Министерство образования Республики Беларусь
Учреждения образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

КАФЕДРА ИНФОРМАТИКИ

Отчет по лабораторной работе №4
По теме «Семантический анализатор»

Выполнил:
студент гр. 053501
Шебеко Ю.А.

Проверил:
Гриценко Н. Ю.

Минск 2023

СОДЕРЖАНИЕ

1 ЦЕЛЬ РАБОТЫ

Разработать свой собственный семантический анализатор выбранного подмножества языка программирования. Показать правильность работы семантического анализатора путём допущения ошибок в коде.

2 ТЕОРИЯ

В процессе семантического анализа проверяется наличие семантических ошибок в исходной программе и накапливается информация о типах для следующей стадии – генерации кода.

При семантическом анализе используются иерархические структуры, полученные во время синтаксического анализа для идентификации операторов и операндов выражений и инструкций. Важным аспектом семантического анализа является проверка типов, когда компилятор проверяет, что каждый оператор имеет операнды допустимого спецификациями языка типа. Например, определение многих языков программирования требует, чтобы при использовании действительного числа в качестве индекса массива генерировалось сообщение об ошибке. В то же время спецификация языка может позволить определенное насильственное преобразование типов, например, когда бинарный арифметический оператор применяется к операндам целого и действительного типов. В этом случае компилятору может потребоваться преобразование целого числа в действительное.

В большинстве языков программирования имеет место неявное изменение типов (иногда называемое приведением типов(coercion)). Реже встречаются языки, подобные Ada, в которых большинство изменений типов должно быть явным. В языках со статическими типами, например C, все типы известны во время компиляции, и это относится к типам выражений, идентификаторам и литералам. При этом неважно, насколько сложным является выражение: его тип может определяться во время компиляции за определенное количество шагов, исходя из типов его составляющих. Фактически, это позволяет производить контроль типов во время компиляции и находить заранее (в процессе компиляции, а не во время выполнения программы) многие программные ошибки.

3 ДЕМОНСТРАЦИЯ РАБОТЫ

На рисунке 3.1 продемонстрирована таблица переменных и их типов, которая использовалась для проверки их типов.

```
{'n0': 'int', 'n1': 'int', 'n2': 'int'}
```

Рисунок 3.1 – Таблица переменных и их типов

На рисунке 3.2 продемонстрирована семантическая несоответствие типа при объявлении. При отсутствии каких-либо ошибок программный продукт продолжает свою работу.

```
input.c:2:3 : error: Неверное объявление переменной int n1 = 3.4f
```

Рисунок 3.2 – Семантическая ошибка

На рисунке 3.3 продемонстрирована семантическая ошибка попытка провести недопустимую операцию между типами float и int.

```
input.c:7:3 : error: Неверное присваивание переменной          n2 = n0 + f ;
```

Рисунок 3.3 – Семантическая ошибка

Также учитывается неявное приведение типов, поэтому если оно есть, ни каких ошибок не будет и программный продукт продолжает свою работу.

На рисунке 3.4 продемонстрирована работа приложения при неявном приведении типа double и float.

```
public static void main(String args) {
    int n0 = 1 ;
    int n1 = 1 ;
    int n2 ;
    double d = 3.4f ;
    System.out.println ( n0+""+n1+"" ) ;
        for  ( int i = 3 ; i <= 11 ; i ++ ) {
            n2 = n0 + n1 ;
            System.out.println ( n2+"" ) ;
            n0 = n1 ;
            n1 = n2 ;
    }
    System.out.println ( ) ;
}
{'n0': 'int', 'n1': 'int', 'n2': 'int', 'd': 'double'}
(prog (def (funcHeader (modifier public)  static  (type void) main ( (args (arg (type String) args)) )) (block { (stat (type int) n0 (oper =)
 (expr 1) ;) (stat (type int) n1 (oper =) (expr 1) ;) (stat (type int) n2 ;) (stat (type double) d (oper =) (expr 3.4f) ;) (stat for ( (type
 int) i = (expr 3)  ;  i (oper <=) (expr 11)  ;  (expr i) (oper ++) ) (block { (stat n2 = (expr n0) (simpleOper  + ) (expr n1) ;) (stat n0 =
 (expr n1) ;) (stat n1 = (expr n2) ;) })) }))))
```

Рисунок 3.4 – Работа приложения при неявном приведении

ПРИЛОЖЕНИЕ А КОД ПРОГРАММЫ

```
import re
import sys


def out_red(text):
    print("\033[31m {}" .format(text))


keyWords = {"while", "for", "if", "else", "int",
            "float", "break", "continue", "double",
            "array", "false", "true"}
loop = {"while", "for"}
operators = {"+", "-", "*", "/", "%",
             "=", ">", "<", ">=", "<=", "==", "!=",
             "++", "--", "+=", "-=", "*=", "/="}
types = {"int ", "float ", "double ", "boolean ", "char ", "string "}
par = {"(", ")", "{", "}"}
prints = {"System.out.println"}
comment = {"//"}
int_perem = []
float_perem = []
double_perem = []
boolean_perem = []
char_perem = []
string_perem = []
int_p = r'[0-9+]$'
boolean_true_p = r'true$'
boolean_false_p = r'false$'
double_p = r'[0-9]+[.]{1}[0-9]+$$'
float_p = r'[0-9]*[.]{1}[0-9]+[f{1}]$'
char_p = r'\'[a-zA-Z-_*!@#$%^&()>/.,<=+0-9]\'$'
string_p = r'\"[a-zA-Z-_*!@#$%^&()>/.,<=+0-9]+\"$'
dict_perem = {}
f = open('D://java_not_error.txt', 'r')
text = f.read()
one_word = text.split()
print(text)
f.close()

line = 0
for words in types:
    for word in text.split("\n"):
        if words in word and word.split()[0] not in loop:
            i = word.index(words)
            if ("int" in words):
                if ("=" in word):
                    line += 1
                    #isTrue = re.match(int_p, word.split()[k+1])
                    if ((re.match(int_p, word.split()[3]) is None) and
(re.match(double_p, word.split()[3]) is None)):
                        out_red("input.c:%d:%d : error: Неверное объявление
переменной int %s = %s " %(line, 3, word.split()[i], word.split()[3]))
                        sys.exit()
                int_perem.append(word.split()[i])
                dict_perem[word.split()[i]] = "int"
            elif ("float" in words):
                if ("=" in word):
                    line += 1
                    #isTrue = re.match(int_p, word.split()[k+1])
                    if ((re.match(int_p, word.split()[3]) is None) and
(re.match(float_p, word.split()[3]) is None)):
                        out_red("input.c:%d:%d : error: Неверное объявление
```

```
переменной float %s = %s " %(line, 3, word.split()[i], word.split()[3]))
                        sys.exit()
                float_perem.append(word.split()[i])
                dict_perem[word.split()[i]] = "float"
            elif ("double" in words):
                if ("=" in word):
                    line += 1
                    #isTrue = re.match(int_p, word.split()[k+1])
                    if ((re.match(int_p, word.split()[3]) is None) and
(re.match(float_p, word.split()[3]) is None) and (re.match(double_p,
word.split()[3]) is None)):
                        out_red("input.c:%d:%d : error: Неверное объявление
переменной float %s = %s " %(line, 3, word.split()[i], word.split()[3]))
                        sys.exit()
                double_perem.append(word.split()[i])
                dict_perem[word.split()[i]] = "double"
            elif ("boolean" in words):
                if ("=" in word):
                    line += 1
                    #isTrue = re.match(int_p, word.split()[k+1])
                    if ((re.match(boolean_true_p, word.split()[3]) is None)
and (re.match(boolean_false_p, word.split()[3]) is None)):
                        out_red("input.c:%d:%d : error: Неверное объявление
переменной boolean %s = %s " %(line, 3, word.split()[i], word.split()[3]))
                        sys.exit()
                boolean_perem.append(word.split()[i])
                dict_perem[word.split()[i]] = "boolean"
            elif ("char" in words):
                if ("=" in word):
                    line += 1
                    #isTrue = re.match(int_p, word.split()[k+1])
                    if (re.match(char_p, word.split()[3]) is None):
                        out_red("input.c:%d:%d : error: Неверное объявление
переменной char %s = %s " %(line, 3, word.split()[i], word.split()[3]))
                        sys.exit()
                char_perem.append(word.split()[i])
                dict_perem[word.split()[i]] = "char"
            elif ("string" in words):
                if ("=" in word):
                    line += 1
                    #isTrue = re.match(int_p, word.split()[k+1])
                    if (re.match(string_p, word.split()[3]) is None):
                        out_red("input.c:%d:%d : error: Неверное объявление
переменной string %s = %s " %(line, 3, word.split()[i], word.split()[3]))
                        sys.exit()
                string_perem.append(word.split()[i])
                dict_perem[word.split()[i]] = "string"
            continue

print(dict_perem)

line = 0
for word in text.split("\n"):
    line += 1
    if len(word.split()) > 3:
        if word.split()[1] == "=":
            if word.split()[0] in int_perem:
                if len(word.split()) == 4:
                    if word.split()[2] in int_perem:
                        continue
                    else:
                        out_red("input.c:%d:%d : error: Неверное присваивание
переменной %s " % (line, 3, word))
```

```
                        sys.exit()
                else:
                    for p in range(len(word.split())):
                        if p % 2 == 0:
                            if word.split()[p] in int_perem:
                                continue
                            else:
                                out_red("input.c:%d:%d : error: Неверное
присваивание переменной %s " % (line, 3, word))
                                sys.exit()
            if word.split()[0] in float_perem:
                if len(word.split()) == 4:
                    if ((word.split()[2] in float_perem) or (word.split()[2]
in int_perem)):
                        continue
                    else:
                        out_red("input.c:%d:%d : error: Неверное присваивание
переменной %s " % (line, 3, word))
                        sys.exit()
                else:
                    for p in range(len(word.split())):
                        if p % 2 == 0:
                            if (word.split()[p] in int_perem) or
(word.split()[p] in float_perem):
                                continue
                            else:
                                out_red("input.c:%d:%d : error: Неверное
присваивание переменной %s " % (line, 3, word))
                                sys.exit()
            if word.split()[0] in double_perem:
                if len(word.split()) == 4:
                    if ((word.split()[2] in float_perem) or (word.split()[2]
in double_perem) or (word.split()[2] in int_perem)):
                        continue
                    else:
                        out_red("input.c:%d:%d : error: Неверное присваивание
переменной %s " % (line, 3, word))
                        sys.exit()
                else:
                    for p in range(len(word.split())):
                        if p % 2 == 0:
                            if (word.split()[p] in double_perem) or
(word.split()[p] in float_perem) or (word.split()[p] in int_perem):
                                continue
                            else:
                                out_red("input.c:%d:%d : error: Неверное
присваивание переменной %s " % (line, 3, word))
                                sys.exit()
            if word.split()[0] in boolean_perem:
                if len(word.split()) == 4:
                    if (word.split()[2] in boolean_perem):
                        continue
                    else:
                        out_red("input.c:%d:%d : error: Неверное присваивание
переменной %s " % (line, 3, word))
                        sys.exit()
            if word.split()[0] in char_perem:
                if len(word.split()) == 4:
                    if (word.split()[2] in char_perem):
                        continue
                    else:
                        out_red("input.c:%d:%d : error: Неверное присваивание
переменной %s " % (line, 3, word))
```

```
                    sys.exit()
            if word.split()[0] in string_perem:
                if len(word.split()) == 4:
                    if (word.split()[2] in string_perem):
                        continue
                    else:
                        out_red("input.c:%d:%d : error: Неверное присваивание
переменной %s " % (line, 3, word))
                        sys.exit()
```

ПРИЛОЖЕНИЕ Б КОД ПРОГРАММЫ

```
grammar tree;

prog:   dec1 | def;

dec1:   funcHeader ';'
    |   type ID ';'
    ;

def :   funcHeader block
    |   type ID ('=' expr)? ';'
    ;

block : '{' stat* '}' ;

stat:  ID ('=' expr)? (simpleOper expr)?';'
    |  'for' '(' type ID ('=' expr)? ' ; ' ID oper expr ' ; ' expr oper
(expr)?')' stat
    |  'while' '(' expr ')' stat
    |  type ID (oper expr)? (simpleOper expr)? ';'
    |  'System.out.println' '(' ((vivod)+)? ')' ';'
    |  'return' expr ';'
    |  expr ';'
    |  block
    ;

expr:   '(' expr ')'
    |   ID '(' expr (',' expr)* ')'
    |   ID
    |   INT
    |   CHAR
    ;

type: 'int' | 'char' | 'void' | 'float' | 'double' | 'String' ;

modifier: 'public' | 'protected' | 'private' ;

vivod: ID'+'((quote)+)?(ID)?((quote)+)?('+')? ;

oper: '<=' | '>=' | '<' | '>' | '+=' | '++' | '--' | '-=' | '=' ;

simpleOper: ' + ' | ' - ' | ' * ' | ' / ' | '+' | '-' | '*' | '/';

quote: ' " ' | '"' | ' "' ;

funcHeader : (modifier)? (' static ')? (type) ID '(' args? ')' ;

args: arg (',' arg)*;

arg : type ID ;

COMMENT : '/*' .*? '*/' -> channel (HIDDEN);

WS : [ \t\n\r]+ -> skip ;

ID : [a-zA-Z_]+ [a-zA-Z0-9_]*;

INT : [0-9]+ ;

CHAR : '\' '~'\''+'\'' ;
```