PCS-2056
Linguagens e Compiladores

Formalização da Sintaxe

Renan Martins Zomignani Mendes
Tiago Schelp Lopes

1. Definição informal da linguagem

A linguagem que será especificada terá principalmente elementos de C e Ruby.

Primeiramente, os tipos nativos são aqueles encontrados em C - `int, float, char, void` - acrescidos de `boolean` para representar resultados de expressões booleanas. Novos tipos podem ser criados fazendo-se uso da palavra reservada `newtype`. Apesar de fortemente tipada, essa linguagem permite também a realização de *castings*, dado que os tipos sejam compatíveis.

Além de novos tipos, é possível definir novas estruturas (ou agregados heterogêneos) a partir da palavra-chave `struct`.

Elementos comuns a C e Ruby foram mantidos, como as formas com as quais são feitas a atribuição e as comparações entre expressões. No entanto, prevaleceu a legibilidade do código ao se adotar os operadores de negação, conjunção e disjunção em sua forma Ruby, usando respectivamente as palavras reservadas `not, and` e `or`.

Os identificadores são definidos por uma letra, seguida ou não, de mais letras e números em qualquer ordem.

As chaves foram eliminadas completamente da linguagem, sendo o escopo delimitado pelo início de comandos de iteração e condição, e funções, e pela palavra reservada `end`. Também não é utilizado o ponto-e-vírgula. Como em Ruby, os diferentes comandos devem ser separados por uma quebra de linha (aqui simbolizada por `\n`)

2. Exemplo de programa

```
int fatorial_recursivo(n)
  if(n <= 1)
    return 1
  else
    return n * fatorial_recursivo(n-1)
  end
end

int fatorial_iterativo(n)
  int fatorial
  fatorial = 1
  while(n > 0)
    fatorial = fatorial * n
    n = n - 1
  end
  return fatorial
end

int main()
     int fatorial_10_recursivo
     int fatorial_10_iterativo
     fatorial_10_recursivo = fatorial_recursivo(10)
     fatorial_10_iterativo = fatorial_iterativo(10)
end
```

3. Definição formal da linguagem

A seguir, a definição formal da linguagem nas notações BNF e Wirth. Uma legenda encontra-se ao final para explicitar as abreviações não evidentes.

a. BNF

<LETRA> ::=	A|B|C|D|E|F|G|H|I|J|K|L|M|N|O|P|Q|R|S| T|U|V|W|X|Y|Z|a|b|c|d|e|f|g|h|i|j|k|l| m|n|o|p|q|r|s|t|u|v|w|x|y|z
<DIGITO> ::=	0|1|2|3|4|5|6|7|8|9
<LETRADIGITO> ::=	<LETRA> | <DIGITO> | <LETRADIGITO> <LETRA> | <LETRADIGITO> <DIGITO>
<NUM> ::=	<NUM INT> | <NUM REAL>
<NUM INT> ::=	<DIGITO> | <DIGITO> <NUM INT>
<NUM REAL> ::=	<NUM INT> . <NUM INT>
<BOOLEAN> ::=	true | false
<EXPR> ::=	<EXPR> + <TERMO> | <EXPR> - <TERMO> | <TERMO>
<TERMO> ::=	<TERMO> * <FATOR> | <TERMO> / <FATOR> | <TERMO> % <FATOR> | <FATOR>
<FATOR> ::=	<VALOR> | <NUM> | (<EXPR>)
<VALOR> ::=	<VALOR>.<ID> | <ID>
<ID> ::=	<LETRA> | <LETRA> <LETRADIGITO>
<TIPO> ::=	int|float|char|boolean|void
<DECL SIMP VAR> ::=	<ID> <ID>
<DECL AGREG HOM> ::=	<ID> <ID>[<NUM INT>]
<DECL AGREG HET> ::=	struct <ID> \n <DECL VAR> \n end
<DECL VAR> ::=	<DECL SIMP VAR> | <DECL AGREG HOM>

<DECL VARS> ::=	<DECL VAR>\n <DECL VARS> | <DECL VAR>
<DECL TIPO> ::=	newtype <ID> <ID>
<DECL FUNCAO> ::=	<ID> <ID>(<DECL PARAMS>) \n <DECL VARS> \n <COMANDOS> end
<DECL PARAMS> ::=	<ID> <ID> | <ID> <ID>, <DECL PARAMS>
<DECL GERAL> ::=	<DECL SIMP VAR> | <DECL AGREG HOM> | <DECL AGREG HET> | <DECL TIPO> | <DECL FUNCAO>
<DECL GERAIS> ::=	<DECL GERAL> | <DECL GERAL> \n <DECL GERAIS>
<COMANDO> ::=	<COMANDO ATR> | <COMANDO COND> | <COMANDO ITER> | <COMANDO RET> | <COMANDO CAST> | <COMANDO ENTR> | <COMANDO SAIDA> | <CHAMADA FUNCAO>
<COMANDOS> ::=	<COMANDOS>\n <COMANDO>\n | <COMANDO>\n |
<COMAND COND> ::=	if(<CONDICAO>) \n <DECL COMANDOS> \n else \n <DECL COMANDOS> \n end \n | if(<CONDICAO>) \n <DECL COMANDOS> \n <ELSIFS> | if(<CONDICAO>) \n <DECL COMANDOS> \n end
<ELSIFS> ::=	elsif(<CONDICAO>) \n <DECL COMANDOS> \n <ELSIFS> | elsif(<CONDICAO>) \n <DECL COMANDOS> \n end \n | else \n <DECL COMANDOS> \n end
<COMANDO ATR> ::=	<ID> = <EXPR>
<COMANDO ITER> ::=	while(<CONDICAO>) \n <DECL COMANDOS> \n end
<CHAMADA FUNCAO> ::=	ID(PARAMS)
<PARAMS> ::=	<PARAM> | <PARAM>, <PARAMS>
<PARAM> ::=	<CONDICAO> | <EXPR>

<CONDICAO> ::=	not (<CONDICAO>) | <CONDICAO> | <CONDICAO> or <CONDICAO> | <CONDICAO> and <CONDICAO> | <ID> | <BOOLEAN> | <EXPR> > <EXPR> | <EXPR> >= <EXPR> | <EXPR> < <EXPR> | <EXPR> <= <EXPR> | <EXPR> == <EXPR> | <EXPR> != <EXPR>
<COMANDO ENTR> ::=	input <LISTA MEM> \n
<LISTA MEM> ::=	<ID> | <ID>, <LISTA MEM>
<COMANDO SAIDA> ::=	output <LISTA EXPR> \n
<LISTA EXPR> ::=	<EXPR> | <EXPR>, <LISTA EXPR> \n
<COMANDO RET> ::=	return <EXPR> \n
<COMANDO CAST> ::=	(<ID>) <EXPR>
<DECL COMANDO> ::=	<DECL VAR> | <COMANDO>
<DECL COMANDOS> ::=	<DECL COMANDO> | <DECL COMANDO> \n <DECL COMANDOS>
<PROGRAM> ::=	<DECL GERAIS> \n int main() \n <DECL OU COMANDOS> \n end

b. Notação de Wirth

LETRA =	“A”|“B”|“C”|“D”|“E”|“F”|“G”|“H”|“I”|“J” |“K”|“L”|“M”|“N”|“O”|“P”|“Q”|“R”|“S”| “T”|“U”|“V”|“W”|“X”|“Y”|“Z”|“a”|“b”|“c” |“d”|“e”|“f”|“g”|“h”|“i”|“j”|“k”|“l”| “m”|“n”|“o”|“p”|“q”|“r”|“s”|“t”|“u”|“v” |“w”|“x”|“y”|“z”.
DIGITO =	“0”|“1”|“2”|“3”|“4”| “5”|“6”|“7”|“8”|“9”.
NUM =	NUM_INT | NUM_REAL.
NUM_INT =	DIGITO{DIGITO}.
NUM_REAL =	NUM_INT “.” NUM_INT.
BOOLEAN =	“true”| “false”.
EXPR =	EXPR “+” TERMO|EXPR “-” TERMO| TERMO.
TERMO =	TERMO “*” FATOR | TERMO “/” FATOR | TERMO “%” FATOR | FATOR.
FATOR =	VALOR | NUM | “(” EXPR “)”.
VALOR =	ID{“.” ID}.
ID =	LETRA{(LETRA|DIGITO)}.
TIPO =	“int”|“float”|“char”|“boolean”|“void”.
DECL_SIMP_VAR =	ID ID.
DECL_AGREG_HOM =	ID ID[NUM_INT].
DECL_VAR =	DECL_SIMP_VAR | DECL_AGREG_HOM.
DECL_VARS =	{DECL_VAR “\n”}.
DECL_AGREG_HET =	“struct” ID “\n” DECL_VARS “end”.
DECL_TIPO =	“newtype” ID ID.
DECL_PARAMS =	ID ID {“,” ID ID}.

COMANDO =	COMANDO_ATR | COMANDO_COND | COMANDO_ITER | CHAMADA_FUNCAO | COMANDO_RETORNO | COMANDO_CAST | COMANDO_SAIDA | COMANDO_ENTR.
COMANDOS =	{COMANDO "\n"}.
CONDICAO =	"not" "("CONDICAO")" | CONDICAO | CONDICAO "or" CONDICAO | CONDICAO "and" CONDICAO | ID | BOOLEAN | EXPR ">" EXPR | EXPR ">=" EXPR | EXPR "<" EXPR | EXPR "<=" EXPR | EXPR "==" EXPR | EXPR "!=" EXPR.
COMANDO_COND =	"if" "(" CONDICAO ")" "\n" DECL_OU_COMANDOS {"elsif" "(" CONDICAO ")" "\n" DECL_OU_COMANDOS} ["else" "\n" DECL_OU_COMANDOS] "end".
COMANDO_ATR =	ID "=" EXPR.
COMANDO_ITER =	"while" "(" CONDICAO ")" "\n" DECL_OU_COMANDOS "\n" "end".
CHAMADA_FUNCAO =	ID(PARAM {"," PARAM}).
PARAM =	EXPR | CONDICAO.
COMANDO_ENTR =	"input" LISTA_MEM.
LISTA_MEM =	ID {"," ID}.
COMANDO_SAIDA =	"output" LISTA_EXPR.
LISTA_EXPR =	EXPR {"," EXPR}.
COMANDO_RETORNO =	"return" EXPR.
COMANDO_CAST =	"(" ID ")" EXPR.
DECL_OU_COMANDO =	DECL_VAR | COMANDO.
DECL_OU_COMANDOS =	{DECL_OU_COMANDO "\n"}.
DECL_FUNCAO =	ID ID(DECL_PARAMS) "\n" {DECL_OU_COMANDO "\n"} "end".

DECL_GERAL =	DECL_SIMP_VAR | DECL_AGREG_HOM | DECL_AGREG_HET | DECL_TIPO | DECL_FUNCAO.
PROGRAM =	{DECL_GERAL "\n"} "int" "main" "(" ")" "\n" {DECL_OU_COMANDO "\n"} "end".

Legenda:
DECL SIMP VAR: declaração de variável simples
DECL AGREG HOM: declaração de agregado homogêneo (arrays)
DECL AGREG HET: declaração de agregado heterogêneo (structs)