Общие принципы

1. При построении графа управления программой надо учитывать только исполнимые операторы, не учитывая операторы описания.

2. Если в программе существует несколько операторов, не изменяющих порядок действий в программе, и они следуют один за другим, их допускается объединять в одну вершину графа.

3. При построении графа управления операторы цикла следует заменить несколькими вершинами:

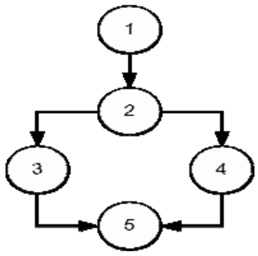
- операторы начального присваивания счетчика цикла;
- операторы тела цикла;
- операторы, определяющие продолжение или завершение выполнения цикла.

Линейная последовательность операторов

$m = 3; n = 4; Z = m – n + 2 = 1$

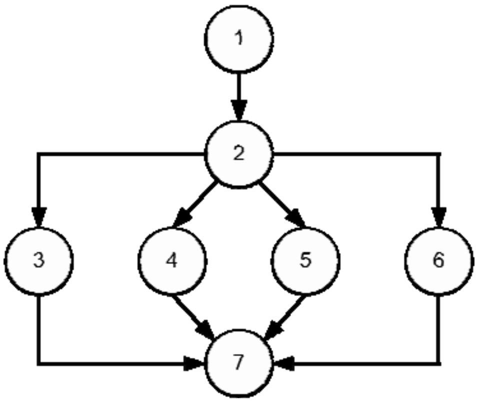
$m = 1; n = 2; Z = m – n + 2 = 1$

Простое ветвление (оператор IF)



$m = 5; n = 5; Z = m – n + 2 = 2$

Переключатель с множественным выбором (switch, select, case)



$m = 9; n = 7; Z = m – n + 2 = 4$

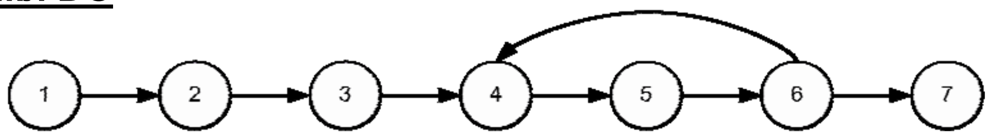
Фрагменты с операторами цикла

Бывают с постусловием и предусловием.

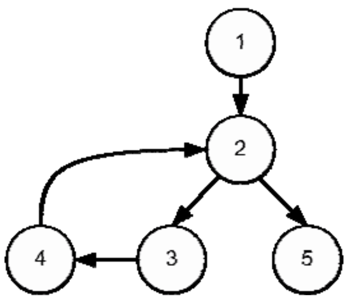
Надо заменить оператор цикла программы на эквивалентную последовательность операторов с применением операторов присваивания и вычисления условий:

- начальные присваивания до выполнения цикла;
- повторяющиеся действия (тело цикла);
- условия окончания (или продолжения) цикла.

Цикл DO



Цикл WHILE



Задача «Расчет значений функции»

Вычислить значение функции $G = F(x, y)$, если

$$G = \begin{cases} true, & \text{если точка с координатами } (x, y) \text{ попадает в фигуру;} \\ false, & \text{если точка с координатами } (x, y) \text{ не попадает в фигуру.} \end{cases}$$

Фигура представляет собой сектор круга радиусом $R = 2$ в диапазоне углов $270° \leq f_i \leq 45°$.

Разработать программу. В соответствии с разработанной программой составить блок-схему алгоритма решения задачи. На основании блок-схемы составить управляющий граф и оценить алгоритмическую сложность программы с использованием метрики Маккейба.

Реализация решения

Текст программы для реализации возможного решения поставленной задачи, разработанной с использованием языка программирования *C#*, приведен на рис. 1.

Номера строк	Строки программы
1	using System;
2	class Operator
3	{
4	public static void Main()
5	{
6	const double R = 2.0;
7	double x, y; bool g;
8	char rep;
9	string str;
10	do
11	{
12	Console.Clear();
13	Console.Write("Введите X: ");
14	str = Console.ReadLine();
15	x = double.Parse(str);
16	Console.Write("Введите Y: ");
17	str = Console.ReadLine();
18	y = double.Parse(str);
19	g = false;
20	if (x * x + y * y <= R * R)
21	if (x >= 0)
22	if (y <= x)
23	g = true;
24	str = string.Format("G({0:f3},{1:f3}) = {2}", x, y, g);
25	Console.WriteLine(str);
26	Console.Write("Для повтора вычислений намите клавишу Y: ");
27	rep = char.Parse(Console.ReadLine());
28	Console.WriteLine();
29	} while (rep == 'Y' || rep == 'y');
30	}
31	}

Рис. 1. Программа расчета значения функции

Алгоритм решения задачи выглядит так, как показано на рис. 2.

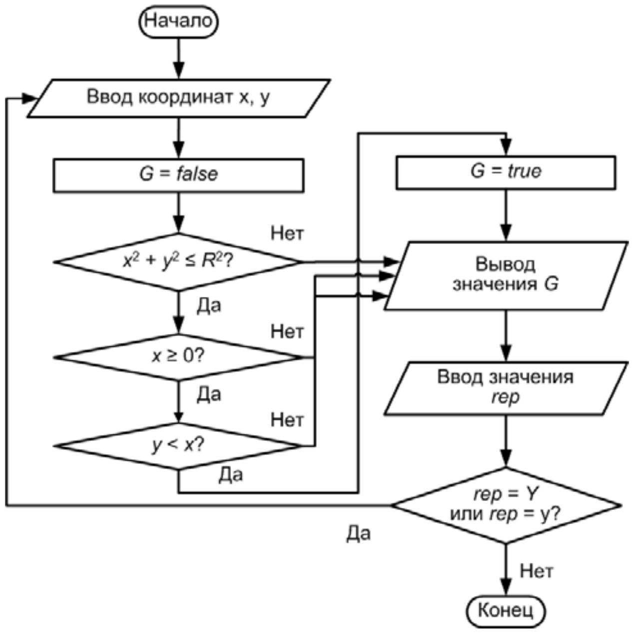


Рис. 2. Алгоритм вычисления значений функции

Оценка алгоритмической сложности

Граф потока управления для задачи расчета значений функции приведен на рис. 3. Тонированные вершины обозначают операторы ветвления.

Первый критерий

Проведем оценку алгоритмической сложности графа **по первому критерию**. Определим минимальный набор маршрутов, проходящих через каждый оператор ветвления и по каждой дуге.

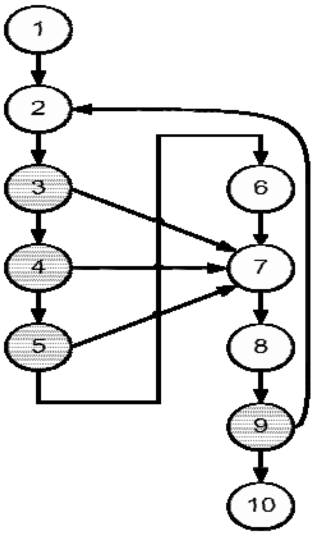


Рис. 3. Граф управления программой расчета значений функции

Для составленного управляющего графа можно выделить два маршрута:

*m*1: 1–2–**3**–**4**–**5**–6–7–8–**9**–2–**3**–7–8–**9**–10; $p_1 = 6$;
*m*2: 1–2–**3**–**4**–7–8–**9**–2–**3**–**4**–**5**–7–8–**9**–10; $p_2 = 7$.

Данные маршруты проходят по всем вершинам ветвления, причем хотя бы один раз по каждой дуге графа, представленного на рис. 3, что соответствует требованиям первого критерия.

В перечне участков маршрутов номера вершин ветвления выделены полужирным шрифтом с подчеркиванием. Таким образом, в соответствии с первым критерием оценки алгоритмической сложности необходимое количество маршрутов равно 2, количество вершин ветвления в маршрутах определяет уровень сложности:

$$S_1 = p_1 + p_2 = 6 + 7 = 13.$$

Второй критерий

Проведем оценку алгоритмической сложности **<u>по второму критерию</u>**. Необходимо определить число проверок каждого линейно независимого цикла и линейно независимого ациклического участка программы. Количество проверок определяется цикломатическим числом графа, которое определяется следующим соотношением: $Z = n_в + 1$, где $n_в$ – число вершин ветвления.

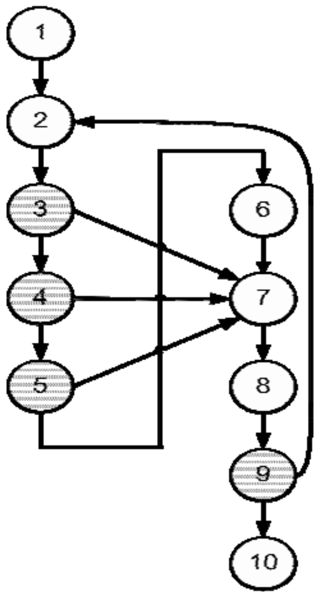


Рис. 3. Граф управления программой расчета значений функции

Число вершин ветвления в составленном графе составляет 4, отсюда

$$Z = n_в + 1 = 4 + 1 = 5.$$

Таким образом, общее число циклических и ациклических участков равно 5. Выделим маршруты на заданном графе:

- ациклические маршруты:

  $m1$: 1–2–**<u>3</u>**–7–8–**<u>9</u>**–10; $p_1 = 2$;
  $m2$: 1–2–**<u>3</u>**–**<u>4</u>**–7–8–**<u>9</u>**–10; $p_2 = 3$;
  $m3$: 1–2–**<u>3</u>**–**<u>4</u>**–**<u>5</u>**–7–8–**<u>9</u>**–10; $p_3 = 4$;
  $m4$: 1–2–**<u>3</u>**–**<u>4</u>**–**<u>5</u>**–6–7–8–**<u>9</u>**–10; $p_4 = 4$;

- циклические маршруты:

  $m5$: 2–**<u>3</u>**–**<u>4</u>**–**<u>5</u>**–6–7–8–**<u>9</u>**; p_5=4.

Тестирование программы по указанным маршрутам позволит проверить все операторы ветвления программы. Метрика структурной сложности определяется по следующему соотношению:

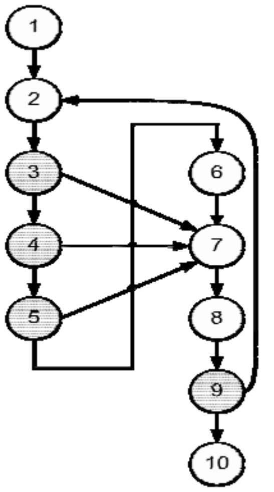
$$S_2 = p_1 + p_2 + p_3 + p_4 + p_5 = 2 + 3 + 4 + 4 + 4 = 17.$$

Матрица смежности

Для организации автоматического анализа заданного графа по второму критерию с помощью вычислительных средств построим матрицы смежности и достижимости.

Напомним, что матрица смежности представляет собой квадратную матрицу, размер которой определяется количеством вершин графа. Полученный граф имеет 10 вершин, следовательно, матрица смежности будет иметь размер 10×10. При заполнении матрицы следует придерживаться следующего правила: для дуги, соединяющей вершину 1 с вершиной 2, в первый столбец и вторую строку матрицы смежности записываем 1 (табл. 1). Номер столбца соответствует номеру вершины, из которой выходит дуга, а номер строки соответствует номеру вершины, в которую входит дуга. Аналогичным образом заполняем матрицу для всех дуг управляющего графа.

Таблица 1. Матрица смежности

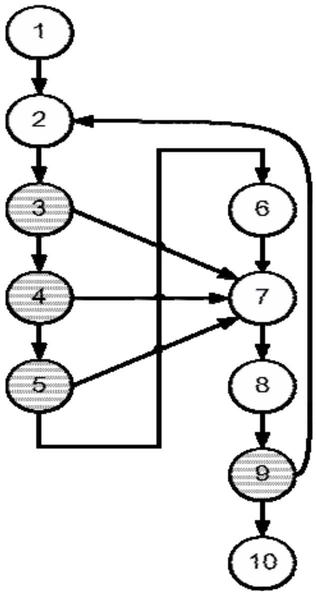


	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1										
2	1								1	
3		1								
4			1							
5				1						
6					1					
7			1	1	1	1				
8							1			
9								1		
10									1	

Матрица достижимости

Для выделения маршрутов можно использовать матрицу достижимости, которая представляет собой для полученного графа управления квадратную таблицу размером 10×10. Номер столбца матрицы определяет номер вершины графа, из которого можно достичь другие вершины этого же графа, используя цикличные и ацикличные маршруты. Номера строк определяют номера достижимых вершин графа управления. Из вершины 1 возможно достичь вершины со второй по десятую, т. е. достижимы все вершины графа (см. рис). Для первого столбца матрицы достижимости строки, начиная со второй, заполняются единицами (табл. 2). Далее аналогично заполняются столбцы для остальных вершин.

Таблица 2. 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	

Выделенные диагональные элементы матрицы определяют номера вершин, которые входят в состав циклических маршрутов. Идентичные строки матрицы определяют номера вершин, которые входят в состав ацикличных маршрутов.

Элементы, выделенные черным цветом, соответствуют маршруту *m*5: 2–**3**–**4**–**5**–6–7–8–**9**. Все строки матрицы получились идентичными, так как все вершины графа входят в состав ацикличных маршрутов.

Третий критерий

Проведем оценку алгоритмической сложности **по третьему критерию**.

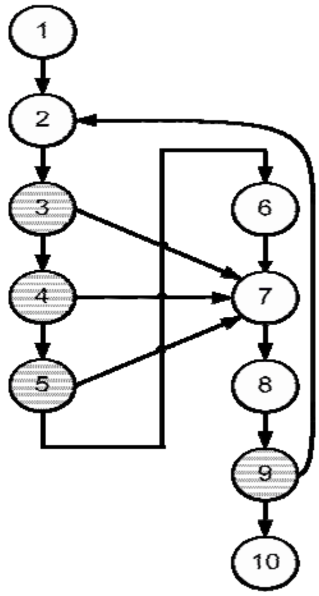


Рис. 3. Граф управления программой расчета значений функции

В соответствии с третьим критерием необходимо выделить все реально возможные маршруты управления:

*m*1: 1–2–**3**–7–8–**9**–10; $p_1 = 2$;
*m*2: 1–2–**3**–**4**–7–8–**9**–10; $p_2 = 3$;
*m*3: 1–2–**3**–**4**–**5**–7–8–**9**–10; $p_3 = 4$;
*m*4: 1–2–**3**–**4**–**5**–6–7–8–**9**–10; $p_4 = 4$;
*m*5: 1–2–**3**–**4**–**5**–6–7–8–**9**–2–**3**–**4**–**5**–6–7–8–**9**–10; $p_5 = 8$;
*m*6: 1–2–**3**–7–8–**9**–2–**3**–**4**–**5**–6–7–8–**9**–10; $p_6 = 6$;
*m*7: 1–2–**3**–**4**–7–8–**9**–2–**3**–**4**–**5**–6–7–8–**9**–10; $p_7 = 7$;
*m*8: 1–2–**3**–**4**–**5**–7–8–**9**–2–**3**–**4**–**5**–6–7–8–**9**–10; $p_8 = 8$.

Оценку структурной сложности программы рассчитываем по следующему соотношению:

$$S_3 = p_1 + p_2 + p_3 + p_4 + p_5 + p_6 + p_7 + p_8 =$$

$$= 2 + 3 + 4 + 4 + 8 + 6 + 7 + 8 = 42.$$

Вывод

Исходя из полученных результатов расчета метрик структурной сложности по первому ($S_1 = 8$), второму ($S_2 = 17$) и третьему ($S_3 = 42$) критериям выделения маршрутов можно сделать вывод, что программа, характеризуемая заданным графом управления, имеет невысокую алгоритмическую сложность, так как количество используемых в тексте операторов условий 5, для проверки которых необходимо проверить от 5 до 42 тестовых вариантов исходных данных.

Оценим алгоритмическую сложность программы **<u>на основе метрики Маккейба</u>**. В соответствии с теорией Маккейба сложность алгоритма оценивается величиной цикломатического числа, которая определяется по следующему соотношению: $Z = m - n + 2$, где m – количество дуг управляющего графа, построенного на основе алгоритма программы, n – количество вершин графа. В соответствии с полученным управляющим графом $m = 13$, $n = 10$, тогда цикломатическое число равно:

$$Z = m - n + 2 = 13 - 10 + 2 = 5.$$

Таким образом, в соответствии со значением цикломатического числа (Z = 5) в полученном графе управления программой можно выделить пять независимых контуров, которые определяют пять управляющих маршрутов, ведущих из начальной вершины в конечную. Значение цикломатического числа для полученного графа (Z = 5) не превышает значения 10, что говорит о незначительной сложности алгоритма решения задачи расчета значений функции.