**Dominik Wiszyński**

**Dokumentacja TKOM**

# Wstęp

Mój język nazywa się Domino. Jest to język napisany w języku programowania C++. Jest on prostym językiem imperatywnym. Będzie on posiadał typizację statyczną.

# Formalna gramatyka języka

DOMINO = FUNCTION\* MAINFUNCTION

MAINFUNCTION = „function main” OROUNDBRACKET PARAMETERS CROUNDBRACKET „void” OPENBRACKET CONTENT CLOSEBRACKET

FUNCTION=”function” NAME OROUNDBRACKET PARAMETERS CROUNDBRACKET returntype OPENBRACKET CONTENT CLOSEBRACKET

NAME=ALPHA{ALPHANUMPLUS}

PARAMETERS = VARIABLE {COMA VARIABLE}

CONTENT = {(STATEMENT SEMICOLON|CONDITIONALEXP)}

CONDITIONALSTATEMENT=LOOPEXP|IFEXP

LOOPEXP = („WHILE” OROUNDBRACKET CONDITIONALEXP CROUNDBRACKET OPENBRACKET CONTENT CLOSEBRACKET|„FOR” OROUNDBRACKET CONDITIONALEXP CROUNDBRACKET OPENBRACKET CONTENT CLOSEBRACKET)

STATEMENT = VARIABLEEXP|FUNCCALL| RETEXP|STREAMEXP|VARDEF

IFEXP=„IF” OROUNDBRACKET CONDITIONALEXP CROUNDBRACKET OPENBRACKET CONTENT CLOSEBRACKET

CONDITIONALEXP =CONDITION{LOGICALOPERATOR CONDITION}

CONDITION=[EXCLAMATION]EXPRESSION {RELATIVEOPERATOR EXPRESSION}

VARDEF = TYPE (NAME| NAME ’[‘ INT ‘]’ | VARIABLEEXP)

VARIABLEEXP=NAME [ ‘[‘ INT ‘]’ ] ASSIGN EXPRESSION

FUNCCALL= NAME OROUNDBRACKET PARAMETERS CROUNDBRACKET

EXPRESSION = SIMPLEEXPRESSION {ADDOPERATOR SIMPLEEXPRESSION) }

SIMPLEEXPRESSION=(VALUE|FUNCCALL|NAME) {MULTIPLYOPERATOR (VALUE|FUNCCALL|NAME)}

RETEXP = „RETURN” (NAME|VALUE|FUNCCALL)

STREAMEXP = WRITEINSTAT|WRITEOUTSTAT

WRITEINSTAT = „writein” {RIGHTTRBRACKETS NAME}

WRITEOUTSTAT = „writeout” { LEFTTRBRACKETS (NAME|VALUE)}

VALUE = STRING|INT|FLOAT|CHAR|RATIONAL|BOOLEAN

STRING={CHAR}

FLOAT = SHORTINT DOT SHORTINT ’f’

RATIONAL = SHORTINT DOT SHORTINT r’

INT = {,9}[::nonzerodigit::]{[::digit::]}

SHORTINT = {,5}[::nonzerodigit::]{[::digit::]}

CHAR=[::asciisign::] //Nie wiem czy potrzeba to głębiej definiować

BOOLEAN=„TRUE”|”FALSE”

ADDOPERATOR=[ADD, SUBTRACT]

MULTIPLYOPERATOR = [DIVIDE, MULTIPLY, REST]

RELATIVEOPERATOR=[LESS, MORE, LESSOREQUAL, MOREOREQUAL, EQUALS, DIFFERENT]

LOGICALOPERATOR=[AND,OR]

TYPE = „STRING”| „INT”| „FLOAT”| „CHAR”| „RATIONAL” | „BOOLEAN”

RIGHTTRBRACKETS = „>>”

LEFTTRBRACKETS = „< <”

ADD=’+’

SUBTRACT=’-‘

DIVIDE=’/’

MULTIPLY=’\*’

REST=’%’

MOREOREQUAL = MORE ASSIGN

LESSOREQUAL = LESS ASSIGN

DIFFERENT = (EXCLAMATION ASSIGN)|(LESS MORE)

MORE = ‘>’

LESS = ‘<’

ASSIGN = ‘=’

EQUALS=’==’

OROUNDBRACKET = ‘(‘

CROUNDBRACKET=’)’

OPENBRACKET = ‘{‘

CLOSEBRACKET=’}’

ALPHA=[a-z, A-Z]

ALPHANUMPLUS=[ALPHANUM,’\_’]

ALPHANUM=[a-z,A-Z,0-9]

SEMICOLON=’;’

COMA = ‘,’

DOT=’.’

EXCLAMATION = ‘!’

AND=”&&”

OR=”||”

# Opis funkcjonalności

Język mój będzie obsługiwał podstawowe operacje matematyczne takie jak dodawanie, odejmowanie, dzielenie, mnożenie oraz reszta z dzielenia, które będą oznaczane w standardowy sposób. Będzie on posiadał również wyrażenia warunkowe ( konstrukcja if(…)) oraz wielokrotne powtarzanie operacji(pętle while(…)). Każda z funkcji będzie się zaczynała od słówka function …():{type}, który będzie jaśnie deklarował zwracany typ. Będą zachowane priorytety operatorów. Będą obsługiwane podstawowe typy takie jak znak (char), liczba całkowita(int), liczba zmiennoprzecinkowa(float), liczba wymierna(rational) oraz strumień znaków. Będzie można podawać znaki na wejście i wyjście programu.

# Implementacja, użyte narzędzia, sposób uruchomienia

Wszystko zaimplementowane będzie w języku C++. Analizator leksykalny będę pisał sam. Kod będzie przechowywany w plikach .txt. Aby zinterpretować nasz kod wywoływać będziemy plik wykonywalny ./interpreter z plikiem kodu jako parametr wejściowy np. ./interpreter kod.txt. Interpreter zinterpretuje nasz kod w języku domino w kodzie C++, po czym wykona go. Do testow uzyta zostala biblioteka boost.

# Przykłady

function main():void{

int c=5;

int d=4;

int e= c\*d;

while(e>0){

int a=1;

e=e-c;

writeout<<e<<" "<<a;

a=a+1;

writeout<<" "<<a<<" ";

}

}

function main():void{

int c=-4;

int d=5;

int e=7;

float z = 123.37f;

rational r = 1234.456r;

int f= e+c\*d;

if(c==3) {

writeout<<r<<","<<c<<","<<f;

}

else{

float dominik = 77.77f;

writeout<<c<<" "<<dominik<<" "<<r;

}

}