ::=
                 | num_exp
       if_stmt ::= if_part
                 | if_part "else" stmt
       if_part ::= "if" "(" rel_exp ")" stmt
       rel_exp ::= num_exp "<" num_exp
                 | num_exp ">" num_exp
                 | num_exp "<=" num_exp
                 | num_exp ">=" num_exp
                 | num_exp "==" num_exp
                 | num_exp "!=" num_exp
   return_stmt ::= "return" num_exp ";"
```

Razmak i kraj linije imaju funkciju separatora simbola.