ЗАКОНЫ ФОРМАЛЬНОЙ ЛОГИКИ И ПРОГРАММИРОВАНИЕ

Ковалёв А.В., студент

Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь

Ратникова И.М. – канд. фил. наук, доцент

В данной работе представлены примеры отражения четырёх основных законов формальной логики в программировании – законов тождества, непротиворечия, исключённого третьего, достаточности основания. Показаны возможные причины допущения логических ошибок в процессе написания кода, а также доказательства применимости этих законов.

При работе с программами нужно учитывать то, что, помимо грамотного синтаксиса, важной составляющей правильно работающего кода является отсутствие логических ошибок, которые возникают при неверном применении логических законов в данном контексте.

К законам формальной логики относят закон тождества, непротиворечия, исключённого третьего и достаточного основания. Каждый из них в той или иной мере проявляется при создании программы, обеспечивая такие свойства кода, как определённость, непротиворечивость, последовательность, обоснованность. Далее пойдёт речь о влиянии каждого из этих законов на код.

Закон тождества

Закон тождества гласит: сущность должна оставаться сама собой в рамках одного контекста рассуждения. Соблюдение этого закона обеспечивает *определённость.*

Примером может быть использование оператора присваивания (=) вместо оператора равенства (==) в при проверке условия, что часто встречается у новичков. Для предотвращения этого в IDE Visual Studio 2022 существует официальное предупреждение, подчёркивающее данную ошибку зелёной волнистой линией.

Подмена понятий – одно из проявлений нарушения закона тождества в логике. Ещё одним ярким примером нарушения закона тождества является попытка сравнения знаковых и беззнаковых чисел: в языках программирования одна и та же битовая последовательность может иметь разные значения в зависимости от того, учтён ли знак. Обычно отрицательные числа переводятся компьютером в двоичные посредством «дополнительного кода». Это значит, что вначале биты числа инвертируются, а затем к ним добавляется единица [1].

Пример:

Запишем в двоичном виде число 37, деля его и последующие целые части частных столбиком на 2 и записывая остатки в обратном порядке.

– 37 : 2 = 18 (ост. 1);
– 18 : 2 = 9 (ост. 0);
– 9 : 2 = 4 (ост. 1);
– 4 : 2 = 2 (ост. 0);
– 2 : 2 = 1 (ост. 0);
– 1 : 2 = 0 (ост. 1).

Таким образом получаем остатки 1, 0, 1, 0, 0, 1. $37_2 = 100101$.

Если 37 – это переменная типа char – она занимает в памяти 1 байт, то есть имеет 8 битов под 1 или 0, а само число 37 записывается 6 цифрами. Все недостающие цифры заполняются нулями перед первой:

char num = 37; // 00100101.

Теперь получим из этого отрицательное число. Для этого нужно провести побитовую операцию отрицания, а затем добавить к результату число 1 (рис. 1).

```
~ 00100101
  11011010

+ 11011010
  00000001
  11011011
```

Рисунок 1 – Получение отрицательного значения от char num = 37 методом дополнительного кода

Знаковое число (signed), применяемое по умолчанию при отсутствии знакового спецификатора перед именем переменной, может принимать 256 значений ($2^8 = 256$ комбинаций) в диапазоне [-128; 127]. Первый бит такого числа является знаковым. Если он равен единице, то число отрицательное. Беззнаковые переменные в свою очередь имеют обязательный спецификатор перед их названием.

Такие числа всегда обладают неотрицательным значением, а это значит что unsigned char будет вмещать числа от 0 до 255. При этом первый бит числа будет определять не знак, а значение, как и все остальные биты числа. Проблемы начинаются как раз при наличии отрицательного знака. Конкретно нарушение закона тождества кроется в том, что одна и та же битовая последовательность (11011011) трактуется компьютером в одном и том же контексте (операция сравнения) как два совершенно разных значения (-37 и 219).

Закон непротиворечия классической логики гласит: два противоречащих суждения о предмете не могут быть одновременно истинными в одно и то же время и в одном и том же отношении. Этот закон помогает выявлять ложь, а его соблюдение обеспечивает *непротиворечивость.*

В программировании простейшие случаи этого закона соблюдаются всегда. Например, никакая переменная не может иметь одновременно два разных значения. При присваивании ей нового значения старое моментально теряется.

Закон исключённого третьего предполагает, что третьего не дано. В программировании же третье — это всегда ошибка, неучтённый случай или состояние неопределённости. Соблюдение этого закона обеспечивает логическую *последовательность.*

Закон исключенного третьего позволяет компьютеру «однозначно решить, куда переходить дальше». Его проявление происходит в стандартных случаях, например при использовании условного оператора. Оператор if-else имеет только два «состояния» – истина и ложь, остальные конструкции этого оператора можно представить в виде последовательности простых условных операторов.

Ещё одним примером могут служить указатели в языке Си. Два состояния, например, в языке Си могут иметь указатели – они либо могут указывать на какую-то переменную со своим значением, либо возвращать NULL (ложное состояние). Практическое применение это находит при работе с динамической памятью, когда функция malloc может возвращать NULL при ошибке выделения памяти [2]. В программировании бывают случаи, когда «ошибка» считается отдельным состоянием, вдобавок к «истине» и «лжи», например, при проверке пользовательского ввода.

Интересно следующее: концепция «кубитов», являющихся основой гипотетических квантовых компьютеров, основывается на том, что данные находятся «в суперпозиции» и один бит может являться не только 1 либо 0, но также 1 и 0 одновременно [3]. Это нарушает закон непротиворечия (два противоположных состояния объекта истинны в одно и то же время) и закон исключения третьего (наличие трёх состояний у, казалось бы, бинарного разряда). Для таких компьютеров существую отдельные правила квантовой логики.

Также при работе с SQL используется трёхзначная логика, к которой неприменим закон классической логики о исключенности третьего. Приходящие данные могут быть TRUE или FALSE (истина или ложь), при сравнении значений, но NULL в SQL обозначает «маркер потерянной информации». При сравнении чего-либо с NULL получается результат UNKNOWN, который не приписывается ни к TRUE, ни к FALSE, потому что имеет вероятностный характер [4].

Закон достаточного основания гласит: только такое суждение может быть признано истинным, которое имеет такие основания, из которых истинность следует с логической необходимостью, как следствие из причины. Соблюдение этого закона обеспечивает *обоснованность.*

Ни одна команда не должна выполняться без обусловленной на то причины. У каждой важной операции должно быть явное, проверенное условие, причём «границы достаточности» основания устанавливаются условием задачи или самим программистом: если программа не подразумевает деления на ноль, нужно чётко это обозначить через оператор assert. В противном случае допускается константа inf, равная бесконечности, получаемая по умолчанию при попытке деления на ноль.

Неявные основания для выполнения строк кода не нужно писать вручную: для выполнения вывода на экран «printf(“Hello world\n”);» требует для себя одного основания: выполнение программы дошло до текущей строки, функции записи в файл требуют основанием открытый файл и т. д. При попытке записать строку в неоткрытый файл программа прервётся с всплывающим окном об ошибке.

Необходимость достаточности оснований просматривается также и при непосредственно логическом осмыслении задачи, не поощряя избыточный код.

Таким образом, можно сделать вывод о том, что соблюдение основных технических принципов работы программы (определённость, непротиворечивость, последовательность, обоснованность) обусловлено четырьмя главными логическими законами, а логические ошибки происходят при их неправильном применении или пренебрежении ими.

Список использованных источников:

1. Назарова Н. Кодирование информации и её единицы измерения / С. Назарова // Наука и мировоззрение. – 2024. – №28. – С. 1-6.

2. Основы алгоритмизации и программирования. Язык Си : учебное пособие / М. П. Батура [и др.]. – Минск : БГУИР, 2007. – 240 с.

3. Универсальный квантовый компьютер – впереди времени и технологий / С. Килин // Наука и инновации. – 2023. – №8(246). С. 10-17.

4. Учебник языка SQL – Трехзначная логика и предложение Where. Режим доступа: https://sql-tutorial.ru/ru/book/second_phase_of_testing/typical_problems/three_valued_logic_and_where_clause – Дата доступа: 20.02.2026.