## Тема 4.2. Основные средства языка программирования VB .NET

[4.2.1. Синтаксис, семантика, алфавит и лексемы языка VB](#_4.2.1._Синтаксис,_семантика,)

[4.2.2. Типы данных, способы их объявления и области видимости переменных и констант](#_4.2.2._Типы_данных,)

[4.2.3. Выражения, встроенные функции и методы](встроенные#_4.2.3._Выражения,_)

[4.2.4. Преобразование данных в выражениях и операторе присваивания](#_4.2.4._Преобразование_данных)

[4.2.5.Задачи для самостоятельного решения по теме «Основные средства языка](#_4.2.5._Задачи_для)

[программирования VB»](#_4.2.5._Задачи_для)

[4.2.6. Тестовые задания «Основные средства языка программирования VB»](#_4.2.6._Тестовые_задания)

### 4.2.1. Синтаксис, семантика, алфавит и лексемы языка VB

Содержательно язык программирования – это средство общения между человеком (программистом) и компьютером (исполнителем). Рассматривая любую знаковую систему (в том числе и язык программирования), обычно выделяют ***синтаксис***- правила построения сообщений в этой системе, ***семантику***– правила истолкования сообщений тем, кому они адресованы, а также ***прагматику*,** сопоставляющую сообщения желаниям того, от кого они исходят.

Всякий язык программирования, в частности, **VB**, можно определить как множество ***предложений*** (***строк***) – т.е. некоторое множество конечных последовательностей элементарных единиц из некоторого непустого конечного множества символов (***алфавита***), называемого ***словарем*** языка. Понятно, что при таком рассмотрении языка программирования мы только фиксируем множество символов, которые можно использовать для записи программ, а также класс ***допустимых***(или, как принято говорить, ***синтаксически правильных***) программ, не сопоставляя никакого смысла этим синтаксически правильным программам.

Ясно, что задавать множество допустимых программ исчерпывающим их перечислением невозможно. Желательно, чтобы описание языка было обозримым (заведомо конечным), хотя описываемый язык может быть и бесконечным. Обычный подход, удовлетворяющий этому требованию, состоит в том, что предложения (строки) языка строятся по определенным правилам, в совокупности составляющим то, что называют ***грамматикой* языка**. Эти грамматические правила приписывают предложениям языка некоторую синтаксическую структуру, которая используется в дальнейшем при определении смысла предложений.

***Грамматика языка*** может быть описана различными способами. Например, ее можно задавать в виде порождающей системы формальных и неформальных правил, т.е. набора правил, применением которых можно породить все предложения языка и только их. При описании конструкций языка **VB** мы будем применять способ, использующий неформальные правила описания предложений языка **VB**.

***Семантика языка*** программирования – это правила придания смысла синтаксически правильным программам. В конечном счете, эти правила определяют ту последовательность действий вычислительной машины, которую она должна выполнить, работая по данной программе. Например, семантика языка команд компьютера определяется самим компьютером: машинная программа описывает в точности то, что реализует вычислительная машина при работе по данной программе. Аналогично ***семантика языка программирования высокого уровня*** может быть определена через описание правил выполнения соответствующей виртуальной машины.

***Алфавит языка*** использует для записи программ сравнительно небольшое множество литер (символов), доступных на устройствах ввода-вывода реальных ЭВМ. Предполагается, что ***входной алфавит*** содержит буквы русского и латинского алфавитов, десятичные арабские цифры и некоторое количество специальных ***литер,*** таких как пробел, точка, запятая и т.п.

Таким образом, ***в алфавит языка программирования* VB** входят:

* прописные (A – Z) и строчные (a – z) буквы латинского алфавита;
* прописные (А – Я) и строчные **(**а – я) буквы кириллицы;
* цифры от 0 до 9;
* символ подчеркивания – \_ ;
* неизображаемые символы (пробел, табуляция, переход на новую строку);
* специальные символы, участвующие в построении конструкций языка: +- \*/; \^ =><[ ] () . :{} ’ ”& @;
* составные символы, воспринимаемые как один символ: <= >= <>.

Для записи даже элементарных конструкций языка **VB** используются не отдельные ***литеры (символы)*** входного алфавита, а ***слова***, называемые ***лексемами***. При этом символы, образующие лексему, теряют свою индивидуальность: ***смысловое значение, приписываемое лексеме семантикой, не обязано вытекать из смысловых значений составляющих ее символов.***

***Множество лексем*** образует ***словарь языка***. Он включает в себя следующие четыре группы лексем:

* множество ***специальных символов***, состоящее из знаков операций (например, **+**,***\****  и т.д.) и ограничителей (например, точка, запятая и т.п.);
* множество ***зарезервированных (ключевых) слов*** (например, Sub, End, If и т.д.);
* множество ***идентификаторов***, используемых в качестве ***имен*** констант, переменных, типов, объектов, модулей, процедур и функций и т.д.;
* множество ***изображений константных значений***, состоящее из целых и вещественных чисел и символьных и строковых констант и т.д.

Рассмотрим правила использования некоторых лексических элементов.

***Идентификаторы*** – это имена лексического элемента языка программирования. Для выбора имен и лексического элемента языка следует придерживаться следующих правил:

* имя должно состоять только из символов латинского алфавита или кириллицы (регистр значения не имеет), цифр и символа подчеркивания;
* имя должно начинаться с буквы;
* имена не могут совпадать с зарезервированными словами;
* имена не должны содержать более 255 символов;
* не следует использовать имена, совпадающие с названиями функций, инструкций и методов языка;
* нельзя использовать одни и те же имена переменных, отличающихся типом;
* не допускается использование повторяющихся имен на одном уровне области определения.

***Ключевые слова*** – это идентификаторы, зарезервированные для специального использования (например,Const, Do, For, If, While и т.д.) и не использующиеся как имена для программных элементов, таких, например, как переменные и процедуры.

***Операторы*** языка программирования – это последовательности команд, из которых состоит ***код программы*** (текст программы). ***Оператор*** – это синтаксическая единица языка програм­мирования, которая определяет некоторое действие. Операторы описывают алгорит­мические действия, которые должны выполняться для решения поставленной задачи. Каждый оператор представляет собой законченную фразу (предложение, строку) языка программирования и может содержать ключевые слова, переменные, константы, выражения, разделители др. Операторы могут быть простыми и структурированными. Простым опера­тором является такой оператор, который не содержит в себе других операторов. Структурированные операторы строятся из других операторов, порядок выполне­ния которых должен быть последовательным, определяемым условной передачей управления (условными операторами) или повторяющимся (определяется операторами цикла). Обычно оператор располагается на одной строке, но его можно продолжить на другой строке с помощью символа продолжения строки – знака– ''***\_***'' ***(подчеркивание), причем символу продолжения должен предшествовать пробел; или на одной строке можно записать и несколько операторов, разделяя их знаком*** – ***: (двоеточие).***

***Комментарии*** – это пояснения отдельных конструкций и их действий в программе. ***Строки комментария начинаются со знака – ' (апостроф).*** Комментарии обычно вставляют в начале или в конце строки программного кода. При выполнении программы **VB** игнорирует комментарии. По умолчанию при выполнении приложения комментарий выделя­ется зеленым цветом.

***Данные*** – это информация, представленная в формальном виде, который обес­печивает возможность ее хранения, обработки и передачи (см. Тема 1.4 и Тема 2.2). В основе любого языка программирования лежит конечное множество данных, неделимых (простых) с точки зрения имеющихся в языке операций. Это множество распадается на четыре подмножества, состоящих соответственно из целых чисел, вещественных чисел, логических значений и литерных (строковых) значений. Каждое из перечисленных подмножеств – это ***стандартный тип данных***, ***который имеет свое имя (идентификатор), характеризуется множеством значений и набором операций***, определяемых языком программирования над этими значениями (например, числа можно складывать, а логические или литерные значения – нет). Как известно, данные в программе могут быть двух видов: постоянные величины (***константы***) и пере­менные величины (***переменные***). Предполагается, что каждая переменная в программе (переменная –это элемент памяти ЭВМ; адрес элемента памяти – это имя переменной; а его содержимое – значение переменной) может получать значения только определенного заранее типа, в то время как элементы памяти **ЭВМ** могут хранить любое по типу значение –его трактовка определяется только выполняемой операцией. Помимо переменных программа может содержать константы – элементы памяти **ЭВМ**, которые постоянно хранят заданные значения. В **VB** значение константы может использоваться двумя способами: или неявно, своим именем – адресом соответствующего элемента памяти, или явно, изображением значения содержимого соответствующего элемента памяти. Например, True – это имя значения "***истина***", а 125– изображение значения числа "***сто двадцать пять***".

### 4.2.2. Типы данных, способы их объявления и области видимости переменных и констант

Итак, переменные используются во время выполнения программы для временного хранения данных в памяти компьютера. Для ***определения (объявления)*** переменных компилятору нужно следующее:

* ***имя переменной*** – по имени компилятор осуществляет связь переменной в программе с оперативной памятью компьютера;
* ***тип переменной*** – тип позволяет компилятору определить, какого вида информация хранится в переменной;
* ***значение переменной*** – определяет содержание информации, которая помещается в переменную.

При выборе имен переменных следует исполь­зовать имена, которые являются короткими и, в то же время, интуитивно понят­ными и простыми для запоминания. Чтобы избежать противоречий, рекомендуется использовать при создании имен переменных следующие соглашения:

* начинать имя каждой переменной с буквы или знака подчеркивания. Это тре­бование **VB** (имена переменных могут содержать только буквы, цифры и знак подчеркивания);
* выбирать имена переменных не длиннее 33 символов, чтобы их было легко читать (хотя имена переменных могут быть практически любой длины);
* делать имена своих переменных описательными, комбинируя, когда это имеет смысл, одно или несколько слов;
* использовать комбинации заглавных и строчных символов и цифр, то есть первую букву каждого слова имени переменной делать заглавной, например, ДлСтроки;
* не использовать в качестве имен переменных ключевые слова **VB**, име­на объектов или свойств (если вы это сделаете, то при попытке запуска про­граммы получите сообщение об ошибке);

Какими бы сложными объектами, массивами или коллекциями мы не пользовались при написании программы, следует помнить, что основой их являются простейшие (базовые) типы. Это основные кирпичики информации, атомы, из которых будет формироваться впоследствии сложная структура огромных объектов. Следует точно знать, какой именно вид информации хранится в той или иной переменной, и в соответствии с этим знанием применять к ней те или иные операции или функции обработки. Обратите внимание, что **Visual Basic .Net** имеет достаточно хорошую ***типизацию данных***.

Таблица 4.2.2-1 содержит перечень базовых (простых) ти­пов данных **VB**. Причем каждый базовый тип языка **VB** соответствует типу данных **.NET Framework**.

В ***Разделе 1. «Арифметические и логические основы работы ЭВМ»*** электронного пособия «**Информатика**» [2] подробно изложены вопросы представления данных в памяти **ЭВМ.**

В программном коде может использоваться любое количество переменных, и они могут со­держать слова, числа, даты, свойства или другие значения. Использова­ние переменных позволяет присвоить каждой части данных, с которыми вы работаете, короткие и легко запоминаемые имена. Переменные могут хранить информацию, введенную пользователем при выполнении про­граммы, результат некоторого вычисления или данные, которые отображаются на форме. Если коротко, то переменные явля­ются удобными ***контейнерами****,* которые используются для хранения ин­формации почти любого типа.

Таблица 4.2.2-1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Тип**  **данных** | **Размер** | **Диапазон** | **Пример использования** | **Что хранит**  **(Тип .NET)** |
| Short | *16-бит* | от-32768  до 32767 | Dim Birds As Short  Birds = 12500 | Целые числа  (System. Int) |
| UShort | *16-бит* | от 0  до 65535 | Dim Days As UShort  Days = 55000 | Целые числа  (System.UInt) |
| Integer | *32-бит* | от-2147483648  до 2147483647 | Dim Insects As Integer  Insects = 37500000 | Целые числа  (System.Int32) |
| UInteger | *32-бит* | от 0  до 4294967295 | Dim Joys As UInteger  Joys = 3000000000 | Целые числа  (System.UInt32) |
| Long | *64-бит* | от-9223372036854775808  до 9223372036854775807 | Dim WorldPop As Long  WorldPop = 4800000004 | Целые числа  (System.Int64) |
| ULong | *64-бит* | от 0  до 18446744073709551615 | Dim Stars As ULong  Stars=180000000000000000 | Целые числа  (System.UInt64) |
| Single | *32-бит* | от -3.4028235Е38  до 3.4028235Е38 | Dim Price As Single  Price = 899.99 | Числа с пл. точкой  (System.Single) |
| Double | *64-бит* | от -1.79769313486231Е308  до 1.79769313486231Е308 | Dim Pi As Double  Pi = 3.1415926535 | Числа с пл. точкой  (System.Double) |
| Decimal | *128-бит* | Полож. и отриц. значения  до 29 знаков | Dim Debt As Decimal  Debt =7600300.50 | Числа с фикс. точкой  (System.Decimal) |
| Byte | *8-бит* | от 0  до 255 без знака | Dim RetKey As Byte  RetKey = 13 | Десятичные числа  (System.Byte) |
| SByte | *8-бит* | от-128  до127 | DimNegValAsSByte  NegVal = -20 | Целые числа  (System.SByte) |
| Char | *16-бит* | любой символ Unicode в диапазоне 0 - 65535 | Dim UnicodeChar As Char UnicodeChar="A" | Один символ  (System.Char) |
| String | *16-бит* | от0 до ≈ 2 миллиардов символов Unicode | DimDogAs String  Dog ="pointer" | Текст  (System.String) |
| Boolean | *16- бит* | True или False (при преобразовании типов 0 преобразуется в False, а все другие значения в True) | Dim Flag As Boolean  Flag =True | Логические знач.  (System.Boolean) |
| Data | *64-бит* | от 1 января 0001  до 31 декабря 9999 | Dim Birthday As Date  Birthday = #3/1/1963# | Дату и время  (System.DateTime) |
| Object | *32-бит* | любой тип, который может быть сохранен в  переменной типа Object | Dim App As Object  App = CreateObject \_ ("Word.Application") | Ссылка на объект  (System.Object) |

Как было отмечено ранее, использование переменных в программе на **VB** требует не­которого планирования. Прежде чем использовать переменную, необ­ходимо выделить в компьютере память, которую эта переменная будет занимать. Этот процесс несколько напоминает бронирование мест в театре или на спортивном матче (ячейки, контейнеры).

Для каждого типа переменных выделяется определенное количество оперативной памяти, и тип данных надо выбирать, исходя из задания на решение конкретной задачи. Если данные должны принимать только целые значения, то тип соответствующих переменных в программе должен быть Integer**,** но никак не Single, поскольку в этом случае мы увеличиваем объем используемой оперативной памяти и не гарантируем получение правильного (точного) результата. Если из постановки задачи известно, что данные могут принимать дробные значения, то соответствующим переменным в программе необходимо назначить тип Single или Double, поскольку любой другой тип переменных может привести к непредвиденным результатам. Поэтому в **VB** следует ***всегда явно объявлять свои переменные до их использования.*** А для того чтобы **VB** информировал программистов о том, что переменная явно не объявлена до ее использования, ***необходимо в начале программного кода (перед всеми процедурами) и объявлениями поместить оператор:***

Option Explicit Оn

Чтобы в **VB *явно объявить*** переменную, необходимо ввести ***имя пере­менной*** после оператора Dim(Dim означает ***dimension***– размер)*,* а ***тип переменной*** после слова AS*.* При этом оператор объяв­ления выделяет для переменной место в памяти, по которо­му компилятор **VB** узнает тип данных, связанный с этой переменной.

Оператор Dim записывается по следующим правилам:

Dim *имена\_переменных* As *тип, имена\_переменных* As *тип,…*

Например, следующие операторы резервируют «место» в памяти **ЭВМ** для различных типов переменных:

|  |
| --- |
| **Dim LName As String**  **Dim Кол As Integer, b As Single**  **Dim Ном, d As Integer** |

Таким образом, для пере­менной с именем LName резервируется оперативная память*,* которая будет хранить ***стро­ковое значение***; для переменных с именем Кол и b резервируется оперативная память**,** которая будет хранить, соответственно, ***целые*** и ***вещественные значения***; для переменных с именем Ном и d резервируется оперативная память, которая хранит ***целые значения***.

Кроме оператора Dim, переменные можно объявить и с помощью операторов Private и Public**:**

Private *имена\_переменных* As *тип***,** *имена\_переменных* As *тип…*

Public *имена\_переменных* As *тип*, *имена\_переменных* As *тип…*

Выбор ключевого слова Dim, Private или Public определяется необходимой ***областью видимости переменной***.

Более подробно области видимости переменных и, соответственно, использование операторов Private и Public будут рассмотрены в следующей теме.

После создания переменной ей можно присваивать конкретные значения, используя оператор присваивания (символ равенства **=**).

***Оператор присваивания*** – это фундаментальный оператор любого языка программирования. Он ***заменяет текущее значение переменной, стоящей слева от знака равенства, новым значением, которое определяется значением выражения, стоящим справа от знака равенства***. Оператор присваивания позволяет также изменять значения свойств формы и размещенных на ней управляющих элементов. Синтаксис оператора имеет следующий вид:

*Идентификатор* = *выражение.*

Например,

|  |
| --- |
| **Res = 20.56**  **X1 = А + 5 / sin(x^ 2)** |

Следующий оператор программы присваивает переменной LName значение «Иванов»:

|  |
| --- |
| **LName = "Иванов"** |

После того, как переменной LName присвоено значение, она может использоваться в программном коде вместо имени «Иванов».

Например, оператор присваивания

|  |
| --- |
| **Label1.Text = LName** |

отобразит «Иванов» в первой надписи (*Label1)* на форме.

Переменные могут хранить одно и то же значение на протяжении всей программы, а могут несколько раз изменять свое значение, в соответствии с программным кодом.

Следующий пример демонстрирует, как перемен­ная с именем LName может содержать различные текстовые значе­ния и как значение этой переменной может присваиваться свойствам объектов.

**Пример 4.2.2-1.Создать проект, в котором изменяются значения переменных.**

1. Запустить **VS**.
2. Выполнить команды, позволяющие создавать новый ***проект с именем Пример-4.2.2-1.***
3. В окне **Конструктор** (**Designer)** появится пустая форма ***Пример-4.2.2-1.***

На ней необходимо создать интерфейс проекта ***Пример-4.2.2-1*** (рис.4.2.2-1).

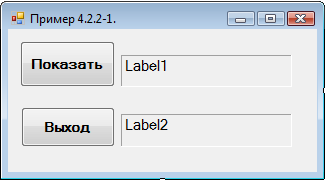


Рис. 4.2.2-1

Эта форма содержит две надписи и две кнопки. Для отображения ин­формации в каждой из этих надписей необходимо использовать пере­менные. Объекты надписей выглядят как поля потому, что свойства *BorderStyle* установлены в значение *Fixed3D*.

1. Дважды щелкнуть мышью на кнопке формы **Показать*.***

В окне **Редактор кода (Code Editor)** появится процедура события Button1\_Click( ).

1. Чтобы объявить и использовать переменную LName, необходимо ввести операторы:

|  |
| --- |
| **Dim LName As String**  **LName = "Петров"**  **Label1.Text = LName**  **LName = "проф. Сидоров"**  **Label2.Text = LName** |

Эти операторы разбиты на три группы. Первый опера­тор объявляет переменную LName типа String, используя для этого оператор Dim. После ввода этой строки **VS** выведет под переменной LName волнистую линию, указывающую, что она была объявлена, но не используется в программе. Здесь нет ошибки –**VS** просто напоминает, что была создана новая переменная, кото­рую следует использовать.

Если написание программы закончено, а имя переменной останется подчеркнутым, то, возможно, где-то в коде програм­мы неправильно написано ее имя или эта переменная действительно не использовалась.

Вторая и третья строки присваивают переменной LName имя «Петров», а затем отображают это имя в первой надписи формы. Этот пример демонстрирует одно из наиболее частых применений пере­менных в программе – передачу информации в свойство. На экране все строковые значения, присваиваемые переменным, отображаются красным цветом.

Четвертая строка присваивает переменной LName значение «проф. Сидоров» (другими словами, она изменяет содержимое этой пе­ременной). Обратите внимание, что вторая строка длиннее, чем пер­вая, и содержит пробелы. Когда переменным присваива­ются текстовые строки или когда строки используются в других местах, текст этих строк требуется заключать в кавычки.

Наконец, необходимо помнить еще об одной важной характеристике переменных, объявленных в этой процедуре события: они поддерживают область видимости или хранят свои значения только внутри той процедуры события, в которой они используются. Далее мы узнаем о том, как объявлять переменные так, чтобы их можно было использовать в любой из процедур событий формы или других процедурах.

1. Чтобы снова отобразить форму, необходимо щелкнуть на вкладке ***Form1.vb [Конструктор] (Form1.vb [Design])***.
2. Дважды щелкнуть мышью на кнопке ***Выход***.

В окне **Редактор кода (Code Editer)** появится процедура события Button2\_Click( ).

1. Чтобы обозначить точку конца программы, где она остановит свое выполнение, необходимо ввести оператор End.

Окно редактора кода будет таким, как показано на рис. 4.2.2-2.

|  |
| --- |
| **Public Class Form1**  **Private Sub Button1\_Click(ByVal sender As System.Object, \_**  **ByVal e As System.EventArgs) \_**  **Handles Button1.Click**  **Dim LName As String**  **LName = "Петров"**  **Label1.Text = LName**  **LName = "проф. Сидоров"**  **Label2.Text = LName**  **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. 4.2.2-2

1. Чтобы сохранить изменения в проекте, следует щелкнуть на кнопке ***Сохранить все (Save All)*** главной панели инструментов.
2. Далее, чтобы запустить программу, необходимо щелкнуть на кнопке ***Начать отладку (Start Debugging)***,расположенной на главной панели инструментов.

Программа запустится, и на экране отобразится первоначальное состояние формы.

1. Щелкнуть на кнопке приложения ***Показать***.

Программа начнет выполняться: объявляется переменная, ей поочередно присваиваются два значения, и каждое из значений копируется в соответствующую надпись на форме. Результат работы программы отображается на экранной форме, показанной на рис. 4.2.2-3.

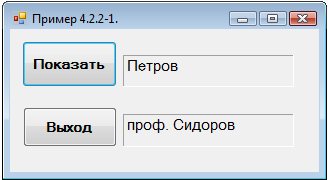


Рис. 4.2.2-3

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

**Пример 4.2.2-2. Создать проект, показывающий использование фундаментальных (базовых) типов данных.**

1. Запустить **VS**.
2. Выполнить команды, позволяющие создавать ***проект с именем Пример-4-2.2-2.***
3. В окне **Конструктор** (**Designer)** появится пустая форма ***Пример-4.2.2-2.***

На ней необходимо создать интерфейс проекта ***Пример-4.2.2-2***(рис.4.2.2-4).

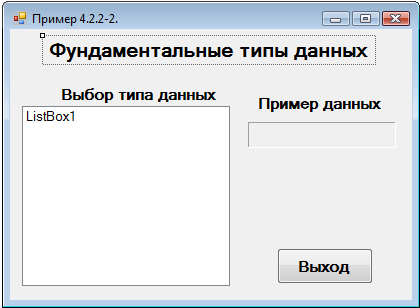


Рис. 4.2.2-4

Проект ***Фундаментальные типы данных***– это учебный проект на **VB**, который будет демонстрировать работу основных типов данных. Необходимо обратить внимание, на то, где должны рас­полагаться объявления переменных так, чтобы они были доступны для всех процедур событий вашей программы.

Кроме того, в этом проекте используется элемент управления ***ListBox.*** Рассмотрим этот элемент более подробно.

***ListBox –*** это список, в который можно записывать и из которого можно выбирать одно из предложенных значений. Значения в списке могут размещаться в одну или несколько колонок в зависимости от значения свойства *MultiColumn*. Если элементы списка расположены в нескольких колонках, с помощью свойства *ColumnWidth* можно изменить заданную по умолчанию ширину колонок.

Элементы в списке могут добавляться во время разработки проекта и программно – во время выполнения программы.

При формировании списка во время проектирования с помощью свойства *Items* вручную задается весь необходимый список. После задания всех элементов нужно нажать кнопку ***OK***, окно редактора элементов управления закроется, а элемент управления ***ListBox*** будет отображать указанные значения.

1. В окне **Редактор кода** (**CodeEditor)** написать программный код такой, как показано на рис. 4.2.2-5.

|  |
| --- |
| **Public Class Form1**  **Dim Birds As Short**  **Dim Insects As Integer**  **Dim WorldPop As Long**  **Dim Price As Single**  **Dim Pi As Double**  **Dim Debt As Decimal**  **Dim RetKey As Byte**  **Dim UnicodeChar As Char**  **Dim Dog As String**  **Dim Flag As Boolean**  **Dim Birthday As Date**  **PrivateSub ListBox1\_SelectedIndexChanged(ByVal sender As \_**  **System.Object, \_**  **ByVal e As System.EventArgs) \_**  **Handles ListBox1.SelectedIndexChanged**  **Select Case ListBox1.SelectedIndex**  **Case 0**  **Birds = 12500**  **Label3.Text = Birds**  **Case 1**  **Insects = 37500000**  **Label3.Text = Insects**  **Case 2**  **WorldPop = 4800000004**  **Label3.Text = WorldPop**  **Case 3**  **Price = 899.99**  **Label3.Text = Price**  **Case 4**  **Pi = 3.1415926535**  **Label3.Text = Pi**  **Case 5**  **Debt = 7600300.5**  **Label3.Text = Debt**  **Case 6**  **RetKey = 13**  **Label3.Text = RetKey**  **Case 7**  **UnicodeChar = "Д"**  **Label3.Text = UnicodeChar**  **Case 8**  **Dog = "pointer"**  **Label3.Text = Dog**  **Case 9**  **Flag = True**  **Label3.Text = Flag**  **Case 10**  **Birthday = #3/1/1963#**  **Label3.Text = Birthday**  **End Select**  **End Sub**  **Private Sub Form1\_Load(ByVal sender As System.Object, \_**  **ByVal e As System.EventArgs) HandlesMyBase.Load**  **ListBox1.Items.Add("Short")**  **ListBox1.Items.Add("Integer")**  **ListBox1.Items.Add("Long")**  **ListBox1.Items.Add("Single")**  **ListBox1.Items.Add("Double")**  **ListBox1.Items.Add("Decimal")**  **ListBox1.Items.Add("Byte")**  **ListBox1.Items.Add("Char")**  **ListBox1.Items.Add("String")**  **ListBox1.Items.Add("Boolean")**  **ListBox1.Items.Add("Date")**  **End Sub**  **Private Sub Button1\_Click(ByVal sender As System.Object, \_**  **ByVal e As System.EventArgs) \_**  **Handles Button1.Click**  **End**  **End Sub**  **EndClass** |

Рис. 4.2.2-5

Необходимо отметить, что все переменные, объявленные в модуле формы, будут доступны, т.е. будут иметь область видимости во всех процедурах событий формы. Таким образом, можно установить значение переменной в одной процедуре события, а прочитать его - в другой. Обычно переменные доступны только в той процедуре собы­тия, в которой они объявлены. Чтобы сделать их доступными для всей формы, их требуется объявить в верхней части кода формы, перед событийными процедурами.

1. Щелкнуть на кнопке ***Начать отладку (StartDebugging)*** на стандартной панели инструментов.

Появится окно приложения, изображенное на рис. 4.2.2-6.

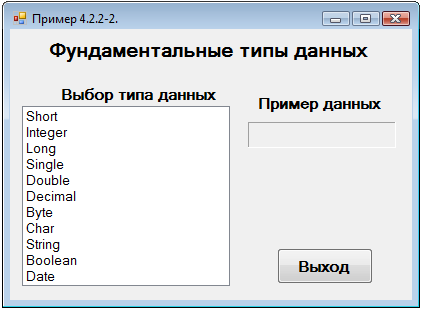


Рис. 4.2.2-6

Программа ***Фундаментальные типы данных*** позволяет экспериментировать с 11 типами данных, в числе которых целый тип, тип значений с плавающей точкой одинарной точности и тип дат. При щелчке на имени типа в поле спис­ка программа отображает пример данных этого типа.

1. Щелкнуть в поле списка на типе Integer*.*

В ***поле формы* Пример данных** появится число 37 500 000. Заметьте, что ни один из типов данных Short,Integer и Long не позволяет вводить или отображать в их значениях десятичные точки.

1. Щелкнуть в поле списка на типе Date.

В ***поле формы* Пример данных** появится дата ***3/1/1963***.

1. Чтобы увидеть, как **Visual Basic** отображает типы данных в ***поле Пример данных***, необходимо щелкнуть в поле списка на каждом из них.
2. Чтобы остановить программу, щелкните на кнопке ***Выход*** приложения*.*

Теперь необходимо посмотреть, как в форме объявляются базовые типы данных и как они используются в процедуре события ListBox1\_Selected\_IndexChanged( ).

1. Изучить процедуру события Form1\_Load( ) (рис.4.2.2-5).

Операторы этой процедуры добавляют значения в элемент управления ***ListBox*** программно при загрузке формы в начале отладки.

1. Изучить процедуру события ListBox1\_SelectedIndexChanged( ) (рис.4.2.2-5).

Процедура события ListBox1\_SelectedIndexChanged( )обрабатывает вы­бор, сделанный в поле списка.

Сердцем этой процедуры события является условная конструкция SelectCase. Далее мы обсудим, как эта конструкция вы­бирает один вариант из нескольких. А сейчас обратите внимание, как каждый выбор блока Select Case присваивает заданное значение од­ной из переменных базового типа данных, а затем присваивает зна­чение этой переменной свойству *Text* объекта *Label3* формы.

1. В процедуре события ListBox1\_SelectedlndexChanged( ) внимательно изучите каждое из присвоений значений переменным.

Попробуйте изменить данные в нескольких операторах присвоения пе­ременных и запустить эту программу, чтобы увидеть, как изменились данные. В частности, вы можете попробовать присвоить переменным значения, которые находятся вне области допустимых значений, пока­занных в таблице типов данных. Если вы сделаете такую ошибку, **VB** в окне **Редактор кода (Code Editor)** подчеркнет некорректное значение волнистой ли­нией, и программа не запустится до тех пор, пока вы не исправите это значение. Чтобы узнать об ошибке поподробнее, поместите мышь над подчеркнутым волнистой линией значением и прочтите всплывающую подсказку, содержащую сообщение об этой ошибке.

1. Если были сделаны какие-то изменения, которые необходимо записать на диск, то щелкнуть на кнопке ***Сохранить все (SaveAll*)** на стандартной панели инстру­ментов.

Из первых программ стало известно, что строка программного кода на **VB** называется оператором программы. Оператор программы является комбинацией ключевых слов **VB**, свойств, имен объектов, переменных, чисел, специальных символов и других зна­чений, которые вместе образуют допустимую инструкцию, распознава­емую компилятором **Visual Basic**. Оператор программы может быть од­ним ключевым словом, например End,которое останавливает выполнение программы на **VB**, или мо­жет быть комбинацией нескольких элементов, как следующая запись, которая для присвоения свойству *Text* объекта *Label1* текущего систем­ного времени использует свойство *TimeString:*

|  |
| --- |
| **Labe11.Text = TimeString** |

Рассмотрим правила ***объявления констант***. Если переменная в программе содержит значение, которое никогда не меняется (такое, например, как число **π***,* являющееся фиксированной математичес­кой величиной), то целесообразно хранение такого значения не в переменной, а в виде константы.

***Имя константы*** – это осмысленное имя, которое заменяет неизменяемое при выполнении программы число или тестовую строку. Константы увеличивают читаемость кода программы, снижают количество программных ошибок и облегчают внесение в код глобальных изменений. Константы действуют как переменные, но при работе программы их значение не изменяется. Они объявляются с помо­щью ключевого слова Const*,* в соответствии со следующим правилом:

Const *имя\_константы* AS *тип = значение\_константы*

|  |
| --- |
| **Const Pi As Double = 3.14159265**  **Label1.Text = CStr(Pi)** |

Этот оператор создает константу с именем Pi*,*которая может быть ис­пользована в коде программы вместо значения 3.14159265. Чтобы сделать констан­ту доступной для всех объектов и процедур формы, по­местите этот оператор в верхней части кода формы вместе с другими объявлениями переменных и структур, область видимости которых рас­пространяется на все процедуры формы.

***Инициализаторы*** – это значения, которые присваиваются переменным во время ее объявления. Это позволяет упростить код и облегчить его чтение. Например,

|  |
| --- |
| **Dim Wd As Single = 100**  **Dim Fn As String = "Петров"**  **Dim Eps As Double= 0.0001** |

Инициализировать переменную можно только тогда, когда это единственная переменная, объявляемая в выражении Dim, Public или Private:

|  |
| --- |
| **Dim x, y, z As Double = 1 ‘ оператор вызовет ошибку** |

### 4.2.3. Выражения, «встроенные» функции и методы

***Выражение*** - это единица программного кода, которая определяет способ вычисления некоторого значения. Выражения в языке программирования формируются в соответствии с определенными правилами ***из констант и перемен­ных, компонент массивов, полей записей, функций и знаков операций***. Запись в тексте программного кода отличается от обычной математической записи. Операции подразделяются на следующие типы:

* арифметические;
* логические;
* строковые.

Здесь подробно рассмотрим только ***арифметические операции***, строковые операции очень кратко, а логические операции будут рассмотрены в другой теме.

Принято различать ***унарные*** и ***бинарные*** операции. Унарный знак плюс «+» перед величи­ной не изменяет ее значения. Унарный знак минус «- » перед величиной изменяет ее знак на противоположный. К бинарным арифметическим операциям относятся сложение, вычитание, умножение, деление, целочисленное деление (без остатка), остаток от деления, возведение в степень.

Если в выражении используются несколько данных (переменных, констант и функций), то их ***типы должны быть идентичными либо совместимыми***. Далее об этом поговорим подробнее. Например, арифметические операции, применимые только к величинам целых и вещественных типов, приведены в таблице 4.2.3-1.

Таблица 4.2.3-1

|  |  |
| --- | --- |
| **Операция** | **Название** |
| **+** | Сложение |
| **-** | Вычитание |
| **\*** | Умножение |
| **/** | Деление |
| **\** | Целочисленное деление (без остатка) |
| **Mod** | Остаток от деления |
| **^** | Возведение в степень |

Таблица 4.2.3-2

|  |  |  |
| --- | --- | --- |
| **Операция** | **Приоритет** | **Тип действий** |
| **()** | 1 | Вычисление в круглых скобках |
| **Функция** | 2 | Вычисление значения функции |
| **^** | 3 | Возведение в степень |
| **Унарный + или –** | 4 | Унарные операции |
| **\*; /** | 5 | Операции умножения и деления |
| **\** | 6 | Целочисленное деление |
| **Mod** | 7 | Остаток от деления |
| **+; -** | 8 | Сложение и вычитание |

В выражениях используются круглые скобки для заключения в них тех частей выражения, которые необходимо вычислить в первую очередь. Количество открывающих скобок должно быть равно количеству закрывающих. Вычисление выражений выполняется в определенном порядке, установленном круглыми скоб­ками, и в соответствии с приоритетом операций (таблица 4.2.3-2).

Операции с равным приоритетом обычно выполняют слева направо. Операция Mod возвращает остаток, полученный путем деления двух операндов целого типа.

**Пример 4.2.3-1.Создать проект,демонстрирующий использование операций +, -, \* и /.**

1. Запустить **VS**.
2. Выполнить команды, позволяющие создавать новый ***проект с именем Пример-4.2.3-1. Базовые математические операции.***
3. В окне **Конструктор форм** (**Designer)** появится пустая форма***.***

На ней необходимо создать интерфейс проекта ***Пример-4.2.3-1*** (рис.4.2.3-1).

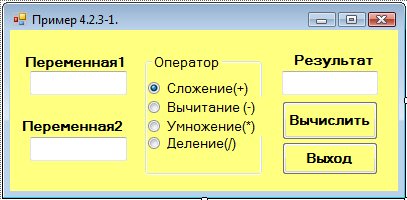


Рис.4.2.3-1. Форма проекта **Пример 4.2.3-1**

Проект ***Пример-4.2.3-1*** демон­стрирует, как с вводимыми числами работают операторы сложения, вычитания, умножения и деления. Он также демонстрирует, как ис­пользовать для обработки в программе ввода пользователя объекты текстового поля, переключателя и кнопки.

В форме проекта ***Пример-4.2.3-1*** используются элементы управления ***RadioButton*** (переключатели). Так как обычно эти элементы располагаются в группах, то они позволяют выбрать одно из нескольких значений. Установка одного переключателя в группе (присвоение его свойству *Checked* значения *True*) автоматически сбрасывает другие переключатели, присваивая аналогичным свойствам значение *False*.

При размещении в форме нескольких групп переключателей, каждая логическая группа должна помещаться в отдельный контейнер, например, в элемент управления ***GroupBox***. В этом случае для создания в форме группы переключателей необходимо сначала поместить в форму рамку, а затем разместить в ней поочередно необходимое количество переключателей ***RadioButton.***

1. Для написания программного кода этого проекта дважды щелкнуть мышью на кнопке ***Вычислить*** формы. Окно **Редактор кода (Code Editor*)*** отобразит пока пустую процедуру события Button1\_Click(). Ввести в нее код, приведенный на рис.4.2.3-3.

Дважды щелкнуть мышью на кнопке ***Выход*** формы и в появившуюся процедуру события Button2\_Click() ввести оператор End. Экран должен выглядеть, как на рис. 4.2.3-3.

1. Щелкнуть на кнопке ***Начать отладку (Start Debugging)****,*на главной панели инструментов.

Программа запустится в **IDE**. Эта программа отображает два текстовых поля, в которые вводятся числовые значения, группу переключателя операторов, поле, которое отображает результаты, и два объекта кнопок (**Вычислить** и **Выход*)***.

1. В текстовом поле ***Переменная1***ввести число 100**,** а затем нажать на**<Tab>**.Курсор ввода переместится во второе текстовое поле (это поле получит фокус ввода).
2. В текстовом поле ***Переменная2*** ввести число17. Теперь к этим двум зна­чениям можно применить любой из математических операторов.
3. Щелкнуть на переключателе ***Сложение,*** а затем щелкнуть на кнопке ***Вычислить.***

К этим двум переменным применится оператор, и в поле ***Ре­зультат*** появится число 117, как показано на рисунке 4.2.3-2.

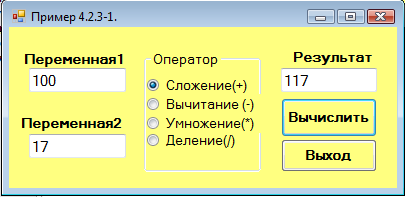


Рис. 4.2.3-2. Результат работы проекта **Пример 4.2.3-1**

1. Попрактикуйтесь в использовании операторов вычитания, умножения и деления для чисел, указанных в полях переменных. (Для вычисления каждой из формул щелкайте на ***Вычислить.***)

Результат появляется в поле ***Результат.*** Также можно поэксперимен­тировать с другими числами, введя их в текстовые поля переменных. (Если хотите, можете попробовать использовать числа с десятичны­ми точками.) Для объявления соответствующих переменных исполь­зован тип данных Double*,* так что можете использовать очень боль­шие числа.

Теперь попробуйте ввести следующие данные и посмотрите, что про­изойдет.

1. В текстовом поле ***Переменная1*** ввести число 100, в текстовом поле***Переменная2***–число 0, щелкните на переключателе ***Деление,*** а затем щелкните на ***Вычислить***.

Деление на ноль при математических вычислениях запрещено, так как оно дает бесконечный результат. Но **Visual Basic** может справить­ся с таким вычислением и отобразить в текстовом поле ***Результат*** значе­ние ***Infinity (Бесконечность)***. Способность обрабатывать некоторые ус­ловия деления на ноль является автоматически предоставляемой функцией **Visual Basic**.

1. Когда закончите изучать этот и другие тесты, необходимо щелкнуть на кнопке ***Выход***.

Программа остановится, и вернется среда разработки.

1. Теперь, чтобы узнать, как были получены такие результаты, необходимо проанализировать код программы (рис.4.2.3-3). Код использует несколько стандартных элементов управления ввода, с которыми вы экспериментировали ранее, и содержит процедуру события, которая использует переменные и операторы для вычисления простых математических формул. Програм­ма объявляет свои переменные в верхней части программного кода формы, и, таким образом, они могут быть использованы во всех процедурах событий ***Form1.***

В верхней части кода записан оператор, который объяв­ляет две переменные типа Double:

|  |
| --- |
| **' Объявляем переменные FirstNum и SecondNum**  **Dim FirstNum, SecondNum As Double** |

Тип Double был использован для того, чтобы использовать тип пере­менных общего назначения, который может работать с различными числами - целыми, числами с десятичной точкой, очень большими чис­лами, маленькими числами и тому подобным. Переменные объявле­ны в одной и той же строке с помощью сокращенной записи. Обе пере­менные FirstNum и SecondNum имеют тип Double, и они используются для хранения значений, вводимых в первом и втором текстовых по­лях соответственно.

|  |
| --- |
| **Public Class Form1**  **Dim FirstNum, SecondNum As Double**  **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**  **FirstNum = TextBox1.Text**  **SecondNum = TextBox2.Text**  **If RadioButton1.Checked = True Then**  **TextBox3.Text = CStr(FirstNum + SecondNum)**  **End If**  **If RadioButton2.Checked = True Then**  **TextBox3.Text = CStr(FirstNum - SecondNum)**  **End If**  **If RadioButton3.Checked = True Then**  **TextBox3.Text = CStr(FirstNum \* SecondNum)**  **End If**  **If RadioButton4.Checked = True Then**  **TextBox3.Text = CStr(FirstNum / SecondNum)**  **End If**  **End Sub**  **End Class** |

Рис. 4.2.3-3

Первые два оператора этой процедуры события передают данные, вве­денные в объектах текстовых полей, в переменные FirstNum и SecondNum.

|  |
| --- |
| **' Присваиваем значения текстовых полей переменным**  **FirstNum = TextBox1.Text**  **SecondNum = TextBox2.Text** |

Элемент ***TextBox*** осуществляет передачу с помощью свойства *Text*, которое принимает текст, введенный пользо­вателем, и делает его доступным для использования в программе.

После того, как значения текстовых полей присваиваются перемен­ным, процедура события определяет, какой переключатель был от­мечен, вычисляет математическую формулу и отображает результат в третьем текстовом поле. Первая проверка переключателя выглядит так:

|  |
| --- |
| **'Определяем выбранную кнопку и вычисляем**  **If RadioButton1.Checked = True Then**  **TextBox3.Text = CStr(FirstNum + SecondNum)**  **End If** |

В каждый момент времени из группы объек­тов может быть выбран только один объект переключателя. Чтобы определить, был ли выбран переключатель, проверьте значение свой­ства *Checked.* Если оно равно *True,* то переключатель был выбран. Если свойство *Checked* равно *False*, то переключатель не был выбран.

В дополнение к четырем базовым математическим операторам **Visual Basic** содержит четыре расширенных оператора, которые выполняют ***целочисленное деление*** (\), берут ***остаток от деления***(Mod),производят ***возведение в степень*** (*^*) и ***слияние (конкатенацию)*** строк (&). Эти опера­торы применяются в специальных математических формулах и прило­жениях, обрабатывающих строки. Следующий пример (небольшая мо­дификация программы Пример 4.2.3-1 – ***Базовые математические операции)*** показывает, как использовать в программе каждый из этих операторов.

**Пример 4.2.3-2.Создать проект для работы с расширенными операциями.**

1. Запустить **VS**.
2. Создать новый ***проект с именем Пример-4.2.3-2.***
3. В окне **Конструктор форм** (**Designer)** появится пустая форма ***Пример-4.2.3-2.***

На ней необходимо создать интерфейс проекта ***Пример-4.2.3-2*** (рис.4.2.3-4).

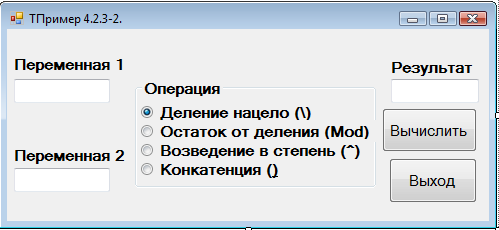


Рис.4.2.3-4

1. Программный код проекта ***Пример-4.2.3-2*** идентичен программе ***Пример-4.2.3-1*** ***Базовые математические операции***, но предназначен для тестирования другого набора операций, поэтому в событийную процедуру Button1\_Click() необходимо ввести немного измененный код, показанный на рис. 4.2.3-5.

|  |
| --- |
| **Private Sub Button1\_Click(ByVal sender As System.Object, \_**  **ByVal e As System.EventArgs) \_**  **Handles Button1.Click**  **' Присваиваем значения текстовых полей переменным**  **FirstNum = TextBox1.Text**  **SecondNum = TextBox2.Text**  **' Определяем выбранную кнопку и вычисляем**  **If RadioButton .Checked = True Then**  **TextBox3.Text = CStr(FirstNum \ SecondNum)**  **End If**  **If RadioButton2.Checked = True Then**  **TextBox3.Text = CStr(FirstNum Mod SecondNum)**  **End If**  **If RadioButton3.Checked = True Then**  **TextBox3.Text = CStr(FirstNum^ SecondNum)**  **End If**  **If RadioButton4.Checked = True Then**  **TextBox3.Text = CStr(FirstNum & SecondNum)**  **End if** |

Рис.4.2.3-5

1. Щелкнуть на кнопке ***Начать отладку (StartDebugging)*** на стандартной панели инструментов.

Эта программа отображает два текстовых поля, в которые вводятся числовые значения, группу переключателя операторов, поле, кото­рое отображает результаты, и две кнопки.

1. В текстовом поле ***Переменная1*** ввести число 9**,** а затем нажать на ***<*Tab>*.***
2. В текстовом поле ***Переменная2*** ввести число 4.

Теперь к этим двум значениям можно применить любой из дополни­тельных операторов.

1. Щелкнуть на переключателе ***Целочисленное деление***, а затем щелкнуть на кнопке **Вычислить**.

К этим двум переменным применится оператор, и в поле **Результат** появится число 2.Целочисленное деление дает в результате операции деления только целое число. Хотя 9, деленное на 4, будет 2.25, операция целочисленно­го деления возвращает только первую – целую – часть (целое число 2).

1. Щелкнуть на переключателе **Остаток от деления,** а затем на кнопке **Вычислить.**

В поле **Результат** появится число 1. Остаток от деления ***(Арифметичес­кий модуль)*** возвращает остаток (оставшуюся часть, которая не де­лится нацело) от деления одного числа на другое. Так как 9**,** деленное на 4, равно 2 с остатком 1 (2 \* 4 + 1 = 9), то результат, производимый оператором Mod*,* равен 1.

1. Щелкнуть на переключателе **Возведение в степень**, а затем щелкнуть на**Вычислить*.***

В поле ***Результат*** появится число 6561. Оператор возведения в степень (***^***) возводит число в указанную степень. Например, 9 ^ 2 равно 92, или 81. В **VisualBasic** формула 94 записывается как 9 ^4.

1. Щелкнуть на переключателе **Конкатенация**, а затем накнопке ***Вычислить.***

В поле **Результат** появится число94**.** Оператор конкатенации строк(&)объединяет две строки в одну, но не через операцию сложения. Ре­зультатом является комбинация символа «9» и символа «4». Конкате­нация строк может выполняться с числовыми переменными, однако гораздо чаще конкатена­ция используется для строковых значений или переменных. Так как переменные FirstNumиSecondNum были объявлены как имеющие тип Double***,***с помощью написанного здесь кода программы объеди­нять слова или буквы невозможно. В качестве примера попробуйте следующий тест, который приводит к возникновению ошибки и за­вершению программы.

1. В текстовом поле ***Переменная1*** ввести слово ***МТУСИ,*** а в текстовом поле ***Переменная2*** ввести слово***"УНИВЕРСИТЕТ***", убедиться, что выбран переключатель ***Конкатенация***, а затем щелкнуть на ***Вычислить***.

**VB** не сможет обработать введенные текстовые значения, в результате чего программа остановится, а на экране появится показанное здесь сообщение об ошибке (рис. 4.2.3-6).

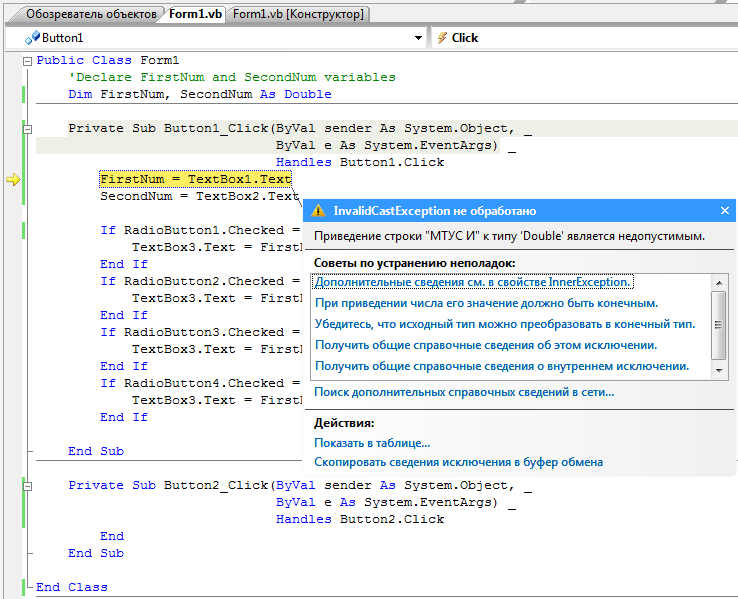


Рис. 4.2.3-6

Этот тип ошибки называется ***ошибкой времени выполнения*** *-* ошиб­кой, которая проявляется не на стадии проектирования и компиля­ции программы, а позднее, когда программа выполняется и наталки­вается на условия, которые она не знает, как обрабатывать. Преобразование из строки «**МТУСИ**» в Doubleнедопустимо и означает, что слова, которые вы ввели в тексто­вые поля (**МТУСИ** и **Университет**), не могут быть преобразованы компилятором **Visual Basic** к типу переменных Double*.* Типы Doubleмогут содержать только числа.

Как будет описано более подробно в дальнейшем, **Visual Studio** боль­ше не оставляет вас наедине с проблемой, а предоставляет диалого­вое окно, содержащее различную информацию, помогающую спра­виться с ошибкой времени выполнения. (Отладка стала одним из наиболее существенных усовершенствований в **Visual Studio**.)

1. Чтобы остановить программу, щелкнуть на кнопке ***Остановить отладку*** (***StopDebugging)***, расположенной на стандартной панели инструментов. Программа завершится и возвратит вас в среду разработки.

Теперь, чтобы увидеть, как были объявлены переменные и как исполь­зованы дополнительные операторы, посмотрите на код программы.

1. Прокрутить код в окне **Редактор кода (CodeEditor)**к самому его началу.

Вы увидите следующие комментарий и оператор программы:

|  |
| --- |
| **' Объявляем переменные FirstNum и SecondNum**  **Dim FirstNum, SecondNum As Double** |

FirstNum и SecondNum – это переменные, которые хранят числа, получаемые из объектов ***TextBox1*** и ***TextBox2****.*

1. Теперь необходимо изменить тип данных с Double на String так, чтобы можно было протестировать, как работает оператор конкатенации (&*)* строк.
2. Прокрутить код в окне **Редактор кода (CodeEditor)** вниз, чтобы увидеть, как в коде программы использованы дополнительные операторы:

Аналогично программе ***Базовые математические операции (Пример 4.2.3-1)***, эта программа загружает данные из текстовых полей и помещает их в переменные FirstNum и SecondNum*.* Затем программа проверяет, какой переключатель выб­рал пользователь, и вычисляет требуемую формулу. В этой процедуре события использованы операторы целочисленного деления (**\**), остатка от деления (Mod*),* возведения в степень   
(^) и конкатенации строк. Те­перь, когда вы изменили тип данных переменных на String*,*снова за­пустите программу и посмотрите, как работает с текстом оператор &*.*

1. Щелкнуть на кнопке ***Начать отладку*** (***StartDebugging)***.
2. Ввести в текстовом поле ***Переменная1*** слово ***МТУСИ,*** в текстовом поле ***Переменная2*** ввести слово ***УНИВЕРСИТЕТ,*** щелкнуть на значении ***Конкатенация,***  а затем щел­кнуть на кнопке приложения ***Вычислить*.**

Теперь программа объединит строковые значения и не выдаст ошиб­ки времени выполнения, как показано на рис. 4.2.3-7.

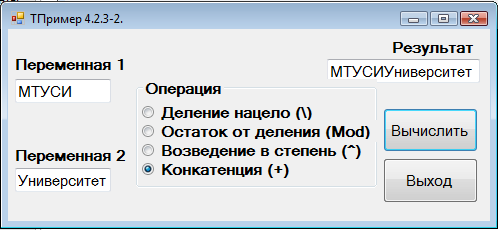


Рис. 4.2.3-7

1. Чтобы закрыть программу, щелкните на кнопке ***Выход***

Часто возникает необходимость выполнить в программе более сложные действия с числами. Например, может понадобиться округлить число, вычислить сложное математическое выражение или использовать в про­грамме случайные числа. Математические методы (функции), показанные в таблице 4.2.3-3, могут помочь в работе с числами в ваших формулах. Эти методы (функции) предоставляются **Microsoft .NET Framework**-***библиотекой клас­сов***, которая позволяет вам воспользоваться мощью операционной сис­темы **MSWindows** и выполнить множество часто возникающих задач программирования, которые требуется реализовать в проекте. **.NET Framework.** Это базовый интерфейс, который становится частью самой операционной системы **MS Windows** и устанавливается на каждый компьютер, на котором работа­ют программы **VS**.

Платформа **.NET Framework** организована в виде библиотек классов, которые могут быть включены в проекты программирования по их именам с помощью опера­тора Imports*.* Этот процесс достаточно прост, и далее мы поэкспериментируем с мате­матическими методами (процедурами) из класса System.Math среды .**NET**.

Таблица 4.2.3-3

|  |  |
| --- | --- |
| **Функция(метод)** | **Возвращаемоезначение** |
| Abs(x) | Абсолютное значение x (т.е. игнорируется знак, Abs(-5)=5 ) |
| Acos(x) | Значение арккосинуса х |
| Asin(x) | Значение арксинуса х |
| Atan(x) | Значение арктангенса x. |
| Ceiling(x) | Целое число, которое больше или равно аргументу |
| Cos(x) | Значение косинуса x |
| DivRem(x,y,r) | Значение остатка (r) от целочисленного деле­ния |
| E | е- oснование натурального логарифма – 2.71828182845905 |
| Exp(x) | Экспонента - ех |
| Fix(x) | Целую часть числа (дробная часть отбрасывается) |
| Floor(x) | Целое число, которое меньше или равно аргументу |
| Int(x) | Значение целой части **х** (ближайшее меньшее) |
| Log(x[,n]) | Логарифм по основанию n |
| Log10(x) | Десятичный логарифм |
| Max(x, y) | Максимальное из двух чисел |
| Min(x, y) | Минимальное из двух чисел |
| PI | π – 3.1415265358979 |
| Pow(x, y) | Возведение x в степень y |
| Rnd( ) | Равномерное случайное число на отрезке [0;1] |
| Round(x, n) | Округление до n знаков после запятой |
| Sign(x) | Знак числа х |
| Sin(x) | Значение синуса x |
| Sqrt(x) | Квадратный корень из х |
| Tan(x) | Значение тангенса x |
| Truncate(x) | отбрасывает дробную часть |

В таблице 4.2.3-3 представлен неполный список математических методов класса System.Math. Аргумент x в этой таблице представляет число, переменную или оператор, которые требуется передавать в метод для обработки. При использовании каких-либо из этих методов необходимо убеди­ться, что оператор

Imports System.Math

находится в окне **Редактор кода (CodeEditor)**в самом начале программного кода.

Кратко прокомментируем некоторые методы.

Функция Log(x[,n]) поддерживает один аргумент (натуральный логарифм) или два аргумента (логарифм по основанию n):

|  |
| --- |
| **Dim result As Double = Math.Log(10) ' = 2.30258509299405 result = Math.Log(1000,10) ' = 3**  **result = Math.Log10(1000) '=3** |

Методы Min(x,y) и Max(x,y) вычисляют минимальное и максимальное значения, соответ­ственно, и используются для работы со всеми числовыми типами данных.

Метод Floor(x) возвращает целое число, которое меньше или равно аргументу мето­да, в то время как метод Ceiling(x) возвращает целое число, которое больше или равно аргументу метода. Метод Truncate(x)просто отбрасывает дробную часть и, следовательно, работает отлично от метода Ceiling(x), если аргумент отрицателен:

|  |
| --- |
| **result = Math.Ceiling(2.5) ‘ = 3**  **result = Math.Floor(-1.5) ‘ = -2**  **result = Math.Truncate(-1.5)‘ = -1** |

Метод DivRem(x,y,r) выполняет целочисленное деле­ние и возвращает остаток (r) в третий аргумент:

|  |
| --- |
| **Dim remainder As Long**  **Dim result As Long= Math.DivRem(20, 3, r) ‘=2** |

Напомним, если логарифм имеет не натуральное основание, то его можно преобразовать к натуральному логарифму по формуле Logab = Log(b)/Log(a).

Значение аргумента х в тригонометрических функциях дается в радианах.

Обычно по традиции, используемые в языках программирования библиотечные функции, называют ***«встроенными функциями»***. На самом деле они представляют собой стандартные методы (процедуры) из библиотеки **VB** или из библиотеки классов **Microsoft .NET Framework**, которым передается один или несколько аргументов и которые возвращают вычисленное значение через имя функции.

Обращение ко всем функциям всегда происходит по их именам, за которыми в скобках следуют аргументы. Вычисле­ние функций в арифметическом выражении предшествует выполне­нию арифметических операций. Аргументом встроенной функции может быть любое ариф­метическое выражение. Кроме того, если в начале программного кода отсутствует оператор Imports System.Math, то перед обращением к функциям необходимо указать имя класса с точкой, например, Math.Sin(x).

***Обратите внимание, что в таблице 4.2.3-3 описаны две системные константы (*E** и **PI*).***

**Пример 4.2.3-3.Создать проект, использующий метод класса *System.Math* для вычисления квадратного корня.**

1. В меню **Файл (File)** щелкнуть на ***Создать проект (New Project).***

Появится диалоговое окно **Создать проект (New Project*).***

1. Создать новый проект типа ***Visual Basic Windows Application*** с именем ***Пример 4.2.3-3*** (***Встроенные математические функции)***.

Будет создан новый проект, и в окне **Конструктор (Designer)** появится пустая форма.

1. Щелкнуть на элементе управления ***Button*** на вкладке ***Windows Forms*** окна **Панель элементов (Toolbox)** и создать в верхней части формы объект кнопки.
2. Щелкнуть в окне **Панель элементов (Toolbox)** на элементе управления ***TextBox*** и отобразить под объектом кнопки два объекта текстового поля.
3. Установить свойство *Text* объекта кнопки равным Квад­ратный корень.
4. Чтобы вызвать окно **Редактор кода (Code Editor)**, дважды щелкнуть мышью на объекте кнопки.
5. В самом верху окна **Редактор кода (Code Editor)**, над оператором Public Class Form1, ввести сле­дующий оператор программы:

Imports System.Math

Оператор Imports добавляет в проект библиотеку объектов, свойств и методов. Этот оператор должен быть первым оператором программы - он должен идти даже перед переменными, которые объявляются для всей формы, и перед оператором Public Class Form1, который в **VB** создается автоматически. Конкретной библиотекой классов, которую мы выбрали, является System.Math*,*которая является коллекцией объек­тов, свойств и методов, предоставляемой .**NET Framework** для выпол­нения математических операций.

1. Спуститься в окне **Редактор кода (Code Editor)**вниз и в процедуру события Button1\_Click()между операторами Private Sub и End Sub добавить следующий код

|  |
| --- |
| **Dim Arg As Double**  **Dim Result As Double**  **Arg = TextBox2.Text**  **Result = Sqrt(Arg)**  **TextBox1.Text = Result** |

Эти операторы объявляют переменную типа Double с именем Result*,* используют метод Sqrt( )для вычисления квадратного корня из числа, введенного в поле ***TextBox2.Text,*** и присваивают значение переменной Result свойству *Text* объекта текстового поля ***TextBox1*** так, что в нем отображается результат вы­числения.

1. Чтобы сохранить изменения, щелкнуть на кнопке ***Сохранить Все (SaveAll)*** стандартной панели инструментов. В качестве места сохранения укажите необходимую папку***.***
2. Щелкнуть на кнопке ***Начать отладку****(****Start Debugging*)** на стандартной панели инструментов.

Программа запустится в **IDE*.***

1. Ввести число **625** в поле ***TextBox2.Text***.
2. Щелкнуть на кнопке **Квадратный корень**.

**Visual Basic** вычислит квадратный корень из 625 и отобразит в тек­стовом поле результат этого вычисления (рис. 4.2.3-8).

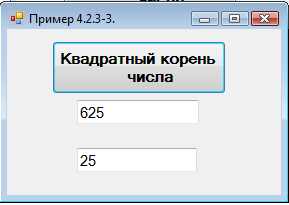


Рис. 4.2.3-8

1. Чтобы завершить выполнение программы, необходимо щелкнуть на кнопке ***Закрыть***.

Чтобы использовать в программе какой-либо класс **.NET Framework**, следует включить оператор Imports и указать требуемую библиотеку классов. Эта методика используется для любых классов **.NET Framework**.

**Пример 4.2.3-4. Записать математическую формулу в виде арифметических выражений**.



В **VB** данной формуле соответствует следующее арифметическое выражение:

Y=Abs (3.2\*Cos(x) + Exp(x+1) + 3)/(2\*b - Log(x)) .

**Пример 4.2.3-5. Записать выражение для вычисления периода электромагнитных колебаний, возникающих в колебательном контуре.**

Проведем формализацию, то есть запишем математическое выражение, описывающее электромагнитные колебания:

.



Запишем данную математическую формулу в виде арифметического выражения:

2\*PI\*Sqrt(L\*C).

Большая часть операций и «встроенных» функций, выполняемых над числами, которые находятся в классе System.Math, уже рассмотрена.

Кратко рассмотрим члены базовых числовых классов, а именно, методы и поля (свойства), которые будут полезны при разработке проектов лабораторных работ.

Для всех числовых типов **VB** и всех соответствующих классов **.NET**(см. таблицу 4.2.2-1) определены статические поля MinValue и MaxValue. Эти поля возвращают наименьшее и наибольшее значение, которое можно присвоить переменной соответствующего типа:

|  |
| --- |
| **'Наибольшее значение, которое можно хранить в переменной типа Double. TextBox1.Text= Double.MaxValue ' = 1.79769313486232Е+308** |

Числовые классы, которые поддерживают значения с плавающей точкой, а имен­но, классы Single и Double, имеют несколько дополнительных статических свойств, предназначенных только для чтения. Свойство Epsilon возвращает наименьшее по­ложительное (не равное нулю) число, которое можно хранить в переменной данно­го типа:

|  |
| --- |
| **TextBox1.Text= Single.Epsilon ‘= 1.401298E-45**  **TextBox2.Text= Double.Epsilon '= 4.94065645841247E-324** |

Поля Negativelnfinity и Positivelnfinity возвращают константу, которая представляет бесконечные значения, а поле NaN возвращает константу, которая представляет значение Not-a-Number (не число), (т.е. значение, которое получается, например, при извлечении квадратного корня из отрицательного числа). В некоторых случаях бесконечные значения можно использовать в выражениях. Например, при делении числа на бесконечность получается ноль.

В классах Single и Double, к тому же, определены статические методы, которые позволяют проверять то, что переменные их типа содержат специальные значения, такие как IsNegativelnfinity, IsPositivelnfinity и IsNaN.

Кроме методов и свойств числовых классов имеется большое число «встроенных» классов, в которых определено множество полезных команд, ме­тодов и свойств. Рассмотрим

некоторые из них.

В классе **Microsoft.Visual Basic.Interaction** определено множество полезных команд и методов взаимодействия программ, включая Beep( ), InputBox( ), MsgBox( ) и другие. Эти методы определены глобально, поэтому при их использовании можно не указывать имя класса.

Кроме числовых данных, при написании программного кода активно используются строковые и символьные данные. Подробно они будут рассмотрены в теме 4.9, где будут описаны функции **VB** и описаны свойства и методы классов **System.String** и **System.StringBuilder.NET**. Здесь кратко рассмотрим строковые типы данных, так как они будут использоваться почти во всех программных кодах проектов для ввода и отображения данных на форме.

Во-первых, единственная операция со строкой– это операция конкатенации –& (+):

|  |
| --- |
| **Dim s As String = "ABC"**  **s = s & "D"‘Результат строка - ABCD** |

Объявленная строковая переменная, которой еще не присвоено никакое значение, содержит значение Nothing или, точнее, не указывает на какую-либо последовательность символов в памяти. Такая переменная отличается от переменной, которая равна пустой строке:

|  |
| --- |
| **Dim s As String =**""**'Длина строки равна 0**  **Dim t As String’Длина строки не определена, а значение равно Nothing** |

Необходимо обратить внимание, что **VB** не предлагает способа ввода в строку непечатных символов, например, символов табуляции или новой строки. Строковые константы предыдущих версий **Visual Basic**, такие как vbCrLf, vbTab и другие, поддерживаются как поля класса **Microsoft.VisualBasic.Constants.** Этот класс обозначен как глобальный, поэтому в программный код необязательно включать имя этого класса.

В качестве альтернативы можно использовать поля класса **Microsoft.VisualBasic. ControlChars.** Поскольку этот класс не объявлен как глобальный, имя класса нужно указывать в программном коде (если только не использовалось выражение Imports для импорта всего этого класса). Имена констант в этом классе такие же, как и в классе **Microsoft.VisualBasic.Constants** за исключением того, что в них не используется префикс vb.

Класс **ControlChars** содержит константы Back, Cr, CrLf, FormFeed, NewLine, NullChar, Quote, Tab и VerticalTab. Наиболее используемые константы представлены в таблице 4.2.3-4.

Таблица 4.2.3-4

|  |  |  |  |
| --- | --- | --- | --- |
| **Константы**  **Класса Constants** | **Константы**  **Класса ControlChars** | **Код символа** | **Описание** |
| **vbCrLF** | **CrLF** | Chr(13)+Chr(10) | Символы возврата каретки и перевода строки |
| **vbCr** | **Cr** | Chr(13) | Символ возврата каретки |
| **vbLf** | **Lf** | Chr(10) | Символ перевода строки |
| **vbNewLine** | **NewLine** | Chr(13)+Chr(10) | Символ новой строки |

Чтобы включить эти символы в строку, необходимо использовать конкатенацию строк. Например:

|  |
| --- |
| **Dim s As String = "МТУСИ"& ControlChars.NewLine**  **s = s & "Кафедра" & "ВМ и П" & vbCrLF** |

Класс **Microsoft.VisualBasic.Strings** содержит большинство строковых функций язы­ка **Visual Basic**, включая Asc, Chr, ChrW, Filter, Format, FormatCurrency, FormatDateTime, FormatNumber, FormatPercent, InStr, InStrRev, Join, LCase, Left, Len, LTrim, Mid, Replace, Right, RTrim, Space, Split, StrComp, StrReverse, TrimиUCase. Эти методы определены глобально, поэтому для их вызова необязательно указывать полное имя класса, хотя нужно указывать полное имя связанных с ними констант.

### 4.2.4. Преобразование данных в выражениях и операторе присваивания

У многих программистов при работе на **VB** часто возникают проблемы с правильным преобразованием вещественных чисел или же строк, содержащих числа, к целым числам.

Рассмотрим более подробно проблему ***преобразования данных в выражениях и операторах присваивания.***

В процессе рассмотрения примеров мы уже встречались с проблемой преобразования данных. При работе с данными в **VB**  нужно иметь в виду поддерживаемые в **VS**  преобразования.

Во-первых, это ***расширяющие, не приводящие к потерям данных, преобразования***, например,

Integer →Long, Decimal, Single, Double.

Во-вторых, это ***сужающие – преобразования (с возможностью потерей данных) в тип данных, содержащий меньшие значения данных***, например,

Single→Integer;

Double→Integer***.***

Потери данных, связанных с сужающими преобразованиями, можно избежать, используя установки

Option Strict On

Option Explicit On.

Напомним, что первая установка (оператор Option Strict On) разрешает все расширяющие преобразования, а все сужающие преобразования заставляет программиста делать осознанно, т.е. явно вызывая нижеописанные функции преобразования типов. Вторая установка (оператор Option Explicit On) принуждает программиста объявлять все переменные до их использования в программе.

***Обратите внимание, что в лабораторных работах приведенные выше установки обязательно необходимо использовать.***

Приведем список функций языка **Visual Basic**, являющихся ключевыми словами, и, которые позволяют осуществлять явные преобразования типов:

* CBool(x)  **-** возвращает булево (логическое) значение. В качестве параметра x функции может быть передано условие или числовое значение.
* CByte(x) - возвращает значение типа Byte. В качестве параметра x передается соответствующая строка или числовое значение.
* CChar(x) - возвращает символ. В качестве параметра x передается целое число, находящееся в промежутке от 0 до 65 535.
* CDate(x) - возвращает значение типа Date. В качестве параметра x можно использовать любое принятое в операционной системе обозначение даты.
* CDbl(x) - преобразовывает значение параметра x к типу Double.
* CDec(x) - применяется для преобразования параметра x к типу Decimal.
* CInt(x) - преобразовывает значения параметра x к целочисленному типу Integer. Если в качестве параметра x передано дробное число, то дробная часть просто округляется по правилам математики.
* CLng(x) - возвращает значение типа Long. Обработка параметра x производится по образцу функции CInt(x).
* CShort(x) - преобразовывает значение параметра x к типу Short.
* CSng(x)- применяется для преобразования значения параметра x к типу Single.
* CStr(x) - используется для преобразования данных в строковый тип String.

Кроме «внутренних» функций языка **Visual Basic** для явных преобразований можно использовать классы и модули как **VB,** так и **.NET.**

В модуле **Microsoft.VisualBasic.Conversion** определено множество методов преобразования значений, включая:

* **Int( ) и Fix( )** –функции возвращают целую часть числа. Предполагается, что аргумент имеет тип Double или является числовым выражением. Возвращается тип Double. А уже этот Double затем можно преобразовать в Integer или Long. Главное, чтобы число уместилось в диапазон нужного вам типа. Разность при использовании Int( ) и Fix( ) заметна только лишь при отрицательном значении аргумента. Дело в том, что Int( ) всегда возвращает ближайшее меньшее целое число, а Fix( ) просто отбрасывает дробную часть. То есть, для числа 4.5 и Int( ) и Fix( ) вернут 4, тогда как для числа -4.5, Int( ) вернет – 5, а Fix( ) – -4.
* **Val(x)** – преобразует значение параметра x строкового типа String к числовому значению типа Double.
* **Str(x)** – преобразует число в строку, причем перед положительным числом вставляется пробел.

Приведем несколько примеров:

|  |
| --- |
| **Dim Number As Single**  **Number = Int(99.8) ' Возвращает 99**  **Number = Fix(99.8) ' Возвращает 99**  **Number = Int(-99.8) 'Возвращает -100**  **Number = Fix(-99.8) ' Возвращает -99**  **Number = Int(-99.2) ' Возвращает -100**  **Number = Fix(-99.2) ' Возвращает -99** |

В модуле **Microsoft.VisualBasic.String** определено множество методов работы со строками, в том числе:

* Format(x, *строка символов*) - преобразует числовое значение параметра xв заданный формат, который определяется *строкой символов*.

Если в качестве параметра функции передается булево значение, то возвращается строка True или False. Если передаются данные типа Date, то функцией возвращается строка, содержащая обозначение даты в принятом для системы формате. Любое числовое значение преобразовывается в символьное представление данного числа.

***Функции CLng( ) и CInt( ).*** Эти функции преобразовывают числовые аргументы в целое число типа Long и Integer соответственно. Дробная часть при этом округляется по правилам математики. Таким образом, если исходное число будет равно 1.5, то результатом преобразования при помощи CInt( ) или CLng( ) будет целое число 2. А если аргументом будет 1.4, то в результате получим 1. Знак числа при этом роли не играет, то есть -1.5 будет преобразована в -2, а -1.4 – в -1.

При преобразовании из строки, содержащей число, важно помнить, что функции преобразования используют национальные настройки для получения символа разделения целой и дробной частей числа. Поэтому, если у вас разделителем целой и дробной частей числа является точка «.», а в строковом представлении числа будет фигурировать запятая, то вы получите ошибку несовпадения типов при попытке преобразовать такую строку в целое число (кстати, эта ошибка возникнет и при попытке использования CDbl( ) или CSng( ) для преобразования строки в вещественное число, а также и при использовании Int( ) или Fix( )).

***Функции Val( ), CDbl( ) и CSng( ).*** При использовании этих функций имеются свои нюансы. CDbl( ) и CSng( ) аналогичны CLng( ) и CInt( ). CDbl( ) преобразовывает аргумент в вещественное число типа Double, а CSng( ) соответственно в Single. Естественно, надо обращать внимание на попадание аргумента в диапазон представления и на правильность разделителя целой и дробной частей. Для функций CDbl( ) и CSng( ) разделителем целой и дробной части числа обязательно должен быть символ запятая «,», а для функции Val( ) разделителем целой и дробной частей числа является точка «.». Функция Val( ) возвращает число, содержащееся в строке - аргументе как числовое значение типа Double. Данная функция будет пытаться преобразовать строку до тех пор, пока не встретит символ, который не может быть распознан как часть числа. В частности, функция прекрасно принимает точку в качестве разделителя, а вот пробелы просто игнорирует.

Таким образом, функция Val( )преобразует строки в соответствии с таблицей 4.2.4-1.

Таблица 4.2.4-1

|  |  |  |
| --- | --- | --- |
| **Строка** | **Число** | **Примечания** |
| "     1 2 34569" | 1234569 | После отбрасывания пробелов получаем целое число 1234569. |
| "     1 2 345,69" | 12345 | После отбрасывания пробелов функция пытается разобрать число и останавливается на символе «,»,и возвращает 12345. |
| "     1 2 345.69" | 12345.69 | В отличие от предыдущего случая, здесь символ «.» является законным разделителем целой и дробной частей числа, поэтому результатом будет 12345.69. |
| "    12 34569 number" | 1234569 | Здесь разбор числа будет остановлен на первой встреченной букве «n», а все, что до нее, преобразовывается в число 1234569. |

Ещё раз обратите внимание, что функции Val(x) и Str(x) при преобразовании в качестве разделителя целой и дробной частей числа функции Val(x) распознают только точку, а функция СDbl( ) и ей подобные – запятую, но та и другая функция игнорирует пробелы в числе.

Рассмотрим несколько примеров:

|  |
| --- |
| **Dim aDbl As Double**  **Dim anInt As Integer**  **aDbl = 2345.5678**  **anInt = CInt(aDbl) ' Переменная anInt будет равна 2346.**  **Dim aDbl1, aDbl2 As Double**  **Dim aLng1, aLng2 As Long**  **aDbl1 = 25427.45 : aDbl2 = 25427.55**  **aLng1 = CLng(aDbl1) ' Переменная aLng1 будет равна 25427.**  **aLng2 = CLng(aDbl2) ' Переменная aLng2 будет равна 25428.**  **Dim aDouble1, aDouble2 As Double**  **Dim aSingle1, aSingle2 As Single**  **aDouble1 = 75.3421105 : aDouble2 = 75.3421567**  **aSingle1 = CSng(aDouble1) ' Переменная aSingle1 будет равна 75.34211.**  **aSingle2 = CSng(aDouble2) ' Переменная aSingle2 будет равна 75.34216.**  **Dim aDouble As Double**  **Dim aString As String**  **aDouble = 437.324**  **aString = CStr(aDouble) ' Переменная aString будет равна " 437.324".**  **aDouble = CDbl(**" **437,324**")**' Переменная aDouble будет равна 437,324.**  **aDouble = Val(**" **437.324**")**' Переменная aDouble будет равна 437.324.** |

***Округление.*** Для округления используется функция Round( ). Она возвращает число с заданным количеством цифр после запятой. Однако есть один нюанс. Функция Round( ) именно округляет (что собственно видно из названия). То есть, вызов Round(1.15, 1) даст результат 1.2. А иногда возникает ситуация, когда необходимо отбросить лишние цифры, то есть для числа 1.15 получить 1.1. Тогда надо пользоваться конструкцией вида: Fix(10\*n)/10.

Здесь вместо 10 должна использоваться степень 10, соответствующая количеству оставляемых цифр после запятой. Универсальный вариант будет таким:

|  |
| --- |
| Fix(10^ *ЧислоЦифрПослеДесТочки* \* n) / 10^*ЧислоЦифрПослеДесТочки* |

Обратите внимание, что здесь используется именно функция Fix( ), а не Int( ), которая для отрицательных чисел будет работать некорректно.

В модуле **Microsoft.VisualBasic.String** определенно множество методов работы со строками, в том числе Format(x, *строка символов*) - преобразует числовое значение параметра xв заданный формат, который определяется *строкой символов*. Если в качестве параметра функции передается булево значение, то возвращается строка True или False. Если передаются данные типа Date, то функцией возвращается строка, содержащая обозначение даты в принятом для системы формате. Любое числовое значение преобразовывается в символьное представление данного числа.

Если параметр *строка символов* опущен, то функция производит тоже действие, что и функция [Str](http://www.kbyte.ru/Refbooks/show.aspx?id=261)( ) для численного выражения. Положительные числа, преобразуемые в строки, теряют пробел слева, зарезервированный для знака. Если положительное число преобразовано в строку функцией [Str](http://www.kbyte.ru/Refbooks/show.aspx?id=261)( ), то пробел слева сохраняется. Например,

|  |
| --- |
| **(Format(2, "000000")) 'возвращает 000002 (Format(10, "0.00"))'возвращает 10.00** |

Для всех числовых типов **VB** и всех соответствующих классов **.NET** определен метод ToString(x), который преобразует числовое значение в строку. Этот метод особенно по­лезен, когда строковое представление числа нужно добавить к другой строке:

|  |
| --- |
| **Dim myValue As Double = 123.45**  **Dim res As String = "The final value is"& myVaIue.ToString()** |

Метод ToString(x) учитывает региональные настройки и по умолчанию используются настройки, связанные с текущим потоком. Например, в качестве десятичного разделителя он использует точку, если национальные настройки – это русский язык.

|  |
| --- |
| **'Вывод PI с указанием 5 знаков после десятичной точки (всего 6 цифр)**  **Dim d As Double = Math.PI**  **TextBox1.Text= d.ToString("G6") ‘рез-т 3.14159 - всего 6 цифр** |

Все числовые классы поддерживают перегруженную форму метода ToString( ), которая позволяет использовать строку форматирования целых действительных чисел в однотипные строки:

|  |
| --- |
| **Dim intValue As Integer = 12345**  **TextBox1.Text = intValue.ToString("##,##0.00")‘= 12 345.00**  **Dim sngValue As Single = 12345.5**  **TextBox2.Text = sngValue.ToString("##,##0.00")'= 12 345.50** |

Запятая в строке форматирования означает отделение каждых трех цифр в числе пробелом, а точка – представление дробной части числа.

Для явного преобразования любого числового типа в строку, кроме функций Str( ) и CStr( ), можно использовать метод ToString( ) класса Convert:

|  |
| --- |
| **Dim Число As Double**  **Число = 3.14**  **Dim Строка As String = Convert.ToString(Число)** |

Напомним также, что обратное преобразование из строки в число можно выполнить или с помощью функций Val( ), CDbl( ), CInt( ), или с помощью соответствующих методов класса Convert – ToDouble( ), ToInt32( ) и других. Причем список методов можно увидеть с помощью технологии ***IntelliSense*** при вводе точки после Convert(рис. 4.2.4-1):

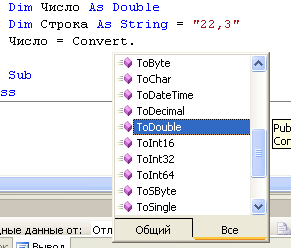


Рис. 4.2.4-1

### 4.2.5. Задачи для самостоятельного решения по теме «Основные средства языка программирования VB .NET»

|  |  |
| --- | --- |
| 1)  2)  3)  4)  5)  6)  7)  8)  9)  10)  11)  12)  13)  14)  15)  16)  17)  18)  19)  20)  21)  22)  23)  24)  25)  26)  27)  28)  29)  30)  31)  32)  33)  34)  35) | Найти площадь треугольника со сторонами a, b, c (a, b, c>0).  Найти среднее арифметическое и среднее геометрическое чисел a, b, c, d.  Вычислить расстояние от точки с координатами (x, y) до точки (0, 0).  Вычислить синус от x градусов.  Найти площадь поверхности куба (длина ребра равна а).  Найти радиус описанной сферы куба (длина ребра равна а**)**.  Найти координаты точки пересечения двух прямых, заданных уравнениями    a1x+b1y+c1=0 и a2x+b2y+c2=0 (прямые не параллельны).  Вычислить дробную часть среднего геометрического трёх вещественных чисел.  Дана длина ребра куба. Найти объем куба и площадь его боковой поверхности.  Дан радиус окружности. Найти длину окружности и площадь круга.  Известны объем и масса тела. Определить плотность материала этого тела.  Найти площадь кольца по заданным внешнему и внутреннему радиусами.  Вычислить период колебания маятника длины L.  Даны катеты прямоугольного треугольника. Найти его периметр.  Даны основания и высота равнобедренной трапеции. Найти ее периметр.  Найти площадь прямоугольной рамки, внутренняя сторона которой равна5, а внешняя – заданному числу г (г>5).  Найти площадь равнобочной трапеции с основаниями a и b и углом α при большом основании.  Треугольник задан длинами сторон. Найти радиусы вписанной и описанной окружностей.  Треугольник задан координатами своих вершин. Найти периметр и площадь треугольника.  Найти площадь сектора, радиус которого равен 13.7, а дуга содержит заданное число радиан ω.  Даны действительные числа a, b, c**.** По трем сторонам с длинами a, b, c можно построить треугольник. Найти углы треугольника.  Найти гипотенузу и площадь прямоугольного треуголь­ника по двум данным катетам.  Вычислить сумму членов арифметической прогрессии, зная ее первый член, разность прогрессии и число чле­нов.  Даны два действительных положительных числа. Найти среднее арифметическое и среднее геометрическое этих чисел.  Даны гипотенуза и катет прямоугольного треугольника. Найти второй катет и радиус вписанной окружности.  Определить время падения камня на поверхность земли с высоты h.  Определить силу притяжения F между телами массы m1 и m2, находящимися на расстоянии r друг от друга.  Треугольник задан величинами своих углов и радиусом описанной окружности. Найти стороны треугольника.  Определить время, через которое встретятся два тела, равноускоренно движущиеся навстречу друг другу, если известны их начальные скорости, ускорения и начальное расстояние между ними.  Найти площадь равнобочной трапеции с основаниями a и b и углом α при большом основании.  Даны два действительных числа. Найти среднее арифметическое этих чисел и среднее геометрическое их модулей.  Даны три числа. Найти среднее арифметическое кубов этих чисел и среднее геометрическое модулей этих чи­сел.  Вычислить периметр и площадь прямоугольного треугольника по длинам двух его катетов.  Дана масса в килограммах. Найти число полных тонн в ней.  Дано расстояние в метрах. Найти число полных километров в нем. |

### 4.2.6. Тестовые задания по теме «Основные средства языка программирования VB .NET»

1. **Синтаксис языка программирования – это**
2. набор правил истолкования отдельных языковых конструкций
3. грамматические правила языка
4. правила компиляции операторов языка
5. **Алфавит языка программирования – это**
6. переменные и операторы языка
7. множество символов, из которых строятся все конструкции языка
8. множество идентификаторов
9. **Идентификаторы – это**
10. имена лексических элементов языка
11. имена переменных
12. ключевые слова
13. имена операторов
14. **Ключевые слова – это**
15. идентификаторы, зарезервированные для специального использования
16. операторы языка
17. константы языка
18. операторы языка
19. **Для ввода данных в схемах алгоритма используют фигуру**
20. ромб
21. прямоугольник
22. треугольник
23. параллелограмм
24. **Внутреннее представление данных в памяти компьютера определяется**
25. типом данных
26. системой счисления
27. системой программирования
28. типом транслятора
29. **Последовательность символов, заключенная в двойные кавычки, называется**
30. строковой переменной
31. строковой константой
32. числовой переменной
33. числовой константой
34. **Величина, не изменяющая своего значения в ходе выполнения программы, называется**
35. функцией
36. переменной
37. константой
38. оператором
39. **Величина, к которой обращаются по имени, принимающая различные значения в ходе выполнения программы, называется**
40. строкой
41. символом
42. константой
43. переменной
44. **Показательная функция записывается как**
45. **Exp(x)**
46. **Abs(x)**
47. **Sqrt(x)**
48. **Atan(x)**
49. **Из перечисленных действий наивысший приоритет имеет**
50. сложение
51. возведение в степень
52. вычитание
53. умножение
54. **В качестве аргумента встроенной математической функции можно применять**
55. число
56. логическую операцию
57. любое арифметическое выражение
58. переменные
59. все ответы верные
60. **Операция Mod определяет**
61. модуль числа
62. квадратный корень числа
63. знак числа
64. остаток от деления
65. **Числовые переменные могут быть**
66. целыми и вещественными
67. целыми и символьными
68. простыми и сложными
69. нет верного ответа
70. **Наибольшее целое число, не превосходящее аргумент, возвращает функция**
71. **Int(Х)**
72. **Fix(X)**
73. **Cint(X)**
74. **Sign(X)**
75. **Ошибки, связанные с неправильной записью конструкций языка программирования, называются**
76. логическими
77. орфографическими
78. математическими
79. синтаксическими
80. **Запись TextBox1.Text=CStr(у) предназначена**
81. для вывода в текстовое поле формы значения переменной у
82. для ввода данных с клавиатуры
83. для вывода числовых данных
84. для печати результатов на принтере
85. **Оператор присваивания изменяет**
86. имя переменной
87. значение переменной
88. тип переменной
89. ничего не изменяет
90. значение и тип одновременно
91. **Тип переменной, определяющей количество участников Олимпиады**
92. целый
93. вещественный
94. символьный
95. любой из вышеперечисленных
96. **Синтаксически неправильная запись**
97. **X = Y \* Sin( X) ^2 + 4**
98. **X = Y \* Sin ^ 2 (X) + 4**
99. **X = Y \* Sin( X ^ 2 ) + 4**
100. **X = Y ^2 \* Sin( X) + 4**
101. **X = Y^(2\*Sin(X) + 4)**
102. **Правильный порядок выполнения операций в арифметическом выражении**
103. возведение в степень, изменение знака числа, целочисленное деление, деление, вычитание
104. изменение знака числа, возведение в степень, деление, целочисленное деление, вычитание
105. возведение в степень, изменение знака числа, деление, целочисленное деление, вычитание
106. любой порядок
107. **Переменная, объявленная с помощью оператора Dim в разделе глобальных объявлений формы, будет доступна во всех**
108. процедурах формы
109. процедурах всех форм и модулей кода
110. Результат работы фрагмента программы

|  |
| --- |
| **…**  **Dim n,m As Integer**  **Dim a, t, b As Single**  **a =24 : b=29 : n=a: m=b: t=a+n**  **TextBox1.Text="n="& n : TextBox2.Text= "m="& m : TextBox3.Text= "t="& t**  **…** |

1. n=3 m=3 t=53
2. n=24 m=29 t=48
3. n=2 m=2 t=44
4. n=24 m=29 t=44