```
    while (abs(vs) >= eps); // Выход из цикла по выполнению условия
    printf("%f %f %f\n", x, y, eps);
}
```

Расчет базовых метрик Холстеда для данной программы приведен в табл. 2.4. По формулам (2.1) – (2.3) рассчитываются словарь, длина и объем программы.

Словарь программы: $\eta = 16 + 6 = 22$.

Длина программы: $N = 38 + 24 = 62$.

Объем программы: $V = 62 \log_2 22 \approx 276$.

Таблица 2.4

Расчет базовых метрик Холстеда для программы, вычисляющей значение функции *Y* = sin *X* на языке *C*

j	Оператор	f_{1j}	i	Операнд	f_{2i}
1.	;	9	1.	x	6
2.	=	4	2.	vs	5
3.	*	4	3.	n	4
4.	,	4	4.	2	4
5.	–	3	5.	y	3
6.	/	2	6.	eps	2
7.	( )	2			
8.	{…}	2			
9.	scanf ( )	1			
10.	do…while ( )	1			
11.	abs( )	1			
12.	>=	1			
13.	printf ( )	1			
14.	++	1			
15.	+=	1			
16.	&	1			
$\eta_1 = 16$		$N_1 = 38$	$\eta_2 = 6$		$N_2 = 24$

2.2. 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 [27, 29]. 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.

Библиотека БГУИР

Частным случаем представления ориентированного графа программы можно считать построенную в соответствии с положениями стандарта *ГОСТ 19.701–90* [1] детализированную схему алгоритма, в которой каждому блоку соответствует один оператор программы. 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 – 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.

Рассчитаем метрику Маккейба для программы, схема алгоритма которой приведена на рис. 2.1 (для простоты понимания ввод/вывод в данной схеме алгоритма не показан, хотя в реальных вычислениях метрики блоки ввода/вывода следует учитывать). Компонент связности графа обозначен штриховой дугой. Число дуг $e = 8$, число вершин $\upsilon = 7$, $p = 1$. Цикломатическое число Маккейба равно $Z(G) = 8 - 7 + 2 = 3$.

Следует обратить внимание, что при расчете метрики Маккейба начальную и конечную вершины алгоритма (блоки «Начало» и «Конец») необходимо учитывать.

Библиотека БГУИР

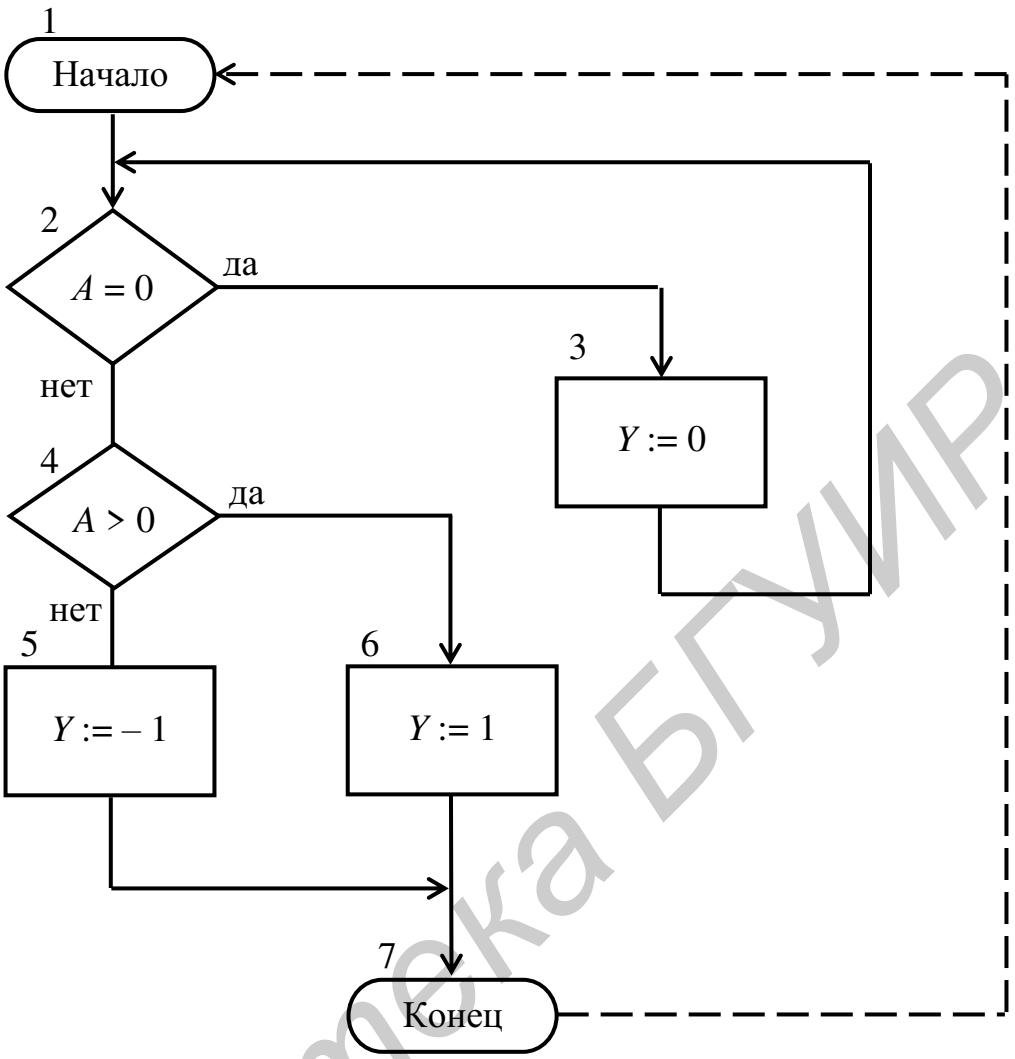


Рис. 2.1. Пример схемы алгоритма программы

Значение метрики Маккейба показывает, что в схеме алгоритма (см. рис. 2.1) можно выделить *три базовых независимых пути* (называемых также линейно независимыми контурами):

1) 1 – 2 (нет) – 4 (нет) – 5 – 7;
2) 1 – 2 (да) – 3 – 2 (нет) – 4 (нет) – 5 – 7;
3) 1 – 2 (нет) – 4 (да) – 6 – 7.

Вторым возможным вариантом совокупности базовых независимых путей является:

1) 1 – 2 (да) – 3 – 2 (нет) – 4 (нет) – 5 – 7;
2) 1 – 2 (нет) – 4 (нет) – 5 – 7;
3) 1 – 2 (да) – 3 – 2 (нет) – 4 (да) – 6 – 7.

Цифры в обозначении путей представляют собой номера блоков в схеме алгоритма программы.

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

Библиотека БГУИР

Метрика Джилба определяет логическую сложность программы как насыщенность программы условными операторами IF–THEN–ELSE и операторами цикла. Обычно используются два вида метрики Джилба: ***CL*** – количество условных и циклических операторов, характеризующее абсолютную сложность программы; ***cl*** – насыщенность программы условными и циклическими 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 и циклического оператора* ***CLI***.

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

Например, на рис. 2.2 приведена схема алгоритма вычисления некоторой функции ***Y***. В данной схеме используется выбор, обозначаемый символом «Решение» (ромб) с пятью разветвлениями (***n*** = 5).

Эквивалентный алгоритм вычисления той же функции ***Y***, использующий несколько операторов IF–THEN–ELSE, представлен на рис. 2.3.

На данном рисунке количество операторов IF–THEN–ELSE равно четырем (***n*** – 1), максимальный уровень вложенности оператора IF–THEN–ELSE равен трем (***n*** – 2).

Таким образом, для схем алгоритмов, приведенных на рис. 2.2 и 2.3, ***CL*** = 4, ***cl*** = 0,36 (количество операторов программы равно 11; блоки «Начало» и «Конец» в метрике Джилба не учитываются), ***CLI*** = 3.

Значения метрики Маккейба для данных алгоритмов также совпадают.

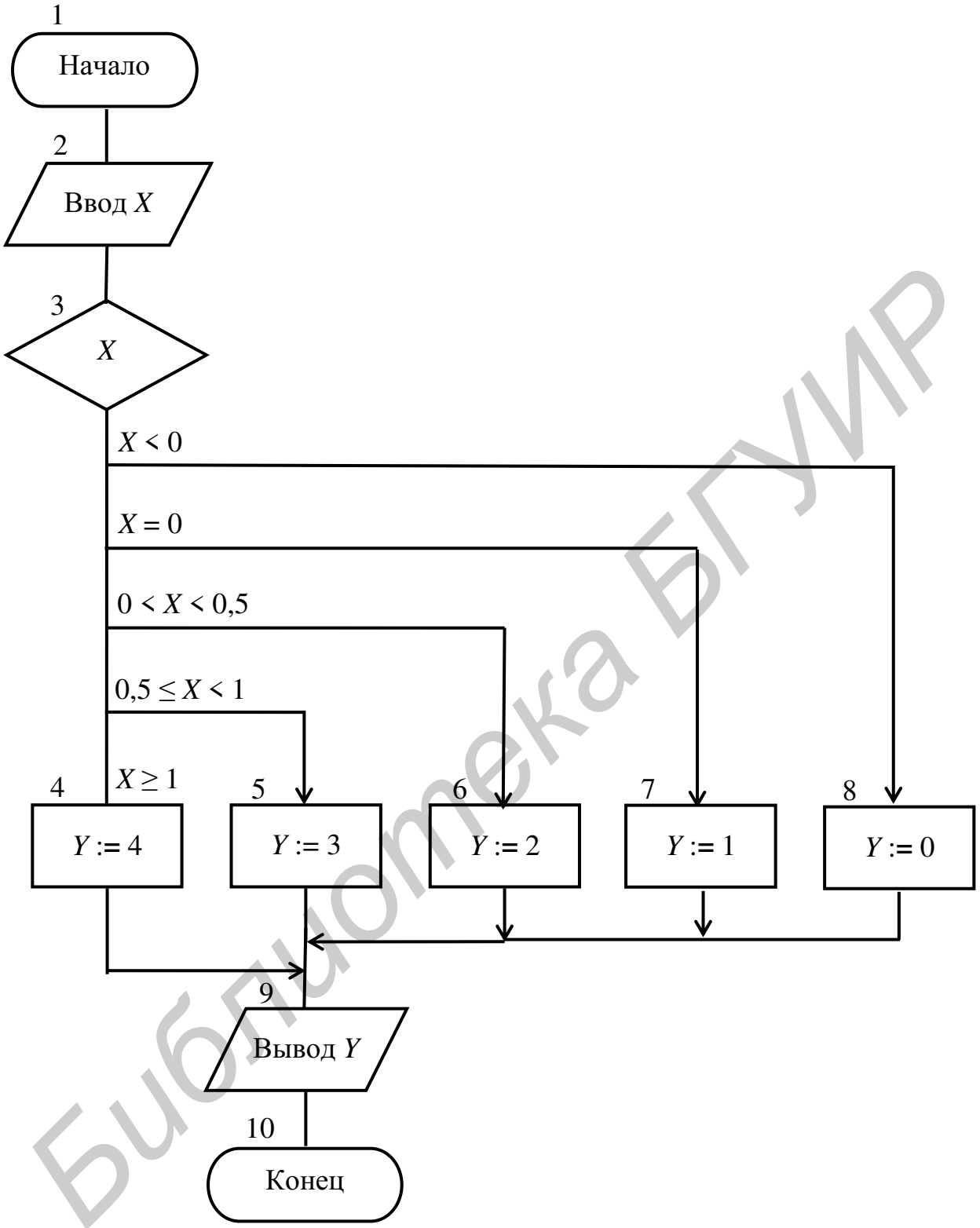
Для схемы алгоритма, представленной на рис. 2.2, $Z(G) = 13 - 10 + 2 = 5$.

Для схемы алгоритма, приведенной на рис. 2.3, $Z(G) = 16 - 13 + 2 = 5$.

Для схемы алгоритма, приведенной на рис. 2.1, значение метрики Джилба ***CL*** = 2, ***cl*** = 0,4 (количество операторов программы равно 5), ***CLI*** = 0.

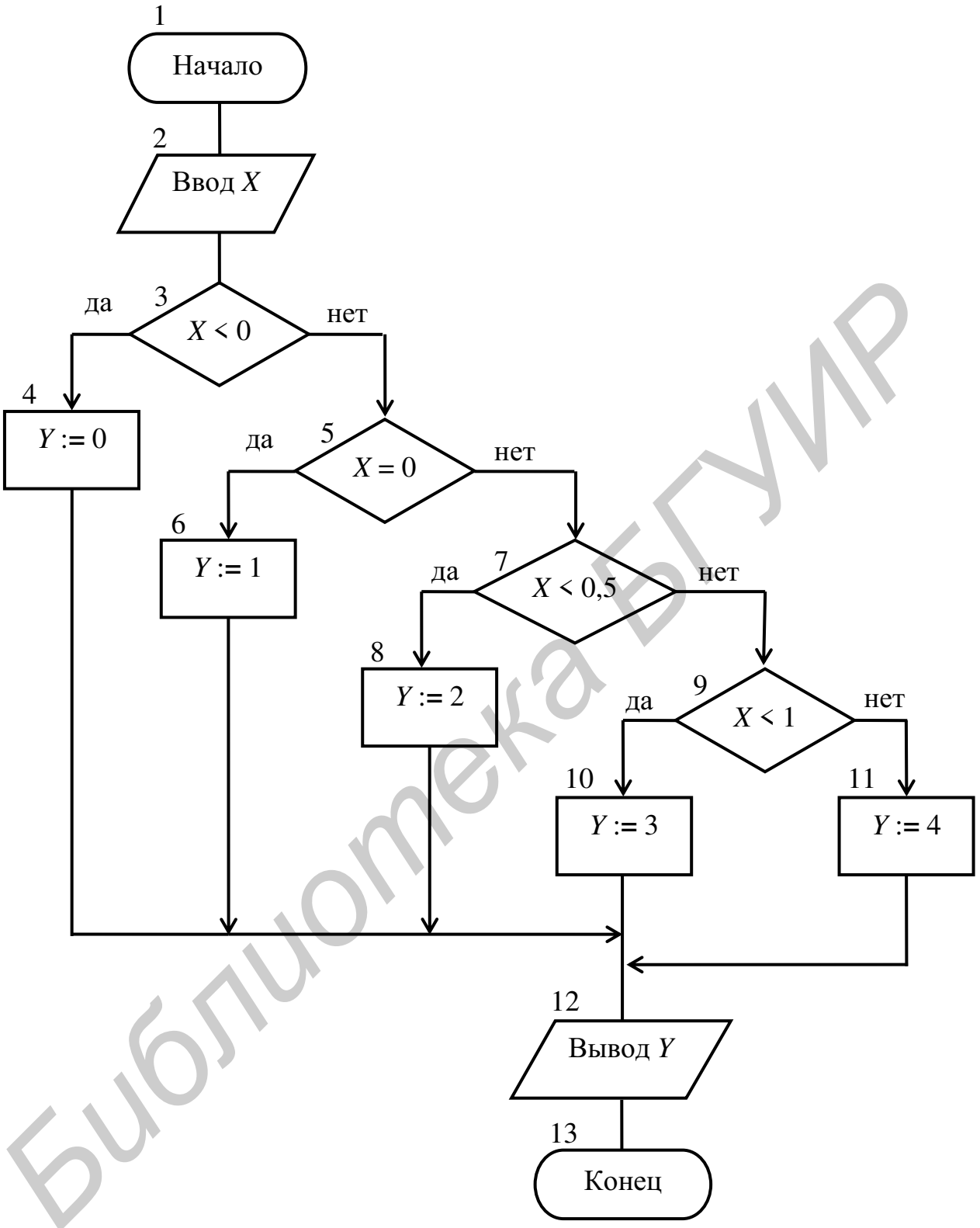
Следует отметить, что сложность программы с помощью метрики Джилба не всегда возможно посчитать на основе схемы алгоритма, т. к. схема алгоритма может быть представлена укрупнено. Поэтому значения метрики Джилба в программе в общем случае следует определять на основе ее исходного текста или детализированной схемы алгоритма, каждый блок которой содержит один оператор программы (как это реализовано в алгоритмах, представленных на рис. 2.1 – 2.3).

Библиотека БГУИР



Библиотека БГУИР

Рис. 2.2. Схема разветвляющегося алгоритма вычисления функции ***Y*** (используется символ «Решение» со многими выходами)



Библиотека БГУИР

Рис. 2.3. Схема разветвляющегося алгоритма вычисления функции ***Y*** (используется символ «Решение» с двумя выходами)

Метрика граничных значений базируется на определении скорректированной сложности вершин графа программы [27].

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.

Каждая вершина выбора имеет скорректированную сложность, равную числу вершин, образующих связанный с ней подграф. Каждая принимающая вершина имеет скорректированную сложность, равную 1. Конечная вершина графа имеет скорректированную сложность, равную 0.

Следует обратить внимание, что если подграф представляет собой обычное разветвление, то вершина выбора, через которую осуществляется вход в такой подграф, не является элементом подграфа. Поэтому при расчете скорректированной сложности такой вершины выбора сама эта вершина не учитывается.

Если подграф представляет собой цикл, то по одной из исходящих дуг вершины выбора осуществляется вход в тело цикла, а по другой – выход из цикла. В этом случае нижней границей подграфа является вершина перехода, находящаяся после вершины выбора на дуге выхода из цикла. После входа в подграф через вершину выбора и выполнения тела цикла для достижения нижней границы подграфа следует снова возвращаться к данной вершине выбора. Таким образом, вершина выбора, через которую осуществляется вход в циклический подграф, является элементом данного подграфа. Поэтому данную вершину выбора следует учитывать при подсчете ее скорректированной сложности.

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 υ – общее число вершин графа программы.

В табл. 2.5 представлены свойства подграфов программы, схема алгоритма которой приведена на рис. 2.3. Скорректированные сложности вершин графа данной программы содержит табл. 2.6. Номера вершин графа соответствуют номерам соответствующих блоков на схеме алгоритма.

Библиотека БГУИР

В рассматриваемой схеме алгоритма все подграфы представляют собой разветвления. Поэтому вершины выбора в данные подграфы не входят и при расчете своей скорректированной сложности не учитываются (см. табл. 2.5).

Таблица 2.5

Свойства подграфов программы*

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

* Схема алгоритма программы приведена на рис. 2.3.

Таблица 2.6

Скорректированные сложности вершин графа программы**

Номер вершины графа программы	1	2	3	4	5	6	7	8	9	10	11	12	13	S_a
Скорректированная сложность вершины графа	1	1	9	1	7	1	5	1	3	1	1	1	0	**32**

** Схема алгоритма программы представлена на рис. 2.3.

Таким образом, абсолютная граничная сложность S_a программы, схема алгоритма которой приведена на рис. 2.3, равна 32. Относительная граничная сложность данной программы равна

$$S_o = 1 - (13 - 1)/32 = 0{,}625.$$

В табл. 2.7 представлены свойства подграфов программы, схема алгоритма которой приведена на рис. 2.1. Скорректированные сложности вершин графа данной программы содержит табл. 2.8.

В рассматриваемой схеме алгоритма подграф с вершиной выбора 2 содержит цикл. Поэтому данная вершина выбора является элементом подграфа и участвует при подсчете своей скорректированной сложности (см. табл. 2.7).

Абсолютная граничная сложность S_a программы, схема алгоритма которой приведена на рис. 2.1, равна 10. Относительная граничная сложность данной программы равна

$$S_o = 1 - (7 - 1)/10 = 0{,}4.$$

Библиотека БГУИР

Таблица 2.7

Свойства подграфов программы*

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

* Схема алгоритма программы представлена на рис. 2.1.

Таблица 2.8

Скорректированные сложности вершин графа программы**

Номер вершины графа программы	1	2	3	4	5	6	7	S_a
Скорректированная сложность вершины графа	1	3	1	3	1	1	0	**10**

** Схема алгоритма программы представлена на рис. 2.1.

Метрики сложности потока управления для программ, схемы алгоритмов которых приведены на рис. 2.1 – 2.3, содержит табл. 2.9.

Таблица 2.9

Метрики сложности потока управления программ

Метрики сложности потока управления	Схемы алгоритмов		
	Рис. 2.1	Рис. 2.2	Рис. 2.3
Метрика Маккейба ***Z*(*G*)**	3	5	5
Абсолютная сложность программы ***CL*** по метрике Джилба	2	4	4
Относительная сложность программы ***cl*** по метрике Джилба	0,4	0,36	0,36
Максимальный уровень вложенности условного оператора ***CLI*** по метрике Джилба	0	3	3
Метрика граничных значений S_a (абсолютная граничная сложность программы)	10	14	32
Метрика граничных значений S_o (относительная граничная сложность программы)	0,4	0,357	0,625

Библиотека БГУИР

2.3. Метрики сложности потока данных

Метрики сложности потока данных связывают сложность программы с использованием и размещением данных в этой программе. Метрики данной группы основаны на анализе исходных текстов программ.

К наиболее известным в рассматриваемой группе метрик можно отнести спен и метрику Чепина [27, 29].

Спен идентификатора – число повторных появлений идентификатора (число появлений после его первого появления) в тексте программы. Идентификатор, встречающийся в тексте программы n раз, имеет спен, равный $n - 1$.

Величина спена связана со сложностью тестирования и отладки программы. Например, если спен идентификатора равен 10, то при трассировании программы по этому идентификатору следует ввести в текст программы 10 контрольных точек, что усложняет тестирование и отладку программы.

Метрика Чепина базируется на анализе характера использования переменных в программе.

Существуют различные варианты метрики Чепина. Ниже рассмотрен вариант (назовем данный вариант полной метрикой Чепина), в котором все множество переменных программы разбивается на четыре функциональные группы:

1. P – вводимые переменные, содержащие исходную информацию, но не модифицируемые в программе и не являющиеся управляющими переменными;
2. M – вводимые модифицируемые переменные и создаваемые внутри программы константы и переменные, не являющиеся управляющими переменными;
3. C – переменные, участвующие в управлении работой программы (управляющие переменные);
4. T – не используемые в программе («паразитные») переменные, например, вычисленные переменные, значения которых не выводятся и не участвуют в дальнейших вычислениях.

Значение метрики Чепина определяется по формуле

$$Q = a_1 p + a_2 m + a_3 c + a_4 t ,$$

где a_1, a_2, a_3, a_4 – весовые коэффициенты; p, m, c, t – количество переменных в группах P, M, C, T соответственно.

Весовые коэффициенты позволяют учитывать различное влияние на сложность программы каждой функциональной группы. Наиболее часто применяются следующие значения весовых коэффициентов: $a_1 = 1$, $a_2 = 2$, $a_3 = 3$, $a_4 = 0{,}5$. С учетом данных значений формула для определения метрики Чепина принимает вид

$$Q = p + 2m + 3c + 0{,}5t .$$

Помимо полной метрики Чепина распространен ее вариант, при котором анализу и разбиению на четыре группы подвергаются только переменные из списка ввода/вывода программы, т. е. те переменные, которые содержатся в

Библиотека БГУИР

списке параметров операторов ввода/вывода программы. Назовем данный вариант метрикой Чепина ввода/вывода.

Метрики сложности потока данных для программы на языке Delphi, вычисляющей значение функции ***Y* = sin *X***, содержат табл. 2.10, 2.11. Исходный текст программы приведен в примере 1.

В тексте программы (см. пример 1) идентификаторы впервые встречаются в разделе объявлений. Поэтому значение спена ***i***-го идентификатора равно количеству его появлений в разделе операторов, т. е. значению f_{2i} в метриках Холстеда.

В список переменных ввода/вывода данной программы входят переменные ***x***, ***y*** и константа ***eps***, являющиеся параметрами операторов ввода и вывода *Readln* и *Writeln*. Остальные переменные (***n***, ***vs***) в расчете метрики Чепина ввода/вывода не участвуют.

Таблица 2.10

Спен программы

Идентификатор	***x***	***n***	***vs***	***y***	***eps***	Суммарный спен программы
Спен	6	5	5	4	2	22

Таблица 2.11

Метрики Чепина программы

Переменные	Полная метрика Чепина				Метрика Чепина ввода/вывода			
Группа переменных	***P***	***M***	***C***	***T***	***P***	***M***	***C***	***T***
Переменные, относящиеся к группе	***x***	***y***, ***n***	***vs***, ***eps***	–	***x***	***y***	***eps***	–
Количество переменных в группе	***p*** = 1	***m*** = 2	***c*** = 2	***t*** = 0	***p*** = 1	***m*** = 1	***c*** = 1	***t*** = 0
Метрика Чепина	***Q*** = 1*1 + 2*2 + 3*2 + + 0,5*0 = 11				***Q*** = 1*1 + 2*1 + 3*1 + + 0,5*0 = 6			

2.4. Вопросы и задания для самоконтроля

1. На какие группы принято подразделять метрики сложности программ?
2. Что является основой для подсчета метрик размера программ?
3. Перечислите метрики, относящиеся к метрикам размера программ.
4. Назовите базовые метрики программы, являющиеся основой метрик Холстеда.
5. Что подразумевается под операторами в метриках Холстеда?
6. Что такое словарь операторов, словарь операндов, словарь программы в метриках Холстеда?
7. Что такое длина и объем программы в метриках Холстеда?
8. Перечислите метрики, относящиеся к метрикам сложности потока управления программ.

Библиотека БГУИР