Практикум на ЭВМ. Интерпретатор. Полиз

Баев А.Ж.

Казахстанский филиал МГУ

07 февраля 2020

План на семестр

- 1. Интерпретатор
- 2. Web сервер
- 3. Параллельное программирование

Интерпретатор

- 1. Арифметические операторы
- 2. Оператор присваивания
- 3. Логические операторы
- 4. Оператор перехода (goto)
- 5. Условный оператор
- 6. Цикл while
- 7. Массивы
- 8. Функции
- 9. Рекурсия (стек для вызова функций)

Интерпретатор

1. Лексический анализ строка

$$10 + 2$$

2. Синтаксический анализ инфикс

3. Вычисление постфикс

инфикс

10	+	2

постфикс

значение

12

Реализация функций

```
#include <string>
#include <vector>
std::vector<Lexem *> parseLexem(
    std::string codeline);
std::vector<Lexem *> buildPoliz(
    std::vector<Lexem *> infix);
int evaluatePoliz(
    std::vector<Lexem *> poliz);
```

3

5

6

8

9 10

11

Реализация основной функции

```
int main() {
       std::string codeline;
3
       std::vector<Lexem *> infix;
4
       std::vector<Lexem *> postfix;
5
       int value;
6
       while (std::getline(std::cin, codeline)) {
8
            infix = parseLexem(codeline);
9
            postfix = buildPostfix(infix);
10
            value = evaluatePostfix(postfix);
11
            std::cout << value << std::endl;
12
13
       return 0;
14
```

Реализация классов

```
class Lexem {
   public:
       Lexem();
4
   };
5
6
   class Number: public Lexem {
        int value;
8
   public:
9
        Number();
10
        int getValue();
11
   };
```

Реализация классов

```
enum OPERATOR {
       PLUS, MINUS,
3
       MULTIPLY,
4
       LBRACKET, RBRACKET
5
   };
6
   int PRIORITY[] = {
       0,0,
8
       1,
9
       2, 2
10
   };
11
   class Oper: public Lexem {
12
        OPERATOR opertype;
13
   public:
14
       Oper();
15
       OPERATOR getType();
16
        int getPriority();
17
        int getValue(const Number& left,
18
                      const Number& right);
19
                                       ◆□▶→□▶→□▶→□▶□□
```

Лексический анализ

1. склеить цифры в числа

```
1    number = 0;
2    for (int i = 0; i < codeline.size(); i++)
3         number := number * 10 + codeline[i] - '0';
```

- 2. игнорировать пробелы и табуляции
- 3. обрабатывать до конца строки

Синтаксический анализ (алгоритм построения полиза «сортировочная станция»)

Пример:

1 + 2

символ входной строки стек операторов

конец строки

+		
+		

выходная строка

1			
1			
1	2		
1	2	+	

Синтаксический анализ (алгоритм построения полиза «сортировочная станция»)

Пример:

1 + 2 * 4

символ входной строки стек операторов

конец строки

+			
+			
+	*		
+	*		

выходная строка

1				
1				
1	2			
1	2			
1	2	4		
1	2	4	*	+

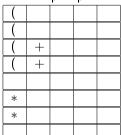
Синтаксический анализ (алгоритм построения полиза «сортировочная станция»)

Пример:

$$(1+2)*4$$

конец строки

стек операторов



выходная строка

1				
1				
1				
1	2			
1	2	+		
1	2	+		
1	2	+	4	
1	2	+	4	*

Синтаксический анализ (алгоритм построения полиза «сортировочная станция»)

Пример:

$$1 + 2 * 3 * (9 - 4)$$

Синтаксический анализ (алгоритм построения полиза «сортировочная станция»)

- 1. Читаем очередной символ.
- 2. Если символ является числом, добавляем его к выходной строке.
- Если символ является открывающей скобкой, помещаем его в стек.
- 4. Если символ является закрывающей скобкой, то выталкиваем все элементы из стека в выходную строку, пока верхним элементом стека не станет открывающая скобка.
- 5. Если символ является бинарной операцией «oper», то пока на вершине стека операция на вершине стека имеет приоритет больше (или равен для левоассоциативной операции), чем «oper», то выталкиваем верхний элемент стека в выходную строку, в конце помещаем «oper» в стек.
- 6. Когда входная строка закончилась, выталкиваем все символы из стека (операторы) в выходную строку.

Пример:

$$1 + 2$$

Польская инверсная запись:

Стек для вычислений:

Пример:

$$1 + 2 * 4$$

Польская инверсная запись:

1 2	4	*	+
-----	---	---	---

Стек для вычислений:

этек для вычислени						
1	1					
2	1	2				
4	1	2	4			
*	1	8				
+	9					

Пример:

$$(1+2)*4$$

Польская инверсная запись:

_				
1	2	+	4	*

Стек для вычислений:

лек для вычислени						
1	1					
2	1	2				
+	3					
4	3	4				
*	12					

Пример:

$$1 + 2 * 3 * (9 - 4)$$

Польская инверсная запись:

1	2	3	*	9	4	_	*	+

Стек для вычислений

стек ,	стек для вычислении:								
1	1								
2	1	2							
3	1	2	3						
*	1	6							
9	1	6	9						
4	1	6	9	4					
_	1	6	5						
*	1	30							
_	31								