Лекция 12. Вычисления на стеке, конкатенативное программирование

Коновалов А. В.

31 октября 2022 г.

Конкатенативное программирование

Конкатенативное программирование — парадигма программирования, в которой композиция функций выражается через конкатенацию строк. Примеры языков: FORTH, Joy, Factor.

Т.е. пусть у нас есть две программы P1 и P2, конкатенация этих двух программ P1 P2 будет выражать применение P2 к результату выполнения P1.

В конкатенативных языках явных переменных нет, данные передаются неявно, через стек. Программа представляет собой последовательность операторов, каждый из которых выполняет какую либо операцию со стеком. Частный случай: константы — это тоже операторы, которые на стек кладут соответствующее значение.

Конкатенативное программирование — работа со стеком

В конкатенативных языках программирования принято записывать стек, растущий слева направо, причём верхушка стека расположена справа.

Действия операторов принято записывать как

```
оператор : ... стек до  =>  ... стек после
```

Например:

```
+ : ... x y  =>  ... (x+y)
```

Обратная польская запись (1)

Программа пишется в обратной польской записи или постфиксной записи: сначала записываются операнды, а потом сама операция.

Местность (арность) операции, функции — количество операндов (аргументов) у неё.

Коместность (коарность) — количество возвращаемых значений.

Поскольку местность каждой операции фиксирована, скобки не нужны.

Местность констант 0 (не принимают аргументов), коместность — 1.

```
константа : ...  =>  ... константа
```

Обратная польская запись (2) — пример

Выражение в обычной (инфиксной записи):

```
(2 - 1) * (3 + 4)
```

Выражение в постфиксной записи (обратной польской):

```
2 1 - 3 4 + *
```

- Аргументы операции `-`: `2` и `1`,
- аргументы операции `+`: `3` и `4`,
- аргументы `*`: `2 1 -`, `3 4 +`.

Можно добавить скобки для наглядности, но их никто не ставит:

```
((2 1 -) (3 4 +) *)
```

Обратная польская запись (3) — реализация

Обратная польская запись допускает простую и эффективную реализацию:

- в цикле читаем очередную операцию,
- снимаем со стека соответствующее количество аргументов операции,
- выполняем операцию,
- кладём на стек её результаты.

Обратная польская запись (4) — реализация, пример

```
Стек                    Программа

...                   [2] 1  -  3  4  +  *
...  2                 2 [1] -  3  4  +  *
...  2 1               2  1 [-] 3  4  +  *
...  1                 2  1  - [3] 4  +  *
...  1 3               2  1  -  3 [4] +  *
...  1 3 4             2  1  -  3  4 [+] *
...  1 7               2  1  -  3  4  + [*]
...  7                 2  1  -  3  4  +  *
программа завершилась
```

Знакомство с языком FORTH

Рекомендуемая литература по FORTH

- Баранов С. Н., Ноздрунов Н. Р. Язык Форт и его реализации. — Л.: Машиностроение. Ленингр. отдние, 1988, — 157 с. ил. (ЭВМ в производстве)
- Лео Броуди. Способ мышления — ФОРТ. Язык и философия для решения задач.

Определения функций в Форте

Функции в FORTH принято называть **статьями,** хранилище функций — **словарём.**

Программа на языке FORTH состоит из последовательности **слов,** словом может быть или целочисленная константа, или некоторое имя. Часть слов предопределены (встроены в язык), часть определяются пользователем в виде статей:

Определение статьи выглядит так

```
: ИМЯ  слова…  ;
```

Знак `:` начинает определение, знак `;` — заканчивает.

Семантика вызовов функций в Форте

Для вызовов функций вводится второй стек — **стек возвратов.** В основном стеке, **стеке данных** находятся значения, которыми обмениваются операции, в классическом FORTH'е это целые числа. В стеке возвратов хранятся адреса команд в словарных статьях.

Интерпретатор работает в следующем цикле:

- Если в статье слова не кончились, читается очередное слово.
  - Если слово есть в словаре, адрес следующего слова кладётся на стек возвратов, управление передаётся на первое слово словарной статьи.
  - Если нет в словаре и слово является записью целого числа, то число кладётся на стек данных.
  - Если слова нет в словаре и оно не является записью числа — ОШИБКА.
- Если слова в статье кончились — со стека возвратов снимается адрес следующего слова и передаётся на него управление.

Некоторые встроенные слова FORTH (1)

- Арифметика: `+`, `-`, `*`, `/`.
- Слова работы со стеком:
  - `DUP : ... x ⇒ ... x x` — дублирует верхушку стека
  - `DROP : ... x ⇒ ...` — удаляет слово с верхушки стека
  - `SWAP : ... x y ⇒ ... y x` — обменивает местами два слова на верхушке
  - `ROT : ... x y z ⇒ ... y z x` — поднимает на верхушку третий по счёту элемент
  - `OVER : ... x y ⇒ ... x y x` — копирует подвершину на верхушку

Некоторые встроенные слова FORTH (2)

- Управляющие конструкции
  - `IF ... THEN` — если на вершине не ноль, выполняются слова между `IF` и `THEN`, иначе ничего не делается. В обоих случаях число со стека снимается.
  - `IF ... ELSE ... THEN` — если на верхушке не ноль, он снимается с верхушки и выполняются слова между `IF` и `ELSE`, иначе ноль снимается с верхушки и выполняются слова между `ELSE` и `THEN`.
  - `WHILE ... WEND` — цикл с предусловием повторяется до тех пор, пока на верхушке не ноль.
- Ввод-вывод
  - `. : ... x ⇒ ...` (точка) — печатает число, снимая его со стека

Примеры программ на FORTH (1)

Функция (слово) `hypot` вычисляет гипотенузу прямоугольного треугольника:

```
\ square : ... x  =>  ... x²
: square DUP * ;

\ hypot : ... x y  =>  ... √(x²+y²)
: hypot square SWAP square + SQRT ;
```

Примеры программ на FORTH (2)

Выполнение слова `square`:

```
Стек                  Программа
-------------------------------
... x               [DUP] *  ;
... x x              DUP [*] ;
... x²               DUP  * [;]

происходит возврат из square
```

Примеры программ на FORTH (3)

Выполнение слова hypot:

```
Стек                    Программа
-------------------------------
 ...  x y            [square] SWAP  square  +  SQRT  ;
```

Когда слово square вызывается, на стек возвратов кладётся указатель на слово SWAP в определении слова hypot.

```
Стек                    Программа
-------------------------------
 ...  x y²           square [SWAP] square  +  SQRT  ;
 ...  y² x           square  SWAP [square] +  SQRT  ;
 ...  y² x²          square  SWAP  square [+] SQRT  ;
 ...  y²+x²          square  SWAP  square  + [SQRT] ;
 ...  √(y²+x²)       square  SWAP  square  +  SQRT [;]
```

происходит возврат из hypot

Примеры программ на FORTH (4)

Пример, характерный для FORTH:

```
: 2 3 ;
2 2 * .
```

Она выведет 9, а не 4.

```
: + - ;
10 5 + .
```

Выведет 5, а не 15.

Примеры программ на FORTH (4)

Слово с переменным числом параметров:

```
: SUM DUP WHILE + SWAP DUP WEND DROP ;
```

Сложит все числа на стеке до ближайшего нуля.

```
1 2 3 0 4 5 6 SUM    =>  1 2 3 15
═══════ ~~~~~~~~          ═══════ ~~
```

```
1 2 3 0 4 5 6        [DUP] WHILE  +  SWAP  DUP  WEND  DROP  ;
1 2 3 0 4 5 6 6       DUP [WHILE] +  SWAP  DUP  WEND  DROP  ;
1 2 3 0 4 5 6         DUP  WHILE [+] SWAP  DUP  WEND  DROP  ;
1 2 3 0 4 11          DUP  WHILE  + [SWAP] DUP  WEND  DROP  ;
1 2 3 0 11 4          DUP  WHILE  +  SWAP [DUP] WEND  DROP  ;
1 2 3 0 11 4 4        DUP  WHILE  +  SWAP  DUP [WEND] DROP  ;
```

Примеры программ на FORTH (5)

```
1 2 3 0 11 4 4        DUP [WHILE] + SWAP DUP WEND DROP ;
1 2 3 0 11 4          DUP WHILE [+] SWAP DUP WEND DROP ;
1 2 3 0 15            DUP WHILE + [SWAP] DUP WEND DROP ;
1 2 3 15 0            DUP WHILE + SWAP [DUP] WEND DROP ;
1 2 3 15 0 0          DUP WHILE + SWAP DUP [WEND] DROP ;
1 2 3 15 0 0          DUP [WHILE] + SWAP DUP WEND DROP ;
1 2 3 15 0            DUP WHILE + SWAP DUP WEND [DROP] ;
1 2 3 15              DUP WHILE + SWAP DUP WEND DROP [;]
```