Общие сведения о метриках сложности программ

Качество программных средств во многом зависит от сложности их кодов. Например, чем сложнее программа, тем ниже ее надежность и сопровождаемость. Поэтому при оценке качества программ обычно оценивается и их сложность.

Метрики сложности программ принято подразделять на ***три основные группы*** [2]:

* метрики размера программ;
* метрики сложности потока управления программ;
* метрики сложности потока данных программ.

**Метрики размера программ**

Метрики этой группы основаны на анализе исходных текстов программ.

Существуют различные метрики, с помощью которых может быть оценен размер программы.

К наиболее простым метрикам размера программы относятся ***количество строк исходного текста программы***и ***количество операторов программы***.

Из метрик размера программ широкое распространение получили ***метрики Холстеда*** [3].

Основу метрик Холстеда составляют *шесть базовых метрик* программы:

* η1 *–* словарь операторов (число уникальных операторов программы);
* η2 *–* словарь операндов (число уникальных операндов программы);
* *N1 –* общее число операторов в программе;
* *N2 –* общее число операндов в программе;
* *f1j –* число вхождений *j*-го оператора, *j* = 1, 2, …, η1;
* *f2i –* число вхождений *i*-го операнда, *i* = 1, 2, …, η2.

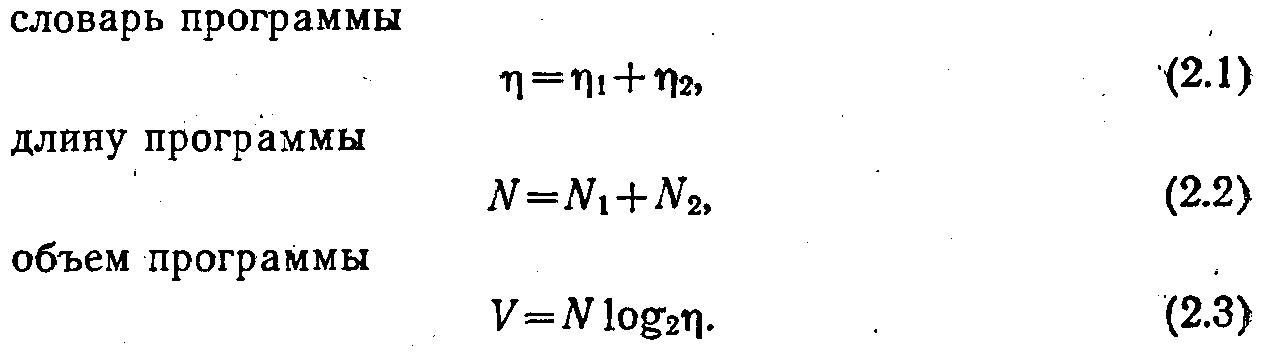
Справедливы следующие соотношения:

****

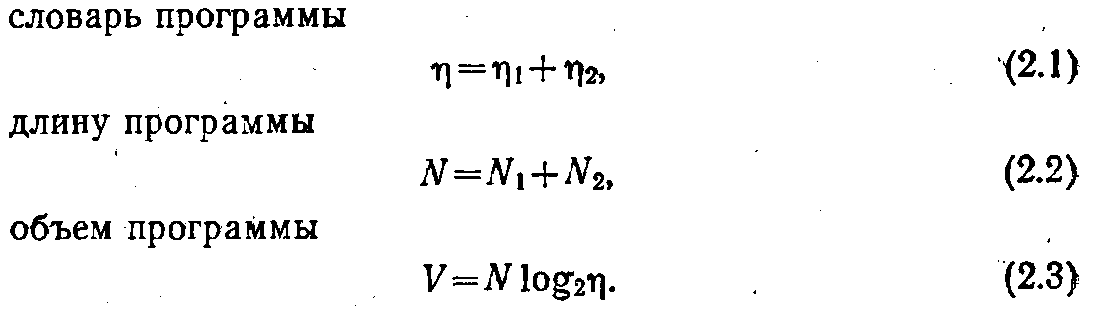
****

Базовые метрики определяются непосредственно при анализе исходных текстов программ. На основе базовых метрик Холстед предложил рассчитывать ряд производных метрик программы. Среди них рассмотрим следующие:

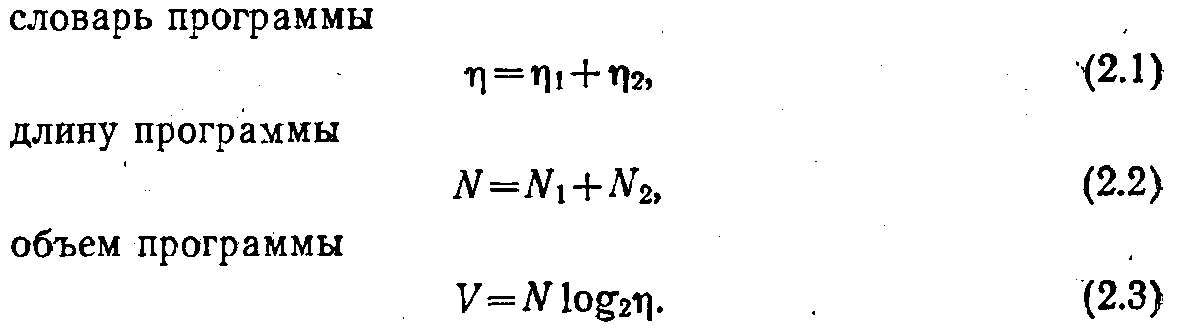
* словарь программы (общее число уникальных операторов и операндов программы):



* длина программы (общее количество операторов и операндов программы):



* объем программы (число битов, т.е. логических единиц информации, необходимых для записи программы):



Операнды программы представляют собой используемые в ней переменные и константы.

Под операторами программы в метриках Холстеда подразумеваются входящие в ее состав символы операций, символ присваивания, символы-разделители точка и точка с запятой, круглая скобка (пара из открывающей и закрывающей скобок считается одним оператором), управляющие операторы, составной оператор, а также имена процедур и функций.

Несколько служебных слов, входящих в состав одного оператора (например, If…Then…Else), считаются одним оператором.

Метки не относятся ни к операторам, ни к операндам.

Очевидно, что совокупность операторов программы и их количество зависят от языка программирования, на котором написана программа.

Операторы языка Паскаль в интерпретации Холстеда приведены в табл. 1. При подсчете количества операторов и операндов в программе, написанной на языке Паскаль, следует анализировать только ее раздел операторов, а также разделы операторов процедур и функций пользователя.

Таблица 1

**Операторы языка Паскаль в интерпретации Холстеда**

|  |  |
| --- | --- |
| Обозначение оператора | Назначение оператора |
| **+** | плюс (сложение, объединение множеств, сцепление строк) |
| ***–*** | минус (изменение знака, вычитание, разность множеств) |
| **\*** | звездочка (умножение, пересечение множеств) |
| **/** | наклонная черта, слэш (знак деления, результат всегда имеет вещественный тип) |
| **<** | меньше |
| **>** | больше |
| **=** | равно |
| **.** | точка (признак конца программы и модуля) |
| **;** | точка с запятой (разделитель операторов программы) |
| **( )** | левая и правая скобки при выделении подвыражений |
| **<=** | меньше или равно |
| **>=** | больше или равно |
| **<>** | не равно |
| **:=** | операция присваивания |
| **^** | знак карата (обращение к динамической переменной) |
| **And** | операция поразрядного логического сложения (И) |
| **Not** | операция поразрядного дополнения (НЕ) |
| **Or** | операция поразрядного логического сложения (ИЛИ) |
| **Xor** | операция поразрядного логического исключающего ИЛИ |
| **Div** | целочисленное деление |
| **Mod** | остаток от целочисленного деления |
| **Shl** | операция сдвига влево |
| **Shr** | операция сдвига вправо |
| **In** | операция проверки вхождения элемента в множество |
| **Begin…End** | составной оператор |
| **Break** | оператор безусловного выхода из цикла |
| **Continue** | оператор передачи управления на конец тела цикла |
| **Goto <Метка>** | оператор безусловного перехода |
| **Case…Of…Else…End** | оператор варианта |
| **If…Then…Else** | оператор условного перехода |
| **Repeat…Until** | оператор цикла с постусловием |
| **While…Do** | оператор цикла с предусловием |
| **For…To…Do** | оператор цикла с параметром (с увеличением параметра) |
| **For…Downto…Do** | оператор цикла с параметром (с уменьшением параметра) |
| **With…Do** | оператор присоединения |

**Пример 1.** Расчет метрик Холстеда для программы, вычисляющей значение функции

***Y = sin X***

через разложение функции в бесконечный ряд

f156

с точностью ***Eps = 0,0001***.

Текст программы, написанной на языке Паскаль, приведен ниже.

*Program Sin1;*

*Const*

*eps = 0.0001;*

*Var*

*y, x****:*** *real; n****:*** *integer; vs****:*** *real;*

*Begin*

*Readln (x);*

*y := x;* {Начальные установки}

*n := 2;*

*vs := x;*

***Repeat***

*vs := –vs \* x \* x / (2 \* n – 1) / (2 \* n –2);* {Формирование слагаемого}

*n := n + 1;*

*y := y + vs*

***Until abs(vs) < eps;* {Выход из цикла по выполнению условия}**

*Writeln (x, y, eps)*

*End.*

Расчет базовых метрик Холстеда для данной программы приведены в табл. 2.

Таблица 2 Расчет базовых метрик Холстеда

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ***j*** | **Оператор** | ***f1j*** | ***i*** | **Операнд** | ***f2i*** |
|  | ; | 7 |  | x | 6 |
|  | := | 6 |  | n | 5 |
|  | \* | 4 |  | vs | 5 |
|  | – | 3 |  | y | 4 |
|  | / | 2 |  | 2 | 4 |
|  | ( ) | 2 |  | 1 | 2 |
|  | + | 2 |  | eps | 2 |
|  | Begin…End | 1 |  |  |  |
|  | Readln ( ) | 1 |  |  |  |
|  | Repeat…Until | 1 |  |  |  |
|  | abs( ) | 1 |  |  |  |
|  | < | 1 |  |  |  |
|  | Writeln ( ) | 1 |  |  |  |
|  | **.** | 1 |  |  |  |
| **η1 *=* 14** |  | ***N1 =* 33** | **η2 *=* 7** |  | ***N2 =* 28** |

Словарь программы **η**=14 + 7 = 21.

Длина программы ***N*** = 33 + 28 = 61.

Объем программы ***V*** = 

**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 *Е —* дуги, соответствующие переходам между операторами. В дуге *(v, и)* вершина *v* является исходной, а *и —* конечной. При этом *и* непосредственно следует за *v,* a *v* непосредственно предшествует *и.* Если путь от *v* до *u* состоит более чем из одной дуги, тогда *и* следует за *v,* а *v* предшествует *и* [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 – ʋ + 2p*,**

где *е —* число дуг ориентированного графа *G; ʋ —* число вершин; *р —* 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 *е* = 8, число вершин *ʋ* = 7*,* *р* = 1. Цикломатическое число Маккейба равно *Z(G)* = 8 – 7 +2 = 3.

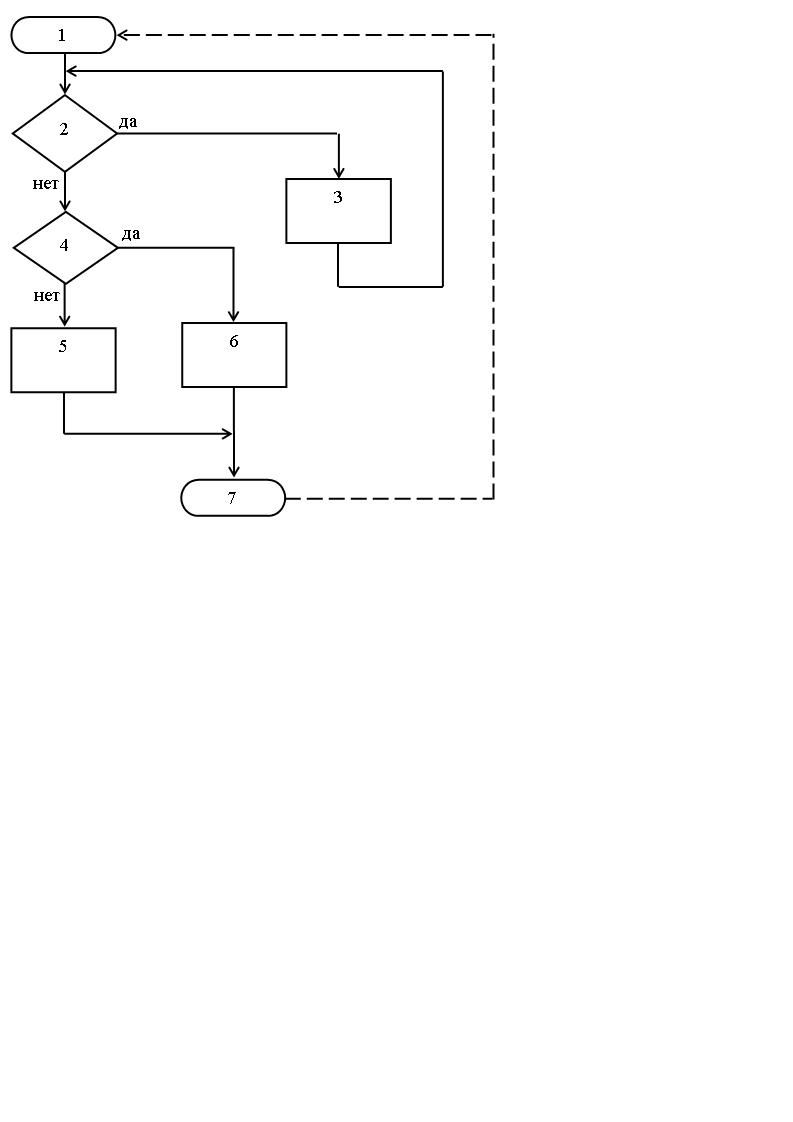


Рисунок 1 – Пример схемы алгоритма программы

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).

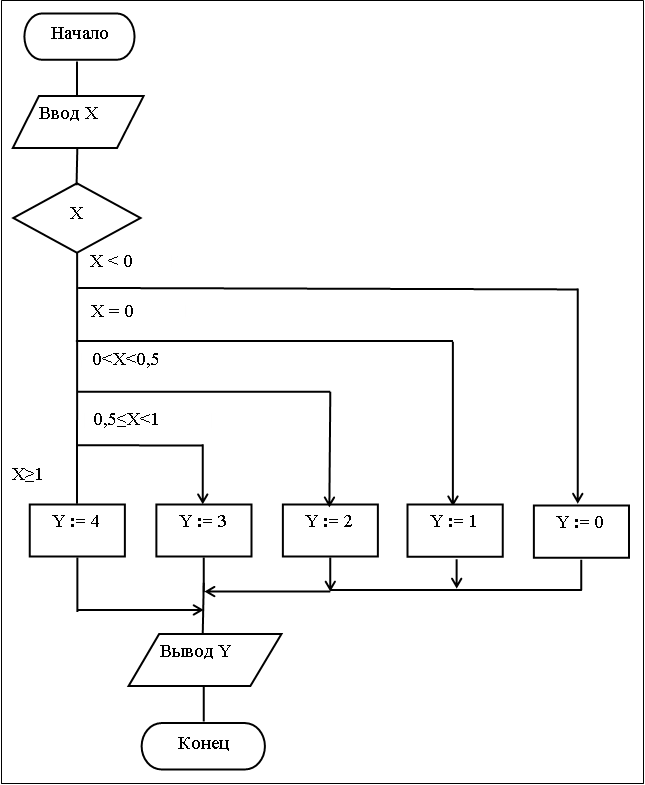


Рисунок 2 – Схема разветвляющегося алгоритма вычисления функции Y (используется символ «Решение с многими выходами»)

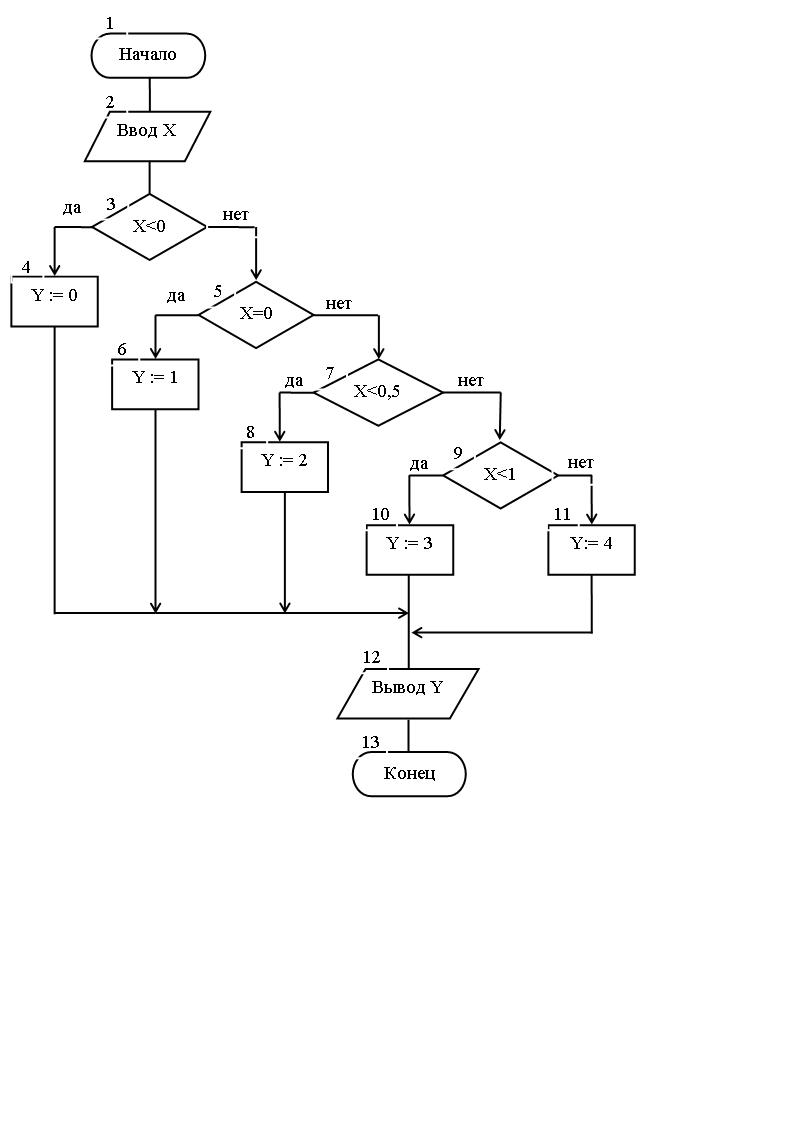


Рисунок 3 – Схема разветвляющегося алгоритма вычисления функции Y (используется символ «Решение с двумя выходами»)

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* *Sa* определяется как сумма скорректированных сложностей всех вершин графа *G*.

*Относительная граничная сложность программы* *So* определяется по формуле:

****

где *So –* относительная граничная сложность программы; *Sa —* абсолютная граничная сложность программы; *ʋ –* 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 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Скорректированная сложность вершины графа | 1 | 1 | 9 | 1 | 7 | 1 | 5 | 1 | 3 | 1 | 1 | 1 | 0 | ***Sa* = 32** |

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

*So*= l – (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 |  |  |  |  |  |  |  |  |
| Скорректированная сложность вершины графа | 1 | 3 | 1 | 3 | 1 | 1 | 0 | ***Sa* = 10** |

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

*So*= l – (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**

|  |  |  |  |
| --- | --- | --- | --- |
| 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) *Sa* | 10 | 14 | 32 |
| Метрика граничных значений (относительная граничная сложность программы) *S0* | 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.

**Метрики сложности потока данных**

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

К наиболее известным в данной группе метрик можно отнести спен и метрику Чепина [2].

***Спен идентификатора*** – это число повторных появлений идентификатора (число появлений после его первого появления) в тексте программы. Идентификатор, встречающийся в тексте программы *п* раз, имеет спен, равный *п—*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 в программе переменных.

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. *Р –* вводимые переменные, содержащие исходную информацию, но не модифицируемые в программе и не являющиеся управляющими переменными;

2. *М –* модифицируемые переменные и создаваемые внутри программы константы и переменные, не являющиеся управляющими переменными;

3. *С –* переменные, участвующие в управлении работой программного модуля (управляющие переменные);

4. *Т –* 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 = a1 p + a2 m + a3 c + a4 t ,*

где *а1, a2,* *a3, a4* – весовые коэффициенты; *p, m, c, t* – количество переменных в группах *Р, М, С, Т* соответственно.

Весовые коэффициенты позволяют учитывать различное влияние на сложность программы каждой функциональной группы. Наиболее часто применяются следующие значения весовых коэффициентов: *а1* = l, *а2* = 2, *а3* = 3, *а4* = 0,5. С учетом данных значений формула для определения метрики Чепина принимает вид :

*Q = p* + 2*m* + 3*c* + 0,5*t .*

Помимо полной метрики Чепина распространен ее вариант, при котором анализу и разбиению на четыре группы подвергаются только переменные из списка ввода/вывода программы, то есть те переменные, которые содержатся в списке параметров операторов ввода/вывода программы. Назовем данный вариант метрикой Чепина ввода/вывода.

Табл. 8, 9 содержат метрики сложности потока данных для программы, вычисляющей значение функции

***Y = sin X .***

Исходный текст программы приведен в примере 1.

Таблица 8 – **Спен программы**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Идентификатор | х | n | vs | y | eps | Суммарный спен программы |
| Спен | 6 | 5 | 5 | 4 | 2 | 22 |

В тексте программы (см. пример 1) идентификаторы впервые встречаются в разделе объявлений. Поэтому значение спена идентификатора равно количеству его появлений в разделе операторов, то есть значению *f2i* в метриках Холстеда.

Таблица 9

**Метрики Чепина программы**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Полная метрика Чепина | | | | Метрика Чепина ввода/вывода | | | |
| Группа переменных | Р | М | С | Т | Р | М | С | Т |
| Переменные, относящиеся к группе | x | y, n | vs, eps | -- | x | y | eps | -- |
| Количество переменных в группе | р = 1 | m = 2 | c = 2 | t = 0 | р = 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 | | | |

В список переменных ввода/вывода программы (см. пример 1) входят переменные *х* (оператор ввода *Readln (x)*) и *y, z* (оператор вывода *Writeln (x, y, eps)*). Остальные переменные (*n, vs*) в расчете метрики Чепина ввода/вывода не участвуют.