Kompilator pseudokodu do Pythona

Barbara Doncer, Bogusław Błachut

1. Przykład działania

Na wejściu podajemy pseudokod:

```
x <- 1;
if (x = 1){
        y<-2;
} else {
        y<-3;
}

function my_print(x){
        for (i <- 1…x){
                print(i);
        }
        return true;
}

my_print(5);

arr <- [1,2,3];
arr[1] <- 5;
z <- arr[2];
```

Jako wynik otrzymujemy plik result.py z kodem w Pythonie:

```
x=1
if x==1:
   y=2
else:
   y=3
def my_print(x):
   for i in range(1,x):
       print(i)
   return True
my_print(5)
arr=[1,2,3]
arr[1]=5
z=arr[2]
```

2. Spis tokenów

TOKEN	DESCRIPTION
assign	<-
skip	continue | break
if_token	if
else_token	else
while_token	while
for_token	for
return_token	return
function_token	function
and_token	and
or_token	or
not_token	not
boolean	true | false
id	[A-Z a-z_][A-Za-z0-9_]*
number	[0-9]+
curly_bracket_begin	{
curly_bracket_end	}
round_bracket_begin	(
round_bracket_end	)
square_bracket_begin	[
square_bracket_end	]
quotation_mark	“
comparison_operators	= | >= | <= | < | > | !=
math_operators	+ | - | * | / | % | ^
comma	,
between	…

whitespace	\s
new_line	\n
semicolon	;
error	fragment pseudokodu, który nie pasuje do żadnego innego tokena

3. Gramatyka

for_statement: [for_token] [round_bracket_begin] [id] [assign] [id | number] between [id | number] [round_bracket_end] [curly_bracket_begin] [statement | skip]+ [curly_bracket_end]

while_statement: [while_token] [round_bracket_begin] [expression] [round_bracket_end] [curly_bracket_begin] [statement | skip]+ [curly_bracket_end]

if_statement: [if_token] [round_bracket_begin] [expression] [round_bracket_end] [curly_bracket_begin] [statement | skip]+ [curly_bracket_end]
[else_token curly_bracket_begin [statement | skip]+ curly_bracket_end]?

return_statement: [return_token] [variable_type] [semicolon]

function_def: [function_token] **[**id] [round_bracket_begin] [[id]([comma][id])*]? [round_bracket_end] [curly_bracket_begin] [statement]+ [return_statement]? [curly_bracket_end]

array: [square_bracket_begin] [variable_type][[comma][variable_type]]* [square_bracket_end]

expression: ([not]? [variable_type | and_expression | or_expression | comparision_operators_expression | math_operators_expression])

statement: [for_statement | while_statement | if_statement | return_statement | declaration | function_call | function_definition]

and_expression: [expression] [and_token] [expression]

or_expression: [expression] [or_token] [expression]

comparision_operators_expression: [expression] [comparision_operator] [expression]

math_operators_expression: expression math_operator expression

declaration: [id] [assign] [variable_type] [semicolon]

function_call: **[**id] [round_bracket_begin] [[variable_type][[comma][variable_type]]*]? [round_bracket_end] [semicolon]

variable_type: [boolean | id | number | array | string | array_element]

string: [quotation_mark] [.]* [quotation_mark]

array_element: [id][square_bracket_begin] [variable_type][square_bracket_end]