Математическая Логика.

Клящицкий Григорий (GK)

1 Введение

Логика в форме математических формул была оформлена английским философом Булем. Булева математика лежит в основе электроники и компьютерных технологий. Несмотря на огромный успех логики Буля, мне кажется, что она заужена, и не включает «обратную» логику. В этой статье делается попытка расширить Булеву алгебру, включив в неё обратные операции.

2 Алгебра Логики

2.1 Логические переменные (числа) и операции

2.1.1 Логические переменные (значения)

Буль определил логические числа как числа, способные принимать два значения: True (1) и False (0).
Эти значения (на мой взгляд) являются неполным исчерпанием возможностей и предлагается ввести в дополнение ещё одно значение – Неопределённость: Undf (?). Переменная имеет значение «Undf», если её значение не определено или может иметь оба значения (True и False).

Логические Числа

0 – False
1 – True
? – Undf (Неопределённость может иметь значения как True так и False)

2.1.2 Логические операции

В Булевай алгебре определены три операции: сложение (OR), умножение (AND) и отрицание (NOT). В обычной математике мы определяем также «обратные» операции. Предлагается дополнить Булевы операнды ещё двумя операциями: логическим вычитанием (RO) и логическим делением (NAD). Смысл этих операций мы определим чуть позже.

Логические операции

^ +	- сложение	OR
& *	- умножение	AND
~ -	- отрицание	NOT
v -	- вычитание	RO
÷ /	- деление	NAD

3 Логические операции над числами

3.1 Булевы операции

3.1.1 Логическое сложение - OR

Сложение Булевых чисел

1 ^ 1 = 1 (1.1)

0 ^ 0 = 0 (1.2)

0 ^ 1 = 1 (1.3)

Сложение с Неопределённостью

0 ^ ? = ? (1.4)

1 ^ ? = 1 (1.5)

? ^ ? = ? (1.6)

Логическая операция «сложение» обладает переставляемостью, то есть порядок слагаемых не меняет результат и потому сложения в обратном порядке будут аналогичны.

3.1.2 Логическое умножение - AND

Умножение Булевых чисел

0 & 0 = 0 (2.1)

1 & 1 = 1 (2.2)

0 & 1 = 0 (2.3)

Умножение с Неопределённостью

0 & ? = 0 (2.4)

1 & ? = ? (2.5)

? & ? = ? (2.6)

Логическая операция «умножение», как и логическое сложение, имеет свойство переставляемости, то есть порядок сомножителей в логическом уравнении не меняет результат и потому умножение в обратном порядке будет аналогично приведённым.

3.1.3 Логическое отрицание - NOT

Отрицание Булевых чисел

~ 1 = 0 (3.1)

~ 0 = 1 (3.2)

Отрицание Неопределённости

~ ? = ? (3.3)

3.2 Обратные операции

Обратные опрерации в математике определяются через прямые операции. То есть, если над неким числом (A) произвести прямую и обратную операции с числом (c), то результатом будет исходное число (A).

$$A\ (+\ c - c) = A$$

$$A\ (*\ c\ /\ c) = A$$

Иными словами в математике прямая и обратная опрерации дают «нулевой эффект»:

$$(+c) - c = 0$$

$$c\ /\ c = 1$$

Давайте попробуем определить обратные логические опреции как противоположность прямым операциям:

$$A\ (\wedge\ c \vee c) = A$$

$$A\ (\&\ c \div c) = A$$

Заметьте, что эффект «прямой-обратной» ($c \vee c$) или ($c \div c$) операций при таком определении обратных операций вообще неопределён.

Существует иная возможность определить обратную операцию как комбинацию с отрицанием (метод суперпозиции):

$$\vee = (\wedge\ \sim)$$

$$\div = (\&\ \sim)$$

При таком подходе получим:

$$A \vee c = A \wedge (\sim c)$$

$$A \div c = A\ \&\ (\sim c)$$

Давайте попробуем подойти к вопросу с другой стороны (всё-таки логика и арифметика – разные вещи). В чём логический смысл обратных операций? Предположим, что есть некое логическое уравнение, в котором:

- Известна Операция
- Известен один их составляющих – Компонент
- Известен Результат

Возможно ли определить другой Компонент исходной операции. Например: X ^ 0 = 1

Тогда обратную операцию можно было бы определить как: X = 1 ∨ 0. То есть как операцию, дающую ответ о втором Компоненте.

3.2.1 Логическое вычитание - RO

Булево Вычитание

Логическое вычитание должно позволять определить второе слагаемое (Компонент) при известном Результате и Слагаемом.

1. Исходное уравнение: X ^ 1 = 1 (3.2.1-1)
   Тогда наша задача выглядит как: X = 1 ∨ 1 (3.2.1-2)
   Заметим, что соответствующие (3.2.1-1) уравнения логики : 1 ^ 1 = 1 и 0 ^ 1 = 1.
   То есть для X возможны значение как 1 (True) так и 0 (False). В таком случае переменная X имеет значение ? (Undf) и исходное уравнение (3.2.1-2) принимает вид:
   1 ∨ 1 = ? (4.1)

2. Исходное уравнение: X ^ 0 = 1 (3.2.1-3)
   Задача выглядит как: X = 1 ∨ 0 (3.2.1-4)
   Заметим, что возможное для (3.2.1-3) решение: 1 ^ 0 = 1,
   и задача (3.2.1-4) имеет однозначное решение:
   1 ∨ 0 = 1 (4.2)

3. Исходное уравнение: X ^ 0 = 0 (3.2.1-5)
   Задача выглядит как: X = 0 ∨ 0 (3.2.1-6)
   Заметим, что уравнение (3.2.1-5) имеет решение: 0 ^ 0 = 0,
   и задача (3.2.1-6) имеет однозначное решение:
   0 ∨ 0 = 0 (4.3)

4. И, наконец: X ^ 1 = 0 (3.2.1-7)
   Тогда наша задача выглядит как: X = 0 ∨ 1 (3.2.1-8)
   Заметим, что уравнения логики не имеют решения для уравненеия (3.2.1-7). Это значит, что в нашем случае нам придётся ввести ещё одну логическую переменную. Её нельзя назвать неопределённость. Это более похоже на результат «деления на ноль», что даёт «Невозможность» - неразрешимую ошибку. Давайте обозначим «Невозможность» символом Ѣ (Imps). Тогда:

0 ∨ 1 = ѣ (4.4)

Варианты: 1 ^ X = 1; 0 ^ X = 1; 0 ^ X = 0 и 1 ^ X = 0 полностью аналогичны выше рассмотренным примерам (в силу перестановочного закона) и ничего не добавляют.

Тогда вычитание для Булевых чисел будет выглядеть как:

1 ∨ 1 = ? (4.1)
1 ∨ 0 = 1 (4.2)
0 ∨ 0 = 0 (4.3)
0 ∨ 1 = ѣ (4.4)

Заметим, что как и в обычной математике порядок компонентов в операции логическое вычитание имеет значение.

Для сравнения приведём результат операций, в методе «нулевой эффект»:

1 ∨ *1* = *?* →	1 (∨ 1 ^ 1) = ? ^ 1	→	1 = ? ^ 1	(1.5)
1 ∨ 0 = 1 →	1 (∨ 0 ^ 0) = 1 ^ 0	→	1 = 1 ^ 0	(1.3)
0 ∨ 0 = 0 →	0 (∨ 0 ^ 0) = 0 ^ 0	→	0 = 0 ^ 0	(1.2)
0 ∨ 1 = ѣ →	0 (∨ 1 ^ 1) = ѣ ^ 1	→	0 = ѣ ^ 1	(X)

Обратите внимание, что результаты оказались близкие (операции с ѣ (Imps) не определены).

Давайте посмотрим как обстоит дело с методом «суперпозиция»:

1 ∨ 1:	1 ^ (~ 1) = 1 ^ 0 = 1	→	<u>1 ∨ 1 = 1</u>	(4.1)
1 ∨ 0:	1 ^ (~ 0) = 1 ^ 1 = 1	→	1 ∨ 0 = 1	(4.2)
0 ∨ 0:	0 ^ (~ 0) = 0 ^ 1 = 1	→	<u>0 ∨ 0 = 1</u>	(4.3)
0 ∨ 1:	0 ^ (~ 1) = 0 ^ 0 = 0	→	<u>0 ∨ 1 = 0</u>	(4.4)

Как видим результаты совсем не совпадают. Из этого уже можно сделать вывод , что метод «суперпозиции» нельзя применять для обратных операций.

Вычитание с Неопределённостью

1. Исходное уравнение имеет вид: X ^ ? = 1 (3.2.1-9)
   Тогда наша задача выглядит как: X = 1 ∨ ? (3.2.1-10)
   Уравнение логики выглядит как: 1 ^ ? = 1.
   То есть наше уравнение принимает вид:
   1 ∨ ? = 1 (4.5)

2. Исходное уравнение: X ^ ? = 0 (3.2.1-11)
   Тогда наша задача выглядит как: X = 0 ∨ ? (3.2.1-12)
   Уравнения логики не имеют решения, и задача имеет результат:
   0 ∨ ? = ѣ (4.6)

3. Исходное уравнение: X ^ ? = ? (3.2.1-13)
   Тогда наша задача выглядит как: X = ? v ? (3.2.1-14)
   Уравнения имеют решения: 0 ^ ? = ? и ? ^ ? = ?,
   и задача имеет очевидное решение:
   ? v ? = ? (4.7)

4. Исходное уравнение: X ^ 0 = ? (3.2.1-15)
   Тогда наша задача выглядит как: X = ? v 0 (3.2.1-16)
   Уравнения имеет решение: 0 ^ ? = ?,
   и задача имеет очевидное решение:
   ? v 0 = ? (4.8)

5. Исходное уравнение: X ^ 1 = ? (3.2.1-17)
   Тогда наша задача выглядит как: X = ? v 1 (3.2.1-18)
   Уравнения не имеют логического решения, и задача имеет решение:
   ? v 1 = Ѣ (4.9)

Вычитание с Неопределённостью будут выглядеть как:

1 v ? = 1 (4.5)
0 v ? = Ѣ (4.6)
? v ? = ? (4.7)
? v 0 = ? (4.8)
? v 1 = Ѣ (4.9)

3.2.2 Логическое деление - NAD

Булево деление

1. Исходное уравнение: X & 1 = 1 (3.2.2-1)
   Тогда наша задача выглядит как: X = 1 ÷ 1 (3.2.2-2)
   Заметим, что уравнения логики имеют одно решение: 1 & 1 = 1.
   и задача имеет однозначное решение:
   1 ÷ 1 = 1 (5.1)

2. Исходное уравнение: X & 0 = 1 (3.2.2-3)
   Тогда наша задача выглядит как: X = 1 ÷ 0 (3.2.2-4)
   Уравнения не имеют решения, и уравнение имеет вид:
   1 ÷ 0 = Ѣ (5.2)

5. Исходное уравнение: X & 0 = 0 (3.2.2-5)

Тогда наша задача выглядит как: X = 0 ÷ 0 (3.2.2-6)

Имеется два решения: 0 & 0 = 0 и 1 & 0 = 0,

и задача имеет решение:

0 ÷ 0 = ? (5.3)

6. Исходное уравнение: X & 1 = 0 (3.2.2-7)

   Тогда наша задача выглядит как: X = 0 ÷ 1 (3.2.2-8)

   Имеется единственное решение для (3.2.2-7): 0 & 1 = 0,

   и решение (3.2.2-8) будет:

   0 ÷ 1 = 0 (5.4)

Деление для Булевых чисел будет выглядеть как:

1 ÷ 1 = 1 (5.1)

1 ÷ 0 = Ъ (5.2)

0 ÷ 0 = ? (5.3)

0 ÷ 1 = 0 (5.4)

Заметим, что как и в обычной математике, порядок компонентов в операции логическое деление имеет значение. (Варианты с переменой сомножителей полностью аналогичны выше рассмотренным примерам в силу перестановочного закона).

Деление с Неопределённостью

1. Исходное уравнение имеет вид: X & ? = 1 (3.2.2-9)

   Наша задача выглядит как: X = 1 ÷ ? (3.2.2-10)

   Уравнение логики не имеют решения, и наше уравнение принимает вид:

   1 ÷ ? = Ъ (5.5)

2. Исходное уравнение: X & ? = 0 (3.2.2-11)

   Тогда наша задача выглядит как: X = 0 ÷ ? (3.2.2-12)

   Уравнение логики выглядит как: 0 & ? = 0,

   и задача имеет решение:

   0 ÷ ? = 0 (5.6)

3. Исходное уравнение: X & ? = ? (3.2.2-13)

   Тогда наша задача выглядит как: X = ? ÷ ? (3.2.2-14)

   Уравнение имеет решения: 0 & ? = ? и ? & ? = ?,

   и задача имеет решение:

   ? ÷ ? = ? (5.7)

4. Исходное уравнение: X & 0 = ? (3.2.2-15)

Тогда наша задача выглядит как: X = ? ÷ 0 (3.2.2-16)

Уравнение не имеет решения, и задача имеет вид:

? ÷ 0 = ѣ (5.8)

5. Исходное уравнение: X & 1 = ? (3.2.2-17)

   Наша задача выглядит как: X = ? ÷ 1 (3.2.2-18)

   Уравнение имеет решение: ? & 1 = ?,

   и задача имеет решение:

   ? ÷ 1 = ? (5.9)

Деления с Неопределённостью будут выглядеть как:

1 ÷ ? = ѣ (5.5)
0 ÷ ? = 0 (5.6)
? ÷ ? = ? (5.7)
? ÷ 0 = ѣ (5.8)
? ÷ 1 = ? (5.9)

3.2.3 Операции методом «нулевой эффект»:

Для сравнения приведём результат операций, по методу «нулевой эффект»:

1 ∨ *1* = *?* → 1 (∨ 1 ^ 1) = ? ^ 1 → 1 = ? ^ 1 (1.5)
1 ∨ 0 = 1 → 1 (∨ 0 ^ 0) = 1 ^ 0 → 1 = 1 ^ 0 (1.3)
0 ∨ 0 = 0 → 0 (∨ 0 ^ 0) = 0 ^ 0 → 0 = 0 ^ 0 (1.2)
0 ∨ 1 = ѣ → 0 (∨ 1 ^ 1) = ѣ ^ 1 → 0 = ѣ ^ 1 (X)

Обратите внимание, что результаты оказались близкие (операции с ѣ (Imps) не определены).

1 ∨ ? = 1 → 1 (∨ ? ^ ?) = 1 ^ ? → 1 = 1 ^ ? (1.5)
0 ∨ ? = ѣ → 0 (∨ ? ^ ?) = ѣ ^ ? → 0 = ѣ ^ ? (X)
? ∨ ? = ? → ? (∨ ? ^ ?) = ? ^ ? → ? = ? ^ ? (1.6)
? ∨ 0 = ? → ? (∨ 0 ^ 0) = ? ^ 0 → ? = ? ^ 0 (1.4)
? ∨ 1 = ѣ → ? (∨ 1 ^ 1) = ѣ ^ 1 → ? = ѣ ^ 1 (X)

В данном случае мы снова имеем совпадение. Однако, не все уравнения первой группы имеют представление. Например уравнение 1 ^ 1 = 1 (1.1) в списке не встречается. В то же время уравнение (1.5) повторяется дважды.

1 ÷ *1* = *1* → 1 (÷ 1 & 1) = 1 & 1 → 1 = 1 & 1 (2.2)
1 ÷ 0 = ѣ → 1 (÷ 0 & 0) = ѣ & 0 → 1 = ѣ & 0 (X)
0 ÷ 0 = ? → 0 (÷ 0 & 0) = ? & 0 → 0 = ? & 0 (2.4)
0 ÷ 1 = 0→ 0 (÷ 1 & 1) = 0 & 1 → 0 = 0 & 1 (2.3)

Результаты совпали.

1 ÷ ? = ƀ →	1 (÷ ? & ?) = ƀ & ?	→	1 = ƀ & ?	(X)
0 ÷ ? = 0 →	0 (÷ ? & ?) = 0 & ?	→	0 = 0 & ?	(2.4)
? ÷ ? = ? →	? (÷ ? & ?) = ? & ?	→	? = ? & ?	(2.6)
? ÷ 0 = ƀ →	? (÷ 0 & 0) = ƀ & 0	→	? = ƀ & 0	(X)
? ÷ 1 = ? →	? (÷ 1 & 1) = ? & 1	→	? = ? & 1	(2.5)

В данном случае мы имеем совпадение. И снова уравнение (2.1) в списке не встречается. В то же время уравнение (2.4) повторяется дважды.

Из приведённых таблиц видно, что метод «нулевой эффект» в некоторых случаях даёт совпадение для обратных операций, однако в случае уравнений (1.1), (2.1) результат может оказаться неполным, в то время как для уравнений (1.5) и (2.4) не определённым.

4 VGK Логика

По роду своей деятельности я занимаюсь програмированием. Последние несколько лет я работал над бизнес-пакетом **VGK Project**. В ходе работы я столкнулся с проблемой дефицитности Булевых чисел.

Задача сводилась к определению свойства объекта. Результат имел Булев тип (т.е. True или False). Для определения значения требовались вычисления. Если значение используется часто, то желательно избежать многократного вычисление, если результат не меняется. Имеет смысл определить переменную (Boolean) и, произведя вычисления, хранить в ней результат (экономит время и CPU). Проблема заключалась в том, что декларируя Булеву переменную (ещё до вычислений), она уже приобретала Булево значение. При таком варианте невозможно определить, является ли значение переменной результатом вычисления (осмысленное значение) или начального значения (неосмыленное значение).

Для решения проблемы я определил новый тип – vgkBool, которое могло иметь три значение: True, False и Undf. При декларации переменная получала значение Undf. Осмысленными могли быть только значения True или False.

Я принял название vgkBool (логических переменных) и vgkLogic (результатов логических операций) для расширенного типа логических переменных и операций. Привожу результаты наших рассуждений, сведённые в единую таблицу.

vgkBool		**vgkLogic**		**Комментарии**
False	**0**	**False**	**0**	
True	**1**	**True**	**1**	
Undf	**?**	**Undf**	**?**	0 и/или 1
		Imps	**ƀ**	Нет значения - невозможно

Table 1: Логические значения

Булевы Операции				
Сложение		**Умножение**		**Отрицание**
1 ^ 1 = 1		0 & 0 = 0		~ 1 = 0
0 ^ 0 = 0		1 & 1 = 1		~ 0 = 1
0 ^ 1 = 1		0 & 1 = 0		~ ? = ?
0 ^ ? = ?		0 & ? = 0		
1 ^ ? = 1		1 & ? = ?		
? ^ ? = ?		? & ? = ?		
Обратные Операции				
Вычитание		**Деление**		
1 v 1 = ?		1 ÷ 1 = 1		
1 v 0 = 1		1 ÷ 0 = ƀ		
0 v 0 = 0		0 ÷ 0 = ?		
0 v 1 = ƀ		0 ÷ 1 = 0		
1 v ? = 1		1 ÷ ? = ƀ		
0 v ? = ƀ		0 ÷ ? = 0		
? v ? = ?		? ÷ ? = ?		
? v 0 = ?		? ÷ 0 = ƀ		
? v 1 = ƀ		? ÷ 1 = ?		

Table 2: Логические операции

5 Что с этим делать?

Мы уже упомянули, что введение значение Undf (?) может иметь вполне реальное применение в программировании сегодняшнего уровня. Появление квантовых компьютеров без сомнения потребует подобного расширения.

В отношении обратных операций ответ, возможно, будет не вполне ожидаемым. Для этого мне бы хотелось взглянуть на программирование в целом.

5.1 Задачи программирования

Итак, изначальной задачей программирования было ускорение вычислений и обработки данных.

Вычислительная

Типичная цель вычислительного программирования состояла в ускоренном решении объёмных вычислительных задач. То есть стояла конкретная задача (научная или прикладная), требующая значительных (или огромных) усилий для вычислений или нахождения решений (системы уравнений). Эта задача поддавалась програмированию. Для каждой конкретной задачи создавалась программа. В программу закладывался набор входных данных и запускался «жёсткий» код – алгоритм, описывающий конкретную задачу. На выходе мы получали решение.

Обработка данных

Другое приложение состояло в обработки больших массивов данных (статистики, научных, финансовых). Задача также была полностью определена и создавался также «жёсткий» алгоритм обработки.

Обе цели определялись на ранней стадии развития компьютерных технологий. Задачи жёстко определялись и решения не предполагали какого-либо диалога с пользователем. Впоследствии подход претерпел существенные изменения и диалоговое програмирование стало главным направление.

Пользовательское приложение

Диалог с пользователем изменил подход к програмированию. Мы более не создаём «жесткий» алгоритм. Напротив, програмист пишет алгоритм «Ответа на Запрос». То есть создаются возможности для пользователя формировать запрос машине. В ответ на запрос пользователя, запускается соответствующий програмный код. В итоге програмист пишет ответы на все предполагаемые запросы. Какой запрос и когда будет инициирован пользователем, програмист не знает и не контролирует. В некоторых случаях мы пишем код, который, возможно, никогда и не вызывается.

Диалоговый подход значительно расширил применение программирования. Но остаётся вопрос: все ли возможные задачи мы уже реализовали? Если всмотреться повнимательнее, то все три типа задач в своей основе сводятся к следующему требованию: задаётся условие и код – считай и выполняй.

5.2 Аналитическая Задача

В жизни есть иная категория задач, которая не относится к рассмотренным типам задач. Например, мы получили некий результат как итог наших усилий. Но он нас не удовлетворил. Мы ожидали иного. Почему мы получили именно такой результат? В чём (возможно) была ошибка или что возможно надо изменить, чтобы результат был другим.

Похожий тип задач – аналитика. Совершено преступление, есть улики, есть масса информации. Необходимо проанализировать результат и известные компоненты дабы определить недостающие элементы или найти ответ на конкретный вопрос (кто преступник, например). Вы, конечно, понимаете, что аналитика гораздо шире, чем только «детективная» задача.

Надеюсь, что вы уже почувствовали, к чему я клоню – Булева логика не имеет обратныхопераций, а они составляют основу подобной аналитики. Расширение Булевой логики до vgk-логики открывает принципиально новые возможности для програмирования, расширяет круг возможных приложений, включая в них вопросы анализа. Возможно, некоторые смелые читатели скажут, что это приблизит нас к созданию «искуственного интеллекта». Я этого не говорил и утверждать подобное не возмусь. Вообще, я был бы очень расстроен, если бы нам удалось приблизится к искуственному интеллекту – что-то должно остаться за человеком. Я так думаю...

5.3 Програмирование с Обратной Логикой

При исследовании обратной логики у нас нет четкой программы. Напротив, возможны несколько вариантов (иногда много), которые нам необходимо исследовать. Каждый отдельный вариант мы назовем логической цепью. В таком случае задача сводится к анализу множества логических цепей.

Результатом такого анализа может служить определённый ответ (вывод) (идеальный случай) или несколько возможных ответов, каждый из которых представляет собой потенциальное решение задачи. В зависимости от конкретной задачи множественные результаты также могут служить ответом на поставленный вопрос (например, в результате каких обстоятельств был получен исследуемый практический результат).

Алгоритм с расширенной логикой должен включать операции логического вычитания и логического деления. Результатом некоторых операций оказываются новые логические значения – Undf (?) и Imps (ѣ). Появление новых результатов требует от программиста принятия решений по их обработки.

Результат Imps (ѣ)

Значение Imps (ѣ) не может быть присвоено логической переменной (Table 1: Логические значения). Оно может возникнуть только как результат логической операции (Table 2:

Логические операции). Для програмиста этот результат означает обрыв логической цепи. При получении результата Imps (ѣ), дальнейшие операции по исследуемой цепи прекращаются и делается вывод о невозможности такого логического варианта.

Значение Undf (?)

Значение Undf (?) также может возникнуть в результате логической опреции. Это означает, что оба варианта (True и False) могут быть правильными для исследуемой логики. В таком случае следует рассматривать оба возможных варианта и мы имеем деление логической цепи и выполнение кода будет продолжено.

5.3.1 Деление и обрыв цепей

Итак, програмный код будет исследовать имеющиеся логические цепи. При этом возможны несколько вариантов:

1. В результате логической операции получен конкретный ответ (True или False). Исследование цепи продолжается с использование полученного (дополнительного) результата.
2. В результате логической операции получен ответ Imps (ѣ). Это означает, что не существует исходных условий, которые могли бы привести к последствиям, исследуемым в этой цепи. Это обрыв цепи и дальнейшее исследование этого варианта (логической цепи) прекращается.
3. В результате логической операции получен ответ Undf (?). Это означает, что оба варианта (True и False) возможны. При таком ответе мы можем продолжать исследование логической цепи, используя значения переменной Undf (?). Другим вариантом (возможно более предпочтительным) является образование двух цепей с присвоением переменной разных значений (True и False). Этот шаг подобен делению цепи (из одной цепи получается две логические цепи).

Из рассмотрения приведённых вариантов видно, что програмный код подобен Цепной реакции (химической или атомной), и подчиняется тем же законам. Исходя из этого можно ожидать, что в некоторых случаях возможен процесс разветвлённой цепной реакции, на практике приводящий к взрыву. Такой сценарий будет реализован при множественном получении результата Undf (?). В таком случае количество цепей будет увеличиваться экспоненциально при каждом логическом шаге. Если в основании экспоненты стоит число 1.3 - 1.5 (или большее), то дальнейшее продолжение выполнения кода не имеет смысла. В програмировании известна ошибка «зацикливания», которая идентифицируется в процессе выполнения программы и приводит к остановке програмного кода. Подобно «циклению» можно определить ошибку выполнения «взрыв» (run-time error Explosion), и останавливать выполнение программы при подобном сценарии. Это будет означать, что при имеющихся исходный данных предлагаемая логика не может быть выполнена.

Вариантов три:

- расширить исходные данные дополнительной информацией.
- проанализировать логику кода и модифицировать её.
- или использовать оба предложенных выше метода.

6 Заключение

Програмирование с vgk-логикой потребует не только пересмотра принципов программирования, разработки языков программирования, но и (возможно) развития основ програмной базы – чипов. Но если мы хотим летать дальше и быстрее, чем сегодня возможно, то без развития новых методов и подходов нам прогресса не добиться.

Не знаю, насколько вы найдёте высказанные здесь идеи полезными и разумными.
Как всегда Вам судить!

2020, Июнь