МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

Московский авиационный институт

(национальный исследовательский университет)

Институт № 8

Компьютерные науки м прикладная математика

Кафедра 806 «Вычислительная математика и программирование»

КУРСОВАЯ РАБОТА

по дисциплине «Системы программирования»

Тема: Разработать и реализовать алгоритм промежуточного представления условного оператора в Python

Выполнил: студент группы М8О-201Б-21

Старцев Иван Романович

(фамилия, имя, отчество)

_____

(подпись)

Принял: доцент кафедры 806

Киндинова Виктория Валерьевна

_____

(подпись)

Оценка: **Дата:**

Москва, 2023

Теоретическая справка

В компиляторах часто используется постфиксная форма записи, где операции записываются справа от операндов. Эта форма записи называется ПОЛИЗ (польская инверсная запись), которая определяется следующими правилами:

I. Каждая переменная или константа является выражением.

II. Если перед выражением стоит унарная операция (одноместная операция), то это новое выражение записывается после выражения с этой операцией.

III. Если перед выражениями стоит бинарная операция (двухместная операция), то это новое выражение записывается после двух выражений с этой операцией.

IV. Если перед выражениями стоит n-местная операция, то это новое выражение записывается после n выражений с этой операцией.

V. Нет других выражений.

Реализуем промежуточное представление операндов, приближенное к постфиксной форме.

Рассмотрим условный оператор языка Python, имеющий формат:

```
if <выражение>:
        <оператор1>
else:
        <оператор2>
```

Для представления оператора if в ПОЛИЗ необходимо ввести понятия следующих операторов в ПОЛИЗ:

1) Оператор присваивания

Для того, чтобы представить оператор присваивания в польской инверсной записи, нужно использовать бинарную операцию присвоения значения (:=). Левый операнд этой операции определяет объект, которому нужно присвоить значение, а правый операнд - вычисленное значение. Например, оператор присваивания "x = 26" в языке Python может быть записан в ПОЛИЗ в виде "x 26 :=".

2) Условный оператор

Чтобы добавить возможность представления условного оператора в польской инверсной записи, нужно ввести два новых объекта:

а) Метки (m_1, m_2, m_3, …, m_i), которые могут помечать определенные элементы ПОЛИЗ и использоваться для динамического изменения хода вычислений.

б) Дополнительные операции:

- BF - условный переход по значению "ложь". Эта операция имеет два операнда: r и m. Смысл этой операции заключается в переходе на метку m, если r = false.
- BRL - безусловный переход на метку. Эта операция имеет один операнд: m. Смысл этой операции заключается в переходе на метку m.
- DEFL - определение метки. Эта операция имеет один операнд: m. Смысл этой операции заключается в определении метки m.

Приведём пример представления условного оператора в Python:

```
if x > y:
    c = 26
else:
    c = 13
```

В ПОЛИЗ будет выглядеть следующим образом:

x y > m1 BF c 26 := m2 BRL m1 DEFL c 13 := m2 DEFL

S - <условный оператор>, A - <заголовок условного оператора>, C - <оператор1, если выражение *true*>, D - <оператор2, если выражение *false*>, X и Z - <переменные для выражения условного оператора>, Y - <логический оператор>.

СУ-схема перевода

N	Правила грамматики G_0	Элемент перевода G_1
1	S → A C D	S → A C D
2	A → if X Y Z : \n	A → X Z Y m_i BF
3	X → cns	X → cns
4	Y → cns	Y → cns
5	Z → cns	Z → cns
6	C → \t cns \n else: \n	C → cns m_j BRL m_i DEFL
7	D → \t cns \n	D → cns m_j DEFL

Рассмотрим следующий пример:

```
if a > b:
    print(a)
```

$S_{КС2}$ =>[1] A C D =>[2] X Z Y m_1 BF C D =>[3] a Z Y m_1 BF C D =>[4,5] a b > m_1 BF C D =>[6] a b > m_1 BF print(a) m_2 BRL m_1 DEFL D =>[7] a b > m_1 BF print(a) m_2 BRL m_1 DEFL m_2 DEFL

$S_{КС1}$ =>[1] A C D =>[2] if X Y Z: \n C D =>[3,4,5] if a > b: \n C D =>[6] if a > b: \n\t print(a) \nelse:\n D =>[7] if a > b: \n\t print(a) \nelse:\n\t\n

Описание алгоритма

Программа принимает на вход строку, которая содержит в себе код на Python, описывающий условный оператор. Создаются переменные, отвечающие за строку с ответом, итераторы по массиву строк и счетчик для меток переходов. Они будут передаваться в рекурсивную функцию по ссылке, чтобы они могли изменять там своё значение и этот результат сохранялся после завершения работы функции.

С помощью вспомогательной функции обрезаются все лишние символы и вызывается рекурсивная функция. Данная рекурсивная функция вызывается для каждого вхождения условного оператора if, включая ветвления условных операторов одного в другом.

В самой функции происходит последовательная обработка всех строк, рассматриваются все возможные ситуации, постепенно условные операторы на Python преобразуются в ПОЛИЗ. Сначала выражение, которое лежит в основе условного оператора, перерабатывается в вид обратной польской записи с помощью вспомогательной функции, добавляются основные операторы ПОЛИЗ и метки для переходов. После этого в зависимости от того какое значение принимает выражение (истина или ложь), обрабатываются остальные строки и добавляются необходимые операторы и метки переходов.

Выход из рекурсии осуществляется, когда итератор по строкам сравнивается с количеством элементов массива строк. После этого выводится ответ.

Пример работы

Разберем работу программы для следующих входных данных:

```
if a > 10:
        print("AAA")
         if z > 20:
                print("ZZZ")
        else:
                print("!Z")
else:
        print("!A")
```

После обрезки всех лишних символов получаем следующий массив строк:

```
if a > 10:
print("AAA")
if z > 20:
print("ZZZ")
else:
print("!Z")
else:
print("!A")
```

Создаем глобальные переменные, которые будут передаваться по ссылке, вызываем рекурсивную функцию POLIZ. Она вызывается каждый раз, когда у нас открывается новое ветвление, начинающиеся с if.

Итого после вызова POLIZ и проверки первого условия строка с ответом такая:

“a 10 > m1 BF”

Далее, так как у нас после if идет простая команда, а не условный оператор, то мы просто записываем её в строку:

“a 10 > m1 BF **print("AAA")**”

После этого обработчик программы снова встречает строку, которая начинается с if и вызывает функцию POLIZ. Программа продолжит работу с нашим основным первым if только после того, как закончится рекурсивный вызов.

Теперь имеем строку :

“a 10 > m1 BF print("AAA") **z 20 > m2 BF print("ZZZ") m3 BRL m2 DEFL print("!Z") m3 DEFL**”

Встречаем строку с “else:” нашего основного условного оператора, добавляем все необходимые метки и операторы ПОЛИЗ. Получаем строку:

“a 10 > m1 BF print("AAA") z 20 > m2 BF print("ZZZ") m3 BRL m2 DEFL print("!Z") m3 DEFL **m4 BRL m1 DEFL**”

Для того, чтобы написать корректную метку для перехода (в данном случае m1 в конце строки с ответом), мы должны сохранить информацию о её порядковом номере для каждого условного оператора.

Снова встречаем простую строку без операторов и дописываем её:

“a 10 > m1 BF print("AAA") z 20 > m2 BF print("ZZZ") m3 BRL m2 DEFL print("!Z") m3 DEFL m4 BRL m1 DEFL **print("!A")**”

Понимаем, что это последняя строка в нашем массиве строк, следовательно больше никакой информации не поступит, поэтому ставим завершающие метки и операторы ПОЛИЗ. Это условие также выполняется, если следующая строка содержит “else:”, потому что это значит, что мы во вложенном условном операторе, и он подошёл к своему концу.

“a 10 > m1 BF print("AAA") z 20 > m2 BF print("ZZZ") m3 BRL m2 DEFL print("!Z") m3 DEFL m4 BRL m1 DEFL print("!A") **m4 DEFL**”

Программа завершила свою работу. Получено представление условного оператора в ПОЛИЗ.

Итоговый ответ:

“a 10 > m1 BF print("AAA") z 20 > m2 BF print("ZZZ") m3 BRL m2 DEFL print("!Z") m3 DEFL m4 BRL m1 DEFL print("!A") m4 DEFL”

Псевдокод

```
function Main()
    source = "…"
    print(source)
    print("**********************************************************")
    lines = SplitString(source)

    foreach line in lines
        print(line)

    print("**********************************************************")
    ans = ""
    linesIterator = 0
    mCounter = 1
    POLIZ(ans, lines, linesIterator, mCounter)

    print(ans)

function POLIZ(ans, lines, linesIterator, mCounter)
    if lines[linesIterator] starts with "if"
        ans += " " + MakePostfix(TrimIf(lines[linesIterator]))
        ans += " m" + mCounter
        prevCounter = mCounter
        mCounter += 1
        ans += " BF"
        linesIterator += 1
        isInIf = true

        while linesIterator < lines.length
            if lines[linesIterator] starts with "if"
                POLIZ(ans, lines, linesIterator, mCounter)
            else if lines[linesIterator] starts with "else"
                isInIf = false
                ans += " m" + mCounter
                ans += " BRL"
```

```
                ans += " m" + prevCounter
                ans += " DEFL"
            else
                if isInIf
                    ans += " " + lines[linesIterator]
                else
                    if linesIterator < lines.length - 1 and lines[linesIterator + 1]
starts with "else:"
                        ans += " " + lines[linesIterator]
                        ans += " m" + mCounter
                        ans += " DEFL"
                        mCounter += 1
                        return
                    else
                        ans += " " + lines[linesIterator]
                        if linesIterator == lines.length - 1
                            ans += " m" + mCounter
                            ans += " DEFL"
                            mCounter += 1
                if linesIterator == lines.length - 1
                    return
                linesIterator += 1

    else
        throw ArgumentException("Invalid input format. Expected format: 'if ...:'")

function SplitString(input)
    lines = input.split("\n")

    for i = 0 to lines.length - 1
        lines[i] = lines[i].trim()
        if lines[i] starts with "\t"
            lines[i] = lines[i].substring(1)

    return lines
```

```
function MakePostfix(input)
    parts = input.split(" ")

    if parts.length != 3
        throw ArgumentException("Invalid input format. Expected format: 'A
logical_expression B'")

    a = parts[0]
    oper = parts[1]
    b = parts[2]

    return join(" ", a, b, oper)

function TrimIf(input)
    startIndex = input.indexOf("if ") + 3
    endIndex = input.lastIndexOf(":")

    if startIndex >= 0 and endIndex >= 0 and endIndex > startIndex
        expression = input.substring(startIndex, endIndex - startIndex).trim()
        return expression

    throw ArgumentException("Invalid input format. Expected format: 'if
<expression>:'")

Main()
```

Листинг программы

```
using System;

public class Program
{
    public static void Main()
    {
        string source = "if a > 10:\n\tprint(\"AAA\")\nif z >
20:\n\t\tprint(\"ZZZ\")\n\telse:\n\t\tprint(\"!Z\")\nelse:\n\tprint(\"!A\")";

        Console.WriteLine(source);


Console.WriteLine("************************************************************");

        string[] lines = SplitString(source);
        foreach (string line in lines)
        {
            Console.WriteLine(line);
        }


Console.WriteLine("************************************************************");

        string ans = "";
        int linesIterator = 0;
        int mCounter = 1;
        POLIZ(ref ans, lines, ref linesIterator, ref mCounter);

        Console.WriteLine(ans);

    }

    public static void POLIZ(ref string ans, in string[] lines, ref int
linesIterator, ref int mCounter)
    {
        if (lines[linesIterator].StartsWith("if"))
        {
            ans += " " + MakePostfix(TrimIf(lines[linesIterator]));
            ans += " m";
            ans += mCounter.ToString();
            int prevCounter = mCounter;
            ++mCounter;
            ans += " BF";
            ++linesIterator;

            bool isInIf = true;

            while (linesIterator < lines.Length)
            {
                if (lines[linesIterator].StartsWith("if"))
                {
                    POLIZ(ref ans, lines, ref linesIterator, ref mCounter);
                }
                else if (lines[linesIterator].StartsWith("else"))
                {
                    isInIf = false;
                    ans += " m";
                    ans += mCounter.ToString();
                    ans += " BRL";
                    ans += " m";
                    ans += prevCounter.ToString();
                    ans += " DEFL";
```

```
                }
                else
                {
                    if (isInIf)
                    {
                        ans += " " + lines[linesIterator];
                    }
                    else
                    {
                        if (linesIterator < lines.Length - 1 && lines[linesIterator
+ 1].StartsWith("else:"))
                        {
                            ans += " " + lines[linesIterator];
                            ans += " m";
                            ans += mCounter.ToString();
                            ans += " DEFL";
                            ++mCounter;
                            return;
                        }
                        else
                        {
                            ans += " " + lines[linesIterator];
                        }

                        if (linesIterator == lines.Length - 1)
                        {
                            ans += " m";
                            ans += mCounter.ToString();
                            ans += " DEFL";
                            ++mCounter;
                        }
                    }
                }

                if (linesIterator == lines.Length - 1)
                {
                    return;
                }

                ++linesIterator;
            }


        } else
        {
            throw new ArgumentException("Invalid input format. Expected format: 'if
...:'");
        }
    }

    public static string[] SplitString(string input)
    {
        string[] lines = input.Split('\n'); // Разбиваем строку на массив строк по
символу '\n'

        for (int i = 0; i < lines.Length; i++)
        {
            lines[i] = lines[i].Trim(); // Удаляем лишние пробелы в начале и конце
каждой строки

            if (lines[i].StartsWith("\t")) // Удаляем символы '\t' в начале каждой
строки
            {
                lines[i] = lines[i].Substring(1);
            }
```

```
        }

        return lines;
    }

    public static string MakePostfix(string input)
    {
        string[] parts = input.Split(' '); // Разбиваем строку на массив подстрок по
пробелам

        if (parts.Length != 3)
        {
            throw new ArgumentException("Invalid input format. Expected format: 'A
лог. выражение B'");
        }

        string a = parts[0];
        string oper = parts[1];
        string b = parts[2];

        return string.Join(" ", a, b, oper); // Соединяем подстроки с пробелами
между ними
    }

    public static string TrimIf(string input)
    {
        int startIndex = input.IndexOf("if ") + 3; // Находим начальный индекс
выражения
        int endIndex = input.LastIndexOf(":"); // Находим конечный индекс выражения

        if (startIndex >= 0 && endIndex >= 0 && endIndex > startIndex)
        {
            string expression = input.Substring(startIndex, endIndex -
startIndex).Trim(); // Извлекаем выражение и обрезаем лишние пробелы
            return expression;
        }

        throw new ArgumentException("Invalid input format. Expected format: 'if
<выражение>:'");
    }

}
```