**Лабораторна робота №5**

**Обернена польська нотація та обчислення математичних виразів**

Необхідно перевірити правильність введеного виразу та обчислити його значення. У виразі можуть бути дужки.

|  |  |  |  |
| --- | --- | --- | --- |
| **№** | **Операнди** | **Бінарні оператори** | **Унарні оператори та функції** |
| **1** |  | **+,-,\*,/,^** | **+,-** |
| 2 |  | \*,/,%,^ | +,- |
| 3 | та змінні | +,-,\* | +,-,правий ++, правий -- |
| 4 |  | +,-,\*,% | +,-,sin,cos, |
| 5 | та змінні | +,-,/ | +,-,exp |
| 6 | та змінні | +,- | +,-,ln |

* *Змінна – послідовність, що складається з латинських літер та цифр і починається з латинської літери. Якщо у виразі є змінні, то інтерпретатор повинен запросити у користувача їх значення.*
* *Кожен варіант дається двом студентам для здобуття навичок командної роботи*

**Приоритет операций**

* **1-я группа (наивысший приоритет)**. Вызов [функции](http://www.vedikhin.ru/2006/07/functions.html) и доступ к элементу [массива](http://www.vedikhin.ru/2006/07/variables.html) (например, a[7]).
* **2-я группа**. Операции с одним операндом:
  + логическое отрицание (!);
  + побитовое отрицание (~);
  + прибавление единицы (++);
  + вычитание единицы (--);
  + изменение знака на противоположный (-).
* **3-я группа.** [Побитовые операции](http://www.vedikhin.ru/2006/07/bit-operations.html):
  + побитовая операция И (&);
  + побитовая операция ИЛИ (|);
  + побитовая операция исключающее ИЛИ (^);
  + побитовый сдвиг вправо (>>);
  + побитовый сдвиг влево (<<).
* **4-я группа**. Ряд [арифметических операций](http://www.vedikhin.ru/2006/07/assignment-operator.html):
  + умножение (\*);
  + деление (/);
  + получение остатка от деления (%).
* **5-я группа**. Оставшиеся арифметические операции:
  + сложение (+);
  + вычитание (-).
* **6-я группа**. [Операции отношения](http://www.vedikhin.ru/2006/07/relation-operations.html):
  + больше (>);
  + больше или равно (>=);
  + меньше (<);
  + меньше или равно (<=);
  + равно (=);
  + не равно (!=).
* **7-я группа**. [Логическая операция И (&&)](http://www.vedikhin.ru/2006/07/bool-operations.html).
* **8-я группа**. [Логическая операция ИЛИ (||)](http://www.vedikhin.ru/2006/07/bool-operations.html).
* **9-я группа**. [Операция присваивания](http://www.vedikhin.ru/2006/07/assignment-operator.html) и [объединенные арифметические и побитовые операции с операцией присваивания](http://www.vedikhin.ru/2006/07/assignment-operator-2.html):
  + операция присваивания (=);
  + сложение с присваиванием (+=);
  + вычитание с присваиванием (-=);
  + умножение с присваиванием (\*=);
  + деление с присваиванием (/=);
  + остаток от деления с присваиванием (%=);
  + побитовая операция "И" с присваиванием (&=);
  + побитовая операция "ИЛИ" с присваиванием (|=);
  + побитовая операция исключающее "ИЛИ" с присваиванием (^=);
  + побитовый сдвиг влево с присваиванием (<<=);
  + побитовый сдвиг вправо с присваиванием (>>=).

**Вычисления на стеке**

**Общий порядок**

Автоматизация вычисления выражений в обратной польской нотации основана на использовании [стека](http://ru.wikipedia.org/wiki/%D0%A1%D1%82%D0%B5%D0%BA). Алгоритм вычисления для стековой машины элементарен:

1. Обработка входного символа
   * Если на вход подан операнд, он помещается на вершину стека.
   * Если на вход подан знак операции, то соответствующая операция выполняется над требуемым количеством значений, извлечённых из стека, взятых в порядке добавления. Результат выполненной операции кладётся на вершину стека.
2. Если входной набор символов обработан не полностью, перейти к шагу 1.
3. После полной обработки входного набора символов результат вычисления выражения лежит на вершине стека.

Реализация стековой машины, как программная, так и аппаратная, чрезвычайно проста и может быть очень эффективной. Обратная польская запись совершенно унифицирована — она принципиально одинаково записывает унарные, бинарные, тернарные и любые другие операции, а также обращения к функциям, что позволяет не усложнять конструкцию вычислительных устройств при расширении набора поддерживаемых операций. Это и послужило причиной использования обратной польской записи в некоторых научных и программируемых микрокалькуляторах.

**Пример вычисления выражений**

Выражение (1 + 2) \times 4 + 3 в ОПН может быть записано так: 1 2 + 4 × 3 +

Вычисление производится следующим образом (указано состояние стека после выполнения операции):

|  |  |  |
| --- | --- | --- |
| **Ввод** | **Операция** | **Стек** |
| 1 | поместить в стек | 1 |
| 2 | поместить в стек | 1, 2 |
| + | сложение | 3 |
| 4 | поместить в стек | 3, 4 |
| \* | умножение | 12 |
| 3 | поместить в стек | 12, 3 |
| + | сложение | 15 |

Результат, 15, в конце вычислений находится на вершине стека.

**Преобразование из инфиксной нотации**

[Эдсгер Дейкстра](http://ru.wikipedia.org/wiki/%D0%94%D0%B5%D0%B9%D0%BA%D1%81%D1%82%D1%80%D0%B0,_%D0%AD%D0%B4%D1%81%D0%B3%D0%B5%D1%80_%D0%92%D0%B0%D0%B9%D0%B1) изобрёл алгоритм для преобразования выражений из [инфиксной нотации](http://ru.wikipedia.org/w/index.php?title=%D0%98%D0%BD%D1%84%D0%B8%D0%BA%D1%81%D0%BD%D0%B0%D1%8F_%D0%BD%D0%BE%D1%82%D0%B0%D1%86%D0%B8%D1%8F&action=edit&redlink=1) в ОПН. Алгоритм получил название «сортировочная станция», за сходство его операций с происходящим на железнодорожных сортировочных станциях. Инфиксная нотация — это форма математических записей, которую использует большинство людей (например, 3 + 4 или 3 + 4 \* (2 - 1)). Как и алгоритм вычисления ОПН, [алгоритм сортировочной станции](http://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC_%D1%81%D0%BE%D1%80%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%BE%D1%87%D0%BD%D0%BE%D0%B9_%D1%81%D1%82%D0%B0%D0%BD%D1%86%D0%B8%D0%B8) основан на стеке. В преобразовании участвуют две текстовых переменных: входная и выходная строки. В процессе преобразования используется стек, хранящий ещё не добавленные к выходной строке операторы. Преобразующая программа читает входную строку последовательно символ за символом (символ — это не обязательно буква), выполняет на каждом шаге некоторые действия в зависимости от того, какой символ был прочитан.

**Простой пример**

Вход: 3 + 4

Добавим 3 к выходной строке (если прочитано число, то оно сразу добавляется к выходной строке).

Помещаем + (или его Идентификатор) в стек операторов.

Добавим 4 к выходной строке.

Мы прочитали всё выражение, теперь выталкиваем все оставшиеся в стеке операторы в выходную строку. В нашем примере в стеке содержится только +.

Выходная строка: 3 4 +

В данном примере проявляются некоторые правила: все числа переносятся в выходную строку сразу после прочтения; когда выражение прочитано полностью, все оставшиеся в стеке операторы выталкиваются в выходную строку.

**Алгоритм**

* Пока есть ещё символы для чтения:
* Читаем очередной символ.
* Если символ является числом, добавляем его к выходной строке.
* Если символ является символом функции, помещаем его в стек.
* Если символ является открывающей скобкой, помещаем его в стек.
* Если символ является закрывающей скобкой:

До тех пор, пока верхним элементом стека не станет открывающая скобка, выталкиваем элементы из стека в выходную строку. При этом открывающая скобка удаляется из стека, но в выходную строку не добавляется. Если после этого шага на вершине стека оказывается символ функции, выталкиваем его в выходную строку. Если стек закончился раньше, чем мы встретили открывающую скобку, это означает, что в выражении либо неверно поставлен разделитель, либо не согласованы скобки.

* Если символ является оператором ***о1***, тогда:

1) пока…

… (если оператор ***o1*** [право-ассоциированый](http://ru.wikipedia.org/wiki/%D0%90%D1%81%D1%81%D0%BE%D1%86%D0%B8%D0%B0%D1%82%D0%B8%D0%B2%D0%BD%D0%BE%D1%81%D1%82%D1%8C)) приоритет ***o1*** меньше приоритета оператора, находящегося на вершине стека…

… (если оператор ***o1*** [ассоциированный](http://ru.wikipedia.org/wiki/%D0%90%D1%81%D1%81%D0%BE%D1%86%D0%B8%D0%B0%D1%82%D0%B8%D0%B2%D0%BD%D0%BE%D1%81%D1%82%D1%8C), либо [лево-ассоциированный](http://ru.wikipedia.org/wiki/%D0%90%D1%81%D1%81%D0%BE%D1%86%D0%B8%D0%B0%D1%82%D0%B8%D0%B2%D0%BD%D0%BE%D1%81%D1%82%D1%8C)) приоритет ***o1*** меньше либо равен приоритету оператора, находящегося на вершине стека…

… выталкиваем верхние элементы стека в выходную строку;

2) помещаем оператор ***o1*** в стек.

* Когда входная строка закончилась, выталкиваем все символы из стека в выходную строку. В стеке должны были остаться только символы операторов; если это не так, значит в выражении не согласованы скобки.