

**Формальные методы верификации программ**

На основе изучения материала лекций по дисциплине «Методы верификации и валидации характеристик программного обеспечения» требуется выполнить ниже перечисленное.

1. Докажите эквивалентность (или не эквивалентность) приведенных ниже программ P,Q и S, которые должны реализовать алгоритм Евклида поиска НОД (наибольшего общего делителя):

Программа P

x := a;

y := b;

while x ≠ y do

if x > y then

x := x – y

else

y := y – x

end

end;

c := x

Программа Q

x := a;

y := b;

while y ≠ 0 do

z := x;

x := y;

y := z % y

end;

c := x

Программа S

x := a;

y := b;

while x ≠ 0 ꓥ z ≠ 0 do

if x > y then

x := x % y

else

y := y % x

end

3

end;

c := x + y

1. Смоделируйте структурой Крипке систему управления стиральной машиной. Машина имеет бак для белья, через который также подаются моющие средства, клапаны для забора и слива воды, датчик наличия воды, мотор, нагревающий элемент, таймер и панель управления с кнопками «Пуск» и «Останов». Предполагается следующий сценарий использования машины: открыть дверцу бака, поместить белье и моющие средства в бак, закрыть дверцу, нажать на кнопку «Пуск». Машина открывает клапан для забора воды, набирает воду и закрывает клапан; подогревает воду; заводит таймер и запускает мотор, вращающий бак; при срабатывании таймера сливает воду. Дверца бака блокируется, пока в баке есть вода. Пользователь имеет возможность в любой момент нажать на кнопку «Останов», чтобы принудительно остановить стирку белья и слить воду.
2. Определите как можно более полный набор требований к системе управления стиральной машиной, описанной в предыдущем задании. Выразите эти требования в логике LTL.
3. Определите программный контракт (пред- и постусловие) для программы, вычисляющей с заданной точностью квадратный корень числа (x — входное число, y — результат, ε – точность).
4. Определите программный контракт (пред- и постусловие) для программыn сортировки числового массива (x — входной массив, y — результатnсортировки, N — размер массива).
5. Докажите завершимость приведенной ниже программы целочисленного деления DIV

Программа DIV:

Программа DIV имеет две входные целочисленные переменные a и b, и две выходные целочисленные переменные, q и r. Для программы задано

предусловие

ϕ≡(a≥b) ꓥ (b>0)

уточняющее, что программа определена только для неотрицательных

значений a и положительных значений b, и постусловие

ϕ≡(a=q·b+r) ꓥ (0≤r<b)

утверждающее, что программа осуществляет деление a на b и сохраняет

частное от деления в переменную q ,а остаток — в переменную r.

{(a ≥ 0)ꓥ(b > 0)}

q := 0;

r := a;

while r ≥ b do

q := q + 1;

r := r - b

end

{(a = q∙b + r) ꓥ (0 ≤ r < b)

**1. Доказательство эквивалентности программ P, Q и S**

Для доказательства эквивалентности программ, реализующих алгоритм Евклида поиска НОД, следует провести формальный анализ их поведения.

**Программа P:**

x := a;

y := b;

while x ≠ y do

if x > y then

x := x – y

else

y := y – x

end

end;

c := x

Эта программа уменьшает большее из чисел, вычитая меньшее, до тех пор, пока они не станут равными. Итоговое значение x (или y) является НОД.

**Программа Q:**

x := a;

y := b;

while y ≠ 0 do

z := x;

x := y;

y := z % y

end;

c := x

Это реализация алгоритма Евклида с использованием операции взятия остатка. Программа повторяет шаги, пока y не станет нулем, и значение x в этот момент будет НОД.

**Программа S:**

x := a;

y := b;

while x ≠ 0 ꓥ y ≠ 0 do

if x > y then

x := x % y

else

y := y % x

end

end;

c := x + y

Эта программа также использует операцию взятия остатка, но условия окончания немного отличаются. Итоговое значение x + y дает НОД.

**Доказательство эквивалентности**

Для доказательства эквивалентности нужно показать, что для всех возможных значений a и b, программы дают одинаковый результат. Для этого можно построить инварианты циклов и показать, что они сохраняются при каждой итерации.

**2. Моделирование системы управления стиральной машиной структурой Крипке**

Структура Крипке состоит из множества состояний, множества переходов и множества истинных высказываний для каждого состояния. Для стиральной машины это могут быть следующие состояния:

* **Idle** — ожидание команды «Пуск».
* **Filling** — забор воды.
* **Heating** — нагрев воды.
* **Washing** — стирка (вращение бака).
* **Draining** — слив воды.
* **Stopped** — остановка.

Переходы между этими состояниями соответствуют событиям, таким как «пуск», «набор воды завершен», «нагрев завершен», «таймер стирки завершен» и так далее.

**3. Определение требований к системе управления стиральной машиной в логике LTL**

Логика LTL (Linear Temporal Logic) позволяет выражать требования к поведению системы во времени. Примеры требований к стиральной машине:

* **Дверца должна быть заблокирована, пока есть вода в баке**: G(Water\_Level>0→Door\_Locked)
* **Если нажата кнопка «Пуск», то в конечном итоге начнется забор воды**: G(Start\_Pressed→F(Filling))
* **После завершения стирки всегда происходит слив воды**: G(Washing\_Done→F(Draining))

**4. Программный контракт для программы, вычисляющей квадратный корень**

Для программы, вычисляющей квадратный корень с точностью ε, можно определить следующие контракты:

* **Предусловие**:
* **Постусловие**:

**5. Программный контракт для программы сортировки массива**

Контракты для сортировки массива:

* **Предусловие**: (размер массива больше нуля).
* **Постусловие**:
  + Массив отсортирован:
  + Массив содержит те же элементы, что и исходный.

**6. Доказательство завершимости программы DIV**

Программа DIV:

q := 0;

r := a;

while r ≥ b do

q := q + 1;

r := r - b

end

Чтобы доказать завершимость, необходимо показать, что цикл **while** в какой-то момент завершится. В данной программе это гарантируется, так как значение переменной r уменьшается на каждом шаге на величину , и поскольку , цикл обязательно завершится, когда .