Государственное учреждение образования

“БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ”

Кафедра: Интеллектуальных информационных технологий

Дисциплина: Языковые процессоры интеллектуальных систем

Отчет по лабораторным работам №1-5

**Вариант 3**

Выполнил:

студент гр.121702

Витковская С. И.

Проверил:

Соколович М. Г.

Минск 2024

[**Свойства языка 3**](#_x9ethtz5fm14)

[**Синтаксис объявления переменных и подпрограмм 3**](#_9g5b7byaixs3)

[**Синтаксис операций над данными 3**](#_97zt5tqvq4rb)

[**Синтаксис управляющих конструкций 4**](#_s6deomow3fx)

[**Оформленный файл грамматики (.g) 4**](#_fl7qbgb1yed7)

[**Встроенные подпрограммы 8**](#_213kp99pfehd)

[**Перечень генерируемых ошибок 9**](#_kxk8x64ns2yq)

[**Примеры работы компилятора 9**](#_p9905mfv90tn)

## **Свойства языка**

Язык для работы с графовыми структурами

Встроенные типы: node, arc, graph

Операции: переопределить +, -, \*, \ и т.д. для встроенных типов

Встроенные функции поиска в ширину, поиска в глубину, поиска кратчайшего пути

Свойства:

1. Объявление переменных - явное
2. Преобразование типов - неявное
3. Оператор присваивания - одноцелевой
4. Структуры, ограничивающие область видимости - подпрограммы
5. Маркер блочного оператора - явные, например, { } или begin end
6. Условные операторы - двухвариантный оператор и многовариантный switch-case
7. Перегрузка подпрограмм - отсутствует
8. Передача параметров в подпрограмму - по ссылке
9. Допустимое место объявления подпрограмм - в любом месте программы, также и внутри другой подпрограммы.

## **Синтаксис объявления переменных и подпрограмм**

Для переменных типа:

1. node – node ID;
2. graph – graph ID,

graph ID = [nodes: (), arcs: ()],

graph = выражение/операция;

1. arc – arc ID,

arc = <node, node>;

1. int – int ID,

int ID = значение.

Функция:

function ID (параметры) {

тело функции} ;

## **Синтаксис операций над данными**

1. (variable) = (value | function\_call | expression | variable) – определение и переопределение значения
2. (variable | function\_call | expression) + (variable | function\_call | expression) – объединение графов (объединение множеств вершин и сложение дуг)
3. (variable | function\_call | expression) - (variable | function\_call | expression) – разность графов (в результирующий граф входят элементы, принадлежащие первой структуре и не принадлежащие второй)
4. (variable | function\_call | expression) & (variable | function\_call | expression) – пересечение графов (в результирующий граф входят элементы, принадлежащие обеим структурам)
5. (variable | function\_call | expression) / (variable | function\_call | expression) – симметрическая разность (в результирующий граф входят элементы, не принадлежащие ни первой ни второй структуре)
6. (variable | function\_call | expression) \* (variable | function\_call | expression) – декартово произведение множеств вершин
7. .node – обращение к множеству вершин графа
8. логические операции and, or
9. операции строго неравенства <, >, ==, !=.
10. операции нестрогого неравенства <=, >=.

## **Синтаксис управляющих конструкций**

1. if (условие) {}

else {};

1. switch (choice){

case X: блок действий;

break;}

1. while (условие) {тело цикла}
2. for (начальное значение счетчика; условие; действие итерации){блок цикла}

## **Оформленный файл грамматики (.g)**

Лексика

| NODE : 'node';  GRAPH : 'graph';  ARC: 'arc';  N\_ELEM : 'nodes';  A\_ELEM : 'arcs';  TYPE\_INT: 'int';  FOR : 'for' ;  WHILE : 'while' ;  IF : 'if' ;  ELSE : 'else' ;  SWITCH : 'switch' ;  CASE : 'case' ;  BREAK : 'break' ;  RETURN : 'return' ;  CONTINUE : 'continue' ;  FUNCTION : 'function' ;  DEPTH\_FIRST\_SEARCH: 'depth\_first\_search';  BREADTH\_FIRST\_SEARCH: 'breadth\_first\_search';  SHORTEST\_PATH: 'shortest\_path';  INT : [0-9]+ ;  ID: [a-zA-Z\_][a-zA-Z\_0-9]\* ;  WS: [ \t\n\r\f]+ -> skip ; | UNION : '+' ;  DIFF : '-' ;  SYMDIFF : '/' ;  INTERSEC : '&' ;  CART : '\*' ;  AND : 'and' ;  OR : 'or' ;  NOT : 'not' ;  EQ : '=' ;  COMMA : ',' ;  SEMI : ';' ;  COLON : ':' ;  EXCLAM : '!' ;  LSQUARE : '[' ;  RSQUARE : ']' ;  LANGLE : '<' ;  RANGLE : '>' ;  LPAREN : '(' ;  RPAREN : ')' ;  LCURLY : '{' ;  RCURLY : '}' ; |
| --- | --- |

Синтаксис:

types : NODE #вершина

| ARC #дуга

| GRAPH #граф

| TYPE\_INT; #целочисленный

operation : UNION # +

| DIFF # -

| SYMDIFF # /

| INTERSEC # &

| CART; # \*

anoun : types ID #объявление переменных

| TYPE\_INT ID '=' INT

| GRAPH ID '=' expr

| ARC ID '=' '<' node COMMA node '>'

| GRAPH ID '=' '[' N\_ELEM ':' '(' node (',' node)\* ')' ',' A\_ELEM ':' '(' arc (',' arc)\* ')' ']';

node : ID;

arc : ID | ARC ID '=' '<' node COMMA node '>' ;

program : (stat | anoun)\* EOF;

stat: ID '=' expr ';'

| expr ';'

| anoun ';'

| cicle

| condition

| def;

def : #объявление функции

'function' ID '(' (types ID (',' types ID)\*) ')' '{' stat\* '}' ;

expr: func

| expr operation expr

| 'not' expr

| expr 'and' expr

| expr 'or' expr

| expr ( ('<' | '>')('=')? | ('!' | '=')'=') expr

| ID

| INT

| BREAK

| CONTINUE

| RETURN expr;

cicle : WHILE '(' expr ')' '{' stat\* '}' #циклы

| FOR '(' (ID | TYPE\_INT ID) '=' expr ';' expr ( ('<' | '>')('=')? | ('!' | '=')'=') expr ';' ID '=' expr ')' '{' stat\* '}';

condition : #условные конструкции

IF '(' expr ')' '{' stat\*'}' ('else' '{' stat\*'}')?

| SWITCH '(' ID ')' '{' ( CASE INT ':' stat\*)\* '}' ;

or\_func : #встроенные функции

DEPTH\_FIRST\_SEARCH

| BREADTH\_FIRST\_SEARCH

| SHORTEST\_PATH ;

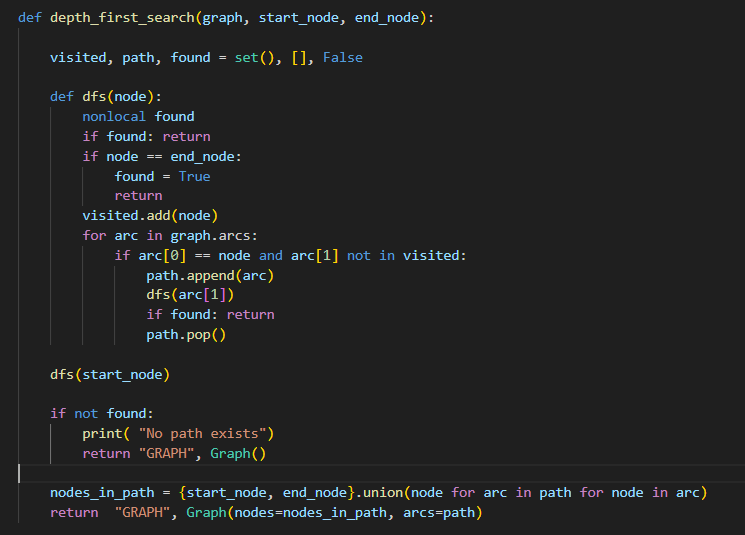
func #вызов функции

: ID '(' expr (',' expr)\* ')'

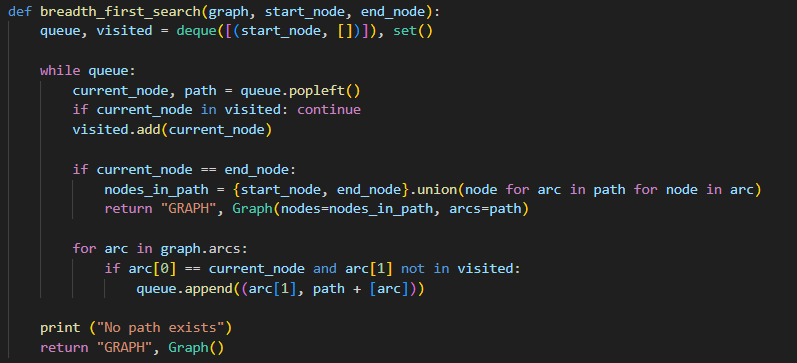
| or\_func '(' expr ',' expr ',' expr ')' ;

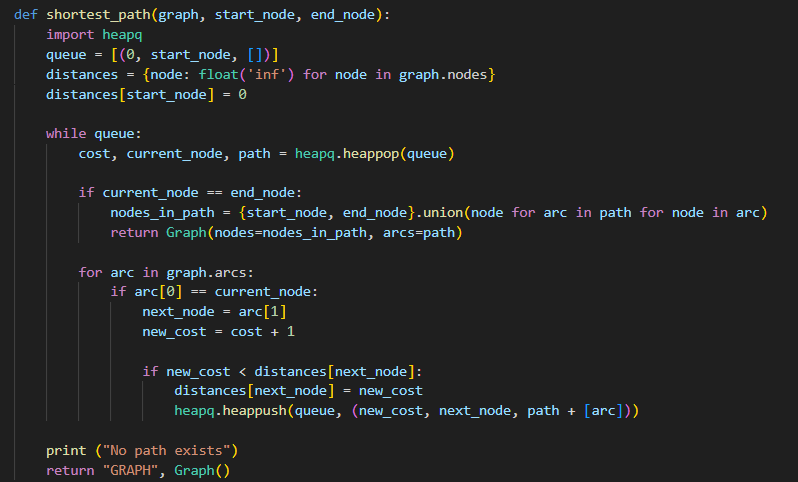
## **Встроенные подпрограммы**

DEPTH\_FIRST\_SEARCH – Поиск в глубину



BREADTH\_FIRST\_SEARCH – Поиск в ширину



SHORTEST\_PATH – Кратчайший путь

## **Перечень генерируемых ошибок**

1. SemanticError – класс исключений, создающий семантические ошибки с указанием строки и столбца, где ошибка произошла. Semantic error at line {line}, column {column}: {message}

Возможные сообщения:

- "Variable '{var\_name}' already declared" — попытка повторного объявления переменной.

- "Graph '{graph\_name}' already declared" — попытка повторного объявления графа.

- "Undefined variable: {var\_name}" — использование неопределенной переменной.

- "Undefined nodes in arc: {node1}, {node2}" — использование узлов, которые не были объявлены для дуги.

- "Undefined function: {func\_name}" — вызов несуществующей функции.

- "Function '{func\_name}' already declared" — попытка повторного объявления функции.

- "Function '{func\_name}' parameter names and types mismatch" — несоответствие количества имен параметров и их типов при объявлении функции.

- "Function '{func\_name}' expects {expected\_args} arguments, got {actual\_args}" — несоответствие количества ожидаемых и переданных аргументов при вызове функции.

- "Argument type mismatch for parameter '{param\_name}', expected {param\_type}, got {arg\_type}" — несоответствие типов аргументов при вызове функции.

- "Return statement must have an expression" — оператор `RETURN` не содержит выражения.

- "Operation requires both operands to be graphs" — операции над графами применены к неподходящим типам данных.

- "Type mismatch: variable '{var\_name}' is of type {expected\_type}, but the expression evaluates to type {value\_type}" — попытка присвоить переменной значение несовместимого типа.

- "Node '{node\_name}' is not declared" — использование узла, который не был объявлен.

- "Arc '{arc\_name}' is not declared" — использование дуги, которая не была объявлена.

- "Undefined nodes for this graph in arc: {node1}, {node2}" — дуга ссылается на узлы, которые не принадлежат графу.

- "Error processing declaration: {e}" — общая ошибка обработки объявления.

- "Error processing statement: {e}" — общая ошибка обработки выражения или инструкции.

- "Not int" — некорректное преобразование значения в целое число.

1. Ошибки стандартных типов TypeError:

- "Can only add two Graph objects" — при попытке сложить два объекта, которые не являются графами.

- "Can only intersect two Graph objects" — при попытке пересечь два объекта, которые не являются графами.

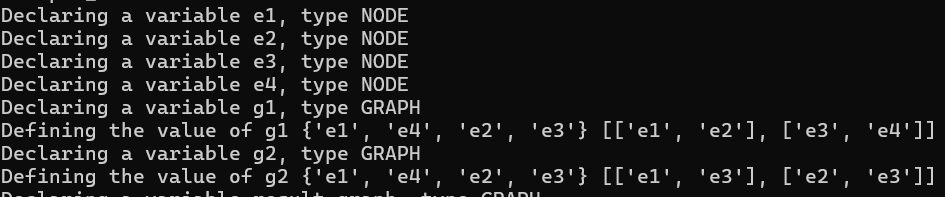
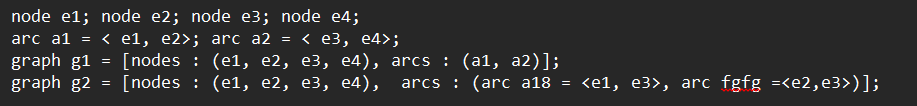
- "Can only subtract two Graph objects" — при попытке вычесть два объекта, которые не являются графами.

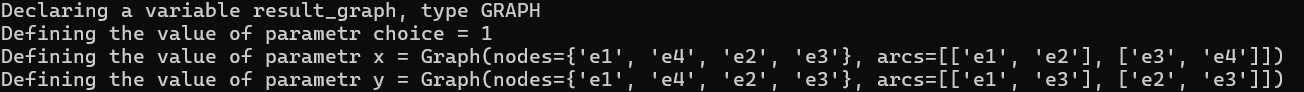
- "Can only perform symmetric difference on Graph objects" — при попытке выполнить симметрическую разность на объектах, которые не являются графами.

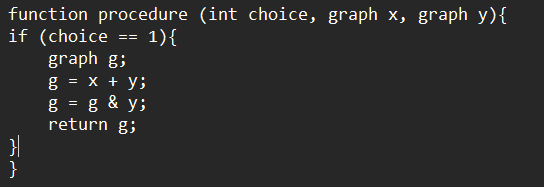
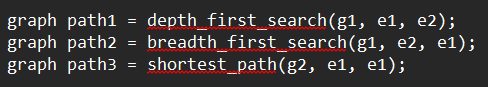
- "Can only perform product on two Graph objects" — при попытке выполнить произведение двух объектов, которые не являются графами.

3. Ошибки в алгоритмах поиска пути "No path exists" — в функциях `depth\_first\_search`, `breadth\_first\_search`, и `shortest\_path`, если путь между указанными узлами не найден.

**Примеры работы компилятора**

Объявление переменных Вызов функции

Обработка функции

Обработка встроенных функций

## 

**Вывод**

В ходе выполнения лабораторных работ был реализован компилятор, обрабатывающий язык для работы с графовыми структурами: были определены встроенные операции и функции, реализована логика их обработки; составлена грамматика языка (лексика + синтаксис), на ее основе построены синтаксический и семантический анализаторы, сформирован результирующий .exe-файл – обработчик.