Д.В. Буздалов, Е.В. Корныхин, А.А. Панфёров,
А.К. Петренко, А.В. Хорошилов

ПРАКТИКУМ ПО ДЕДУКТИВНОЙ ВЕРИФИКАЦИИ ПРОГРАММ

Москва
2014

УДК 004.415.5
ББК 32.973-18

Печатается по решению Редакционно-издательского Совета факультета вычислительной математики и кибернетики МГУ имени М.В. Ломоносова

Рецензенты:
И.В. Машечкин, профессор, д.ф.-м.н.
С.Ю. Соловьев, профессор, д.ф.-м.н.

Д.В. Буздалов, Е.В. Корныхин, А.А, Панфёров, А.К. Петренко, А.В. Хорошилов.
Практикум по дедуктивной верификации программ: учебно-методическое пособие. — М. Издательский отдел факультета ВМК МГУ имени М.В. Ломоносова (лицензия ИД №05899 от 24.09.2001); МАКС Пресс, 2014. — 100 с.

ISBN 978-5-89407-531-0
ISBN

Данное пособие посвящено основным понятиям дедуктивной верификации последовательных программ. Дедуктивная верификация позволяет формально обосновать, что программа выполняет формализованные требования путем построения набора утверждений и доказательства их истинности. В качестве методов дедуктивной верификации в пособии рассматриваются методы Флойда (или Хоара-Флойда). Пособие рекомендуется студентам, осваивающим методы формальной верификации программ, а также аспирантам и преподавателям.

УДК 004.415.5
ББК 32.973-18

This textbook is devoted to foundations of deductive verification of software. Deductive verification is aimed to prove that a program under verification consists with function requirements by proving some theorems about behavior of this program. R. Floyd's methods are using as an example of deductive verification methods. The textbook is aimed at students learning the formal verification of software and at postgraduate students and lecturers.

Ключевые слова: формальная спецификация, формальная верификация, дедуктивная верификация, методы Флойда, формальные методы разработки программ.
Keywords: formal specification, formal verification, deductive verification, Floyd's methods, formal methods of software development.

ISBN 978-5-89407-531-0
ISBN

Оглавление

1 Методы верификации Флойда

Данное пособие является введением в методы *дедуктивной верификации* последовательных программ (или, как ее еще называют, *аналитической верификации*).

Целью любой верификации программы (и, в частности, дедуктивной верификации) является установление соответствия программы ее требованиям. Дедуктивная верификация устанавливает это соответствие в виде логического вывода утверждения о том, что программа соответствует требованиям. При этом доказывается соответствие программы требованиям на всех входах программы. В отличие от этого, например, в таком методе верификации как тестирование программа проверяется лишь при некоторых входах программы (тестах).

Далее, без ограничения общности, под «методами верификации» будут пониматься «дедуктивные методы верификации».

Поскольку дедуктивная верификация основана на использовании логических (математических) заключений, она должна оперировать с математическими («формальными») моделями как программ, требований, так и с формально определенным отношением их соответствия.

Мы начнем рассмотрение методов верификации на простейших моделях последовательных программ. Первая модель будет соответствовать программам, написанным на структурном языке программирования

- без массивов;
- без использования адресной арифметики;
- без рекурсии (вызова подпрограмм);

- без взаимодействия с окружением (например, посредством операторов ввода-вывода).

Методы верификации рекурсивных программ не входят в материал лекций.

Программы, взаимодействующие с окружением, относятся к параллельным (или, как их еще называют, реактивным) программам и мы их тоже пока не рассматриваем.

Мы рассмотрим методы верификации последовательных программ, которые называют *методами Флойда* (или *методами Флойда-Хоара*). Корни этих методов уходят к Алану Тьюрингу, который в своей лекции Лондонскому математическому обществу в 1947 впервые озвучил идею индуктивных утверждений. До полноценных методов верификации эта идея была доведена независимо Робертом Флойдом [1] и Тони Хоаром [2] в конце 60-х годов 20 века. В данном пособии изложение этих методов следует их трактовке в монографиях [5] и [6].

1.1 Математическая модель программы

Описание математической модели мы начнем с нескольких вспомогательных определений.

Переменные программы

Каждая программа работает с конечным числом переменных. Переменные разделяются на три типа: *входные*, *промежуточные* и *выходные*. Вектора этих переменных мы будем обозначать $\mathbf{x} = (x_1, x_2, ..., x_a)$, $\mathbf{y} = (y_1, y_2, ..., y_b)$ и $\mathbf{z} = (z_1, z_2, ..., z_c)$ соответственно. Входные переменные содержат исходные входные значение и никогда не меняются во время работы программы. Промежуточные

переменные используются для хранения промежуточных результатов в процессе вычисления. Выходные переменные содержат значения, вычисляемые данной программой. Далее, если не будет сказано специально, входные переменные будем обозначать как x_1, x_2, ..., промежуточные как y_1, y_2, ..., выходные как z_1, z_2, ...

Каждая переменная v может принимать значения из некоторого множества D_v, которое называется *доменом* переменной. Также, мы будем выделять три непустых домена:

- *входной домен* $D_x = Dx_1 \times Dx_2 \times ... \times Dx_a$
- *домен программы* $D_y = Dy_1 \times Dy_2 \times ... \times Dy_b$
- *выходной домен* $D_z = Dz_1 \times Dz_2 \times ... \times Dz_c$

Множество значений всех возможных переменных образует *универсальный домен* D. Это значит, что для любой переменной v выполнено соотношение: $D_v \subseteq D$. Кроме того, мы выделим два специальных значения: T (истина) и F (ложь). Функции, принимающие значения только из множества {T, F}, мы будем называть *предикатами* на области определения функции. Напомним, что функцией из некоторого множества X в некоторое множество Y называется отображение, ставящее каждому элементу множества X некоторый элемент множества Y единственным образом.

Расширенным доменом D_v^+ переменной v будем называть домен этой переменной, дополненный специальным значением ω, которое не входит в универсальный домен: $D_v^+ = D_v \cup \{\omega\}$.

Операторы программы

Мы будем рассматривать 5 видов операторов программы над данным множеством переменных. Некоторым из них приписана функция:

1. *Начальный оператор* START: $\mathbf{y} \leftarrow f(\mathbf{x})$. Здесь f является функцией $D_{\mathbf{x}} \rightarrow D_{\mathbf{y}}^{+}$, инициализирующей промежуточные переменные программы на основе значений ее входных переменных.
2. *Оператор присваивания* ASSIGN: $\mathbf{y} \leftarrow g(\mathbf{x}, \mathbf{y})$. Здесь g является функцией $D_{\mathbf{x}} \times D_{\mathbf{y}} \rightarrow D_{\mathbf{y}}^{+}$, вычисляющей новые значения промежуточных переменных.
3. *Условный оператор* TEST: t $(\mathbf{x}, \mathbf{y})$. Здесь t является предикатом на множестве значений входных и промежуточных переменных программы.
4. *Оператор соединения* JOIN.
5. *Оператор завершения* HALT: $\mathbf{z} \leftarrow h(\mathbf{x}, \mathbf{y})$. Здесь h является функцией $D_{\mathbf{x}} \times D_{\mathbf{y}} \rightarrow D_{\mathbf{z}}^{+}$, устанавливающей значения выходных переменных программы.

Графическое представление операторов показано на рисунке 1.

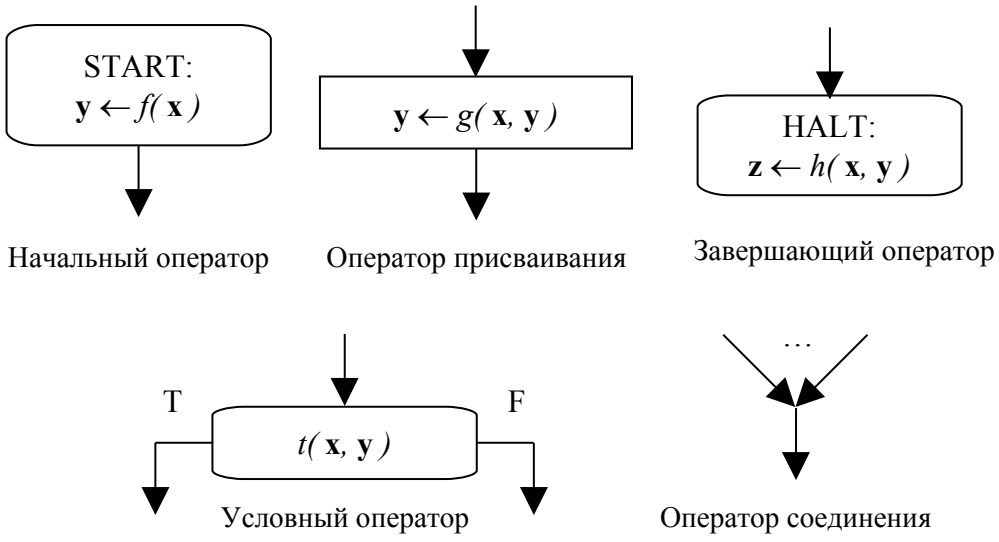


Рисунок 1. Графическое представление операторов блок-схемы

В программу могут входить несколько операторов одного и того же типа, помеченных одной и той же функцией. Поэтому для обеспечения уникальности одинаковых операторов в рамках одной программы, мы будем помечать каждый оператор уникальной меткой ℓ_i. Таким образом, каждый оператор программы состоит из *метки оператора* и *тела оператора*, принадлежащего к одному из пяти возможных типов. Множество меток всех операторов программы P будет обозначаться как Λ_P.

Блок-схемы

В качестве модели программы мы будем использовать блок-схемы. *Блок-схемой* называется тройка (V, N, E), где

V − конечное множество переменных программы,

N − конечное множество операторов блок-схемы,

$E \subseteq N \times \{\mathrm{T}, \mathrm{F}, \varepsilon\} \times N$ − конечное множество связок блок-схемы, помеченных символами T, F или ε.

Заметим, что блок-схема соответствует ориентированному графу, вершинами которого являются операторы программы, а ребрами − ее связки. При этом все ребра помечены одним из трех символов.

Корректно-определенной блок-схемой мы будем называть блок-схему удовлетворяющую следующим требованиям:

1. В блок-схеме присутствует ровно один начальный оператор и не менее одного завершающего оператора.

2. Любой оператор находится на ориентированном пути от начального оператора к некоторому завершающему оператору.
3. Число связок, выходящих из каждого оператора, и пометки этих связок соответствуют типу оператора:
    a. Из начального оператора выходит ровно 1 дуга, помеченная символом ε.
    b. Из оператора присваивания выходит ровно 1 дуга, помеченная символом ε.
    c. Из условного оператора выходит ровно 2 дуги, причем одна из них помечена символом T, а другая − символом F.
    d. Из оператора соединения выходит ровно 1 дуга, помеченная символом ε.
    e. Из завершающего оператора не выходит ни одной дуги.
4. Число связок, входящих в каждый оператор, соответствует его типу:
    a. В начальный оператор не входит ни одна дуга.
    b. В оператор присваивания, условный и завершающий оператор входит ровно одна дуга.
    c. В оператор соединения входит не менее одной дуги.

Заметим, что для каждого оператора n и символа s в корректно-определенной блок-схеме (V, N, E) существует не более одного оператора n', что $(n, s, n') \in E$. Такой оператор n' (если он существует) мы будем называть *последователем* оператора n по пометке s и обозначать как $succ(n, s)$.

Далее мы будем рассматривать только корректно-определенные блок-схемы, и, поэтому, слово «корректно-определенная» будет опускаться.

В графическом представлении блок-схем мы будем опускать некоторые детали. Например, ребра, помеченные символом ε, будут изображаться без соответствующей метки. Как правило, будут опускаться метки операторов блок-схемы, а в операторах соединения не все входящие ребра будут изображаться со стрелками на конце.

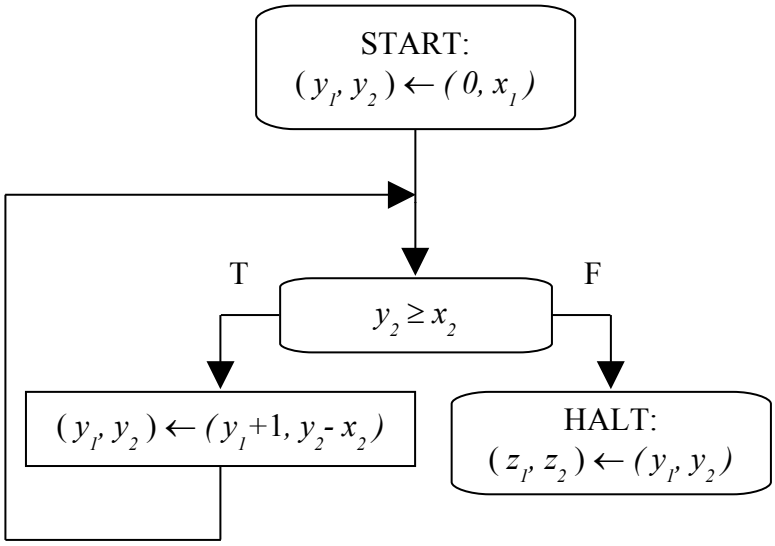


Рисунок 2. Блок-схема программы целочисленного деления

Для определения функций будет использоваться следующая нотация:

$$(y_1, y_2, ..., y_b) \leftarrow (f_1(\mathbf{x}, \mathbf{y}), f_2(\mathbf{x}, \mathbf{y}), ..., f_b(\mathbf{x}, \mathbf{y}))$$

Пример графического представления блок-схем можно увидеть на рисунке 2, где представлена блок-схема программы целочисленного деления. В этом примере множество переменных $V = \{x_1, x_2, y_1, y_2, z_1, z_2\}$ состоит из двух входных, двух промежуточных и двух выходных переменных. Доменом всех переменных является множество целых чисел.

Далее без ограничения общности под «программами» понимаются их блок-схемы.

Семантика блок-схем

Конфигурацией программы P будем называть пару (ℓ, σ), где

$\ell \in \Lambda_P$ – метка текущего оператора программы,

$\sigma = (d_1, d_2, ..., d_{a+b}) \in D_x^+ \times D_y^+$ – вектор значений входных и промежуточных переменных программы.

Если $\sigma \in D_x^+ \times D_y^+$ – вектор значений входных и промежуточных переменных программы, а функция $f : D_x^+ \times D_y^+ \to D_y^+$ вычисляет новые значения переменных **y**, то вектор значений входных и промежуточных переменных программы, полученный путем замены в σ значений переменных **y** на $f(\sigma)$, мы будем обозначать как $\sigma[\mathbf{y} \leftarrow f(\mathbf{x}, \mathbf{y})]$.

Конечная или бесконечная последовательность конфигураций $\{ C_i \mid i = 1, ..., n, ... \}$ программы P называется *вычислением*, если

1. Метка первой конфигурации программы является меткой начального оператора.
2. Значения всех входных переменных программы являются определенными ($\neq \omega$) и неизменными во всех конфигурациях вычисления.
3. Значения промежуточных переменных в первой конфигурации равны ω (*не определены*).
4. Если метка ℓ_i текущего оператора конфигурации C_i является меткой начального оператора START: $\mathbf{y} \leftarrow f(\mathbf{x})$, то следующая конфигурация C_{i+1} состоит из метки оператора $succ(n_i, \varepsilon)$ и вектора значений переменных $\sigma_{i+1} = \sigma_i[\mathbf{y} \leftarrow f(\mathbf{x})]$.

5. Если метка ℓ_i текущего оператора конфигурации C_i является меткой оператора присваивания ASSIGN: $\mathbf{y} \leftarrow g(\mathbf{x}, \mathbf{y})$, то следующая конфигурация C_{i+1} состоит из метки оператора $succ(n_i, \varepsilon)$ и вектора значений переменных $\sigma_{i+1} = \sigma_i[\mathbf{y} \leftarrow g(\mathbf{x}, \mathbf{y})]$.
6. Если метка ℓ_i текущего оператора конфигурации C_i является меткой условного оператора TEST: $t(\mathbf{x}, \mathbf{y})$ и предикат $t(\mathbf{x}, \mathbf{y})$ при значениях переменных σ_i принимает значение T, то следующая конфигурация C_{i+1} состоит из метки оператора $succ(n_i, T)$ и вектора значений переменных $\sigma_{i+1} = \sigma_i$.
7. Если метка ℓ_i текущего оператора конфигурации C_i является меткой условного оператора TEST: $t(\mathbf{x}, \mathbf{y})$ и предикат $t(\mathbf{x}, \mathbf{y})$ при значениях переменных σ_i принимает значение F, то следующая конфигурация C_{i+1} состоит из метки оператора $succ(n_i, F)$ и вектора значений переменных $\sigma_{i+1} = \sigma_i$.
8. Если метка ℓ_i текущего оператора конфигурации C_i является меткой оператора соединения JOIN, то следующая конфигурация C_{i+1} состоит из метки оператора $succ(n_i, \varepsilon)$ и вектора значений переменных $\sigma_{i+1} = \sigma_i$.
9. Если метка ℓ_i текущего оператора конфигурации C_i является меткой завершающего оператора HALT: $\mathbf{z} \leftarrow h(\mathbf{x}, \mathbf{y})$, то C_i является последней конфигурацией вычисления.
10. Если в конфигурации C_{i+1} значение какой-либо промежуточной переменной равно ω, то это последняя конфигурация вычисления.

Тем самым, возможны три вида вычислений:

1. конечные последовательности конфигураций, в последней конфигурации никакие значения переменных не равны ω;
2. конечные последовательности конфигураций, в последней конфигурации значения некоторых переменных равны ω;
3. бесконечные последовательности конфигураций.

Лемма 1. *Для каждой блок-схемы P и вектора значений ее входных переменных* ***x*** *существует единственное вычисление, в первой конфигурации которого значения входных переменных равны* ***x****.*

Каждой блок-схеме P мы поставим в соответствие функцию M[P] из входного домена блок-схемы в выходной домен, расширенный специальным значением ω (M[P]: $D_x \rightarrow D_z^+$). Если вычисление блок-схемы P на векторе входных переменных **x** является конечным, в последней конфигурации которого нет ω, и завершается на операторе HALT, то функция M[P] **(x)** принимает значение $h(\mathbf{x}, \mathbf{y}_n)$, где h − функция завершающего оператора последней конфигурации вычисления, а $\mathbf{y}_n$ − вектор значений промежуточных переменных из последней конфигурации вычисления. В противном случае функция M[P] **(x)** принимает значение ω.

Задачи и упражнения

В данных задачах предполагается, что доменом всех входных и выходных переменных является множество целых чисел, а функции, приписанные операторам, должны задаваться арифметическими формулами над операциями сложения, вычитания и умножения и операциями сравнения.

1.1.1 Перечислите все операторы в блок-схеме целочисленного деления, приведенной на рисунке 2.

1.1.2 Приведите блок-схему с единственной входной и единственной выходной переменной, вычисляющую квадрат входной переменной.

1.1.3 Приведите блок-схему с единственной входной и единственной выходной переменной, вычисляющую факториал значения входной переменной, если оно неотрицательно. Можно предполагать, что отрицательные значения входной переменной не будут использоваться в вычислениях этой блок-схемы.

1.1.4 Приведите блок-схему с единственной входной и единственной выходной переменной, вычисляющую $\left[\frac{x+1}{x}\right]$, где x — это значение входной переменной, для положительных значений x. Можно предполагать, что неположительные значения x не будут использоваться в вычислениях этой блок-схемы.

1.1.5 Приведите блок-схему с единственной входной и единственной выходной переменной, вычисляющую $\frac{x\cdot(x+1)}{2}$, где x — это значение входной переменной, для неотрицательных значений x. Можно предполагать, что отрицательные значения x не будут использоваться в вычислениях этой блок-схемы.

1.1.6 Приведите блок-схему с единственной входной переменной, не завершающуюся на бесконечном числе ее значений.

1.1.7 Приведите блок-схему с двумя входными переменными x_1 и x_2 и единственной выходной переменной, которая вычисляет количество простых чисел между x_1 и x_2.

1.1.8 Ответьте на вопрос, зацикливается ли блок-схема, изображенная на рисунке 2. Если да, то приведите всевозможные пары значений входных переменных, при которых она зацикливается. Если нет, обоснуйте свой ответ.

1.1.9 Приведите пример блок-схемы каждого из следующих видов, если они существуют:

a) в которых количество циклов меньше числа операторов соединения;
b) в которых количество циклов равно числу операторов соединения;
c) в которых количество циклов больше числа операторов соединения.

1.2 Математическая модель требований

Математическую модель требований к верифицируемой программе мы будем называть *спецификацией программы*. Семантика спецификации состоит в формальном описании требований к поведению программы.

Требований к поведению программ может быть огромное множество, но в данном разделе мы будем рассматривать только *требования к функциональности программ*. Под требованиями к функциональности понимаются ограничения на результат вычисления программы в зависимости от значений ее входных данных.

Входной и выходной предикаты

Спецификацией Ф программы над переменными V мы будем называть два предиката:

входной предикат φ: $D_x \to \{ T, F \}$

выходной предикат ψ: $D_x \times D_z \to \{ T, F \}$

Выходной предикат (или *постусловие*) определяет, какие значения выходных переменных программы являются допустимыми (правильными) относительно значений входных переменных. А *входной предикат* (или *предусловие*) определяет, при каких значениях входных переменных требуется выполнение ограничений, описанных в выходном предикате.

Задачи и упражнения

В двух первых задачах предполагается, что доменом всех входных и выходных переменных является множество целых чисел, а все используемые в качестве предикатов функции должны быть формально определены. Можно считать априорным, что уже формально определены следующие арифметические операции над целыми числами: сложение, вычитание, умножение — а также операции сравнения (равно, неравно, больше, меньше и т. п.).

1.2.1 Опишите словами требования, описанные каждой из следующих спецификаций:

a) $\varphi(x) = (x > 0) \quad \psi(x, z) = (x \cdot x = z)$

b) $\varphi(x) = (x > 0) \quad \psi(x, z) = (0 \le z < x)$

c) $\varphi(x_1, x_2) = (x_1 > x_2 + 1) \quad \psi(x_1, x_2, z) = (x_1 > z > x_2)$

d) $\varphi(x) = (x > 0) \quad \psi(x, z) = (x < z < 0)$

1.2.2 Дайте формальную спецификацию каждого из следующих требований:

a) программа должна вычислять модуль данного ей числа.

b) программа для двух данных целых чисел должна вычислять максимальное из них.

c) программа должна для двух данных ей положительных целых чисел вычислять их сумму, а для двух данных ей отрицательных целых чисел — произведение.

d) программа должна для любого неотрицательного целого числа вычислять ближайшее снизу приближение к квадратному корню из него.

e) программа должна для любого положительного целого числа большего единицы вычислять его максимальный простой делитель.

f) программа должна для двух заданных положительных целых чисел вычислять количество простых чисел между ними.

1.2.3 Обоснованно ответьте на следующий вопрос. Является ли для любых спецификаций вида (α, β) над переменными (A, Y^*, B) и (β, γ) над переменными (B, Y^{**}, C) пара утверждений (α, γ) спецификацией над переменными (A, Y^{***}, C) при некоторых Y^*, Y^{**}, Y^{***}? Зависит ли ответ от выбора указанных множеств переменных?

1.3 *Задача верификации*

Частичная и полная корректность программ

Пусть программа задана своей моделью в виде блок-схемы P, а ее спецификация Φ − предикатами φ и ψ. Мы будем говорить, что

- программа P *частично корректна* относительно φ и ψ, если для любого вектора значений входных переменных **x**, такого что φ(**x**) = T и M[P](**x**) ≠ ω, выполнено ограничение ψ(**x**, M[P](**x**))=T. Частичную корректность программы P относительно φ и ψ мы будем обозначать {φ}P{ψ}.
- программа P *полностью корректна* относительно φ и ψ, если для любого вектора значений входных переменных **x**, такого что φ(**x**) = T, выполнены ограничения M[P] (**x**) ≠ ω и ψ(**x**, M[P](**x**)) = T. Полную корректность программы P относительно φ и ψ мы будем обозначать ⟨φ⟩P⟨ψ⟩.

Заметим, что полная корректность программы P относительно входного предиката φ и выходного предиката T эквивалентна тому, что программа P завершается всегда, когда вектор значений входных переменных удовлетворяет φ. В этом случае мы будем говорить, что P *завершается* на φ. Напомним, что это означает, что соответствующее вычисление заканчивается в одном из операторов HALT со значениями всех переменных не равными ω.

Лемма 2. *Пусть даны программа* P *и спецификация* Φ = (φ, ψ). *В этом случае* ⟨φ⟩P⟨ψ⟩ *тогда и только тогда, когда* {φ}P{ψ} *и* ⟨φ⟩P⟨T⟩.

Исходя из данной леммы, для доказательства полной корректности программы необходимо и достаточно доказать ее частичную корректность и завершаемость.

Из определения корректности также следует, что и частичная, и полная корректность сохраняется при замене входного предиката на более сильный и выходного на более слабый. Таким образом, верна следующая лемма:

Лемма 3. *Пусть дана программа* P. *Пусть предикаты* φ, φ', ψ *и* ψ' *таковы, что формулы* $\varphi' \to \varphi$ *и* $\psi \to \psi'$ *истинны. Тогда*

- *из* $\{\varphi\}P\{\psi\}$ *следует* $\{\varphi'\}P\{\psi\}$ *и* $\{\varphi\}P\{\psi'\}$,
- *из* $\langle\varphi\rangle P\langle\psi\rangle$ *следует* $\langle\varphi'\rangle P\langle\psi\rangle$ *и* $\langle\varphi\rangle P\langle\psi'\rangle$.

Следующая лемма позволяет по нескольким утверждениям о частичной и полной корректности получать новые утверждения:

Лемма 4. *Пусть дана программа* P. *Тогда для любых предикатов* φ, ψ_1 *и* ψ_2 *выполнены следующие утверждения:*

- *из* $\{\varphi\}P\{\psi_1\}$ *и* $\{\varphi\}P\{\psi_2\}$ *следует* $\{\varphi\}P\{\psi_1 \wedge \psi_2\}$,
- *из* $\langle\varphi\rangle P\langle\psi_1\rangle$ *и* $\langle\varphi\rangle P\langle\psi_2\rangle$ *следует* $\langle\varphi\rangle P\langle\psi_1 \wedge \psi_2\rangle$.

Верификация программы целочисленного деления

В дальнейшем мы сформулируем методы доказательства полной корректности программ в общем случае, но сначала обратимся к примеру.

Для этого вернемся к программе целочисленного деления, блок-схема которой представлена на рисунке 2 (далее она будет обозначаться как P_{div}). Мы докажем ее полную корректность относительно спецификации, заданной следующими предикатами:

$\varphi \equiv (x_1 \geq 0) \wedge (x_2 > 0)$

$\psi \equiv (x_1 = z_1 x_2 + z_2) \wedge (0 \leq z_2 < x_2)$

Входной предикат спецификации утверждает, что нас будет интересовать поведение программы только на неотрицательных значениях переменной x_1 и положительных значениях переменной x_2. Выходной предикат определяет, что значения выходных переменных программы должно удовлетворять определению целочисленного деления с остатком.

Доказательство полной корректности будет разбито на два этапа. Сначала мы докажем, что программа является частично корректной относительно входного предиката $\varphi_0 \equiv (x_1 \geq 0) \wedge (x_2 \geq 0)$ и выходного предиката ψ. А затем мы докажем завершаемость программы на φ. Из этого, по леммам 2 и 3, будет следовать требуемое утверждение.

Заметим, что входной предикат φ_0, используемый при доказательстве частичной корректности, является более слабым, чем φ. Это связано с тем, что при значении входной переменной x_2 равном 0, программа является частично корректной, но не завершается.

Частичная корректность. Поставим в соответствие начальному оператору блок-схемы входной предикат φ_0, завершающему оператору − выходной предикат ψ, а ребру между оператором соединения и условным оператором − промежуточный предикат p, задаваемый формулой $p(x_1, x_2, y_1, y_2) \equiv (x_1 = y_1x_2 + y_2) \wedge (y_2 \geq 0)$. Это ребро на рисунке 3 обозначено буквой B.

1. Рассмотрим путь от начального оператора программы до ребра B. После выполнения начального оператора переменные принимают следующие значения:

x_1	x_1
x_2	x_2
y_1	0
y_2	x_1

Таким образом, $p(x_1, x_2, y_1, y_2) \equiv (x_1 = y_1x_2 + y_2) \wedge (y_2 \geq 0) \equiv (x_1 = 0 \cdot x_2 + x_1) \wedge (x_1 \geq 0) \equiv (x_1 \geq 0)$. А последнее неравенство является истинным в предположении, что выполнено предусловие программы $\varphi_0 \equiv (x_1 \geq 0) \wedge (x_2 \geq 0)$. То есть: $\varphi_0 \Rightarrow (x_1 \geq 0)$.

2. Предположим, что предикат p истинен в точке B и рассмотрим путь B-D-B.

После выполнения условного оператора значения переменных не изменяются, то есть в точке D предикат p также будет истинен, а так как точка D лежит на ребре, помеченном символом T, то в точке D будет истинно следующее утверждение:

$(x_1 = y_1x_2 + y_2) \wedge (y_2 \geq 0) \wedge (y_2 \geq x_2)$

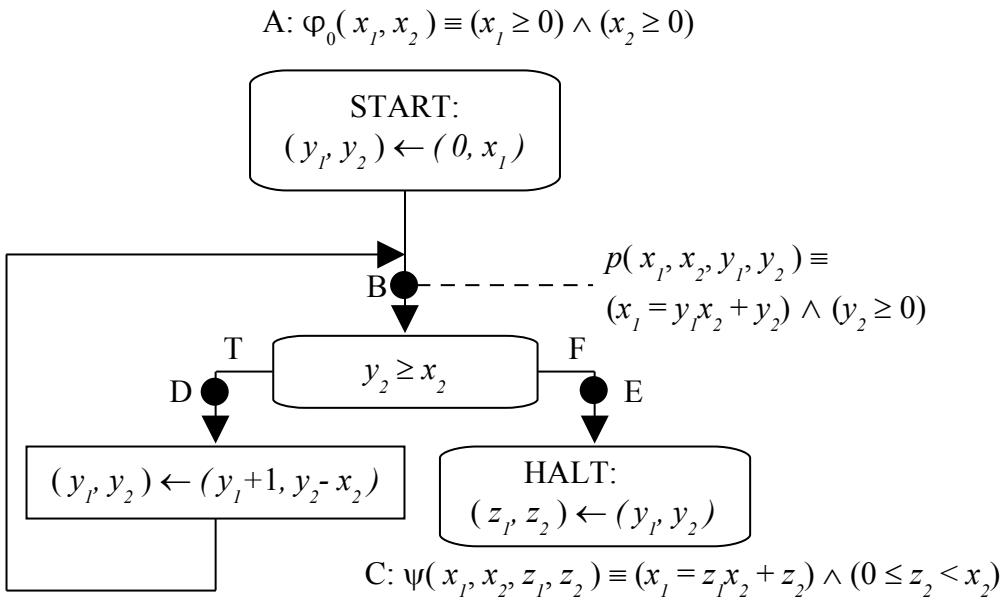


Рисунок 3. Блок-схема целочисленного деления и её предикаты

Докажем, что после выполнения последующего оператора присваивания предикат p также будет истинен:

D: $(x_1 = y_1x_2 + y_2) \wedge (y_2 \geq 0) \wedge (y_2 \geq x_2)$

B: $p(x_1, x_2, y_1 + 1, y_2 - x_2) \equiv (x_1 = (y_1+1)\cdot x_2 + y_2 - x_2) \wedge (y_2 - x_2 \geq 0)$

поэтому нужно показать, что истинно следующее утверждение:

$(x_1 = y_1x_2 + y_2) \wedge (y_2 \geq 0) \wedge (y_2 \geq x_2) \Rightarrow (x_1 = (y_1+1)\cdot x_2 + y_2 - x_2) \wedge (y_2 - x_2 \geq 0)$

$(x_1 = y_1x_2 + y_2) \wedge (y_2 \geq 0) \wedge (y_2 \geq x_2) \Rightarrow (x_1 = y_1\cdot x_2 + y_2) \wedge (y_2 - x_2 \geq 0)$

T

3. И в завершении рассмотрим последний путь: от точки B до завершающего оператора.

Предположим, что в точке B истинен предикат p. Тогда при попадании в точку E будет истинно следующее утверждение:

$(x_1 = y_1x_2 + y_2) \wedge (y_2 \geq 0) \wedge (y_2 < x_2)$

Докажем, что после завершающего оператора будет истинен выходной предикат ψ:

E: $(x_1 = y_1x_2 + y_2) \wedge (y_2 \geq 0) \wedge (y_2 < x_2)$

C: $\psi(x_1, x_2, y_1, y_2) \equiv (x_1 = y_1x_2 + y_2) \wedge (0 \leq y_2 < x_2)$

для этого необходимо показать, что истинно следующее утверждение:

$(x_1 = y_1x_2 + y_2) \wedge (y_2 \geq 0) \wedge (y_2 < x_2) \Rightarrow (x_1 = y_1x_2 + y_2) \wedge (0 \leq y_2 < x_2)$

T

Из рассмотренных свойств программы следует, что для любого конечного вычисления программы целочисленного деления при значениях входных переменных, удовлетворяющих предусловию, значения выходных переменных будут удовлетворять постусловию. Или, другими словами, что программа целочисленного деления яв-

ляется частично корректной относительно спецификации $\Phi_0 = (\varphi_0, \psi)$.

Завершаемость. До сих пор мы доказали корректность программы только условно. Мы доказали, что если программа завершается, то ее результат удовлетворяет предъявляемым к нему требованиям. Теперь докажем, что программа целочисленного деления действительно завершается при значениях входных переменных, удовлетворяющих входному предикату φ.

Во время доказательства частичной корректности программы мы доказали, что если значения входных переменных удовлетворяют предикату φ_0, то при всяком прохождении точки B значения входных и промежуточных переменных будут удовлетворять следующему условию: $(x_1 = y_1x_2 + y_2) \wedge (y_2 \geq 0)$. Из этого следует, что если значения входных переменных удовлетворяют более сильному предикату φ, то при всяком прохождении точки B значение промежуточной переменной y_2 будет неотрицательным. С другой стороны, значения входных переменных не изменяются в ходе выполнения программы, поэтому предикат φ является истинным в любой промежуточной точке. Отсюда следует, что в точке B будет выполнено следующее условие: $(y_2 \geq 0) \wedge (x_2 > 0)$.

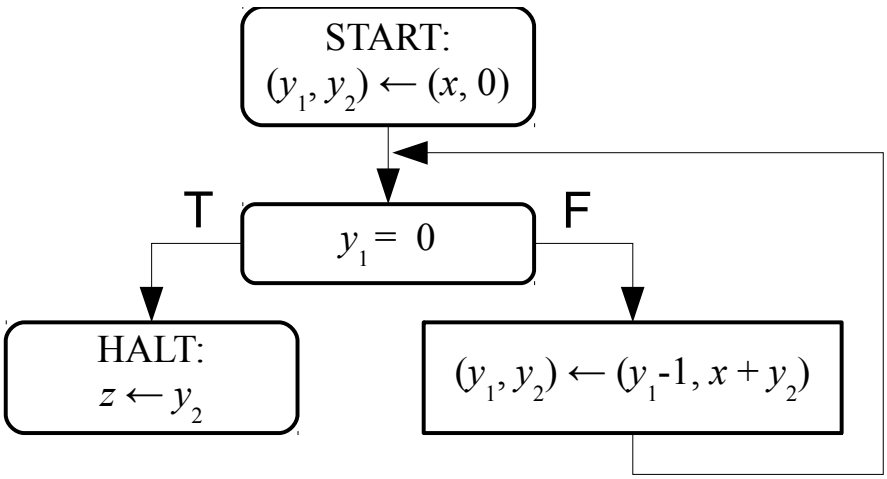
Рассмотрим цикл B-D-B. После прохождения этого пути значение переменной y_2 уменьшится на положительную величину (x_2). С другой стороны, значение переменной y_2 останется неотрицательным. И так как не существует бесконечной убывающей последовательности неотрицательных целых чисел, то цикл B-D-B не может выполняться бесконечное число раз, если значения входных переменных программы удовлетворяли предикату φ. Следовательно,

программа целочисленного деления завершается на входном предикате φ.

Мы доказали, что $\{\varphi_0\}P_{div}\{\psi\}$ и $\langle\varphi\rangle P_{div}\langle T\rangle$. Отсюда, по леммам 2 и 3, следует $\langle\varphi\rangle P_{div}\langle\psi\rangle$, то есть полная корректность программы целочисленного деления относительно φ и ψ.

Задачи и упражнения

1.3.1 Для указанной ниже блок-схемы и каждой из спецификаций ответить на вопрос, является ли эта блок-схема частично корректной относительной этой спецификации, является ли эта блок-схема полностью корректной относительно той же спецификации. Множество переменных состоит из одной входной, двух промежуточных и одной выходной переменной $V = \{x, y_1, y_2, z\}$. Доменом всех переменных является множество целых чисел.



a) $\varphi(x) = (x = 0)$
$\psi(x, z) = (z = 0)$

b) $\varphi(x) = (x = 0)$
$\psi(x, z) = (z = 1)$

c) $\varphi(x) = (x \geq 0)$
$\psi(x, z) = (z = x^2)$

d) $\varphi(x) = T$
$\psi(x, z) = (z = x^2)$

e) $\varphi(x) = (|x| > 10)$ $\psi(x, z) = (z > x)$

$\psi(x, z) = (|z| > 100)$ g) $\varphi(x) = (x \geq 0)$

f) $\varphi(x) = (x > 0)$ $\psi(x, z) = (z \geq x)$

1.3.2 Для каждой из указанных ниже спецификаций приведите следующие блок-схемы или обоснуйте, почему они не существуют. Доменом всех входных, промежуточных и выходных переменных должно быть множество целых чисел. В качестве функций, приписанных операторам блок-схемы, можно использовать любую функцию, задаваемую арифметической формулой над операциями сложения, вычитания и умножения, и операциями сравнения (равно, неравно, больше, меньше и т. п.).

- не являющуюся частично корректной относительно данной спецификации;
- являющуюся частично корректной, но не являющуюся полностью корректной относительно данной спецификации;
- являющуюся частично, но не полностью корректной относительно данной спецификации, и при этом завершающуюся на максимально возможном числе входных данных (т. е. завершающуюся при всех значениях входных данных, удовлетворяющих предусловию, при которых существуют значения выходных переменных, на которых выполнено постусловие);
- являющуюся полностью корректной относительно данной спецификации.

a) $\varphi(x) = \mathrm{T}$ $\psi(x, z) = \mathrm{T}$

b) $\varphi(x) = \mathrm{T}$ $\psi(x, z) = \mathrm{F}$

c) $\varphi(x) = F \quad \psi(x, z) = T$

d) $\varphi(x) = (x = 0) \quad \psi(x, z) = (z = 0)$

e) $\varphi(x_1, x_2) = (x_1 = x_2) \quad \psi(x_1, x_2, z) = (x_1 = z + x_2)$

f) $\varphi(x_1, x_2) = (x_1 > x_2 \geq 0) \quad \psi(x_1, x_2, z_1, z_2) = (z_1 > x_1 > z_2 > x_2)$

g) $\varphi(x_1, x_2) = (0 > x_1 > x_2) \quad \psi(x_1, x_2, z_1, z_2) = (x_1^2 > z_1 > x_2^2 > z_2)$

h) $\varphi(x_1, x_2) = (x_1 > x_2 \geq 0) \quad \psi(x_1, x_2, z_1, z_2) = (x_2^2 > z_1 > x_1^2 > z_2)$

i) $\varphi(x_1) = (x_1 > 0) \qquad \psi(x_1, z_1, z_2) = (z_1 = x1 * z_2) \wedge (z_1 < z_2)$

j) $\varphi(x_1, x_2) = (x_1 + x_2 < 3) \quad \psi(x_1, x_2, z_1, z_2) = (z_1 + x_2 > z_2) \wedge (z_1 < x_1 + z_2)$

1.3.3 Доказать или опровергнуть утверждение. Заглавными буквами обозначены блок-схемы, строчными — предикаты.

a) $\forall\, a\ \forall\, P\ \exists\, b\ \ \{a\}P\{b\} \Rightarrow \{b\}P\{a\}$

b) $\forall\, a\ \forall\, P\ \exists\, b\ \ \langle a\rangle P\langle b\rangle \Rightarrow \{b\}P\{a\}$

c) $\forall\, P\ \exists\, a\ \exists\, b\ \ \{a\}P\{b\}$

d) $\forall\, P\ \exists\, a\ \exists\, b\ \ \langle a\rangle P\langle b\rangle$

e) $\forall\, P\ \exists\, a\ \exists\, b\ \ \{a\}P\{b\} \wedge \neg\, \langle a\rangle P\langle b\rangle$

1.4 Формулировка методов Флойда для доказательства корректности программ

Мы рассмотрели доказательство полной корректности программы на примере программы целочисленного деления. Теперь мы определим методы доказательства корректности программ, представленных в виде блок-схем, в общем случае.

Как и в примере, доказательство полной корректности будет проводиться в два этапа. Сначала доказывается частичная корректность программы. Для этого мы будем использовать *метод индуктивных утверждений Флойда*. Вторым шагом проводится доказа-

тельство завершаемости программы. Для решения этой задачи мы будем использовать *метод фундированных множеств Флойда*.

Метод индуктивных утверждений Флойда

Будем рассматривать пути в блок-схемах и обозначать их следующим образом: α − путь в блок-схеме, начинающийся со связки e_1 и завершающийся связкой e_k (связка e_1 выходит из некоторого оператора n_1, связка e_k входит в некоторый оператор n_{k+1}): $n_1 \ -e_1\rightarrow\ n_2\ -e_2\rightarrow\ \ldots\ \ -e_k\rightarrow\ n_{k+1}$. Внутри этого пути находятся операторы $n_2, n_3,\ldots, n_k$.

Дадим несколько вспомогательных определений. Определим *предикат допустимости пути* α (или просто, *предикат пути* α) $R_\alpha(\mathbf{x},\mathbf{y}) : D_\mathbf{x} \times D_\mathbf{y} \rightarrow \{T, F\}$ и *функцию пути* α $r_\alpha(\mathbf{x},\mathbf{y}) : D_\mathbf{x} \times D_\mathbf{y} \rightarrow D_\mathbf{y}$. Предикат пути $R_\alpha(\mathbf{x},\mathbf{y})$ определяет, какое должно быть значение входных и промежуточных переменных в начале пути (в момент выполнения перехода по связке e_1), чтобы дальнейшее вычисление шло по пути α. Функция пути $r_\alpha(\mathbf{x},\mathbf{y})$ определяет, как изменятся значения промежуточных переменных в результате исполнения последователь-ности операторов блок-схемы, находящихся внутри пути α.

Наиболее простым способом составления формул для предиката пути $R_\alpha(\mathbf{x},\mathbf{y})$ и функции пути $r_\alpha(\mathbf{x},\mathbf{y})$ является *метод обратных подстановок*.

Для каждого $m \in \{ 1, \ldots, k \}$ определим предикат $R_\alpha^m(\mathbf{x},\mathbf{y})$ и функцию $r_\alpha^m(\mathbf{x},\mathbf{y})$. При $m = k$ это будут предикат и функция пути, состоящего из одной связки (внутри этого пути не будет ни одного оператора). А при положительных целых $m < k$ это будут предикат

и функция пути от связки e_m в связку e_k (т. е. пути $n_m \xrightarrow{e_m} n_{m+1} \xrightarrow{e_{m+1}} \ldots n_k \xrightarrow{e_k} n_{k+1}$, проходящего через операторы n_{m+1}, ..., n_k).

Таким образом,

$R_\alpha(\mathbf{x},\mathbf{y}) \equiv R_\alpha^1(\mathbf{x},\mathbf{y})$

$r_\alpha(\mathbf{x},\mathbf{y}) \equiv r_\alpha^1(\mathbf{x},\mathbf{y})$

Предикаты $R_\alpha^m(\mathbf{x},\mathbf{y})$ и функции $r_\alpha^m(\mathbf{x},\mathbf{y})$ определим индукцией по m.

Базис индукции (для путей, состоящих из одной связи):

$R_\alpha^k(\mathbf{x}, \mathbf{y}) \equiv T$

$r_\alpha^k(\mathbf{x}, \mathbf{y}) \equiv \mathbf{y}$

Индуктивное предположение:

Зафиксируем некоторое $m < k$. Предположим, что $R_\alpha^{m+1}(\mathbf{x},\mathbf{y})$ и $r_\alpha^{m+1}(\mathbf{x},\mathbf{y})$ уже определены, т. е. определены предикат и функция пути $n_{m+1} \xrightarrow{e_{m+1}} \ldots n_k \xrightarrow{e_k} n_{k+1}$, проходящего через операторы n_{m+2}, ..., n_k.

Индуктивный переход:

Тогда в пути из связки e_m в связку e_k связка e_m будет входить в оператор n_{m+1}, за которым идет путь, рассмотренный в индуктивном предположении. Поэтому можно определить $R_\alpha^m(\mathbf{x},\mathbf{y})$ и $r_\alpha^m(\mathbf{x},\mathbf{y})$ в зависимости от оператора n_{m+1}:

- Если n_{m+1} – оператор присваивания ASSIGN: $\mathbf{y} \leftarrow g(\mathbf{x}, \mathbf{y})$, то

  $R_\alpha^m(\mathbf{x}, \mathbf{y}) \equiv R_\alpha^{m+1}(\mathbf{x}, g(\mathbf{x}, \mathbf{y})) \wedge g(\mathbf{x}, \mathbf{y}) \neq \omega$

  $r_\alpha^m(\mathbf{x}, \mathbf{y}) \equiv r_\alpha^{m+1}(\mathbf{x}, g(\mathbf{x}, \mathbf{y}))$

- Если n_{m+1} – условный оператор TEST: $t(\mathbf{x}, \mathbf{y})$ и связка e_m помечена символом T, то

  $R_\alpha^m(\mathbf{x}, \mathbf{y}) \equiv R_\alpha^{m+1}(\mathbf{x}, \mathbf{y}) \wedge t(\mathbf{x}, \mathbf{y})$

$$r_\alpha^m(\mathbf{x}, \mathbf{y}) \equiv r_\alpha^{m+1}(\mathbf{x}, \mathbf{y})$$

- Если n_{m+1} – условный оператор TEST: $t(\mathbf{x}, \mathbf{y})$ и связка e_m помечена символом F, то

$$R_\alpha^m(\mathbf{x}, \mathbf{y}) \equiv R_\alpha^{m+1}(\mathbf{x}, \mathbf{y}) \wedge \neg t(\mathbf{x}, \mathbf{y})$$

$$r_\alpha^m(\mathbf{x}, \mathbf{y}) \equiv r_\alpha^{m+1}(\mathbf{x}, \mathbf{y})$$

- Если n_{m+1} – оператор соединения JOIN, то

$$R_\alpha^m(\mathbf{x}, \mathbf{y}) \equiv R_\alpha^{m+1}(\mathbf{x}, \mathbf{y})$$

$$r_\alpha^m(\mathbf{x}, \mathbf{y}) \equiv r_\alpha^{m+1}(\mathbf{x}, \mathbf{y})$$

Обратите внимание, что в этом перечислении нет начального и завершающего оператора — причина в том, что эти операторы не могут встречаться внутри пути. Однако в них могут входить и выходить первая или последняя связка пути. Если началом первой связки пути α является начальный оператор START: $\mathbf{y} \leftarrow f(\mathbf{x})$, то будем использовать предикат $R'_\alpha(\mathbf{x}) \equiv R_\alpha(\mathbf{x}, f(\mathbf{x})) \wedge f(\mathbf{x}) \neq \omega$ и функцию $r'_\alpha(\mathbf{x}) \equiv r_\alpha(\mathbf{x}, f(\mathbf{x}))$. Если концом последней связки пути α является завершающий оператор HALT: $\mathbf{z} \leftarrow h(\mathbf{x}, \mathbf{y})$, то будем использовать предикат $R''_\alpha(\mathbf{x}, \mathbf{y}) \equiv R_\alpha(\mathbf{x}, \mathbf{y}) \wedge h(\mathbf{x}, r_\alpha(\mathbf{x}, \mathbf{y})) \neq \omega$ и функцию $r''_\alpha(\mathbf{x}, \mathbf{y}) \equiv h(\mathbf{x}, r_\alpha(\mathbf{x}, \mathbf{y}))$ $(r''_\alpha(\mathbf{x}, \mathbf{y}) : D_\mathbf{x} \times D_\mathbf{y} \to D_\mathbf{z})$.

Индуктивные утверждения. Пусть P – блок-схема, а $\Phi = (\varphi, \psi)$ – ее спецификация. Рассмотрим следующий метод доказательства частичной корректности программы P относительно спецификации Φ.

Шаг 1. Точки сечения. Выберем подмножество связок блок-схемы. Эти связки мы будем называть *точками сечения*. Выбранное множество точек сечения должно быть таким, чтобы каждый цикл в блок-схеме содержал, по крайней мере, одну точку сечения.

Все ориентированные пути между точками сечения, не содержащие других точек сечения, мы будем называть *промежуточными базовыми путями*. Пути, начинающиеся в начальном операторе, заканчивающиеся в точке сечения и не содержащие других точек сечения, мы будем называть *начальными базовыми путями*. Пути, начинающиеся в точке сечения, заканчивающиеся в одном из завершающихся операторов и не содержащие других точек сечения, мы будем называть *конечными базовыми путями*. И наконец, пути, начинающиеся в начальном операторе, заканчивающиеся в одном из завершающихся операторов и не содержащие других точек сечения, мы будем называть *простыми базовыми путями*.

Шаг 2. Индуктивные утверждения. Выберем для каждой точки сечения i предикат $p_i(\mathbf{x}, \mathbf{y})$, который характеризует отношение между переменными блок-схемы при прохождении данной связки. Будем называть эти предикаты *индуктивными утверждениями*. Кроме того, свяжем входной предикат $\varphi(\mathbf{x})$ с начальным оператором блок-схемы, а выходной предикат $\psi(\mathbf{x}, \mathbf{z})$ – со всеми завершающими операторами.

Шаг 3. Условия верификации. На третьем шаге сконструируем для каждого промежуточного базового пути α, начинающегося в точке сечения i и завершающегося в точке сечения j, *условия верификации*:

$$\forall\ \mathbf{x} \in D_x\ \ \forall\ \mathbf{y} \in D_y\ \ [\ \varphi(\mathbf{x}) \wedge p_i(\mathbf{x}, \mathbf{y}) \wedge R_\alpha(\mathbf{x}, \mathbf{y}) \Rightarrow p_j(\ \mathbf{x}, r_\alpha(\mathbf{x}, \mathbf{y})\ )\ ]$$

Эти условия утверждают, что если предикат $p_i(\mathbf{x}, \mathbf{y})$ истинен для некоторых значений переменных $\mathbf{x}$ и $\mathbf{y}$, и эти значения такие, что, начиная из точки сечения i, вычисление пойдет по пути α и

успешно достигнет конца пути, то предикат $p_j(\mathbf{x}, \mathbf{y})$ будет истинен для значений переменных **x** и **y**, после прохождения по пути α.

Для начального базового пути α, завершающегося в точке сечения *j*, условия верификации будут выглядеть следующим образом:

$$\forall\ \mathbf{x} \in D_x\ \ [\ \varphi(\mathbf{x}) \wedge R'_\alpha(\mathbf{x}) \Rightarrow p_j(\ \mathbf{x},\ r'_\alpha(\mathbf{x})\ )\ ]$$

В этом случае условия утверждают, что если входной предикат $\varphi(\mathbf{x})$ истинен для некоторых значений входных переменных и эти значения такие, что начальное вычисление пойдет по пути α и успешно достигнет конца пути, то предикат $p_j(\mathbf{x}, \mathbf{y})$ будет истинен для значений переменных **x** и **y**, после прохождения по пути α.

Для каждого конечного базового пути α, начинающегося в точке сечения *i*, условия верификации конструируются следующим образом:

$$\forall\ \mathbf{x} \in D_x\ \ \forall\ \mathbf{y} \in D_y\ \ [\ \varphi(\mathbf{x}) \wedge p_i(\mathbf{x}, \mathbf{y}) \wedge R''_\alpha(\mathbf{x}, \mathbf{y}) \Rightarrow \psi(\mathbf{x}, r''_\alpha(\mathbf{x}, \mathbf{y}))\ ]$$

Здесь условия верификации утверждают, что если предикат $p_i(\mathbf{x}, \mathbf{y})$ истинен для некоторых значений переменных **x** и **y** и эти значения такие, что, начиная из точки сечения *i*, вычисление пойдет по пути α, то предикат $\psi(\mathbf{x}, \mathbf{z})$ будет истинен для значений переменных **x** и **z** после успешного завершения работы блок-схемы при прохождении по пути α.

Если в блок-схеме существуют простые базовые пути, то для каждого такого пути α (входящего в оператор HALT с функцией *h*) условие верификации будут выглядеть следующим образом:

$$\forall\ \mathbf{x} \in D_x\ \ [\ \varphi(\mathbf{x}) \wedge R'_\alpha(\mathbf{x}) \wedge h(\mathbf{x}, r'_\alpha(\mathbf{x})) \neq \omega \Rightarrow \psi(\mathbf{x}, h(\mathbf{x}, r'_\alpha(\mathbf{x})))\ ]$$

В этом случае условия утверждают, что если входной предикат $\varphi(\mathbf{x})$ истинен для некоторых значений входных переменных и эти значения такие, что вычисление пойдет по пути α и успешно достигнет его конца, то выходной предикат $\psi(\mathbf{x}, \mathbf{z})$ будет истинен для значений переменных **x** и **z** после завершения работы блок-схемы при прохождении по пути α.

Лемма 5. *Пусть все условия верификации истинны. Пусть дано вычисление блок-схемы* P*, входные переменные которого удовлетворяют входному предикату* φ*. Тогда для каждого прохода вычисления* $C_k - C_{k+1}$ *через точку сечения i, предикат* $p_i(\mathbf{x}, \mathbf{y})$ *будет истинен на значениях переменных* **x** *и* **y** *в конфигурации* C_k*.*

Теорема 1. ***(Метод индуктивных утверждений Флойда)***

Пусть даны блок-схема P *и ее спецификация* $\Phi = (\varphi, \psi)$*. Выполним следующие действия:*

1. *Выберем точки сечения;*
2. *Найдем подходящий набор индуктивных утверждений;*
3. *Построим условия верификации для всех базовых путей.*

Если все условия шага 3 истинны, то блок-схема P *частично корректна относительно спецификации* Φ.

Доказательства леммы и теоремы предоставляется читателю в качестве упражнения.

Все шаги метода Флойда, за исключением второго, могут быть выполнены относительно автоматически. А вот выбор подходящего набора индуктивных утверждений требует хорошего понимания функционирования программы и поэтому сложно поддается автоматизации.

Метод фундированных множеств Флойда

Далее мы рассмотрим метод доказательства завершаемости программ, представленных в виде блок-схем.

Напомним некоторые определения. *Частично-упорядоченным множеством* $(W, \prec)$ называется непустое множество W («носитель») и любое бинарное отношение $\prec$ на этом множестве, которое удовлетворяет следующим свойствам:

1. Для всех $a, b, c \in W$ из $a \prec b$ и $b \prec c$ следует $a \prec c$ [Транзитивность]
2. Для всех $a, b \in W$ из $a \prec b$ следует $\neg(b \prec a)$ [Асиметричность]
3. Для всех $a \in W$ $\neg(a \prec a)$ [Иррефлексивность]

Знаком $\succ$ будем обозначать бинарное отношение такое, что для любых $a, b \in W$ $a \succ b$ тогда и только тогда, когда $b \prec a$. Частично-упорядоченное множество $(W, \prec)$ называется *фундированным*, если не существует бесконечно убывающей последовательности его элементов, т. е. такой последовательности $\{a_i\}$, что $a_1 \succ a_2 \succ a_3 \succ \ldots$. Наиболее известным примером фундированного множества является множество натуральных чисел с отношением порядка $<$.

Пусть P – блок-схема, а φ – ее входной предикат. Рассмотрим следующий метод доказательства завершаемости программы P на φ.

Шаг 1. Точки сечения. Выберем множество точек сечения блок-схемы таким образом, чтобы каждый цикл в блок-схеме содержал, по крайней мере, одну точку сечения, и некоторое фундированное

множество (W, $\prec$). Для каждой точки сечения i выберем индуктивное утверждение $q_i(\mathbf{x}, \mathbf{y})$. Построим условия верификации для индуктивных утверждений $q_i(\mathbf{x}, \mathbf{y})$ согласно методу, рассмотренному ранее, для начальных базовых и промежуточных базовых путей и докажем их истинность.

Шаг 2. Оценочные функции. Определим для каждой точки сечения i *оценочную функцию* $u_i(\mathbf{x}, \mathbf{y}) : D_{\mathbf{x}} \times D_{\mathbf{y}} \to W_i$, где $W_i \supseteq W$, и сформулируем *условие корректности определения оценочной функции*:

$$\forall\ \mathbf{x} \in D_{\mathbf{x}}\ \ \forall\ \mathbf{y} \in D_{\mathbf{y}}\ \ [\ \varphi(\mathbf{x}) \wedge q_i(\mathbf{x}, \mathbf{y}) \Rightarrow u_i(\mathbf{x},\mathbf{y}) \in W]$$

Это условие утверждает, что для всех векторов значений переменных $\mathbf{x}$ и $\mathbf{y}$, удовлетворяющих в точке сечения i индуктивному утверждению $q_i(\mathbf{x}, \mathbf{y})$, оценочная функция ставит в соответствие элемент множества W (а не его надмножества).

Шаг 3. Условия завершимости. Сконструируем для каждого промежуточного базового пути α, начинающегося в точке сечения i и завершающегося в точке сечения j, *условие завершимости*:

$$\forall\ \mathbf{x} \in D_{\mathbf{x}}\ \ \forall\ \mathbf{y} \in D_{\mathbf{y}}\ \ [\ \varphi(\mathbf{x}) \wedge q_i(\mathbf{x}, \mathbf{y}) \wedge R_\alpha(\mathbf{x}, \mathbf{y}) \Rightarrow (\ u_i(\mathbf{x},\mathbf{y}) \succ u_j(\mathbf{x}, r_\alpha(\mathbf{x}, \mathbf{y}))\ )\ ]$$

Условие завершимости утверждает, что если предикат $q_i(\mathbf{x}, \mathbf{y})$ истинен для некоторых значений переменных $\mathbf{x}$ и $\mathbf{y}$, и эти значения такие, что, начиная из точки сечения i, вычисление пойдет по пути α, то результат оценочной функции $u_j(\mathbf{x}, \mathbf{y})$ на значениях переменных $\mathbf{x}$ и $\mathbf{y}$ после прохождения по пути α будет меньше результата оценочной функции $u_i(\mathbf{x}, \mathbf{y})$ на исходных значениях.

Шаг 4. Условия успешности вычисления функций. Для доказательства полной корректности необходимо показать, что функции, приписанные операторам блок-схемы, которые могут равняться ω, не вызываются с такими аргументами, при которых они равны ω.

Для оператора START, которому приписана функция *f*, составим следующее условие успешности вычисления функции *f*:

$$\forall\ \mathbf{x} \in D_x\ \ [\ \varphi(\mathbf{x}) \Rightarrow f(\mathbf{x}) \neq \omega\ ]$$

Для каждого пути α, начинающегося в точке сечения *i,* не содержащего в себе других точек сечения и завершающегося перед оператором ASSIGN с функцией *g*, составим следующее условие успешности вычисления функции *g*:

$$\forall\ \mathbf{x} \in D_x\ \ \forall\ \mathbf{y} \in D_y\ \ [\varphi(\mathbf{x}) \wedge q_i(\mathbf{x}, \mathbf{y}) \wedge R_\alpha(\mathbf{x}, \mathbf{y}) \Rightarrow g(\mathbf{x}, r_\alpha(\mathbf{x}, \mathbf{y})) \neq \omega]$$

Аналогичные условия составим для каждого пути α, начинающегося в точке сечения *i,* не содержащего в себе других точек сечения и завершающегося перед оператором HALT с функцией *h*:

$$\forall\ \mathbf{x} \in D_x\ \ \forall\ \mathbf{y} \in D_y\ [\varphi(\mathbf{x}) \wedge q_i(\mathbf{x}, \mathbf{y}) \wedge R_\alpha(\mathbf{x}, \mathbf{y}) \Rightarrow h(\mathbf{x}, r_\alpha(\mathbf{x}, \mathbf{y})) \neq \omega]$$

Для каждого пути α из оператора START с функцией *f* в оператор HALT с функцией *h*, внутри которого нет точек сечения:

$$\forall\ \mathbf{x} \in D_x\ [\varphi(\mathbf{x}) \wedge R'_\alpha(\mathbf{x}) \Rightarrow h(\mathbf{x}, r'_\alpha(\mathbf{x})) \neq \omega]$$

Теорема 2. ***(Метод фундированных множеств Флойда)***

Пусть даны блок-схема P *и ее входной предикат* φ. *Выполним следующие действия:*

1. *Выберем точки сечения с подходящим набором индуктивных утверждений и фундированное множество;*

2. *Выберем подходящий набор оценочных функций;*
3. *Построим условия верификации для всех базовых начальных и промежуточных путей, условия корректности определения всех оценочных функций и условия завершимости для всех промежуточных базовых путей.*
4. *Построим условия успешности вычисления функций, приписанных всем операторам START, ASSIGN и HALT.*

Если все условия на шагах 3 и 4 истинны, то блок-схема P *успешно завершается на* φ.

Доказательство теоремы предоставляется читателю в качестве упражнения.

Замечания к методам Флойда

При доказательстве полной корректности блок-схемы на этапе доказательства завершаемости можно использовать те же точки сечения и индуктивные утверждения, которые были использованы для доказательства частичной корректности. Это позволит сократить объем доказательства полной корректности. Однако если индуктивные утверждения достаточно громоздки, при доказательстве завершаемости следует рассмотреть возможность использования «новых» индуктивных утверждений, более компактных.

После успешного доказательства частичной корректности с некоторыми индуктивными утверждениями можно использовать их следствия при доказательстве успешной завершимости и не составлять и передоказывать условия верификации для них.

Далеко не всегда удается сразу предложить набор индуктивных утверждений, достаточный для доказательства частичной коррект-

ности или завершаемости. При этом если в процессе доказательства обнаружится, что выбранного индуктивного утверждения недостаточно, необходимо начать доказательство частичной корректности сначала, чтобы составить все необходимые условия верификации с нужными измененными индуктивными утверждениями и доказать их истинность. К сожалению, этим часто пренебрегают, что ведет к неверным доказательствам полной корректности, например, из-за того, что измененные индуктивные утверждения не являются корректными для путей, уже ранее рассмотренных в этом доказательстве.

Тем не менее, в некоторых случаях можно вносить такое «изменение» в индуктивное утверждение как «приписывание предусловия». А именно, в условиях верификации, завершимости и корректности определения оценочной функции можно дописывать в конъюнкцию посылки импликации входной предикат, т. к. значения входных переменных не меняются во время выполнения блок-схемы. Например, условие верификации для промежуточного базового пути можно изменить следующим образом:

$\forall\ \mathbf{x} \in D_{\mathbf{x}}\ \ \forall\ \mathbf{y} \in D_{\mathbf{y}}\ \ [\ p_i(\mathbf{x}, \mathbf{y}) \wedge \varphi(\mathbf{x}) \wedge R_\alpha(\mathbf{x}, \mathbf{y}) \Rightarrow p_j(\ \mathbf{x}, r_\alpha(\mathbf{x}, \mathbf{y})\ )\ ]$

Доказательство полной корректности программы целочисленного деления при помощи методов Флойда

Докажем полную корректности блок-схемы целочисленного деления, представленной на рисунке 2, относительно следующей спецификации:

$\varphi \equiv (x_1 \geq 0) \wedge (x_2 > 0)$

$\psi \equiv (x_1 = z_1 x_2 + z_2) \wedge (0 \le z_2 < x_2)$

Доказательство разбиваем на два этапа: сначала покажем частичную корректность этой блок-схемы относительно указанной спецификации (при помощи метода индуктивных утверждений), а затем – завершаемость блок-схемы (при помощи метода фундированных множеств).

Доказываем частичную корректность. Выберем в качестве единственную точку сечения – B. Она разбивает единственный цикл в этой блок-схеме. Сопоставим этой точке сечения индуктивное утверждение $p(x_1, x_2, y_1, y_2) \equiv (x_1 = y_1 x_2 + y_2) \wedge (y_2 \ge 0)$ – см. рисунок 3. Необходимо составить условия верификации для следующих путей в блок-схеме: START – B, B – T – B, B – F – HALT.

Условие верификации для пути START – B:

$\forall\, x_1, x_2 \in \mathbf{Z}\ [\ \varphi(x_1, x_2) \Rightarrow p(x_1, x_2, 0, x_1)\ ]$

Подставляем определения предикатов:

$\forall\, x_1, x_2 \in \mathbf{Z}\ [\ (x_1 \ge 0) \wedge (x_2 > 0) \Rightarrow (x_1 = 0 \cdot x_2 + x_1) \wedge (x_1 \ge 0)\ ]$

Очевидно, что данное утверждение истинно.

Условие верификации для пути B – T – B:

$\forall\, x_1, x_2, y_1, y_2 \in \mathbf{Z}\ [\ (x_1 \ge 0) \wedge (x_2 > 0) \wedge p(x_1, x_2, y_1, y_2) \wedge (y_2 \ge x_2) \Rightarrow p(x_1, x_2, y_1 + 1, y_2 - x_2)\ ]$

Подставляем определения предикатов:

$\forall\, x_1, x_2, y_1, y_2 \in \mathbf{Z}\ [\ (x_1 \ge 0) \wedge (x_2 > 0) \wedge (x_1 = y_1 x_2 + y_2) \wedge (y_2 \ge 0) \wedge (y_2 \ge x_2) \Rightarrow (x_1 = (y_1 + 1) \cdot x_2 + y_2 - x_2) \wedge (y_2 - x_2 \ge 0)\ ]$

Достаточно раскрыть скобки и убедиться в истинности этого утверждения.

Условие верификации для пути B − F − HALT:

$\forall\ x_1, x_2, y_1, y_2 \in \mathbf{Z}\ [\ (x_1 \geq 0) \wedge (x_2 > 0) \wedge p(x_1, x_2, y_1, y_2) \wedge \neg (y_2 \geq x_2) \Rightarrow \psi(x_1, x_2, y_1, y_2)\ ]$

Подставляем определения предикатов:

$\forall\ x_1, x_2, y_1, y_2 \in \mathbf{Z}\ [\ (x_1 \geq 0) \wedge (x_2 > 0) \wedge (x_1 = y_1 x_2 + y_2) \wedge (y_2 \geq 0) \wedge \neg(y_2 \geq x_2) \Rightarrow (x_1 = y_1 x_2 + y_2) \wedge (0 \leq y_2 < x_2)\ ]$

Очевидно, что и это утверждение истинно.

Тем самым, согласно теореме 1 блок-схема целочисленного деления на рисунке 2 является частично корректной относительно указанной выше спецификации.

Доказываем завершаемость блок-схемы целочисленного деления на рисунке 2 при всех значениях входных данных, удовлетворяющих предикату $\varphi \equiv (x_1 \geq 0) \wedge (x_2 > 0)$.

В качестве множества точек сечения выберем единственную точку сечения B (ту же, которая была использована для доказательства частичной корректности) и стандартное фундированное множество ($\{0, 1, 2, ...\}, <$). В этой точке сечения выберем то же индуктивное утверждение, которое было использовано при доказательстве частичной корректности, а в качестве оценочной функции выберем функцию $u \equiv y_2$ (см. рисунок 4). В этом случае условия верификации повторно составлять не нужно − их мы возьмем из доказательства частичной корректности. Необходимо составить условие корректности определения оценочной функции для точки сечения B и условие завершимости для единственного промежуточного базового пути B − T − B.

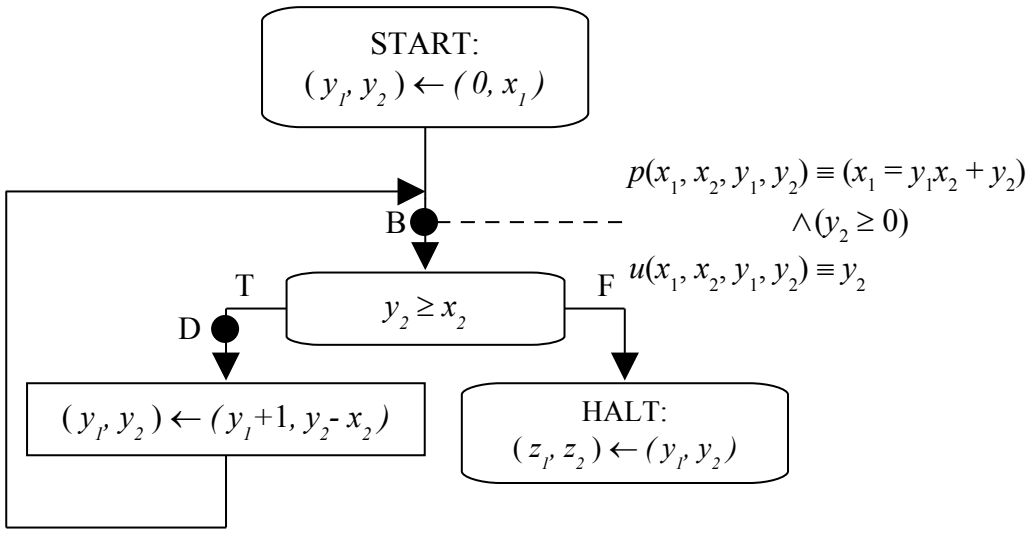


Рисунок 4. Доказательство завершаемости программы целочисленного деления

Условие корректности определения оценочной функции в точке сечения B:

$\forall\ x_1, x_2, y_1, y_2 \in \mathbf{Z}$ [$(x_1 \geq 0) \wedge (x_2 > 0) \wedge$ p$(x_1, x_2, y_1, y_2) \Rightarrow$ u$(x_1, x_2, y_1, y_2) \in$ W]

Подставляем определения предиката p, функции u и множества W:

$\forall\ x_1, x_2, y_1, y_2 \in \mathbf{Z}$ [$(x_1 \geq 0) \wedge (x_2 > 0) \wedge (x_1 = y_1 x_2 + y_2) \wedge (y_2 \geq 0) \Rightarrow (y_2 \geq 0)$]

Очевидно, что это условие истинно.

Условие завершимости для пути B − T − B:

$\forall\ x_1, x_2, y_1, y_2 \in \mathbf{Z}$ [$(x_1 \geq 0) \wedge (x_2 > 0) \wedge$ p$(x_1, x_2, y_1, y_2) \wedge (y_2 \geq x_2) \Rightarrow$ u$(x_1, x_2, y_1, y_2) >$ u$(x_1, x_2, y_1 + 1, y_2 - x_2)$]

Подставляем определения предиката p и функции u:

$\forall\ x_1, x_2, y_1, y_2 \in \mathbf{Z}\ [\ (x_1 \geq 0) \wedge (x_2 > 0) \wedge (x_1 = y_1 x_2 + y_2) \wedge (y_2 \geq 0) \wedge (y_2 \geq x_2) \Rightarrow y_2 > y_2 - x_2\ ]$

Это условие является истинным.

Тем самым, согласно теореме 2 блок-схема целочисленного деления на рисунке 2 завершается при всех значениях входных данных, удовлетворяющих предикату φ.

Нами доказана частичная корректность и завершаемость блок-схемы, а, значит, доказана полная корректность.

Задачи и упражнения

В некоторых задачах верифицируемая программа записана в виде функции на языке программирования Си. При решении задачи необходимо (явно или неявно) составить модель этой функции в виде блок-схемы. Блок-схема должна дословно повторять текст функции без преобразования выражений, моделируя последовательность операторов функции последовательностью соответствующих подсхем, условный оператор и операторы цикла – при помощи соответствующих операторов языка блок-схем. Все локальные переменные функции объявляются в её начале и моделируются соответствующим оператором START. Для упрощения считать, что все арифметические операции выполняются точно, разрядной сетки хватает для проведения всех необходимых вычислений в этой функции.

1.4.1 В блок-схеме на рисунке 5 заданы путь α и предикат p_1: $D_x \times D_y \rightarrow \{T, F\}$ – на значения входных и промежуточных переменных в начале этого пути.

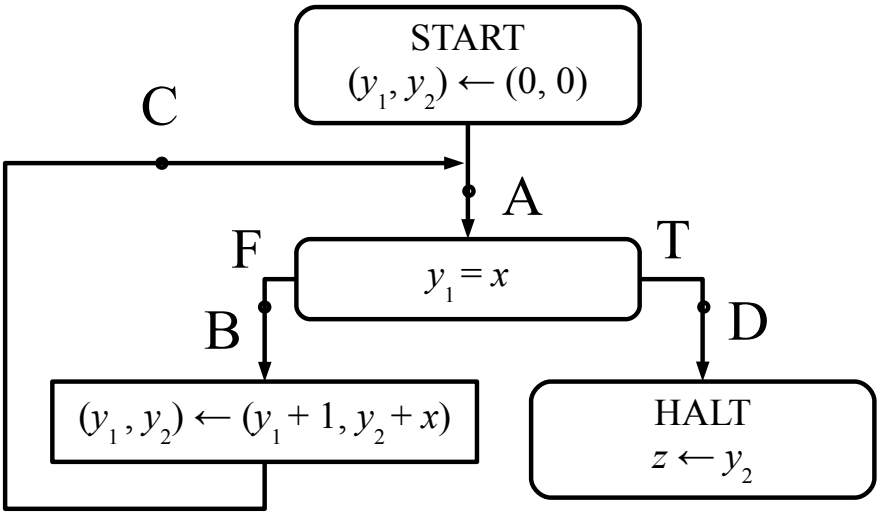


Рисунок 5. Рисунок к задаче

Записать точный предикат p_2: Dx × Dy → {T, F} − на значения входных и промежуточных переменных в конце этого пути, если вычисление идет по пути α и начинается из конфигурации со значениями входных и промежуточных переменных, удовлетворяющими предикату p_1.

Т.е. записать предикат p_2 истинный на тех и только тех значениях x' ∈ Dx и y' ∈ Dy, для которых существуют x ∈ Dx и y ∈ Dy такие, что выполнено p_1(x, y), и существует вычисление, начинающееся в конфигурации со значениями переменных (x, y), выполняющееся по пути α и завершающееся в конфигурации в конце пути со значениями переменных (x', y'). Множество переменных V = {x, y_1, y_2, z} состоит из одной входной, двух промежуточных и одной выходной переменной. Доменами всех переменных является множество целых чисел.

a) α: B − C; $p_1(x, y_1, y_2) = (x = 0 \wedge y_1 = 0 \wedge y_2 = 1)$

b) α: B − C; $p_1(x, y_1, y_2) = (y_2 = 1)$

c) α: B − C; $p_1(x, y_1, y_2) = (y_1 = x)$

d) α: B − C; $p_1(x, y_1, y_2) = (1 < y_1 < 10 \wedge y_2 > x)$

e) α: B − C − B; $p_1(x, y_1, y_2) = (y_1 \cdot x = y_2)$

f) α: B − C − B − C − B; $p_1(x, y_1, y_2) = (y_1 \cdot x = y_2)$

g) α: C − B − C; $p_1(x, y_1, y_2) = (y_1 \cdot x = y_2)$

h) α: C − B − C; $p_1(x, y_1, y_2) = (y_2 < 0)$

i) α: A − B − C − A; $p_1(x, y_1, y_2) = (y_1 \geq x)$

j) α: B − C − D; $p_1(x, y_1, y_2) = (y_1 = 1 \wedge y_2 = 0)$

k) α: B − C − D; $p_1(x, y_1, y_2) = (y_1 = 1)$

l) α: START − A − B − C − B − C − A − D − HALT; $p_1(x, y_1, y_2) = (x = 2 \wedge 0 \leq y_1 < x)$

1.4.2 Для заданной блок-схемы, спецификации, точек сечения, индуктивных утверждений, фундированного множества и оценочных функций выпишите все те и только те условия верификации, корректности и завершимости, которые не являются истинными при доказательстве полной корректности блок-схемы относительно спецификации при помощи методов Флойда. Доменами всех переменных является множество всех целых чисел.

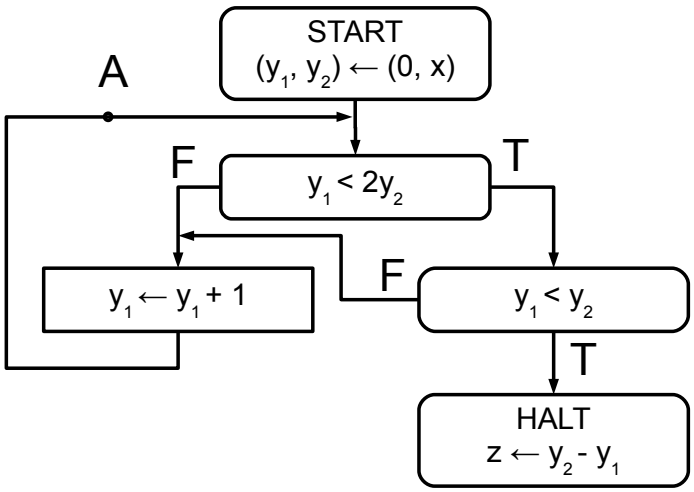
a) $\phi(x) = (x > 0)$

$\psi(x, z) = (z > 0)$

$p_A(x, y_1, y_2) = (y_1 \leq 2y_2)$

$W = (Nat_0, <)$

$u_A(x, y_1, y_2) = y_1$



b) $\varphi(x) = (|x| < 10)$

$\psi(x, z) = (|x| = |z|)$

$p_A(x, y_1, y_2) = (y_1 = -x)$

$p_B(x, y_1, y_2) = (y_1 = -x)$

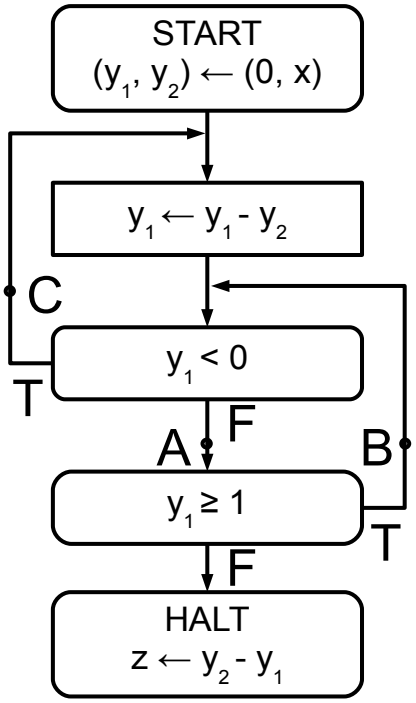
$p_C(x, y_1, y_2) = (y_2 = x \wedge x > 0)$

$W = (Nat_0, <)$

$u_B(x, y_1, y_2) = y_1 - 1$

$u_A(x, y_1, y_2) = y_1$

$u_C(x, y_1, y_2) = y_1$



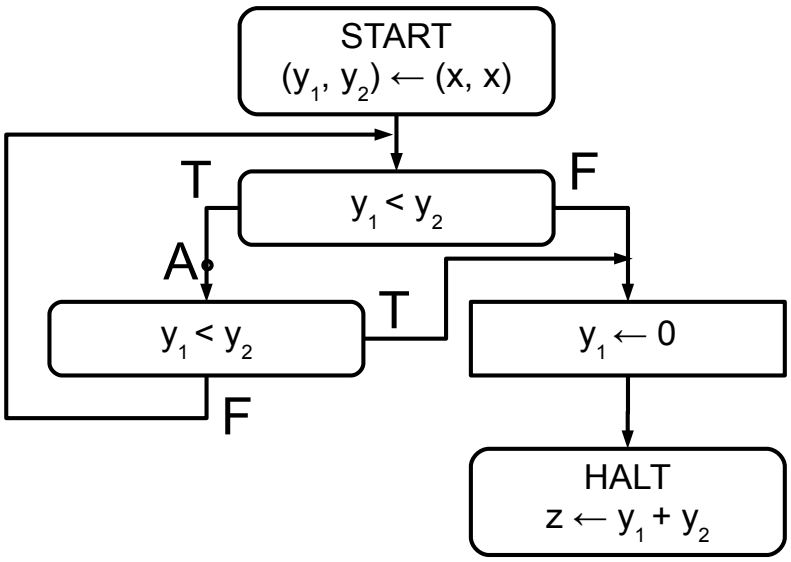
c) $\phi(x) = (x > 0)$

$\psi(x, z) = (z \geq x)$

$p_A(x, y_1, y_2) = (y_1 = x)$

$W = (Nat_0, <)$

$u_A(x, y_1, y_2) = y_2$



d) $\phi(x) = (|x| < 5)$

$\psi(x, z) = (z \geq 0)$

$p_A(x, y_1, y_2) = (y_1 = -x)$

$p_B(x, y_1, y_2) = (y_1 = -x)$

$p_C(x, y_1, y_2) =$

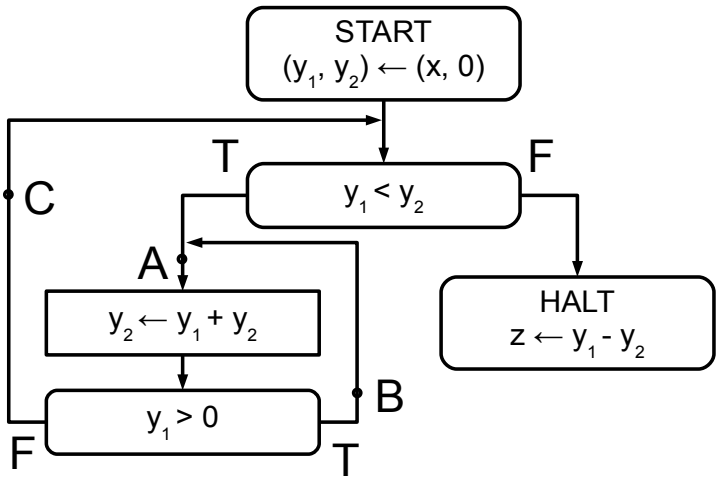
$(y_2 = x \wedge x > 0)$

$W = (Nat_0, <)$

$u_B(x, y_1, y_2) = y_1 - 1$

$u_A(x, y_1, y_2) = y_1$

$u_C(x, y_1, y_2) = y_1$



e) $\phi(x) = (x \geq 0)$

$\psi(x, z) = (z > 0)$

$p_A(x, y_1, y_2) = (y_2 = x)$

$p_B(x, y_1, y_2) = (y_2 = x)$

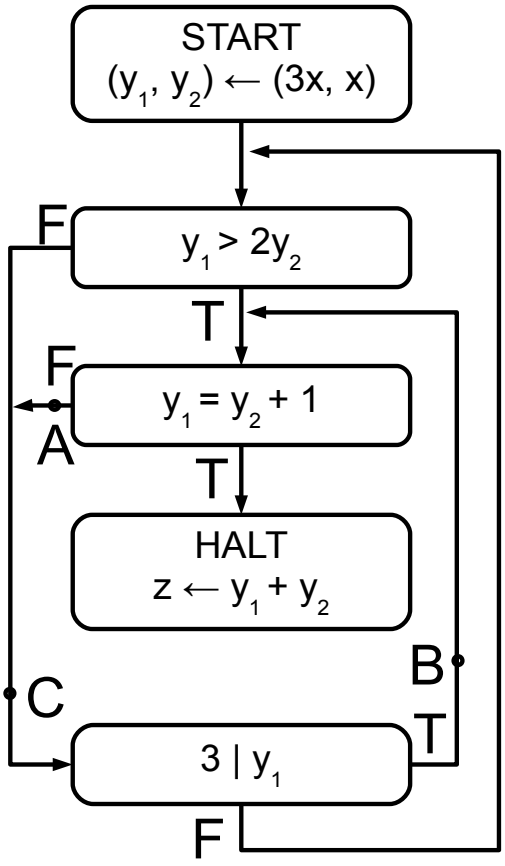
$p_C(x, y_1, y_2) = (y_2 = x \wedge y_1 \geq 0)$

$W = (Nat_0, <)$

$u_A(x, y_1, y_2) = y_1 + 2$

$u_B(x, y_1, y_2) = y_1$

$u_C(x, y_1, y_2) = y_1 + 1$



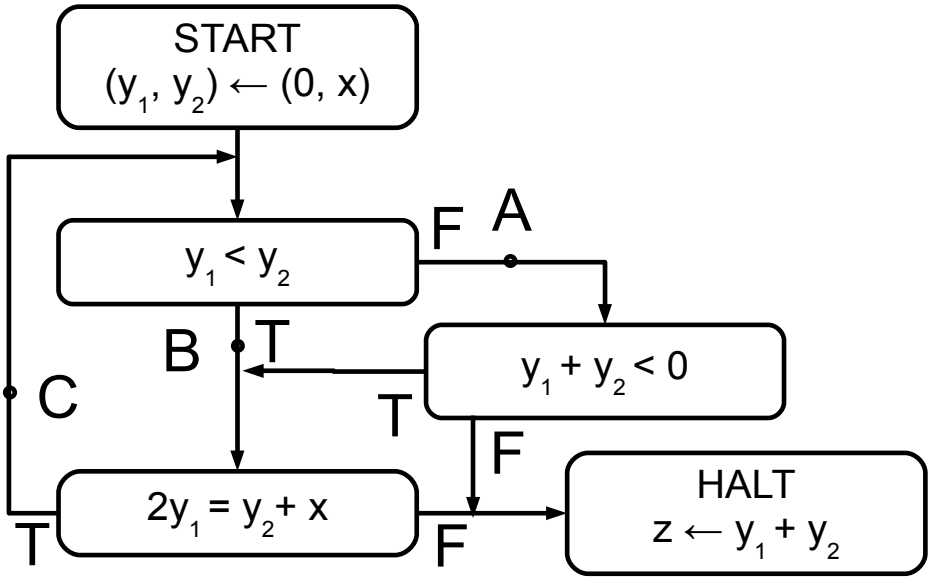
f) $\varphi(x) = (x \geq 0)$ $\psi(x, z) = (|z| = x)$

$p_A(x, y_1, y_2) = (y_2 = x \wedge y_1 = 0)$

$p_B(x, y_1, y_2) = (y_2 = 0 \Rightarrow y_1 = 0)$ $p_C(x, y_1, y_2) = (2y_1 = y_2 + x)$

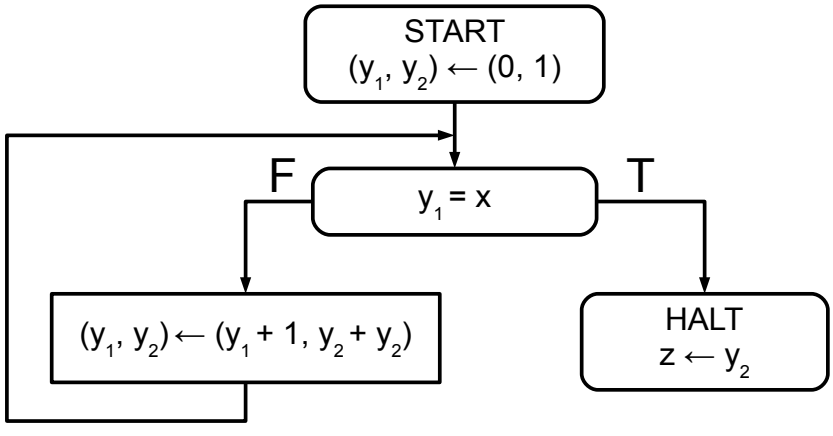
$W = (Nat_0, <)$ $u_C(x, y_1, y_2) = y_2 - 1$ $u_A(x, y_1, y_2) = y_2$

$u_B(x, y_1, y_2) = y_2 + 1$

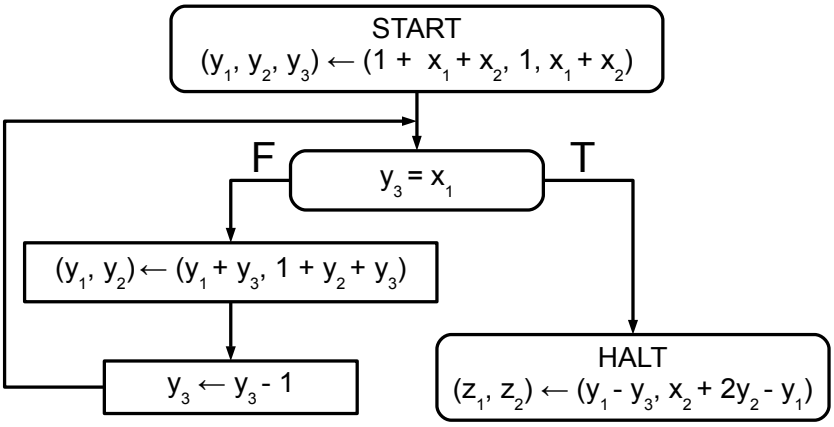


1.4.3 При помощи метода индуктивных утверждений доказать, что следующая блок-схема частично корректна относительно указанной спецификации (φ, ψ). Доменом каждой переменной является множество всех целых чисел.

Спецификация: $\varphi(x) = T$ $\psi(x, z) = (z = 2^x)$



1.4.4 При помощи метода индуктивных утверждений доказать, что следующая блок-схема частично корректна относительно указанной спецификации (φ, ψ). Доменом каждой переменной является множество всех целых чисел.

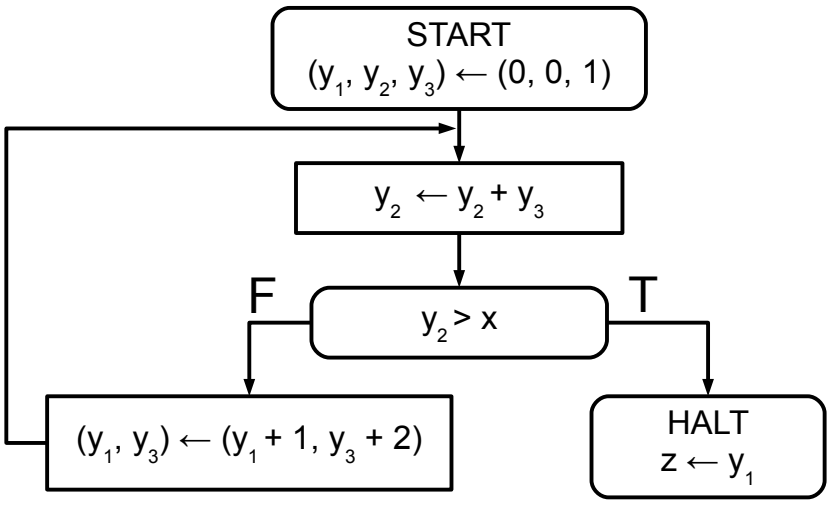


Спецификация:

$$\varphi(x_1, x_2) = (x_2 \geq 0) \qquad \psi(x_1, x_2, z_1, z_2) = (z_1 - x_1 = z_2 - x_2)$$

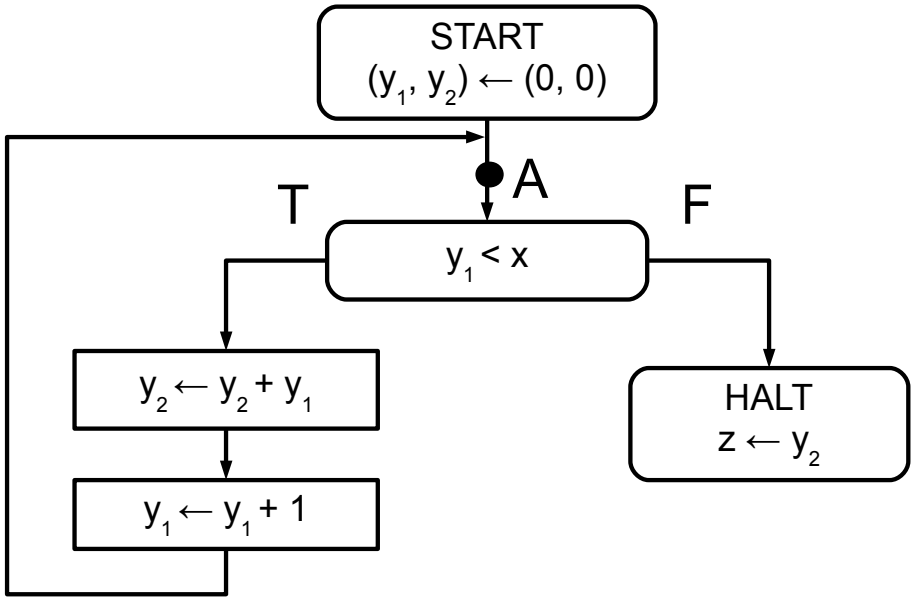
1.4.5 При помощи метода индуктивных утверждений доказать, что следующая блок-схема частично корректна относительно указанной спецификации (φ, ψ), доменом всех переменных является множество всех целых чисел:

$$\varphi(x) = (x \geq 0) \qquad \psi(x, z) = (z^2 \leq x < (z + 1)^2)$$



1.4.6 Следующая функция языка Си была промоделирована указанной ниже блок-схемой. В блок-схеме выделена точка сечения A, которой приписано утверждение $p(x, y_1, y_2) = (0 \le y_1 < x)$. Обоснуйте или опровергните следующее утверждение: p выполнено всякий раз, когда вычисление находится в точке A (т. е. может быть использовано для доказательства частичной корректности). Функция вызывается только с положительными значениями x.

```
int sum(int x)
{
   int i = 0, s = 0;
   for(; i < x; i++)
   {
     s += i;
   }
   return s;
}
```



1.4.7 Обоснованно ответить на вопрос: какое требуется минимальное число точек сечения, чтобы доказать **частичную корректность** хотя бы одной блок-схемы, имеющей указанные особенности, относительно хотя бы одной спецификации при помощи методов Флойда, если блок-схемы с указанными особенностями существуют.

Если ответ зависит от спецификации, привести не менее двух разных ответов и обосновать их.

a) блок-схема содержит 2 цикла;

b) блок-схема содержит не менее одного оператора соединения;

c) блок-схема зацикливается на всех значениях входных данных из своих доменов;

d) операторы и связки блок-схемы разбиваются на 3 независимые связные части так, что первая и вторая часть обладают единственной общей связкой, вторая и третья часть обладают единственной общей связкой, а первая и третья часть не обладают общими связками. Общих операторов у частей нет.

1.4.8 Обоснованно ответить на вопрос: какое требуется минимальное число точек сечения, чтобы доказать **завершаемость** хотя бы одной блок-схемы, имеющей указанные особенности, относительно хотя бы одной спецификации при помощи методов Флойда, если блок-схемы с указанными особенностями существуют. Если ответ зависит от спецификации, привести не менее двух разных ответов и обосновать их. Если доказать завершаемость невозможно, обоснуйте это.

a) блок-схема содержит 2 цикла;

b) блок-схема содержит не менее одного оператора соединения;

c) блок-схема зацикливается на всех значениях входных данных из своих доменов;

d) операторы и связки блок-схемы разбиваются на 3 независимые связные части так, что первая и вторая часть обладают единственной общей связкой, вторая и третья часть обладают единственной общей связкой, а первая и третья часть не обладают общими связками. Общих операторов у частей нет.

1.4.9 Дано выражение. Обосновать или опровергнуть утверждение: существует блок-схема P, спецификация (φ, ψ) для нее и доказательство $\langle\varphi\rangle P\langle\psi\rangle$ при помощи методов Флойда, в котором данное выражение используется в качестве оценочной функции.

a) x (x − целое число);

b) x^2 (x − целое число);

c) $\sin^2 x + \cos^2 x$ (x − вещественное число);

d) 1 / x (x − вещественное число);

e) x (x − очередь из целых чисел);

f) x (x − очередь из неотрицательных целых чисел).

1.4.10 Каково минимальное и максимальное число утверждений, которые необходимо доказать при использовании лишь одной точки сечения в рамках доказательства частичной корректности P (она задана в виде программой на языке Си)? полной корректности?

Ответ обосновать. Если одной точки сечения недостаточно, обосновать это. Напишите, как зависит ответ от спецификации этой блок-схемы.

a)
```
int P(int x) {
    int y1 = x, y2 = x;
    do {
        if (y1 > y2 + y2) {
            y2 ++;
        }
        y1 += y2;
    } while (y1 < y2);
    return y1 + y2;
  }
```

b)
```
int P(int x1, int x2) {
    int y1 = x1, y2 = x2;
    while (x1 > y1) {
        switch (y1 + y2) {
            case 0: y1 ++;
            case 1: y2 ++;
        }
        return -1;
    }
    return y1 + y2;
  }
```

c)
```
int P(int x1, int x2) {
        int y1 = x1, y2 = x2, y3 = 0;
        while (y3 < x) {
            if (y1 < y2)
                return y3;
            y1 = y2 = y3;
```

```
        }
        while (y2 > 0) {
             if (y1 < 2 * y3)
                  break;
             y2 --;
        }
        return 0;
   }
```

d)
```
char *strcpy(char *dest, const char *src) {
      char *p = dest, *q = src;
       while (*q != 0) {
            *p = *q;
            q++;
       }
       return dest;
   }
```

1.4.11 Можно ли только при помощи методов Флойда доказать, что в указанной функции на языке Си значение указанной переменной всегда принадлежит указанному множеству значений? Если да, покажите, как (предъявив хотя бы несколько формальных выкладок). Если нет, обоснуйте это. Если спецификация не указана, считается, что она содержит тождественно истинные предусловие и постусловие. Если в этой функции указанная переменная не всегда принадлежит указанному множеству значений, обоснуйте это.

a)
```
void swap(int *x, int *y) {
       int t = *x;
       *x = *y;
```

```
        *y = t;
    }
```

переменная: x, область значений: не-0 указатели.

b)
```
int abs(int *x) {
        int y = -1;
        if (x != 0 && (y = *x) < 0) {
            *x = - y;
            return *x;
        }
        return -1;
    }
```

переменная: y, область значений: отрицательные числа; предусловие: x != 0.

c)
```
int search(int x, int *data, size_t len) {
        int i = 0;
        for ( ; i < len; i++) {
            if (data[i] == x) return i;
        }
        return -1;
    }
```

переменная: i, область значений: 0 <= i < len; предусловие: data указывает на массив размера не менее len.

1.4.12 Можно ли только при помощи методов Флойда доказать, что в приведенной ниже функции на языке Си никогда не выполняется указанное условие на значения переменных? Если да, покажите, как (предъявив хотя бы несколько формальных выкладок). Если нет, обоснуйте это. Если спецификация не указана, считается, что она содержит тождественно истинные предусловие и постусловие. Если в функции указанное условие может быть выполнено, обоснуйте это.

a)
```
void swap(int *x, int *y) {
    int t = *x;
    *x = *y;
    *y = t;
}
```
непосредственно в этой функции происходит разыменование нулевого указателя.

b)
```
long list_sum(struct node_t *list) {
    long sum = 0L; struct_node_t *p = list;
    while (p != 0) {
        sum += p->value;
        p = p->next;
    }
    return sum;
}
```
сумма элементов списка из одного элемента больше заглавного элемента.

2 Практикум по методам Флойда

В данном разделе приводится большое число задач на доказательство полной корректности блок-схем.

2.1 ***Ещё один пример доказательства полной корректности***

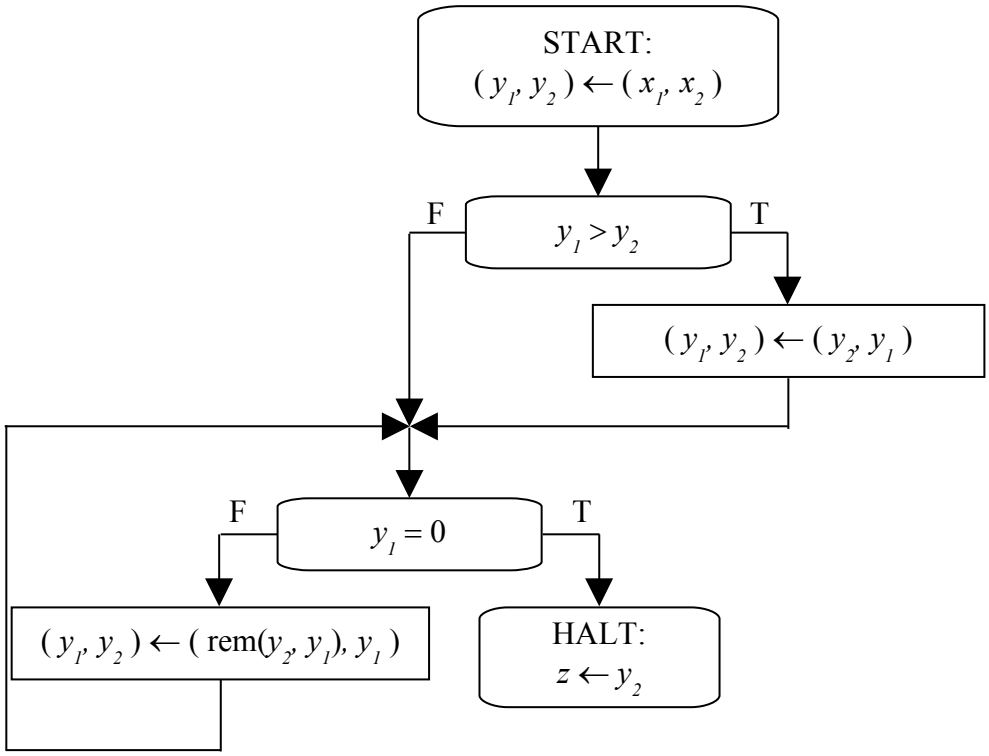


Рисунок 6. Блок-схема программы вычисления наибольшего общего делителя.

На рисунке 6 представлена блок-схема программы над множеством переменных V, вычисляющая наибольший общий делитель. Множество переменных $V = \{x_1, x_2, y_1, y_2, z\}$ состоит из двух входных, двух промежуточных и одной выходной переменных. Доменом всех переменных является множество целых чисел.

Оператор rem(y_1, y_2) обозначает остаток при целочисленном делении y_1 на y_2.

Необходимо доказать полную корректность блок-схемы относительно входного предиката $\varphi \equiv (x_1 > 0) \wedge (x_2 > 0)$ и выходного предиката $\psi \equiv (z = \text{НОД}(x_1, x_2))$.

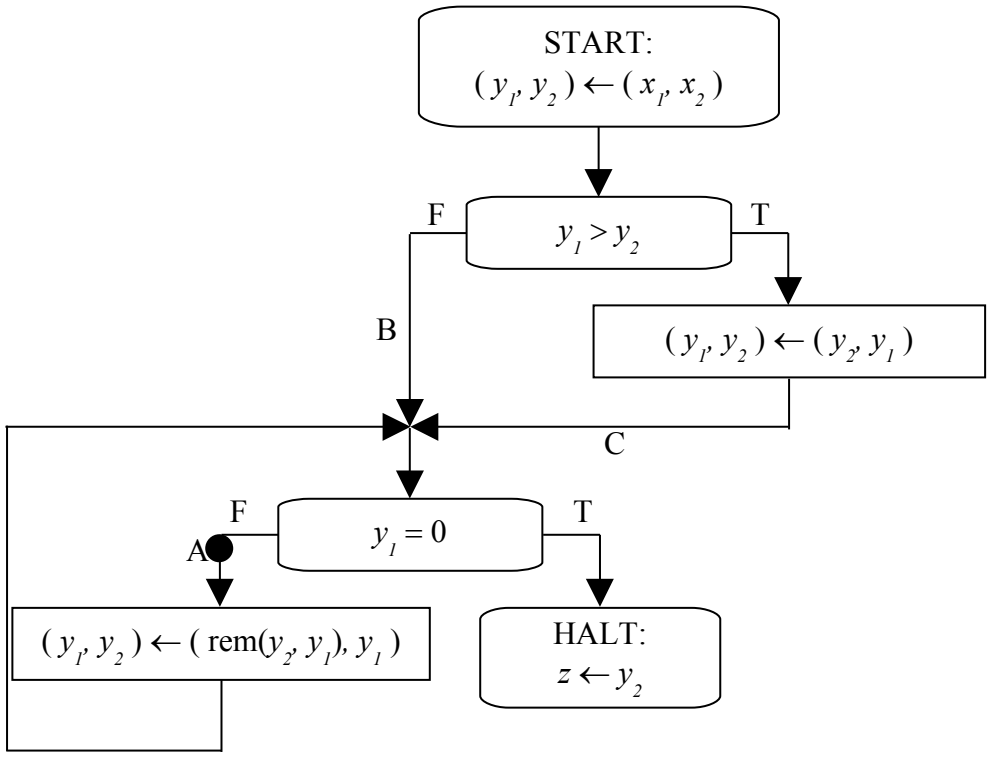


Рисунок 7. Блок-схема вычисления НОД с точкой сечения

Решение:

Для доказательства корректности программы нам будут полезны следующие свойства наибольшего общего делителя:

Утверждение 1. Если натуральное число a не делится без остатка на натуральное число b, то НОД(a, b) = НОД(rem(a,b), b).

Утверждение 2. Если натуральное число a делится на натуральное число b, то НОД(a, b) = b.

1. ***Доказательство частичной корректности программы.***

Шаг 1. Точки сечения. Выберем точку сечения программы на переходе, выходящем из нижнего условного оператора и помеченном символом F. На рисунке 7 эта точка сечения помечена символом A. Эта точка сечения разбивает единственный цикл, имеющийся в программе, и поэтому определять другие точки сечения не требуется.

Шаг 2. Индуктивные утверждения. Поставим в соответствие единственной точке сечения индуктивное утверждение

$$p(x_1, x_2, y_1, y_2) = (0 < y_1 \le y_2) \wedge (НОД(x_1, x_2) = НОД(y_1, y_2)).$$

Шаг 3. Условия верификации. Перечислим все базовые пути программы:

1. START-B-HALT
2. START-C-HALT
3. START-B-A
4. START-C-A
5. A-A
6. A-HALT

Рассмотрим условия верификации, соответствующие этим путям.

START-B-HALT

$$\forall\ x_1\ x_2\ (x_1 > 0) \wedge (x_2 > 0) \wedge \neg(x_1 > x_2) \wedge (x_1 = 0) \Rightarrow (x_2 = НОД(x_1, x_2))$$

Предпосылка является ложной, так как x_1 не может одновременно равняться 0 и быть больше 0. Следовательно, данное условие верификации является истинным.

START-C-HALT

$$\forall\ x_1\ x_2\ (x_1 > 0) \wedge (x_2 > 0) \wedge (x_1 > x_2) \wedge (x_2 = 0) \Rightarrow (x_1 = НОД(x_1, x_2))$$

Опять предпосылка является ложной, так как x_2 не может одновременно равняться 0 и быть больше 0. Следовательно, данное условие верификации является истинным.

START-B-A

$$\forall\ x_1\ x_2\ (x_1 > 0) \wedge (x_2 > 0) \wedge \neg(x_1 > x_2) \wedge \neg(x_1 = 0) \Rightarrow (0 < x_1 \leq x_2) \wedge (\text{НОД}(x_1, x_2) = \text{НОД}(x_1, x_2))$$

START-C-A

$$\forall\ x_1\ x_2\ (x_1 > 0) \wedge (x_2 > 0) \wedge (x_1 > x_2) \wedge \neg(x_2 = 0) \Rightarrow (0 < x_2 \leq x_1) \wedge (\text{НОД}(x_1, x_2) = \text{НОД}(x_2, x_1))$$

A-A

$$\forall\ x_1\ x_2\ y_1\ y_2\ (x_1 > 0) \wedge (x_2 > 0) \wedge (0 < y_1 \leq y_2) \wedge (\text{НОД}(x_1, x_2) = \text{НОД}(y_1, y_2)) \wedge \neg(\text{rem}(y_2, y_1) = 0) \Rightarrow (0 < \text{rem}(y_2, y_1) \leq y_1) \wedge (\text{НОД}(x_1, x_2) = \text{НОД}(\text{rem}(y_2, y_1), y_1))$$

A-HALT

$$\forall\ x_1\ x_2\ y_1\ y_2\ (x_1 > 0) \wedge (x_2 > 0) \wedge (0 < y_1 \leq y_2) \wedge (\text{НОД}(x_1, x_2) = \text{НОД}(y_1, y_2)) \wedge (\text{rem}(y_2, y_1) = 0) \Rightarrow (\ y_1 = \text{НОД}(x_1, x_2)\ )$$

2. ***Доказательство завершимости программы.***

Шаг 1. Точки сечения. Выберем ту же точку сечения программы, что и в предыдущем случае. В качестве индуктивного утверждения выберем $q(x_1, x_2, y_1, y_2) = (0 < y_1 \leq y_2)$, которое является следствием индуктивного утверждения, использовавшегося ранее. Это мы сделаем для сокращения доказательства завершаемости: во-первых, не нужно повторно составлять условия верификации, а, во-вторых, условия корректности и завершимости будут короче. В качестве фундированного множества мы возьмем множество натуральных чисел с естественным отношением порядка.

Шаг 2. Оценочные функции. Поставим в соответствие единственной точке сечения оценочную функцию $u(x_1, x_2, y_1, y_2) = y_1$. Значения этой переменной неотрицательны и убывают на каждой итерации. Осталось это показать формально.

Сформулируем условие корректности определения оценочной функции: $\forall\ x_1\ x_2\ y_1\ y_2\ \ (x_1 > 0) \wedge (x_2 > 0) \wedge (0 < y_1 \le y_2) \Rightarrow (y_1 > 0)$

Это условие является истинным.

Шаг 3. Условия завершимости. Рассмотрим все промежуточные базовые пути блок-схемы. В данном случае − это единственный путь А-А. Запишем для него условие завершимости:

$\forall\ x_1\ x_2\ y_1\ y_2\ \ (x_1 > 0) \wedge (x_2 > 0) \wedge (0 < y_1 \le y_2) \wedge \neg\ (\mathrm{rem}(y_2, y_1) = 0)\ \Rightarrow (y_1 > \mathrm{rem}(y_2, y_1))$

Это условия является истинным, так как при целочисленном делении одного положительного числа на другое остаток по определению меньше делителя.

Шаг 4. Условия успешности вычисления функций. В блок-схеме используется операция $\mathrm{rem}(y_2, y_1)$, которая не определена при $y_1 = 0$. Поэтому необходимо доказать, что всякий раз при вычислении этой операции будет соблюдено условие $y_1 \neq 0$. Для этого составим следующее условие корректности:

$\forall\ x_1\ x_2\ y_1\ y_2\ \ (x_1 > 0) \wedge (x_2 > 0) \wedge (0 < y_1 \le y_2)\ \Rightarrow (y_1 \neq 0)$

Это условие также является истинным.

Мы доказали частичную корректность относительно (φ, ψ) и успешную завершаемость программы на φ. Из этого по лемме 2 следует полная корректность программы.

2.2 ***Один подход к построению индуктивных утверждений и оценочных функций***

Основной сложностью данных задач является выбор подходящих индуктивных утверждений и оценочных функций.

Индуктивные утверждения выбираются, исходя из семантики программы, как инварианты на значение переменных в соответствующей точке. Эти инварианты должны быть достаточны для доказательства всех условий в методе индуктивных утверждений (и, при необходимости, в методе фундированных множеств).

Для выбора индуктивных утверждений бывает полезно выписать значения всех промежуточных переменных в предполагаемой точке сечения для нескольких наборов входных данных. Это может подсказать зависимость между переменными и выразить её в виде предиката. Последующая проверка корректности выбранных утверждений выполняется при доказательстве сформулированных условий верификации, корректности и завершимости. Если какое-то из условий не доказывается, то, как правило, становится видно, что именно необходимо изменить в индуктивном утверждении для исправления ситуации.

Оценочные функции выбираются для обеспечения монотонного убывания некоторой величины при каждом переходе по промежуточному базовому пути.

Прокомментируем сказанное об индуктивных утверждениях на примере. Требуется доказать частичную корректность блок-схемы на рисунке 8 относительно входного предиката $\varphi \equiv (x \geq 0)$ и выходного предиката $\psi \equiv (z = x^2)$. Домены всех переменных — целые числа.

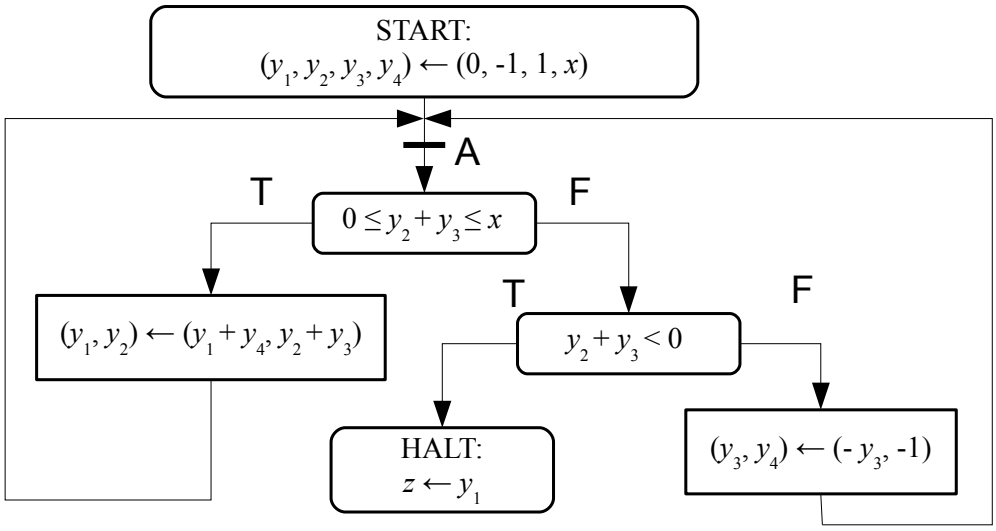


Рисунок 8. К демонстрации метода построения индуктивых утверждений

Выберем точки сечения так, чтобы уменьшить число условий верификации. Для этого можно выбрать точку A, как это показано на рисунке 8, и обозначить индуктивное утверждение, приписанное этой точке, как $p(x, y_1, y_2, y_3, y_4)$. Тогда мы должны подобрать такое выражение для предиката p, чтобы были выполнены все следующие условия верификации (для сокращения записи кванторы опущены):

1. $x \ge 0 \Rightarrow p(x, 0, -1, 1, x)$
2. $x \ge 0 \wedge 0 \le y_2 + y_3 \le x \wedge p(x, y_1, y_2, y_3, y_4) \Rightarrow p(x, y_1 + y_4, y_2 + y_3, y_3, y_4)$
3. $x \ge 0 \wedge y_2 + y_3 > x \wedge p(x, y_1, y_2, y_3, y_4) \Rightarrow p(x, y_1, y_2, -y_3, -1)$
4. $x \ge 0 \wedge y_2 + y_3 < 0 \wedge p(x, y_1, y_2, y_3, y_4) \Rightarrow y_1 = x^2$

Самым простым индуктивным утверждением, инвариантом, является тождественно истинный инвариант. Проверим, может ли быть

$p(x, y_1, y_2, y_3, y_4) \equiv \mathrm{T}$

При таком p первые три формулы истинны, а последняя ложна. Посмотрим контрпример для последней формулы, если $p \equiv$ T, т. е. подберем такие значения для переменных, при которых посылка истинна, а заключение ложно. Например, контрпримером будут такие значения: x = 0, $y_1 = 1$, $y_2 = 0$, $y_3 = -1$, $y_4 = 0$.

Проанализируем, может ли такой контрпример реализоваться при указанном x. Для этого достаточно выписывать вычисление при этом значении переменной x. Получится, что, например, y_1 во всех конфигурациях равен 0, но не 1, как в контрпримере. Т.е. индуктивное утверждение $p \equiv$ T не позволяет доказать последнюю формулу, т. е. из него не следует, что $y_1 = x^2$. Проанализируем, какие инвариантные свойства обеспечит выполнение этого условия. Можно попытаться это определить, выписав разные вычисления.

Можно сделать вывод, что y_1 сначала возрастает, начиная с 0, увеличиваясь на x, а затем убывает до x^2. Причем при возрастании y_3 равен 1, а при убывании он равен -1. Тогда можно сформулировать такие следствия:

$y_3 = 1 \Rightarrow y_1 = x\,(y_2 + 1)$

$y_3 = -1 \Rightarrow y_1 = x^2 + y_2$

Тогда возьмем в качестве p такой предикат: $(y_3 = 1 \Rightarrow y_1 = x\,(y_2 + 1)) \wedge (y_3 = -1 \Rightarrow y_1 = x^2 + y_2)$ и подставим его в написанные выше 4 формулы:

1. $x \geq 0 \Rightarrow (1 = 1 \Rightarrow 0 = x\,(-1 + 1)) \wedge (1 = -1 \Rightarrow 0 = x^2 + (-1))$
2. $x \geq 0 \wedge 0 \leq y_2 + y_3 \leq x \wedge (y_3 = 1 \Rightarrow y_1 = x\,(y_2 + 1)) \wedge (y_3 = -1 \Rightarrow y_1 = x^2 + y_2) \Rightarrow (y_3 = 1 \Rightarrow y_1 + y_4 = x\,(y_2 + y_3 + 1)) \wedge (y_3 = -1 \Rightarrow y_1 + y_4 = x^2 + y_2 + y_3)$

3. $x \geq 0 \wedge y_2 + y_3 > x \wedge (y_3 = 1 \Rightarrow y_1 = x\ (y_2 + 1)) \wedge (y_3 = -1 \Rightarrow y_1 = x^2 + y_2) \Rightarrow (-y_3 = 1 \Rightarrow y_1 = x\ (y_2 + 1)) \wedge (-y_3 = -1 \Rightarrow y_1 = x^2 + y_2)$
4. $x \geq 0 \wedge y_2 + y_3 < 0 \wedge (y_3 = 1 \Rightarrow y_1 = x\ (y_2 + 1)) \wedge (y_3 = -1 \Rightarrow y_1 = x^2 + y_2) \Rightarrow y_1 = x^2$

Первая формула истинна, т. к. истинно следствие.

Проведем эквивалентное преобразование второй формулы, разбив ее на три случая:

2а. $y_3 = 1 \Rightarrow (x \geq 0 \wedge 0 \leq y_2 + 1 \leq x \wedge y_1 = x\ (y_2 + 1) \Rightarrow y_4 = x)$

2б. $y_3 = -1 \Rightarrow (x \geq 0 \wedge 0 \leq y_2 - 1 \leq x \wedge y_1 = x^2 + y_2 \Rightarrow y_4 = -1)$

2в. $y_3 \neq 1 \wedge y_3 \neq -1 \Rightarrow (x \geq 0 \wedge 0 \leq y_2 + y_3 \leq x \Rightarrow \text{T})$

Формулы 2а и 2б опровержимы: для 2а достаточно взять y_4 не равным x, а для 2б взять y_4 не равным -1. Посмотрев еще раз на рассмотренные только что вычисления, можно заметить, что, действительно, при $y_3 = 1$ выполнено, что $y_4 = x$, а при $y_3 = -1$ выполнено, что $y_4 = -1$. Тогда рассмотрим в качестве p такой предикат:

$(y_3 = 1 \Rightarrow y_1 = x\ (y_2 + 1) \wedge y_4 = x) \wedge (y_3 = -1 \Rightarrow y_1 = x^2 + y_2 \wedge y_4 = -1)$

Подставим его в указанные выше 4 формулы:

1. $x \geq 0 \Rightarrow (1 = 1 \Rightarrow 0 = x\ (-1 + 1) \wedge x = x) \wedge (1 = -1 \Rightarrow 0 = x^2 + (-1) \wedge x = -1)$
2. $x \geq 0 \wedge 0 \leq y_2 + y_3 \leq x \wedge (y_3 = 1 \Rightarrow y_1 = x\ (y_2 + 1) \wedge y_4 = x) \wedge (y_3 = -1 \Rightarrow y_1 = x^2 + y_2 \wedge y_4 = -1) \Rightarrow (y_3 = 1 \Rightarrow y_1 + y_4 = x\ (y_2 + y_3 + 1) \wedge y_4 = x) \wedge (y_3 = -1 \Rightarrow y_1 + y_4 = x^2 + y_2 + y_3 \wedge y_4 = -1)$

3. $x \geq 0 \wedge y_2 + y_3 > x \wedge (y_3 = 1 \Rightarrow y_1 = x\ (y_2 + 1) \wedge y_4 = x) \wedge (y_3 = -1 \Rightarrow y_1 = x^2 + y_2 \wedge y_4 = -1) \Rightarrow (-y_3 = 1 \Rightarrow y_1 = x\ (y_2 + 1) \wedge -1 = x) \wedge (-y_3 = -1 \Rightarrow y_1 = x^2 + y_2 \wedge -1 = -1)$

4. $x \geq 0 \wedge y_2 + y_3 < 0 \wedge (y_3 = 1 \Rightarrow y_1 = x\ (y_2 + 1) \wedge y_4 = x) \wedge (y_3 = -1 \Rightarrow y_1 = x^2 + y_2 \wedge y_4 = -1) \Rightarrow y_1 = x^2$

Первая формула, очевидно, истинна. Вторую формулу разобьем на три формулы, как это было сделано ранее:

2а. $y_3=1 \Rightarrow (x \geq 0 \wedge 0 \leq y_2 + 1 \leq x \wedge y_1 = x\ (y_2 + 1) \wedge y_4 = x \Rightarrow y_4=x)$

2б. $y_3=-1 \Rightarrow (x \geq 0 \wedge 0 \leq y_2 - 1 \leq x \wedge y_1 = x^2 + y_2 \wedge y_4 = -1 \Rightarrow y_4 = -1)$

2в. $y_3 \neq 1 \wedge y_3 \neq -1 \Rightarrow (x \geq 0 \wedge 0 \leq y_2 + y_3 \leq x \Rightarrow \text{T})$

Все три формулы истинны. Сделаем теперь то же с третьей формулой:

3а. $y_3 = 1 \Rightarrow (x \geq 0 \wedge y_2 + 1 > x \wedge y_1 = x\ (y_2 + 1) \wedge y_4 = x \Rightarrow y_1 = x^2 + y_2)$

3б. $y_3 = -1 \Rightarrow (x \geq 0 \wedge y_2 - 1 > x \wedge y_1 = x^2 + y_2 \wedge y_4 = -1 \Rightarrow y_1 = x\ (y_2 + 1) \wedge -1 = x)$

3в. $y_3 \neq 1 \wedge y_3 \neq -1 \Rightarrow (x \geq 0 \wedge y_2 + y_3 > x \Rightarrow \text{T})$

Формула 3а опровержима, контрпример: x = 3, y_1 = 15, y_2 = 4, y_3 = 1, y_4 = 3. Анализируем вычисление при x = 3. Получаем, что y_2 не может принимать значение 4, т. к. он не должен быть больше x. Если добавить в индуктивное утверждение условие $y_2 \leq x$, то формулы 3а и 3б становятся истинными. Формула 3в осталась истинной. Тем самым, предлагаем в качестве нового индуктивного утверждения следующее:

$y_2 \le x \ \wedge (y_3 = 1 \Rightarrow y_1 = x\ (y_2 + 1) \wedge y_4 = x) \wedge (y_3 = -1 \Rightarrow y_1 = x^2 + y_2 \wedge y_4 = -1)$

Подставляем в условия верификации:

1. $x \ge 0 \Rightarrow -1 \le x \ \wedge (1 = 1 \Rightarrow 0 = x\ (-1 + 1) \wedge x = x) \wedge (1 = -1 \Rightarrow 0 = x^2 - 1 \wedge x = -1)$
2. $x \ge 0 \ \wedge 0 \le y_2 + y_3 \le x \wedge y_2 \le x \ \wedge (y_3 = 1 \Rightarrow y_1 = x\ (y_2 + 1) \wedge y_4 = x) \wedge (y_3 = -1 \Rightarrow y_1 = x^2 + y_2 \wedge y_4 = -1) \Rightarrow y_2 + y_3 \le x \ \wedge (y_3 = 1 \Rightarrow y_1 + y4 = x\ (y_2 + y_3 + 1) \wedge y_4 = x) \wedge (y_3 = -1 \Rightarrow y_1 + y4 = x^2 + y_2 + y_3 \ \wedge y_4 = -1)$
3. $x \ge 0 \ \wedge y_2 + y_3 > x \wedge y_2 \le x \ \wedge (y_3 = 1 \Rightarrow y_1 = x\ (y_2 + 1) \wedge y_4 = x) \wedge (y_3 = -1 \Rightarrow y_1 = x^2 + y_2 \wedge y_4 = -1) \Rightarrow y_2 \le x \ \wedge (-y_3 = 1 \Rightarrow y_1 = x\ (y_2 + 1) \wedge -1 = x) \wedge (-y_3 = -1 \Rightarrow y_1 = x^2 + y_2 \wedge -1 = -1)$
4. $x \ge 0 \ \wedge y_2 + y_3 < 0 \wedge y_2 \le x \ \wedge (y_3 = 1 \Rightarrow y_1 = x\ (y_2 + 1) \wedge y_4 = x) \wedge (y_3 = -1 \Rightarrow y_1 = x^2 + y_2 \wedge y_4 = -1) \Rightarrow y_1 = x^2$

Первые три формулы истинны. Четвертую формулу разобьем на три подформулы:

4а. $y_3 = 1 \Rightarrow (x \ge 0 \ \wedge y_2 < -1 \ \wedge y_2 \le x \wedge y_1 = x\ (y_2 + 1) \wedge y_4 = x \Rightarrow y_1 = x^2)$

4б. $y_3 = -1 \Rightarrow (x \ge 0 \ \wedge y_2 < 1 \ \wedge y_2 \le x \wedge y_1 = x^2 + y_2 \wedge y_4 = -1 \Rightarrow y_1 = x^2)$

4в. $y_3 \ne 1 \ \wedge \ y_3 \ne -1 \Rightarrow (x \ge 0 \ \wedge y_2 + y_3 < 0 \ \wedge y_2 \le x \Rightarrow y_1 = x^2)$

Формула 4а ложна, контрпример: x = 2, $y_1 = -2$, $y_2 = -2$, $y_3 = 1$, $y_4 = 2$. Но этот контрпример не реализуется при вычислении x = 2, т. к. y_2 не может быть равен -2, он должен быть больше или равен -1. Формула 4б ложна даже при выполнении только что найденного условия для y_2, контрпример: x = 2, $y_1 = 3$, $y_2 = -1$, $y_3 = -1$, $y_4 = -1$. В этом вычислении при $y_3 = -1$ y_2 не может быть равен

-1. Формула 4в ложна, т. к. всегда можно подобрать значение y_1, не равное x^2, контрпример: x = 0, y_1 = 1, y_2 = 0, y_3 = -2, y_4 = 0. Этот пример не реализуется, т. к. y_3 не может равняться -2: из анализа вычислений следует, что y_3 всегда равно 1 или -1. Объединяя вышесказанное, составляем новый вариант индуктивного утверждения:

$-1 \le y_2 \le x \wedge ((y_3 = 1 \wedge y_1 = x\ (y_2 + 1) \wedge y_4 = x) \vee (y_3 = -1 \wedge y_1 = x^2 + y_2 \wedge y_4 = -1 \wedge y_2 \ge 0))$

Подставим это индуктивное утверждение в 4 формулы:

1. $x \ge 0 \Rightarrow -1 \le -1 \le x \wedge ((1 = 1 \wedge 0 = x\ (-1 + 1) \wedge x = x) \vee (1 = -1 \wedge 0 = x^2 - 1 \wedge x = -1 \wedge -1 \ge 0))$
2. $x \ge 0 \wedge 0 \le y_2 + y_3 \le x \wedge -1 \le y_2 \le x \wedge ((y_3 = 1 \wedge y_1 = x\ (y_2 + 1) \wedge y_4 = x) \vee (y_3 = -1 \wedge y_1 = x^2 + y_2 \wedge y_4 = -1 \wedge y_2 \ge 0)) \Rightarrow -1 \le y_2 + y_3 \le x \wedge ((y_3 = 1 \wedge y_1 + y_4 = x\ (y_2 + y_3 + 1) \wedge y_4 = x) \vee (y_3 = -1 \wedge y_1 + y_4 = x^2 + y_2 + y_3 \wedge y_4 = -1 \wedge y_2 + y_3 \ge 0))$
3. $x \ge 0 \wedge y_2 + y_3 > x \wedge -1 \le y_2 \le x \wedge ((y_3 = 1 \wedge y_1 = x\ (y_2 + 1) \wedge y_4 = x) \vee (y_3 = -1 \wedge y_1 = x^2 + y_2 \wedge y_4 = -1 \wedge y_2 \ge 0)) \Rightarrow -1 \le y_2 \le x \wedge ((-y_3 = 1 \wedge y_1 = x\ (y_2 + 1) \wedge -1 = x) \vee (-y_3 = -1 \wedge y_1 = x^2 + y_2 \wedge -1 = -1 \wedge y_2 \ge 0))$
4. $x \ge 0 \wedge y_2 + y_3 < 0 \wedge -1 \le y_2 \le x \wedge ((y_3 = 1 \wedge y_1 = x\ (y_2 + 1) \wedge y_4 = x) \vee (y_3 = -1 \wedge y_1 = x^2 + y_2 \wedge y_4 = -1 \wedge y_2 \ge 0)) \Rightarrow y_1 = x^2$

Все формулы истинны, подходящее индуктивное утверждение найдено.

2.3 *Задачи на метод индуктивных утверждений*

Во всех следующих задачах нужно доказать частичную корректность блок-схемы при помощи метода индуктивных утверждений

относительно спецификации (φ, ψ). Доменом всех переменных являются целые числа.

2.3.1 Блок-схема на рисунке 9, $\varphi(x) = (x \geq 0)$, $\psi(x, z) = (z \geq 0)$.

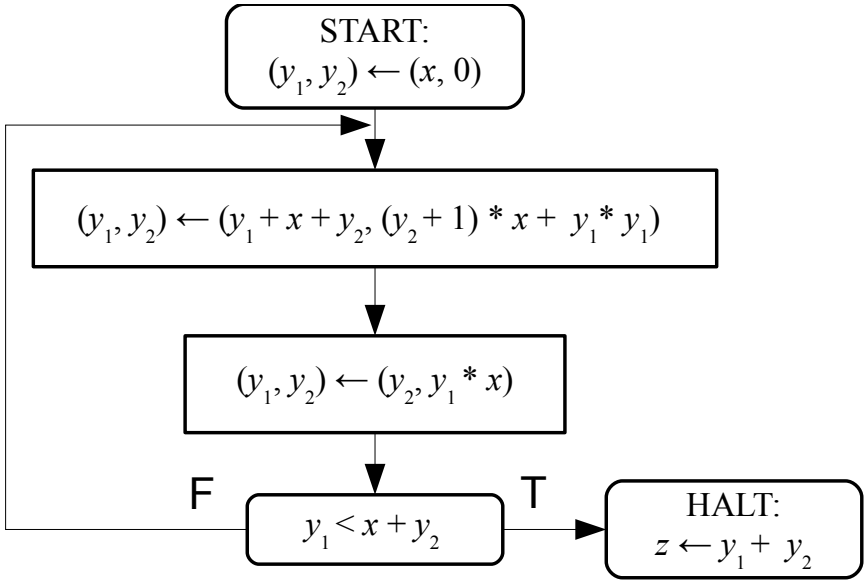


Рисунок 9. Рисунок к задаче

2.3.2 Блок-схема на рисунке 10, $\varphi(x) = (x > 0)$, $\psi(x, z) = (z > 0)$.

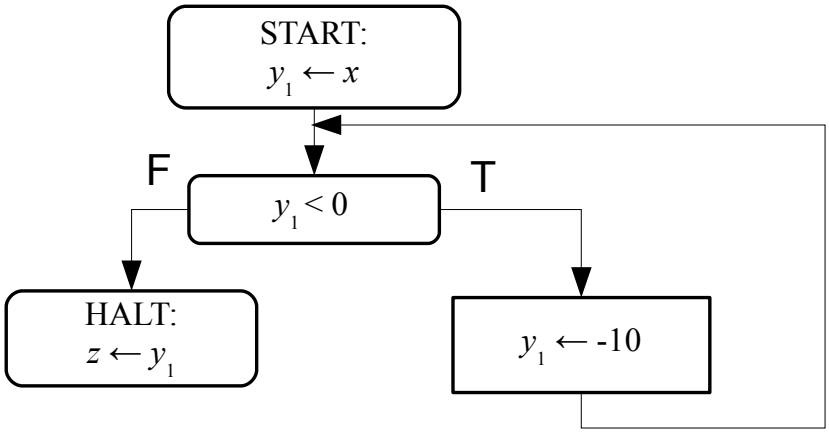


Рисунок 10. Рисунок к задаче

2.3.3 Блок-схема на рисунке 10, $\varphi(x) = (x < 0)$, $\psi(x, z) = (z < 0)$.

2.3.4 Блок-схема на рисунке 11, $\varphi(x_1, x_2) = (x_1 > x_2)$, $\psi(x_1, x_2, z_1, z_2) = (z_1 - x_1 = z_2 - x_2)$.

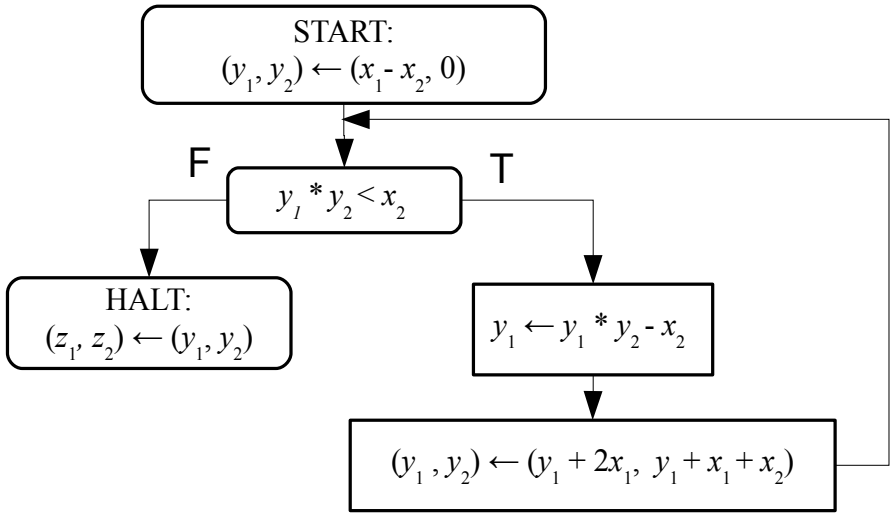


Рисунок 11. Рисунок к задаче

2.3.5 Блок-схема на рисунке 12, $\varphi(x) = (x \geq 0)$, $\psi(x, z) = (z = x^2)$.

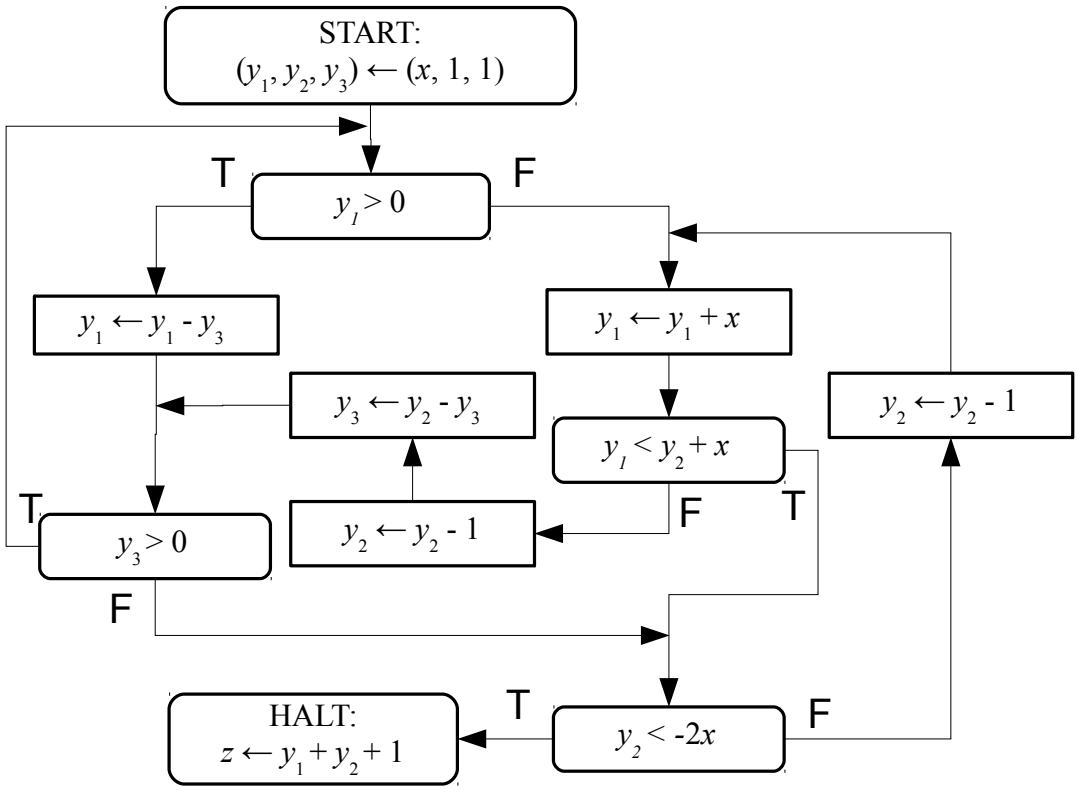


Рисунок 12. Рисунок к задаче

2.3.6 Блок-схема на рисунке 13, $\varphi(x) = (x \geq 0)$, $\psi(x, z) = (z^2 \leq x < (z+1)^2)$.

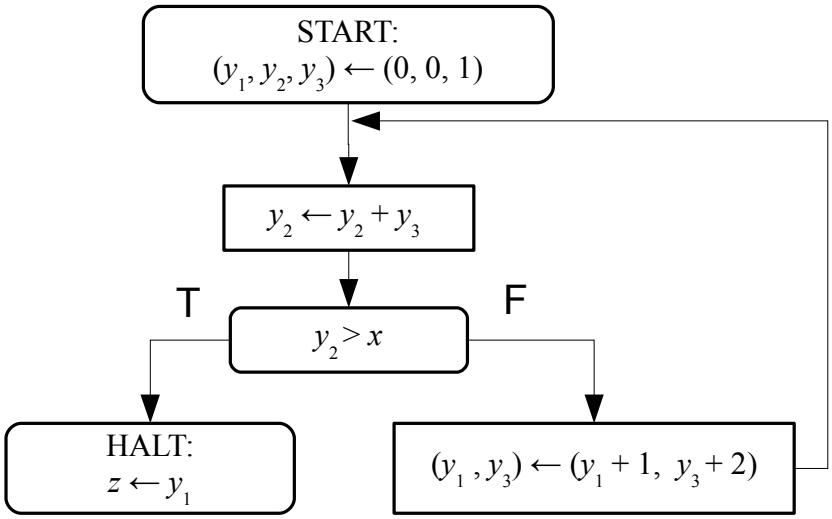


Рисунок 13. Рисунок к задаче

2.4 Задачи на метод фундированных множеств

Во всех задачах доменом всех переменных является множество целых чисел. Доказательство проводить при помощи метода фундированных множеств.

2.4.1 Доказать, что блок-схема на рисунке 14 всегда завершается.

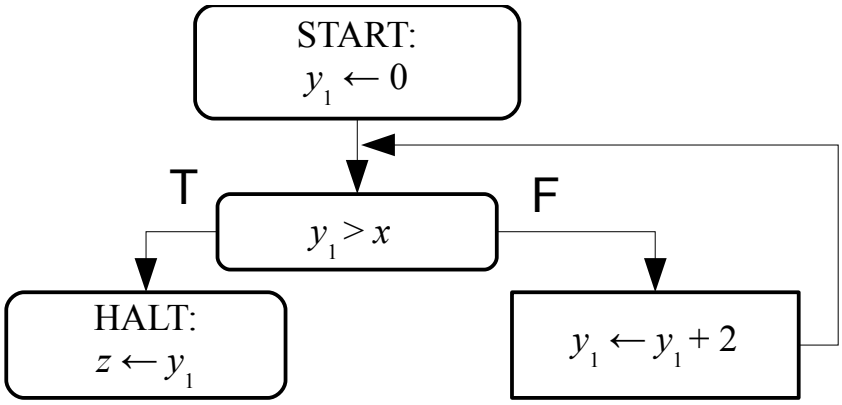


Рисунок 14. Рисунок к задаче

2.4.2 Доказать, что блок-схема на рисунке 15 завершается при всех неотрицательных значениях входной переменной x.

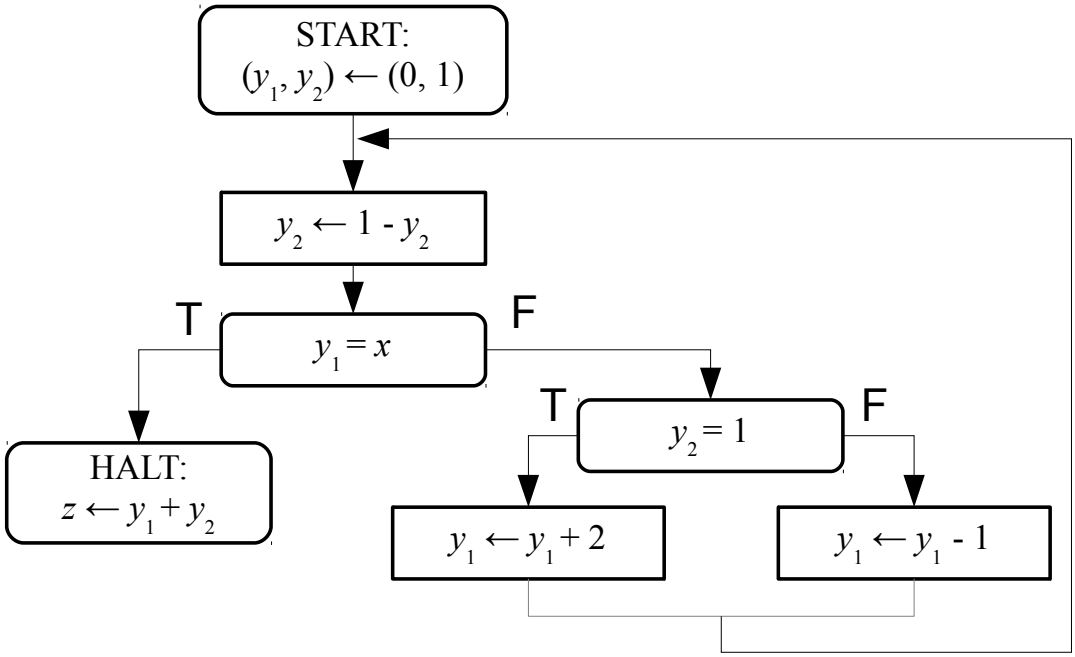


Рисунок 15. Рисунок к задаче

2.4.3 Доказать, что блок-схема на рисунке 16 завершается при всех неотрицательных значениях входной переменной x.

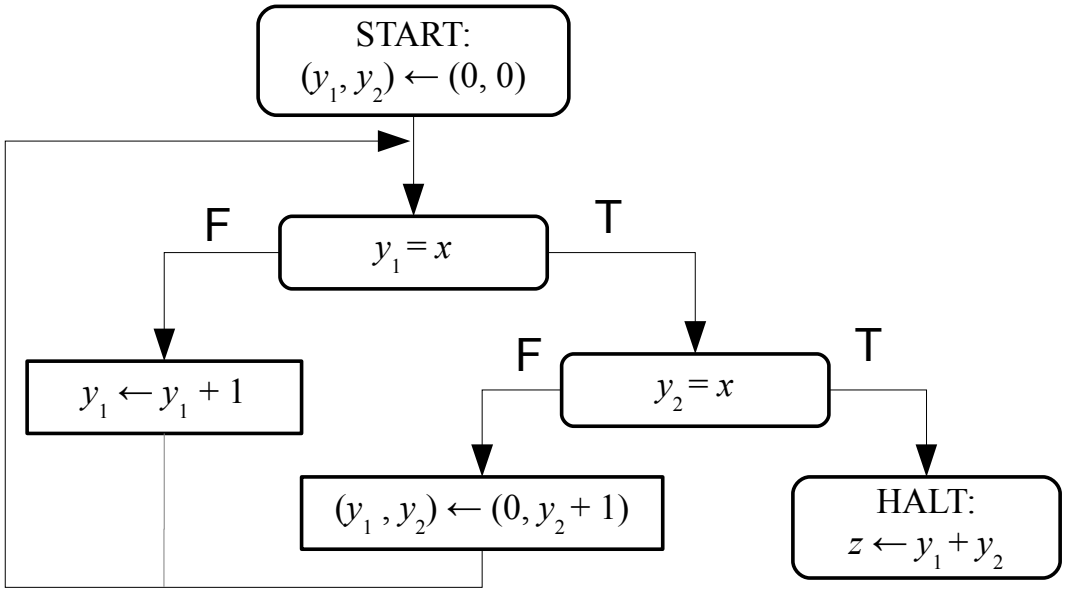


Рисунок 16. Рисунок к задаче

2.4.4 Доказать, что блок-схема на рисунке 17 завершается при всех неотрицательных значениях входной переменной x.

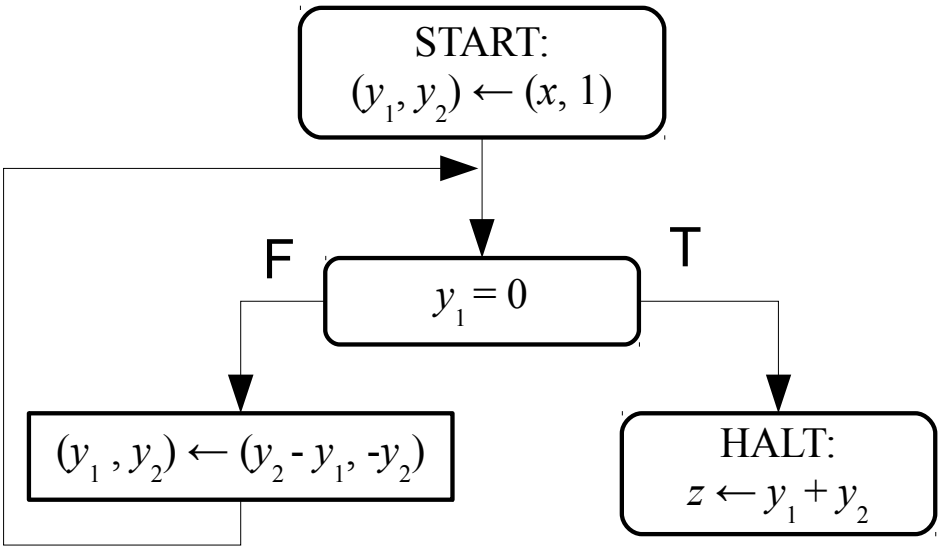


Рисунок 17. Рисунок к задаче

2.4.5 Доказать, что блок-схема на рисунке 18 завершается при всех значениях входной переменной $x>0$.

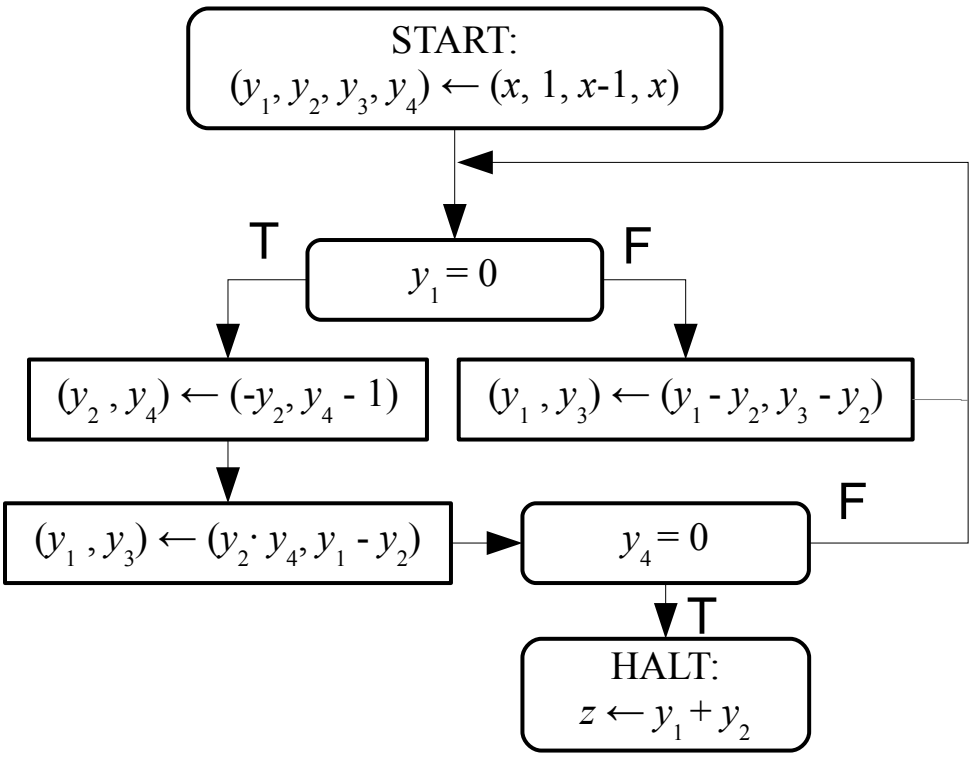


Рисунок 18. Рисунок к задаче

2.4.6 Доказать, что блок-схема на рисунке 19 завершается при всех неотрицательных значениях входной переменной x.

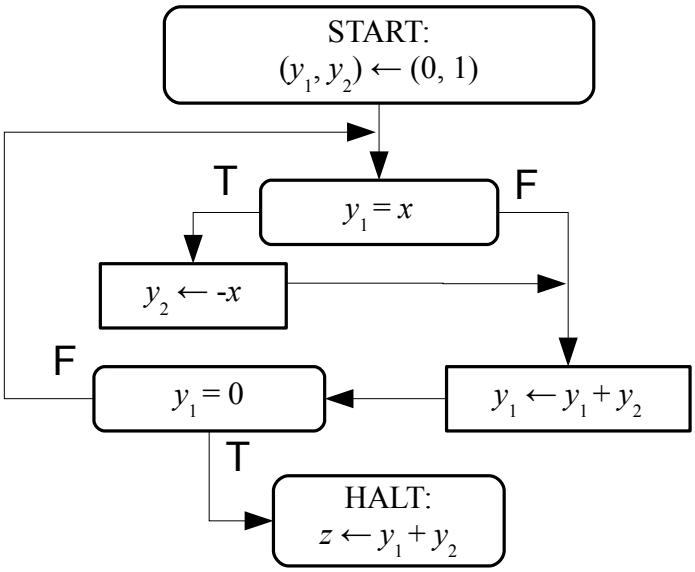


Рисунок 19. Рисунок к задаче

2.4.7 Доказать, что блок-схема на рисунке 20 завершается при всех положительных значениях входной переменной x.

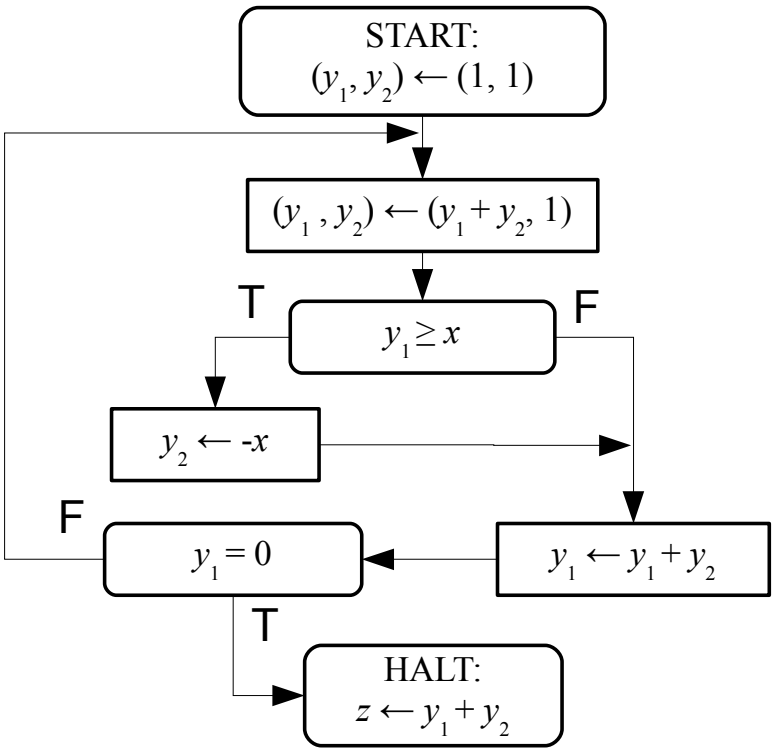


Рисунок 20. Рисунок к задаче

2.4.8 Доказать, что блок-схема на рисунке 21 завершается при всех неотрицательных значениях входной переменной x.

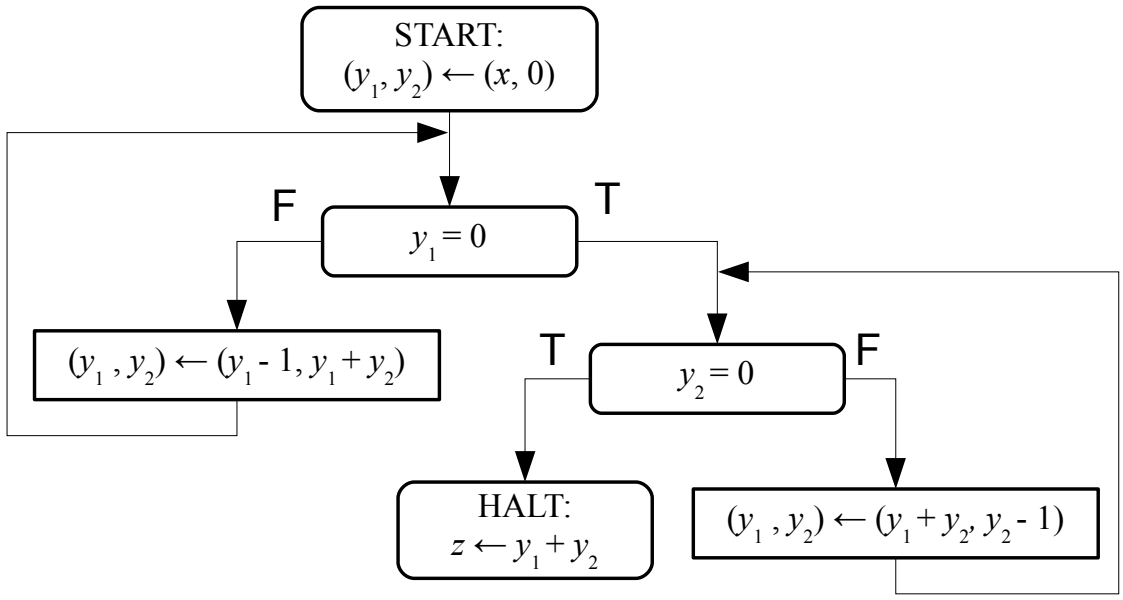


Рисунок 21. Рисунок к задаче

2.4.9 Доказать, что блок-схема на рисунке 22 всегда завершается.

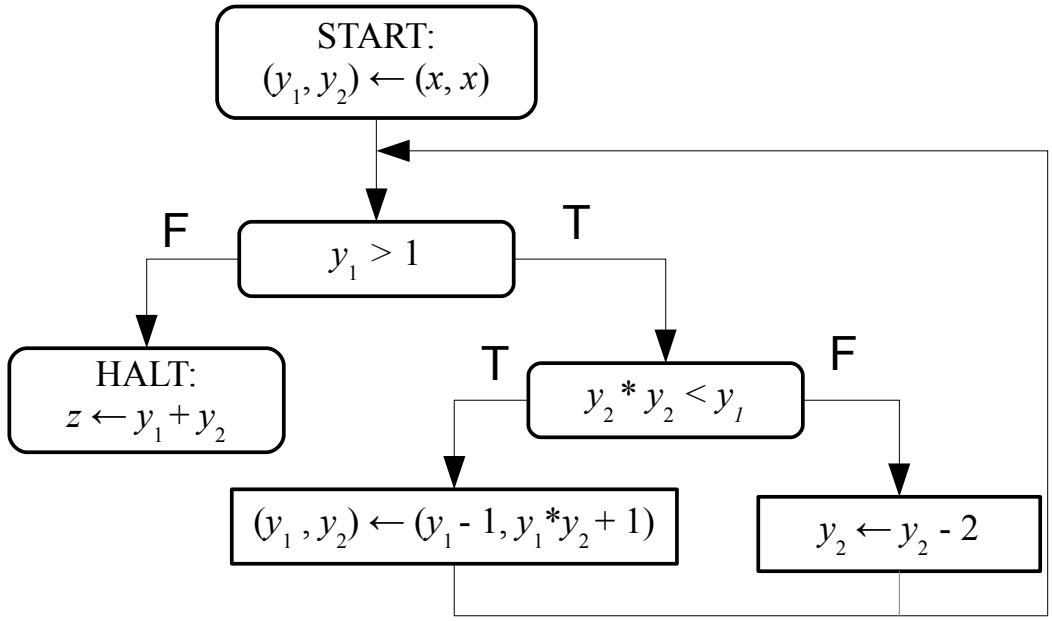
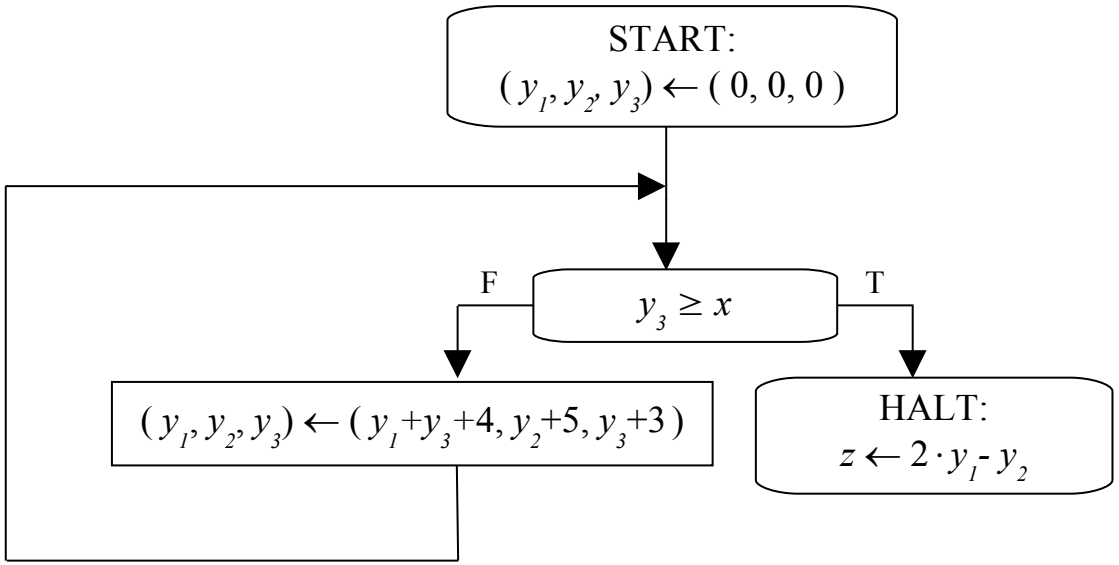


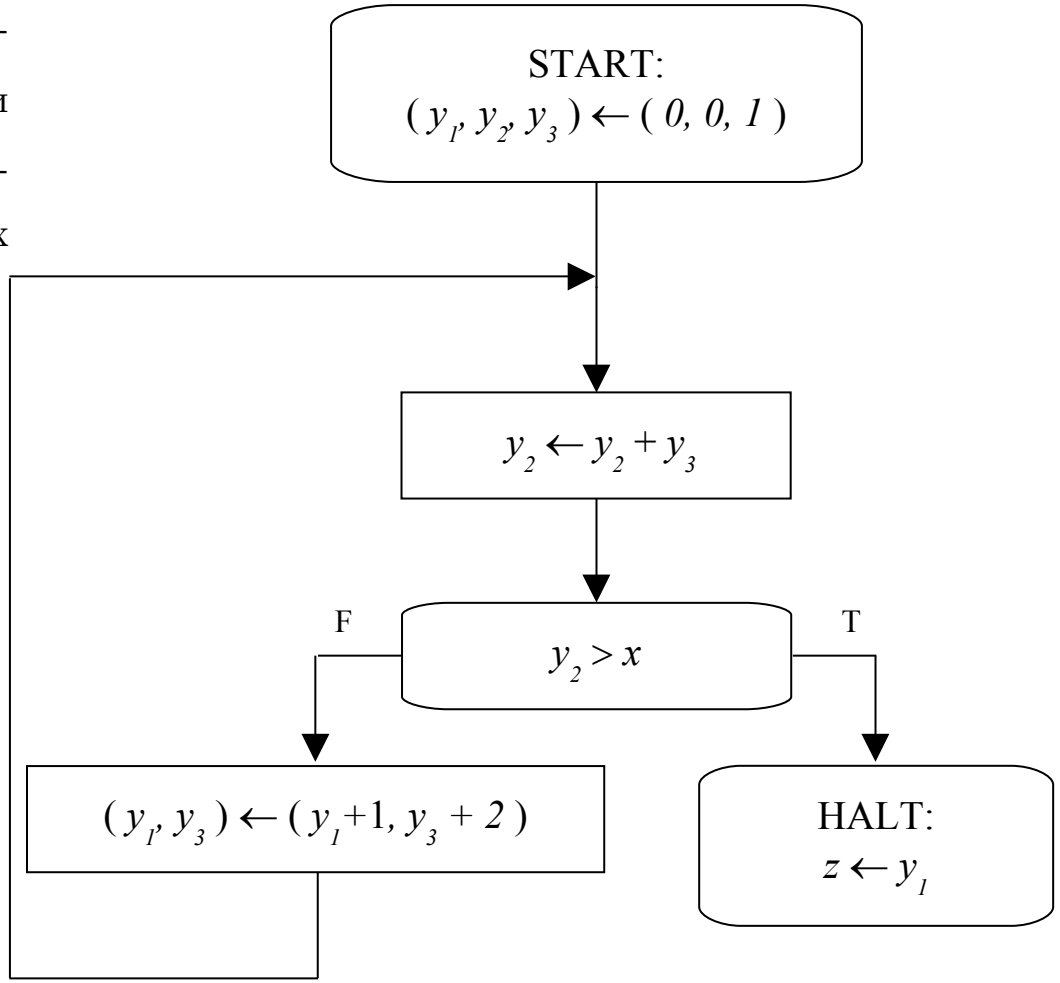
Рисунок 22. Рисунок к задаче

2.5 ***Общие задачи***

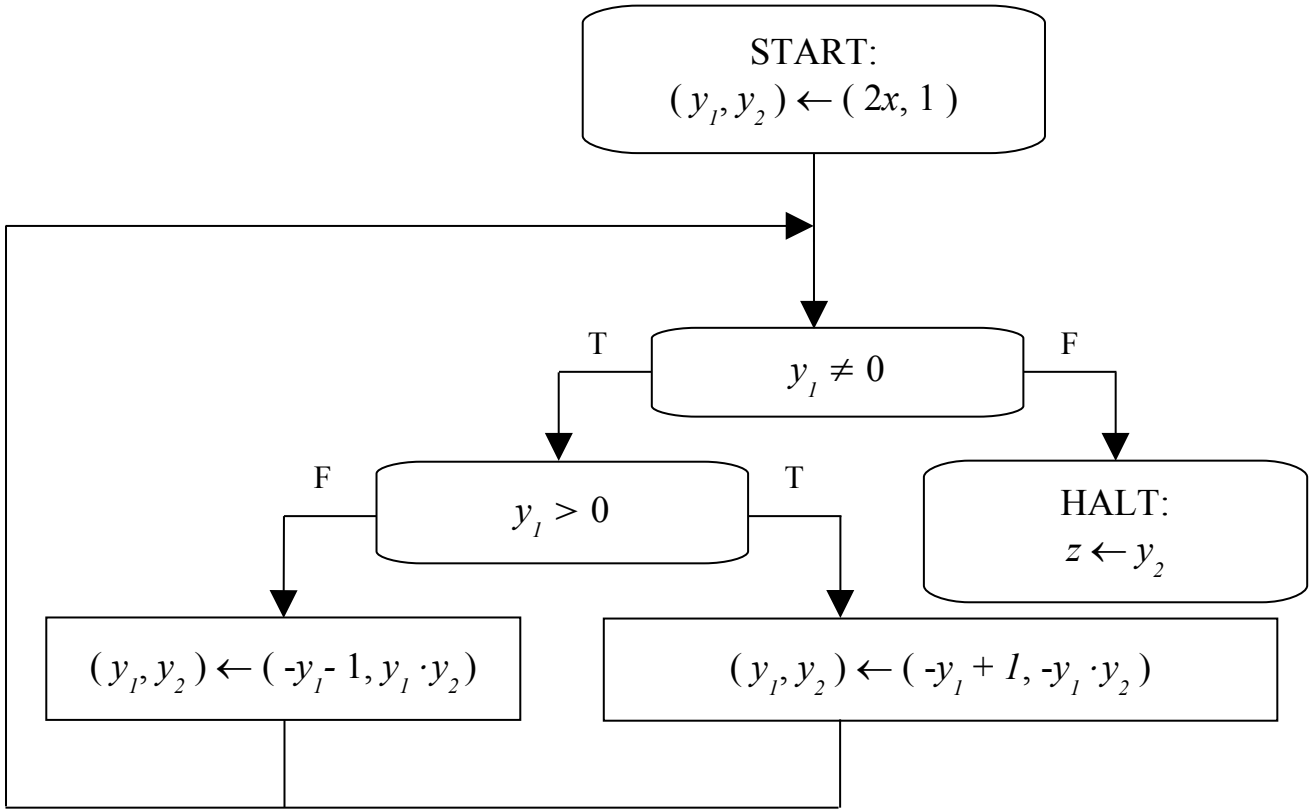
2.5.1 Доказать полную корректность программы относительно входного предиката $\varphi \equiv (x \geq 0) \wedge (x \vdots 3)$ и выходного предиката $\psi \equiv (3 \cdot z = x^2)$ при помощи методов Флойда. Доменом всех переменных является множество целых чисел.



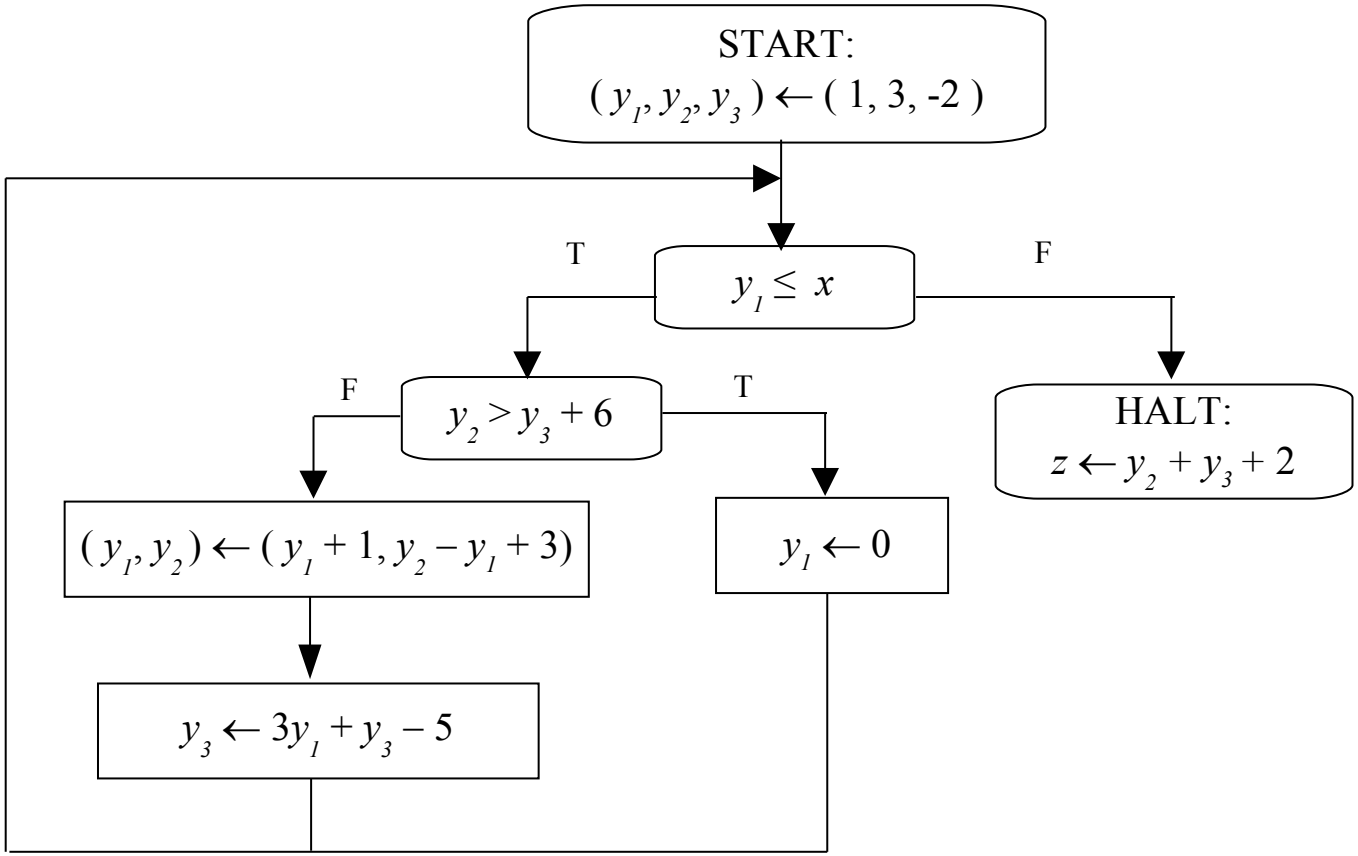
2.5.2 Доказать полную корректность программы относительно входного предиката $\varphi \equiv (x \geq 0)$ и выходного предиката $\psi \equiv (z^2 \leq x < (z + 1)^2)$. Доменом всех переменных является множество целых чисел.



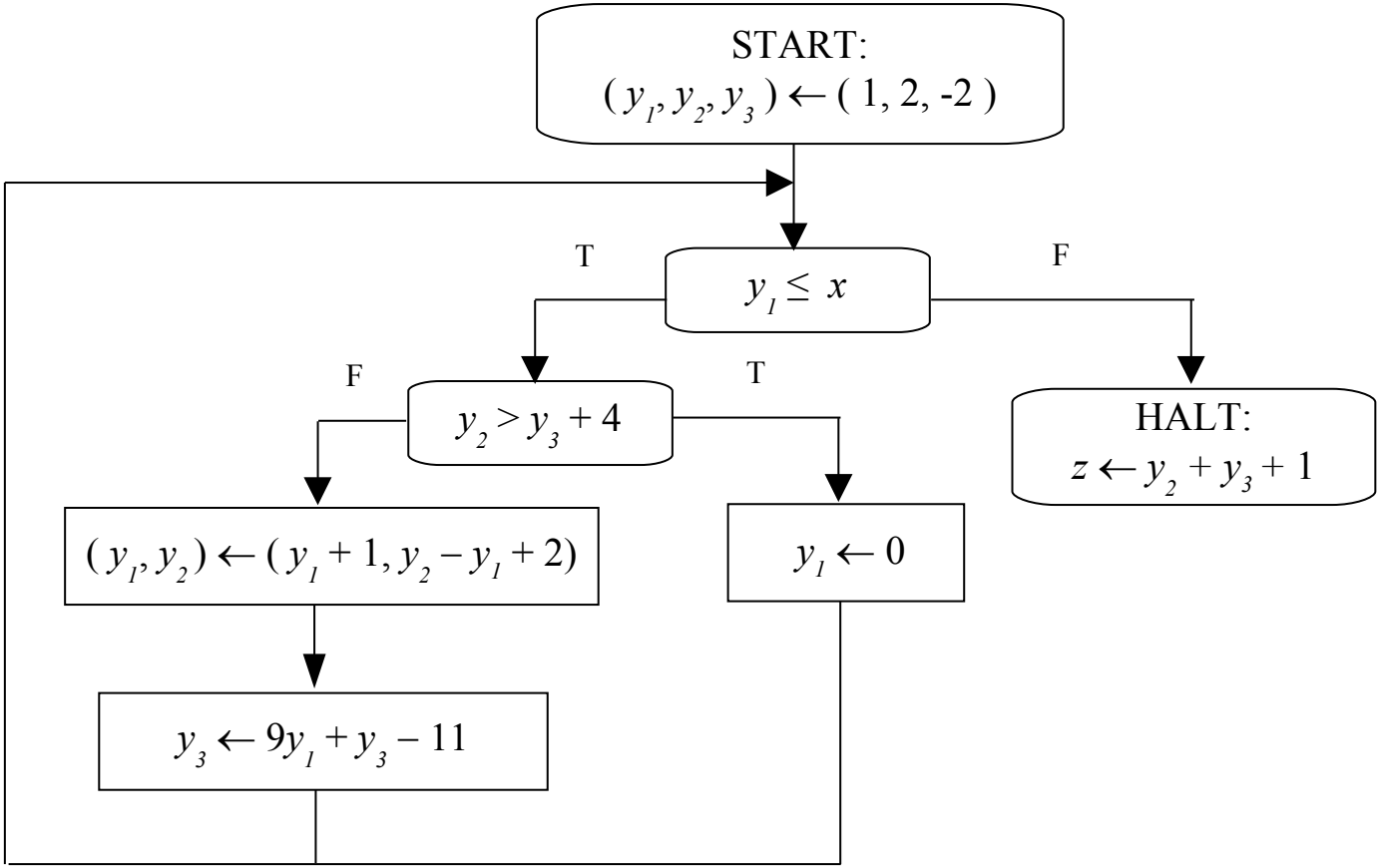
2.5.3 Доказать полную корректность программы относительно входного предиката $\varphi \equiv (x \geq 0)$ и выходного предиката $\psi \equiv (z = (2x)!)$ при помощи методов Флойда. Доменом всех переменных является множество целых чисел.



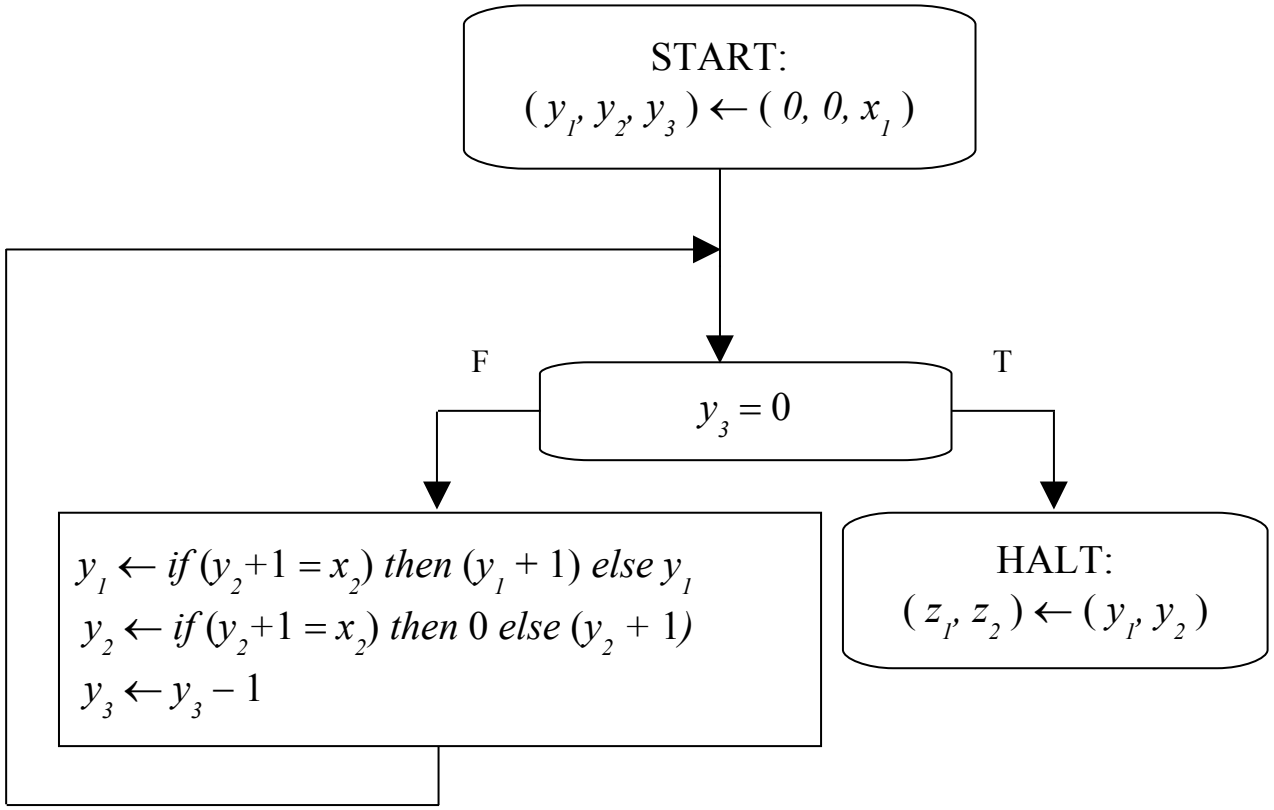
2.5.4 Доказать полную корректность программы относительно спецификации $\varphi \equiv (x \geq 0) \wedge (x \vdots 2)$ и $\psi \equiv (\ z = x^2 + 2x + 3)$. Доменом всех переменных является множество целых чисел.



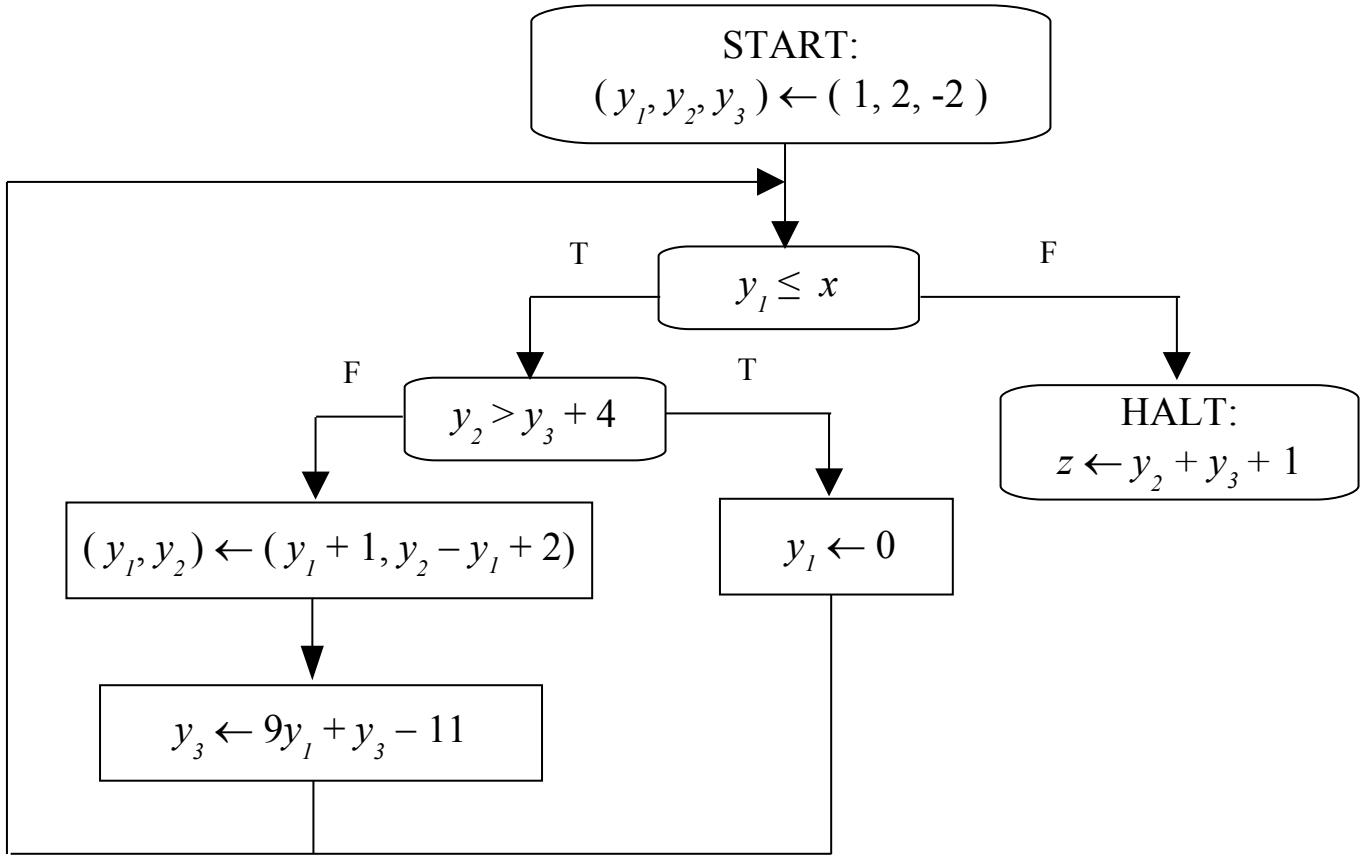
2.5.5 Доказать полную корректность программы относительно спецификации $\varphi \equiv (x \geq 1) \wedge (x \vdots 3)$ и $\psi \equiv (z = (2x + 1)^2)$. Доменом всех переменных является множество целых чисел.



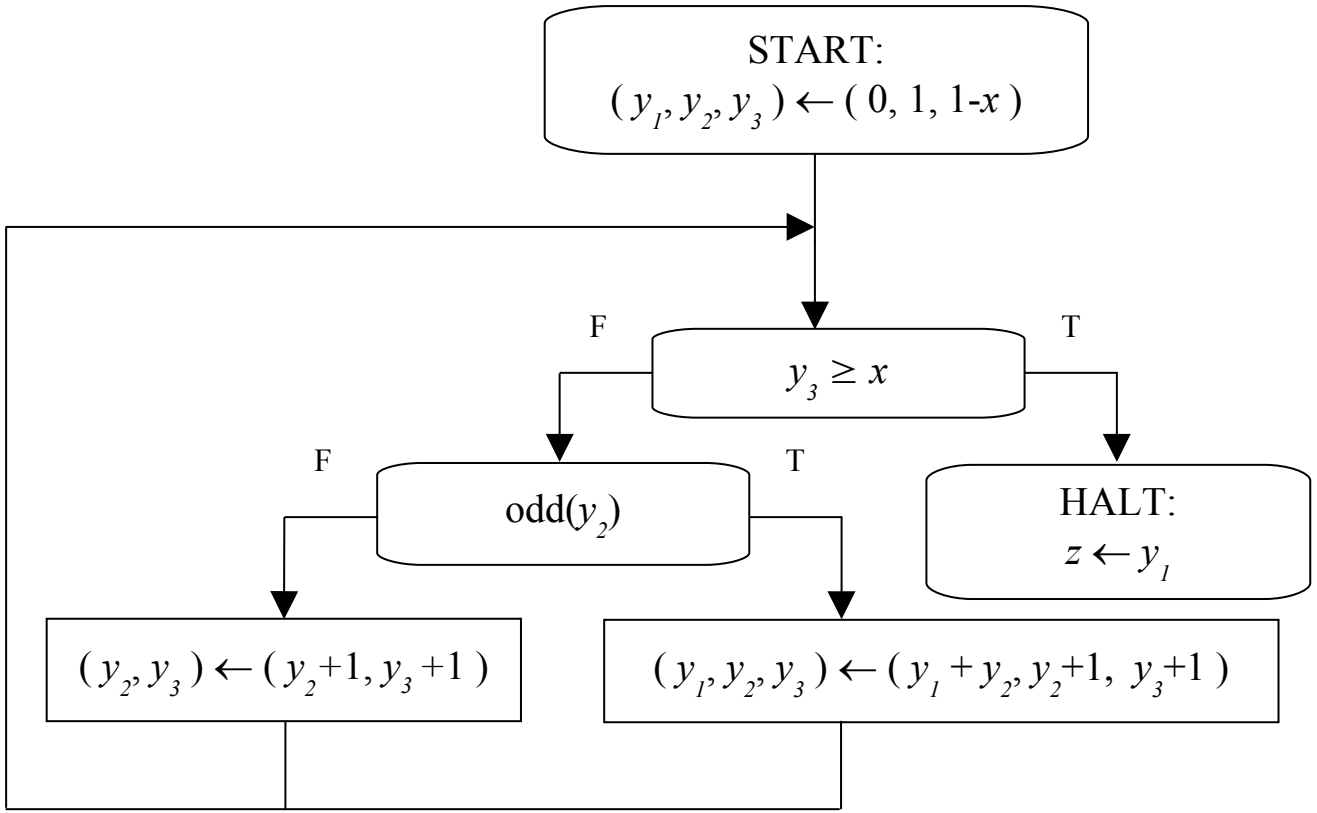
2.5.6 Доказать полную корректность программы относительно входного предиката $\varphi \equiv (x_1 > 0) \wedge (x_2 > 0)$ и выходного предиката $\psi \equiv (0 \le z_2 < x_2) \wedge (x_1 = z_1 \cdot x_2 + z_2)$. Доменом всех переменных является множество целых чисел.



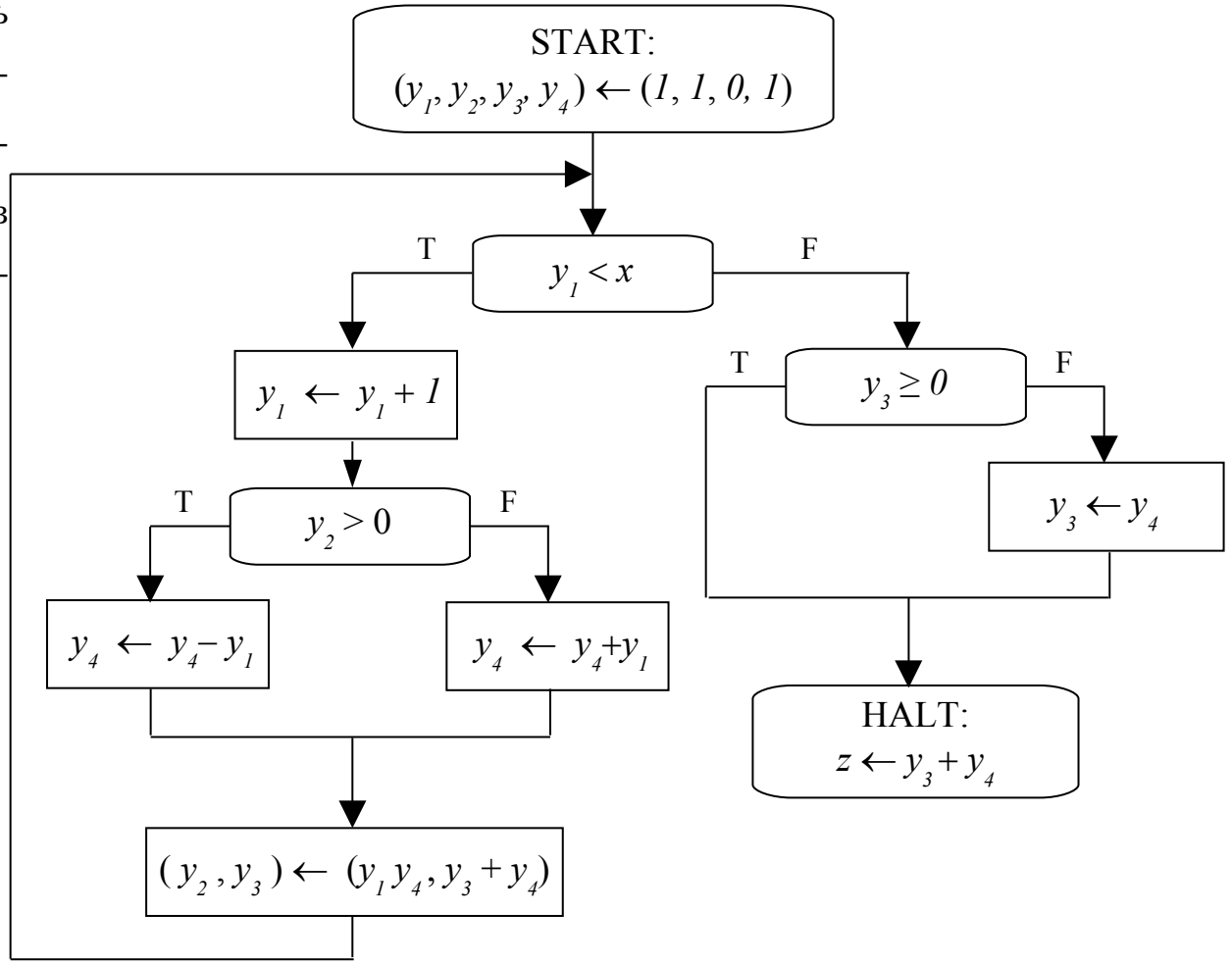
2.5.7 Доказать полную корректность программы относительно спецификации $\varphi \equiv (x \geq 3)$ и $\psi \equiv (z = (x + 2)^2)$. Доменом всех переменных является множество целых чисел.



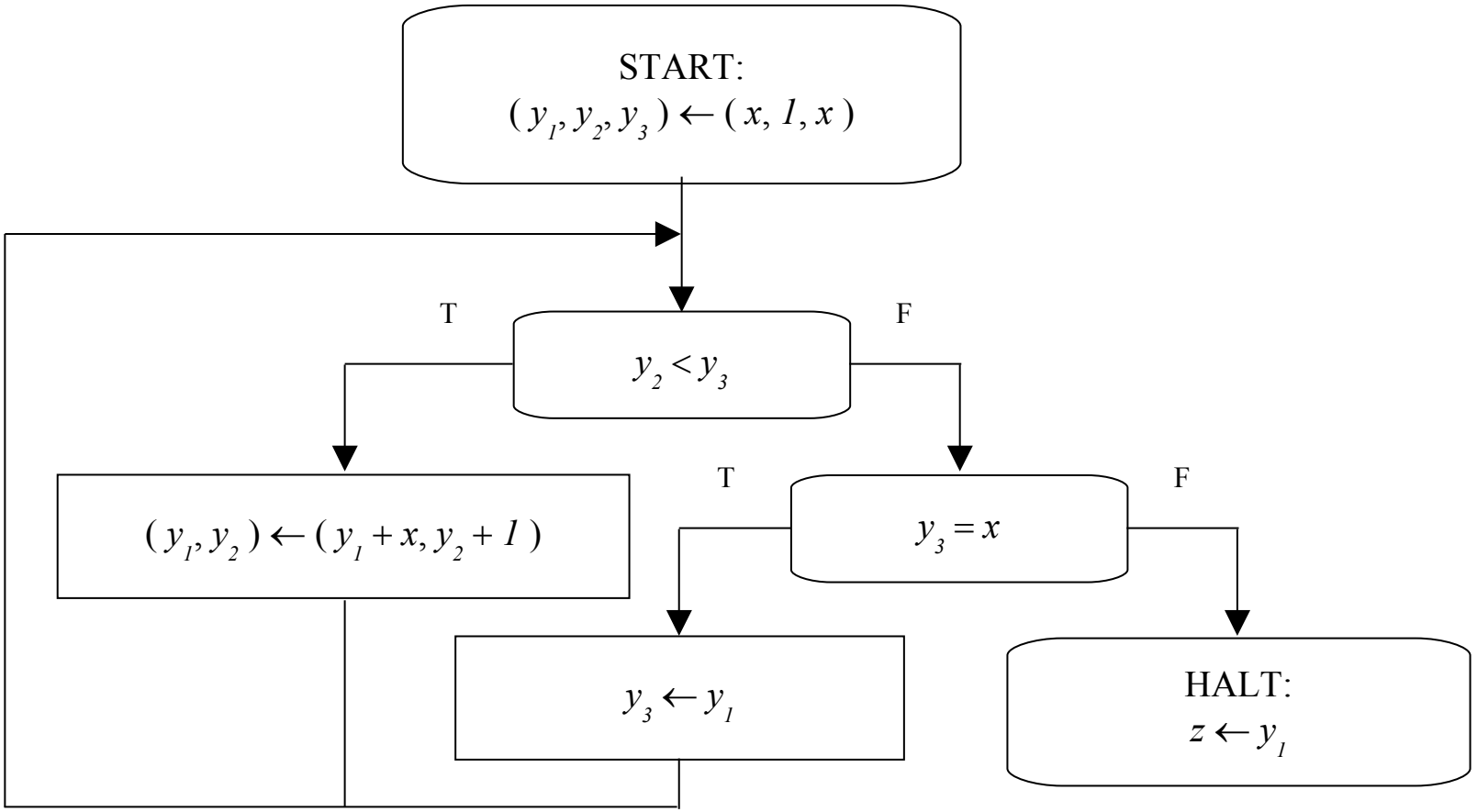
2.5.8 Доказать полную корректность программы относительно входного предиката $\varphi \equiv (x \geq 0)$ и выходного предиката $\psi \equiv (z = x^2)$. Доменом всех переменных является множество целых чисел. Оператор $\mathrm{odd}(y)$ принимает истинное значение, при нечетном значении аргумента y.



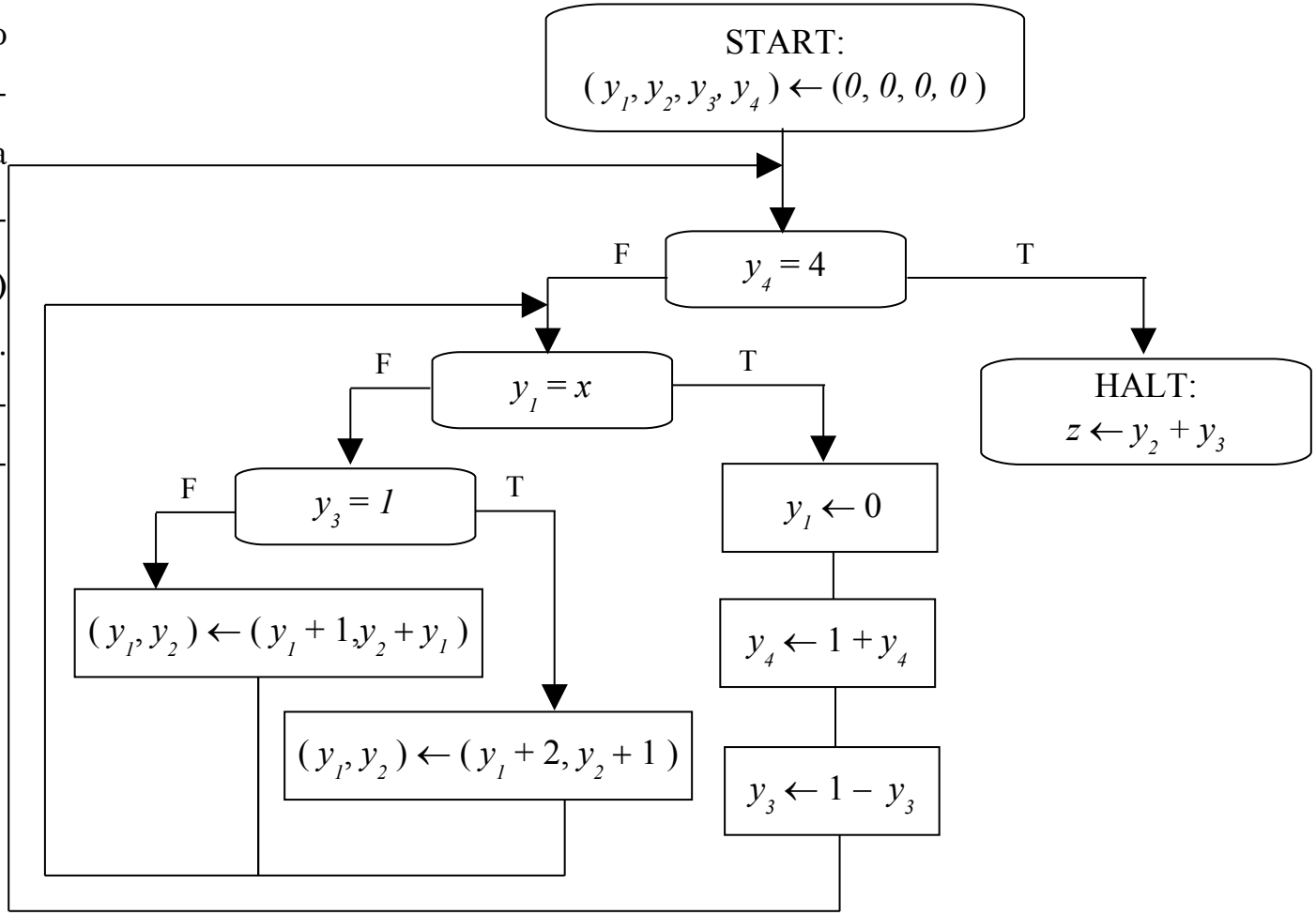
2.5.9 Доказать полную корректность программы относительно входного предиката $\varphi \equiv (x > 0)$ и выходного предиката $\psi \equiv (\,|z| = |x|\,)$ при помощи методов Флойда. Доменом всех переменных является множество целых чисел.



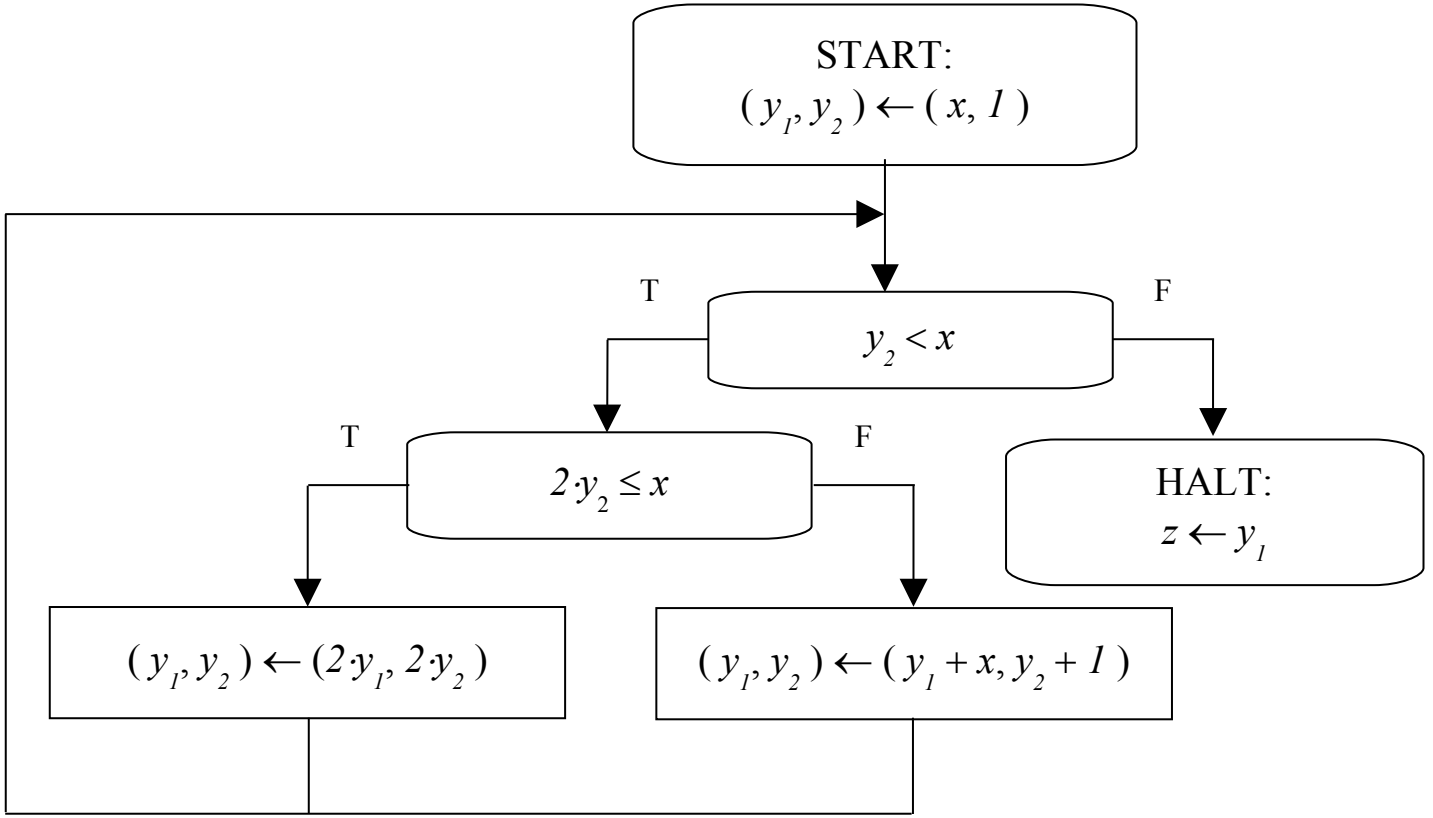
2.5.10 Доказать полную корректность программы относительно входного предиката $\varphi \equiv (x > 1)$ и выходного предиката $\psi \equiv (z = x^3)$. Доменом всех переменных является множество целых чисел.



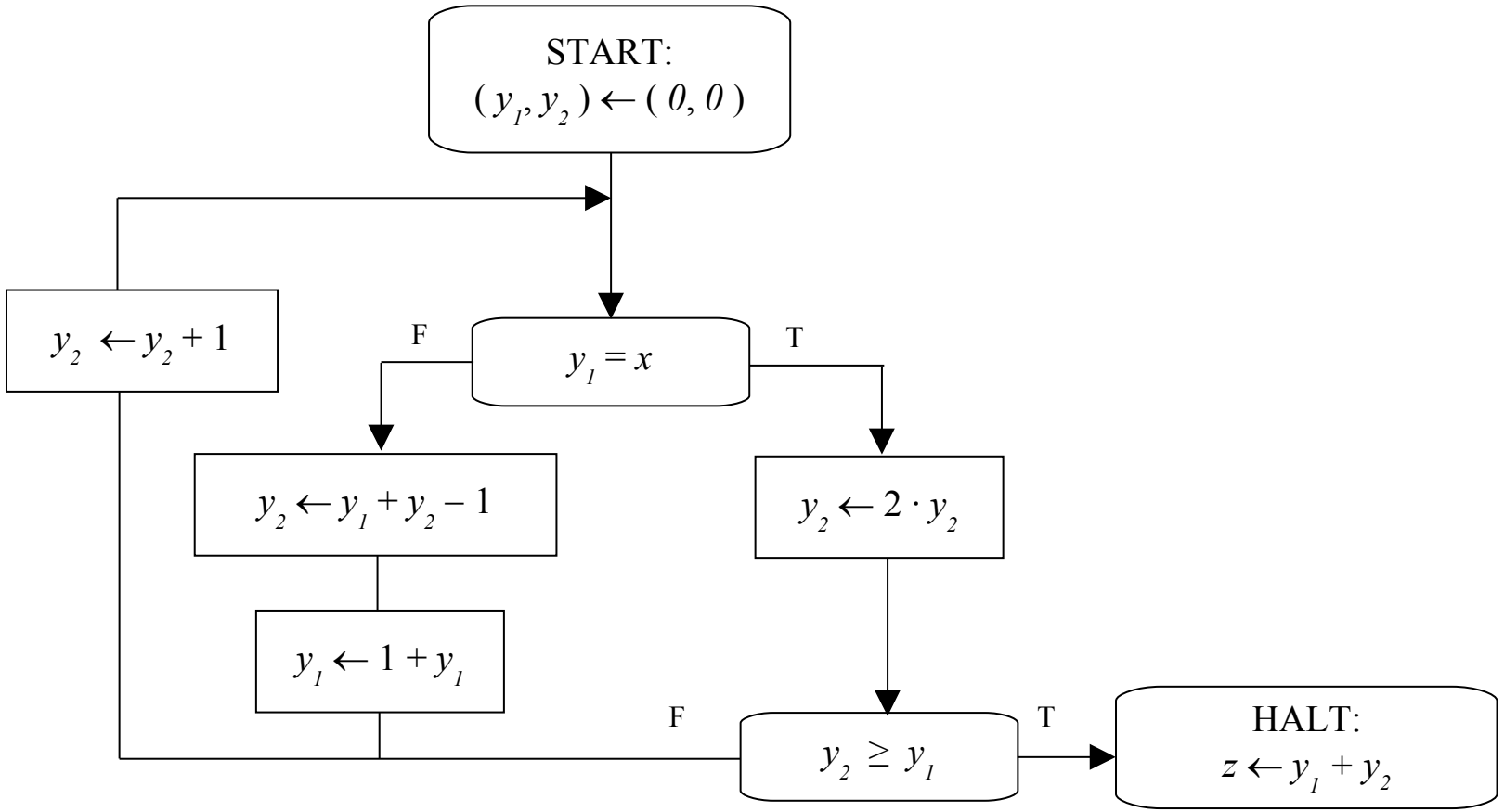
2.5.11 Доказать полную корректность программы относительно входного предиката $\varphi \equiv (x \geq 0) \wedge (x \vdots 2)$ и выходного предиката $\psi \equiv (z = x^2)$ при помощи методов Флойда. Доменом всех переменных является множество целых чисел.



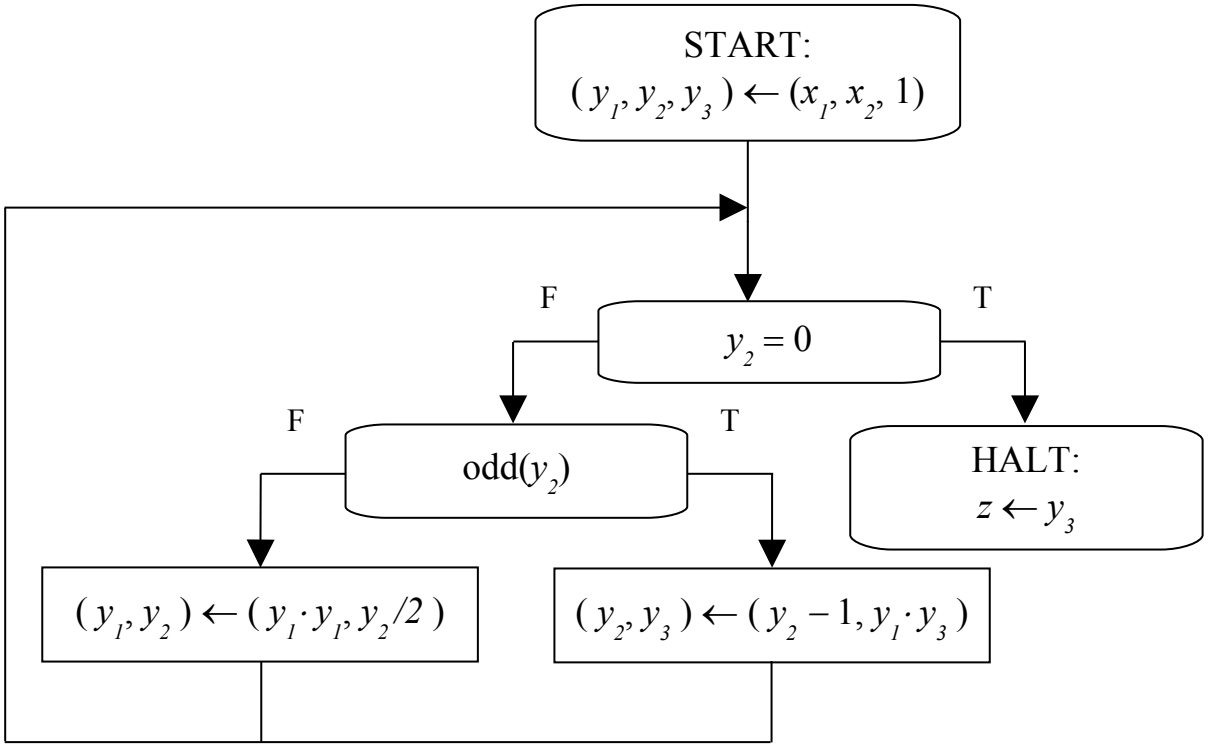
2.5.12 Доказать полную корректность программы относительно входного предиката $\varphi \equiv (x \geq 0)$ и выходного предиката $\psi \equiv (z = x^2)$ при помощи методов Флойда. Доменом всех переменных является множество целых чисел.



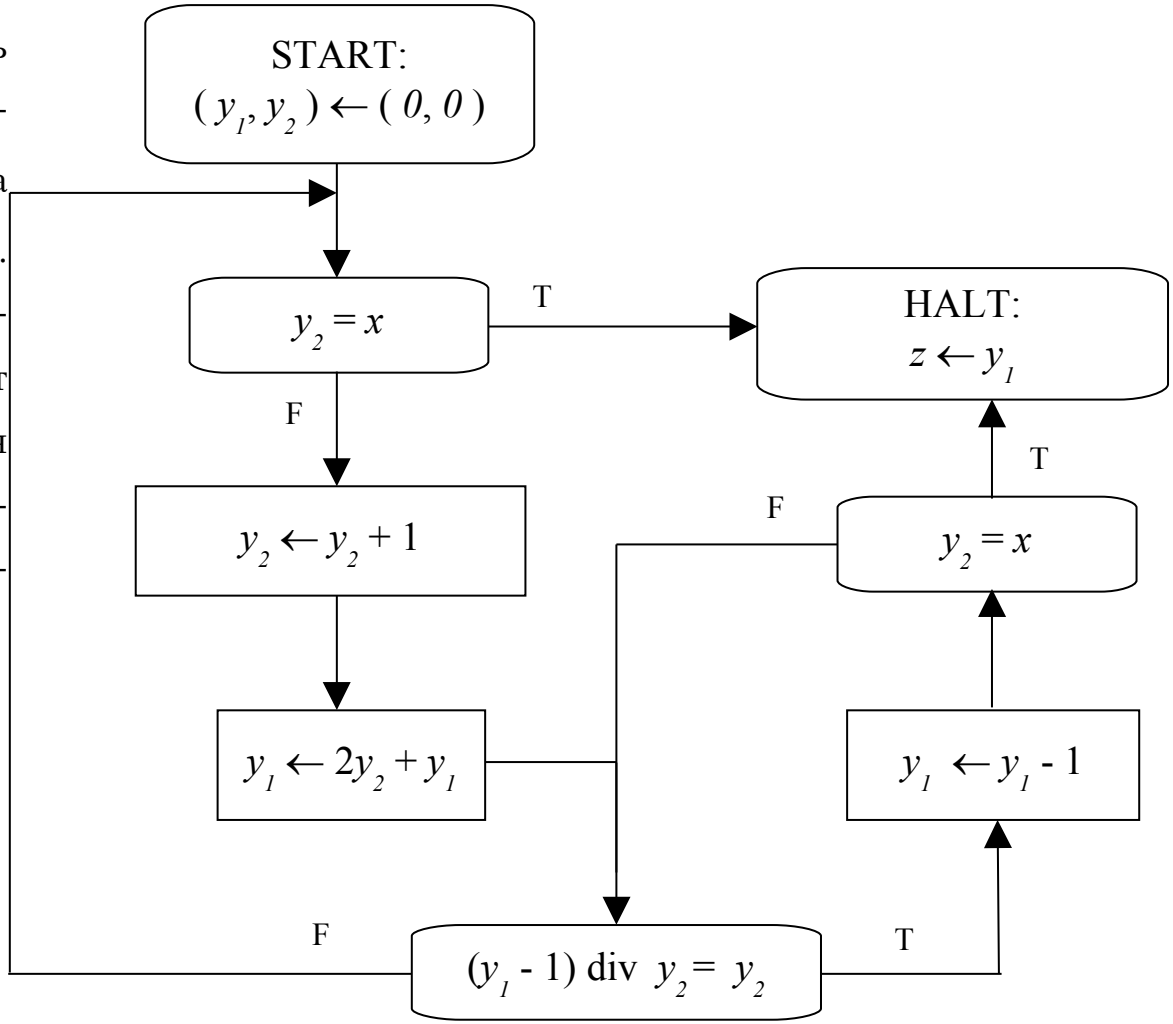
2.5.13 Доказать полную корректность программы относительно спецификации $\varphi \equiv (x \geq 0)$ и $\psi \equiv$ (if $x = 1$ then $z = 3$ else $z = x^2$). Доменом всех переменных является множество целых чисел.



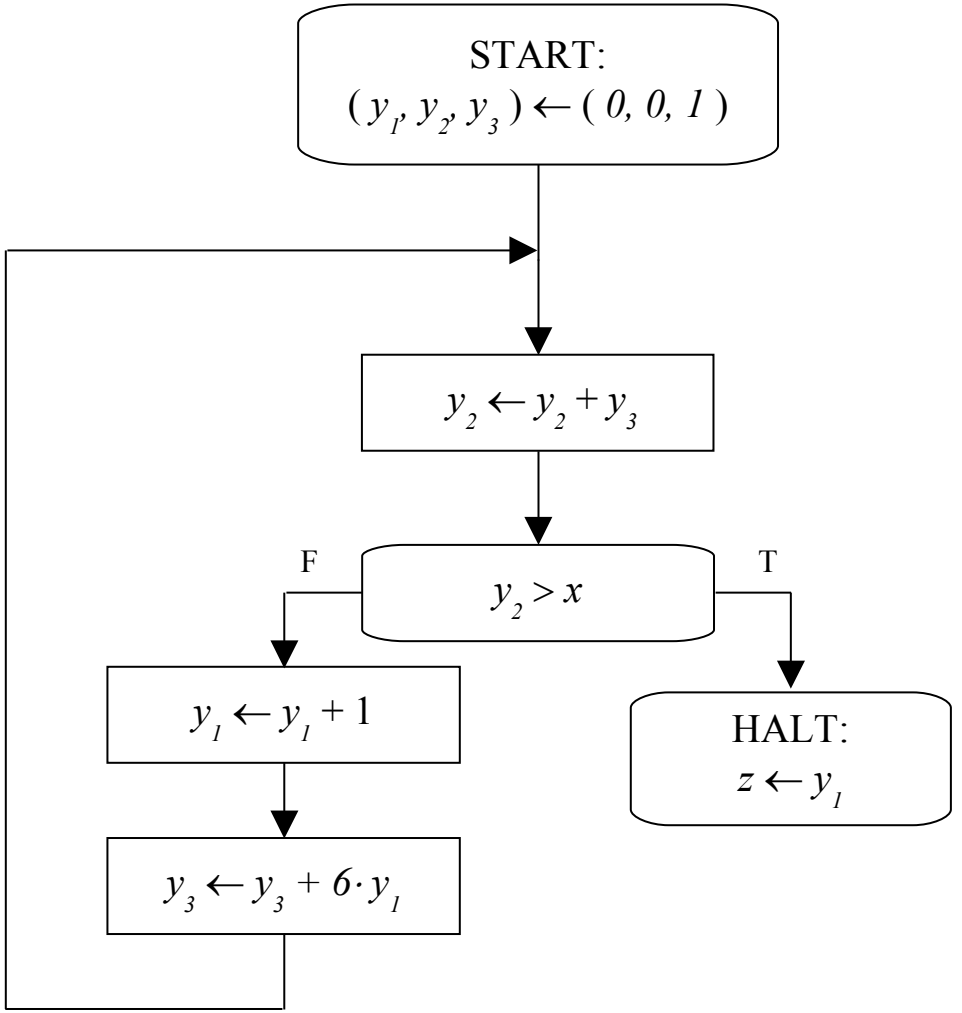
2.5.14 Доказать полную корректность программы относительно входного предиката $\varphi \equiv (x_2 \geq 0)$ и выходного предиката $\psi \equiv (z = x_1^{x_2})$ при помощи методов Флойда. 0^0 считается равным 1. Доменом всех переменных является множество целых чисел. Оператор odd(y) принимает истинное значение, при нечетном значении аргумента y.



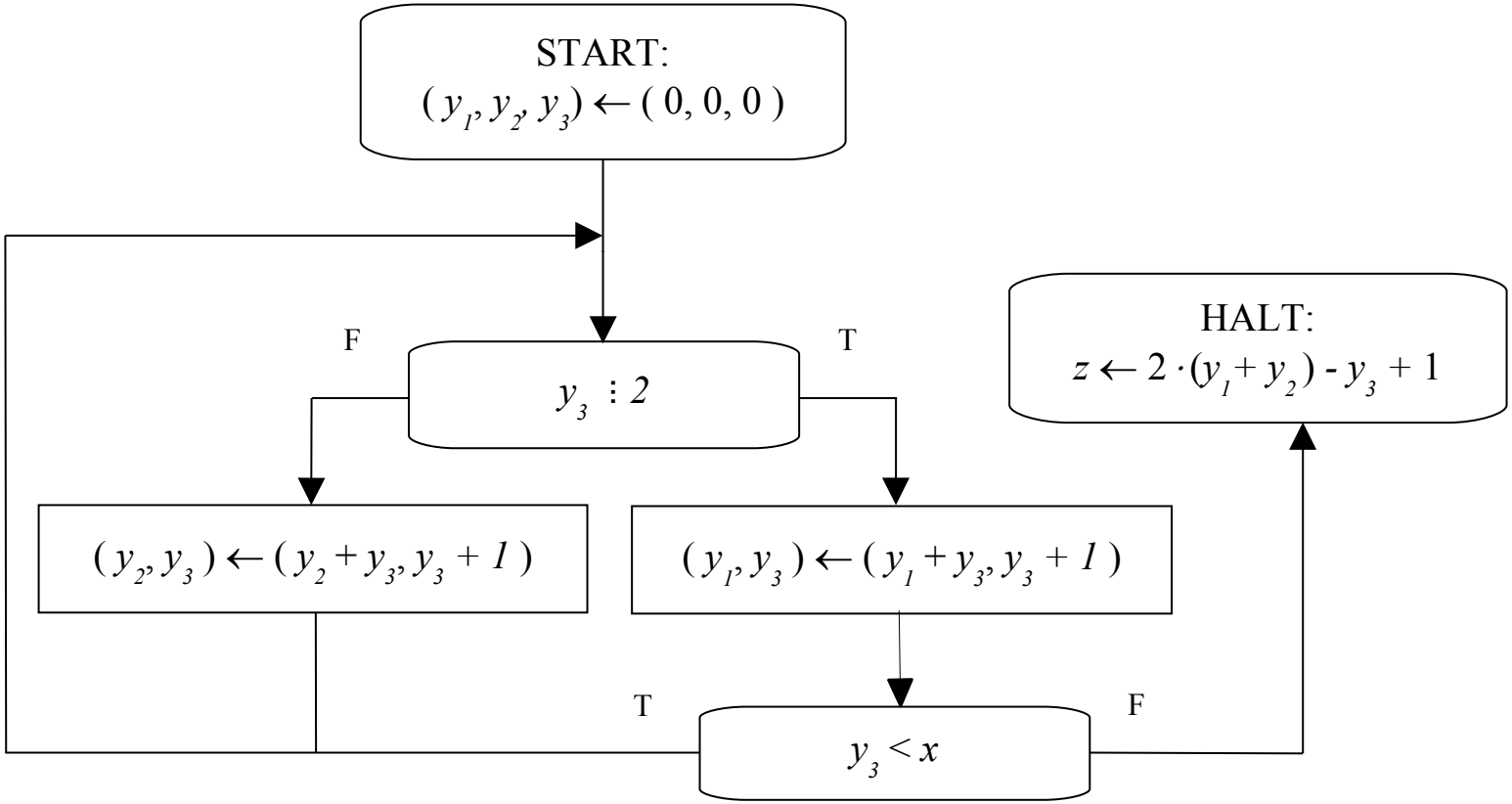
2.5.15 Доказать полную корректность программы относительно входного предиката $\varphi \equiv (x \geq 0)$ и выходного предиката $\psi \equiv (z = x^2)$ при помощи методов Флойда. Доменом всех переменных является множество целых чисел. A div B обозначает операцию получения частного от деления целого A на целое B. Эта операция определена только для неотрицательных A и положительных B.



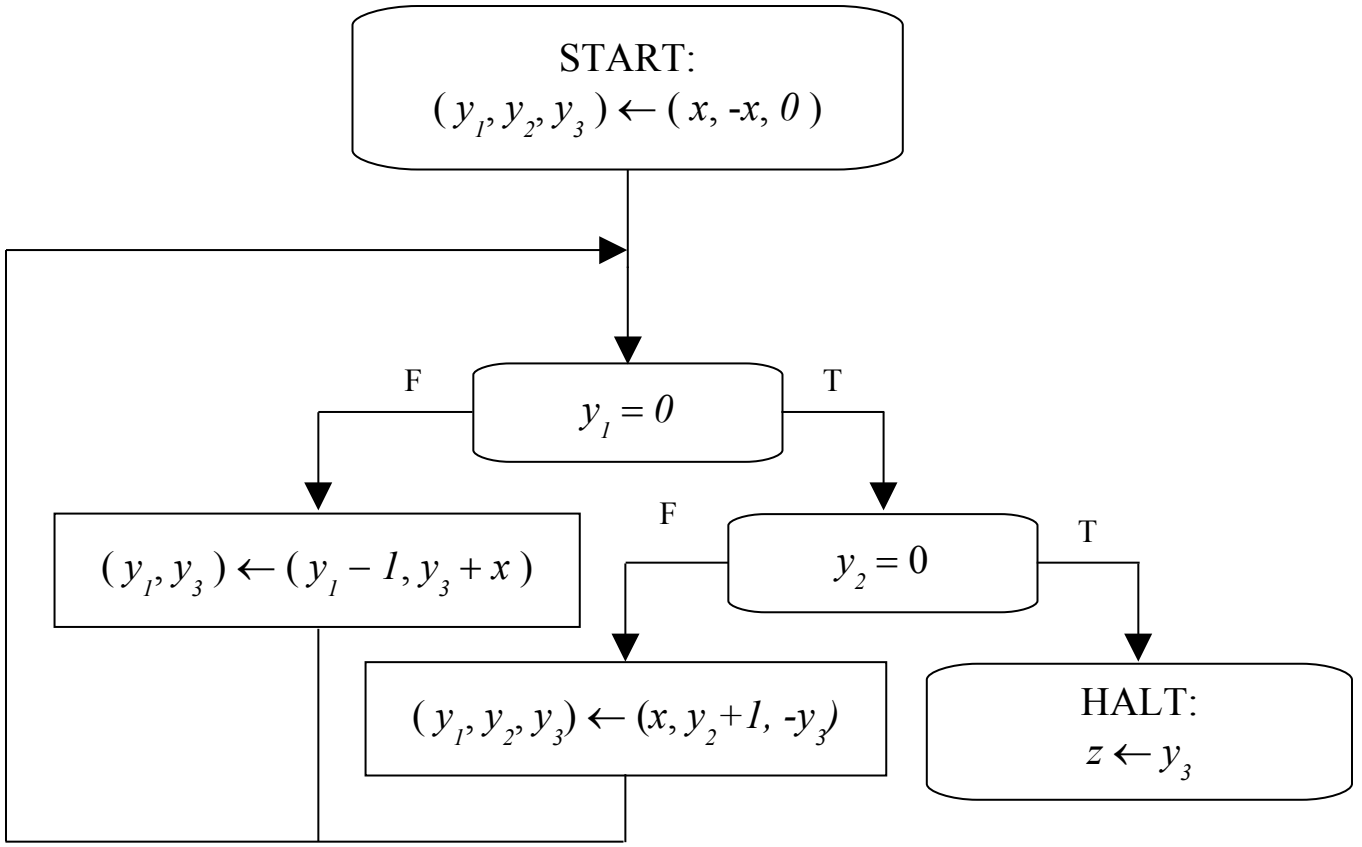
2.5.16 Доказать полную корректность програ-м-мы относительно следующего входного предиката $\varphi \equiv (x \geq 0)$ и выходного предиката $\psi \equiv (z^3 \leq x < (z + 1)^3)$. Доменом всех переменных является множество целых чисел.



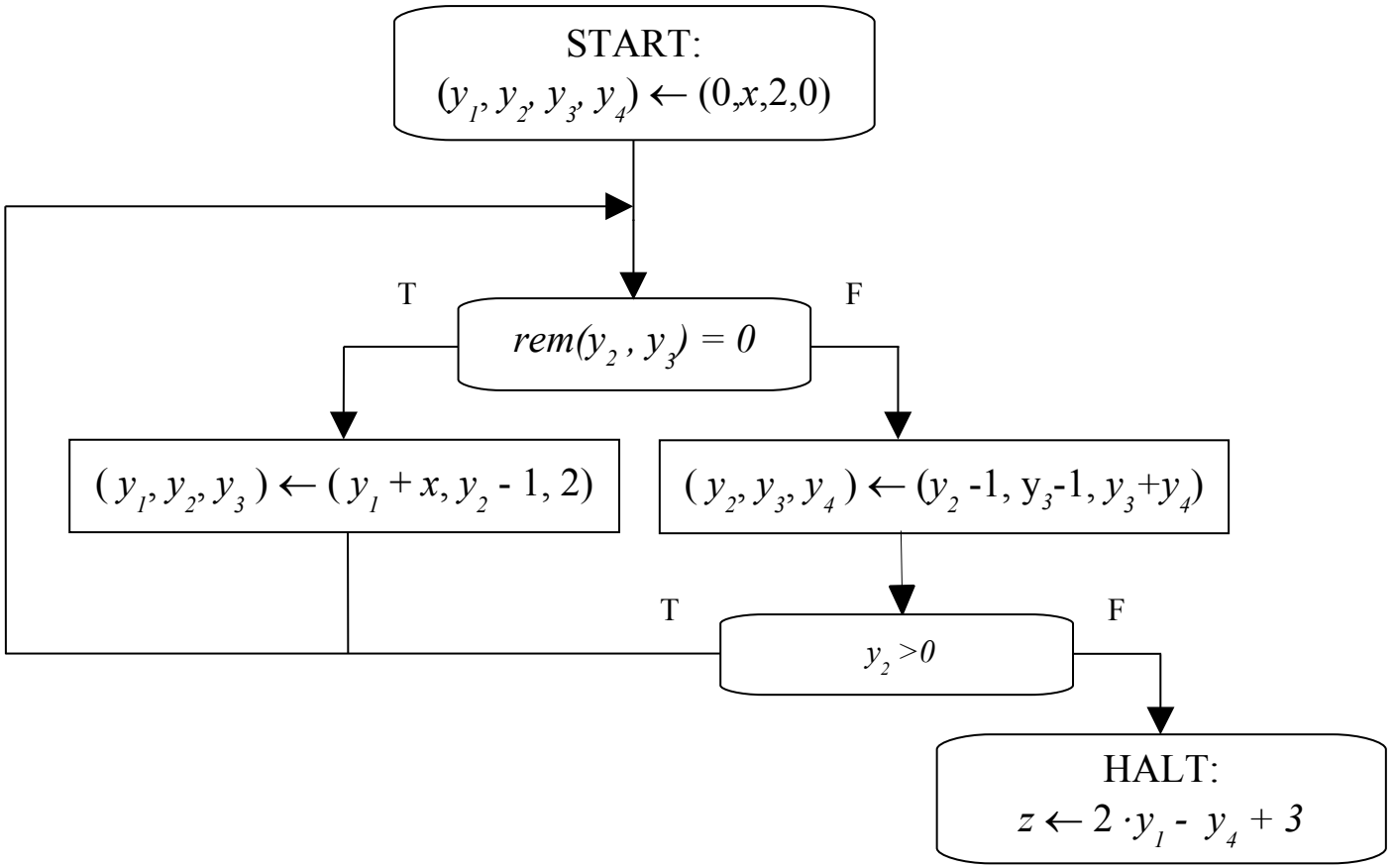
2.5.17 Доказать полную корректность программы относительно входного предиката $\varphi \equiv (x > 0) \wedge (x \vdots 2)$ и выходного предиката $\psi \equiv (z = x^2)$ при помощи методов Флойда. Доменом всех переменных является множество целых чисел.



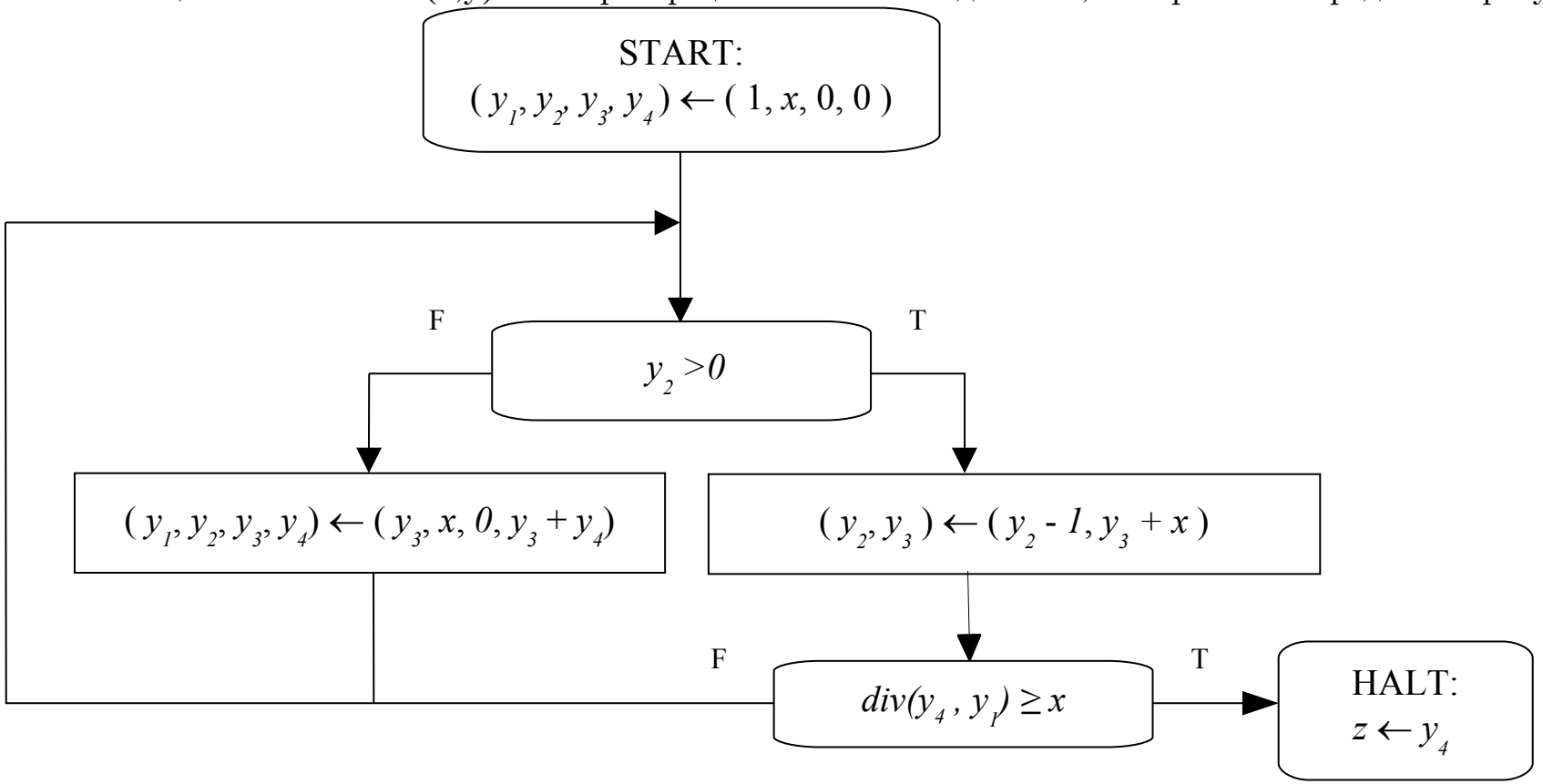
2.5.18 Доказать полную корректность программы относительно входного предиката $\varphi \equiv (x > 0) \wedge (x \vdots 2)$ и выходного предиката $\psi \equiv (z = x^2)$ при помощи методов Флойда. Доменом всех переменных является множество целых чисел.



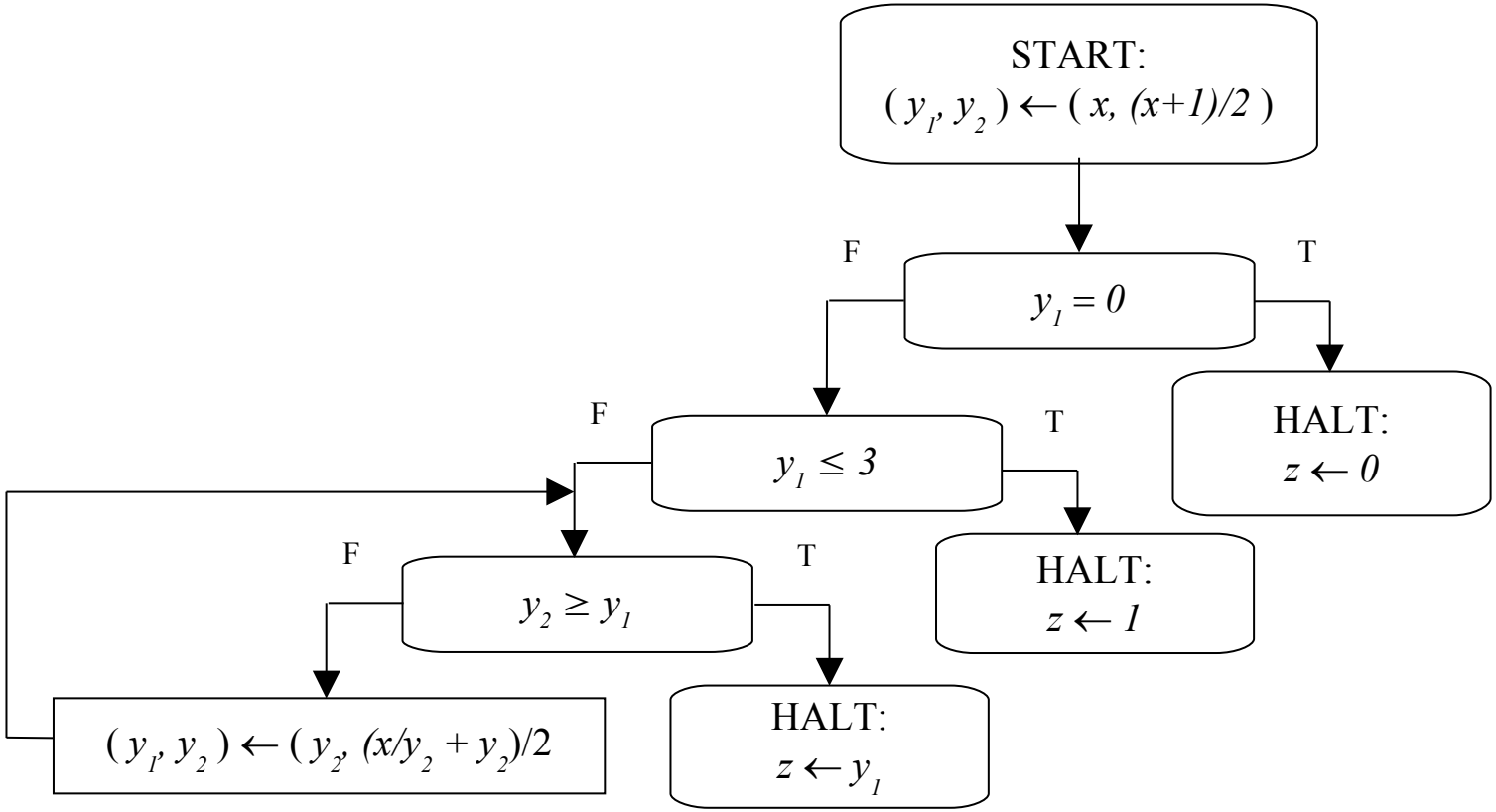
2.5.19 Доказать полную корректность программы относительно входного предиката $\varphi \equiv (x > 0) \wedge \neg(x \vdots 2)$ и выходного предиката $\psi \equiv (z = x^2)$ при помощи методов Флойда. Доменом всех переменных является множество целых чисел. rem(x,y) — оператор взятия остатка от целочисленного деления, который не определен при y=0.



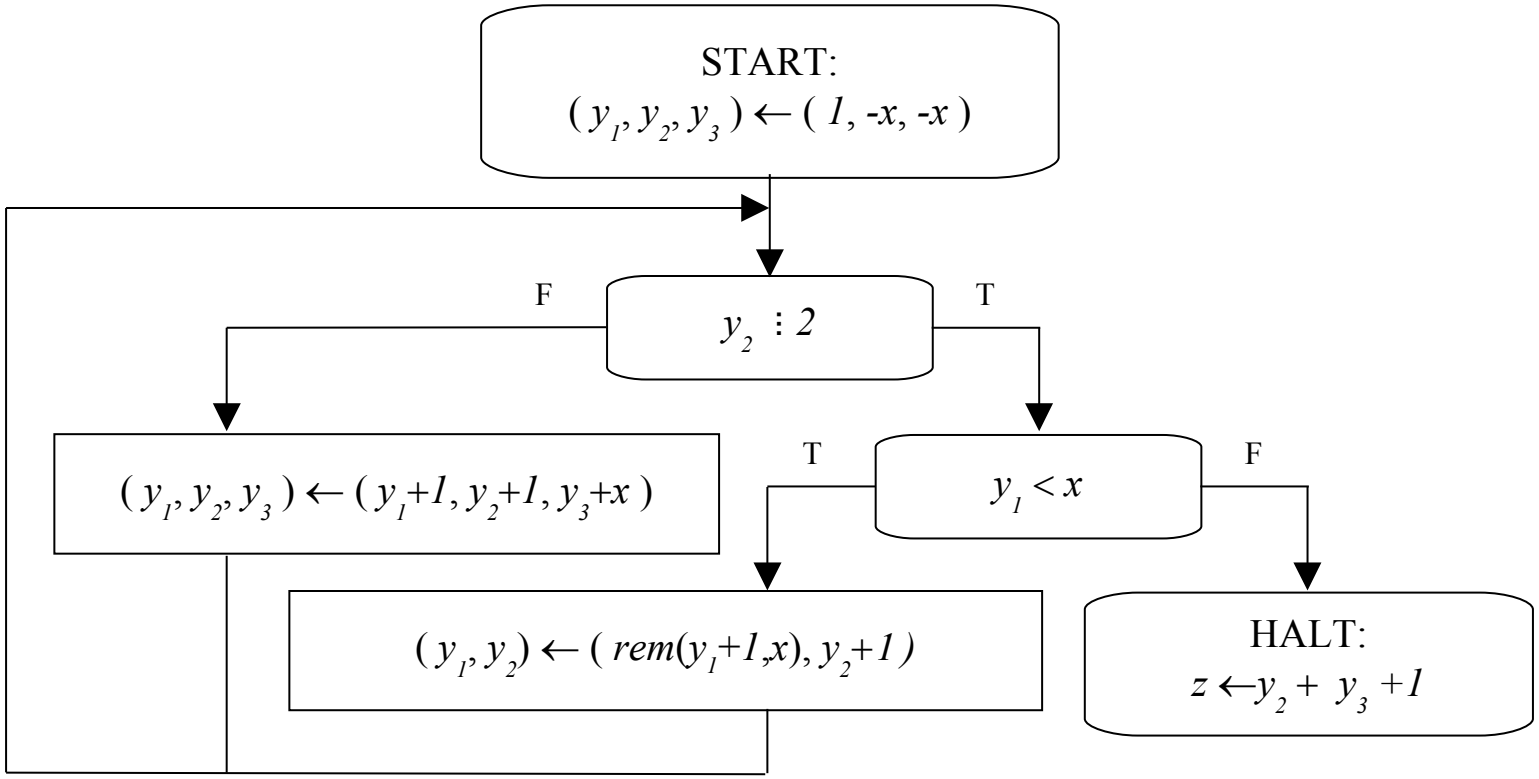
2.5.20 Доказать полную корректность программы относительно входного предиката $\varphi \equiv (x > 0)$ и выходного предиката $\psi \equiv (z = x^3)$ при помощи методов Флойда. Доменом всех переменных является множество целых чисел. div(x,y) — оператор целочисленного деления, который не определен при y=0.



2.5.21 Доказать полную корректность программы относительно входного предиката $\varphi \equiv (x \geq 0)$ и выходного предиката $\psi \equiv (z^2 \leq x < (z + 1)^2)$ при помощи методов Флойда. Доменом всех переменных является множество целых чисел.



2.5.22 Доказать полную корректность программы относительно входного предиката $\varphi \equiv (x > 0) \wedge \neg(x \vdots 2)$ и выходного предиката $\psi \equiv (z = x^2)$ при помощи методов Флойда. Доменом всех переменных является множество целых чисел. rem(x,y) — оператор целочисленного деления, который не определен при y=0.



Литература

1. R. W. Floyd, "Assigning meanings to programs", Proc. Symp. Appl. Math., 19; in: J.T.Schwartz (ed.), Mathematical Aspects of Computer Science, pp. 19-32, American Mathematical Society, Providence, R.I., 1967.
2. N. Francez, "Verification of programs", Addison-Wesley Publishers Ltd., 1992.
3. Z. Manna, "Mathematical theory of computation", McGraw-Hill, 1974.
4. Z. Manna, A. Pnueli, "The Temporal Logic of Reactive and Concurrent Systems", Springer-Verlag, 1991.
5. Р. Андерсон, "Доказательство правильности программ", Москва, Мир, 1982.
6. Д. Жуков, "Методы верификации программ", Переславль-Залесский, 2002.