Дмитриева Марина Валерьевна
Поздняков Сергей Николаевич

ЗАНЯТИЕ 8. ФОРМУЛЫ, ФОРМУЛЫ, ФОРМУЛЫ...

1. СКОБОЧНАЯ ЗАПИСЬ

Давайте задумаемся о том, как компьютер хранит и обрабатывает привычные нам формулы. Ниже приведены несколько формул, перечень которых каждый из вас без труда расширит.

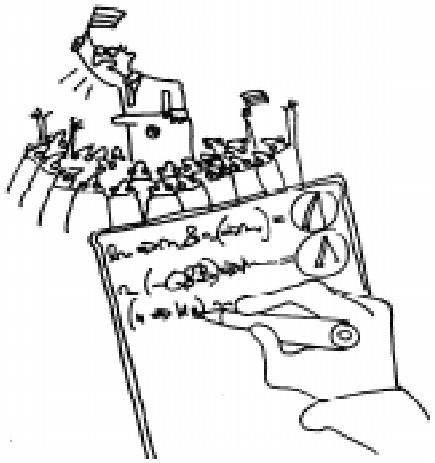
а) $m = 2^{2^n}$

б) $a = a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{a_3}}}$

в) $y = x\sin x^2 + \sqrt[3]{x} - 3\log_2 x$

Формула чем-то напоминает иероглиф: от того, выше или ниже записан значок, короче или длиннее проведена линия, зависит порой смысл всей формулы. Формулу всегда стараются записать красиво. Попробуем «мысленно» ввести записанные формулы в компьютер. Как нам сразу начинает мешать их «красивая запись»! Ведь ввод с клавиатуры «многоэтажной» формулы предполагает выстраивание всех символов в очередь.

Запишем формулу по другому – в строку. Такую запись используют во всех алгоритмических языках. И тут не обойтись без скобок:



Ввод «многоэтажной» формулы...
Формулу всегда стараются записать красиво...

а) m=2^(2^n)
б) a=a0+1/(a1+1/(a2+1/a3))
в) y=x*sin(x^2)+x^(1/3)–3*(ln(x)/ln(2))

Запись стала выглядеть куда менее привлекательно, зато теперь понятно, как ее вводить в компьютер: посимвольно, слева-направо.

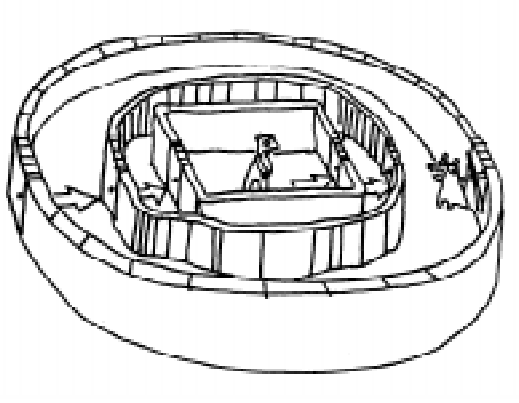
При записи формул в строку используют дополнительные договоренности, позволяющие уменьшить число скобок. Так, сочетательный закон позволяет, вместо скобочных записей (a+b)+c и a+(b+c), писать просто a+b+c. То же верно для умножения. А вот для вычитания и деления сочетательный закон не выполняется: (a–b)–c≠a–(b–c). Конечно, скобки в этом случае можно «раскрыть», но тогда получим, вообще говоря, другую формулу.

Другим способом уменьшения количества скобок в записи является использование приоритетов: операциям умножения и деления присваивается больший приоритет по отношению к операциям сложения и вычитания. Это означает, что при отсутствии скобок в записи сначала выполняются операции большего приоритета (умножение и деление), а затем меньшего (сложение и вычитание). Например, в следующей формуле скобки в правой части опускаются a*(b+c)=a*b+a*c.

Наконец, если операции одного приоритета выполняются в порядке их появления в строке, то скобки тоже можно опустить. Например, формула ((a+b)–c)–d записывается без скобок a+b–c–d.

Разберем одну из задач, связанную с анализом скобочной записи формулы.

Рассмотрим формулы, в которых разрешено использовать переменные, константы, арифметические операции сложения, вычитания, умножения и деления и три типа скобок: фигурные, квадратные и круглые. Требуется проверить, имеет ли



...неправильная скобочная структура, потому что скобки в формуле несогласованны...

формула правильную скобочную структуру. Будем говорить, что формула имеет правильную скобочную структуру, если выполняются следующие условия:

1. При просмотре формулы слева направо в любой момент времени число просмотренных закрывающих скобок меньше или равно числу просмотренных открывающих скобок.

2. В формуле число открывающих скобок равно числу закрывающих.

3. Скобки трех типов согласованы, то есть открывающей круглой скобке «(» должна соответствовать закрывающая круглая скобка «)», открывающей квадратной скобке «[» должна соответствовать закрывающая квадратная скобка «]», и, наконец, фигурной скобке «{» – фигурная скобка «}».

Формула [A+B*(C+D)]*{(A–B/C)/K+L} имеет правильную скобочную структуру. Формула (A+B))*(C+D) имеет неправильную скобочную структуру: нарушено первое условие. Следующая формула ((A+B) имеет также неправильную скобочную структуру: нарушено второе условие. В следующей формуле [A+B)(C+D] неправильная скобочная структура, потому что скобки в формуле несогласованны.

Задача 1.

Уровень 1. При записи формулы ученик постоянно путает круглые, квадратные и фигурные скобки. При этом он не путает открывающие и закрывающие скобки. Предложите алгоритм исправления ошибок ученика. Используйте для этого алгоритм анализа скобочной структуры, приведенный в указании. Проиллюстрируйте работу построенного алгоритма на следующей формуле: ((({a/(b*[c–d)}]+e]/(f–g}}].

Как модифицировать этот алгоритм, чтобы в исправленной формуле вложенные скобки чередовались (круглые – квадратные – фигурные – круглые).

Уровень 2. Напишите программу, которая, получив в качестве исходных данных формулу, определяет, имеет ли она правильную скобочную структуру.

Указание. Для построения алгоритма анализа скобочной структуры формулы полезно использовать стек (см. Занятие № 4, «Компьютерные инструменты в образовании» № 3-4, 1999 г.). Тогда алгоритм решения задачи может быть следующим. Просматриваем символ за символом, причем, встретив открывающую скобку, помещаем ее в стек, а встретив закрывающую скобку, сравниваем ее со скобкой на вершине стека. Если там оказывается открывающая скобка соответствующего типа, то верхняя скобка из стека удаляется, иначе скобочная запись является неправильной. Для того чтобы обойти ситуацию, связанную с исчерпанием стека, если в исходной формуле закрывающих скобок оказывается больше, чем открывающих, поместим в стек так называемый маркер (символ, который не может встретиться в формуле). Символы, не являющиеся скобками, будем просто игнорировать.

Формат ввода:

Формула, записанная строкой символов.

Формат вывода:

Результат анализа: «правильная запись скобок» или «неправильная запись скобок».

2. БЕССКОБОЧНАЯ ЗАПИСЬ

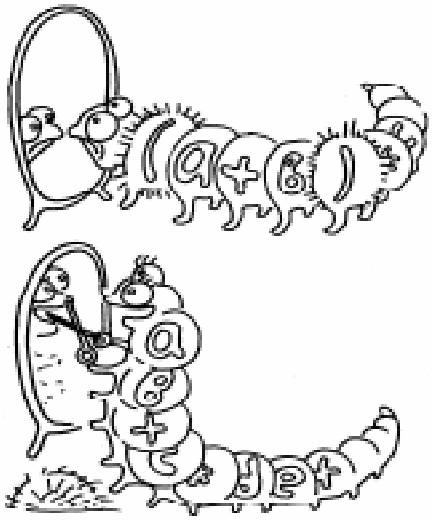
Попробуем представить, как вычисляется значение формулы, записанной строчкой символов.

Для этого надо найти действие, которое может быть выполнено первым, выделить операнды, выполнить это действие, а затем повторить операции с «укороченной» строкой, в которой вычисленная формула обозначена одним символом.

При определении порядка вычислений нужно уметь сравнивать приоритеты

операций, находить выражения, заключенные в скобки. Все это не так просто.

Поставим задачу шире: как автоматизировать работу с формулами. Мы имеем «внешнее» для компьютера представление, неудобное для автоматической обработки, хотим же перейти к другому «внутреннему» представлению, для которого автоматическая обработка формул будет обеспечиваться простыми, но эффективными алгоритмами.



Одним из преимуществ полученной формулы является отсутствие скобок.

Оказывается, можно придумать другие, более удобные для вычислений, способы задания формулы строкой. Наиболее известные и употребимые – префиксная и постфиксная записи. Как же они определяются?

Для простых формул типа a+b постфиксная запись выглядит как ab+, а префиксная +ab, то есть мы просто перемещаем знак операции (который по традиции ставится между операндами, и поэтому обычную запись называют еще инфиксной) после перечисления операндов или перед ними.

Обратим внимание, что для операций с одним аргументом, таких как возведение в степень или извлечение корня, мы обычно употребляем постфиксную или префиксную записи:

x^2 – действие записано после аргумента.

$\sqrt{x}$ – действие записано перед аргументом.

Как же строить префиксную и постфиксную записи формулы, которая содержит большее количество операций, например, (a+b*c)/(d–c)?

Воспользуемся общим приемом: сведем задачу к уже решенной. Для этого выделим последнюю выполняемую операцию и перенесем ее в конец (для постфиксной) или в начало (для префиксной) записи: [a+b*c][d–c]/, где в квадратных скобках стоят необработанные части формулы. Далее применим этот алгоритм к формулам в скобках.

Будем так действовать, пока не останется ни одной необработанной части формулы: abc*+dc–/. Последняя строка и есть постфиксная запись исходной формулы.

Задача 2.

Уровень 1.

а) преобразуйте формулу из задачи 1 в постфиксную и префиксную форму;

б) пусть формула наряду со знаками арифметических операций содержит операции возведения в квадрат и извлечения квадратного корня. Например: $(a*b*c)/\sqrt{((a-b)^2+(b-c)^2+(a-c)^2)}$.

Сформулируйте алгоритмы построения постфиксной и префиксной записей таких формул. Примените построенные алгоритмы к записанной выше формуле.

Одним из преимуществ полученной формулы является отсутствие скобок. Однако, возникает вопрос, можно ли по этой записи однозначно восстановить исходную формулу?

Удивительно, но по этим бесскобочным записям формула однозначно восстанавливается.

Опишем алгоритм, осуществляющий реконструкцию формулы по ее постфиксной записи и, тем самым, докажем сформулированное утверждение.

Первому символу постфиксной записи, который всегда является буквой для правильных записей, сопоставляем 1. Каждому следующему – число на 1 большее предыдущего, если это буква, и число на 1 меньшее, если это знак операции.

Так, для нашего примера получим такую запись:

a b c * + d e – /

1 2 3 2 1 2 3 2 1

Признаком того, что запись формулы правильная, является последняя единица. То, что это условие необходимо, понятно. Ведь в любой правильной фор-

муле число букв на 1 больше числа операций. Но является ли это условие достаточным? Иными словами, будет ли любой последовательности чисел, начинающейся и заканчивающейся единицей и обладающей свойством, что любые два соседних числа отличаются ровно на единицу, соответствовать некоторая формула?

Оказывается, да! Например, другая последовательность, состоящая также из 9 чисел: 121234321, – соответствует, в частности, такой формуле:

ab*cde+–/ или $\dfrac{a * b}{c - (d + e)}$

Задача 3.

Уровень 1.

а) Перечислите все «правильные» последовательности из 9 чисел, то есть последовательности натуральных чисел, начинающиеся и заканчивающиеся единицей, у которых соседние числа отличаются на единицу. Постройте по ним формулы, используя буквы a, b, c, d, e и знаки операций *, +, –, / в указанном порядке.

б) Может ли последовательность с указанными выше свойствами состоять из четного количества чисел?

в)* Придумайте способ, как подсчитывать количество «правильных» последовательностей, зная количества более коротких последовательностей. Найдите количество «правильных» последовательностей из 11 чисел.

Уровень 2. В произведении n чисел a_1, a_2, a_3, ..., a_n требуется расставить скобки, полностью определяющие порядок вычисления произведения, всеми возможными способами. Напишите программу, выполняющую эту работу. В качестве проверяемого результата подсчитайте количество вариантов, начинающихся с двух открывающих скобок.

Формат ввода:

Число букв в последовательности.

Формат вывода:

Число вариантов.

Пример.

Ввод:

4

Вывод:

2

..ТО запись формулы правильная и можно переходить к следующему шагу...

Пояснение. Последовательность abcd допускает следующие расстановки скобок: (((ab)c)d), ((ab)(cd)), ((a(bc))d), (a((bc)d)), (a(b(cd))).

Различные расстановки скобок в инфиксной записи формулы соответствуют различным «правильным» последовательностям, связанным с постфиксными записями формул. Поэтому «правильных» последовательностей из 2n–1 чисел столько же, сколько вариантов расстановки скобок в формуле с n переменными и n–1 операциями. Число этих способов носит название чисел Каталана. Обозначая их за k_n получаем: $k_1=1$, $k_2=1$, $k_3=2$.

Каждое следующее число Каталана выражается через предыдущие по формуле: $k_{n+1}=k_1k_n+k_2k_{n-1}+...+k_nk_1$

Алгоритм восстановления формулы по постфиксной записи:

ШАГ 1.

Сделаем нумерацию символов строки по правилам, перечисленным выше.

ЕСЛИ нумерация заканчивается единицей,

ТО запись формулы правильная и можно переходить к следующему шагу,

ИНАЧЕ в постфиксной записи формулы имеется ошибка.

ШАГ 2.

Найдем предпоследнюю единицу в записи и разделим формулу на три части:

1-ая: все символы от первого до того, который помечен предпоследней единицей включительно;

2-ая: все символы от помеченного предпоследней единицей до симво-

ла, помеченного последней единицей, не включая их;

3-я: символ, помеченный последней единицей.

Тогда

первая часть записи - первый операнд,

вторая часть - второй операнд,

третья часть - знак последней выполняемой операции.

Далее алгоритм нахождения последней выполняемой операции применяется к каждому из операндов.

Проиллюстрируем работу алгоритма на примере.

a b * c d e + – /
1 2 1 2 3 4 3 2 1

[a b *] / [c d e + –]
1 2 1 1 2 3 2 1

(a * b) / (c – [d e +])
1 2 1

((a * b) / (c – (d + e)))

В предложенном алгоритме не указывалось, как записывать результат разбора постфиксной записи. В примере мы использовали привычную инфиксную скобочную запись. Обратите внимание, что каждую возникающую инфиксную запись мы брали в скобки. Разумеется, какие-то из них могли быть лишними (в нашем случае лишней является пара внешних скобок). Алгоритм, который мы использовали в примере, порождает формулу в инфиксной записи, в которой расставлены все скобки.

3. ВЫЧИСЛЕНИЕ ЗНАЧЕНИЙ

У постфиксной записи есть замечательное свойство – по ней очень легко производить вычисления значений формулы. Этот факт применяется в некоторых моделях калькуляторов.

Для того чтобы описать алгоритм, воспользуемся структурой стека.

Предположим, что формула состоит только из однобуквенных переменных (a, b, c, ..., z) и знаков операций (*, +, –, /), а на вход алгоритма подается его правильная постфиксная запись. Алгоритм вычисления формулы в постфиксной форме с использованием стека чрезвычайно прост:

ПОКА в строке не кончатся символы,
ПОВТОРЯТЬ

Читаем очередной символ строки

ЕСЛИ очередной символ - буква,

ТО записываем в стек значение переменной, заданное этой буквой.

ЕСЛИ очередной символ - знак операции,

ТО извлекаем последовательно два (верхних) числа из стека и выполняем над ними операцию, считая самое верхнее число значением второго операнда; результат операции заносим в стек.

КОНЕЦ ЦИКЛА

Единственное оставшееся в стеке число и будет результатом вычисления формулы.

Докажем корректность этого алгоритма по индукции.

База индукции. При n=1 правильная формула может состоять только из одной буквы, например, a.

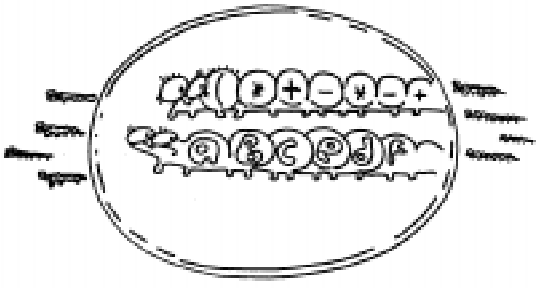
Тогда, в соответствии с алгоритмом, значение a будет записано в стек, и алгоритм закончит работу. Единственное число в стеке – значение a и будет значением формулы.

Для n=2 не существует правильной записи формулы.

Для n=3 правильная запись формулы в постфиксной форме будет выглядеть, например, так: ab– (разумеется, имена переменных и знак операции могут быть другими).

Нетрудно видеть, что работа по алгоритму приведет к следующей последовательности состояний стека:

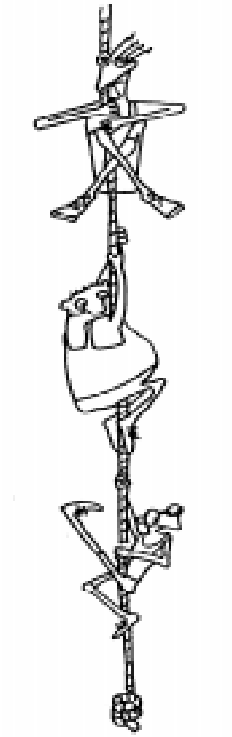
Значение a	Значение b	Значение a–b
	Значение a	



...алгоритм вычисления формулы в ... с использованием стека чрезвычайно прост...

Индукционный переход. По доказанной выше теореме постфиксная формула разбивается на части: первый операнд, второй операнд, знак операции. Докажем, что тогда значение этой формулы будет вычислено правильно.

Положим, что правильная постфиксная формула состоит из n символов и что для постфиксных записей с меньшим числом символов теорема доказана. Тогда после ввода всех символов первой части в стеке по индукционному предположению получится значение первого операнда. После ввода начального символа второго операнда значение первого операнда сдвигается на 1 вниз и не участвует в вычислении значений второго операнда. Таким образом, после окончательного ввода символов второго операнда стек будет выглядеть так:



...значение первого операнда сдвигается ... вниз...

Значение 2-ого операнда
Значение 1-ого операнда

Задача свелась к случаю n=3, и ввод знака операции приведет к окончательному вычислению значения формулы.

Корректность алгоритма доказана.

Пример.

Проиллюстрируем вычисление значения формулы в постфиксной записи ab*cde+–/ при a=2, b=6, c=9, d=1, e=5

Входная строка	a	b	*	c	d	e	+	–	/
Стек	2	6 2	12	9 12	1 9 12	5 1 9 12	6 9 12	3 12	4
Количество чисел в стеке	1	2	1	2	3	4	3	2	1

Теперь, когда мы узнали алгоритм вычисления значений формулы в постфиксной записи, можно объяснить смысл последовательности чисел, сопоставленных постфиксной записи при доказательстве корректности алгоритма: они показывают количество чисел, находящихся в стеке в процессе вычисления значения формулы.

Задача 4.

Уровень 1.

Проиллюстрируйте вычисление значения формулы

$$a_0 + \cfrac{b_1}{a_1 + \cfrac{b_2}{a_2 + \cfrac{b_3}{a_3}}} \quad \text{при} \quad a_0=1,$$

$a_1=2$, $a_2=3$, $a_3=6$, $b_1=7$, $b_2=5$, $b_3=2$.

Какого размера стек необходим для вычисления этой формулы. Используя переместительный закон сложения, перепишите формулу так, чтобы для его вычисления мог использоваться стек на 2 позиции короче.

Уровень 2. Пусть на входе алгоритма находится постфиксная запись, состоящая из букв и знаков арифметических действий +, –, /, *. Напишите программу, преобразующую произвольную постфиксную запись в такую, которая для вычисления своих значений требует стек минимальной длины.

Формат ввода:

Последовательность малых латинских букв и знаков арифметических операций, представляющая правильную постфиксную запись.

Формат вывода:

Преобразованная постфиксная запись.

Пример.

Ввод:

abc++

Вывод:

ab+c+

Идея алгоритма. Вычислим наполняемость стека для каждого из операндов и, если у второго операнда она больше и операция коммутативна, то операнды меняем местами.

Пример.

Задача 5.

Уровень 1. Придумайте алгоритм для вычисления значений формул, содержащих наряду с арифметическими операциями, операцию извлечения квадратного корня.

Проиллюстрируйте работу алгоритма при вычислении формулы:

$$\sqrt{\sqrt{a_0} * (a_1 - \sqrt{\sqrt{a_2}})}, \; a_0=9, \; a_1=5, \; a_2=16$$

Уровень 2. Напишите программу, реализующую алгоритм вычисления арифметических примеров с квадратным корнем в постфиксной записи, считая, что пример состоит из цифр, квадратных корней и арифметических операций и что вычисления не приведут к невыполнимым операциям типа деления на ноль или извлечения корня из отрицательного числа.

Формат ввода:

Правильная постфиксная запись. Вместо знака √ используйте \.

Формат вывода:

Числовой результат.

Пример: $\sqrt{\frac{\sqrt{6} * \sqrt{6}}{1+2} * 2}$

Ввод:

6\6*12+/2*\

Вывод:

2

Задача 6.

Уровень 1. Придумайте алгоритм для вычисления значений формул в префиксной записи.

а) для просмотра префиксной записи, начиная с конца (справа налево).

б) для естественного просмотра префиксной записи (слева направо).

Уровень 2. Напишите программу, реализующую алгоритм для вычисления значения формулы в префиксной записи.

Формат ввода:

Префиксная запись.

Значения всех аргументов через пробел в порядке их появления в строке ввода.

Формат вывода:

Значение формулы.

Пример: a-b*c, a=7, b=2, c=3.

Ввод:

–a*bc

7 2 3

Вывод:

1

4. ПЕРЕВОД В БЕССКОБОЧНУЮ ЗАПИСЬ

После того, как был рассмотрен алгоритм эффективного вычисления значений формулы в постфиксной записи, естественным стал вопрос об алгоритме перевода обычной скобочной записи в бесскобочную постфиксную.

Использование стека позволяет написать простой и эффективный алгоритм. По-прежнему будем считать, что инфиксная запись состоит из скобок, букв и знаков бинарных операций.

```
ПОКА во входной строке есть символы,
ПОВТОРЯТЬ
  Ввести очередной символ
  ЕСЛИ текущий символ - буква,
  ТО передать его в выходную строку.
  ЕСЛИ текущий символ - «(»,
  ТО записать ее в стек.
  ЕСЛИ текущий символ - «)»,
  ТО извлечь поочередно все символы
  из стека в выходную строку до первой открывающей скобки, которую убрать из стека.
  ЕСЛИ текущий символ - знак операции,
  ТО
    ПОКА приоритет операции на вершине стека не меньше приоритета текущей операции,
    ПОВТОРЯТЬ
    извлечь знак операции из стека в выходную строку.
    КОНЕЦ ЦИКЛА
    Текущий знак операции занести в стек.
```

КОНЕЦ ЦИКЛА

Извлечь из стека в выходную строку поочередно все оставшиеся в нем символы.

Пример.

Рассмотрим обработку строки a*b/(c–(d+e)).

Входная строка	a	*	b	/	(	c	–	(	d	+	e	)	)	
стек		*	*	/	(/	(/	– (/	(– (/	(– (/	+ (– (/	+ (– (/	– (/	/	
выходная строка	a		b	*		c			d		e	+	–	/

Задача 7.

Уровень 1. Предложите алгоритм перевода в постфиксную форму скобочной записи арифметической формулы с корнями. Проиллюстрируйте работу алгоритма на примере:

$$((\sqrt{a} + \sqrt{(b + c)}) / (e - \sqrt{(f * (k - m))}))$$

Уровень 2. Напишите программу перевода скобочной записи с корнями в постфиксную форму.

Формат ввода:

Скобочная запись.

Формат вывода:

Постфиксная запись.

...ЕСЛИ текущий символ – буква ТО передать его в выходную строку...

Этот процесс удобно изобразить с помощью ... бинарного дерева...

Пример.

Ввод:

\(a+b*\\(c–d))

Вывод:

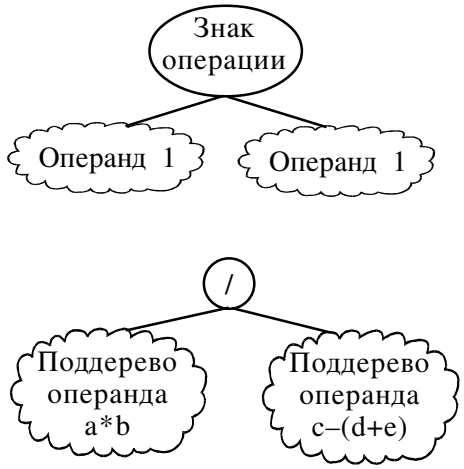
abcd–*+\

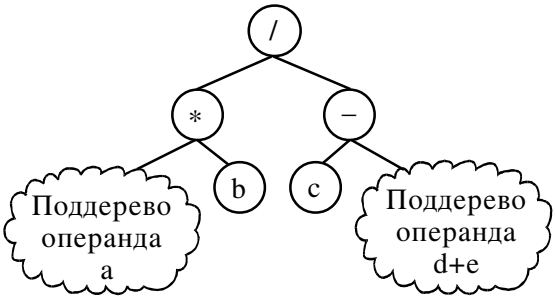
5. ПРЕДСТАВЛЕНИЕ ФОРМУЛ БИНАРНЫМИ ДЕРЕВЬЯМИ

Делая разбор арифметических формул, мы рекурсивно применяли процедуру нахождения последней операции и ее операндов:

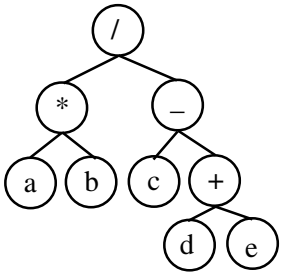
операнд 1	знак операции	операнд 2

Этот процесс удобно изобразить с помощью построения дерева.





Таким образом формулу $\frac{a * b}{c - (d + e)}$ можно представить бинарным деревом:



А построенное дерево можно задать следующей таблицей:

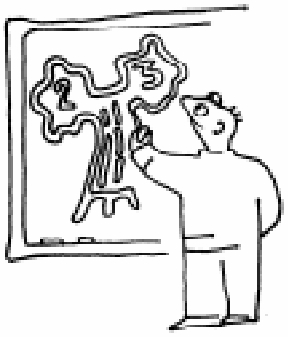
1	/	2	3
2	*	4	5
3	–	6	7
4	a	–	–
5	b	–	–
6	c	–	–
7	+	8	9
8	d	–	–
9	e	–	–

В таблице первое поле каждой записи содержит знак операции или имя переменной, а остальные два поля – ссылки на поддеревья, связанные с данной вершиной.

Э. Дейкстра предложил алгоритм, в котором дерево операций строится за один просмотр формулы. При работе алгоритма используются два стека: стек операций и стек операндов. В стек операций помещаютя операции и открывающие круглые скобки.

В стек операндов помещаются ссылки на уже сформированные узлы дерева, то есть ссылки на операнды. Опишем работу алгоритма.

Просматриваем формулу слева направо.
НАЧАЛО ЦИКЛА
ЕСЛИ текущий символ - буква,
ТО создается узел с информационным полем, равным ее значению, и ссылочными полями, содержащими пустую ссылку. Ссылка на построенный узел помещается в стек операндов.
ЕСЛИ текущий символ - «(»,
ТО она заносится в стек операций.
ЕСЛИ текущий символ - знак операции,
ТО (дальнейшие действия зависят от содержимого стека операций):
ЕСЛИ стек операций не пуст,
ТО (приоритет анализируемой операции сравнивается с приоритетом операций на вершине стека операций):
ЕСЛИ приоритет анализируемой операции больше,
ТО операция заносится в стек (приоритет открывающей скобки считается самым маленьким).
ИНАЧЕ
НАЧАЛО ЦИКЛА
формируется узел дерева, в информационное поле которого помещается знак операции, ссылка на правое поддерево - элемент на вершине стека операндов, ссылка на левое поддерево - следующий элемент из стека операндов. Обработанные операнды удаляются из стека, а вместо них помещается ссылка на созданный узел дерева.
Эти действия
ПОВТОРЯТЬ до тех пор,
ПОКА приоритет обрабатываемой операции не станет больше приоритета операции на вершине стека или стек не станет пустым.
После этого знак операции помещается в стек операций.
ЕСЛИ текущий символ - «)»,
ТО создается узел, информационное поле которого - знак операции из вершины стека, правое и левое поддеревья - элементы из стека операндов. После построения дерева знак опера-



...два поля — ссылки на поддеревья...

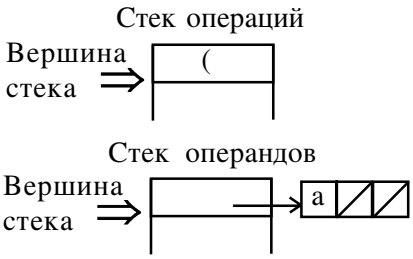
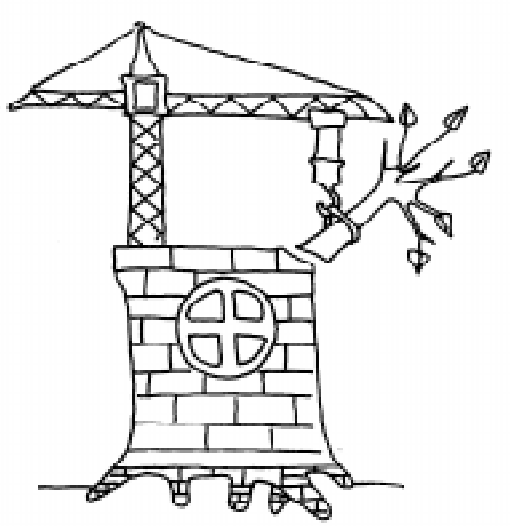


Рисунок 1.

ции исключается из стека операций, а ссылки на операнды из стека операндов. Созданая ссылка помещается в стек операндов. Этот процесс продолжается до тех пор, пока на вершине стека операций не окажется открывающая скобка, которая исключается из стека.

ЕСЛИ текущий символ - признак конца формулы,
ТО,
ЕСЛИ стек операций не пуст,
ТО формируется узел дерева, информационное поле которого - знак операции из вершины стека операций, а ссылки на правое и левое поддеревья - элементы из вершины стека операндов (которые после присоединения к дереву исключаются из стека операндов); ссылка на построенный узел помещается в стек операндов.
ЕСЛИ стек операций пуст,
ТО на вершине стека операндов лежит ссылка на корень сформированного дерева и ничего делать не надо.
Все действия с начала алгоритма
ПОВТОРЯТЬ до тех пор,
ПОКА стек операций не станет пустым.
В стеке операндов остается один элемент - корень построенного дерева.



...дерево операций строится за один просмотр...

Пример.

Рассмотрим формулу (a+b–c*d)/k. Так как первый символ – открывающая скобка, то она заносится в стек операций. При анализе следующего символа создается узел дерева, и ссылка на него хранится в стеке операндов (рисунок 1). Следующий символ – «+», его приоритет выше приоритета скобки, поэтому знак помещается в стек операций. После анализа символа «b» наступит состояние, изображенное на рисунке 2. Знак операции «–» имеет тот же приоритет, что и знак операции «+», находящейся в вершине стека операций. В этом случае формируется узел, информационное поле которого равно знаку операции в вершине стека операций, ссылка на правое поддерево – элемент в вершине стека операндов, ссылка на левое поддерево – следующий элемент из стека операндов. Сформированное дерево показано на рисунке 3. Знак операции убирается из стека операций: элементы, являющиеся ссылками на правое и левое поддеревья, убираются из

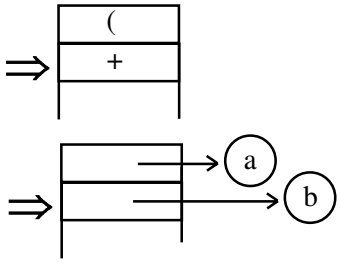


Рисунок 2.

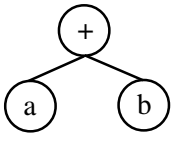


Рисунок 3.

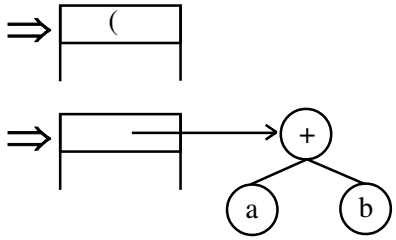


Рисунок 4.

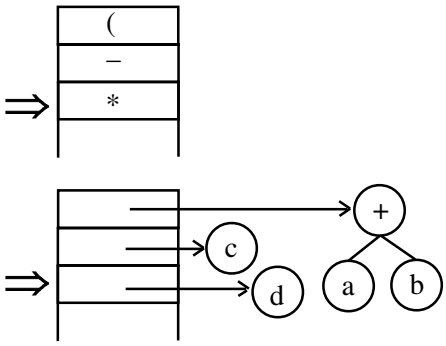


Рисунок 5.

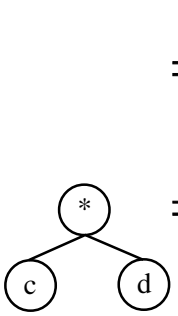


Рисунок 6.

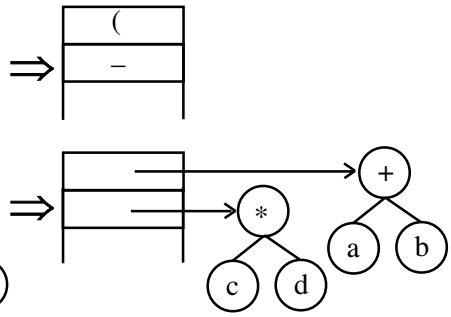


Рисунок 7.

стека операндов. Таким образом, стек операций и стек операндов примут вид, как на рисунке 4. Так как приоритет «(» на вершине стека меньше приоритета знака «–» , то знак «–» помещается в стек операций. После анализа следующих трех символов состояние стеков будет такое, как изображено на рисунке 5. Анализируемый символ – закрывающая скобка. В этом случае создается узел дерева (рисунок 6), и ссылка на него помещается в стек операндов (рисунок 7). Приоритет очередного символа «)» ниже приоритета знака операции в вершине стека операций, поэтому, как описано ранее, создается узел дерева, представленного на рисунке 8. Теперь на вершине стека операций – открывающая скобка (которая исключается из стека), в стек операндов помещается ссылка на построенное дерево. Далее знак операции «/» заносится в стек операций, в стек операндов заносится ссылка на узел, соответствующий операнду. Когда анализируем символ «k», ситуация такая, как на рисунке 9. Так как символ «.» означает, что анализ формулы завершен, то строится узел дерева (рисунок 10), и ссылка на него помещается в стек операндов. Стек операций пуст, в стеке операндов находится единственный элемент – ссылка на построенное дерево.

Задача 8.

Уровень 1. Продемонстрируйте работу описанного алгоритма, построив дерево операций для формулы

A*(B+D*(C–(D+E)))*(A+B)).

Проследите, как изменяется содержимое стека при анализе формулы.

Уровень 2. Напишите программу, которая, получив в качестве исходных данных формулу, строит по ней дерево операций согласно алгоритму Э. Дейкстры.

Формат ввода:

Скобочная инфиксная запись формулы.

Формат вывода:

Номер узла	Информац. поле	Ссылка на левое поддерево	Ссылка на правое поддерево
1	ИП 1	Л 1	П 1
2	ИП 2	Л 2	П 2
...	...	...	...
n	ИП n	Л n	П n

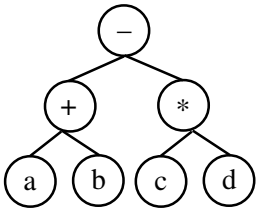


Рисунок 8.

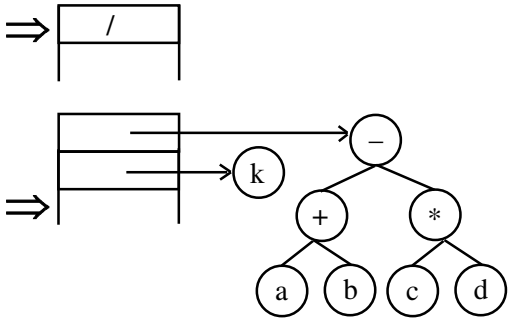


Рисунок 9.

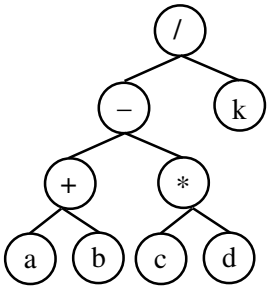


Рисунок 10.

Пример.

Ввод:

a*(b–c/d)

Вывод:

1	*	2	3
2	a	–	–
3	–	4	5
4	b	–	–
5	/	6	7
6	c	–	–
7	d	–	–

6. ПОСТРОЕНИЕ ФОРМУЛЫ ПО ДЕРЕВУ

Обсудим, как по дереву операций получить формулу в инфиксной записи. Самый простой способ состоит в том, чтобы при обработке дерева операций каждый операнд заключался в круглые скобки. В этом случае будет получена формула в инфиксной записи, но она будет содержать слишком много скобок.

Рассмотрим рекурсивный алгоритм, который по заданной формуле, представленной деревом операций, строит формулу в инфиксной записи, расставляя в ней лишь необходимые скобки.

В основе алгоритма лежат следующие действия:

ЕСЛИ в узле дерева знак операции,
ТО (будем проверять, какая информация хранится в корнях левого и правого поддеревьев).

Обсудим, как по дереву операций получить формулу...

ЕСЛИ приоритет операции выше приоритета операций в корнях поддеревьев,
ТО операнд, соответствующий поддереву, следует заключить в скобки.
ЕСЛИ операции имеют одинаковый приоритет,
ТО левый операнд в скобки заключать не надо, а для правого операнда требуется дополнительно проанализировать сочетание знаков в корне и в правом поддереве. Например, если в корне хранится минус, а в корне правого поддерева плюс, то правый операнд требуется заключить в скобки.
ЕСЛИ приоритет операции ниже приоритета операции в корне поддерева,
ТО операнд в скобки не заключается.

Задача 9

Уровень 1. Представьте формулу (A*(B+(C+(D–E)))) деревом операций. Постройте для данной формулы, представленной деревом операций, формулу в инфиксной форме по предложенному алгоритму. Сравните построенную по дереву операций формулу с исходной формулой.

Уровень 2. Напишите программу, которая по формуле, представленной деревом операций, строит формулу в инфиксной записи, расставляя в ней лишь необходимые скобки.

Формат ввода:

Номер узла	Информац. поле	Ссылка на левое поддерево	Ссылка на правое поддерево
1	ИП 1	Л 1	П 1
2	ИП 2	Л 2	П 2
...	...	...	...
n	ИП n	Л n	П n

Формат вывода:

Инфиксная скобочная запись формулы.

Пример.

Ввод:

1	*	2	3
2	–	4	5
3	a	–	–
4	b	–	–
5	c	–	–

Вывод:

a*(b-c)

7. ОБХОД ДЕРЕВА

Если формула задана деревом операций, то бесскобочные записи формул легко получить, выполняя рекурсивные алгоритмы обхода дерева операций. Если формула содержит лишь переменную или константу, то такая формула уже представлена как в префиксной, так и в постфиксной записях. Действие «обработать корень» может состоять, например, в том, чтобы записать информацию, соответствующую корню, в строку результата.

Алгоритм построения префиксной записи (pref_form):

- **Обработать корень.**
- **Построить префиксную запись формулы, соответствующей левому поддереву, то есть применить алгоритм pref_form к левому поддереву.**
- **Построить префиксную запись формулы, соответствующей правому поддереву, то есть применить алгоритм pref_form к правому поддереву.**

Алгоритм построения постфиксной записи (post_form):

- **Построить постфиксную запись формулы, соответствующей левому поддереву, то есть применить алгоритм post_form к левому поддереву.**
- **Построить постфиксную запись формулы, соответствующей правому поддереву, то есть применить алгоритм post_form к правому поддереву.**
- **Обработать корень.**

Задача 10.

Уровень 1. В приведенных выше рекурсивных алгоритмах обхода дерева используются три операции:

1. Обработать корень.
2. Обработать левое поддерево.
3. Обработать правое поддерево.

Если эти операции стоят в порядке 1, 2, 3, получается префиксная запись, если в порядке 2, 3, 1, то получается постфиксная запись.

Рассмотрите остальные 4 возможных порядка выполнения этих операций и дайте трактовку всем получающимся при этом результатам работы алгоритмов.

Проиллюстрируйте алгоритмы на следующем примере: (A/(B*C+D–E)).

Уровень 2. Напишите процедуру, которая по формуле, представленной деревом операций, строит формулу в префиксной форме.

Формат ввода:

Номер узла	Информац. поле	Ссылка на левое поддерево	Ссылка на правое поддерево
1	ИП 1	Л 1	П 1
2	ИП 2	Л 2	П 2
...	...	...	...
n	ИП n	Л n	П n

Формат вывода:

Префиксная запись формулы.

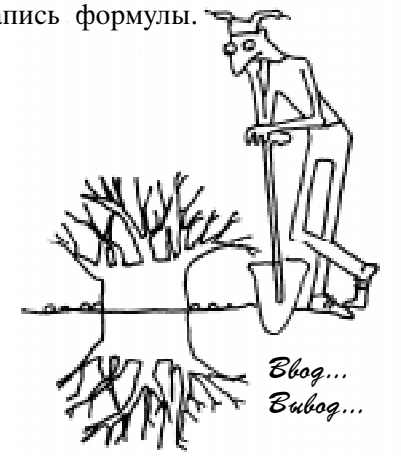
Пример.

Ввод:

1 * 2 3
2 – 4 5
3 a – –
4 b – –
5 c – –

Вывод:

*a–bc



ДРУГИЕ ЗАДАЧИ

8. МЕТОД ИСЧЕРПЫВАЮЩЕГО ПЕРЕБОРА

Рассмотрим формулу вида $a_1 \oplus a_2 \oplus \ldots \oplus a_n$, в которой переменные $a_1, a_2, \ldots, a_n$ принимают целые значения, знак $\oplus$ – обозначает операцию плюс или минус. Требуется определить набор знаков операций, при котором формула принимает заданное значение b, то есть верно $a_1 \oplus a_2 \oplus \ldots \oplus a_n = b$

Эту задачу можно решить методом исчерпывающего перебора. При решении задач методом исчерпывающего перебора следует обратить внимание на два момента:

1. Требуется определить порядок, в котором следует рассматривать варианты.
2. Убедиться в том, что все варианты рассмотрены.

Если формула содержит n значений, то в конструируемой формуле должно быть использовано n–1 знаков операций. Так как по условию задачи разрешено использовать только знаки плюс или минус, то возможно 2^{n-1} различных комбинаций знаков. Например, если исследуется формула, содержащая четыре переменных вида $a_1 \oplus a_2 \oplus a_3 \oplus a_4$, то следует рассмотреть и вычислить восемь формул, варианты представлены в таблице.

№ варианта	Формула	Комбинация знаков
1	$a_1-a_2-a_3-a_4$	– – –
2	$a_1-a_2-a_3+a_4$	– – +
3	$a_1-a_2+a_3-a_4$	– + –
4	$a_1-a_2+a_3+a_4$	– + +
5	$a_1+a_2-a_3-a_4$	+ – –
6	$a_1+a_2-a_3+a_4$	+ – +
7	$a_1+a_2+a_3-a_4$	+ + –
8	$a_1+a_2+a_3+a_4$	+ + +

Если знаку минус сопоставить ноль, а знаку плюс единицу, то комбинацию знаков можно трактовать как двоичную запись числа α, где $0 \le \alpha \le 2^{n-1}-1$. Разбор всех возможных комбинаций знаков означает анализ двоичных представлений всех чисел α, где $0 \le \alpha \le 2^{n-1}-1$. Если n равно 4, то выписанные ранее комбинации знаков соответствуют двоичным представлениям чисел от 0 до 7, представленным в таблице:

№ варианта	Число в двоичной системе счисления	Комбинация знаков
1	000	– – –
2	001	– – +
3	010	– + –
4	011	– + +
5	100	+ – –
6	101	+ – +
7	110	+ + –
8	111	+ + +

Теперь надо уточнить порядок рассмотрения вариантов. Самый простой подход состоит в том, чтобы переход от одной комбинации к другой означал переход от двоичной записи числа α к двоичной записи числа $\alpha+1$, то есть прибавление к текущему значению двоичного числа единицы. Если в качестве начальной комбинации знаков взять комбинацию из всех минусов (комбинация соответствует числу ноль), то после выполнения $2^{n-1}-1$ числа сложений будет получена комбинация знаков, состоящая только из знаков плюс и соответствующая двоичной записи числа $2^{n-1}-1$. Все возможные комбинации знаков будут исчерпаны.

Задача 11.

Уровень 1. На одной из олимпиад по программированию для школьников была предложена следующая задача. В формуле ((((1?2)?3)?4)?5)?6 вместо знака ? поставьте знаки арифметических операций (плюс, минус, умножить, разделить нацело) так, чтобы результат вычислений равнялся 35. Достаточно найти одно решение.

Уровень 2. Напишите программу, которая для формулы вида $a_1 \oplus a_2 \oplus ... \oplus a_n$ определяет комбинации знаков, при которых значение формулы совпадает с заданным значением b. Переменные a_i принимают целые значения, знак $\oplus$ обозначает операцию сложения, вычитания, умножения или деления нацело.

Формат ввода:

Сначала вводится число переменных в формуле, затем значения переменных через пробел, затем значение формулы:

n

$a_1\ a_2\ ...\ a_n$

b

Формат вывода:

$a_1 \oplus a_2 \oplus ... \oplus a_n = b$

Пример.

Ввод:

2
3 5
15

Вывод:

3*5=15

Вы, наверное, знаете задачу о количестве «счастливых» билетов, то есть шестизначных наборов цифр, у которых сумма трех первых цифр равна сумме трех последних.

Встречаются и другие определения «счастливых» билетов. Например, сумма четных цифр равна сумме нечетных. Дальше всех идет математик А.В. Степанов. Он считает билет счастливым, если между его цифрами можно расставить знаки арифметических операций таким образом, чтобы в результате получился ноль. Можно увидеть, что билеты, «счастливые» по первым двум определениям, являются счастливыми по Степанову. Для шестизначного набора цифр: $\overline{a_1a_2a_3a_4a_5a_6}$

$a_1+a_2+a_3=a_4+a_5+a_6 \Rightarrow a_1+a_2+a_3-a_4-a_5-a_6=0$

$a_1+a_3+a_5=a_2+a_4+a_6 \Rightarrow a_1-a_2+a_3-a_4+a_5-a_6=0$

Найдите количество билетов, «счастливых» в обычном смысле и «счастливых» по Степанову, и определите, во сколько раз выгоднее придерживаться мнения математиков по сравнению с бытовыми представлениями.

Задача 12.

Уровень 2. Напишите программу, которая удаляет из введенной формулы все лишние скобки.

Формат ввода:

Последовательность латинских букв, открывающих и закрывающих круглых скобок, знаков арифметических операций, являющаяся правильной записью формулы.

Формат вывода:

Последовательность с теми же буквами и знаками операций, стоящими в том же порядке, но с удаленными лишними скобками.

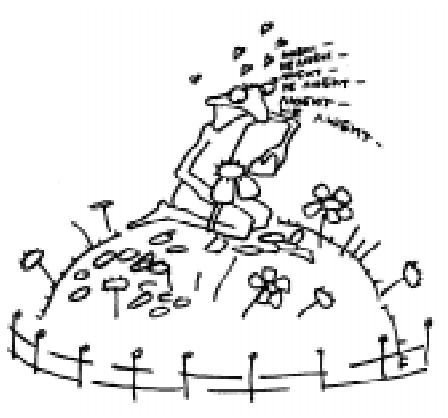
Пример.

Ввод:

((a*(b/c))–(d+m)

Вывод:

a*b/c–(d+m)



Метод исчерпывающего перебора...

Дмитриева Марина Валерьевна, доцент кафедры информатики СПбГУ.

Поздняков Сергей Николаевич, профессор кафедры высшей математики СПбГЭТУ.