



yadc

VLADIMÍRA KIMLOVÁ

DOMINIK ZAPPE

30. 10. 2023

Popis jazyka – základ

Silně typovaný

Datové typy:	int, float, bool, (string?), (void?)
--------------	--------------------------------------

Aritmetika:	+ - * / %
-------------	-----------

Přiřazení:	=
------------	---

Porovnání:	== < > <= >= !=
------------	-----------------

Logické operátory:	&& !
--------------------	---------

Řídící konstrukce:	while, for, if (else)
--------------------	-----------------------

(Klíčová slova možná budou ve finální verzi změněna na atypická)

Popis jazyka – typy a aritmetika

Implicitní přetypování: z operandů se zvolí „větší“ datový typ (float > int) a v jeho aritmetice se provede výpočet

```
float a; int b, c;
```

```
c = a / b // a / b bude float dělení, výsledek se převede na int
```

Logické hodnoty true a false: true == (int) 1 a false == (int) 0; jiná čísla se transformují na 1

```
true + 1 == 1 // True
```

```
true - true == false // 1 - 1 == 0 -> True
```

(Řetězce se nemohou míchat s jinými datovými typy (řetězec + číslo apod.))
(Řetězce nejspíš budou řešeny pomocí pole znaků)

Technologie + Cílová platforma



flex + bison (lex + yacc)



Instrukce rozšířené PL/0

Gramatika (1/7)

```
<program> ::=  
    <decl_var_stmt> <program>  
    | <decl_func_stmt> <program>  
    | /* empty */  
  
<decl_var_stmt> ::=  
    <TYPE> <ID> <SEMICOLON>  
    | <CONSTANT> <TYPE> <ID> <SEMICOLON>  
    | <TYPE> <assign_stmt>  
    | <CONSTANT> <TYPE> <assign_stmt>  
  
<assign_stmt> ::=  
    <ID> <ASSIGN_OP> <expr> <SEMICOLON>
```

Gramatika (2/7)

```
<decl_func_stmt> ::=  
    <TYPE> <ID> <L_BRACKET> <params> <R_BRACKET> <block>  
    | <TYPE> <ID> <L_BRACKET> <params> <R_BRACKET> <SEMICOLON>  
  
<params> ::=  
    <params_list>  
    | /* empty */  
  
<params_list> ::=  
    <TYPE> <ID> <COMMA> <params_list>  
    | <TYPE> <ID>  
  
<block> ::=  
    <BEGIN_BLOCK> <stmts> <END_BLOCK>
```

Gramatika (3/7)

```
<stmts> ::=  
    <stmt> <stmts>  
    | /* empty */  
  
<stmt> ::=  
    <decl_var_stmt>  
    | <assign_stmt>  
    | <if_stmt>  
    | <while_stmt>  
    | <for_stmt>  
    | <call_func_stmt>  
    | <return_stmt>
```

Gramatika (4/7)

```
<if_stmt> ::=  
    <IF> <L_BRACKET> <expr> <R_BRACKET> <block> <else_stmt>  
  
<else_stmt> ::=  
    <ELSE> <block>  
    | /* empty */  
  
<while_stmt> ::=  
    <WHILE> <L_BRACKET> <expr> <R_BRACKET> <block>  
  
<for_stmt> ::=  
    <FOR> <L_BRACKET> <expr> <SEMICOLON> <expr> <SEMICOLON> <expr> <R_BRACKET> <block>  
  
<call_func_stmt> ::=  
    <call_func_expr> <SEMICOLON>  
  
<return_stmt> ::=  
    <RETURN> <expr> <SEMICOLON>
```


Gramatika (5/7)

```
<expr> ::=
    <ID>
    | <LITERAL>
    | <L_BRACKET> <expr> <R_BRACKET>
    | <arithm_expr>
    | <logic_expr>
    | <compare_expr>
    | <cast_expr>
    | <call_func_expr>

<arithm_expr> ::=
    <expr> <SUM> <expr>
    | <expr> <SUB> <expr>
    | <expr> <MUL> <expr>
    | <expr> <DIV> <expr>
    | <expr> <MOD> <expr>
    | <U_MINUS> <expr> /* unary minus */
```

Gramatika (6/7)

```
<logic_expr> ::=  
    <expr> <AND> <expr>  
    | <expr> <OR> <expr>  
    | <NOT> <expr>  
  
<compare_expr> ::=  
    <expr> <EQ> <expr>  
    | <expr> <NEQ> <expr>  
    | <expr> <LESS> <expr>  
    | <expr> <LESSEQ> <expr>  
    | <expr> <GRT> <expr>  
    | <expr> <GRTEQ> <expr>  
  
<cast_expr> ::=  
    <L_BRACKET> <TYPE> <R_BRACKET> <expr>
```

Gramatika (7/7)

```
<call_func_expr> ::=  
    <ID> <L_BRACKET> <args> <R_BRACKET>
```

```
<args> ::=  
    <args_list>  
    | /* empty */
```

```
<args_list> ::=  
    <expr> <COMMA> <args_list>  
    | <expr>
```



Děkujeme za pozornost!