Московский Энергетический Институт

(Технический университет)

**Лабораторная работа № 1.**

*Выполнил*: *ст. гр.* А-13-05 Петров С. А.

*Проверила*: *ст.преп*. Гречкина П.В.

24.02.2009.

Оглавление

[Постановка задачи. 3](#_Toc223261813)

[Описание предметной области. 3](#_Toc223261814)

[Обратная польская нотация и вычисления на стеке. 3](#_Toc223261815)

[Алгоритм преобразования выражения из инфиксной записи в обратную польскую. 3](#_Toc223261816)

[Алгоритм вычисления значения выражения, представленного в обратной польской записи. 4](#_Toc223261817)

[Описание табличной модели 5](#_Toc223261818)

[Пример работы. 8](#_Toc223261819)

[Приложение. 9](#_Toc223261820)

# Постановка задачи.

Разработать модель перевода инфиксной записи выражения в постфиксную, с последующим подсчётом значения выражения.

# Описание предметной области.

Далее представлены основные определения и используемые алгоритмы для реализации задачи.

## Обратная польская нотация и вычисления на стеке.

Говорят, что выражение записано *в инфиксной форме*, если знак операции (сложения, умножения, вычитания либо деления) стоит между своими аргументами, например, 5 + 7. Каждая операция имеет приоритет выполнения (сначала выполняются умножение и деление, затем сложение и вычитание). Для изменения приоритета выполнения операций используются круглые скобки.

Вычислять программно значение выражения, записанного в инфиксной форме, неудобно. Проще сначала перевести его в *постфиксную*, или *обратную польскую* запись, в которой знак операции записывается после своих операндов, например, 5 7 +.

## Алгоритм преобразования выражения из инфиксной записи в обратную польскую.

Для перевода выражения из инфиксной формы в постфиксную с учетом приоритетов операций и скобок существует простой алгоритм (Дейкстры). Алгоритм работает со стеком, в котором хранятся знаки операций. Сначала стек пуст. На вход алгоритму подается последовательность лексем (числа, скобки или знаки операций), представляющая некоторое арифметическое выражение, записанное в инфиксной форме. Результатом работы алгоритма является эквивалентное выражение в постфиксной форме. Вводятся приоритеты операций: открывающая скобка имеет приоритет 0, знаки + и – — приоритет 1 и знаки \* и / — приоритет 2.

1. Пока не достигнут конец входной последовательности, читать очередную лексему и выполнять с ней следующие операции:
   1. если прочитан операнд (число), записать его в выходную последовательность;
   2. если прочитана открывающая скобка, положить её в стек;
   3. если прочитана закрывающая скобка, вытолкнуть из стека в выходную последовательность всё до открывающей скобки, при этом сами скобки уничтожаются (удаляются из стека и в ответ не идут);
   4. если прочитан знак операции, вытолкнуть из стека в выходную последовательность все операции с большим либо равным приоритетом, а прочитанную операцию положить в стек.
2. Если достигнут конец входной последовательности, вытолкнуть всё из стека в выходную последовательность и завершить работу.

Порядок операндов в выходной последовательности не отличается от порядка операндов в исходной последовательности. В выходной последовательности отсутствуют скобки.

## Алгоритм вычисления значения выражения, представленного в обратной польской записи.

Для вычисления значения выражения, записанного в постфиксной форме, можно использовать описанный далее алгоритм. На вход подается последовательность лексем (числа или знаки операций), представляющая некоторое арифметическое выражение, записанное в постфиксной форме. Результатом работы алгоритма является значение этого выражения.

1. Пока не достигнут конец входной последовательности, читать очередную лексему и выполнять с ней следующие операции:
   1. если прочитан операнд (число), положить его в стек;
   2. если прочитан знак операции, вытолкнуть из стека два операнда и положить в стек результат применения прочитанной операции к этим операндам, взятым в обратном порядке.
2. Если достигнут конец входной последовательности, завершить работу. В стеке останется единственное число — значение исходного выражения.

Схематично работа такого алгоритма показана на рисунке:



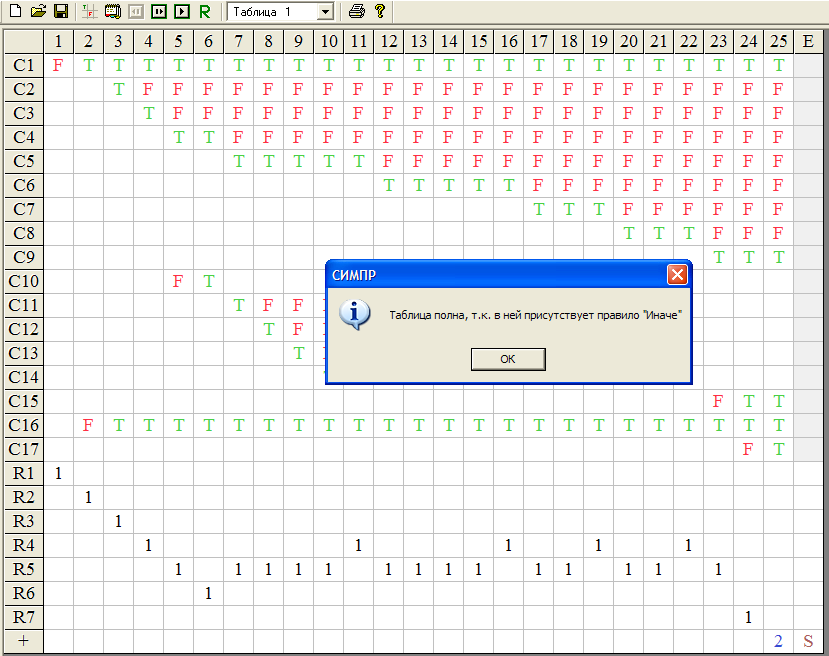
# Описание табличной модели

Модель включает в себя две таблички, содержащие правила соответствующих алгоритмов, первая табличка для перевода выражения в постфиксную запись, вторая для подсчёта выражения.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | E |
| На входе пусто? | T | F | F | F | F | F | F | F | F | F | F | F | F | F |  |
| На входе число? |  | T | F | F | F | F | F | F | F | F | F | F | F | F |  |
| На входе '+' ? |  |  | T | T | T | F | F | F | F | F | F | F | F | F |  |
| На входе '-' ? |  |  |  |  |  | T | T | T | F | F | F | F | F | F |  |
| На входе '\*' ? |  |  |  |  |  |  |  |  | T | T | T | F | F | F |  |
| На входе '/' ? |  |  |  |  |  |  |  |  |  |  |  | T | T | T |  |
| Стек пуст? |  |  | F | F |  | F | F |  | F | F |  | F | F |  |  |
| Операнд 2 есть? |  |  | F | T | T | F | T | T | F | T | T | F | T | T |  |
| Операнд 1 есть? |  |  |  | F | T |  | F | T |  | F | T |  | F | T |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Ждать |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Положить в стек |  | 1 |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Вытолкнуть из стека операнд 2 |  |  | 1 |  |  | 1 |  |  | 1 |  |  | 1 |  |  |  |
| Вытолкнуть из стека операнд 1 |  |  |  | 1 |  |  | 1 |  |  | 1 |  |  | 1 |  |  |
| Сложить |  |  |  |  | 1 |  |  | 1 |  |  | 1 |  |  | 1 |  |
| Отнять |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Перемножить |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Разделить |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Очистить операнды |  |  |  |  | 2 |  |  | 2 |  |  | 2 |  |  | 2 |  |
| Убрать символ со входа |  |  |  |  | 3 |  |  | 3 |  |  | 3 |  |  | 3 |  |
| Записать ответ | 1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | S |  |  |  |  |  |  |  |  |  |  |  |  |  | S |

**Таблица 2**

Ниже показана полнота и не противоречивость таблиц.



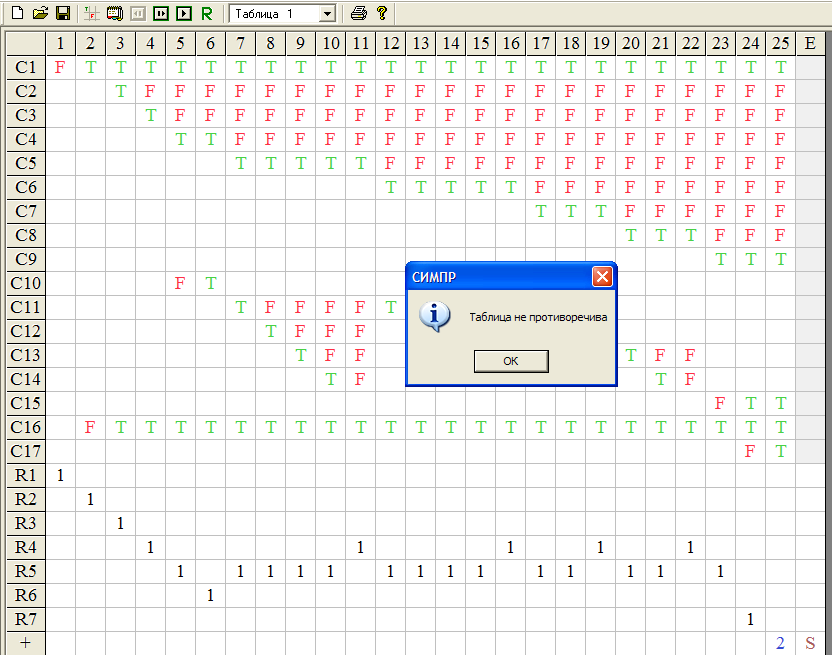


Таблица 1 не противоречива, и полна, так как присутствует правило иначе

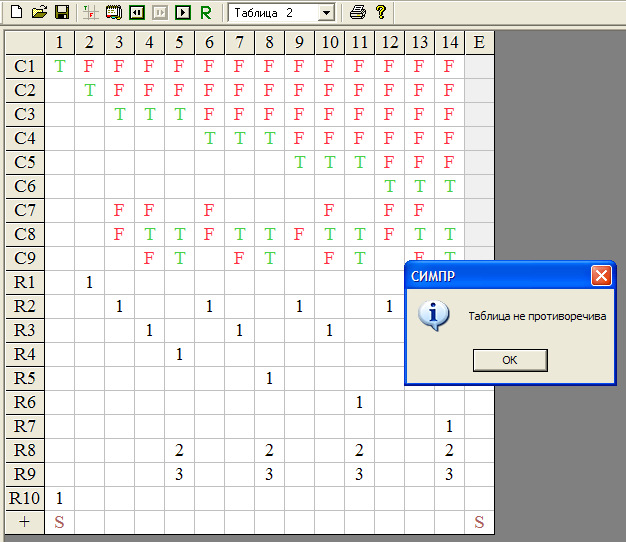
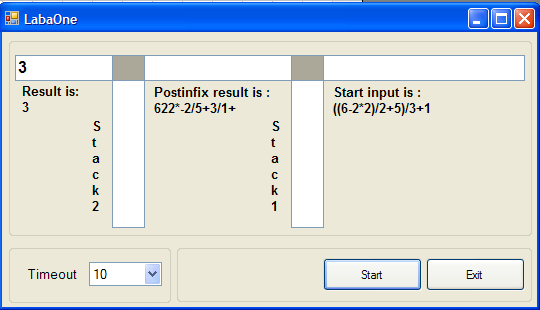


Таблица 2 не противоречива, и полна, так как присутствует правило иначе.

# Пример работы.



# Приложение.

protected override void WndProc(ref Message m)

{

int wparamhi,wparamlo,wparam;

int lParam = Convert.ToInt32("" + m.LParam);

if (m.Msg == simpr)

{

wparam = Convert.ToInt32("" + m.WParam);

wparamhi = wparam / 65536;

wparamlo = wparam - wparamhi \* 65536;

#region Условия

if (wparamhi == 0)

{

if (wparamlo == 1) // Таблица 1

{

switch (lParam)

{

case (1): m.Result = f.IsStarting(); break;

case (2): m.Result = f.IsInputNumeral(); break;

case (3): m.Result = f.IsInputOpenStaple();break;

case (4): m.Result = f.IsInputCloseStaple(); break;

case (5): m.Result = f.IsInputPlus(); break;

case (6): m.Result = f.IsInputMinus(); break;

case (7): m.Result = f.IsInputMultiply(); break;

case (8): m.Result = f.IsInputDiv(); break;

case (9): m.Result = f.IsInputEmpty(); break;

case (10): m.Result = f.IsStack1OpenStaple();break;

case (11): m.Result = f.IsStack1Plus(); break;

case (12): m.Result = f.IsStack1Minus(); break;

case (13): m.Result = f.IsStack1Multiply(); break;

case (14): m.Result = f.IsStack1Div(); break;

case (15): m.Result = f.IsStack1Empty(); break;

case (16): m.Result = f.IsSaveInput(); break;

case (17): m.Result = f.IsSaveRes1(); break;

}

}

else if (wparam == 2) // Таблица 2

{

switch (lParam)

{

case (1): m.Result = f.IsRes1Empty(); break;

case (2): m.Result = f.IsRes1Numeral(); break;

case (3): m.Result = f.IsRes1Plus(); break;

case (4): m.Result = f.IsRes1Minus(); break;

case (5): m.Result = f.IsRes1Multiply(); break;

case (6): m.Result = f.IsRes1Div(); break;

case (7): m.Result = f.IsStack2Empty(); break;

case (8): m.Result = f.IsOp2Exist(); break;

case (9): m.Result = f.IsOp1Exist(); break;

}

}

}

#endregion

#region Действия

else if (wparamhi == 1)

{

if (wparamlo == 1) // Таблица 1

{

switch (lParam)

{

case (1): f.Sleep(); break;

case (2): f.SaveInput(); break;

case (3): f.WriteInputToRes1(); break;

case (4): f.WriteInputToStack1(); break;

case (5): f.WriteStack1ToRes(); break;

case (6): f.DeleteStamp(); break;

case (7): f.SaveRes1(); break;

}

}

else if (wparamlo == 2) // Таблица 2

{

switch (lParam)

{

case (1): f.WriteRes1ToStack2(); break;

case (2): f.GetOp2(); break;

case (3): f.GetOp1(); break;

case (4): f.SumOp1Op2(); break;

case (5): f.MinusOp1Op2(); break;

case (6): f.MultiplyOp1Op2(); break;

case (7): f.DivOp1Op2(); break;

case (8): f.ClearOp1Op2(); break;

case (9): f.ClearLeftCharInRes1(); break;

case (10): f.WriteResult(); break;

}

}

m.Result = new IntPtr(1); // Ответом на запрос действия со стороны СИМПР должна быть единица

}

#endregion

Application.DoEvents();

}

else

{

base.WndProc(ref m);

}

}