ОЦЕНКА СТРУКТУРНОЙ СЛОЖНОСТИ ПРОГРАММ

Реализация программы на языке C#

|  |  |
| --- | --- |
| **Строка** | **Текст программы** |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42 | using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace ConsoleApp7  {  class Program  {  static void Main(string[] args)  {  bool flag = false;  Console.WriteLine("Введите А: ");  Int32 a = Convert.ToInt32(Console.ReadLine());  Console.WriteLine("Введите B: ");  Int32 b = Convert.ToInt32(Console.ReadLine());  Console.WriteLine("Введите S: ");  Int32 s = Convert.ToInt32(Console.ReadLine());  Console.WriteLine("Ответ: ");  for (Int32 i = a; i <= b; i++)// 1 3  {  Int32 sum = 0;  string chislo = Convert.ToString(i);  for (Int32 k = 0; k < chislo.Length; k++)  {  sum += Convert.ToInt32(Convert.ToString(chislo[k]));  }  if (sum == s)  {  Console.WriteLine(chislo);  flag = true;  }  }  if (flag != true)  Console.WriteLine("Требуемых чисел нет");  Console.ReadKey();  }  }  } |
|  |  |

**Алгоритм решения задачи**



Рис.1

**Оценка алгоритмической сложности :**

Граф потока управления приведен на рис. 2. Тонированные вершины обозначают операторы ветвления.



Рис.2 Граф управления программой

Проведем оценку алгоритмической сложности графа по первому критерию.

Определим минимальный набор маршрутов, проходящих через каждый оператор ветвления и по каждой дуге.

Для составленного графа можно выделить маршруты:

𝑚1:1 − 2 − 3 − 4 − 5 − 6 −7− 8 − 9 – 10− 11− 9− 12− 13− 14− 5 − 15− 16− 17

𝑝1 = 5.

𝑚2:1 − 2 − 3 − 4 − 5 − 6 −7− 8 − 9 – 10− 11− 9− 12− 14− 5 − 15− 16− 17

𝑝2 = 5.

𝑚3:1 − 2 − 3 − 4 − 5 − 6 −7− 8 − 9 – 10− 11− 9− 12− 13− 14− 5 − 15− 17

𝑝3 = 5.

𝑚4:1 − 2 − 3 − 4 − 5 − 6 −7− 8 − 9 – 10− 11− 9− 12− 14− 5 − 15− 17

𝑝4 = 5.

В перечне участков маршрутов номера вершин ветвления выделены подчеркиванием.

Таким образом, в соответствии с первым критерием оценки алгоритмической сложности необходимое количество маршрутов равно 1, количество вершин ветвления в маршрутах определяет уровень сложности:

𝑆1 = 20.

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, которое определяется следующим соотношением:

𝑍 = 𝑛в + 1,

где 𝑛в – число вершин ветвления.

Число вершин ветвления в составленном графе составляет 3, отсюда

𝑍 = 𝑛в + 1 = 4 + 1 = 5.

Таким образом, общее число циклических и ациклических участков равно 4.

Выделим маршруты на заданном графе:

* ациклические маршруты:

𝑚1:1 − 2 − 3 − 4 − 5 − 6 −7− 8 − 9 – 10− 11− 9− 12− 13− 14− 5 − 15− 16− 17

𝑝1 = 5.

* циклические маршруты:

𝑚2: 5–9–12–14; 𝑝2 = 3;

𝑚3: 9 – 10 – 11; 𝑝3 = 3;

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

𝑆2 = 𝑝1+ 𝑝2+ 𝑝3= 5+3+3=12.

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

Полученный граф имеет 11 вершины, следовательно, матрица смежности будет иметь размер 17x17.

Табл.1

Матрица смежности

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** | **13** | **14** | **15** | **16** | **17** |
| **1** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **2** |  |  | 1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **3** |  |  |  | 1 |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **4** |  |  |  |  | 1 |  |  |  |  |  |  |  |  |  |  |  |  |
| **5** |  |  |  |  |  | 1 |  |  |  |  |  |  |  |  |  |  |  |
| **6** |  |  |  |  |  |  | 1 |  |  |  |  |  |  |  |  |  |  |
| **7** |  |  |  |  |  |  |  | 1 |  |  |  |  |  |  |  |  |  |
| **8** |  |  |  |  |  |  |  |  | 1 |  |  |  |  |  |  |  |  |
| **9** |  |  |  |  |  |  |  |  |  | 1 |  | 1 |  |  |  |  |  |
| **10** |  |  |  |  |  |  |  |  |  |  | 1 |  |  |  |  |  |  |
| **11** |  |  |  |  |  |  |  |  | 1 |  |  |  |  |  |  |  |  |
| **12** |  |  |  |  |  |  |  |  |  |  |  |  | 1 | 1 |  |  |  |
| **13** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **14** |  |  |  |  | 1 |  |  |  |  |  |  |  |  |  |  |  |  |
| **15** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 1 |  |
| **16** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **17** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

Табл.2

Матрица достижимости

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** | **13** | **14** | **15** | **16** | **17** |
| **1** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **2** | 1 | *1* | 1 |  | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |  |
| **3** | 1 | 1 | *1* | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |  |
| **4** | 1 | 1 | 1 | *1* | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |  |
| **5** | 1 | 1 | 1 | 1 | *1* | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |  |
| **6** | 1 | 1 | 1 | 1 | 1 | *1* | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |  |
| **7** | 1 | 1 | 1 | 1 | 1 | 1 | *1* | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |  |
| **8** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | *1* | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |  |
| **9** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | *1* | 1 | 1 | 1 | 1 | 1 | 1 | 1 |  |
| **10** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | *1* | 1 | 1 | 1 | 1 | 1 | 1 |  |
| **11** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | *1* | 1 | 1 | 1 | 1 | 1 |  |
| **12** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | *1* | 1 | 1 | 1 | 1 |  |
| **13** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | *1* | 1 | 1 | 1 |  |
| **14** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | *1* | 1 | 1 |  |
| **15** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | *1* | 1 |  |
| **16** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | *1* |  |
| **17** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |  |

Проведем оценку алгоритмической сложности по третьему критерию. В соответствии с третьим критерием необходимо выделить все реально возможные маршруты управления:

𝑚1:1 − 2 − 3 − 4 − 5 − 6 −7− 8 − 9 – 10− 11− 9− 12− 13− 14− 5 − 15− 16− 17

𝑝1 = 5.

Оценку структурной сложности программы рассчитываем по следующему соотношению:

𝑆3 = 𝑝1=5;

Исходя из полученных результатов расчета метрик структурной сложности по первому (𝑆 1 = 20), второму (𝑆2 = 12) и третьему (𝑆3 = 5) критериям выделения маршрутов можно сделать вывод, что программа, характеризуемая заданным графом управления, имеет невысокую алгоритмическую сложность.

Оценим алгоритмическую сложность программы на основе метрики Маккейба.

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 = 20, 𝑛 = 17, тогда цикломатическое число равно: 𝑍 = 𝑚 − 𝑛 + 2 = 20 − 17 + 2 = 4.

Таким образом, в соответствии со значением цикломатического числа (𝑍 = 4) 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 = 4) не превышает значения 10, что говорит о незначительной сложности алгоритма решения задачи расчета значений функции.