## Тема 3

## Структура VB-программ, классы и процедуры. Средства программирования алгоритмов линейной структуры

### 4.3.1. Структура простейших VB-программ

При создании простых **Приложений Windows Forms *программный код проекта***, написанный на VB**,** может состоять из ***программного кода, генерируемого средой разработки, и программного кода, написанного или дописанного программистом.*** Причем базовая структура программы создается автоматически в тот момент, когда разработчик проекта выбирает в окне **Создание проекта** шаблон **Приложение Windows Forms.**

Простые **Приложения Windows Forms** могут содержать следующие составляющие:

* операторы **Option**, определяющие основные правила обработки программного кода, предупреждающие синтаксические и логические ошибки:

Option Explicit,

Option Compare,

Option Strict;

* операторы **Imports…**, импортирующие в программу компоненты библиотек, определяемые в импортируемом пространстве;
* модули форм:

Public Class Имя\_Формы

…

End Class;

* модули классов:

Public Class Имя\_класса

…

End Class;

операторы описания типов, данных с ключевыми словами **Dim, Private и**

**Public,** находящиеся внутри модулей, классов и процедур;

процедуры, генерируемые средой разработки внутри модулей формы;

процедуры, написанные программистом, которые могут находиться

как внутри модулей формы, так и внутри стандартных модулей (общие

процедуры).

Таким образом, программный код VBхранится в программных моду

лях в виде процедур (методов). Эти программные модули могут быть

двух видов:

* модуль формы;
* модуль класса.

***При рассмотрении базовых средств программирования VB будут использоваться только* Приложения Windows Forms*, которые состоят из одной формы, и содержат, как правило, только один модуль – модуль формы.***

В свою очередь модуль формы может содержать ***процедуры событий***, ***процедуры пользователей*** и ***операторы описания данных*** (простых переменных, массивов и констант) с ключевыми словами **Dim, Private** и **Public**.

Таким образом, программный код простого приложения будет иметь структуру, приведенную на рис. 4.3.1-1.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | **Option Explicit On**  **Option Strict On**  **Option Strict On**  **Imports System.Math**  **Imports** |   **Private Sub** **Form1**   |  | | --- | | *Описание глобальных данных* |  |  | | --- | | *Процедура пользователя - 1*  *(Общая процедура - 1)* |   ***…***   |  | | --- | | *Процедура пользователя - n*  *(Общая процедура – n)* |  |  | | --- | | *Процедура обработки событий - 1* |   **…**   |  | | --- | | *Процедура обработки событий - m* |   **End Class** |

*Рис. 4.3.1-1. Структура простейшего приложения*

В дальнейшем, по мере усложнения приложения в нем могут создаваться модули классов, стандартные модули, а если программа достаточно большая, в ней будет несколько форм и большое число процедур событий, которые используют одни и те же переменные и процедуры. Причем некоторый программный код можно выделить в отдельный программный код, который будет являться общим для всех. Такой программный код может быть частью ***стандартного модуля***. Кроме того, ***стандартные модули***, так же, как и ***модули форм***, могут содержать объявления переменных, констант и типов данных, описания общих процедур и процедур обработки событий.

По умолчанию переменные ***локальны*** для любой процедуры – они могут быть использованы и изменены только в той процедуре, в которой они ***созданы (объявлены)***. Также можно объявить переменные в верхней части программного кода формы, до описания процедур пользователей и процедур событий, и тем самым при­дать переменным более широкую ***область видимости***, распространяющу­юся на всю форму. Однако, при создании в проекте нескольких форм, переменные, объявленные в верхней части формы, будут доступны толь­ко в той форме, в которой они объявлены. Точно также, процедуры собы­тий по умолчанию объявляются как *частные* (**Private*)*** и являются локальными для той формы, в которой они созданы. Например, нельзя из второй формы с именем **Form2**вызвать процедуру события **Button1\_Click(…)**, если она объявлена в **Form1**как частная.

Чтобы сделать переменные и процедуры доступными для всех форм и процедур событий проекта, их следует объявить в одном или несколь­ких стандартных (общих) модулях, включенных в проект.

Изучение общих модулей и общих процедур пока отложим, а подробно изучим структуру и содержание модуля формы, а также правила написания процедур пользователя и другие средства программирования VB.

Объявление ***глобальной***, или ***открытой*** переменной в модуле формы достаточно просто – необходимо перед заголовками процедуры (сразу после **Public Class Form1**) ввести ключевое слово **Public** или **Dim**, за которыми следует ***имя\_переменной*** и объявление ее типа. После того, как переменная будет объявлена, использовать ее содержимое, изменять его или отображать на экране можно в любой процедуре программного кода модуля формы.

Например, операторы объявления, приведенные ниже, объявляют открытую переменную с именем **Total** типа **Integer**:

|  |
| --- |
| **Public Total As Integer**  **Dim Total As Integer** |

***Обратите внимание на то, что в соответствии с технологией ООП все переменные, используемые внутри процедур, являются локальными, а число глобальных переменных должно быть минимально необходимо.***

### 4.3.2. Создание и использование процедур пользователей

Ранее мы познакомились со «***встроенными***» (т.е. готовыми к употреблению) процедурами (например, **Sqrt()**), которые в VB являются методами класса. Мы также познакомились с процедурами обработки событий, которые являются частью пользовательского интерфейса и которые выполняются, только если происходит некоторое событие. В отличие от «***встроенных процедур***», эти процедуры разрабатывает программист (разработчик проекта).

Что же собой представляют процедуры? Процедуры представляют собой отдельные блоки, из которых складывается программный код любого модуля проекта, причем каждая процедура должна выполнять какую-то законченную задачу (функцию).

Кроме процедур обработки событий, в программный код модуля можно включить процедуры, не связанные с событиями. Они выполняют отдельные действия (функции) и могут быть использованы (вызваны) неоднократно, но с различными параметрами. Как известно, они называются общими или методами класса. Процедуры (методы) общего назначения вызываются на выполнение из программного кода другой процедуры или класса. Использование процедур экономит время разработки приложения и позволяет избежать лишних ошибок.

Процедуры могут быть двух видов: **процедуры-подпрограммы** и **процедуры-фун**кции. Процедуры-функции отличаются от процедур-подпрограмм тем, что возвращают какое-то одно значение и поэтому могут участвовать в выражениях. Причем под подпрограммой или функцией понимается последовательность операций, которую нужно многократно выполнять в различных местах приложения. При этом требуемый блок команд описывается в программном коде только один раз, после чего к нему можно обращаться из любой части программы и неоднократно.

Процедуры, которые будут рассмотрены ниже, не относятся ни к встроенным процедурам, ни к процедурам обработки событий – это общие процедуры.

Общие процедуры, в отличие от процедур обработки событий, начинают выполняться не в ответ на какое-либо событие, а после их явного вызова в программном коде. После выполнения такой процедуры происходит автоматический возврат в то место программного кода, откуда процедура была вызвана.

**Использование процедур предполагает, что они должны обязательно быть определены (описаны) и вызваны (должно быть к ним обращение по имени).** Таким образом, каждой процедуре должно быть присвоено уникальное имя. Кроме того, для вызываемой процедуры может быть установлен перечень входных и выходных параметров.

**Входные параметры** процедуры – это переменные, значения которых при вызове этой процедуры должны быть определены (присвоены допустимые значения) и, которые участвуют в вычислительном процессе (реализации) процедуры. При описании процедуры перед ними ставиться ключевое слово **ByVal**.

В общем случае процедуры являются способом объединения набора связанных между собой элементов программного кода для выполнением конкретной задачи (алгоритма). В VBимеются два основных типа процедур:

1. Процедуры типа **Function** **– процедура-функци**я или **процедура-Function** вызываются из процедур со­бытий или других процедур по имени. **Процедуры**-**Function** часто используются для вычислений различных выражений, могут получать входные параметры, и предназначены для возврата одного значения через свое имя**.**
2. Процедуры типа **Sub** **– процедура–подпрограмма** или **процедура**-**Sub** вызываются из процедур со­бытий или других процедур по имени. Они могут получать входные параметры и возвращать выходные параметры, значения которых получены в результате работы процедуры. Однако, в отличие от **процедуры**-**Function, процедуры**-**Sub** не возвращают значения, свя­занного с их именами.

Процедуры-**Function** и процедуры-**Sub** должны быть предварительно определены (описаны) в коде программы формы до их вызова.

Использование процедур общего назначения предоставляет следующие преимущества:

* позволяет связать часто используемую группу операторов программ кода со знакомым именем;
* устраняет повторы фрагментов программного кода, т.е. можно один раз определить процедуру и вызывать (обращаться) любое количество раз;
* делает программный код более простым и легко читаемым, так как, разделенный на небольшие части он, легче воспринимается, чем оформленный в виде одного большого фрагмента;
* упрощает разработку проектов, так как проект, разделенный на логические (функциональные) единицы, легче разрабатывать и отлаживать; кроме того, если программный проект разрабатывается группой разработчиков, то можно обмениваться готовыми процедурами и модулями;
* позволяет повторно использовать процедуры в других проектах и решениях (можно легко встроить процедуры из стандартного модуля в другие программные проекты);
* расширяет язык VB (процедуры часто могут выполнять задачи, которые не могут быть выполнены отдельными операторами VB).

***Описание*** *процедуры****-*Function**– это группа операторов, расположенных между ключевыми словами **Function** и **End Function***.* Операторы в *процедуре****-*Function** выполняют осмысленную работу – в соответствии с заданным алгоритмом. Они используются только тогда, когда результатом выполнения процедуры является значение, какой-либо одной величины числового, строкового или логического типа. Чтобы ***выполнить*** (***вызвать***) – функцию, необходимо поместить в любое допустимое выражение программного кода имя этой функции и все требуемые для нее **фактические** параметры.

***Аргументы (формальные параметры)*** процедуры**-Function** – это переменные, используемые для работы функции, и они должны быть заключены в круглые скобки после имени и разделены запятыми. В целом использование процедуры-**Function** в точности совпадает с использованием встроенных функций. Функции, объявленные (описанные) в модуле формы, по умолчанию являются открытыми в пределах данной формы. В результате они могут использоваться (вызываться) из лю­бой процедуры события проекта или из другой процедуры модуля формы.

***Описание*** *процедуры****-функции*** имеет следующий синтаксис:

|  |
| --- |
| **Function** *ИмяФункции (ФормальныеПараметры)* **As** *ТипВозЗначения*  …  *ОператорыФункции*  …  **Return** *значение или ИмяФункции = значение*  **…**  **End Function** |

Здесь важными являются следующие синтаксические элементы: *ИмяФункции* **–** это уникальное имя создаваемой функции**;**

* *ТипВозЗначения* **–** это тип значения, возвращаемого функцией, то есть тип величины, которая является результатом работы функции**;**
* *ФормальныеПараметры (аргументы)* **–** это список необязательных аргументов, разделенных запятыми и используемых в данной функции. Каждый аргумент дол­жен быть объявлен с указанием конкретного типа данных и типа возвращаемого значения. По умолчанию VB добавляет к каждому аргументу ключевое слово **ByVal**, которое указывает на то, что в функцию через данный аргу­мент передается *копия значения фактического параметра*, и все изменения значения этого ар­гумента не будут возвращены в вызывающую процедуру – *передача параметров по значению*;

ОператорыФункции **–** это блок операторов, который выполняет работу функции;

**Return** **–** это оператор, с помощью которого можно указать место, где в блоке кода функции требу­ется возвратить значение в вызывающую программу, и каково это возвращаемое значение. После выполнения этого оператора происходит выход из процедуры-функции и управление передается в то место программы, откуда эта процедура-функция была вызвана.

Процедуры-**Sub** похожи на процедуры-**Function**, за исключе­нием того, что процедура-**Sub** не возвращает значения, связанного с ее именем.

Базовый синтаксис ***описания процедуры* Sub** имеет вид:

|  |
| --- |
| **Sub** *ИмяПроцедуры (ФормальныеПараметры***)**  **…**  *ОператорыПроцедуры*  **…**  **End Sub** |

Отметим следующие синтаксические элементы:

* *ИмяПроцедуры*– это имя создаваемой процедуры-**Sub***.*
* *Формальные параметры (аргументы)**–* это список необязательных аргументов, разделенных запятыми, если их больше одного***.*** Причем *к*аждый аргумент должен быть объявлен с указанием конкретного типа данных и типа возвращаемого значения. Формальные параметры в процедуре-**Sub**  могут быть как ***входные*** так и ***выходные***. По умолчанию VS добавляет к каждому ар­гументу ключевое слово **ByVal**, которое указывает на то, что в процедуру-подпрограмму через данный аргумент передается копия значения, и все изме­нения значения этого аргумента не будут возвращены в вызывающий код. Кроме параметров типа **ByVal** в процедуре-**Sub** могут параметры типа **ByRef.**
* *ОператорыПроцедуры*– это блок операторов, который выполняет работу процедуры по заданному алгоритму.

При вызове процедуры-**Sub** количество и типы аргументов, переда­ваемых в процедуру (они называются **фактическими** параметрами), должны совпадать с количеством и типом аргумен­тов, указанных при объявлении (описании) процедуры (это **формальные** параметры), и вся группа аргументов должна быть заключена в круглые скобки. Таким образом, количество и типы соответствующих друг другу формальных и фактических параметров должны совпадать. Если переменные, передава­емые в процедуру-**Sub**, изменяют свое значение при ее выполнении, то значения этих переменных не возвращаются в вызываемую процедуру, если эти аргу­менты не объявлены с помощью ключевого слова **ByRef**. По умолчанию процедуры-**Sub** в модуле объявляются как открытые, так что они могут вызываться лю­бой процедурой проекта.

### 4.3.3. Передача параметров по значению (ByVal) и по ссылке (ByRef)

Итак, каждой процедуре присваивается уникальное название – имя процедуры. Кроме того, для процедуры четко устанавливается перечень входных и выходных аргументов (входных и выходных параметров). Прежде чем определять процедуру, необходимо четко представить, чтопроцедура делает, что она должна получать (что у нее на входе) и какие результаты возвращать (что на выходе). Параметры, перечисляемые в скобках при описании процедуры, называются **формальными** параметрами (перед такими параметрами обязательно ставится клю­чевое слово **ByVal** или **ByRef** (тип вызова данного параметра) и после имени параметра указывается его тип значений данного параметра). Параметры, которые указываются при вызове процедуры, называются **фактическими**. Передача параметров в процедуру из вызывающей процедуры может осуществляться двумя способами: по значению (**By Value**) или по ссылке (**By Reference**). Обычно по значению передаются входные параметры, а по ссылке – выходные.

**Входной параметр процедуры** – это переменная, значение которой при вызове процедуры должно быть установлено до начала работы процедуры и которая участвует в работе процедуры. Перед таким параметром ставится клю­чевое слово **ByVal** (передача по значению). В этом случае в процедуру передается не сама переменная, а ее копия. В момент вызова процедуры для хранения формального параметра в ней создается локальная переменная, которая получает копию значения фактического параметра, а после окончания работы процедуры эта локальная переменная уничтожается. Поэтому изменение формального параметра в процедуре не затрагивает фактический параметр, так как изменяется не сама переменная, а ее копия и таким образом все изменения, сделанные в переменной, переданной по значению, не передаются об­ратно в вызывающую процедуру.

***Выходной параметр процедуры***– это переменная, которая получает свое значение в результате работы процедуры. Использование ключевого слова **ByRef** указывает, что переменные должны быть переданы в процедуру по ссылке. Это значит, что процедура не создает локальную переменную под формальный параметр, а получает доступ к области памяти, в которой хранится фактический параметр, т.е. получает его адрес. Поэтому все изменения, сделанные в рамках процедуры, будут производиться прямо на тех же ячейках памяти, в которых хранится фактический параметр, и следовательно переданы обратно в вызывающий код.

При выборе способа передачи параметра (по ссылке или по значению) решающий критерий – должен ли параметр изменять свое значение в процедуре.

Вот некоторые рекомендации о том, когда следует использовать **ByVal**, а когда - **ByRef**:

* используйте **ByVa**l тогда, когда требуется, чтобы процедура не изме­няла передаваемую в нее через аргумент переменную;
* используйте **ByRef** тогда, когда требуется позволить процедуре из­менять передаваемую в нее переменную;
* когда сомневаетесь, используйте ключевое слово **ByVal**, так как в этом случае фактический параметр гарантированно не будет изменен в ходе выполнения процедуры.

Однако так как при передаче по значению происходит копирование объекта, данные большого объема (например, структуры или многомерные массивы) более эффективно передавать по ссылке.

Структуру программы, оператор присваивания и выражения мы уже разобрали. Для реализации алгоритмов линейной структуры необходимо знать, как может быть реализован в VB ввод исходных данных и вывод результатов.

### 4.3.4. Средства программирования алгоритмов линейной структуры и процедуры ввода и вывода

**Последовательными** называются такие алгоритмические струк­туры, в которых функциональные блоки вы­полняются в том порядке, в котором они записаны или изо­бражены на схеме алгоритма.

|  |  |
| --- | --- |
|  | P1, P2, Pn – выражения,  операторы,  процедуры |

Рис. 4.3.4-1. Схема алгоритма линейной структуры

Такая структура может быть составлена из совокупности программных блоков «**Пуск/останов**», «**Ввод/вывод**», а также блоков «**Процесс**» (рис. В.3-2). Cхема алгоритма линейной структуры приведена на рис. 4.3.4-1.

Обычно для ***ввода информации*** в программном коде проекта программист может использовать ***объект текстового поля*** формы **TextBox** и его свойство Text.

Поскольку технология современного программирования предполагает ***максимальное использование процедур при разработке программных проектов***, в своих работах для ввода исходных данных, например, типа **Double** можно использовать следующую процедуру-**Function**:

|  |
| --- |
| *'процедура-Function ввода данных типа Double*  *'из текстового поля TextBox*  **Function vvodDbl1(ByVal T As TextBox) As Double**  **Return (Val(T.Text))**  **End Function** |

Вызов (обращение) такой процедуры, например, для ввода значения переменной **x1** типа **Double** из текстового поля **TextBox1**формы можно осуществлять следующим образом:

|  |
| --- |
| **Dim x1 As Double**  **x1 = vvodDbl1(TextBox1)** |

Вторым способом ввода информации является использование библиотечной функции **InputBox( )** VBдля отображения на экране диалогового окна, и последующее использо­вание переменной для хранения текста, введенного пользователем.

Функция **InputBox( )** вызывает окно **InputBox,** которое состоит из четырех элементов (рис. 4.3.4-2):

*Окно ввода* – это стандартное диалоговое окно, которое появляется на экране в результате вызова функции InputBox(). Значение функции   
InputBox() – строка, которую ввёл пользователь. Поэтому, если программе надо получить число, то введённая строка должна быть преобразована в число при помощи соответствующей функции преобразования. Стандартное окно InputBox для ввода представлено на рис. 4.3.4-2.

|  |  |
| --- | --- |
|  |  |

Рис.4.3.4-2. Стандартное окно для ввода данных

Синтаксис функции InputBox() ввода данных выглядит так:

|  |
| --- |
| *Переменная* = InputBox ("*Приглашение*", "*Заголовок*") |

Где,

* *Приглашение* – это любой текст, который должен находиться в *Окне ввода*. Его назначение – служить *подсказкой* пользователю, какую информацию он должен ввести в специальное *поле ввода,* находящееся в этом окне;
* *Заголовок* – *это надпись в строке заголовка* *Окна ввода.*

Например,

|  |
| --- |
| Name = InputBox("Введите, пожалуйста, первое число", "Ввод числа") |

Вывод в окно сообщения **(**рис. 4.3.4-3**)** можно осуществить с помощью функции **MsgBox()**. Она позволяет поместить в окно с сообщением один из стандартных значков, например, **"**Внимание**"**, задать количество и тип командных кнопок и определить, какую из кнопок нажал пользователь.

Синтаксис функции MsgBox() выглядит так:

|  |
| --- |
| В = MsgBox ("*Сообщение*"*, НомерКнопки,* "*Заголовок*") |

Где,

* *Сообщение* – представляет собой текст, который будет отображен на экране;
* *НомерКнопки* – указывает тип отображаемых кнопок, тип используемой пиктограммы;
* *Заголовок* – это надпись в строке заголовка *Окна вывода*.

Возможные установки для *НомерКнопки* представлены в табл. 4.3.4-1.

Таблица 4.3.4-1

|  |  |
| --- | --- |
| **Константа** | **Описание** |
| **MsgBoxStyle**.Ok | Отображает только кнопку **Ок** |
| **MsgBoxStyle**.OkCancel | Отображает кнопки **Ок** и **Отмена** |
| **MsgBoxStyle**.AbortRetryIgnore | Отображает кнопки **Стоп,Повтор,Пропустить** |
| **MsgBoxStyle**.YesNoCancel | Отображает кнопки **Да, Нет, Отмена** |
| **MsgBoxStyle**.YesNo | Отображает кнопки **Да, Нет** |
| **MsgBoxStyle**.RetryCancel | Отображает кнопки **Повтор** и **Отмена** |
| **MsgBoxStyle**.Critical | Отображает запрещающую пиктограмму |
| **MsgBoxStyle**.Question | Отображает предупреждающую пиктограмму (знак вопроса на белом фоне) |
| **MsgBoxStyle**.Exclamation | Отображает предупреждающую пиктограмму (знак вопроса в красном треугольнике) |
| **MsgBoxStyle**.Information | Отображает информационную пиктограмму |

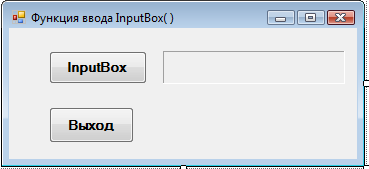
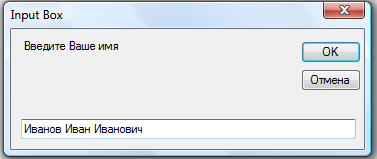


Рис.4.3.4-3. Форма для ввода и вывода данных

**Пример 4.3.4-1. Ввод данных с помощью функции InputBox( ).**

1. Создать проект c именем **Пример 4.3.4-1**.
2. В IDE откроется пустая форма проекта **Пример 4.3.4-1**.
3. Создать форму, которая содержит одну надпись (**Label**) и две кнопки(**Button**). Для получения входных данных от пользователя использовать функцию **InputBox()**. За­тем введенное значение необходимо отобразить в надписи формы (рис. 4.3.4-4).



ис. 4.3.4-4. Окно ввода функции **InputBox()**

1. Дважды щелкнуть мышью на кнопке **InputBox**.
2. В окне **Редактора код**а появится процедура события

Button1\_Сliск(…).

1. Чтобы объявить две переменные и использовать функцию **InputBox()**, необходимо ввести операторы программы, показанные на рис. 4.3.4-5.

|  |
| --- |
| **Private Sub Button1\_Click(ByVal sender As System.Object, \_**  **ByVal e As System.EventArgs)Handles Button1.Click**  **Dim Prompt, FullName As String**  **Prompt = "Введите Ваше имя"**  **FullName = InputBox(Prompt)**  **MsgBox(FullName, , "Введено имя")**  **Label1.Text = FullName**  **End Sub** |

*Рис. 4.3.4-5. Программный код процедуры события*

*Проекта* ***Пример 4.3.4-1***

С помощью оператора **Dim** производится объявление сра­зу двух переменных: **Prompt** и **FullName**. Обе переменные объявля­ются с использованием типа **String**. (Вы можете объявить в одной стро­ке столько переменных, сколько хотите, но при условии, что они имеют один и тот же тип.)

Вторая строка процедуры события присваивает переменной **Prompt** текстовую строку. Это сообщение будет использовано как текстовый параметр для функции **InputBox( )**. Следующая строка вызывает функцию **InputBox( )** и присваивает результат этого вызова (текстовую строку, введенную пользователем) переменной **FullName. InputBox()** – это функция VB, которая отобра­жает на экране диалоговое окно и запрашивает пользовательский ввод.

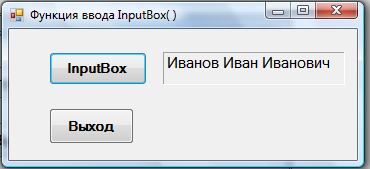
После того как **InputBox()** возвратит в программу текстовую строку, чет­вертый оператор этой процедуры помещает имя пользователя в свой­ство Text объекта **Label1** и, таким образом, отображает его в форме.

1. Сохранить изменения.
2. Запустить программу на выполнения.
3. Щелкнуть на кнопке **InputBox** формы.

VB начнет выполнять процедуру события **Button1\_Click(…)**, и на

экране появится диалоговое окно **InputBox**.

1. Ввести свое полное имя, а затем щелкнуть на ОК (рис. 4.3.4-6).



*Рис. 4.3.4-6. Результат выполнения проекта* **Пример 4.3.4-1**

Функция **InputBox()** возвращает введенное имя в программу и поме­щает его в переменную **FullName**. Затем программа использует эту переменную для отображения этого имени на форме.

Функцию **InputBox()** необходимо использовать в своих программах всегда, когда требуется запросить у пользователя какую-либо информацию. Вы мо­жете использовать эту функцию совместно с другими элементами управления ввода и управлять потоком данных, поступающих в программу и исходящих из нее.

1. Чтобы завершить выполнение программы, щелкнуть на кнопке   
    **Выход**.

Выполнение проекта остановится.

1. Процедура-**Function vvodDbl2()**, которая использует **InputBox()**,

может быть реализована следующим образом:

|  |
| --- |
| *'процедура-Function ввода данных типа Double*  **Function vvodDbl2(ByVal p As String, ByVal T As TextBox) As Double**  **T.Text = InputBox(P)**  **Return CDbl(Val(T.T**e**xt))**  **End Function** |

Вызов такой процедуры можно осуществлять, например, для ввода значения переменной **x1**  типа **Double.**

|  |
| --- |
| **Dim x1 As Double**  **x1 = vvodDbl2("Введите имя", TextBox1)** |

**Для вывода информации**, во-первых, необходимо отобразить содержимое переменной на форме, что можно сделать, присвоив значение этой пе­ременной свойству (например, свойству Textобъекта элемента управления формы), или пере­дав переменную как аргумент в функцию диалогового окна.

|  |
| --- |
| *'Процедура вывода данных типа Double в TextBox*  **Sub vivodDbl1(ByVal Z As Double, ByVal T As TextBox)**  **T.Text = CStr(Z)**  **End Sub** |

Вызов такой процедуры можно осуществлять, например, для вывода значения переменной **x1** типа **Double** в текстовое поле**:**

|  |
| --- |
| **Dim x1 As Double**  **vivodDbl1(x1, TextBox1)** |

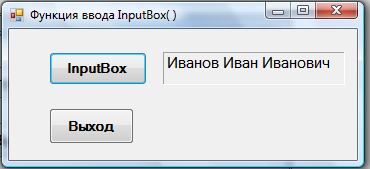
**Пример 4.3.4-2. Вывод сообщения с помощью функции MsgBox()*.***

1. Если окно **Редактор кода** из предыдущего примера не отображается в IDE, следует сделать двойной щелчок мышью на кнопке **InputBox** формы на рис. 4.3.4-4.
2. В окне **Редактор кода** появится процедура события **Button1\_Click(…)** (это код, который был введен в предыдущем примере).
3. Выделить в процедуре события оператор (последняя строка):
4. Label1.Text = FullName.
5. Это оператор отображает в надписи содержимое перемен­ной **FullName**.
6. Нажать клавишу <Delete**>**, чтобы удалить эту строку.
7. Оператор будет удален.
8. Вместо удаленной строки в процедуре события необходимо ввести следующую строку**: MsgBox(FullName, , "Введено имя")**.
9. Этот новый оператор вызывает функцию **MsgBox()**, отображает в диа­логовом окне содержимое переменной **FullName** и помещает в его стро­ку заголовка слова Введено имя (рис. 4.3.4-7).

|  |
| --- |
| **Public Class Form1**  **Private Sub Button2\_Click(ByVal sender As System.Object, \_**  ***ByVal e As System.EventArgs) Handles* Button2.Click**  **End**  **End Sub**  **Private Sub Button1\_Click(ByVal sender As System.Object, \_**  **ByVal e As System.EventArgs) Handles Button1.Click**  **Dim Prompt, FullName As String**  **Prompt = "Введите Ваше имя"**  **FullName = InputBox(Prompt)**  **MsgBox(FullName, , "Введено имя")** *'Label1.Text = FullName*  **End Sub**  **End Class** |

*Рис. 4.3.4-7. Программный код проекта* **Пример 3.4-2**

1. Щелкнуть на кнопке ***Начать отладку*** на панели инструментов.
2. Щелкнуть на кнопке ***InputBox*** формы, а затем ввести в поле ввода имя и щелкнуть на ОК***.***
3. VB сохранит введенное значение в переменной **FullName**, а за­тем отобразит его в окне сообщения. Форма будет выглядеть, как на   
   рис. 4.3.4-8.



*Рис. 4.3.4-8. Результат работы проекта* **Пример 3.4-2**

1. Чтобы закрыть окно сообщения, щелкнуть на ОК. Затем выполнить команду **Остановить отладку** элемента **Отладка** основного меню, чтобы закрыть проект.
2. Процедура-**Sub vivodDbl2()**, использующая для вывода **MsgBox()** может быть реализована следующим образом:

|  |
| --- |
| *'Процедура вывода данных типа Double функцией MsgBox*  **Sub vivodDbl2(ByVal Z As Double, \_**  **ByVal S As String, ByVal T As TextBox)**  **T.Text = MsgBox(CStr(Z), S)**  **End Sub** |

Вызов такой процедуры можно осуществлять, например, для вывода

значения переменной **x1** типа **Double** в текстовое поле:

|  |
| --- |
| **Dim z As Double**  **vivodDbl2(z, "Вывод значения z", TextBox1)** |

**Пример 4.3.4-3. Вычисление периметра и площади треугольника с использованием процедур для ввода исходных данных и вывода результатов.**

Программный код **Проекта 3.4-3** представлен на рис. 4.3.4-9.

|  |
| --- |
| **Option Strict On**  **Option Explicit On**  **Imports System.Math**  **Public Class Form1**  **Function vvodSng3(ByVal T As TextBox) As Single**  **Return CSng(Val(T.Text))**  **End Function**  **Sub vivodSng3(ByVal Z As Single, ByVal T As TextBox)**  **T.Text = CStr(Z)**  **End Sub**  **Sub PS(ByVal A As Single, ByVal B As Single, \_**  **ByVal C As Single, ByRef P As Single, \_**  **ByRef S As Single)**  **Dim Pp As Single**  **P = (A + B + C) : Pp = P / 2**  **S = CSng(Sqrt(Pp \* (Pp - A) \* (Pp - B) \* (Pp - C)))**  **End Sub**    **Sub PSPr(ByVal A As Single, ByVal B As Single, \_**  **ByVal C As Single)**  **Dim P As Single, S As Single**  **PS(A, B, C, P, S)**  **vivodSng3(P, TextBox4) : vivodSng3(S, TextBox5)**  **End Sub**  **Private Sub Button1\_Click(ByVal sender As System.Object,\_**  **ByVal e As System.EventArgs) \_   Handles Button1.Click**  **Dim A, B, C As Single**  **A = vvodSng3(TextBox1)**  **B = vvodSng3(TextBox2)**  **C = vvodSng3(TextBox3)**  **PSPr(A, B, C)**  **End Sub**  **Private Sub Button2\_Click(ByVal sender As System.Object,\_**  **ByVal e As System.EventArgs) \_**  **Handles Button2.Click**  **End**  **End Sub**  **End Class** |

*Рис. 4.3.4-9. Программный код проекта* **Проекта 3.4-3**

Пример 4.3.4-4. Вычислить значения арифметических выражений и вывести на экран результаты вычислений.

**Исходные данные: x=1,4444 b=0,318 t=2,1 a=1,3**

****

Выполним запись арифметических выражений на языке VB:

|  |
| --- |
|  |

Программный код показан на рис. 4.3.4-10.

|  |
| --- |
| **Option Strict On**  **Option Explicit On**  **Imports System.Math**  **Public Class Form1**  **Sub Pr348(ByVal x As Single,ByVal a As Single,ByVal b As Single,\_**  **ByVal t As Single, ByRef y As Single, ByRef z As Single)**  **y = CSng(9 \* x ^ 2 + Sin(x) \* Sin(x) \* Sqrt(a + b))**  **z = CSng((Sqrt(x^t))^(1 / 3) \* (a \* x ^ 3 - (x \* x) / (1/2)))**  **End Sub**  **Private Sub Button1\_Click(sender As Object, e As EventArgs) \_**  **Handles Button1.Click**  **Dim a, b, x, t, z, y As Single**  **a = CSng(Val(TextBox1.Text)) : b = CSng(Val(TextBox2.Text))**  **x = CSng(Val(TextBox4.3.Text)) : t = CSng(Val(TextBox4.Text))**  **Pr344(x, a, b, t, y, z)**  **TextBox5.Text = Str(y)**  **TextBox6.Text = Str(z)**  **End Sub**  End Class |

*Рис. 4.3.4-10. Программный код* **Пример 3.4-4**

Результат программы представлен на рис. 4.3.4-11.

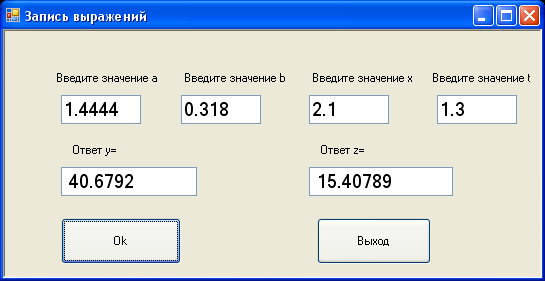


Рис. 4.3.4-11. Результат программы **Пример 3.4-4**

Выполнить ту же задачу с использованием функций **InputBox()** и   
**MsgBox( )** (Рис. 4.3.4-12).

|  |
| --- |
| **Option Strict On**  **Option Explicit On**  **Imports System.Math**  **Public Class Form1**  **Sub Pr3410(ByVal x As Single,ByVal a As Single, \_**  **ByVal b As Single, ByVal t As Single, \_**  **ByRef y As Single, ByRef z As Single)**  **y = CSng(9 \* x ^ 2 + Sin(x) \* Sin(x) \* Sqrt(a + b))**  **z = CSng((Sqrt(x^t))^(1 / 3) \* (a \* x^3 - (x \* x) / (1 \* 2)))**  **End Sub**  **Private Sub Button1\_Click(sender As Object, e As EventArgs) \_**  **Handles Button1.Click**  **Dim a, b, x, t, z, y As Single**  **a = CSng(InputBox("Введите значение А", "Ввод 1 числа"))**  **b = CSng(InputBox("Введите значение В", "Ввод 2 числа"))**  **x = CSng(InputBox("Введите значение X", "Ввод 3 числа"))**  **t = CSng(InputBox("Введите значение T", "Ввод 4 числа"))**  **Pr3410(x, a, b, t, y, z)**  **MsgBox("Значение Y = " & y, vbInformation, "Получаем ответ")**  **MsgBox("Значение Z = " & z, vbInformation, "Получаем ответ")**  **End Sub**  **End Class** |

Рис.4.3.4-12. Программный код Вычисления арифметических выражений использованием функций **InputBox()** и **MsgBox( )**

Результаты работы программы представлены на рис. 4.3.4-113,   
4.3.4-14.

|  |  |  |
| --- | --- | --- |
|  |  | |
|  |  |

Рис. 4.3.4-13 Вывод значения Y Рис. 4.3.4-14. Полученное значение Z

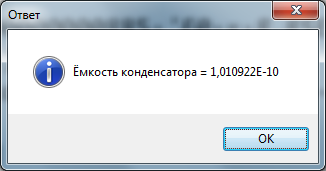
**Пример 4.3.4-5. Найти ёмкость конденсатора с, если известны: площадь пластин s и расстояние между ними L. Следует учесть, что в конденсатор вставлена пластина толщиной d, параллельная его обкладкам.**

|  |  |
| --- | --- |
|  | **С – емкость конденсатора.**  **E0 – электрическая постоянная =** 8,85.10-12 |

Код программы представлен на рис. 4.3.4-15.

|  |
| --- |
| **Option Strict On**  **Option Explicit On**  **Imports System.Math**  **Public Class Form1**  **Private Sub Pr3413(ByRef C As Single)**  **Dim s, L, d, E0 As Single**  **s = CSng(InputBox("Введите площади пластин", \_**  **"Ввод площади пластин"))**  **L = CSng(InputBox("Введите расстояние между пластинами", \_**  **"Ввод расстояния между пластинами"))¬**  **d = CSng(InputBox("Введите толщину пластины", \_**  **"Ввод толщины пластин"))**  **E0 = 0.00000000000885 'E0 = 8.85E-12**  **C = (E0 \* s) / (L - d)**  **End Sub**  **Private Sub Button1\_Click(ByVal sender As System.Object, \_**  **ByVal e As System.EventArgs) Handles Button1.Click**  **Dim A, B, C As Single**  **Pr3413(C)**  **MsgBox("Ёмкость = " & C, vbInformation, "Ответ")**  **End Sub**  **End Class** |

*Рис. 4.3.4-15. Программный код* **Примера 3.4-5**



*Рис. 4.3.4-16. Результат* **Примера 3.4-5**

При вводе исходных данных s=5,7, L=0,5 и d=0,001, емкость конденсатора c=1,010922E-10, что показано на рис. 4.3.4-14.

### 4.3.5. Перегрузка методов (процедур)

***Перегрузка*** – это создание более чем одной процедуры в модуле класса с одинаковыми именами, но с различными типами и количеством аргументов.

***Перегрузка*** методов представляет собой один из частных случаев **полиморфизма**. Напомним, что ***полиморфизм*** в переводе с греческого языка означает «много форм». В отношении к методам класса полиморфизм позволяет с помощью одного имени представлять различный программный код.

Перегрузка особенно полезна, когда в программном коде нужно использовать одинаковые имена процедур, работающих с различными типами данных и различным числом параметров. Например, в примерах из предыдущего параграфа для вывода данных были использованы процедуры:

|  |
| --- |
| *'Процедура вывода данных типа Double в TextBox*  **Sub vivodDbl1(ByVal Z As Double, ByVal T As TextBox)**  **T.Text = CStr(Z)**  **End Sub** |

|  |
| --- |
| **Sub vivodSng3(ByVal Z As Single, ByVal T As TextBox)**  **T.Text = CStr(Z)**  **End Sub** |

Для вызова этих процедур использовались следующие операторы:

|  |
| --- |
| **vivodDbl1(x1, TextBox1)**  **vivodSng3(S, TextBox5)** |

При описании эти методы можно перегрузить, то есть присвоить им одно имя, Для этого необходимо использовать модификатор процедуры **Overloads** перед описанием процедур. Например,

|  |
| --- |
| *'Процедура вывода данных типа Double в TextBox*  **Overloads Sub vivod (ByVal Z As Double, ByVal T As TextBox)**  **T.Text = CStr(Z)**  **End Sub** |

|  |
| --- |
| **Overloads Sub vivod (ByVal Z As Single, ByVal T As TextBox)**  **T.Text = CStr(Z)**  **End Sub** |

Для вызова этих процедур использовались следующие операторы:

|  |
| --- |
| **vivod(x1, TextBox1)**  **vivod(S, TextBox5)** |

Перегруженный элемент класса создается путем добавления двух или более методов с тем же именем. Все перегруженные элементы должны иметь разные списки параметров. При перегрузке процедуры следующие элементы не могут быть использованы как средства определения:

* модификаторы, такие как **ByVal** или **ByRef**, которые применимы к элементу, или параметрам элемента;
* имена параметров;
* возвращаемые типы процедур.

Ключевое слово **Overloads** не обязательно при перегрузке, но, если некоторый перегруженный элемент использует ключевое слово **Overloads**, то все остальные перегруженные элементы с этим же именем также должны быть определены этим ключевым словом.

Таким образом, при перегрузке процедур, ***сигнатура метода*** – это комбинация его имени, списка параметров и служебных слов.

### 4.3.6. Тестовые задания

1. Алгоритм называется линейным, если его выполнение предполагает многократное повторение одних и тех же действий
2. ход его выполнения зависит от истинности тех или иных условий
3. его команды выполняются в порядке их естественного следования независимо от каких-либо условий
4. он включает в себя вспомогательный алгоритм
5. Программа – это
6. протокол взаимодействия компонентов компьютерной сети
7. понятное и точное предписание человеку совершить последовательность

действий, направленных на достижение поставленных целей

1. ориентированный граф, указывающий порядок исполнения команд компьютера
2. алгоритм, записанный на языке программирования
3. набор команд операционной системы компьютера
4. Процедура – это
5. последовательность логически связанных фрагментов программы, оформленная

как отдельная часть программы специальным способом

1. последовательность отдельных фрагментов программы
2. последовательность только циклических фрагментов программы
3. последовательность только разветвляющихся фрагментов программы
4. Глобальные переменные – это
5. переменные, доступные во всех других процедурах и модулях без механизма

передачи параметров

1. все переменные одной отдельной процедуры
2. все переменные всех используемых процедур
3. все переменные, описанные в модуле
4. в списке нет правильного ответа
5. Локальные переменные – это
6. все переменные с одинаковыми именами во всех процедурах
7. переменные только одной процедуры
8. переменные всех процедур
9. переменные, область видимости которых процедура, в которой они описаны
10. Формальными параметрами называются
11. список передаваемых параметров, указанных в описании процедуры
12. список параметров, указанных в модуле программе
13. все переменные, используемые в процедуре
14. список передаваемых параметров, указанных при вызове процедуры
15. Формальные параметры являются для данной процедуры
16. глобальными
17. фактическими
18. локальными
19. формализованными
20. Формальные параметры определяют
21. имя и тип переменных
22. только имена переменных
23. только типы массивов
24. в списке нет правильного ответа
25. При вызове процедуры происходит замена формальных параметров
26. локальными
27. глобальными
28. статическими
29. фактическими
30. Фактические параметры – это
31. конкретные значения переменных, на которые заменяются формальные параметры при обращении к процедуре
32. глобальные переменные
33. локальные переменные
34. статические переменные
35. Параметры могут передаваться в процедуру
36. только по значению
37. по значению или по ссылке
38. только по ссылке
39. любым образом
40. При передаче параметра по значению в процедуру передается
41. копия передаваемого параметра
42. адрес передаваемого параметра
43. адрес и значение передаваемого параметра
44. По значению могут передаваться
45. входные параметры процедуры
46. выходные параметры процедуры
47. как входные, так и выходные параметры процедуры
48. только глобальные переменные
49. По ссылке могут передаваться
50. как входные, так и выходные параметры процедуры
51. только входные параметры процедуры
52. только выходные параметры процедуры
53. в списке нет правильного ответа
54. Изменение в процедуре значения параметра, передаваемого по значению
55. не приводит к изменению значения этого параметра в вызывающей процедуре
56. приводит к изменению значения этого параметра в вызывающей процедуре
57. приводит к изменению значений всех параметров вызывающей процедуры
58. в списке нет правильного ответа
59. К средствам процедурного программирования в VB относятся
60. процедуры-функции и процедуры-подпрограммы
61. процедуры-функции
62. процедуры-подпрограммы
63. процедуры-функции, процедуры-подпрограммы и событийные процедуры
64. Результатом выполнения функции является
65. скалярная величина
66. несколько скалярных величин
67. коллекция данных
68. коллекция данных и скалярные величины
69. Переменные, описанные внутри функций, являются
70. глобальными
71. локальными
72. статическими
73. фактическими
74. При вызове функций и процедур контролируется
75. количество и тип аргументов, их порядок следования
76. количество аргументов, их порядок следования
77. порядок следования аргументов
78. количество и тип аргументов
79. При вызове функций и процедур контролируется
80. количество и тип аргументов, их порядок следования
81. количество аргументов, их порядок следования
82. порядок следования аргументов
83. количество и тип аргументов
84. Передача фактических параметров-выражений в процедуру и в процедуру-функцию производится
85. по значению
86. по ссылке
87. как по ссылке, так и по значению
88. в процедуру - по ссылке, в процедуру-функцию – по значению

### 4.3.7. Лабораторная работа по теме *«Структура VB-программ и процедуры. Средства программирования алгоритмов линейной структуры»*

**Цель** выполнения данной лабораторной работы состоит в получении практических навыков разработки, написания и отладки последовательных программ с использованием процедур, а также в изучении средств отладки.

### Вопросы, подлежащие изучению

1. Средства процедурного программирования в VB.
2. Средства описания процедур в VB.
3. Преимущества программ, имеющих процедуры.
4. Организация обмена данными между процедурами.
5. Роль фактических и формальных параметров.
6. Способы передачи параметров по значению и адресу.
7. Отличия в передаче значения результата выполнения функций и процедур.
8. Средства реализации алгоритмов линейной структуры.
9. Особенности разработки блок-схем алгоритмов, использующих процедуры.

### Общее задание

1. ***Изучите структуру VB-программ, процедуры и средства программирования алгоритмов линейной структуры»***
2. **Первое задание**:
3. ***Выберите индивидуальный вариант*** *из табл. П.1.2-1.*
4. ***Разработайте три алгоритма для решения 1-го задания****, по которым будут реализованы три процедуры пользователя для вычисления заданного арифметического выражения: процедура-функция с формальными параметрами, процедура-подпрограмма с формальными параметрами и процедура без параметров (передача данных в соответствующем проекте будет осуществляться через глобальные объекты) и представьте их в виде схем алгоритмов.*
5. ***Разработайте три процедуры пользователя*** *по алгоритмам из п. 2.2 и создайте решение, содержащее три проекта, каждый из которых использует модуль формы, одну из разработанных в п.2.2 процедур пользователя, вычисляющую заданное арифметическое выражение, и событийную процедуру, в которой осуществляется ввод исходных данных, вызов соответствующей процедуры п.1.2 и вывод результатов.*
6. ***Выполните созданные проекты и получите результаты****.*
7. ***Выполните созданные проекты по шагам****, с помощью отладчика,*

*поставив точку останова после ввода данных.*

1. ***Докажите, что результат правильный****.*
2. **Второе задание**:
3. ***Выберите индивидуальный вариант задания*** *из таблицы 4.3.7-1.*
4. ***Запишите заданные или полученные в процессе формализации математические формулы*** *в виде арифметических выражений.*
5. ***Разработайте алгоритм решения заданной задачи*** *и*

*представьте его в виде схем алгоритмов.*

1. ***Создайте проект, использующий модуль формы, процедуры пользователя****, решающие предписанные задачи, процедуры ввода исходных данных и процедуру вывода результата. Обмен данными между процедурами должен осуществляться через параметры, без использования глобальных переменных. Событийная процедура должна содержать только операторы вызова пользовательских (общих) процедур.*
2. ***Подготовьте тесты*** *для контрольного решения задачи.*
3. ***Выполните*** *проект и* ***получите*** *результаты.*
4. ***Докажите,*** *что результат правильный.*

### Варианты индивидуальных заданий

Таблица 4.3.7-1

|  |  |
| --- | --- |
| **№** | **Задача** |
| **1)** | Определите периметры правильных n – угольников (10 – угольника, 50 – угольника, 100 – угольника), вписанных в окружность заданного радиуса R.  **Назначение процедуры**: Вычисление стороны правильного n-угольника  вычисление периметра n-угольника |
| **2)** | Определите длины всех медиан треугольника, заданного длинами сторон a,b,c.  **Назначение процедуры**: Вычисление медианы, проведенной к стороне **a**: |
| **3)** | Определите углы между тремя векторами, направленными из общей начальной точки с координатами (0,0) в конечные точки: точку (2;5); точку (7;6); точку (9;3).  **Назначение процедуры**: Вычисление угла между 2-мя векторами, проведенными из точки (0, 0) в точки (x1, y1) и (x2, y2) по формуле |
| **4)** | Вычислите площадь пятиугольника, заданного прямоугольными координатами своих вершин: A1(3;2), A2(9;6), A3(14;2), A4(10;-3), A5(7;-2)  Использовать формулу площади треугольника:    **Назначение процедуры:** Вычисление площади треугольника по заданным координатам его вершин: (x1;y1), (x2;y2), (x3;y3) |
| **5)** | Определите длины всех биссектрис треугольника, заданного длинами сторон a, b, c.  **Назначение процедуры:** Вычисление биссектрисы угла |
| **6)** | Вычислите **R** – расстояние между двумя точками **A** и **B**, заданными сферическими координатами. Соотношение между сферическими координатами и декартовыми    **Назначение процедуры:** Вычисление декартовых координат точки по заданным сферическим координатам |
| **7)** | Вычислите значение площади полной поверхности треугольной пирамиды, если известны длины всех ребер:  |AB| = 3; |AD| = 5; |DB| = 4; |DC| =; |BC| =; |AC| = 5.  Для вычисления площади треугольника использовать формулу Герона:, где a, b, c - длины сторон треугольника.  **Назначение процедуры:** Вычисление площади треугольника по формуле Герона |
| **8)** | Определите высоту, на которой будет мяч, подброшенный вертикально вверх с высоты y0=1м и начальной скоростью V0=20м/сек через время t=1сек, 3сек и 4сек  **Назначение процедуры:** Вычисление высоты в момент t , где м/сек |
| **9)** | Определитеплощади правильных n - угольников (10-угольника; 50-угольника; 100-угольника), вписанных в окружность радиуса R  **Назначение процедуры:** Вычисление стороны правильного n-угольника  - радиус вписанной окружности;  площадь n-угольника |
| **10)** | Определите площадь кольца, внутренний радиус которого равен R1; а внешний R2 (R2>R1).  **Назначение процедуры:** Вычисление площади круга радиуса R: |
| **11)** | Вычислите полярные координаты 3-х точек, заданных прямоугольными координатами в правой полуплоскости. Формулы преобразования координат:.  **Назначение процедуры:** Вычисление полярных координат по заданным прямоугольным координатам точки правой полуплоскости |
| **12)** | Определите площадь каждого из 3-х секторов с радиусами **R1, R2, R3** и с центральными углами  **Назначение процедуры:** Площадь сектора радиуса R c центральным углом (в градусах) равна . |
| **13)** | Вычислите  **Назначение процедуры:** Вычисление |
| **14)** | Определите стороны треугольника, заданного величинами своих углов и радиусом описанной окружности.  **Назначение процедуры:** Применить теорему синусов |
| **15)** | Определите площадь каждого из 3-х кругов, ограниченных тремя окружностями, длины которых **L1,L2,L3** известны.  **Назначение процедуры:** Вычислить площадь круга  предварительно вычислив R по формуле |
| **16)** | Определите углы треугольника, длины сторон которого **a, b, c** заданы.  **Назначение процедуры:** Примените теорему половинного угла  где угол, противолежащий стороне с |
| **17)** | Вычислите  **Назначение процедуры:** Вычисление |
| **18)** | Определите общую длину дуги, образованной полуокружностями  3  2  2.5  1  **Назначение процедуры:** Вычисление длины половины окружности |
| **19)** | Вычислите стороны треугольника **A** и **B** при условии, что заданы сторона **C** и углы треугольника и .  **Назначение процедуры:** Вычисление стороны треугольника по формуле ,где  - угол, противолежащий стороне A,  - угол, противолежащий стороне C |
| **20)** | Вычислите значения медиан треугольника, сторонами которого являются медианы исходного треугольника со сторонами **a, b, c.**  **Назначение процедуры:** Вычисление длины медианы, проведенной к стороне **a**: |
| **21)** | Вычислите  **Назначение процедуры:** Вычисление |
| **22)** | Вычислите  **Назначение процедуры:** Вычисление |
| **23)** | Определите длину дуги каждого из трех секторов с радиусами R1, R2, R3 и с центральными углами  **Назначение процедуры:** Длина дуги сектора радиуса R c центральным углом (в градусах) равна |
| **24)** | Вычислите координаты точки пересечения двух прямых:  , вычисляемые по формулам:, где .  **Назначение процедуры:** Вычисление определителя второго порядка. |
| **25)** | Вычислите  **Назначение процедуры:** Вычисление |
| **26)** | Определите все углы треугольникапри заданных значениях сторон **a, b, c.**  **Назначение процедуры:** Вычисление угла по теореме косинусов: |
| **27)** | Вычислите определенный интеграл для функции |
| **28)** | Вычислите площадь трех круговS1, S2 и S3с заданными диаметрами d1, d2 и d4.3.  **Назначение процедуры:** Вычисление |
| **29)** | Вычислите  **Назначение процедуры:** Вычисление |
| **30)** | Определите общую площадь фигуры, образованной полукругами  3  2  2.5  1  **Назначение процедуры:** Вычисление площади полукруга . |

### Содержание отчёта

1. Тема и название лабораторной работы.
2. Фамилия, имя студента, номер группы, номер варианта.
3. **Реализация 1-го задания:**
4. Задание на разработку задачи.
5. Формализация и уточнение задания.
6. Элементы, разрабатываемых проектов в одном решении:
7. графический интерфейс пользователя;
8. таблица свойств объектов;
9. схемы двух алгоритмов;
10. программный код трех проектов;
11. Результаты выполнения трех проектов.
12. Доказательства правильности результатов.
13. **Реализация 2-го задания:**
14. Задание на разработку задачи.
15. Формализация и уточнение задания.
16. Элементы, разрабатываемого проекта:
17. графический интерфейс пользователя;
18. таблица свойств объектов;
19. схемы алгоритмов;
20. программный код проекта;
21. Результаты выполнения проекта 2-го
22. Доказательства правильности результатов.

### Пример выполнения заданий (1-го и 2-го)

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

Структура VB-программ и процедуры.

Средства программирования алгоритмов линейной структуры.

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

Иванов И., БИН1405, вариант 14.3.

1. **Реализация 1-го задания – три проекта в одном решении**:
2. **Задание на разработку задачи:**

Создайте приложение (решение), состоящее из трех проектов с именем *Решение 3.1* для вычисления арифметического выражения (**Тема 3**):



В каждом проекте для вычисления заданного арифметического выражения необходимо реализовать одну из следующих процедур: ***процедуру-функцию с формальными параметрами, процедуру-подпрограмму с формальными параметрами и процедуру без параметров (передача данных в соответствующем проекте будет осуществляться через глобальные объекты).***

1. **Формализация и уточнение задания:**

Для формализации и уточнения задания запишите арифметическое выражение на языке программирования:

**z = (0.002 - Exp(x^y)) / ((100 – y)\*(x + 2))**

В соответствии с требованиями задания разобьем решение задачи на несколько отдельных задач и создадим три процедуры пользователя, вычисляющие заданное арифметическое выражения: процедуру-функцию **func()** с двумя формальными параметрами, необходимыми для вычисления выражения и передающимися по значению как входные аргументы; процедуру-подпрограмму **proc1()** с тремя формальными параметрами, из которых два необходимы для вычисления выражения и передаются по значению как входные аргументы и один является результатом и передается по ссылке; процедуру-подпрограмму **proc2()** без параметров, в которой два необходим для вычисления выражения аргумента, а результат передаются через глобальные объекты.

1. **Элементы, разрабатываемых проектов в одном решении:**
2. Графический интерфейс пользователя:

Разработайте форму каждого проекта (форма проектов

может иметь такой же вид, как в примере **Темы 2;**

1. Таблица свойств объектов:

Установите и сведите в табл. 4.3.7-2 свойства всех   
 объектов (эта таблица аналогична табл. 4.2.8-2);

1. Схемы алгоритмов процедур:
2. Разработайте схемы алгоритмов трех процедур   
    (рис. 4.3.7-1, . 4.3.7-2, . 4.3.7-3) и событийную процедуру

(рис. 4.3.7-4).

|  |  |
| --- | --- |
|  |  |

*Рис. 4.3.7-1. Схема алгоритма процедуры функция* **func()**

**Проекта 3.1.1**

|  |
| --- |
|  |

*Рис. 4.3.7-2. Схема алгоритма процедуры подпрограмма* **proc1()**

**Проекта 3.1.2**

|  |
| --- |
|  |

*Рис. 4.3.7-3. Схема алгоритма процедуры* **proc2()**

**Проекта 3.1.3**

|  |
| --- |
|  |

*Рис. 4.3.7-4**Схема алгоритма событийной процедуры*

**Проекта 3.1.1, Проекта 3.1.2, Проекта 3.1.3**

1. Программные коды проектов Проекта 3.1.1, Проекта 3.1.2,

Проекта 3.1.3

Разработайте программные коды проектов (рис. 4.3.7-5 –

4.3.7-6):

|  |
| --- |
| **Option Strict On**  **Imports System.Math**  **Public Class Form1**  *'Функция, вычисляющая выражение*  **Function func(ByVal x As Double, ByVal y As Double)As Double**  **Dim z As Double 'локальный объект**  **z = (0.002 - Exp(x ^ y)) / ((100 - y) \* (x + 2))**  **Return z** *'возвращение результата*  **End Function**  **Private Sub Button1\_Click(sender As Object, \_**  **e As EventArgs)Handles Button1.Click**  **Dim x, y, z, m, n As Double, p As Single, \_**  **k As Integer**  **Dim s1, s2 As String**  **x = CDbl(TextBox1.Text)**  **y = CDbl(TextBox2.Text)**  **z = func (x, y)** *'вызов функции*  **p = CSng(z)**  **k = CInt(z)**  **m = Fix(z)**  **n = Int(z)**  **s1 = CStr(z)**  **s2 = CStr(p)**  **TextBox4.3.Text = CStr(z)**  **TextBox4.Text = CStr(k)**  **TextBox5.Text = CStr(m) : TextBox6.Text = CStr(n)**  **TextBox7.Text = CStr(p): TextBox8.Text = CStr(z + p)**  **TextBox9.Text = s1 + s2**  **End Sub**  **Private Sub Button2\_Click(sender As Object, \_**  **e As EventArgs)Handles Button2.Click**  **End**  **End Sub**  **END Class** |

*Рис. П.1.3-5. Программный код* **Проект 3.1.1**

|  |
| --- |
| **Option Strict On**  **Imports System.Math**  **Public Class Form1**  **Sub proc1(ByVal x As Double, ByVal y As Double, \_**  **ByRef z As Double)**  **z = (0.002 - Exp(x ^ y)) / ((100 - y) \* (x + 2))**  **End Sub**    **Private Sub Button1\_Click(sender As Object,e As EventArgs)\_**  **Handles Button1.Click**  **Dim x, y, z, m, n As Double, p As Single, k As Integer**  **Dim s1, s2 As String**  **x = CDbl(TextBox1.Text)**  **y = CDbl(TextBox2.Text)**  **proc1(x, y, z)** *'вызов процедуры*  **p = CSng(z)**  **k = CInt(z)**  **m = Fix(z)**  **n = Int(z)**  **s1 = CStr(z) : s2 = CStr(p)**  **TextBox4.3.Text = CStr(z)**  **TextBox4.Text = CStr(k)**  **TextBox5.Text = CStr(m)**  **TextBox6.Text = CStr(n)**  **TextBox7.Text = CStr(p)**  **TextBox8.Text = CStr(z + p)**  **TextBox9.Text = s1 + s2**  **End Sub**  **Private Sub Button2\_Click(sender As Object, \_**  **e As EventArgs)Handles Button2.Click**  **End**  **End Sub**  **End Class** |

*Рис. П.1.3-6. Программный код* **Проект 3.1.2**

|  |
| --- |
| **Option Strict On**  **Imports System.Math**  **Public Class Form1**  *'Глобальные объекты*  **Dim x As Double, y As Double, z As Double**  **Sub proc2()***'Процедура,вычисляющая выражение*  **z = (0.002 - Exp(x ^ y)) / ((100 - y) \* (x + 2))**  **End Sub**  **Private Sub Button1\_Click(sender As Object, \_**  **e As EventArgs)Handles Button1.Click**  **Dim m, n As Double, p As Single, k As Integer**  **Dim s1, s2 As String**  **x = CDbl(TextBox1.Text)**  **y = CDbl(TextBox2.Text)**  **proc2()** *'вызов процедуры*  **p = CSng(z)**  **k = CInt(z)**  **m = Fix(z)**  **n = Int(z)**  **s1 = CStr(z)**  **s2 = CStr(p)**  **TextBox4.3.Text = CStr(z)**  **TextBox4.Text = CStr(k)**  **TextBox5.Text = CStr(m)**  **TextBox6.Text = CStr(n)**  **TextBox7.Text = CStr(p)**  **TextBox8.Text = CStr(z + p)**  **TextBox9.Text = s1 + s2**  **End Sub**  **Private Sub Button2\_Click(sender As Object, \_**  **e As EventArgs)Handles Button2.Click**  **End**  **End Sub**  **End Class** |

*Рис. П.1.3-7. Программный код* Проект 3.1.3

1. **Результаты выполнения трех проектов:**
2. Учтите, что результаты текущей лабораторной работы должны совпасть с данными, приведенными на рис. 4.2.8-11.
3. При выполнении каждого из проектов не забудьте назначить его запускаемым проектом.
4. Выполните первый проект по шагам с заходом в процедуру-функцию (<F11>), поставив точку останова после ввода исходных данных на операторе вызова функции. Обратите внимание на значения всех объектов в окнах **Локальные** и **Видимые** при заходе и выходе из процедуры-функции и возврате в точку вызова в событийной процедуре.
5. Удалите из текста процедуры-функции оператор **Return z** и снова выполните проект по шагам, обосновав результат.
6. Аналогичным образом выполните по шагам с заходом в процедуру второй проект, обращая внимание на значения входных и выходных параметров в окне **Локальные** при заходе и выходе из процедуры.
7. Измените в заголовке процедуры параметр по ссылке на параметр по значению, вместо строки

|  |
| --- |
| **Sub proc1(ByVal x As Double, ByVal y As Double, ByRef z As Double)**  *вставьте строку*  **Sub proc1(ByVal x As Double, ByVal y As Double, ByVal z As Double)** |

и снова выполните проект по шагам, обоснуйте результаты.

1. Точно таким же образом изучите с помощью отладчика по шагам работу третьего проекта.
2. Далее измените его программный код, добавив строку

**Dim x As Double, y As Double, z As Double**

**внутрь событийной процедуры,** при этом оставив такую же строку с описанием глобальных объектов (в самом начале программного кода модуля формы, перед описаниями всех процедур). Обоснуйте результат работы измененного программного кода, выполнив отладку проекта по шагам.

1. Снова измените, программный код третьего проекта, убрав строку с описанием глобальных объектов (закомментируйте или удалите ее). Объясните возникшие ошибки при выполнив отладку проекта по шагам.
2. Результаты выполнения трех проектов должны быть одинаковыми (рис. 4.3.7-8).



*Рис.* 4.3.7*-8. Результаты выполнения* **Проекта 3.1.1**

1. **Доказательства правильности результатов:**

Для доказательства правильности результатов выполните расчет арифметического выражения с использованием калькулятора или программы Microsoft Excel, сделайте вывод о том, что результаты, полученные вручную, подтверждают правильность выполнения программы.

1. **Реализация 2-го задания:**

Вычисления периметра и площади треугольника по заданным координатам трех его вершин**.**

1. **Задание на разработку задачи:**

Создайте приложение с именем Проект 3.4, для вычисления периметра и площади треугольника по заданным координатам трех его вершин**: x1, y1; x2, y2; x3, y4.3.**

Т.е. определите: **Р** – периметр треугольника и **S** – площадь

треугольника. Ограничения на значения исходных данных и их

соотношения: А>0, В>0, C>0, А+В>С, А+C>В, В+C>А

одновременно.

1. **Формализация и уточнение задания:**

Для решения задачи существуют известные формулы:

**Р = А + В + С;**

**S=  (**формула Герона)**;**

**A = ; B = ;**

**C = .**

Где **Рр = Р/2** – полупериметр; **A, B, C** – стороны треугольника.

В соответствии с требованиями задания разобьем решение задачи на несколько отдельных задач и создадим процедуры пользователя, реализующие эти задачи:

* процедуру-**Function ДлинаОтрезка()**, вычисляющую длину отрезка по координатам двух точек;
* процедуру-**Sub PS()**, вычисляющую периметр и площадь треугольника;
* процедуру- **Function vvod()** для ввода действительного числа из объекта типа **TextBox**;
* процедуру-**Sub vivod()** для вывода действительного числа в объект типа **TextBox**.

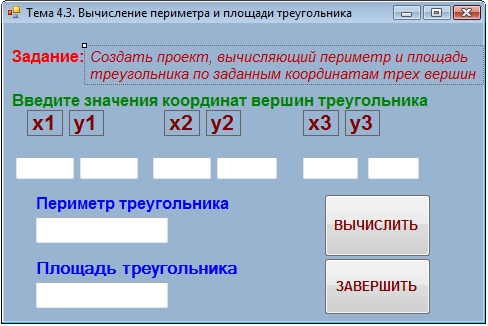
Выбор типа процедуры (**Function** или **Sub**) определяется следующими критериями. Процедуры-функции **Function** предназначены для вычислений и могут быть частью выражения. Результатом выполнения процедуры-**Function** является значение какой-либо одной величины числового, строкового или логического типа, которое функция возвращает или через свое имя, или оператором **Return**. Следовательно, **ДлинаОтрезка( )** и **vvod( )** – это процедуры типа **Function**.

Если же процедура должна вычислить и возвратить несколько величин (**PS()**), или просто выполнить определенные действия, не возвращая никаких значений (**vivod()**), то такую процедуру следует оформить как процедуру-**Sub**.

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

Разработайте графический интерфейс пользователя.

Форма приложения изображена на рис. 4.3.7-9.



*Рис. 4.3.7-9. Форма* проекта **Проект 3.4**

1. Таблица свойств объектов:

Определите и сведите в таблицу свойства объектов

(эта таблица аналогична табл. 4.2.8-2);

1. Схемы алгоритмов процедур проекта Проект 3.4

Разработайте схемы алгоритмов процедур **PS()** и

**ДлинаОтрезка()** приведены на рис. 4.2.8-10, а процедура   
 события на рис. 4.2.8-11.

|  |  |
| --- | --- |
|  |  |

*Рис. 4.3.7-10. Схемы алгоритмов процедур* **ДлинаОтрезка() и PS() Проекта 3.4**

|  |
| --- |
|  |

*Рис. 4.3.7-11. Схемы алгоритмов событийной процедуры   
проекта* **Проект 3.4**

1. Программный код проекта Проект 3.4

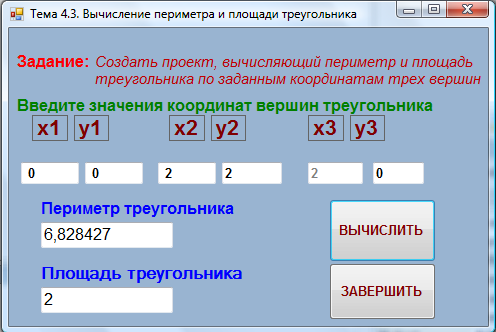
Разработайте программный код проекта (рис. 4.3.7-9).

|  |
| --- |
| **Imports System.Math**  **Public Class Form1**  *'Функция ввода координаты точки в TextBox*  **Function vvod(ByVal T As TextBox) As Single**  **Return CSng(Val(T.Text))**  **End Function**  *'Процедура вывода результата в TextBox*  **Sub vivod(ByVal Z As Single, ByVal T As TextBox)**  **T.Text = CStr(Z)**  **End Sub**  *'Функция вычисления длины отрезка по координатам двух точек*  **Public Function ДлинаОтрезка(ByVal x1 As Single, \_**  **ByVal y1 As Single, ByVal x2 As Single, \_**  **ByVal y2 As Single) As Single**  **Return CSng(Sqrt((x2 - x1) ^ 2 + (y2 - y1) ^ 2))**  **End Function**  *'Процедура вычисления периметра и площади треугольника*  **Sub PS(ByVal x1 As Single, ByVal y1 As Single, ByVal x2 As\_**  **Single, ByVal y2 As Single, ByVal x3 As Single,\_**  **ByVal y3 As Single, ByRef P As Single, ByRef S As Single)**  **Dim A, B, C, Pp As Single**  **A = ДлинаОтрезка(x1, y1, x2, y2)**  ***'Вызов ф-ции выч-я дл.отр.***  **B = ДлинаОтрезка(x2, y2, x3, y3)**  **C = ДлинаОтрезка(x3, y3, x1, y1)**  **P = (A + B + C)**  **Pp = P / 2**  **S = CSng(Sqrt(Pp \* (Pp - A) \* (Pp - B) \* (Pp - C)))**  **End Sub**  **Private Sub Button1\_Click(sender As Object, \_**  **e As EventArgs) Handles Button1.Click**  **Dim x1, x2, x3, y1, y2, y3, Пер, Пло As Single**  **x1 = vvod(TextBox1) : y1 = vvod(TextBox2)**  **x2 = vvod(TextBox3) : y2 = vvod(TextBox4)**  **x3 = vvod(TextBox5) : y3 = vvod(TextBox6)**  **PS(x1, y1, x2, y2, x3, y3, Пер, Пло)**  **vivod(Пер, TextBox7) : vivod(Пло, TextBox8)**  **End Sub**  **Private Sub Button2\_Click(sender As Object, \_**  **e As EventArgs) Handles Button2.Click**  **End**  **End Sub**  **End Class** |

*Рис. 4.3.7-9. Программный код* **Проект 3.4**

1. **Результаты выполнения проекта Проект 3.4**:

Результаты выполнения проекта приведены на рис. 4.3.7-10:



*Рис. 4.3.7-10. Результаты выполнения проекта* **Проект 3.4**

1. **Доказательство правильности результата:**

#### A = = 2.82843 B = = 2

C =  = 2 Р = А + В + С = 6.828427

Рр = Р/2 = 3.414213 S=  = 2

### Контрольные вопросы

1. Какой алгоритм называется линейным?
2. В чем отличие передачи результатов выполнения функций и процедур?
3. Доказывает ли получение правдоподобного результата правильность программы?
4. Какие ошибки могут остаться не выявленными, если не провести проверку (просмотр, прокрутку) программы?
5. Чем тестирование программы отличается от её отладки?
6. Можно ли с помощью тестирования доказать правильность программы?
7. Назовите основные этапы процесса тестирования.
8. В чём отличие синтаксических ошибок от семантических?
9. О чём свидетельствует отсутствие сообщений машины о синтаксических ошибках?
10. Какие разновидности ошибок транслятор не в состоянии обнаружить?
11. Что такое программа?
12. На каком этапе происходит поиск и устранение ошибок в программе?
13. Что такое система программирования?
14. Что такое процедура?
15. Какими преимуществами обладает программа, имеющая процедуры?
16. Какие средства процедурного программирования имеются в языке VB?
17. Каким образом происходит обмен данными между отдельными процедурами программы?
18. Что такое фактические и формальные параметры?
19. Каким образом передать в качестве параметра: константу, переменную, выражение?
20. Как используются глобальные переменные?
21. Как используются локальные переменные?
22. Что происходит при вызове процедуры?
23. Каким образом параметры могут передаваться в процедуру?
24. Какие параметры передаются по значению?
25. Какие параметры передаются по адресу?
26. Что относится к средствам процедурного программирования?
27. Что является результатом выполнения функции?
28. Каким образом надо объявить отдельные переменные, чтобы внутри функции они были локальными?
29. Какой оператор используется для досрочного выхода из функции?
30. Что контролируется при вызове функций и процедур?
31. Как необходимо объявить формальные параметры, чтобы передать параметр-переменную по значению?
32. Как необходимо объявить формальные параметры, чтобы передать параметр-переменную по адресу?
33. Каким образом осуществляется вызов процедуры-функции из вызывающей программы?
34. Что может быть результатом выполнения процедуры?
35. Какой оператор используется для досрочного выхода из процедуры?
36. Где могут находиться описания функций и процедур?
37. Где будет доступна переменная, объявленная с помощью оператора Public?
38. Где может находиться оператор Private?
39. Какие значения имеют переменные в начале программы?
40. Для чего описываются (определяются) процедуры и функции?
41. Как осуществляется вызов процедур?
42. Как осуществляется вызов функций?
43. В чем заключается принцип процедурного программирования?
44. Какие типы процедур существуют в VB .NET?
45. Напишите синтаксис Sub-процедуры, каков синтаксис списка параметров?