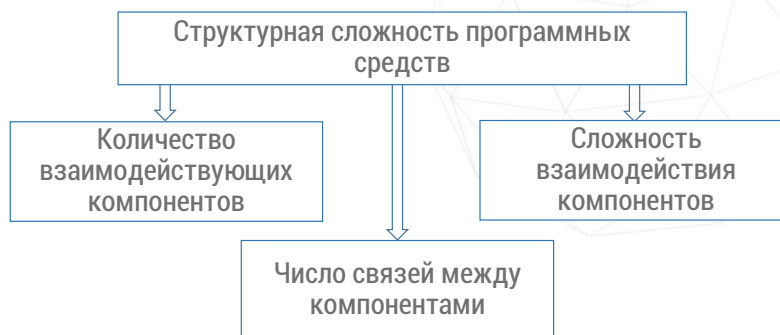


ПОНЯТИЕ СТРУКТУРНОЙ СЛОЖНОСТИ ПРОГРАММ



Тема 3.3. Оценка структурной сложности программ

Рассмотрим метрики, определяющие структурную сложность программ. К ним относятся:

- «количество взаимодействующих компонентов;
- число связей между компонентами;
- сложность взаимодействия компонентов» [1].

Как было отмечено ранее, при работе программы

многообразие ее поведения и разнообразие связей между ее входными и результирующими данными определяются набором маршрутов, по которым исполняется программа.

В данном контексте маршрут представляет собой последовательность, состоящую из вершин и дуг графа управления.

Сложность программных модулей связана не столько с размером программы, которая определяет количество выполняемых команд, сколько с числом маршрутов ее исполнения, а также с их сложностью.

При создании качественной программы основную сложность ее разработки определяют маршруты возможной обработки данных. Эти маршруты следует тщательно проверить.

Такую метрику сложности, связанную с анализом маршрутов, можно использовать для оценки трудоемкости тестирования и сопровождения

модуля. Эту же метрику можно применить при оценке потенциальной надежности его функционирования.

«Проведенные исследования подтверждают высокую адекватность использования структурной сложности программ для оценки трудоемкости тестирования, вероятности ненайденных ошибок и затрат на разработку программных модулей в целом» [1].

Маршруты исполнения программы условно можно разделить на две группы: вычислительные маршруты и маршруты принятия логических решений. Их мы рассмотрим далее.

МАРШРУТЫ ИСПОЛНЕНИЯ ПРОГРАММ



Сложность вычислительных маршрутов:

$$S_1 = \sum_{i=1}^m \sum_{j=1}^{l_i} v_j, \quad (33)$$

где m – количество маршрутов исполнения программы; l_i – число данных, обрабатываемых в i -м маршруте; v_j – число значений обрабатываемых данных j -го типа ($2 \leq v_j \leq 5$).

Сложность маршрутов принятия логических решений:

$$S_2 = \sum_{i=1}^m p_i, \quad (34)$$

где p_i – число ветвлений или число проверяемых условий в i -м маршруте.

Общая сложность программы:

$$S = c \cdot \sum_{i=1}^m \left(\sum_{j=1}^{l_i} v_j + p_i \right), \quad (35)$$

где c – коэффициент пропорциональности, корректирующий взаимное использование метрик сложности вычислительных маршрутов и маршрутов принятия логических решений.



«Группа вычислительных маршрутов объединяет в себе маршруты арифметической обработки данных. Эта группа предназначена для непосредственного преобразования величин, являющихся элементарными результатами измерения каких-либо характеристик.

Для проверки вычислительных маршрутов можно сформировать достаточно простую стратегию

их проверки.

Во всём диапазоне преобразования входных переменных нужно выбрать несколько характерных точек. В этих точках проверяется работоспособность и корректность работы программы. К таким контрольным точкам можно отнести предельные значения, значения в точках разрыва, несколько промежуточных значений» [1].

Сложность вычислительных маршрутов можно оценить по формуле 33. Расчет сложности программы по этой формуле имеет высокую неопределенность. Это обусловлено отсутствием конкретных требований к выбору количества значений, обрабатываемых данных при изменении входных данных.

Удельный вес вычислительной части во многих сложных программных комплексах обработки информации относительно невелик. Поэтому

вычислительные маршруты не играют доминирующей роли в определении структурной сложности программ. Это связано с тем, что проведение вычислений подчиняется определенным правилам, несущественно влияющим на логику проведения этих вычислений.

«Группа маршрутов принятия логических решений объединяет пути, отражающие логику выполнения программы, которая может:

- изменять последовательность выполнения команд;
- переводить управление на удаленные участки программного модуля или досрочно завершать его выполнение.

Все эти факторы могут значительно влиять на сложность программы» [1].

Сложность маршрутов принятия логических решений оценивается по формуле 34.

Общая сложность программы определяется по формуле 35 с учетом сложности вычислительных маршрутов и сложности маршрутов принятия логических решений.

КРИТЕРИИ ВЫДЕЛЕНИЯ МАРШРУТОВ



По каждому критерию анализируется граф потока управления.

Граф потока управления – ориентированный граф, моделирующий поток управления программой.

Поток управления – последовательность выполнения различных модулей и операторов программы.



Выделение маршрутов исполнения программы для оценки структурной сложности может осуществляться по различным критериям. Наилучшим считается такой критерий, который позволит выделить все возможные маршруты исполнения программы при любых сочетаниях исходных и промежуточных данных.

Формирование маршрутов зависит не только от

структуры программы, но и от значений переменных в различные моменты времени и на различных участках выполнения программы. Такое выделение маршрутов трудно формализовать. Оно представляется весьма трудоемким для оценки показателей сложности программной структуры.

«Рассмотрим более простые критерии выделения маршрутов, учитывающие только структурные характеристики программных модулей. При этом будем анализировать граф потока управления. Под ним понимается множество всех возможных путей исполнения программы, представленное в виде графа.

Граф потока управления – это ориентированный граф, моделирующий поток управления программой. Поток управления представляет собой последовательность выполнения различных модулей и операторов программы» [1].

Критерии для оценки сложности программных модулей характеризуют минимально необходимые способы проверок. Для проверки реальных программ этого количества проверок может оказаться недостаточно. Например, циклы целесообразно проверять не однократно, а на нескольких промежуточных значениях. Кроме того, следует проверять их на максимальном и минимальном количествах исполнения циклов.

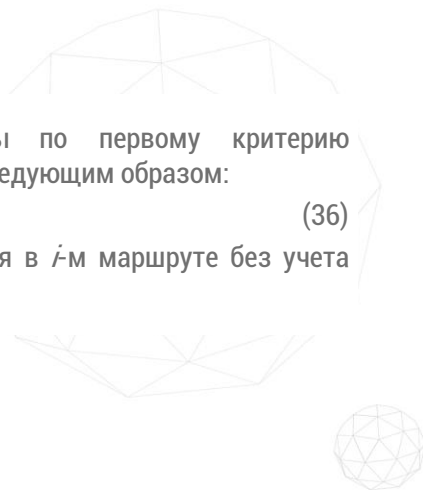
Оценить достаточность проверок программы значительно труднее. Дело в том, что помимо сложности структуры необходимо анализировать сложность преобразования каждой переменной во всём диапазоне ее изменения и при сочетаниях с другими переменными.

ПЕРВЫЙ КРИТЕРИЙ

Структурная сложность программы по первому критерию выделения маршрутов вычисляется следующим образом:

$$S_1 = \sum_{i=1}^m p_i, \quad (36)$$

где p_i – количество вершин ветвления в i -м маршруте без учета последней вершины.



«Первый критерий оценки структурной сложности предполагает, что граф потока управления программы проверяется по минимальному набору маршрутов, проходящих через каждый оператор ветвления по каждой дуге.

Прохождение по каждому маршруту осуществляется только один раз, повторная проверка дуг не проводится и считается избыточной. При этом

в процессе проверки гарантируется выполнение всех передач управления между операторами программы и каждого оператора не менее одного раза» [1].

В настоящее время существуют алгоритмы, позволяющие автоматизировать процесс получения минимального множества маршрутов по этому критерию.

Для оценки структурной сложности программы строится граф потока управления, который содержит определенное число вершин и дуг. Выделяют вершины, в которых происходит ветвление процесса, и подсчитывают моменты ветвления при прохождении маршрутов.

Структурная сложность программы по первому критерию выделения маршрутов вычисляется по формуле 36.

Недостатком первого критерия считается то, что

он не учитывает комбинаторику сочетаний условий на разных участках маршрутов.

ВТОРОЙ КРИТЕРИЙ

Второй критерий выделения маршрутов основан на анализе базовых маршрутов в программе, которые оцениваются на основе цикломатического числа исходного графа потока управления программой:

$$Z = m - n + 2 \cdot p, \quad (37)$$

где m – общее число дуг в графе; n – общее число вершин в графе; p – число связных компонентов графа.

Для правильно структурированных программ цикломатическое число можно представить следующим образом:

$$Z = n_{\text{в}} + 1, \quad (38)$$

где $n_{\text{в}}$ – число вершин, в которых происходит ветвление.



«Второй критерий оценки структурной сложности основан на анализе базовых маршрутов в программе, которые оцениваются на основе цикломатического числа исходного графа потока управления программой, формула 37.

Число связных компонентов графа равно количеству дуг, необходимых для превращения исходного графа в максимально связный граф.

Максимально связным графом называется такой граф потока управления, у которого любая вершина доступна из любой другой вершины.

Максимально связный граф может быть получен из исходного графа потока управления программой путем замыкания конечной и начальной вершин. Под корректными понимаются такие графы потоков управления, у которых не содержится висячих и тупиковых вершин» [1].

Второй критерий, выбранный по цикломатическому числу, требует однократной проверки или тестирования каждого линейно независимого цикла и каждого линейно независимого ациклического участка, не имеющего циклических участков программы.

Каждый линейно независимый ациклический маршрут или цикл отличается от всех остальных хотя бы одной вершиной или дугой.

При использовании второго критерия количество проверяемых маршрутов равно цикломатическому числу.

Для правильно структурированных программ характерно отсутствие циклов с несколькими выходами. Такие программы не имеют переходов внутрь циклов или условных операторов. Они также не имеют принудительных выходов из внутренней части циклов или условных операторов.

Цикломатическое число для таких программ можно определить по формуле 38.

Для автоматического анализа графов по второму критерию с помощью средств вычислительной техники используются матрицы смежности и достижимости графов. Эти матрицы содержат информацию о структуре проверяемой программы.

ТРЕТИЙ КРИТЕРИЙ

Третий критерий выделения маршрутов основан на формировании полного состава базовых структур графа потока управления программой.

Заключается в анализе каждого из реальных ациклических маршрутов исходного графа программы и каждого цикла, достижимого из всех этих маршрутов.

Каждый из указанных компонентов структуры программы должен быть проанализирован хотя бы один раз.



Более сильные критерии проверки и определения структурной сложности программы включают требования однократной проверки не только линейно независимых, но и всех линейно зависимых циклов и ациклических маршрутов.

К таким критериям относится третий критерий формирования маршрутов для тестирования программ.

«Если при прохождении какого-либо ациклического маршрута исходного графа потока управления достижимы несколько элементарных циклов, при тестировании должны исполняться все достижимые циклы» [1].

Приведенные критерии для оценки сложности программных модулей характеризуют в каждом случае минимально необходимые величины проверок по каждому критерию.

Для проверки реальных программ этого количества проверок может быть недостаточно.

Как говорилось ранее, циклы, к примеру, имеет смысл проверять не один раз, а на нескольких промежуточных значениях. А также на максимальном и минимальном количествах исполнения циклов.

МЕТРИКА МАККЕЙБА (THOMAS G. MCCABE)

Метрика Маккейба характеризует цикломатическое число графа потока управления программы и определяется следующим выражением:

$$M = m - n + 2, \quad (39)$$

где m – количество ребер графа; n – число вершин графа.



Часто используемой метрикой оценки структурной сложности программ является метрика Маккейба. Она основана на результатах анализа потока управления от одного оператора к другому. Это позволяет учесть логику построения программы при оценке ее сложности.

Автор предложил «стратегию проверки корректности программного средства, которая

получила название основного маршрута тестирования Маккейба.

Программное средство должно быть представлено в виде управляющего ориентированного графа. Вершины в нем соответствуют операторам, а дуги характеризуют переход управления от одного оператора к другому.

Граф, описывающий программу в виде вершин-операторов и дуг-переходов, называют графом потока управления программы, или управляющим графом программы» [1].

Для представления программы в виде графа необходимы определенные соглашения, определяющие положения о том, что считать узлом графа. Дело в том, что синтаксис операторов в различных языках программирования может существенно отличаться. При этом учитывают только исполнимые операторы и исключают группы

операторов, назначением которых является описание данных.

«Линейные участки программы можно заменить одним узлом графа, относя к нему группы операторов, выполнение которых осуществляется в прямой последовательности без каких-либо условий.

При любом варианте представления программы желательно преобразовать операторы цикла в последовательность операторов ветвления. При необходимости можно добавить операторы подсчета числа повторений цикла с верхним или нижним окончанием» [1].

Метрика Маккейба, характеризующая цикломатическое число графа потока управления программы, определяется по формуле 39.

Величину, рассчитанную по этой формуле, называют цикломатическим числом Маккейба.

ЦИКЛОМАТИЧЕСКАЯ СЛОЖНОСТЬ ПРОГРАММЫ

Цикломатическая сложность программы – структурная (или топологическая) мера сложности программы, применяемая для измерения качества программного обеспечения и основанная на методах статического анализа кода ПС.

Цикломатическая сложность программы равна цикломатическому числу графа программы, увеличенному на единицу.



«Цикломатическая сложность программы – это структурная мера сложности программы. Она основана на методах статического анализа кода программного средства и применяется для измерения качества программного обеспечения.

При вычислении цикломатической сложности используется граф потока управления программы. Узлы графа соответствуют неделимым группам

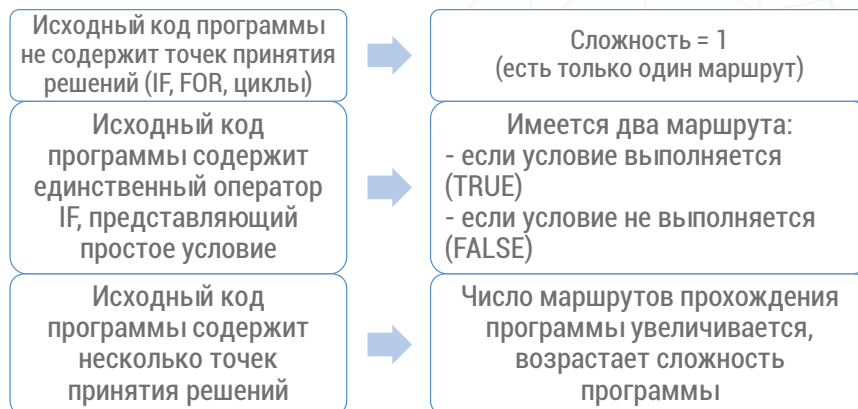
команд программы и ориентированным ребрам. Каждый из них соединяет два узла и соответствует двум командам, вторая из которых может быть выполнена сразу после первой.

Цикломатическая сложность может также быть применена для отдельных функций, модулей, методов или классов в пределах анализируемого программного средства» [1].

Данная стратегия тестирования называется основным маршрутом тестирования Маккейба. Это тестирование представляет собой проверку каждого линейно независимого маршрута через программу. При такой проверке программы количество тестов должно быть равно ее цикломатической сложности.

Цикломатическая сложность части программного кода является счетным числом линейно независимых маршрутов, проходящих через программный код.

ХАРАКТЕРИСТИКА ВЛИЯНИЯ ТОЧЕК ПРИНЯТИЯ РЕШЕНИЙ НА СЛОЖНОСТЬ ПРОГРАММЫ



Рассмотрим примеры влияния точек принятия решений на сложность программы. Если исходный код не содержит никаких точек решений, таких как указания условий *if* [иф] или границы циклов *for* [фо], то сложность должна быть равна единице. Объясняется это тем, что есть только единственный маршрут выполнения программного кода.

Если в тексте программы размещен

единственный оператор *if* [иф], содержащий простое условие, то должно быть два пути выполнения кода. Один путь – через оператор *if* [иф] с оценкой выполнения условия как ИСТИНА, другой – для альтернативной ветви при значении ЛОЖЬ, в котором заданное условие не выполняется.

Если исходный код программы содержит несколько точек принятия решений, сложность программы возрастает, так как число маршрутов выполнения программы увеличивается.

Теоретической основой определения цикломатического числа Маккейба является теория графов. Число компонентов связности можно рассматривать как минимально необходимое количество ребер, которые нужно добавить к графу, чтобы сделать его полносвязным.

«Полносвязным считается граф, у которого существует путь из любой вершины графа в любую

другую вершину графа» [1].

Для получения полносвязности графа достаточно добавления одной фиктивной дуги из его конечной вершины в начальную. Поэтому можно считать, что практически для любого графа управления программы число компонент связности равно единице, то есть $p = 1$ [пэ равно единице].

Подстановка значения $p = 1$ [пэ, равное единице] в формулу 37 определения цикломатического числа дает значение цикломатического числа Маккейба.

Цикломатическое число определяет количество независимых контуров в полносвязном графе и, как следствие, количество различных маршрутов, ведущих из начальной вершины в конечную.

При оценке сложности программы с применением цикломатического числа Маккейба справедливо следующее положение. Если значение цикломатического числа больше 10, это означает, что

программа обладает излишней сложностью. В таком случае ее следует разбить на составные части, для которых цикломатическое число будет иметь меньшее значение.

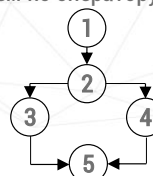
ОСОБЕННОСТИ ПОСТРОЕНИЯ УПРАВЛЯЮЩИХ ГРАФОВ

Пример построения графа для программы на языке программирования C вычисления максимального значения двух чисел.

Фрагмент программы с разветвлением по оператору *if*:

```
1 scanf ("%f %f", &a,&b);  
2 if (a > b)  
3   c = a  
   else  
4   c = b;  
5 printf ("%f", c);
```

Граф управления программы с разветвлением по оператору *if*:



Характеристики графа сложности фрагмента программы: количество дуг $m = 5$; количество вершин $n = 5$; цикломатическое число Маккейба $Z = m - n + 2 = 2$.



Пример графа управления для типовых блоков, реализуемых основными операторами языков программирования, приведен на схеме.

Ориентированный управляющий граф позволяет учесть логику программы, обеспечивая возможность анализа потока передачи управления между операторами. Как отмечалось ранее, этот граф формируется из вершин, которые соответствуют

операторам программы, и дуг, задающих переходы между этими операторами.

Отметим несколько общих замечаний, справедливых практически для любых программных блоков.

При построении графа управления программой следует учитывать только исполнимые операторы. Операторы описания не учитываются.

Допускается объединять операторы в одну вершину графа. Это возможно, если в программе существует несколько операторов, не изменяющих порядок действий, и они следуют один за другим.

При построении графа управления операторы цикла следует заменить несколькими вершинами. Например, для оператора типа *for* [фо] должны быть вершины, соответствующие операторам начального присваивания счетчика цикла, его изменению и ветвлению, определяющим продолжение или

завершение выполнения цикла.

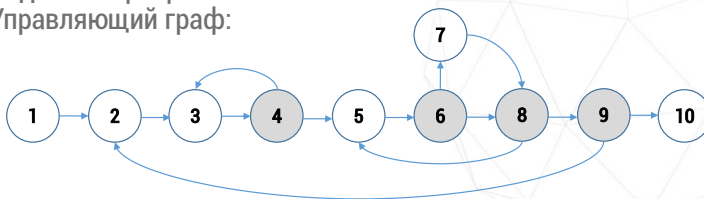
Аналогично несколькими вершинами заменяют и другие операторы цикла. Они должны соответствовать действиям, предшествующим циклу, определяющим продолжение или прекращение выполнения цикла. А также группе операторов, составляющих тело цикла, – множеству повторяемых действий.

В графе должны быть три группы вершин, определяющих три элемента, составляющих любой цикл: начало цикла, тело цикла, ветвление при окончании цикла – завершение или возвращение к исходному оператору.

Правильно построенный управляющий граф позволяет оценить структурную или топологическую сложность программы, определяемую количеством независимых контуров в полносвязном графе.

ПРИМЕР ОЦЕНКИ СТРУКТУРНОЙ СЛОЖНОСТИ ПРОГРАММЫ

Задача «Сортировка массива».
Управляющий граф:



Возможный маршрут управления:

m_1 : 1-2-3-4-3-4-5-6-7-8-5-6-8-9-2-3-4-5-6-8-9-10; $p_1 = 11$.

Количество вершин ветвления в маршруте: $S_1 = p_1 = 11$.

Количество проверок: $Z = n_b + 1 = 4 + 1 = 5$,

где n_b – число вершин ветвления.

Рассмотрим в качестве примера оценки структурной сложности программы задачу сортировки массива. Необходимо провести сортировку значений элементов одномерного массива по возрастанию. По разработанному алгоритму требуется построить управляющий граф и оценить алгоритмическую сложность возможной программы.

В соответствии с разработанным алгоритмом

решения задачи сортировки массива построим граф потока управления, который представлен в виде схемы. Проведем оценку алгоритмической сложности графа по первому критерию. Для этого определим минимальный набор маршрутов, проходящих через каждый оператор ветвления и по каждой дуге.

Анализ выделенного маршрута показывает, что он проходит по всем вершинам ветвления и по меньшей мере один раз – по каждой дуге представленного графа.

Минимальное количество маршрутов для заданного графа равно единице, так как этим маршрутом охватываются все дуги и вершины графа. Тонированные вершины обозначают операторы ветвления, в перечне пунктов маршрута номера́ этих вершин выделены шрифтом с подчеркиванием. Количество таких пунктов маршрута – 11.

Таким образом, в соответствии с первым

критерием алгоритмической сложности количество маршрутов равно единице, количество вершин ветвления в маршруте – 11.

Проведем оценку алгоритмической сложности по второму критерию. Необходимо определить число проверок каждого линейно независимого цикла и линейно независимого ациклического участка программы.

Количество проверок определяется цикломатическим числом графа как число вершин ветвления плюс единица. Число вершин ветвления в заданном графе равно четырем. Следовательно, общее число циклических и ациклических маршрутов составляет 5.

ПРИМЕР ОЦЕНКИ СТРУКТУРНОЙ СЛОЖНОСТИ ПРОГРАММЫ

Ациклические маршруты:

$m1: 1-2-3-4-5-6-7-8-9-10-11-12-13; p_1 = 4;$

$m2: 1-2-3-4-5-6-7-9-10-11-12-13; p_2 = 4.$

Циклические маршруты:

$m3: 3-4; p_3 = 1;$

$m4: 5-6-8; p_4 = 2;$

$m5: 2-3-4-5-6-7-8-9; p_5 = 4.$

Метрика структурной сложности: $S_2 = 4 + 4 + 1 + 2 + 4 = 15.$

Матрица смежности:

	1	2	3	4	5	6	7	8	9	10
1										
2	1								1	
3		1		1						
4			1							
5				1				1		
6					1					
7						1				
8						1	1			
9								1		
10									1	

Выделим циклические и ациклические маршруты на заданном графе. Тестирование указанных маршрутов позволяет проверить все операторы ветвления программы.

Расчетная метрика структурной сложности по второму критерию равна 15.

Для организации автоматического анализа заданного графа по второму критерию с помощью

вычислительных средств строятся матрицы смежности и достижимости.

Матрица смежности представляет собой квадратную матрицу, размер которой определяется количеством вершин графа. Заданный граф имеет десять вершин. Следовательно, матрица смежности будет иметь размер 10×10 .

При заполнении матрицы рекомендуется придерживаться следующего правила: для дуги, соединяющей вершину 1 с вершиной 2, в первый столбец и вторую строку матрицы смежности записываем единицу.

Номер столбца соответствует номеру вершины, из которой выходит дуга. Номер строки соответствует номеру вершины, в которую входит дуга. Таким же образом заполняем матрицу для всех дуг управляющего графа.


Для выделения маршрутов можно использовать

матрицу достижимости. Для заданного графа управления она представляет собой квадратную таблицу размером 10×10 .

Номер столбца матрицы определяет номер вершины графа, из которого возможно достичь другие вершины этого же графа, используя цикличные и ацикличные маршруты. Номера́ строк определяют номера достижимых вершин графа управления.

ПРИМЕР ОЦЕНКИ СТРУКТУРНОЙ СЛОЖНОСТИ ПРОГРАММЫ

Матрица достижимости:



	1	2	3	4	5	6	7	8	9	10
1										
2	1		1	1	1	1	1	1	1	1
3	1	1		1	1	1	1	1	1	1
4	1	1	1		1	1	1	1	1	1
5	1	1	1	1		1	1	1	1	1
6	1	1	1	1	1		1	1	1	1
7	1	1	1	1	1	1		1	1	1
8	1	1	1	1	1	1	1		1	1
9	1	1	1	1	1	1	1	1		1
10	1	1	1	1	1	1	1	1	1	

Возможные маршруты управления:

$m1$: 1-2-3-4-5-6-7-8-9-10; $p_1 = 4$; $m2$: 1-2-3-4-5-6-8-9-10; $p_2 = 4$;

$m3$: 1-2-3-4-3-4-5-6-7-8-5-6-8-9-10; $p_3 = 6$;

$m4$: 1-2-3-4-5-6-8-9-2-3-4-5-6-8-9-10; $p_4 = 8$.

Оценка структурной сложности программы: $S_3 = 4 + 4 + 6 + 8 = 22$.



Из вершины 1 возможно достичь вершины со второй по десятую, то есть достижимы все вершины графа. Для первого столбца матрицы достижимости строки, начиная со второй, заполняются единицами. Далее аналогично заполняются столбцы для остальных вершин.

Выделенные диагональные элементы матрицы определяют номера столбцов-вершин, которые

ВХОДЯТ В СОСТАВ ЦИКЛИЧЕСКИХ МАРШРУТОВ. Идентичные строки матрицы определяют номера столбцов-вершин, которые входят в состав ациклических маршрутов.

Элементы, выделенные штриховкой в левую сторону, соответствуют маршруту $m3$ [эм три]. Элементы со штриховкой в правую сторону – маршруту $m4$ [эм четыре].

Элементы, помеченные единицами полужирного шрифта белого и черного цвета, соответствуют маршруту $m5$ [эм пять]. Элементы, помеченные единицами белого цвета, соответствуют маршруту $m6$ [эм шесть].

Все строки матрицы получились идентичными, так как все вершины графа входят в состав ациклических маршрутов $m1$ [эм один] и $m2$ [эм два].

Проведем оценку алгоритмической сложности по третьему критерию. В соответствии с ним

необходимо выделить все реально возможные маршруты управления.

Итак, получены результаты расчета метрик структурной сложности: по первому критерию – 11, по второму критерию – 15, по третьему – 22. По критериям выделения маршрутов можно сделать вывод, что программа имеет невысокую алгоритмическую сложность.

Это связано с тем, что количество используемых в тексте операторов условий равно четырем. Для их проверки необходимо провести от 11 до 22 тестовых вариантов исходных данных.