

**Kauno technologijos universitetas**

Informatikos fakultetas

**Programavimo kalbų teorija (P175B124)**

Projekto ataskaita

**Komandos nariai**

Emilis Kasiulynas IFF-1/5

Laimis Černiauskas IFF-1/5

Mildaras Karvelis IFF-1/4

**Dėstytojai**

Lekt. Guogis Evaldas

Lekt. Fyleris Tautvydas

Doc. Sajavičius Svajūnas

**Kaunas, 2023**

**Turinys**

[1. I dalis 3](#_Toc135819302)

[1.1. Kalbos idėja ir pavadinimas, komandos pavadinimas ir jos nariai 3](#_Toc135819303)

[1.2. Kalbos savybės, palaikomos struktūros bei duomenų tipai (t.y. ataskaitoje pridėtas jūsų kalbos kodas) 3](#_Toc135819304)

[1.3. Baziniai ir palaikomų kalbos konstrukcijų pavyzdžiai (pvz.: sąlygos sakiniai, ciklai, kintamieji, kt.) (t.y. kodas kuriamoje kalboje) 4](#_Toc135819305)

[1.4. Unikali savybė ir jos naudojimo kodo pavyzdys (jūsų programavimo kalboje parašytas pavyzdys) 4](#_Toc135819306)

[1.5. Pasirinkti darbo įrankiai (ir pasirinkimo priežastys) bei darbui naudojama programavimo kalba 4](#_Toc135819307)

[1.6. Galutinė kalbo gramatika 5](#_Toc135819308)

[1.7. Kopiliavimas ir paleidimas 9](#_Toc135819309)

[1.8. Pavyzdinis kodas ir gauti rezultatai 9](#_Toc135819310)

[2. Šaltiniai 13](#_Toc135819311)

# I dalis

## Kalbos idėja ir pavadinimas, komandos pavadinimas ir jos nariai

|  |
| --- |
| Kalbos pavadinimas |
| Frukt |

|  |
| --- |
| Kalbos idėja |
| Lengvai skaitoma ir patogiai rašoma programvimo kalba |

|  |
| --- |
| Komandos pavadinimas |
| Rimti Fruktai |

|  |
| --- |
| Komandos nariai |
| Emilis Kasiulynas |
| Laimis Černiauskas |
| Mildaras Karvelis |

## Kalbos savybės, palaikomos struktūros bei duomenų tipai (t.y. ataskaitoje pridėtas jūsų kalbos kodas)

|  |
| --- |
| Kalbos savybės |
| Kodas rašomas mažiau mašinine kalba. |
| Lengviau skaitomas kodas. |
| Lengviau atsiminti komandas ir kalbos konstrukcijas. |

|  |
| --- |
| Palaikomos struktūros |
| Array[] |

|  |
| --- |
| Duomenų tipai – kalba pati nustato koks tai duomenų tipas |
| Character  Text  Number  Expression |

## Baziniai ir palaikomų kalbos konstrukcijų pavyzdžiai (pvz.: sąlygos sakiniai, ciklai, kintamieji, kt.) (t.y. kodas kuriamoje kalboje)

|  |  |
| --- | --- |
| Konstrukcijos | |
| **Kontrukcija** | **Pavyzdys** |
| While ciklas | While A != 2 {  Print(A).  } |
| Sąlyga | If 10 >= 5 {  10 \* 5.  }  else 10 < 5 {  10 / 2.  } |
| “string” | Text = “vaisiai yra sveika vartoti žmogaus organizmui”. |
| “Integer" + “double” + “float”+  “decimal" | Number = 10  Number = 10.4 |
| “Char” | Symbol = “a” |
| “Boolean" | Expression = true/false. |
| Array | Array = [1, 2.3, 3, true, “Cia yra tekstas”, “#”]. |
| Write/WriteLine | Print(“Sveikas, apelsine!”).  PrintTo(failas, argumentas1, ...). --Pirmas funkcijos elementas  --privalo būti nuoroda į failą |

## Unikali savybė ir jos naudojimo kodo pavyzdys (jūsų programavimo kalboje parašytas pavyzdys)

|  |
| --- |
| Unikali savybė |
| Galimas paprastas dvigubas palyginimas su vienu kintamuoju.  Pvz. vietoj If(-3 < n && n < 5) galima būtų užrašyti If(-3 < n < 5) |

## Pasirinkti darbo įrankiai (ir pasirinkimo priežastys) bei darbui naudojama programavimo kalba

|  |
| --- |
| Darbo įrankiai |
| Visual Studio Code |
| * Plati priedų biblioteka. * Didžiausia turima naudojimo patirtis. |
| ANTLR4 |
| * Lengvai sugeneruojami lexer ir parser scriptai. * Palengvina darbą kuriant gramatiką. |

.

|  |
| --- |
| Programavimo kalba |
| C# |

## Galutinė kalbo gramatika

|  |
| --- |
| grammar Frukt;  /\*  --------------- Parser rules ----------------        Should be written in lower case  \*/  program                 : line\* EOF;  line                    : functionStatement | statement | ifBlock | whileBlock;  statement               : (assignment | functionCall | arrayDeclaration) '.';  block                   : '{' line\* '}';  //---Variables        assignment              : IDENTIFIER '=' expression;        constant                : NUMBER | FLOAT | BOOL | TEXT;  //---If        ifBlock                 : 'if' expression block (elseifBlock | elseBlock)?;        elseifBlock             : 'else' expression block (elseifBlock | elseBlock)?;        elseBlock               : 'else' block;  //---While        whileBlock              : WHILE expression block;  //---Function        functionStatement       : IDENTIFIER '(' (expression (',' expression)\*)? ')' block;        functionCall            : IDENTIFIER '(' (expression (',' expression)\*)? ')';  //---Array        arrayAccess             : IDENTIFIER '[' expression ']';        arrayDeclaration        : IDENTIFIER '=' '[' expressionList ']';        expressionList          : expression (',' expression)\*;  expression      : constant                                              #constantExpression      | IDENTIFIER                                            #identifierExpression      | functionCall                                          #functionCallExpression      | arrayAccess                                           #arrayAccessExpression      | '(' expression ')'                                    #parenthesesExpression      | expression multdivOp expression                       #multdivExpression      | expression addsubOp expression                        #addsubExpression      | expression compareOp expression compareOp expression  #multipleComparisonExpression      | expression compareOp expression                       #comparisonExpression      ;  multdivOp               : '\*' | '/';  addsubOp                : '+' | '-';  compareOp               : '==' | '!=' | '<' | '>' | '<=' | '>=';  /\*  ---------------- Lexer rules ----------------        Should be written in upper case  \*/  fragment DIGIT          : [0-9];  fragment T              : ('t' | 'T');  fragment R              : ('r' | 'R');  fragment U              : ('u' | 'U');  fragment E              : ('e' | 'E');  fragment F              : ('f' | 'F');  fragment A              : ('a' | 'A');  fragment L              : ('l' | 'L');  fragment S              : ('s' | 'S');  fragment W              : ('w' | 'W');  fragment H              : ('h' | 'H');  fragment I              : ('i' | 'I');  WHILE                   : W H I L E;  NUMBER                  : DIGIT+;  FLOAT                   : DIGIT+ ',' DIGIT+;  TEXT                    : ('"' ~'"'\* '"');  BOOL                    : (T R U E) | (F A L S E);  WHITESPACE              : ( ' ' | '\t' ) -> skip;  IDENTIFIER              : [a-zA-Z\_][a-zA-Z0-9\_]\*; |

## Kopiliavimas ir paleidimas

A screenshot of a computer

Description automatically generated with medium confidenceKompiliuoti galima per VS Code programą. Parašius \*.frtk scriptą galima pradėti kompiliuoti, tai padarius ./bin/Debug/net7.0 kataloge atsiras Frukt.exe failas, kurį paleisime per cmd (comand prompt) VS Code aplinkoje (kitaip reikėtų aprašyti abosliutų kelią iki failų), kad gauti tinkamus rezultatus.

CMD komandos:

1. (cd Frukt\bin\Debug\net7.0) suranda kelia iki reikiamų failų
2. (start cmd /k ".\Frukt.exe .\sample.frkt") paleidžia programą atskirame command lange su sample.frkt scriptu

## Pavyzdinis kodas ir gauti rezultatai

„sample.frkt“ failas:

|  |
| --- |
| PrintTo("output.txt", "Printinu", "i faila").  Fibonacci(limit, number1, number2)  {  if number1 + number2 > limit  {  Print("done").  PrintTo("fibonacci.txt", "done").  }  else  {  number12 = number1 + number2.  Print(number12).  PrintTo("fibonacci.txt", number12).  number1 = number2.  number2 = number12.  Fibonacci(limit, number1, number2).  }  }  Fibonacci(10, 1, 1).  Print("Sveikas, apelsine!").  b = "Bananas".  numbers = [1, "Obuolys", b, 4].  i = 0.  Print("Array testas!").  while i <= 3  {  Print(numbers[i]).  i = i + 1.  }  if 10 <= 30 <= 20  {  Print("if").  }  else 3 > 1  {  Print("else if").  }  else  {  Print("else").  }  i = 0.  Print("While testas!").  while i < 10  {  Print(i).  i = i + 1.  }  boolkint = true.  doublekint = 1,4233.  integerkint = 1.  stringkint = "Sveikas, apelsine!".  arraykint = [1, 2, "Bananas", integerkint]. |

Rezultatai komandų lange ir „output.txt“, „fibonacci.txt“ failuose:

A screenshot of a computer

Description automatically generated

„output.txt“

A screenshot of a computer

Description automatically generated

„fibonacci.txt“

A screenshot of a computer

Description automatically generated

# Šaltiniai

<https://www.youtube.com/watch?v=bfiAvWZWnDA> [žiūrėta 2023-05-23]

<https://github.com/P175B124> [žiūrėta 2023-05-23]