

Практикум на ЭВМ.
План на семестр.
Интерпретатор. Полиз

Баев А.Ж.

Казахстанский филиал МГУ

07 февраля 2020

План на семестр

1. Интерпретатор
2. Web сервер
3. Параллельное программирование
4. Обработка звука (VAD, diarization, STT, TTS)

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 (полиз)

Пример:

$$1 + 2$$

Польская инверсная запись:

1	2	+
---	---	---

Стек для вычислений:

1	1		
2	1	2	
+	3		

Интерпретатор (полиз)

Пример:

$$1 + 2 * 4$$

Польская инверсная запись:

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

Стек для вычислений:

1	1		
2	1	2	
4	1	2	4
*	1	8	
+	9		

Интерпретатор (полиз)

Пример:

$$(1 + 2) * 4$$

Польская инверсная запись:

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

Стек для вычислений:

1	1		
2	1	2	
+	3		
4	3	4	
*	12		

Интерпретатор (полиз)

Пример:

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

Польская инверсная запись:

1	2	3	*	9	4	-	*	+
---	---	---	---	---	---	---	---	---

Стек для вычислений:

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				

Алгоритм построения полиза «сортировочная станция»

Пример:

$$1 + 2$$

символ входной строки

1

+

2

конец строки

стек операторов

+				
+				

выходная строка

1				
1				
1	2			
1	2	+		

Алгоритм построения полиза «сортировочная станция»

Пример:

$$1 + 2 * 4$$

символ входной строки

1

+

2

*

4

конец строки

стек операторов

+				
+				
+	*			
+	*			

выходная строка

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

Алгоритм построения полиза «сортировочная станция»

Пример:

$$(1 + 2) * 4$$

символ входной строки

(
1
+
2
)
*
4

конец строки

стек операторов

(
(
(+			
(+			
*				
*				

выходная строка

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

Алгоритм построения полиза «сортировочная станция»

Пример:

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

Алгоритм построения полиза «сортировочная станция»

1. Читаем очередной символ.
2. Если символ является числом, добавляем его к выходной строке.
3. 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 «орег», то пока на вершине стека операция на вершине стека имеет приоритет больше (или равен для левоассоциативной операции), чем «орег», то выталкиваем верхний элемент стека в выходную строку, в конце помещаем «орег» в стек.
6. Когда входная строка закончилась, выталкиваем все символы из стека (операторы) в выходную строку.

Реализация

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

Реализация

```
1  enum OPERATOR {
2      PLUS, MINUS,
3      MULTIPLY,
4      LBRACKET, RBRACKET
5  };
6  int PRIORITY[] = {
7      0, 0,
8      1,
9      2, 2
10 };
11
12 class Oper: public Lexem {
13     OPERATOR opertype;
14 public:
15     Oper();
16     OPERATOR getType();
17     int getValue(const Number& left,
18                 const Number& right);
19 };
```

Реализация

```
1  #include <string>
2  #include <vector>
3
4  std::vector<Lexem> parseLexem(
5      std::string input
6  );
7  std::vector<Lexem> buildPoliz(
8      std::vector<Lexem> infix
9  );
10 int evaluatePoliz(
11     std::vector<Lexem> poliz
12 );
```

Реализация

```
1  int main() {
2      std::string codeline;
3      std::vector<Lexem *> infix;
4      std::vector<Lexem *> postfix;
5      int value;
6
7      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 }
```