

Министерство образования Республики Беларусь
Учреждение образования «Белорусский государственный университет
информатики и радиоэлектроники»

Факультет компьютерных систем и сетей
Кафедра информатики
Дисциплина «Методы трансляции»

ОТЧЕТ
к лабораторной работе № 5
на тему «Интерпретация исходного кода»

Выполнил

Я. Ю. Прескурел

Проверил

Н. Ю. Гриценко

Минск 2024

СОДЕРЖАНИЕ

1 Постановка задачи.....	3
2 Краткие теоретические сведения.....	4
3 Результаты выполнения лабораторной работы.....	5
Выводы	6
Список использованных источников	9
Приложение А (обязательное) Листинг исходного кода	10

1 ПОСТАНОВКА ЗАДАЧИ

Целью выполнения данной лабораторной работы является на основе результатов анализа лабораторных работы 1-4 выполнить трансляцию программы с языка программирования Python на язык программирования C#, после чего выполнить интерпретацию программы.

2 КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

К этапам трансляции относятся следующие этапы:

- лексический анализ;
- синтаксический анализ;
- семантический анализ;
- оптимизация;
- генерация кода.

На этапе генерации компилятор создает код, который представляет собой набор инструкций, понятных для целевой аппаратной платформы, итоговый файл компилируется в исполняемый файл, который может быть запущен на целевой платформе без необходимости наличия кода.

Фаза эмуляции интерпретатора происходит во время выполнения программы. В отличие от компилятора, интерпретатор работает с кодом напрямую, без предварительной генерации машинного кода.

Лексический анализатор – первый этап трансляции. Лексический анализатор читает поток символов, составляющих исходную программу, и группирует эти символы в лексемы или значащие последовательности. Лексема – это элементарная единица, которая может являться ключевым словом, идентификатором, константным значением. Для каждой лексемы анализатор строит токен, который по сути является кортежем, содержащим имя и значение.[1]

Синтаксический анализатор выясняет, удовлетворяют ли предложения, из которых состоит исходная программа, правилам грамматики языка программирования. Синтаксический анализатор получает на вход результат лексического анализатора и разбирает его в соответствии с грамматикой. Результат синтаксического анализа обычно представляется в виде синтаксического дерева разбора.[2]

Семантический анализ обычно заключается в проверке правильности типа и вида всех идентификаторов и данных, используемых в программе.

Семантический анализатор использует синтаксическое дерево и информацию из таблицы символов для проверки исходной программы на семантическую согласованность с определением языка. Он также собирает информацию о типах и сохраняет ее в синтаксическом дереве или в таблице идентификаторов для последующего использования в процессе генерации промежуточного кода.

В данной лабораторной работе были использованы результаты анализа лексического, синтаксического и семантического анализаторов, после чего каждый узел дерева разбора был переведен с языка программирования Python на язык программирования C#. После чего была выполнена интерпретация программ. Программами называются тестовые исходные коды, представленные в лабораторной работе 1.

3 РЕЗУЛЬТАТЫ ВЫПОЛНЕНИЯ ЛАБОРАТОРНОЙ РАБОТЫ

В ходе лабораторной работы был реализован транслятор программ с языка программирования Python на язык программирования C# с последующей интерпретацией кода.

Листинг первого тестового кода представлен на рисунке 3.1.

```
a = []
b = []

a[0] = 4
a[1] = 4
a[2] = 2
a[3] = 1

b[0] = 8
b[1] = 8
b[2] = 6
b[3] = 5

c = a + b
print(c)

i = 0
while i < len(c):
    j = i + 1
    while j < len(c):
        if c[i] > c[j]:
            temp = c[i]
            c[i] = c[j]
            c[j] = temp
        j += 1
    i += 1
print(c)

temp = []

i = 0
while i < len(c):
    x = c[i]
    r = 1
    j = 0
    while j < len(temp):
        if temp[j] == x:
            r = 0
        j += 1
    if r:
        temp += x
    i += 1

print('Updated list after removing duplicates = ' + temp)
```

Рисунок 3.1 – Листинг первого тестового кода

Результат интерпретации первого исходного кода представлен на рисунке 3.2.

```
C:\WINDOWS\system32\cmd. x + v
Analysing test.txt
Interpeting Assigning a = []
Interpeting Assigning b = []
Interpeting Assigning a[0] = 4
Interpeting Assigning a[1] = 4
Interpeting Assigning a[2] = 2
Interpeting Assigning a[3] = 1
Interpeting Assigning b[0] = 8
Interpeting Assigning b[1] = 8
Interpeting Assigning b[2] = 6
Interpeting Assigning b[3] = 5
Interpeting Assigning c = a + b
Interpeting calling function print with (c)
[4, 4, 2, 1, 8, 8, 6, 5]
Interpeting Assigning i = 0
Interpeting while statement: i < calling function len with (c)
```

Рисунок 3.2 – Результат интерпретации первого исходного кода

Листинг второго исходного кода представлен на рисунке 3.3.

```
from random import randint

N = 10
a = []
for i in range(N):
    a = a + i
print(a)

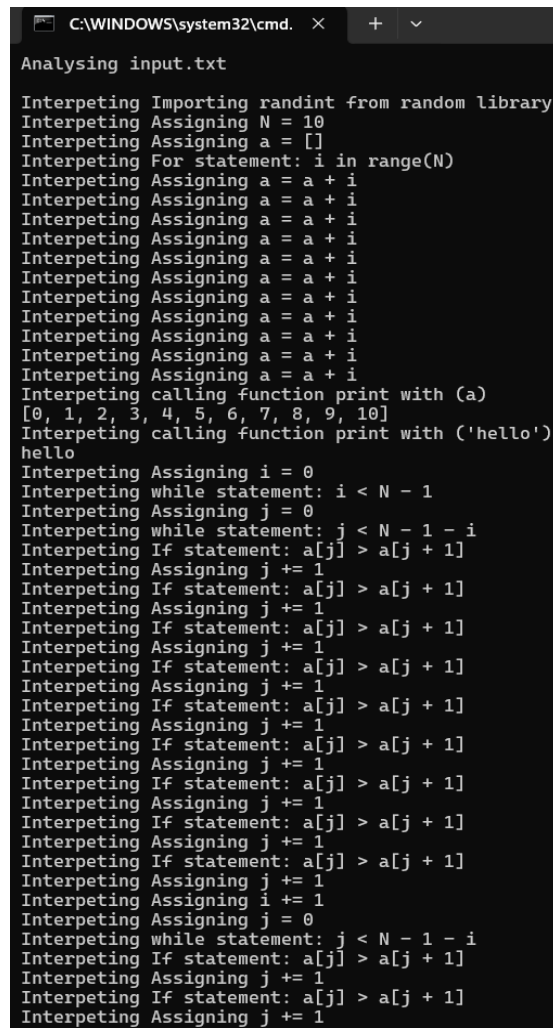
print('hello')

i = 0
while i < N - 1:
    j = 0
    while j < N - 1 - i:
        if a[j] > a[j + 1]:
            temp = a[j + 1]
            a[j + 1] = a[j]
            a[j] = temp
        j += 1
    i += 1

print(a)
```

Рисунок 3.3 – Листинг второго тестового кода

Результат интерпретации второго тестового кода представлен на рисунке 3.4.



```
C:\WINDOWS\system32\cmd.  X  +  v
Analysing input.txt

Interpeting Importing randint from random library
Interpeting Assigning N = 10
Interpeting Assigning a = []
Interpeting For statement: i in range(N)
Interpeting Assigning a = a + i
Interpeting Assigning a = a + i
Interpeting Assigning a = a + i
Interpeting Assigning a = a + i
Interpeting Assigning a = a + i
Interpeting Assigning a = a + i
Interpeting Assigning a = a + i
Interpeting Assigning a = a + i
Interpeting Assigning a = a + i
Interpeting Assigning a = a + i
Interpeting Assigning a = a + i
Interpeting calling function print with (a)
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
Interpeting calling function print with ('hello')
hello
Interpeting Assigning i = 0
Interpeting while statement: i < N - 1
Interpeting Assigning j = 0
Interpeting while statement: j < N - 1 - i
Interpeting If statement: a[j] > a[j + 1]
Interpeting Assigning j += 1
Interpeting If statement: a[j] > a[j + 1]
Interpeting Assigning j += 1
Interpeting If statement: a[j] > a[j + 1]
Interpeting Assigning j += 1
Interpeting If statement: a[j] > a[j + 1]
Interpeting Assigning j += 1
Interpeting If statement: a[j] > a[j + 1]
Interpeting Assigning j += 1
Interpeting If statement: a[j] > a[j + 1]
Interpeting Assigning j += 1
Interpeting If statement: a[j] > a[j + 1]
Interpeting Assigning j += 1
Interpeting If statement: a[j] > a[j + 1]
Interpeting Assigning j += 1
Interpeting Assigning i += 1
Interpeting Assigning j = 0
Interpeting while statement: j < N - 1 - i
Interpeting If statement: a[j] > a[j + 1]
Interpeting Assigning j += 1
Interpeting If statement: a[j] > a[j + 1]
```

Рисунок 3.4 – Результат интерпретации второго тестового кода

Таким образом в ходе лабораторной работы был реализован интерпретатор для программ на языке Python, который переводит их на язык программирования C# после чего проводит интерпретацию полученного при трансляции кода.

ВЫВОДЫ

В ходе лабораторной работы был реализован был реализован интерпретатор для программ на языке Python, который переводит их на язык программирования C# после чего проводит интерпретацию полученного при трансляции кода.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Лексический анализатор [Электронный ресурс]. – Режим доступа: <https://csc.sibsutis.ru/sites/csc.sibsutis.ru/files/courses/trans/>. – Дата доступа: 14.04.2024.

[2] Синтаксический анализатор [Электронный ресурс]. – Режим доступа: <https://csc.sibsutis.ru/sites/csc.sibsutis.ru/files/courses/trans/>. – Дата доступа: 14.04.2024.

[3] Введение в Python [Электронный ресурс]. – Режим доступа: <https://metanit.com/cpp/tutorial/2.5.php>. – Дата доступа: 14.04.2024.

[4] Типы данных [Электронный ресурс]. – Режим доступа: <https://metanit.com/cpp/tutorial/2.3.php>. – Дата доступа: 14.04.2024.

[5] Операторы в Python [Электронный ресурс]. – Режим доступа: <https://learn.microsoft.com/ru-ru/cpp/c-operators>. – Дата доступа: 14.04.2024.

[6] Функции Python [Электронный ресурс]. – Режим доступа: <https://metanit.com/cpp/tutorial/3.1.php>. – Дата доступа: 14.04.2024.

[7] Классы Python [Электронный ресурс]. – Режим доступа: <https://ravesli.com/urok-113-klassy-obekty-i-metody-klassov/>. – Дата доступа: 14.04.2024.

ПРИЛОЖЕНИЕ А (обязательное) Листинг исходного кода

Листинг 1 – Программный код Interpreter.cs

```
using System;
using System.Collections.Generic;

namespace mtran
{
    internal class Interpreter
    {
        Ast ast;
        int statementIndex = 0;
        List<string> importedModules;
        List<Variable> variables;

        internal Interpreter(Ast ast)
        {
            this.ast = ast;
            this.importedModules = new List<string>();
            this.variables = new List<Variable>();
        }

        internal bool Run()
        {
            while (statementIndex != -1 && statementIndex <
ast.statements.Count)
            {
                var stat = ast.statements[statementIndex];
                var result = InterpretStatement(stat);
                if (!result)
                {
                    return result;
                }
            }

            return true;
        }

        private bool InterpretStatement(Statement stat)
        {
            //Console.WriteLine($"Interpeting {stat}");
            statementIndex++;
            switch (stat.statementType)
            {
                case StatementType.STATEMENT_TYPE_IMPORT:
                {
                    bool result = InterpretImport(stat as Import);
                    if (!result)
                    {
                        ReportError("", stat);

                        return false;
                    }
                }
                break;
                case StatementType.STATEMENT_TYPE_ASSIGNMENT:
                {
```

```

        bool result = InterpretAssignment(stat as
Assignment);
        if (!result)
        {
            ReportError("", stat);

            return false;
        }
    }
    break;
case StatementType.STATEMENT_TYPE_IF:
    {
        bool result = InterpretIf(stat as If);
        if (!result)
        {
            ReportError("", stat);

            return false;
        }
    }
    break;
case StatementType.STATEMENT_TYPE_ELSE:
    {
        bool result = InterpretElse(stat as Else);
        if (!result)
        {
            ReportError("", stat);

            return false;
        }
    }
    break;
case StatementType.STATEMENT_TYPE_FOR:
    {
        bool result = InterpretFor(stat as For);
        if (!result)
        {
            ReportError("", stat);

            return false;
        }
    }
    break;
case StatementType.STATEMENT_TYPE_WHILE:
    {
        bool result = InterpretWhile(stat as While);
        if (!result)
        {
            ReportError("", stat);

            return false;
        }
    }
    break;
case StatementType.STATEMENT_TYPE_FUNCTION_CALL:
    {
        var result = InterpretFunctionCall(stat as
FunctionCall);
        if (result == null)
        {
            ReportError("", stat);

```

```
                return false;
            }
        }
        break;
    default:
    case StatementType.STATEMENT_TYPE_EXPRESSION:
        return false;
    }

    return true;
} }
```