

ПРОСТОЙ ЦИКЛ

1. Базовые структуры алгоритмов

Алгоритм – это правило получения решения некоторой задачи, выраженное в виде совокупности конечного числа элементарных действий. Мы часто пользуемся алгоритмами. Например, когда в столбик складываем два числа, то применяем алгоритм. Известны различные способы записи алгоритма. Любая программа на языке VB – это тоже запись соответствующего алгоритма. Однако наибольшей наглядностью обладает изображение алгоритма в виде блок-схемы. В сложных случаях, чтобы правильно разработать алгоритм иногда полезно перед записью кода изобразить его блок-схему.

При разработке алгоритма необходимо применять такие технологические рекомендации, при соблюдении которых алгоритм получался бы наиболее понятным, а ошибки при записи алгоритма в виде программы были бы наименее вероятны. В соответствии с современными воззрениями в сегодняшней технологии разработки программ, которые представляют собой одну из сторон метода структурного программирования, алгоритм должен быть структурирован. Он может включать только базовые структуры, которых всего три: следование, разветвление и цикл.

1.1. Следование

Эта структура, изображенная на рис. 1, предполагает последовательное выполнение входящих в нее инструкций. Существенно, что структура следование, рассматриваемая как единое целое (на рис. 1 она заключена в пунктирный прямоугольник), имеет один вход и один выход.

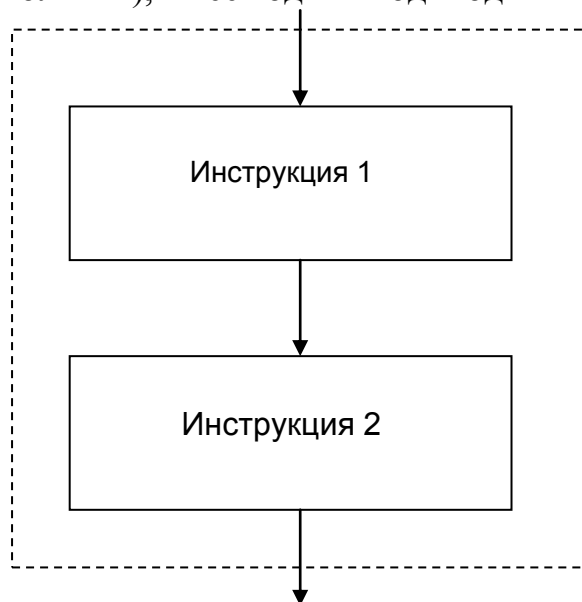


Рис. 1. Следование

1.2. Разветвление

Разветвление, блок-схема которого приведена на рис. 2, предполагает проверку некоторого условия. В зависимости от того выполняется это условие или нет, выполняется либо одна инструкция, либо другая.

Если на момент проверки условие оказалось выполнено, то будет выполнена инструкция 1, а инструкция 2 игнорируется. Если же оказывается, что условие не выполнено, то будет выполнена инструкция 2, а инструкция 1 игнорируется.

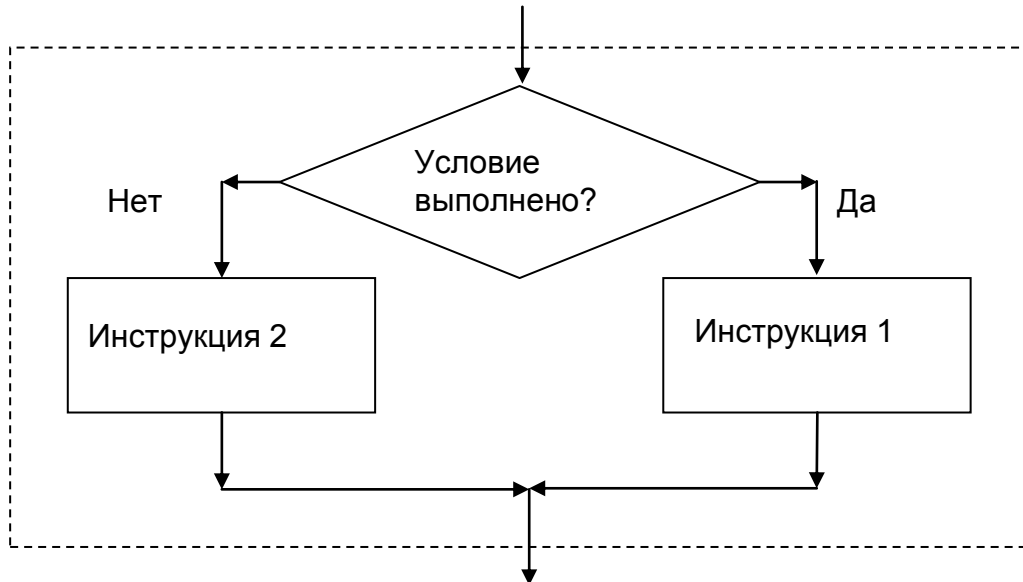


Рис. 2. Разветвление

Разветвление также имеет один вход и один выход.

1.3. Цикл

На рис. 3 изображены два варианта блок-схемы базового цикла. Цикл имеет место, когда на блок-схеме по линии стрелок имеет место замкнутый контур.

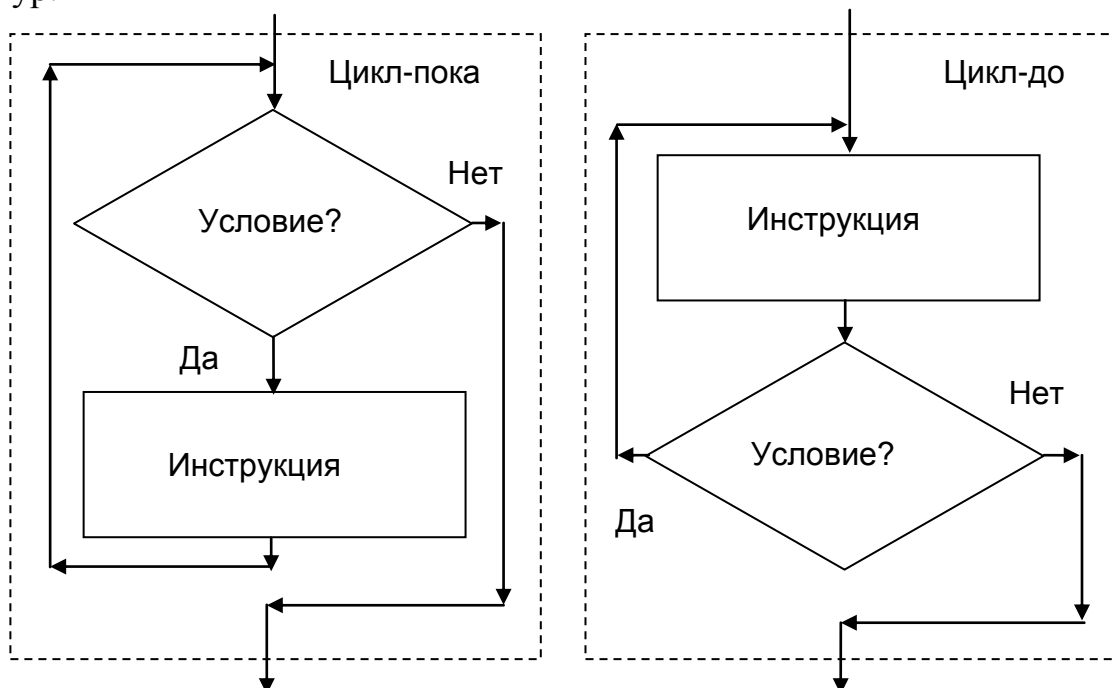


Рис. 3. Две разновидности цикла

Цикл предназначен для многократного выполнения некоторой инструкции. Если сначала выполняется проверка условия необходимости выпол-

нения инструкции, а затем ее выполнение, то такой цикл называется циклом – пока. Если же сначала выполняется инструкция, а затем выполняется проверка необходимости ее повторения, то такой цикл называется циклом – до. Базовый цикл имеет один вход и один выход.

Итак, каждая из трех рассмотренных видов базовых структур имеет один вход и один выход. Это весьма существенное обстоятельство допускает, чтобы любой прямоугольник с надписью «Инструкция», изображенный на рисунках 1 – 3, мог быть базовой структурой. Значит, цикл может включать в свой состав базовые структуры: следование, разветвление, цикл. Это утверждение также относится и к двум другим базовым структурам – следование и разветвление.

Программирование на VB структуры – следование не встречает затруднений.

Программирование разветвлений было в упрощенном виде рассмотрено ранее в работе «Сигнал». Сейчас займемся программированием циклов.

2. Инструкции цикла

Инструкция цикла представляет собой указание, из которого ясно, какой набор инструкций (этот набор инструкций называется телом цикла) нужно выполнять многократно и каково правило окончания выполнения цикла.

2.1. Параметрический цикл **For... Next**

Этот цикл управляется параметром, который при повторении выполнения тела цикла изменяет с заданным шагом свое значение от заданного начального значения до заданного конечного значения. Такой цикл используют в случае, когда заранее известно, сколько необходимо совершить повторений выполнения тела цикла.

Синтаксис цикла:

For ИПЦ = НЗПЦ **To** КЗПЦ [**Step** ШИПЦ]

Тело цикла (одна или несколько инструкций)

Next [ИПЦ]

Слова **For** (для), **To** (до), **Step** (шаг), **Next** (затем) являются зарезервированными.

Применены обозначения:

ИПЦ – имя параметра цикла (переменная любого числового типа);

НЗПЦ – начальное значение параметра цикла (выражение любого числового типа), которое параметр цикла будет иметь при первом выполнении тела цикла;

КЗПЦ – конечное значение параметра цикла (выражение любого числового типа), с которым сравнивается текущее значение параметра цикла;

ШИПЦ – шаг изменения параметра цикла (выражение любого числового типа) – необязательная часть инструкции цикла.

Числовые значения НЗПЦ и КЗПЦ задают интервал, в котором будет изменяться параметр цикла. Необязательный параметр ШИПЦ задает шаг изменения значения параметра цикла после каждого прохода. По умолчанию, если он отсутствует, то значение шага изменения параметра цикла принимается равным 1.

Наконец, после инструкций, составляющих тело цикла, следует строка `Next`, обозначающая границу действия цикла. В случае вложенных циклов (в тело цикла входит инструкция цикла) полезно (но не обязательно) указывать, к какому из них относится команда `Next`. Это достигается добавлением после слова `Next` имени параметра цикла.

Процесс выполнения инструкции `For... Next` для положительного шага иллюстрирует рис. 4.

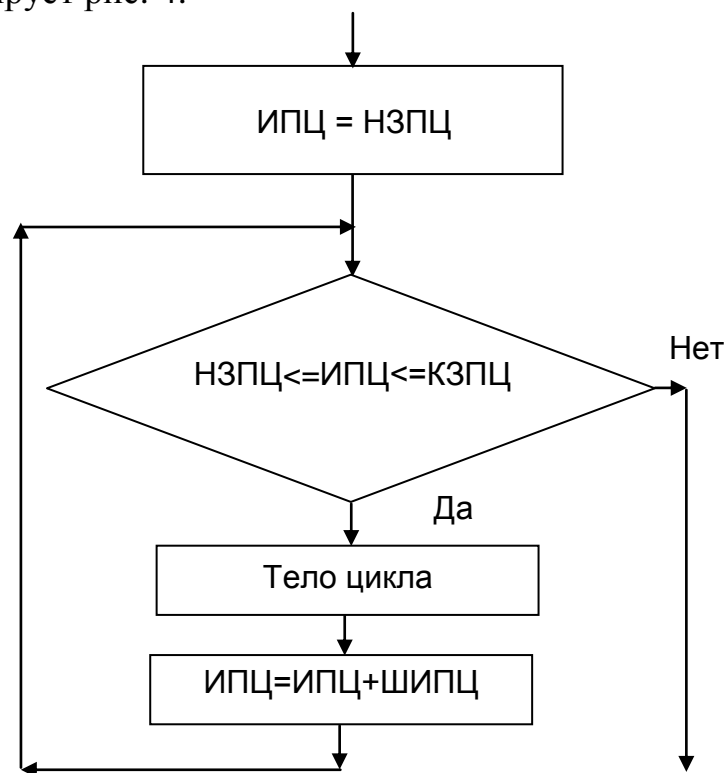


Рис. 4. Блок-схема цикла `For...Next`

Рассмотрим примеры.

В первом примере запишем инструкции для вычисления суммы всех целых нечетных чисел от 1 до 100.

```

Dim i, Сумма As Integer
For i = 1 To 100 Step 2
    Сумма = Сумма + i
Next
  
```

В следующем примере будут продемонстрированы две возможности: явно заданный шаг цикла и отсчет в обратном направлении. Последнее достигается заданием отрицательного шага и тем, что начальное значение параметра цикла больше, чем конечное.

```

Dim n As Integer
  
```

```

For n = 100 To 60 Step -10
    MsgBox (n)
Next

```

Этот код последовательно выведет окна функции MsgBox, содержащие значения: 100, 90, 80, 70, 60.

Следующая инструкция заставляет компьютер подавать звуковой сигнал 50 раз. Инструкция For определяет, что параметром цикла является переменная x, ее начальное, конечное значения и шаг изменения параметра цикла. Команда Next изменяет значение параметра с заданным шагом.

```

Dim x As Integer
For x = 1 To 50
    Beep
Next

```

Инструкцию For...Next можно завершить досрочно с помощью инструкции Exit For. Выполнение этой инструкции приводит к немедленному выходу из цикла.

2.2. Инструкция цикла **Do While...Loop** или **Do...Loop While**

Здесь While (пока) и Loop (цикл) зарезервированные слова. Циклы типа While предназначены для ситуаций, когда количество повторений тела цикла (итераций) заранее не известно. Вот синтаксис двух разновидностей цикла While:

```

Do While УсловиеПовторения
    Группа инструкций
Loop

```

```

Do
    Группа инструкций
Loop While УсловиеПовторения

```

Различие между ними заключается в том, что УсловиеПовторения (условие повторения выполнения тела цикла) проверяется в первом случае до выполнения тела цикла (цикл – пока), а во втором случае – после выполнения тела цикла (цикл – до).

Перейдем к примерам.

Рассмотрим действие следующего участка программы.

```

Счетчик = 0
Номер = 20
Do While Номер > 10
    Номер = Номер - 1
    Счетчик = Счетчик + 1
Loop
MsgBox ("Тело цикла выполнено " & Счетчик & " раз.")

```

При выполнении этого участка программы в окне функции MsgBox будет выведено:

Тело цикла выполнено 10 раз.

В этой программе условие проверяется до входа в цикл. Если переменной Номер задать значение, равное 9 вместо 20, инструкции внутри цикла вообще выполняться не будут.

В следующей программе инструкции внутри цикла выполняются только один раз до того, как условие не будет выполнено.

Счетчик = 0

Номер = 9

Do

Номер = Номер - 1

Счетчик = Счетчик + 1

Loop While Номер > 10

MsgBox ("Тело цикла выполнено " & Счетчик & " раз.")

Инструкцию Do...Loop можно завершить досрочно с помощью инструкции Exit Do.

2.3. Инструкция цикла **Do Until...Loop** или **Do...Loop Until**

Until (до) – зарезервированное слово. По своей логике цикл Until подобен циклу While с той лишь разницей, что выполнение условия означает необходимость выхода из цикла. Как и в случае цикла While, проверка условия выхода в цикле Until может осуществляться перед очередным проходом или после него. Вот синтаксис этих двух вариантов:

Do Until УсловиеВыхода

Группа инструкций

Loop

Do

Группа инструкций

Loop Until УсловиеВыхода

Перейдем к примерам.

Рассмотрим действие участка программы:

Счетчик = 0

Номер = 20

Do Until Номер = 10

Номер = Номер - 1

Счетчик = Счетчик + 1

Loop

Переменная Счетчик получит значение 10.

Еще один пример:

Счетчик = 0

Номер = 1

Do

Номер = Номер + 1

Счетчик = Счетчик + 1

Loop Until Номер = 10

Переменная Счетчик получит значение 9.

Инструкцию Do...Loop можно завершить досрочно с помощью инструкции Exit Do.

3. Операции и встроенные математические функции

В расположенной ниже таблице перечислены операции, которые могут быть применены к числовым данным:

– операнд	Изменение знака	
операнд1 / операнд2	Деление	
операнд1 \ операнд2	Целочисленное деление. Результат – целая часть отношения операндов	
операнд1 MOD операнд2	Деление по модулю. Результат – дробная часть отношения операндов	
операнд1 ^ операнд2	Возведение в степень	
Операции отношения		
операнд1 < операнд2	Меньше	Результат True, если отношение выполняется, или False, если отношение не выполняется
операнд1 > операнд2	Больше	
операнд1 <= операнд2	Меньше или равно	
операнд1 >= операнд2	Больше или равно	
операнд1 = операнд2	Равно	
операнд1 <> операнд2	Не равно	

Если в выражении использовано несколько операций, то в первую очередь выполняются операции, имеющие наивысший приоритет. Если приоритет операций одинаковый, то они выполняются слева направо.

Таблица приоритетов

Приоритет	Операция	Приоритет	Операция
1	Вызов функции, скобки.	5	\
2	^	6	MOD
3	– (изменение знака)	7	+, –
4	*, /		

Основные функции класса Math (углы выражаются в радианах)

Обращение к функции	Возвращаемое значение
Abs (x)	Абсолютное значение x
Atan (x)	arctg x
Cos (x)	cos x
Exp (x)	e ^x
Fix (x) , Int (x)	Целая часть
Log (x)	ln x
Log10 (x)	lg x
Round (x[, n])	Округление (n – число разрядов дробной части после округления)
Sin (x)	sin x
Sqrt (x)	Квадратный корень от x
Tan (x)	tg x
Rnd	Случайное число

В VB.NET имеются разнообразные встроенные функции, которые разделены на классы и реализованы как методы классов. Например, математические функции, являются методами класса Math. Наиболее часто применяемые математические функции приведены в приведенной выше таблице. Если, например, требуется вычислить значение $\sin x$, то это в программе следует записать `Math.Sin(x)`. Чтобы воспользоваться этими функциями, не указывая их полное наименование (например, `Math.Sin(x)`), в проекте необходимо выполнить импорт пространства имен `System.Math`, добавив в исходном коде (но не внутри процедуры) следующую строку:

```
Imports System.Math
```

Функции `Int` и `Fix` возвращают значение типа, совпадающего с типом аргумента, равное целой части числа.

Обязательный аргумент должен иметь числовое значение типа `Double`. Обе функции `Int` и `Fix` отбрасывают дробную часть числа и возвращают целое значение. Различие между функциями `Int` и `Fix` состоит в том, что для отрицательного значения аргумента функция `Int` возвращает ближайшее отрицательное целое число, меньшее либо равное указанному, а `Fix` ближайшее отрицательное целое число, большее либо равное указанному. Например, функция `Int` преобразует -8.4 в -9, а функция `Fix` преобразует -8.4 в -8.

Функция `Rnd` возвращает значение типа `Single`, содержащее случайное число, меньшее 1 и большее или равное нулю.

Перед первым вызовом функции `Rnd` надо использовать инструкцию `Randomize` без аргумента для инициализации генератора случайных чисел. Для получения случайных целых чисел в заданном диапазоне используйте следующую формулу:

```
CInt(Int((ВерхнееЗначение - НижнееЗначение + 1) * Rnd()  
+ НижнееЗначение))
```


4. Задание

4.1. Условие

Задано аналитическое выражение для некоторой функции $f(x)$. Составьте проект для получения таблицы значений функции $f(x)$ при n значениях аргумента x , изменяющегося от начального значения a до конечного значения b с постоянным шагом $dx=(b-a)/(n-1)$

Вид $f(x)$, значения a и b возьмите из таблицы вариантов, приведенной в пункте 4.2.

При выполнении проекта задайте $n = 11$.

Результаты вычислений должны быть получены в виде таблицы:

Аргумент	Функция
0.0000000	-1.0000000
⋮	⋮
1.0000000	0.4596979

Отчет по выполненной работе должен включать: титульный лист (см. приложение 5), условие задачи вместе с условием варианта, описание применяемых данных, блок-схему алгоритма, и программный код, а также график функции, построенный по результатам вычислений.

4.2. Таблица вариантов

Варианты задания приведены в таблице:

Номер варианта	Заданная функция $f(x)$	Отрезок $[a, b]$	Значение $f(a)$
1	$\frac{3,8 - 3 \sin \sqrt{x}}{0,35} - x$	$[2; 3]$	0,3905776
2	$\frac{1}{3 + \sin 3,6x} - x$	$[0; 0,85]$	0,3333333
3	$\cos \sqrt{1 - 0,3x^3} - x$	$[0; 1]$	0,5403023
4	$\sin \sqrt{1 - 0,4x^2} - x$	$[0; 1]$	0,841471
5	$0,25x^3 - x - 1,2502$	$[2; 3]$	-1,2502
6	$0,1x^2 - x \ln x$	$[1; 2]$	0,1
7	$3x - 4 \ln x - 5$	$[2; 4]$	-1,772588
8	$e^x - e^{-x} - 2$	$[0; 1]$	-2
9	$x + \sqrt{x} + \sqrt[3]{x} - 2,5$	$[0,4; 1]$	-0,7307382
10	$tgx - \frac{tg^3 x}{3} + \frac{tg^5 x}{5} - \frac{1}{3}$	$[0; 0,8]$	-0,3333333
11	$\cos \frac{2}{x} - 2 \sin \frac{1}{x} + \frac{1}{x}$	$[1; 2]$	-1,099089

Номер варианта	Заданная функция f(x)	Отрезок [a,b]	Значение f(a)
12	$\sin(\ln x) - \cos(\ln x) + 2\ln x$	[1; 3]	-1
13	$\ln x - x + 1,8$	[2; 3]	0,4931472
14	$0,4 + \arctg \sqrt{x} - x$	[1; 2]	0,1853982
15	$x \operatorname{tg} x - \frac{1}{3}$	[0,2; 1]	-0,2927913
16	$\operatorname{tg}(0,55x + 0,1) - x^2$	[0; 1]	0,1003347
17	$2 - \sin \frac{1}{x} - x$	[1,2; 2]	0,0598231
18	$1 + \sin x - \ln(1+x) - x$	[0;1, 5]	1
19	$-\cos(x^{0,52+2}) - x$	[0,5; 1]	0,4029458
20	$\sqrt{\ln(1+x) + 3} - x$	[2; 3]	0,024503
21	$e^x + \ln x - 10x$	[3; 4]	-8,815851
22	$3x - 14 + e^x - e^{-x}$	[1; 3]	-8,649598
23	$2\ln^2 x + 6\ln x - 5$	[1; 3]	-5
24	$2x \sin x - \cos x$	[0,4; 1]	-0,6095263
25	$\cos x - e^{\frac{x^2}{2}} + x - 1$	[1; 2]	-0,0662284
26	$\sqrt{1-x} - \operatorname{tg} x$	[0; 0,9]	1
27	$\sin^2 x + \cos^2 x - 10x$	[0; 1]	1
28	$e^x + \sqrt{1 + e^{2x}} - 2$	[-1; 0]	-0,5665994
29	$\sqrt{1-x} - \cos \sqrt{1-x}$	[0; 0,9]	0,4596977
30	$\operatorname{tg} \frac{x}{2} - \operatorname{ctg} \frac{x}{2} + x$	[1; 2]	-0,2841852
31	$x - \cos x$	[0; 1]	-1,0000000

5. Пример выполнения задания

Выполним задание для варианта № 31.

5.1. Выбор данных

Сначала зададим имена и типы применяемых переменных. При выборе имени переменной нужно насколько возможно, придерживаться принятых обозначений в условии задачи, или выбирать имя таким, чтобы оно напоминало назначение переменной.

Для переменных, принимающих числовые значения, целый тип следует применять только тогда, когда переменная в силу своего назначения может принимать только целочисленные значения, например, обозначающая количество студентов в группе или номер строки в некотором списке. Для переменных, значения которых могут иметь дробную часть, надо применять тип с плавающей точкой. Точность выбирается ординарная, если достаточно

6 – 7 значащих десятичных разрядов, или двойная, обеспечивающая 14 – 15 значащих десятичных разрядов.

Различают:

- исходные данные, значения которых задаются при каждом выполнении программы;
- результаты, значения которых мы должны получить в результате выполнения программы;
- промежуточные данные, значения которых получаются в ходе выполнения программы, но не интересуют нас после окончания ее выполнения.

Итак, для рассматриваемого примера будем применять следующие данные.

Исходные:

a – левая граница отрезка, переменная обычной точности с плавающей точкой;

b – правая граница отрезка, переменная обычной точности с плавающей точкой;

n – количество узлов, в которых вычисляется значение заданной функции.

Результаты:

x – аргумент функции $f(x)$, переменная обычной точности с плавающей точкой;

f_x – значение функции $f(x)$ при заданном значении x , переменная обычной точности с плавающей точкой.

Промежуточные:

dx – шаг изменения аргумента, переменная обычной точности с плавающей точкой;

i – параметр цикла, переменная целого типа.

5.2. Разработка алгоритма

Алгоритм включает в свой состав задание значений исходных данных, вычисление шага dx изменения аргумента, а также вычисление и вывод в цикле результатов. Блок-схема алгоритма изображена на рис. 5.

Выполнение алгоритма начинается с задания значений исходных данных, к которым отнесены переменные n , a и b . После этого вычисляется значение шага dx изменения аргумента x , а также задается начальное значение аргумента.

После этого следует циклический участок, в котором для каждого узла, начиная с узла номер 1, вычисляется значение функции f_x , выводятся полученные результаты (переменные x и f_x), изменяется значение аргумента, а также номер узла и проверяется необходимость продолжения цикла.

5.3. Выполнение проекта на компьютере.

1. Создайте новый проект с именем ПростойЦикл, следуя приложению 1.

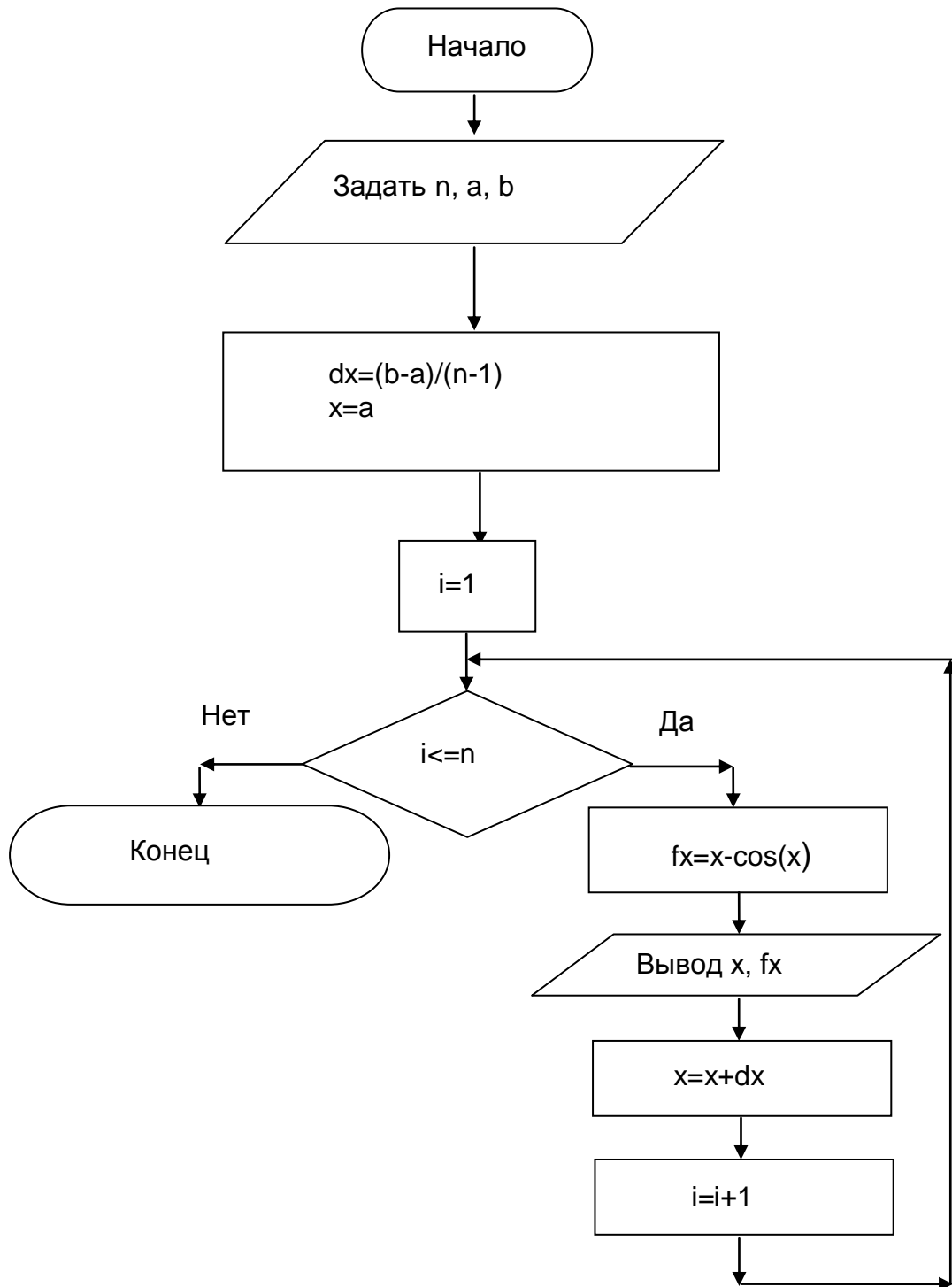



Рис. 5. Блок – схема алгоритма

2. Если окно конструктора форм не открыто, то откройте его щелчком на кнопке  View Designer (просмотреть конструктор), расположенной на панели инструментов окна обозревателя решений. Если же эта кнопка на панели инструментов отсутствует, то предварительно щелкните в окне обозревателя решений на компоненте проекта Form1.vb.

3. Интерфес, который Вам предстоит разработать, будет неоднократно применяться в следующих лабораторных работах. Поэтому описание

порядка его разработки вынесено в отдельное приложение 2. Для разработки интерфейса следуйте приложению 2.

4. Двойным щелчком на кнопке btnПуск вставьте в программный код заготовку подпрограммы btnПуск_Click – обработчика события, заключающегося в щелчке на кнопке btnПуск. Это событие будет для проекта командой выполнить требуемые вычисления.

5. Введите код, показанный на листинге 1, используя уже созданную в предыдущем пункте заготовку подпрограммы btnПуск_Click.

Листинг 1. Код проекта

```
Imports System.Math
Public Class Form1
    Private Sub btnПуск_Click(ByVal sender As System.Object, _
        ByVal e As System.EventArgs) Handles btnПуск.Click
1:      txtЖурнал.Clear()
        Dim a, b, x, fx, dx As Single, n, i As Integer
        n = InputBox("n = ?")
        txtЖурнал.AppendText("n = " & n & vbCrLf)
5:      a = InputBox("a = ?")
        txtЖурнал.AppendText("a = " & a & vbCrLf)
        b = InputBox("b = ?")
        txtЖурнал.AppendText("b = " & b & vbCrLf)
        dx = (b - a) / (n - 1)
10:     x = a
        txtЖурнал.AppendText(" Аргумент" & vbTab & "Функция" _
            & vbCrLf)
        For i = 1 To n
            fx = x - Cos(x)
            txtЖурнал.AppendText(x _
                & vbTab & vbTab & fx & vbCrLf)
15:     x = x + dx
        Next
    End Sub
End Class
```

Первая строка кода импортирует в проект класс System.Math, что необходимо при применении математических функций.

Далее приведены пояснения к подпрограмме btnПуск_Click.

В строке кода 1 этой подпрограммы выполняется очистка текстового окна от информации, выведенной в нем при возможном предыдущем запуске вычислений.

В строке кода 2 этой подпрограммы объявляются исходные данные, результаты и промежуточные данные.

Строка 3 является инструкцией присвоения. В правой части этой инструкции обеспечивается ввод значения переменной n. При этом происходит обращение к функции InputBox, упрощенный синтаксис которой InputBox ("Сообщение"). Эта функция выводит диалоговое окно, содержащее "Сообщение". В поле этого диалогового окна можно ввести

строку. После щелчка на кнопке ОК эта функция возвращает строку, набранную в ее диалоговом окне.

В строке кода 4 для контроля правильности выполненного ввода значения n , это значение с помощью метода `AppendText` выводится как строка в текстовом поле `txtЖурнал`. Метод `AppendText(Строка)` выводит в текущей позиции текущей строки текстового поля строку, являющуюся его аргументом. В этой строке 4 аргумент `"n = " & n & vbCrLf` метода `AppendText` состоит из трех частей, объединенных в одну строку операциями сцепления. Последняя часть `vbCrLf` в этой строке обеспечивает, что текущим становится первый символ следующей строки текстового поля.

Строки кода 6, 8 и 11 должны быть понятны после пояснения, относящегося к строке 4. Константа `vbTab` в строке 11 обеспечивает включение интервала в строке вывода.

Строки кода 12 – 16 являются инструкцией цикла. В строке кода 13 `Cos(x)` – это метод класса математических функций `Math`, возвращающий значение функции *cos* от аргумента x .

6. Двойным щелчком на надписи `Label1` вставьте в программный код заготовку подпрограммы `Label1_Click` – обработчика события, заключающегося в щелчке на надписи `Label1`. Это событие будет для проекта командой прекратить его выполнение.

7. Введите в процедуре единственную инструкцию в соответствии с листингом 2.

Листинг 2. Код подпрограммы `Label1_Click`

```
Private Sub Label1_Click(ByVal sender As System.Object, _
    ByVal e As System.EventArgs) Handles Label1.Click
    End
End Sub
```

8. Запустите свой проект. Щелкните на кнопке «Выполнить вычисления», задайте значения $n = 11$, $a = 0$ и $b = 1$. Если Вы не допустили ошибок, то результат работы проекта должен соответствовать рис. 6.

Обратите внимание, аргумент x изменяется с шагом, равным 0.1 . Но это понятное правило нарушено в тех двух узлах, где аргумент должен был быть равен 0.8 и 0.9 соответственно. Объясните, по какой причине это произошло?

В отображении результатов в текстовом поле таблица значений функции, столбцы которой не выровнены по заголовку, выглядит не лучшим образом. Для устранения этого недостатка можно потребовать, например, чтобы значение аргумента всегда выводилось с одинаковым количеством разрядов дробной части. Для этого при преобразовании с помощью метода `ToString` числового значения аргумента x в строку следует задать формат этого преобразования. Например, можно записать `x.ToString("#0.0000000")`. Формат, который указывается как

аргумент метода ToString, является строкой. Символы 0 в этой строке означают, что в соответствующей позиции цифра обязательна, даже если она является начальным или заключительным нулем. Символ # означает, что начальный ноль в этой позиции должен быть опущен. Приведенный пример формата задает преобразование числового значения в строку с 7 разрядами дробной части.

9. Внесите изменения в код строки 14 процедуры btnПуск_Click. Теперь в ней должно выполниться форматирование преобразований в строку значений аргумента x. Кроме этого колонка значений функции сдвинута вправо еще на одну позицию табуляции. Эта строка должна выглядеть так:

```
txtЖурнал.AppendText(x.ToString("#0.0000000") & vbTab & vbTab & fx & vbCrLf)
```

10. Снова запустите проект и получите результаты решения для тех же значений исходных данных. Результат работы проекта должен соответствовать рис. 7.

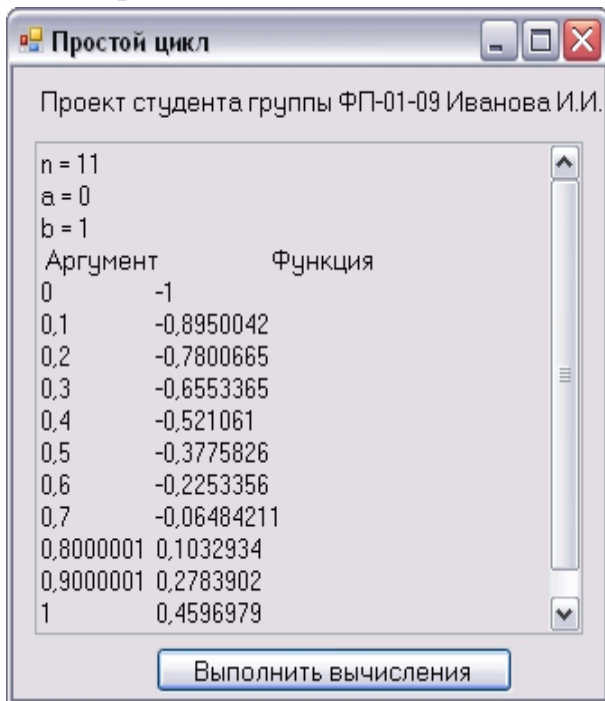


Рис. 6. Результат вычислений

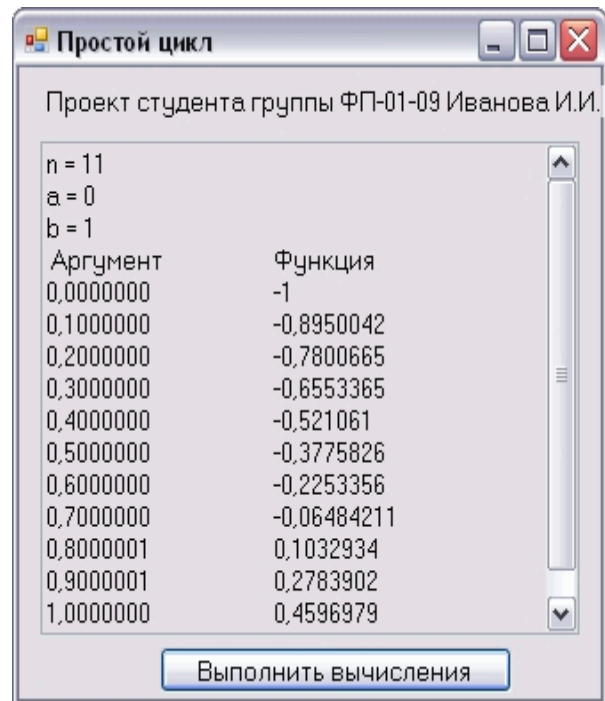


Рис. 7. Результат вычислений с выравниванием столбцов

11. Внесите в код процедуры btn_Click изменение вида функции, соответствующие Вашему индивидуальному варианту задания.

12. Запустите проект и выполните счет, предварительно задав $n = 11$ и указанные для Вашего варианта задания значения a и b . Осмыслите полученный результат.



Проверьте, совпадает ли для Вашего варианта указанные в таблице вариантов тестовое значение $f(a)$ с полученным значением в результате выполнения счета. Если совпадения нет, найдите и устраните ошибку.

13. Остановите работу приложения.

14. Покажите полученный результат преподавателю.

15. Внесите необходимые изменения в программу для замены инструкции `For ... Next` на инструкцию `Do While ... Loop`. Проверьте работу приложения. Результаты должны совпадать с результатами, полученными в пункте 12.

После запуска проекта на выполнение возможна ситуация, когда время идет, а результаты счета либо безостановочно выводятся, либо их вообще нет. Это может являться следствием ошибки, допущенной при программировании цикла. Происходит так называемое заикливание. Оно состоит в том, что условие выхода из цикла вообще никогда не выполняется. Бесконечно повторно выполняются инструкции тела цикла, а выход из цикла так и не происходит.

Для выхода из режима выполнения (run) следует нажать на стандартной панели инструментов на кнопку  Stop Debugging (остановить отладку) или в заголовке формы нажать на кнопку  Закрыть и затем в открывшемся окне Завершение программы нажать на кнопку Завершить сейчас.

После этого следует найти и устранить ошибку, которая привела к заикливанию.

Проанализируйте полученные результаты на их соответствие заданному отрезку $[a, b]$.

16. Покажите полученный результат преподавателю.

17. Внесите необходимые изменения в программу для замены инструкции `Do While ... Loop` на инструкцию `Do Until ... Loop`. Проверьте работу приложения. Результаты должны совпадать с результатами, полученными в пункте 12.

18. Попробуйте ответить на вопросы для контроля.

19. Покажите полученный результат и отчет по выполненной работе преподавателю.

20. Копируйте рабочую папку либо на сетевой диск *o* в папку с шифром Вашей учебной группы, либо на собственную флэш-панель.

21. Удалите на диске *d* свою рабочую папку.

6. Вопросы для контроля

1. Какие структуры алгоритмов относят к базовым структурам?
2. Приведите пример следования.
3. Приведите пример разветвления.
4. Приведите пример цикла-до.
5. Приведите пример цикла-пока.
6. Поясните синтаксис и порядок выполнения инструкции `For`.

7. Что является причиной того, что значение аргумента x при выполнении цикла может не совпадать точно с теоретически рассчитанным значением аргумента?
8. Поясните синтаксис и порядок выполнения инструкции `While`.
9. Поясните синтаксис и порядок выполнения инструкции `Until`
10. Каково назначение в коде проекта строки: `Imports System.Math`?
11. Что определяют значения свойств надписи: `Text`, `TextAlign`?
12. Что определяют значения свойств кнопки: `Name`, `Text`, `Font`?
13. Что определяют значения свойств текстового поля: `Name`, `Text`, `Font`, `Multiline`, `ReadOnly`, `Anchor`, `SkrollBars`?
14. Каково назначение и в чем состоит выполнение функции `InputBox ("Сообщение")`?
15. Каково назначение метода `AppendText (Строка)`?
16. Каково назначение метода `ToString` и варианта его применения `ToString("#0.0000000")`?
17. Каково назначение метода `Clear()`?