Метрики сложности потока управления программ

Метрики сложности потока управления программ принято определять на основе представления программ в виде управляющего ориентированного графа *G=(V, E)*, где *V* — вершины, соответствующие операторам, а *E* — дуги, соответствующие переходам между операторами. В дуге *(v, u)* вершина *v* является исходной, а *u* — конечной. При этом *u* непосредственно следует за *v,* а *v* непосредственно предшествует *u.* Если путь от *v* до *u* состоит более чем из одной дуги, тогда *u* следует за *v,* а *v* предшествует *u* [2].

Частным случаем представления ориентированного графа программы можно считать детализированную схему алгоритма, в которой каждому блоку соответствует один оператор программы, построенную в соответствии с положениями стандарта ГОСТ 19.701-90 [1]. Аналогами вершин графа являются блоки алгоритма, причем данные блоки имеют разное графическое представление в зависимости от их назначения. Дугам графа соответствуют линии передачи управления между блоками алгоритма.

Ниже рассмотрены наиболее распространенные метрики сложности потока управления программ.

Метрика Маккейба (цикломатическая сложность графа программы, цикломатическое число Маккейба) предназначена для оценки трудоемкости тестирования программы. Данная метрика определяется по формуле:

$$Z(G) = e - \upsilon + 2p,$$

где e — число дуг ориентированного графа G; υ — число вершин; p — число компонентов связности графа.

Число компонентов связности графа – это количество дуг, которые необходимо добавить для преобразования графа в сильносвязный. Сильносвязным графом называется граф, любые две вершины которого взаимно достижимы. Для корректных программ, не имеющих недостижимых от начала программы участков и «висячих» точек входа и выхода, сильносвязный граф получается путем соединения дугой вершины, обозначающей конец программы, с вершиной, обозначающей начало этой программы.

Метрика Маккейба определяет минимальное количество тестовых прогонов программы, необходимых для тестирования всех ее ветвей (разветвлений).

Рассчитаем метрику Маккейба для программы, схема алгоритма которой приведена на рис. 1. Действия, выполняемые блоками программы, в примере не показаны. Внутри каждого блока помещены их номера. Компонент связности графа обозначен штриховой дугой. Число дуг $e = 8$, число вершин $\upsilon = 7$, $p = 1$. Цикломатическое число Маккейба равно $Z(G) = 8 - 7 + 2 = 3$.

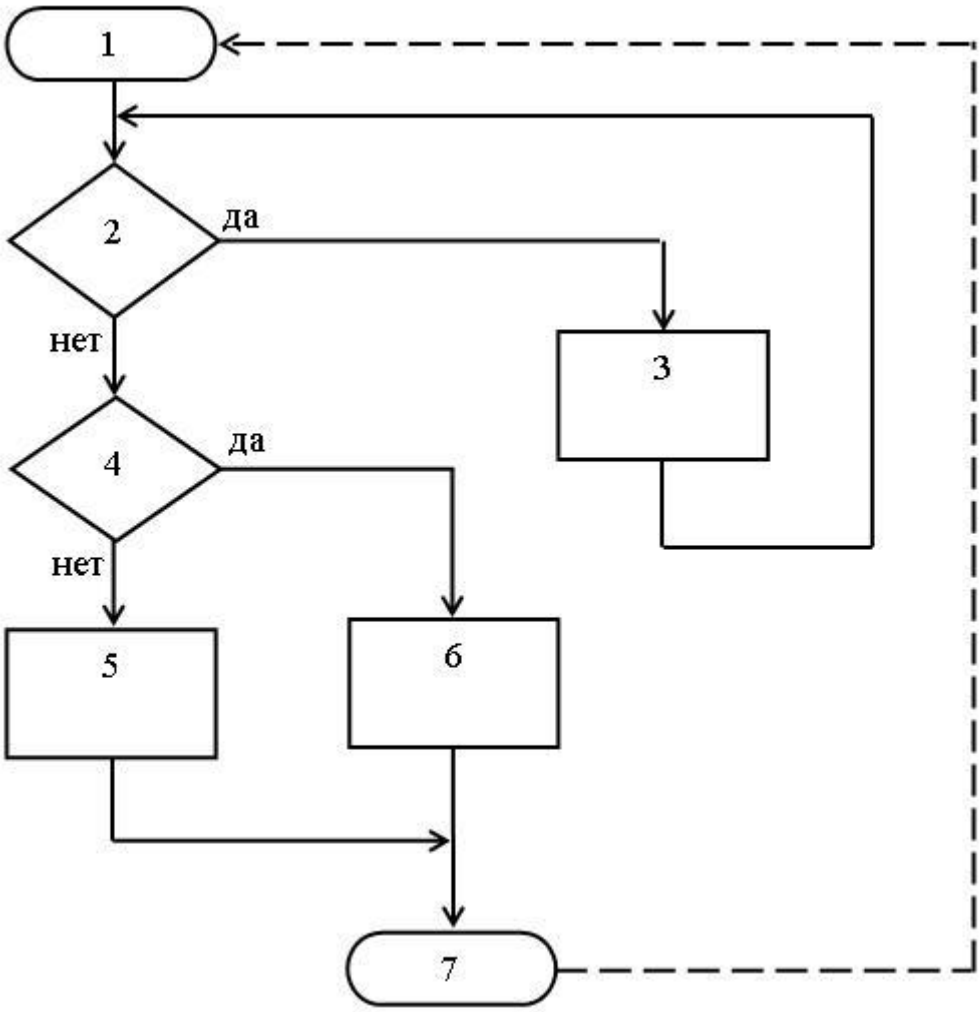

Рисунок 1 – Пример схемы алгоритма программы

Значение метрики Маккейба показывает, что в схеме алгоритма (см. рис.1) можно выделить три базисных независимых пути (называемых также линейно независимыми контурами):

1-й путь. 1 – 2 (нет) – 4 (нет) – 5 – 7.

2-й путь. 1 – 2 (да) – 3 – 2 (нет) – 4 (нет) – 5 – 7.

3-й путь. 1 – 2 (нет) – 4 (да) – 6 – 7.

Вторым возможным вариантом совокупности базисных независимых путей является:

1-й путь. 1 – 2 (да) – 3 – 2 (нет) – 4 (нет) – 5 – 7.

2-й путь. 1 – 2 (нет) – 4 (нет) – 5 – 7.

3-й путь. 1 – 2 (да) – 3 – 2 (нет) – 4 (да) – 6 – 7.

Таким образом, для тестирования совокупности базисных независимых путей исследуемой программы необходимо выполнить минимально три тестовых прогона.

Метрика Джилба определяет логическую сложность программы как насыщенность программы условными операторами IF–THEN–ELSE. Обычно используются два вида метрики Джилба: *CL* — количество условных операторов, характеризующее абсолютную сложность программы; *cl* — насыщенность программы условными операторами, характеризующая относительную сложность программы; *cl* определяется как отношение *CL* к общему количеству операторов программы (здесь под оператором подразумевается оператор конкретного языка программирования в классическом представлении, а не в интерпретации Холстеда).

Расширением метрики Джилба является *максимальный уровень вложенности условного оператора CLI.*

Использование в программе оператора выбора (например, CASE) с n разветвлениями эквивалентно применению $n - 1$ оператора IF–THEN–ELSE с глубиной вложенности $n - 2$.

Например, на рис. 2 приведена схема алгоритма вычисления некоторой функции Y. В данной схеме используется выбор, обозначаемый символом «Решение» с пятью разветвлениями ($n = 5$). Эквивалентный алгоритм вычисления той же функции Y. использующий несколько операторов IF–THEN–ELSE, представлен на рис. 3. На данном рисунке количество операторов IF–THEN–ELSE равно четырем ($n - 1$), максимальный уровень вложенности оператора IF–THEN–ELSE равен трем ($n - 2$).

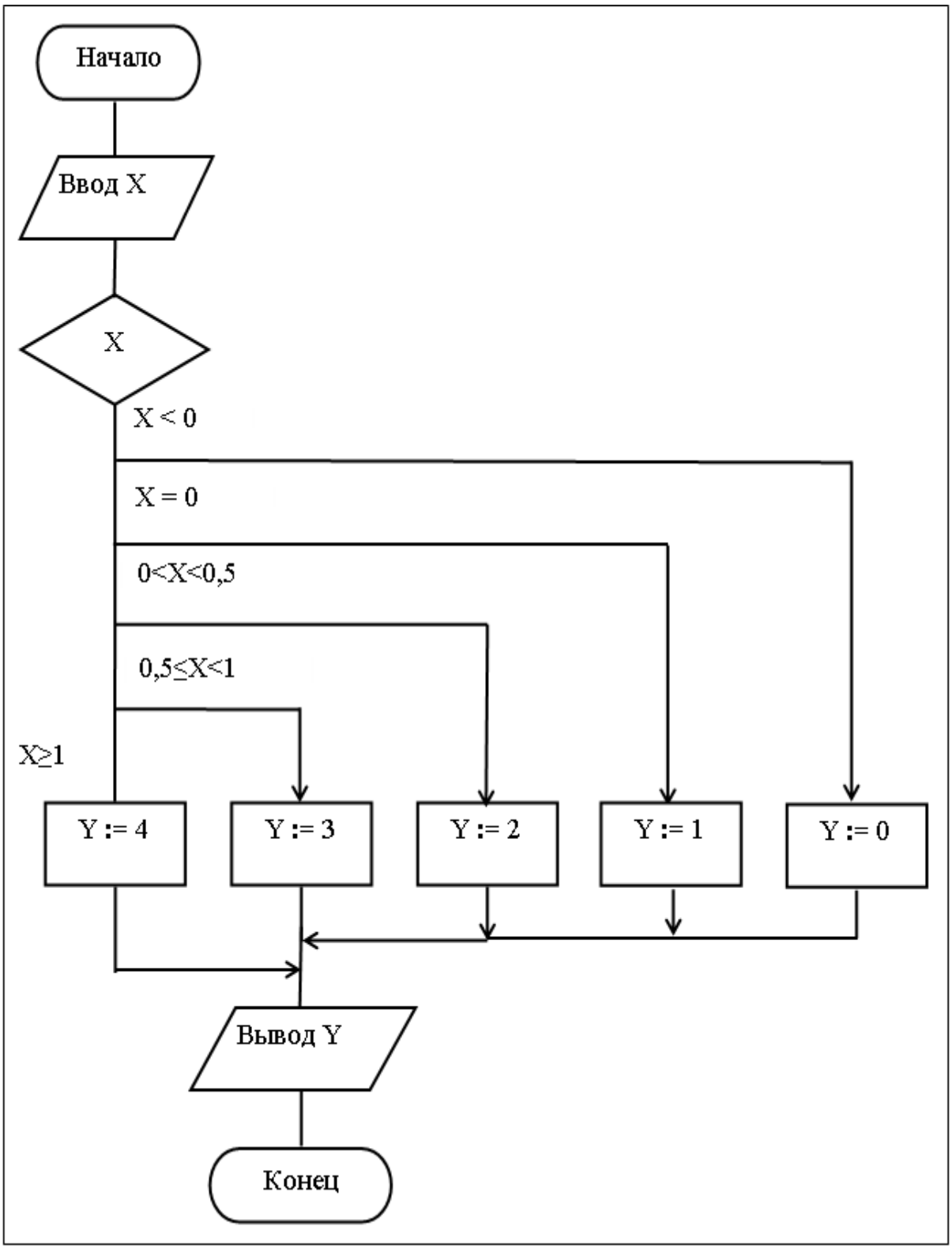

Рисунок 2 – Схема разветвляющегося алгоритма вычисления функции **Y** (используется символ «Решение» с многими выходами)

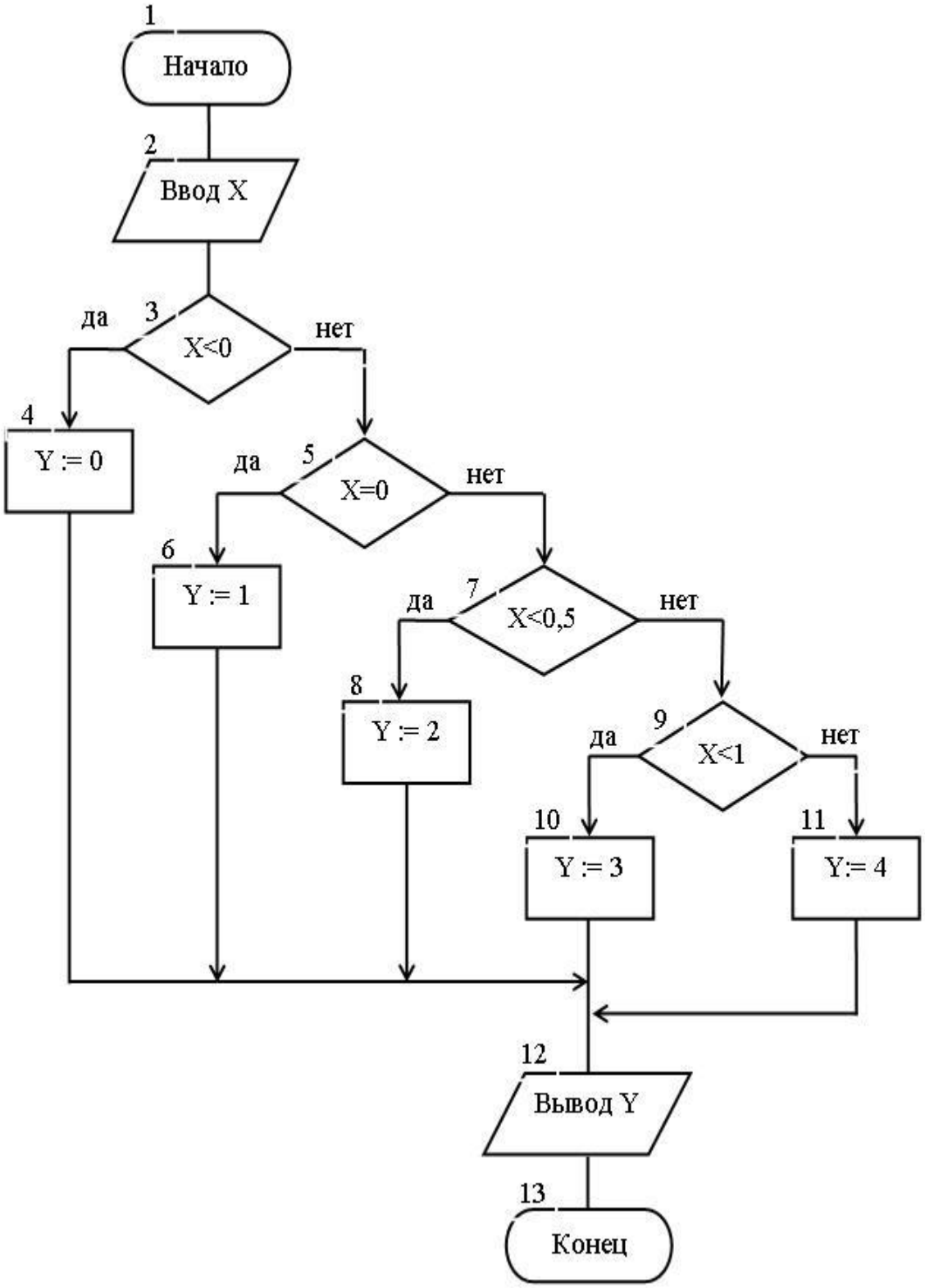

Рисунок 3 – Схема разветвляющегося алгоритма вычисления функции **Y** (используется символ «Решение» с двумя выходами)

Таким образом, для схем алгоритмов, приведенных на рис. 2 и 3, *CL* = 4, *cl* = 0,36 (количество операторов программы равно 11), *CLI* = 3.

Значения метрики Маккейба для данных алгоритмов также совпадают. Для схемы алгоритма, представленной на рис. 2,

$Z(G) = 13 – 10 +2 = 5.$

Для схемы алгоритма, приведенной на рис. 3,

$Z(G) = 16 – 13 +2 = 5.$

Следует отметить, что сложность программы с помощью метрики Джилба не всегда возможно посчитать на основе схемы алгоритма, так как схема алгоритма может быть представлена укрупнено. Поэтому значения метрики Джилба в программе в общем случае следует определять на основе ее исходного текста или детализированной схемы алгоритма, каждый блок которой содержит один оператор программы.

Метрика граничных значений базируется на определении скорректированной сложности вершин графа программы [2].

Пусть *G=(V, E)*—ориентированный граф программы с единственной начальной и единственной конечной вершинами. В этом графе число входящих в вершину дуг называется *отрицательной степенью вершины*, а число исходящих из вершины дуг — *положительной степенью вершины*. С учетом этого набор вершин графа можно разбить на две группы: вершины, у которых положительная степень меньше или равна 1; вершины, у которых положительная степень больше или равна 2. Вершины первой группы называются *принимающими вершинами*, вершины второй группы – *вершинами выбора* (или предикатными вершинами, условными вершинами, вершинами отбора).

Для оценки сложности программы с использованием метрики граничных значений граф *G* разбивается на максимальное число подграфов**,** удовлетворяющих следующим условиям: вход в подграф осуществляется через вершину выбора; каждый подграф включает вершину (нижнюю границу подграфа), в которую можно попасть из любой другой вершины подграфа.

Число вершин, образующих подграф (исключая вершину выбора, через которую осуществляется вход в подграф), равно скорректированной сложности вершины выбора. Каждая принимающая вершина имеет скорректированную сложность, равную 1. Конечная вершина имеет скорректированную сложность, равную 0.

Абсолютная граничная сложность программы S_a определяется как сумма скорректированных сложностей всех вершин графа *G*.

Относительная граничная сложность программы S_o определяется по формуле:

$$S_o = 1 - \frac{\upsilon - 1}{S_a}$$

где S_o – относительная граничная сложность программы; S_a — абсолютная граничная сложность программы; v – общее число вершин графа программы.

В таблице 3 представлены свойства подграфов программы, схема алгоритма которой приведена на рис. 3. Табл. 4 содержит скорректированные сложности вершин графа данной программы. Номера вершин графа соответствуют номерам соответствующих блоков на схеме алгоритма.

Таблица 3
Свойства подграфов программы (рис. 3)

Свойства подграфов программы	Номер вершины выбора			
	3	5	7	9
Номера вершин перехода	4, 5	6, 7	8, 9	10, 11
Скорректированная сложность вершины выбора	9	7	5	3
Номера вершин подграфа	4, 5, 6, 7, 8, 9, 10, 11	6, 7, 8, 9, 10, 11	8, 9, 10, 11	10, 11
Номер нижней границы подграфа	12	12	12	12

Таблица 4
Скорректированные сложности вершин графа программы (рис. 3)

Номер вершины графа программы	1	2	3	4	5	6	7	8	9	10	11	12	13	
Скорректированная сложность вершины графа	1	1	9	1	7	1	5	1	3	1	1	1	0	$\boldsymbol{S_a = 32}$

Таким образом, абсолютная граничная сложность S_a программы, схема алгоритма которой приведена на рис. 3, равна 32. Относительная граничная сложность данной программы равна

$S_o = 1 – (13 – 1)/32 = 0{,}625.$

В табл. 5 представлены свойства подграфов программы, схема алгоритма которой приведена на рис. 1. Табл. 6 содержит скорректированные сложности вершин графа данной программы.

Таблица 5 – **Свойства подграфов программы (рис.1)**

Свойства подграфов программы	Номер вершины выбора	
	2	4
Номера вершин перехода	3, 4	5, 6
Скорректированная сложность вершины выбора	3	3
Номера вершин подграфа	2, 3	5, 6
Номер нижней границы подграфа	4	7

Таблица 6 – Скорректированные сложности вершин графа программы (рис.1)

Номер вершины графа программы	1	2	3	4	5	6	7	
Скорректированная сложность вершины графа	1	3	1	3	1	1	0	$\boldsymbol{S_a = 10}$

Абсолютная граничная сложность S_a программы, схема алгоритма которой приведена на рис. 1, равна 10. Относительная граничная сложность данной программы равна

$S_o = 1 – (7 – 1)/10 = 0{,}4.$

Табл. 7 содержит метрики сложности потока управления для программ, схемы алгоритмов которых приведены на рис. 1 – 3.

Таблица 7 – Метрики сложности потока управления программ (рис.1-3)

Метрики сложности потока управления	Схемы алгоритмов		
	Рис. 1	Рис. 2	Рис. 3
Метрика Маккейба *Z(G)*	3	5	5
Абсолютная сложность программы *CL* по метрике Джилба	2	4	4
Относительная сложность программы *cl* по метрике Джилба	0,29 *	0,36	0,36
Максимальный уровень вложенности условного оператора *CLI* по метрике Джилба	1	3	3
Метрика граничных значений (абсолютная граничная сложность программы) S_a	10	14	32
Метрика граничных значений (относительная граничная сложность программы) S_0	0,4	0,357	0,625

Примечание.

* - относительная сложность *cl* по метрике Джилба для программы, схема алгоритма которой представлена на рис. 1, рассчитана в предположении, что каждый блок схемы алгоритма содержит один оператор программы.