## Тема 4.6

## Программирование алгоритмов итеративных циклических

## структур

### 4.6.1. Средства программирования

### итеративных циклических структур

 Напомним, что ***циклическая структура***, в которой число повторений цикла заранее неизвестно, а определяется только в процессе выполнения алгоритма, называется ***итеративной циклической структурой****.*

В зависимости от места расположения условия продолжения цикла (или выхода из цикла) ***итеративные циклические структуры*** подразделяются на два вида: ***с предусловием*** и ***с постусловием*** (рис. В.3-10 и рис. В.3-11).

При организации цикла с предусловием блоки тела цикла, следующие за блоком, в котором проверяется условие выхода из цикла, выполняются всякий раз, когда логическое выражение **L** принимает значение **True**. При первом невыполнении этого условия происходит выход из цикла. Таким образом, возможен случай, когда тело цикла не будет выполнено ни разу. Поэтому циклические структуры с известным числом повторений (регулярные циклы) относятся к числу циклических алгоритмов с предусловием.

При организации ***циклов с постусловием***, для которых условие выхода из цикла (или повторения тела цикла) проверяется после выполнения цикла, цикл всегда выполняется хотя бы один раз, независимо от значения L, и только после его выполнения принимается решение – продолжать выполнение цикла или выйти из него.

Необходимо отметить, что логическое выражение (условие) **L** имеет тот же смысл, что и в **Теме 4**.**4**.

При вычислении значений функций или каких-либо числовых последовательностей (например, арифметической прогрессии), их часто записывают в виде специальных сумм, называемых рядами. Многие числа, функции, алгоритмы численных методов могут быть записаны с помощью рядов или итерационных алгоритмов, которые позволяют вычислять их приближенные значения с заданной точностью. При программировании таких задач используются итерационные методы для организации цикла, в котором производится вычисление некоторой рекуррентной формулы.

***Рекуррентная формула*** – это такая формула, которая сводит вычисления **n**-го члена последовательности к вычислению нескольких предыдущих членов (или, часто, одного предыдущего члена этой последовательности – **n-1**). В общем случае такая формула имеет вид:   
**Sn= Sn-1 + Un**, где **Sn** – сумма первых **n** слагаемых ряда, которая образуется из суммы, полученной на предыдущем шаге **Sn-1**и слагаемого **Un**,полученного на текущем шаге.

Многие числа и функции можно записать с помощью рядов, например, для числа ряд можно записать двумя способами:

** + +  + . . .**

**=** 1 **− + − + . . .**

Второй ряд сходится более медленно, т.е. требует большего числа слагаемых для достижения той же точности.

Конструкции, реализующие в VBитеративные циклы с предусловием, могут иметь следующие форматы:

|  |  |
| --- | --- |
| **Do While** **L**  *Операторы Тела Цикла*  [ **Exit Do** ]  **Loop** | **Do Until** **L**  *Операторы Тела Цикла*  [ **Exit Do** ]  **Loop** |

А итеративные циклы с постусловием, имеют следующие форматы:

|  |  |
| --- | --- |
| **Do**  *Операторы Тела Цикла*  [ **Exit Do** ]  **Loop While L** | **Do**  *Операторы Тела Цикла*  [ **Exit Do** ]  **Loop Until  L** |

Если используется ключевое слово **While**, то цикл повторяется до тех пор, пока условие **L** истинно, а если **Until** – то пока ложно. Таким образом, после ключевого слова **While** записывается условие продолжения цикла, а после **Until** – условие завершения цикла (выхода из цикла).

Другими словами, циклы **Do…Loop** различаются местом расположения условия, которое помещается либо в заголовке цикла (в первой строке **Do…**), либо в конце цикла (в последней строке **Loop…**).

Для досрочного выхода из цикла используется оператор **Exit Do.**

**Пример 4.6.1-1. Написать процедуру-Function, которая вычисляет произведение натуральных чисел от 1 до n (факториал n!=1\*2\*3\*4\*…n).**

В принципе такая задача решается с помощью регулярной алгоритмической структуры (**Пример 4.5.2-5**). Здесь этот пример представлен в учебных целях.

|  |
| --- |
| **Function Pr611(ByVal n As Integer) As Long**  **Dim I As Integer, p As Long**  **p = 1 : I = 0**  **Do While I < n**  **I = I + 1 : p = p \* I**  **Loop**  **Return p**  **End Function** |

*Рис. 4.6.1-1. Программный код процедуры* **Pr611(),***которая вычисляет факториал* **n Примера 6.1-1**

Процедура-**Futncion Pr611(),** использующая итеративную циклическую структуру с предусловием, представлена на рис. 4.6.1-1.

Цикл **Do…Loop** в представленной программе будет выполняться до тех пор, пока значение переменной **I** не достигнет значения **n**. Здесь логическое условие продолжения цикла (**I<n)**, помещенное после ключевых слов **Do While**, позволяет проверять его перед началом выполнения первого оператора тела цикла. Если условие продолжения цикла для начальных условий истинно (т.е. равно **True**)**,** то все операторы цикла будут выполнены. Потом условие **I<n** будет проверено снова, и так до тех пор, пока условие не станет ложным – **False**. После этого выполнение операторов в теле цикла будет прекращено, и программа переходит к выполнению оператора, следующего за ключевым словом **Loop.**

Если, например, по условиям задачи необходимо, чтобы тело цикла выполнилось, по крайней мере, один раз, то в этом случае нужно использовать вторую форму оператора **Do…Loop**, в которой условие помещается в конце цикла, после ключевых слов **Loop While…** или **Loop Until….**

Процедура-**Function,** реализующая такую структуры цикла, рассмотрена в Примере 4.6.1-2.

**Пример 4.6.1-2. Написать процедуру-Function, которая вводит натуральное число n, значение которого находится на отрезке [1; 15], с проверкой ввода (т.е. должны выполняться условия n>=1 AND n<=15).**

Самой удобной структурой для такой проверки будет структура цикла с постусловием **Do…Loop Until**. Такой цикл будет повторяться до тех пор, пока это условие равно **False**. Как только условие станет равно **True**, т.е. число попадет в границы диапазона, то цикл прекращается.

Процедура-**Function** **Pr612( )** представлена на рис. 4.6.1-2.

|  |  |
| --- | --- |
|  | **Function Pr612( ) As Integer**  **Dim n As Integer**  **Do**  **n = vvodInt9("Введите n (1<n<15)", TextBox1)**  **Loop Until n >=1 And n <= 15**  **Return n**  **End Function** |

*Рис. 4.6.1-2. Программный код процедуры* **Pr612(),***которая проверяет введенное значение*

**Примера 4.6.1-2**

**Пример 4.6.1-3. Написать процедуры вывода, которые могут использоваться в алгоритмах итеративных циклических структур.**

Некоторые процедуры ввода и вывода приведены в **Теме 4**. Остальные процедуры ввода и вывода, которые используются при написании базовых алгоритмов итеративных циклических структур, представлены на   
рис. 4.6.1-3 – 4.6.1-4.6.

|  |
| --- |
| *'Процедура вывода значений переменных Integer и Double в TextBox*  **Sub vivodID11(ByVal n As Integer,ByVal T1 As String \_**  **ByVal a As Double, ByVal T2 As String, \_**  **ByVal T As TextBox)**  **T.Text = T.Text & T1 & CStr(n) & T2 & CStr(a) & vbCrLf**  **End Sub** |

*Рис. 4.6.1-3. Программный код процедуры* **vivodID11()**

*вывода значений двух переменных типа* **Integer** *и* **Double** *в* **TextBox**

**Примера 4.6.1-3**

|  |
| --- |
| *'Процедура вывода целого результата в ListBox*  **Sub vivodIntLs12(ByVal n As Integer, ByRef LB As ListBox)**  **LB.Items.Add(CStr(n))**  **End Sub** |

*Рис. 4.6.1-4. Программный код процедуры* **vivodIntLs12()**

*вывода целого результата в* **ListBox**

**Примера 4.6.1-3**

|  |
| --- |
| *'Процедура вывода вещественного результата в ListBox*  **Sub vivodDblLs13(ByVal Z As Double, ByRef LB As ListBox)**  **LB.Items.Add(CStr(Z))**  **End Sub** |

*Рис. 4.6.1-5. Программный код процедуры* **vivodDblLs13()**

*вывода вещественного результата в* **ListBox**

**Примера 4.6.1-3**

|  |
| --- |
| *'Процедура форматного вывода значений переменных* ***Integer в Т****extBox*  **Sub vivodFxn14(ByVal x As Long, ByVal n As Integer, \_  ByVal T As TextBox)**  **T.Text = T.Text & Format(x, "0") & Space(8) & \_**  **Format(n, "0") & vbCrLf**  **End Sub** |

*Рис. 4.6.1-6. Программный код процедуры* **vodFxn14()**

*форматного вывода значений двух переменных типа* **Integer***в***ТextBox**

**Примера 4.6.1-3**

### 4.6.2. Базовые алгоритмы итеративных циклических структур и примеры их программирования

Рассмотрим сумму вида **S= a1 + a2 + a3 +** . . . пусть имеется положительное число **ε > 0**, называемое точностью вычислений. Во многих задачах точность вычисления суммы **S** считается достигнутой, когда выполняется одно из следующих условий:

* разность │Si – Si-1│ < ε, где Si – сумма на i–м шаге цикла (т.е. сумма содержит i слагаемых), а Si-1 – сумма на предыдущем шаге;
* разность │ai – ai-1│< ε, где ai – значение слагаемого на i-м шаге, а   
  ai-1 – на предыдущем шаге цикла;
* значения │Si│ < ε и значение │ai│ < ε.

Абсолютное значение (модуль) условия обычно требуется для знакопеременных рядов. Следует заметить также, что приведенные выше условия во многих задачах практически эквивалентны.

Кроме вычислений, связанных с рядами, существует много задач, в которых количество повторений цикла определяется в процессе его выполнения. Как уже известно, подобные циклы называют ***итеративными***.

При описании базовых алгоритмов регулярных циклических структур мы приводили примеры вычисления сумм и произведений конечного (заранее известного) числа членов последовательности (Примеры 4.5.2-3, 4.5.2-4, 4.5.2-5).

Рассмотрим теперь некоторую последовательность, содержащую бесконечное число членов: **a0, a1, a2, a3,…, ai,…, an,…** В таких задачах требуется вычислять члены последовательности до тех пор, пока очередной вычисленный член не будет удовлетворять некоторому условию.

Если задано некоторое число **ε,** условия окончания итерационного процесса могут, например, быть следующими:

* для убывающей последовательности **an < ε**;
* для возрастающей последовательности **an > ε (**примеры 4.6.2-1, 4.6.2-3,   
  4.6.2-5**)**;
* для убывающей знакопеременной последовательности **|an| < ε   
  (**пример 4.6.2-4);
* для некоторых других последовательностей |**an+1 -an|<ε** (пример 4.6.2-2).

**Пример 4.6.2-1**. **Написать процедуру-Function, которая среди последовательности чисел (факториал n! =1\*2\*3\*4\* . . . n) находит первое число, большее заданного значения переменной a.**

Алгоритм решения данной задачи относится к алгоритмам вычисления членов бесконечных последовательностей (рис.4.6.2-1).

Этот алгоритм использует итеративный цикл с предусловием и реализуется с помощью конструкции **Do While ... Loop.**

Для решения данной задачи проведем ***формализацию***. Для этого введем следующие обозначения: **b** – очередной член бесконечной последовательности; **n** – номер этого члена, который в данной задаче совпадает со значением знаменателя дроби, добавляемой к предыдущему члену для получения значения очередного члена последовательности. Таким образом, итерационная формула вычисления очередного члена последовательности будет иметь следующий вид: **(т.е. должны выполняться условия n>=1 AND n<=15)**, где **n** – номер члена.

|  |  |
| --- | --- |
|  | **Function Pr621(ByVal a As Double)As Double**  **Dim b As Double**  **Dim n As Integer**  **b = 1**  **n = 1**  **Do  While b <= a    n = n + 1  b = b + 1 / n Loop Return b**  **End Function** |

*Рис. 4.6.2-1.Схема алгоритма и программный код процедуры* **Pr621(),**которая среди последовательности чисел находит первое число,   
большее заданного значения, используя оператор **Do While...Loop**

**Примера 4.6.2-1**

Процедура-функция **Pr621()**может быть вызвана из любой другой процедуры или из модуля формы, например, как показано на рис. 4.6.2-2.

|  |
| --- |
| **Dim aa, bb As Double**  **aa = vvodDbl2("Введите значение a=", TextBox1 )**  **bb = Pr621(aa)**  **vivodDbl1(bb,TextBox2)** |

*Рис. 4.6.2-2. Пример обращения к процедуре* **Pr621()**

Решение данного примера может быть реализовано также с использованием конструкции **Do Until…Loop** (рис. 4.6.2-3), а цикл с предусловием можно заменить на цикл с постусловием и соответствующим ему изменением настройки цикла (рис. 4.6.2-4).

|  |  |
| --- | --- |
|  | **Function Pr623(ByVal a As Double) As Double**  **Dim b As Double**  **Dim n As Integer**  **b = 1**  **n = 1**  **Do  Until b > a    n = n + 1  b = b + 1 / n  Loop  Return b**  **End Function** |

*Рис. 4.6.2-3. Схема алгоритма и программный код процедуры* **Pr621(),**которая среди последовательности чисел находит первое число,   
большее заданного значения, и, используя оператор **Do Until…Loop**

**Примера 4.6.2-1**

Изменим в цикле условие продолжения выполнения цикла   
(на рис. 4.6.2-1 заменим условие **b <= a** на условие **b > a**), т.е. будем проводить вычисления членов последовательности до тех пор, пока не встретится член, больший заданного числа. Заменим условие **b <= a** на условие **b > a.**

Тогда алгоритм и функция будут выглядеть, как на рис.4.6.2-3.

Если вычисление членов последовательности проводится в теле цикла, начиная с первого, то алгоритм и процедура-функция примут вид, показанный на рис 4.6.2-4.

|  |  |
| --- | --- |
|  | **Function P624(ByVal a As Double) \_**  **As Double**  **Dim b As Double**  **Dim n As Integer**  **b = 0**  **n = 0**  **Do**  **n = n + 1  b = b + 1 / n Loop Until b > a Return b**  **End Function** |

*Рис. 4.6.2-4. Схема алгоритма и программный код процедуры* **Pr624(),**которая среди последовательности чисел находит первое число,   
большее заданного значения, используя, оператор **Do…Loop Until**

**Примера 4.6.2-1**

Мы получили структуру итеративного цикла с постусловием, изменив настройку цикла: **n = 0**, **b = 0**. В данном случае условием окончания вычислительного процесса служит значение **True** логического выражения   
b **> a**, а продолжением – значение **False** (конструкция **Do…Loop Until**).

Изменив условие на **b <= a,** получим конструкцию цикла с   
**Do…Loop Wile.**

Следовательно, задача может быть решена с использованием как цикла с предусловием, так и цикла с постусловием.

**Пример 4.6.1-2. Написать процедуру-Function, которая вводит натуральное число n, значение которого находится на отрезке [1;15], с проверкой ввода (т.е. должны выполняться условия n>=1 AND n<=15).**

Вос­пользуемся известной итерационной формулой, где **i=0, 1, 2,...; x0=0.** Следует закончить итеративный процесс, как только **|xi+1-xi|** станет меньше **ε=10-4**.

Для решения этой задачи необходимо из очередного приближения вычисленного корня **xi+1** вычитать значение предыдущего приближения корня **xi**. Для этого при каждом повторении цикла перед вычислением очередного значения корня **x** сохраняем в переменной **a** текущее значение **x** (оно становится предыдущим). Цикл прекращаем, если разность между **a** (т.е. **xi**) и **x** (т.е. **xi+1**) станет меньше **ε=10-4**.

Алгоритм решения данной задачи относится к итерационным алгоритмам (рис. 4.6.2-5) и может быть реализован, например, с помощью конструкции **Do ... Loop Until** c постусловием.

|  |  |
| --- | --- |
|  | **Function P625( ) As Double**  **Dim a, x, d As Double**  **x = 0**  **d = 1E-4**  **Do**  **a = x**  **x = -Exp(a)**  **Loop Until Abs(x - a) < d**  **Return x**  **End Function** |

*Рис. 4.6.2-5. Схема алгоритма и программный код процедуры* **Pr625(),**

*которая реализует ввод данных с их проверкой*

**Примера 4.6.2-2**

Процедура-**Function Pr625()**может быть вызвана из любой другой процедуры или из модуля формы, например, как показано на рис. 4.6.2-4.6.

|  |
| --- |
| **Dim xx As Double**  **xx = Pr625()**  **vivodDbl1(xx, TextBox1)** |

*Рис. 4.6.2-6. Пример обращения к процедуре* **Pr625()**

**Примера 4.6.2-2**

**Пример 4.6.2-3. Задана возрастающая последовательность **

**Требуется написать программу, которая вычисляет все члены последовательности, до тех пор, пока значение очередного члена не превысит некоторое заданное число d, например, (3 <d <100).**

В нашей задаче для вычисления любого члена последовательности можно воспользоваться формулой, , где **n=0, 1, 2,…-** номер члена.

В задачах, использующих итеративные алгоритмические структуры, рекомендуется предусмотреть так называемую «страховку от зацикливания», так как иногда условие продолжения цикла может оставаться истинным бесконечно. В данном примере цикл с постусловием будет выполняться не более **100** раз, даже если очередной член последовательности будет оставаться меньше **d** (рис. 4.6.2-7). Так как вывод в **TextBox**членов последовательности должен происходить в процессе вычисления (внутри цикла), то для решения задачи напишем процедуру-**Sub**.

|  |  |
| --- | --- |
|  | **Sub Pr627(ByVal x As Double, \_**  **ByVal d As Double)**  **Dim n As Integer = 0**  **Dim a As Double**  **Do**  **а = x^n / 3^n**  **vivodID11(n, "n=",a, "a=", TextBox3)**  **n=n+1**  **Loop While a <= d And n < 100**  **End Sub** |

*Рис. 4.6.2-7. Схема алгоритма и программный код процедуры* **Pr627(),**

*которая вычисляет члены последовательности   
до выполнения определенных условий*

**Примера 4.6.2-3**

Процедура-**Sub Pr627( )** может быть вызвана из любой другой процедуры или из модуля формы, например, как показано на рис. 4.6.2-8.

|  |
| --- |
| **Dim xx, dd As Double**  **xx = vvodDbl2("Введите значение xx= ", TextBox1)**  **dd = vvodDbl2("Введите значение dd= ", TextBox2)**  **Pr627(xx, dd)** |

*Рис. 4.6.2-8. Пример обращения к процедуре* **Pr627()**

**Примера 4.6.2-3**

**Пример 4.6.2-4. Написать процедуру-функцию, которая вычисляет сумму членов знакопеременной убывающей последовательности с заданной точностью ε:**  **.**

Вычисление с заданной точностью ε означает, что суммирование членов ряда надо продолжать до тех пор, пока очередной вычисленный член ряда не станет меньше по абсолютной величине числа ε.

Отметим, что во многих задачах непосредственный подсчет очередного члена связан с вычислительными трудностями. В этом случае целесообразно использовать рекуррентную формулу, которая позволяет вычислить значение переменной на следующем шаге, используя ее значение на текущем шаге **–** . Выражение для q можно получить, разделив an+1 член на an член.

|  |  |
| --- | --- |
|  | **Function Pr629(ByVal x As Double,\_**  **ByVal e As Double) As Double**  **Dim a, s As Double**  **Dim n As Integer = 0**  **a = x – 1**  **s = 0**  **Do Until Abs(a) < e Or n > 100**  **vivodIntLs12(n, ListBox1)**  **vivodDblLs13(a, ListBox2)**  **s = s + a**  **a = -a \* (x - 1) / (n + 2)**  **n = n + 1**  **Loop**  **Return s**  **End Function** |

*Рис. 4.6.2-9. Схема алгоритма и программный код процедуры* **Pr629(),**

которая вычисляет сумму членов знакопеременной убывающей последовательности с точностью ε

**Примера 4.6.2-4**

Приведем вывод рекуррентной формулы для заданного в примере ряда. Формула для n-го члена приведена в задании:

Приведем вывод рекуррентной формулы для заданного в примере ряда. Формула для n-го члена приведена в задании:

 тогда формула n+1 члена



Разделив an+1 член на an, получим выражение для q



Таким образом, рекуррентная формула для данного ряда:



Выбор начального значения номера члена ряда (n) для нашего случая будет n=0, так как при подстановке этого значения в формулу n-го члена ряда



мы получим значение первого члена, равного x-1 или a0=x-1.

Схема алгоритма и процедура-**Function** приведены на рис. 4.6.2-9, причем алгоритм этот с предусловием, в котором предусмотрено выполнение цикла не более **100** раз, чтобы избежать зацикливания.

**Процедура-Function Pr629( )** может быть вызвана из любой другой процедуры или из модуля формы, например, как показано на рис. 4.6.2-10.

|  |
| --- |
| **Dim xx, ee, ss As Double**  **xx=vvodDbl2("Введите значение xx=", TextBox1)**  **ee=vvodDbl2("Введите значение ee=", TextBox2)**  **ss = Pr629(xx, ee)**  **vivodDbl1(ss, TextBox3)** |

*Рис. 4.6.2-10. Пример обращения к процедуре* **Pr629()**

**Примера 4.6.2-4**

Приведем вывод рекуррентной формулы для заданного в примере ряда. Формула для n-го члена приведена в задании:

 тогда формула n+1 члена



Разделив an+1 член на an, получим выражение для q



Таким образом, рекуррентная формула для данного ряда:



Выбор начального значения номера члена ряда (n) для нашего случая будет n=0, так как при подстановке этого значения в формулу n-го члена ряда



мы получим значение первого члена, равного x-1 или a0=x-1.

|  |  |
| --- | --- |
|  | **Function P6211(ByVal x Double, \_**  **ByVal e As Double) As Double**  **Dim a, s As Double**  **Dim n As Integer = 0**  **a = x – 1**  **s = 0**  **Do Until Abs(a) < e Or n > 100**  **vivodIntLs12(n,ListBox1)**  **vivodDblLs13(a,ListBox2)**  **s = s + a**  **n = n + 1**  **a =- a \* (x - 1) / (n + 1)**  **Loop**  **Return s**  **End Function** |

*Рис. 4.6.2-11. Схема алгоритма и программный код процедуры* **Pr6211(),**

которая вычисляет сумму членов знакопеременной убывающей последовательности с точностью ε

**Примера 4.6.2-4**

Схема алгоритма и код процедуры-функции приведены на   
рис. 4.6.2-11. В этом случае, в отличие от предыдущего, увеличивать n (n=n+1) следует до, а не после вычисления очередного члена ряда.

**Пример 4.6.2-5. Написать процедуру-Sub, вычисляющую сумму всех чисел Фибоначчи, которые не превосходят заданного натурального числа m, и определить количество таких чисел.**

Числа Фибоначчи (**Fi**) определяются по формулам

**F0 = F1 = 1; Fi = Fi –1 + Fi –2,**при **i = 2, 3,...,…**

т.е. каждое очередное число равно сумме двух предыдущих.

Подобная задача была решена при рассмотрении регулярных циклических структур (пример 5.2-4). Однако здесь число слагаемых (членов последовательности) заранее неизвестно, поэтому организуем итеративный цикл с предусловием. Обозначим это число слагаемых **k**. В данном примере до начала цикла задаются значения трех чисел Фибоначчи **F0, F1, и F2 = F0 + F1 = 2** и их сумма **S = F0 + F1 + F2 = 4**. В цикле переопределяем значение переменных F**0** и **F1**, выводим очередное (предыдущее) число Фибоначчи, вычисляем следующее число и добавляем его в сумму. Когда текущее число Фибоначчи превосходит заданное число **m** (**F2 ≤ m** – **False**), то происходит выход из цикла. Так как это число Фибоначчи уже добавлено в сумму, то после выхода из цикла его необходимо вычесть из нее (**S = S - F2**). Таким образом, в окончательном значении суммы будут учтены только те числа Фибоначчи, которые еще не превосходят m.

|  |  |
| --- | --- |
|  | **Sub P6212(ByVal m As Integer, \_**  **ByRef S As Long, ByRef k As Integer)**  **Dim F, F1, F2 As Integer**  **F0 = 1 : F1 = 1 : F2 = 2**  **S = F0 + F1 + F2 : k = 3**  **Do While F2 <= m**  **F0 = F1 : F1 = F2**  **vivodIntLs12(F1, ListBox1)**  **F2 = F0 + F1**  **S = S + F2**  **K = k + 1**  **Loop**  **S = S – F2**  **k = k - 1**  **End Sub** |

*Рис. 4.6.2-12. Схема алгоритма и программный код процедуры* **Pr6212(),**

которая вычисляет *сумму чисел Фибоначчи,   
не превосходящих заданного натурального числа* **m**

**Примера 4.6.2-5**

Схема алгоритма и код процедуры-**Sub** приведены на рис. 4.6.2-12.

***Процедура-*Sub Pr6212()** может быть вызвана из любой другой процедуры или из модуля формы, например, как показано на рис. 4.6.2-13.

|  |
| --- |
| **Dim SS As Long**  **Dim mm, kk As Integer**  **mm=vvodInt9("Введите значение mm= ", TextBox1)**  **Pr6212(mm, SS, kk)**  **vivodFxn14(SS, kk, TextBox2)** |

*Рис. 4.6.2-13. Пример обращения к процедуре* **Pr6212()**

**Примера 4.6.2-5**

Пример 4.6.2-6. **Написать процедуру-Function, которая вычисляет с точностью ε=0.001 сумму членов ряда**

.

Задана знакопеременная убывающая последовательность, поэтому для вычисления суммы ряда с точностью **ε** будем вычислять и суммировать члены последовательности до тех пор, пока очередной член ряда не станет меньше заданной точности по модулю.

Заданный ряд вычислять через рекуррентную формулу нецелесообразно, так как это, во-первых, трудно, а во-вторых, скорее всего, невозможно вывести общую формулу **n**-гол члена ряда. Поэтому можно воспользоваться следующим универсальным приемом вычисления подобных рядов. Каждый член последовательности можно представить, как  где **z**– множитель, равный 1 или -1 (знак члена ряда); **n** – первый сомножитель в числителе каждого слагаемого (в данном случае совпадает с номером члена ряда); **b** – второй сомножитель в числителе; **с** – знаменатель каждого слагаемого.

В свою очередь, знаменатель члена ряда представляет собой произведение нескольких сомножителей, причем **с** увеличением номера члена ряда увеличивается и количество сомножителей в знаменателе. Таким образом, знаменатель можно представить, как **с = с \* d**, где **d** – сомножитель, добавленный к значению знаменателя предыдущего члена ряда.

Схема алгоритма и код процедуры-функции приведены на   
рис. 4.6.2-14.

|  |  |
| --- | --- |
|  | **Function Pr6214(ByVal E As Double) \_**  **As Double**  **Dim a, s As Double**  **Dim n, z, b, c, d As Integer**  **n = 1**  **z = 1**  **b = 5**  **c = 2 \* 4**  **d = 4**  **a = z \* n^ 2 \* b / c**  **s = 0**  **Do While Abs(a) > E**  **vivodIntLs12(n, ListBox1)**  **vivodDblLs13(a, ListBox2)**  **s = s + a**  **n = n + 1**  **z = -z**  **b = b + 1**  **d = d + 2**  **c = c \* d**  **a = z \* n^ 2 \* b / c**  **Loop**  **Return s**  **End Function** |

*Рис. 4.6.2-14. Схема алгоритма и программный код процедуры* **Pr6214(),**

которая вычисляет с точностью ε=0.001 сумму членов заданного ряда

**Примера 4.6.2-6**

**Процедура-Function Pr6214( )** может быть вызвана из любой другой процедуры или из модуля формы, например, как показано на рис. 4.6.2-15.

|  |
| --- |
| **Dim EE, SS As Double**  **EE =vvodDbl1(TextBox1)**  **SS = Pr6214(EE)**  **vivodDbl1(SS, TextBox2)** |

*Рис. 4.6.2-15. Пример обращения к процедуре* **Pr6214()**

**Примера 4.6.2-6**

### 4.6.3. Тестовые задания

1. **Оператор Dо…Lооp – это:**
2. оператор итеративного цикла;
3. оператор выбора
4. оператор регулярного цикла
5. составной оператор.
6. **В итеративной циклической структуре число повторений операторов тела цикла**
7. может быть известно заранее
8. заранее неизвестно
9. заранее известно или может быть предварительно вычислено
10. нет верного ответа
11. **Телом цикла в операторе Do…Loop могут быть**
12. только оператор условного перехода или оператор присваивания
13. только арифметические или логические выражения
14. любые операторы
15. нет верного ответа
16. **Для досрочного прекращения итеративного цикла используется оператор**
17. **Exit Do**
18. **Exit**
19. **Break**
20. нет верного ответа
21. **Алгоритмическая структура цикла итеративного типа может быть**
22. с предусловием или с постусловием
23. только с предусловием
24. только с постусловием
25. безусловная
26. **Если при программировании циклической структуры используется оператор**

**Do while…Loop, то тело цикла**

1. обязательно выполнится хотя бы 1 раз
2. выполняется заданное число раз
3. оператор не относится к средствам программирования итеративного цикла
4. может ни разу не выполниться
5. **Если при программировании циклической структуры используется оператор   
   Do…Loop While, то тело цикла**
6. обязательно выполнится хотя бы 1 раз
7. может ни разу не выполниться
8. выполняется заданное число раз
9. оператор не относится к средствам программирования итеративного цикла
10. **Если при программировании циклической структуры используется оператор**

**Do…Loop Until, то тело цикла**

1. обязательно выполнится хотя бы 1 раз
2. может ни разу не выполниться
3. выполняется заданное число раз
4. оператор не относится к средствам программирования итеративного цикла
5. **После ключевых слов While или Until в операторе итеративного цикла Do…Loop**

**записывается**

1. арифметическое или логическое выражение
2. оператор выбора
3. любой оператор
4. нет верного ответа
5. **Результатом работы фрагмента программы**

|  |
| --- |
| **Dim n As Integer**  **n = 0**  **Do While n < 5**  **n = n + 1**  **Loop**  **TextBox1.Text = CStr(n)** |

будет

1. вывод на экран 6
2. вывод на экран 0
3. вывод на экран 5
4. сообщения об ошибке
5. «зацикливание»
6. **Что будет на экране в результате работы фрагмента программы**

|  |
| --- |
| **Dim n As Integer**  **n = 0**  **Do Until n < 5**  **n = n + 1**  **Loop**  **TextBox1.Text = CStr(n)** |

1. 6
2. 5
3. сообщение об ошибке
4. «зацикливание»
5. 0
6. **Что будет выведено на форму после выполнения заданного фрагмента программы**

|  |
| --- |
| **i = 1**  **DO**  **i = i + 2**  **TextBox1.Text = CStr(i) & vbCrLf**  **LOOP WHILE i < 7** |

1. столбик чисел от 1 до 7
2. строка чисел от 1 до 7
3. столбик чисел от 3 до 7
4. строка чисел от 3 до 7
5. **При каких начальных значениях переменных алгоритм закон­чит работу без зацикливания**

|  |
| --- |
|  |

1. А=-2 С=-3
2. А=-3 С=-2
3. А=-3 С=-3
4. А=-2 С=-1
5. А=-4 С=-3
6. **Цикл с предусловием выполняется следующим образом**
7. выполняется тело цикла, изменяется параметр цикла, проверяется условие продолжения цикла
8. изменяется параметр цикла, проверяется условие продолжения выполнения цикла,
9. выполняется тело цикла
10. проверяется условие продолжения выполнения цикла, выполняется тело цикла
11. тело цикла выполняется n раз (n – натуральное)
12. определяется, сколько раз должен выполнен цикл, и далее цикл с предусловием
13. сводится к циклу с параметром
14. **При каких начальных значе­ниях переменных алгоритм закончит работу без зацикливания**



1. А=-2 С=-1
2. А=-2 С=-3
3. А=-3 С=3
4. А=-3 С=-2
5. **Определить выходные значения переменных А и С после выполнения алгоритма**

|  |
| --- |
|  |

1. 1 7
2. 0 -4
3. 1 3
4. 0 -5
5. зацикливание

## 4.6.4. Лабораторная работа по теме «Программирование алгоритмов итеративных циклических структур»

**Цель** данной лабораторной работы состоит в получении практических навыков формализации, разработки, написания и отладки проектов, использующих итеративные циклические структуры.

### Вопросы, подлежащие изучению

1. Алгоритмы организации итеративных циклических структур: цикл с предусловием; цикл с постусловием.
2. Базовые алгоритмы, использующие итеративные циклические структуры: алгоритм вычисления суммы (или произведения) членов бесконечной последовательности; алгоритмы вычислений по итеративным формулам.
3. Операторы, реализующие выполнение итеративных циклов: **Do While…Loop; Do Until…Loop; Do...Loop While…; Do…Loop…Until.**

### Общее задание на разработку проекта

1. ***Изучите вопросы программирование алгоритмов итеративных   
   циклических структур*** *(Тема 6).*
2. ***Создайте приложение*** *с именем* ***Проект-4.6.***
3. ***Выберите вариант*** *задания из таблицы табл. 4.6.4-1.*
4. ***Проведите формализацию*** *поставленной задачи.*
5. ***Разработайте графический интерфейс*** *пользователя. Предусмотрите отображение на форме номера итерации и значения вычисляемого члена бесконечной последовательности или корня уравнения.*
6. ***Разработайте схемы алгоритмов*** *решения поставленной задачи.*
7. ***Напишите программный код процедур*** *пользователя в соответствии со схемами алгоритмов. Обмен данными между процедурами должен осуществляться через параметры, без использования глобальных переменных. Событийная процедура должна содержать только операторы вызова пользовательских (общих) процедур.*
8. ***Выполните созданный проект****.*
9. *полученных результатов на заранее разработанных тестах.*

### Варианты индивидуальных заданий

Таблица 4.6.4-1

|  |  |
| --- | --- |
| **№** | **Задача** |
| **1)** | Вычислите с точностью **ε = 0.00001** константу Эйлера (основание натурального логарифма), воспользовавшись разложением в ряд:    Сравните результат со значением, полученным с помощью соответствующей встроенной функции. |
| **2)** | Вычислите и выведите те члены последовательности  ,  значения, которых больше **ε = 0.001** при **x = 0.2** |
| **3)** | Вычислите с точностью **ε = 0.00001** значение функции  при **x = 2**, воспользовавшись рекуррентной формулой:    Сравните результат со значением, полученным с помощью соответствующей встроенной функции. |
| **4)** | Вычислите и выведите те члены последовательности,    значения которых больше  **ε**  **= 0.01,** при **x = 0.6** |
| **5)** | Вычислите константу с точностью до **ε = 0.00001**, воспользовавшись разложением в ряд:    и соотношением  Сравните результат со значением, полученным с помощью соответствующей встроенной функции. |
| **6)** | Вычислите с точностью **ε** **= 0.00001** значение функции  при **x = 2**, воспользовавшись формулой:    Сравните результат со значением, полученным с помощью соответствующей встроенной функции |
| **7)** | Вычислите **sin 0.5** с точностью **ε = 0.0001**, воспользовавшись  разложением в ряд:    Сравните результат со значением, полученным с помощью соответствующей встроенной функции. |
| **8)** | Вычислить с точностью **ε = 0.00001**, воспользовавшись разложением в ряд:  Сравнить результат со значением, полученным с помощью соответствующей встроенной функции. |
| **9)** | Вычислите **cos(0.6)** с точностью **ε = 0.00001**, воспользовавшись разложением в ряд:  Сравните результат со значением, полученным с помощью соответствующей встроенной функции. |
| **10)** | Вычислите с точностью **ε = 0.0001** корень уравнения  воспользовавшись формулой:    Проверьте правильность решения подстановкой найденного корня в уравнение. |
| **11)** | Вычислите и выведите те члены последовательности,    значения, которых по модулю больше **ε** **= 0.001** при **x = 0.5**. |
| **12)** | Вычислить  при |x|<1 с точностью до **ε = 0.0001**, воспользовавшись разложением в ряд:    Сравнить результат со значением, полученным с помощью соответствующей встроенной функции. |
| **13)** | Вычислите корень уравнения с точностью **ε=0.0001**, воспользовавшись итерационной формулой    Проверьте правильность решения подстановкой найденного корня в уравнение. |
| **14)** | Вычислите значение с точностью **ε = 0.00001**, воспользовавшись представлением в виде цепной дроби:    Значение дроби равно пределу числовой последовательности, члены которой вычисляются по рекуррентной формуле до достижения заданной точности  .  Сравните результат со значением, полученным с помощью соответствующей встроенной функции. |
| **15)** | Вычислите и выведите те члены последовательности,    значения, которых по модулю больше **ε = 0.001** при **x = 0.3**. |
| **16)** | Вычислить **ln(x)** с точностью **ε = 0.0001**, воспользовавшись разложением в ряд:  Сравнить результат со значением, полученным с помощью  соответствующей встроенной функции при x=1.5. |
| **17)** | Вычислите **sh(0.3)** с точностью до **ε= 0.00005**, воспользовавшись разложением в ряд:  Сравните результат со значением, полученным с помощью встроенной функции для вычисления **ex**, используя соотношение: |
| **18)** | Вычислите корень уравнения **x-0.5(sinx2-1)=0** с точностью **ε = 0.001,** воспользовавшись итерационной формулой:    Проверьте правильность решения подстановкой найденного корня в уравнение. |
| **19)** | Вычислите **ln(2)** с точностью **ε = 0.001**, воспользовавшись представлением в виде ряда:  Сравните результат со значением, полученным с помощью встроенной функции. |
| **20)** | Вычислите с точностью **ε = 0.00001** корень уравнения  воспользовавшись итерационной формулой    Проверьте правильность решения подстановкой найденного корня в уравнение. |
| **21)** | Вычислите **ch 0.7** с точностью до **ε = 0.00005**, воспользовавшись разложением в ряд:  Сравните результат со значением, полученным с помощью встроенной функции, используя соотношение: |
| **22)** | Пусть  Дано действительное число **ε>0**. Найдите первый член , для которого выполнено условие . |
| **23)** | Вычислить  при |x|>1 с точностью до **ε = 0.0001**, воспользовавшись разложением в ряд:  Сравнить результат со значением, полученным с помощью соответствующей встроенной функции. |
| **24)** | Вычислить **ln(x+1)** с точностью **ε = 0.0001**, воспользовавшись разложением в ряд:  Сравнить результат со значением, полученным с помощью  соответствующей встроенной функции при x=0.5. |
| **25)** | Дано действительное число **b<0.** Последовательность **а1, а2, …**образована по следующему закону:  **a1=b; ak=(ak-1+1)/(1-sin2k), k=2, 3, … .**  Найдите первый неотрицательный член последовательности. |
| **26)** | Дано действительное число **x**. Вычислите приближенное значение бесконечной суммы с точностью **ε=0,00001**:  **(x-1)/x+(x-1)2/(2x2)+(x-1)3/(3x3)+… (x>1/2)** |
| **27)** | Дано действительное x. Вычислить приближенное значение  **1/x+1/(3x3)+1/(5x5)+…  (x>1)**  бесконечной суммы с точностью **ε=0.0001.** |
| **28)** | Вычислите приближенное значение бесконечной суммы с точностью **ε=0,0001** (справа от суммы дается ее точное значение, с которым можно сравнить полученный результат):  **1+1/22+1/32+… π2/6** |
| **29)** | Вычислите приближенное значение бесконечной суммы с точностью **ε=0,0001** (справа от суммы дается ее точное значение, с которым можно сравнить полученный результат):  **1-1/3+1/5-1/7+… π/4** |
| **30)** | Вычислите приближенное значение бесконечной суммы с точностью **ε=0,0001** (справа от суммы дается ее точное значение, с которым можно сравнить полученный результат):  **1/(1\*3)+1/(2\*4)+1/(3\*5)+… 3/4** |

### Содержание отчёта

1. Тема и название лабораторной работы.
2. Фамилия, имя студента, номер группы, номер варианта.
3. Задание на разработку проекта.
4. Формализация и уточнение задания.
5. Элементы, разрабатываемого проекта:
6. графический интерфейс пользователя;
7. таблица свойств объектов;
8. схемы алгоритмов процедур проекта;
9. программный код проекта.
10. Результаты выполнения проектов.
11. Доказательство правильности работы проекта.

### Первый пример выполнения задания

1. **Тема и название лабораторной работы:**

Программирование алгоритмов итеративных циклических структур.

Вычисление с заданной точностью корня заданного уравнения.

1. **Фамилия, имя студента, номер группы, номер варианта:**

Иванов И., БИН1405, вариант 13.

1. **Задание на разработку проекта:**

Создайте проект с именем *Проект-6-1* для вычисления с точностью   
 ε=10-5 корня уравнения f(x)=x3 - 2x2+x - 3=0, воспользовавшись

формулой



1. **Формализация и уточнение задания:**

Проверьте правильность решения подстановкой найденного корня в уравнение. Разработайте схему алгоритма и напишите программный код проекта в соответствии с заданием.

Вычислите производную f’(x)=3x2-4x+1 и обозначьте:

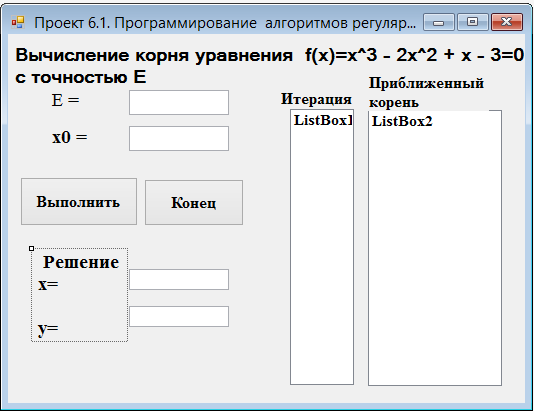
* x – текущее приближение к корню;
* a – предыдущее приближение;
* f – значение функции f(x) для предыдущего значения;
* p – значение производной f'(x) для предыдущего значения;
* i – номер итерации, совпадающий с номером текущего приближения к корню;
* y – значение функции f(x) для найденного с заданной точностью корня.

Будем считать, что заданная точность ε обеспечена, если модуль разности между текущим и предыдущим значениями корня меньше точности ε, то есть для нашего случая |x-a| < ε.

Для решения поставленной задачи необходимо реализовать процедуру   
**Kop(),** которая в качестве входных параметров получает начальное значение x0=2.2 и точность ε=10-5, и возвращает найденный корень **xl**. Процедура для вычисления корня по заданной формуле должна использовать две процедуры Function: одна – **Funy(),** вычисляет значение f(x), а другая – **Fproiz()** – значение производной этой функции ’(x). Заметим, что процедуру **Kop()** можно было оформить как Function, так как она возвращает только одно значение – вычисленный корень уравнения.

1. **Элементы, разрабатываемого проекта:**
2. Графический интерфейс пользователя:

Разработанная форма проекта имеет вид, как на рис. 4.6.4-1.



*Рис. 4.6.4-1. Форма проекта* 1-го задания ***Проект 6-1:*** Вычисление с заданной точностью корня уравнения **x3-2x2+x-3=0**

1. Таблица свойств объектов:

установите и сведите в табл. 4.6.4-2 свойства объектов.

Таблица 4.6.4-2

|  |  |  |
| --- | --- | --- |
| **Имя объектов** | **Свойство** | **Значение свойства** |
| ***Form1*** | Text | *Проект 4.6.1. Программирование алгоритмов итеративных циклических структур* |
| **Label1** | Name | *Label1* |
| Text | *Вычисление корня уравнения f(x)=x^3-2x^2+x-3=0* |
| ForeColor | *Черный* |
| Font | *Microsoft Sans Serif, Жирный, 10 пунктов* |
| **Label2** | Name | *Label2* |
| Text | *E=* |
| ForeColor | *Черный* |
| Font | *Microsoft Sans Serif, Жирный, 8 пунктов* |
| **Label3** | Name | *Label3* |
| Text | *X0=* |
| ForeColor | *Черный* |
| Font | *Microsoft Sans Serif, Жирный, 8 пунктов* |
| **Label4** | Name | *Label4* |
| Text | *Итерация* |
| ForeColor | *Черный* |
| Font | *Microsoft Sans Serif, Обычный, 8 пунктов* |
| **Label5** | Name | *Label5* |
| Text | *Приближенный корень* |
| ForeColor | *Черный* |
| Font | *Microsoft Sans Serif, Обычный, 8 пунктов* |
| **Label6** | Name | *Label6* |
| Text | *Решение x= y=* |
| ForeColor | *Черный* |
| Font | *Arial, Жирный, 12 пунктов* |
| **TextBox1** | Name | *TextBox1* |
| ForeColor | *Черный* |
| Font | *Microsoft Sans Serif, Обычный, 8 пунктов* |
| **TextBox2** | Name | *TextBox2* |
| ForeColor | *Черный* |
| Font | *Microsoft Sans Serif, Жирный, 8 пунктов* |
| **TextBox3** | Name | *TextBox3* |
| ForeColor | *Черный* |
| Font | *Microsoft Sans Serif, Жирный, 8 пунктов* |
| **TextBox4** | Name | *TextBox4* |
| ForeColor | *Черный* |
| Font | *Microsoft Sans Serif, Жирный, 8 пунктов* |
| **ListBox1** | Name | *ListBox1* |
| ForeColor | *Черный* |
| Font | *Microsoft Sans Serif, Жирный, 8 пунктов* |
| **ListBox2** | Name | *ListBox2* |
| ForeColor | *Черный* |
| Font | *Microsoft Sans Serif, Жирный, 8 пунктов* |
| **Button1** | Name | *Button1* |
| Text | *Выполнить* |
| **Button2** | Name | *Button2* |
| Text | *Конец* |

1. Схемы алгоритмов процедур проекта:

схема алгоритма процедуры **Kop()** представлена на рис. 4.6.4-2.



*Рис. 4.6.4-2. Схема алгоритма процедуры* **Kop(x)** *проекта*  ***Проект 6-1:***  Вычисление с заданной точностью корня уравнения **x3-2x2+x-3=0**

1. Программный код проекта:

разработанный программный код проекта приведен на   
 рис. 4.6.4-3.

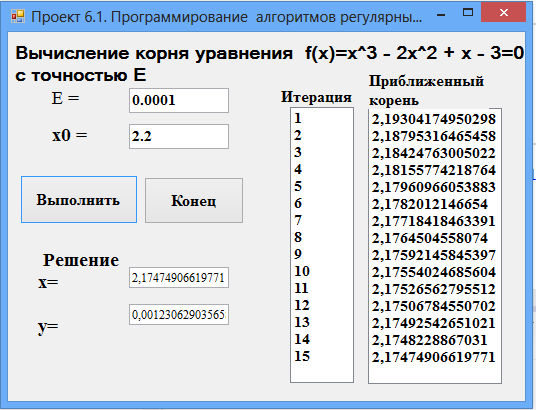
|  |
| --- |
| **Imports System.Math**  **Public Class Form1**  *'Функция ввода исходных данных из TextBox*  **Function vvod(ByVal T As TextBox) As Double**  **Return Val(T.Text)**  **End Function**  *'Процедура вывода вещественного результата в TextBox*  **Sub vivod(ByVal Z As Double, ByVal T As TextBox)**  **T.Text = CStr(Z)**  **End Sub**  *'Процедура вывода вещественного результата в ListBox*  **Sub vivodList(ByVal Z As Double, ByVal LB As ListBox)**  **LB.Items.Add(CStr(Z))**  **End Sub**  **' Процедура вывода целого результата в ListBox**  **Sub vivodListint(ByVal Z As Integer, ByVal LB As ListBox)**  **LB.Items.Add(CStr(Z))**  **End Sub**  *'процедура-Function, вычисляющая производную*  **Public Function FProiz(ByVal x As Double) As Double**  **Dim p As Double**  **p = 3 \* x ^ 3 - 4 \* x + 1**  **Return p**  **End Function**  *'процедура-Function, вычисляющая заданную функцию*  **Public Function Funy(ByVal x As Double) As Double**  **Dim f As Double**  **f = x ^ 3 - 2 \* x ^ 2 + x - 3**  **Return f**  **End Function**      *' Процедура решения задачи поиска корня*  **Public Sub Kop(ByVal E As Double, ByVal x0 As Double, \_**  **ByRef xe As Double)**  **Dim x, a As Double**  **Dim i As Integer**  **i = 0**  **x = x0**  **Do**  **a = x**  **x = a - Funy(a) / FProiz(a)**  **i = i + 1**  **vivodListint(i, ListBox1)**  **vivodList(x, ListBox2)**  **Loop Until Abs(x - a) < E**  **xe = x**  **End Sub**  **Private Sub Button1\_Click(sender As Object,e As EventArgs)\_**  **Handles Button1.Click**  **Dim EE, x0x0, xn, y As Double**  **EE = vvod(TextBox1) : x0x0 = vvod(TextBox2)**  **Kop(EE, x0x0, xn)**  **vivod(xn, TextBox3)**  **y = Funy(xn)**  **vivod(y, TextBox4)**  **End Sub**  **Private Sub Button2\_Click(sender As Object,e As EventArgs)\_**  **Handles Button2.Click**  **End**  **End Sub**  **End Class** |

*Рис. 4.6.4-3. Программный код проекта* ***Проект 6-1:***

Вычисление с заданной точностью корня уравнения **x3-2x2+x-3=0**

1. **Результаты выполнения проектов:**

Результаты выполнения проекта приведены на рис. 4.6.4-4.



*Рис. 4.6.4-4. Результаты выполнения проекта* ***Проект 6-1:***

Вычисление с заданной точностью корня уравнения **x3-2x2+x-3=0**

1. **Доказательство правильности работы программы:**

Значение функции при подстановке корня в уравнение  
f(x) = 0.00012315320113. Это говорит о том, что значение функции f(2.17457839205816) = 0.00012315320113 близко к нулю.

### Второй пример выполнения задания

1. **Тема и название лабораторной работы:**

Программирование алгоритмов итеративных циклических структур.

Вычисление членов заданной последовательности, значения которых по

модулю больше заданного числа.

1. **Фамилия, имя студента, номер группы, номер варианта:**

Иванов И., БИН1405, вариант 13.

1. **Задание на разработку проекта:**

Вычислить и вывести на экран те члены последовательности

,

значения, которых по модулю больше ε=0.0001, при x=1.5.

1. **Формализация и уточнение задания:**

Для решения поставленной задачи необходимо вывести рекуррентную формулу вычисления члена последовательности. Очевидно, что выражение для n-го члена заданной последовательности имеет вид:

.

Тогда формула для (n+1) члена последовательности имеет вид:



Имея в виду, что (n+1)!=n! ∙ (n+1), получим



Откуда получаем следующую рекуррентную формулу

 - начальный член последовательности при n=1.

1. **Элементы, разрабатываемого проекта:**
2. **Графический интерфейс пользователя:**

Разработанная форма проекта имеет вид, как на рис. 4.6.4-5.



*Рис. 4.6.4-5. Форма проекта 2-го задания* ***Проект 6-2:***

*Вычисление членов заданной последовательности, значения которых по модулю больше заданного числа*

1. **Таблица свойств объектов:**

Определите, установите и сведите в табл. П.1.6-3 свойства всех

объектов.

Таблица4.6.4-3

|  |  |  |
| --- | --- | --- |
| **Имя**  **объекта** | **Свойство** | **Значение свойства** |
| **Form1** | Text | *Проект 4.6.2. Пример 2.* |
| **Label1** | Name | *Label1* |
| Text | *E=* |
| ForeColor | *Синий* |
| Font | *Microsoft Sans Serif, Жирный, 8 пунктов* |
| **Label2** | Name | *Label2* |
| Text | *X=* |
| ForeColor | *Черный* |
| Font | *Microsoft Sans Serif, Жирный, 8 пунктов* |
| **Label3** | Name | *Label3* |
| Text | *Задание:* |
| ForeColor | *Красный* |
| Font | *Microsoft Sans Serif, Жирный, 10 пунктов* |
| **Label4** | Name | *Label4* |
| Text | *Вычислить все члены, заданной последовательности,*  *значения которых по модулю больше 0.0001* |
| ForeColor | *Черный* |
| Font | *Microsoft Sans Serif, Жирный, 8 пунктов* |
| **TextBox1** | Name | *TextBox1* |
| ForeColor | *Черный* |
| Font | *Microsoft Sans Serif, Жирный, 10 пунктов* |
| **TextBox2** | Name | *TextBox2* |
| ForeColor | *Черный* |
| Font | *Microsoft Sans Serif, Жирный, 8 пунктов* |
| **ListBox1** | Name | *ListBox1* |
| ForeColor | *Черный* |
| Font | *Microsoft Sans Serif, Жирный, 8 пунктов* |
| **ListBox2** | Name | *ListBox2* |
| ForeColor | *Черный* |
| Font | *Microsoft Sans Serif, Жирный, 8 пунктов* |
| **Button2** | Name | *Button2* |
| Text | *Вычислить значения* |
| **Button1** | Name | *Button1* |
| Text | *Стоп* |

1. **Схемы алгоритмов процедур проектов**

Схема алгоритма процедуры **Pos()** представлена на рис. 4.6.4-6.

|  |
| --- |
|  |

*Рис. 4.6.4-6. Схема алгоритма процедуры Pos() проекта* **Проект 6-2***:*

*Вычисление членов заданной последовательности, значения которых по модулю больше заданного числа*

1. **Программный код проекта:**

Разработанный программный код проекта приведен на рис. 4.6.4-7.

|  |
| --- |
| **Public Class Form1**  *'Функция ввода исходных данных из TextBox*  **Function vvod(ByVal T As TextBox) As Double**  **Return Val(T.Text)**  **End Function**  *' Процедура вывода вещественного результата в ListBox*  **Sub vivodList(ByVal Z As Double, ByVal LB As ListBox)**  **LB.Items.Add(CStr(Z))**  **End Sub**  *' Процедура вывода целого результата в ListBox*  **Sub vivodint(ByVal Z As Integer, ByVal LB As ListBox)**  **LB.Items.Add(CStr(Z))**  **End Sub**    **' Процедура вычисления и вывода членов последовательности с**  **' заданной точностью**  **Private Sub Pos(ByVal x As Double, ByVal E As Double)**  **Dim n As Integer**  **Dim a As Double**  **a = x - 1**  **n = 1**  **Do While a > E**  **vivodint(n, ListBox1)**  **vivodList(a, ListBox2)**  **a = a \* (x - 1) / (n + 1)**  **n = n + 1**  **Loop**  **End Sub**  **Private Sub Button1\_Click(sender As Object, e As EventArgs)\_**  **Handles Button1.Click**  **Dim EE, x0, y As Double**  **EE = vvod(TextBox1) : x0 = vvod(TextBox2)**  **Pos(x0, EE)**  **End Sub**  **End Class** |

*Рис. П.1.6-7. Программный код проекта 2-го задания* **Проект 6-2***:*

*Вычисление членов заданной последовательности, значения которых по модулю больше заданного числа*

1. **Результат выполнения проекта**

Результат выполнения проекта приведен на рис. 4.6.4-8.

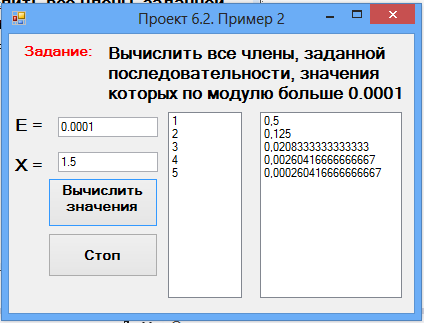


Рис. 4.6.4-8. Результаты выполнения проекта 2-го задания **Проект 6-2**

1. **Доказательство правильности работы программы:**

Все выведенные значения последовательности больше 0.0001.

### Контрольные вопросы

1. Какой алгоритм называется циклическим?
2. Какой вид имеет цикл с оператором **Do While…Loop** или оператором **Do…Loop Until** в схемах алгоритмов?
3. Как записывается цикл с оператором **Do While…Loop** или оператором **Do…Loop Until** в программах?
4. Как работают циклы с оператором **Do While…Loop, Do…Loop Until**?
5. Какой цикл называется итеративным?
6. В каком случае целесообразно использовать рекуррентную формулу?
7. Какие формулы называются рекуррентными соотношениями?
8. Назовите условия окончания итерационного процесса?
9. Что представляет собой итеративный цикл?
10. Что представляет собой цикл с предусловием?
11. Что представляет собой цикл с постусловием?
12. Что представляют собой циклы **Do** и каковы их разновидности?
13. Что такое рекуррентная формула?
14. В чем отличие организации регулярных и итеративных циклов?
15. Какой оператор предназначен в языке VB для организации итеративных циклов?
16. В чем отличие использования в операторе **Do** конструкции **While** и **Until**?
17. Каким образом можно выйти из цикла до его завершения?
18. Как определяется число повторений операторов тела цикла в итеративной циклической структуре?
19. Какие операторы могут находиться в теле цикла оператора **Do…Loop**?
20. Сколько раз могут выполняться операторы тела цикла **Do…Loop**?
21. Какой оператор используется для программирования циклических алгоритмических структур с неизвестным числом повторений?
22. Какова алгоритмическая структура цикла итеративного типа?
23. Сколько раз будет выполняться тело цикла, если при программировании циклической структуры используется оператор **Do While…Loop**?
24. Сколько раз будет выполняться тело цикла, если при программировании циклической структуры используется оператор **Do …Loop While**?
25. Сколько раз будет выполняться тело цикла, если при программировании циклической структуры используется оператор **Do …Loop Until**?
26. Какие алгоритмы можно реализовать с использованием итеративных циклических структур?
27. Что записывается после ключевых слов **While** или **Until** в операторе итеративного цикла?