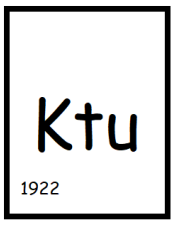

KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS

Programavimo kalbų teorija (P175B124)

Projekto galutinė ataskaita

Atliko:

IFF-1/9 Martynas Kuliešius

IFF-1/9 Ugnius Rinkevičius

IFF-1/9 Lukas Navickas

Priėmė:

lekt. Guogis Evaldas

lekt. Fyleris Tautvydas

Kaunas 2023

Turinys

Kuriama kalba ir komandos pavadinimas

Kuriame naują programavimo kalbą (interpretatorių), kurią nusprendėme pavadinti Array++.

Mūsų komandos pavadinimas yra “Binary skaitytojai”. Šią komandą sudaro trys nariai:

- Martynas Kuliešius – Ataskaitos ir skaidrių paruošimas, testavimas, gramatikos dalis.
- Ugnius Rinkevičius – Pagrindinis kalbos rašymas/realizavimas, testavimas.
- Lukas Navickas - Pagrindinis kalbos rašymas/realizavimas, testavimas.

Esminės kalbos savybės

Mes nusprendėme, kad norime papildyti tipizuotų masyvų funkcijas.

Programavimo kalbos savybės:

- Išplėstas array funkcionalumas
- Remove funkcija
- Insert funkcija
- Length funkcija
- Duomenų sumaišymas su Randomise funkcija
- Duomenų pripildymas pasirinktais elementais su Fill funkcija
- Pasikartojančių element pašalinimas su Unique funkcija
- Masyvo intarpo gavimas su Slice funkcija
- Neigiamo masyvo indekso atskaičiavimas nuo masyvo galo.

Kalbos kūrimo priemonės

Sukurti kalbai pasinaudojome:

- JetBrains Rider
- ANTLR
- VS Code
- ChatGPT
- Twitch
- Youtube

Kalbos apribojimai

Nėra pilnai realizuoti metodai, kad veiktų su String tipo kintamaisiais.

Kodas įrašomas į failą, kurį interpretatorius nuskaito ir atlieka faile aprašytas instrukcijas.

Galutinė kalbos gramatika

```
grammar ArrayPP;

program: line* EOF;

line: statement;

statement
    : declaration ';'
    | variableDeclaration ';'
    | assignment ';'
    | functionCall ';'
    | ifBlock
    | forCycle
    | whileCycle
    | function
    | printToScreen ';'
    | return ';'
    | arrayAssignment ';'
    | arrayRemoval ';'
    | arrayInsert ';'
    | arrayRandomiser ';'
    | arrayFill ';'
    | arrayFilter ';'
    | arrayUnique ';'
    | arraySlice ';'
    ;

ifBlock: 'if' '(' expression ')' block ('else' elseIfBlock)?;
elseIfBlock: block | ifBlock;
declaration: type IDENTIFIER;
variableDeclaration: declaration '=' (expression | arrayLiteral);
assignment:  IDENTIFIER '=' expression;

arrayAssignment: IDENTIFIER '[' expression ']' '=' expression;
arrayRemoval: IDENTIFIER '[' expression ']' '.remove';
arrayInsert: (IDENTIFIER | functionCall) '.insert(' expression ',' expression ')';
arrayLength: (IDENTIFIER | functionCall) '.length';
arrayRandomiser: (IDENTIFIER | functionCall) '.randomise';
```

```
arrayFill: (IDENTIFIER | functionCall ) '.fill(' expression ',' expression ')';
arrayFilter: (IDENTIFIER | functionCall ) '.filter(' expression ',' expression ')';
arrayUnique: (IDENTIFIER | functionCall ) '.unique';
arraySlice: (IDENTIFIER | functionCall ) '.slice(' expression ',' expression ')';

functionCall: IDENTIFIER '(' (expression (',' expression)*)? ')';

forCycle: 'for' '(' (variableDeclaration | assignment) ';' expression ';' assignment
')' block;
whileCycle: 'while' '(' expression ')' block;

function: 'function' IDENTIFIER '(' (declaration (',' declaration)*)? ')' (':' type)?
block;

return:  'return' expression?;

printToScreen: 'print(' expression ')';
readFromFile: 'readFile(' expression ')';

substring: 'substring(' expression ',' expression ')';
split:  (IDENTIFIER | functionCall)'.split(' (expression) ')';


expression
    : IDENTIFIER                            #identifierExpression
    | functionCall                          #functionCallExpression
    | '(' expression ')'                    #parenExpression
    | expression multOp expression          #multExpression
    | expression addOp expression           #addExpression
    | expression compareOp expression       #compareExpression
    | expression boolOp expression          #boolExpression
    | arrayLiteral                          #arrayExpression
    | indexAccess                           #indexAccessExpression
    | readFromFile                          #readFromFileExprission
    | substring                             #substringExpression
    | split                                 #splitExpression
    | arrayLength                           #lengthExpression
    | arrayUnique                           #uniqueExpression
    | constant                              #constantExpression
    ;

multOp: '*' | '/';
addOp: '+' | '-';
compareOp: '==' | '!=' | '<' | '>' | '<=' | '>=';
boolOp: '&&' | '||';
```

```
arrayLiteral: ('[' expressionList? ']') | '[]';
expressionList: expression (',' (expression))*;


indexAccess: IDENTIFIER '[' expression ']';

primitiveType:
    'int'
    |'char'
    |'string'
    |'bool'
    |'float';

type: primitiveType | (primitiveType '[]');

constant:
    INTEGER     #integerConstant
    | FLOAT     #floatConstant
    | STRING    #stringConstant
    | BOOL      #boolConstant
    | CHAR      #charConstant
    | NULL      #nullConstant
    ;

INTEGER: ('-')? [0-9]+;
FLOAT: ('-')? [0-9]+ '.' [0-9]+;
STRING: '"' (~["\\])* '"';
CHAR: '\'' ~'\'' '\'';
BOOL: 'true' | 'false';
NULL: 'null';

block: '{' line* '}';
COMMENT : ( '//' ~[\r\n]*  ) -> skip ;
WS: [ \t\r\n]+ -> skip;
IDENTIFIER: [a-zA-Z_][a-zA-Z0-9_]*;
```

Kaip sukompiliuoti ir paleisti

Per JetBrains Rider atsidaromas projektas. Projektas subuildinamas. Prieš buildinant yra atsisiunčiamas ANTLRV4 pluginas iš Rider Plugin Marketplace. Į failą pavadinimu „file“ yra įrašomas norimas kodas. Rider programoje paspaudžiamas mygtukas „run“ ir norimas kodas paleidžiamas

Pavyzdinis kodas ir rezultatai

Pavyzdinis Programos kodas:

```
int[] arr = [1, 2, 3, 4, 5, 6, 7];

print("");
print("-----------------------------------------");

print("");
print("Pradinis masyvas:");
print(arr);

print("");
print("Išfiltruojami visi skaičiai nuo 4 iki 6:");
arr.filter(4, 6);
print(arr);

print("");
print("Įterpiamas skaičius 10 į -2 indeksą (antrą nuo galo):");
arr.insert(-2, 10);
print(arr);

print("");
print("Ištrinamas -3 indekse esantis skaičius (trečias nuo galo):");
arr[-3].remove;
print(arr);

print("");
print("Sumaišomas masyvas:");
arr.randomise;
print(arr);

print("");
print("Užpildomas masyvas 10-čia 5-etų:");
arr.fill(10, 5); //index value
print(arr);

print("");
print("Įterpiamas skaičius 1 į -1 indeksą (paskutinį):");
arr.insert(-1, 1);
print(arr);

print("");
print("Masyvas atkerpamas nuo 2 iki 9 indekso:");
arr.slice(2, 9); // iki ntojo indexo nuo xtojo indekso
print(arr);

print("");
print("Atrenkami unikalūs masyvo skaičiai:");
arr.unique;
```

```
print(arr);

print("");
print("Nukopijuojamas masyvas:");
int []arr2;
arr2 = arr;
print(arr2);

print("");
print("-------------------------------------------");

print("");
print("Programos veikimas (for, if):");

int skaicius = 1;
for(int i = 0; i < 10; i=i+1)
{
    if ( skaicius != 6)
    {
        skaicius = skaicius + 1;
    }
}
print(skaicius);

print("");
print("-------------------------------------------");

print("");
print("Programos veikimas (while):");

int sk = 1;
while(sk < 8)
{
    sk = sk + 1;
}
print(sk);

print("");
print("-------------------------------------------");

print("");
print("Programos veikimas (rekursija):");

function recursive(int rec)
{
    if(rec != 10)
    {
        rec = rec + 1;
        print(rec);
        recursive(rec);
    }
    else
    {
        return;
    }
}
```

```
int rekursija = 0;
recursive(rekursija);

print("");
print("-----------------------------------------");
```

Testavimo rezultatas:

Running file: "file"

Pradinis masyvas:

[1, 2, 3, 4, 5, 6, 7]

Išfiltruojami visi skaičiai nuo 4 iki 6:

[1, 2, 3, 7]

Įterpiamas skaičius 10 į -2 indeksą (antrą nuo galo):

[1, 2, 3, 10, 7]

Ištrinamas -3 indekse esantis skaičius (trečias nuo galo):

[1, 2, 10, 7]

Sumaišomas masyvas:

[10, 7, 2, 1]

Užpildomas masyvas 10-čia 5-etų:

[5, 5, 5, 5, 5, 5, 5, 5, 5, 5]

Įterpiamas skaičius 1 į -1 indeksą (paskutinį):

[5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 1]

Masyvas atkerpamas nuo 2 iki 9 indekso:

[5, 5, 5, 5, 5, 5, 5, 5, 1]

Atrenkami unikalūs masyvo skaičiai:

[5, 1]

Nukopijuojamas masyvas:

[5, 1]

Programos veikimas (for, if):

6

Programos veikimas (while):

8

Programos veikimas (rekursija):

1

2

3

4

5

6

7

8

9

10

Process finished with exit code 0.

Naudoti pavyzdžiai, šaltiniai

https://tomassetti.me/antlr-mega-tutorial/

https://www.antlr.org

https://medium.com/@fwouts/a-quick-intro-to-antlr4-5f4f35719823

https://www.baeldung.com/java-antlr

https://putridparrot.com/blog/antlr-in-c/

https://www.youtube.com/watch?v=bfiAvWZWnDA

https://www.youtube.com/watch?v=lc9JlXyBG4E

https://stackoverflow.com/questions/19327831/antlr4-c-sharp-application-tutorial-example

https://medium.com/@konstantin.poklonskiy/how-to-create-your-own-language-with-antlr-in-net-a57beba84ecc

https://learn.microsoft.com/en-us/events/dotnetconf-2020/create-a-text-parser-in-c-with-antlr

https://stackoverflow.com/questions/2842809/lexers-vs-parsers

https://www.techtarget.com/searchapparchitecture/definition/parser

https://www.techopedia.com/definition/7793/interpreter