

ОТЛАДКА

1. Введение

Различают ошибки программы:

- Синтаксические (состоят в нарушении правил синтаксиса).
- Логические (неправильный алгоритм).
- Периода выполнения программы (их называют исключениями).

О синтаксических ошибках среда сообщает на этапе ввода программного кода. Те части кода, где обнаружены синтаксические ошибки, среда Integrated Development Environment системы Microsoft Visual Studio 2008 (сокращенно IDE) подчеркивает синей волнистой линией. Это упрощает обнаружение и исправление ошибок.

Логические ошибки – это неправильный результат работы программы или зацикливание. Чтобы убедиться, что программа работает правильно, выполняют ее тестирование, которое заключается в прогоне программы на тестах. Тест – это некоторый набор исходных данных, для которого результаты работы программы вычисляют вручную. Затем на этом же наборе исходных данных получают фактические результаты работы программы. Если между предполагаемыми и фактическими результатами есть расхождение – это сигнал того, что программа работает неправильно. Найти логическую ошибку – это значит найти то место в программном коде, где дело идет не так, как ожидалось. Причиной может быть как ошибка в алгоритме, так и опечатка в выражении. Для выявления причин логической ошибки и ее устранения IDE предлагает некоторые эффективные инструменты, которые позволяют:

1. После запуска программы остановить выполнение кода в месте, вызывающем подозрение.

2. Продолжить выполнение кода по одной инструкции (по шагам).

3. После выполнения каждого шага контролировать значения данных.

IDE может находиться в одном из трех состояний:


- а) Design (разработка),

- б) Running (выполнение),

- в) Debugging (отладка).

Нахождение среды в состоянии выполнения или отладки указывается в заголовке проекта в скобках справа от имени проекта.


2. О работе в режиме отладки


В режиме отладки выполняется поиск логических ошибок. Переход в режим отладки из режима выполнения выполняется нажатием клавиш **Ctrl+Break** или щелчком на кнопке  Break стандартной панели инструментов. Недостаток этого способа состоит в том, что неизвестно, после выполнения какой инструкции осуществится переход из режима выполнения в режим отладки.


Чаще применяется другая возможность переключения приложения в режим отладки. Это возможно благодаря установке точки останова (Breakpoint). Точка останова – это выделенная инструкция программы, на которой автоматически останавливается выполнение программы. По достижении этой строки программы IDE переходит в режим отладки. Чтобы установить точку останова нужно щелкнуть левой кнопкой мыши на полосе индикатора (серая вертикальная полоса слева от программного кода). В этом месте появится красная точка и находящаяся рядом строка закрашивается красным цветом. Точки останова предназначены для принудительной остановки программы в нужном месте, где наиболее вероятно по мнению программиста находится ошибка, и перехода IDE в режим отладки.

В режиме отладки становятся доступными все средства отладки. В этом режиме IDE особым образом выделяет строку, которая должна выполняться следующей. Сама строка выделяется желтым цветом, а на полосе индикатора рядом с ней появляется желтая стрелка. Если выполнение программы прерывается в точке останова, то оба выделения комбинируются.

Существует несколько команд пошагового выполнения программы в режиме отладки. Эти команды можно выполнить, пользуясь стандартной панелью инструментов:

1. Шаг с заходом (Step into). Для выполнения следует нажать кнопку стандартной панели инструментов  Step into. При пошаговом выполнении инструкции кода выполняются по одной. После выполнения инструкции маркер перемещается на одну строку.

2. Шаг с обходом (Step Over). Шаг с обходом подобен шагу с заходом. Различие проявляется только при вызове текущей процедурой других процедур. Если при шаге с заходом осуществляется переход в вызываемую процедуру, то шаг с обходом выполняет все инструкции тела процедуры как единичный оператор. Шаг с обходом выполняется нажатием кнопки стандартной панели инструментов  Step Over.

3. Шаг с выходом (Step Out). Команда Step Out позволяет выполнить оставшуюся часть текущей процедуры. Для вызова этой команды можно воспользоваться кнопкой стандартной панели инструментов  Step Out. Если текущая строка находится в вызванной процедуре, то с помощью команды Step Out оставшаяся не выполненная часть кода процедуры будет выполнена за один раз до конца.

Кроме контроля хода выполнения программы важной задачей инструментов отладки VB является обеспечение возможности проверки на каждом шаге значений данных. Контроль значений возможен только в режиме отладки. Более того, контролируемое выражение доступно только в определенных местах, например, значение локальной переменной можно проверить только в процедуре, в которой она объявлена. К инструментам контроля относятся окна: Tooltip (подсказки), Autos (видимые переменные), Locals (локальных переменных), Watch (контрольных значений), Immediate (непосредственное выполнение). Далее ограничимся рассмотрением только двух из этих инструментов – Tooltip и Watch:

1. Окно подсказки Tooltip

Самый простой вариант просмотра значения переменной или выражения – использование окна Tooltip. Для открытия этого окна достаточно установить курсор мыши на соответствующей переменной в окне кода. Если необходимо увидеть значение выражения, его следует выделить, а затем установить на нем курсор мыши.

2. Окно контрольных значений Watch

Для открытия этого окна в режиме Running (выполнение) или в режиме Debugging (отладка) следует выполнить команду меню Debug, Window, Watch1. Можно открыть несколько (до четырех) таких окон. Объекты, значения которых Вы собираетесь контролировать, следует в режиме отладки выделить и перетащить в это окно из окна программного кода или ввести в окне Watch. Можно перетащить и выражение или управляющий элемент, предварительно выделив его имя. В окне Watch (рис. 1) можно наблюдать значения переменных, выражений и свойств объектов.

Name	Value	Type
i	2	Integer
Button1	{Text = "Подсчитать гласные"}	System.W
s.Length - 1	311	Integer
s.Substring(i, 1)	"л"	String
k	1	Integer

Рис. 1. Окно Watch

3. Обработка исключений

Программисты не любят слово ошибка. Возможно, по этой причине ошибки времени выполнения называют не ошибками, а исключениями. Чаще всего исключения возникают по причине возможных ошибок пользователя. Если не предусмотрен перехват исключений и исключение возникает после запуска выполняемого EXE-файла (не из среды VB.NET), то появляется соответствующее сообщение и продолжение выполнения приложения невозможно, что является катастрофическим завершением работы приложения. Если же обработка исключения программистом предусмотрена, то при возникновении исключения пользователю предоставляется возможность исправить свою ошибку и продолжить работу с приложением. Перехват исключений следует предусмотреть на этапе разработки приложения.

Для обработки исключения используется блок кода Try ... Catch ... End Try. Оператор Try следует поместить непосредственно перед той подозрительной инструкцией, выполнение которой может привести к генерации исключения. А после этой инструкции помещается оператор Catch вместе с теми инструкциями, которые нужно выполнить, если будет сгенерировано исключение.

Синтаксис обработчика исключений выглядит так:

Try

Инструкции, которые могут вызвать исключение

Catch

Инструкции, которые выполняются, если исключение возникло

[Finally

Дополнительные инструкции, выполняемые при возникновении исключения независимо от его типа]

End Try

Обработка исключения должна выполняться сразу после ее обнаружения. Сначала следует установить тип исключения. Для этого в VB существует объект Err, свойство которого Err.Number содержит номер последнего возникшего исключения. Свойство Err.Description содержит краткое сообщение системы об этом исключении. Список номеров исключений, которые можно перехватывать и обрабатывать, Вы найдете в системе справок Visual Studio.

После определения типа исключения по ее номеру (свойство Number объекта Err) следует попытаться обработать исключение так, чтобы оно не мешало выполнению программы. Например, при ошибках обращения к диску может отображаться диалоговое окно для принятия пользователем решения о прекращении операции или повторном выполнении.

Свойство Description объекта Err возвращает текст системного описания исключения. Это сообщение можно использовать для вывода текста сообщения о возникшем исключении в обработчике ошибок, поскольку системный вывод сообщения об исключении при использовании обработчика ошибок не производится. Между

инструкциями Try и End Try может находиться несколько блоков обработки исключений, каждый из которых предназначен для обработки определенного типа исключения, например:

```
Try
    PictureBox.Image = _
    System.Drawing.Bitmap.FromFile("c:\Image\crach.bmp")
Catch When Err.Number = 53 'Файл не найден
    MsgBox("Такого файла нет")
Catch When Err.Number = 7 'Недостаточно памяти
    MsgBox("Слишком большой файл")
Catch
    MsgBox("Не удалось загрузить файл. " & _
    & Err.Description)
End Try
```

Здесь Catch When реагирует на появление конкретной ошибки времени выполнения. Если при загрузке рисунка в объект PictureBox.Image будет неправильно указано имя файла рисунка, то это приведет к сообщению "Такого файла нет". Если же при загрузке рисунка в объект PictureBox.Image размер файла окажется больше свободной памяти, то это приведет к сообщению "Слишком большой файл". Последняя инструкция Catch обрабатывает все остальные исключения. При любой другой ошибке, которая может возникнуть во время открытия файла будет выведено сообщение "Не удалось загрузить файл", сцепленное с системным сообщением об ошибке, содержащимся в Err.Description.

Блок кода, расположенный между строками **Finally** и **End Try** выполняется, если исключение возникло независимо от типа исключения. Это бывает необходимо для закрытия файлов при возникновении исключения с целью освобождения ресурсов.

4. Проект «Отладка»

1. Создайте новый проект с именем Отладка, следуя приложению 1.
2. Поместите на форме кнопку Button1 и текстовое поле TextBox1.
3. Установите свойства объектов, исходя из соответствия рис. 2:

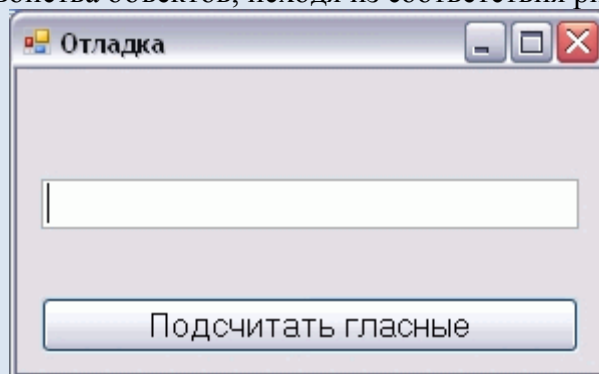


Рис. 2. Интерфейс проекта

4. Щелкните дважды на кнопке Подсчитать Гласные. Введите код, содержащийся в листинге 1.

Листинг 1. Код проекта

```
Private Sub Button1_Click(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles Button1.Click
    Button1.Text = ЧислоГласных _
    (TextBox1.Text)
End Sub
```

```

Function ЧислоГласных(ByVal s As String) As Integer
    Dim i As Integer = 1
    Dim k As Integer = 0
    Do While i <= s.Length
        Select Case s.Substring(i, 1).ToLower
            Case "а", "я", "о", "ё", "ы", "и", _
                "э", "е", "у", "ю"
                k += 1
        End Select
    Loop
End Function

```

Подпрограмма Button1_Click обращается к функции ЧислоГласных и отображает возвращаемое этой функцией значение на кнопке Button1.

Функция ЧислоГласных поочередно проверяет каждый символ строки *s*, не является ли он гласной буквой и подсчитывает число обнаруженных гласных. В строке 3 тела функции ЧислоГласных фрагмент *s.Length* – это возвращенное свойством *Length* число символов строки *s*. В строке 4 тела функции ЧислоГласных фрагмент *s.Substring(i, 1).ToLower* означает следующее:

1. Метод *Substring(i, 1)* возвращает подстроку строки *s*, начинающуюся с позиции *i* и включающую 1 символ.

2. Метод *ToLower* возвращает копию подстроки *s.Substring(i, 1)*, после преобразования всех ее символов к нижнему регистру.

Цикл, занимающий с 3 по 9 строки тела функции ЧислоГласных, обеспечивает поочередное рассмотрение всех символов строки *s*. Инструкция выбора, включающая строки 4 – 7 тела функции ЧислоГласных, обеспечивают проверку очередного рассматриваемого символа на предмет его принадлежности к гласным буквам. Инструкция *k += 1* эквивалентна по результату действия инструкции *k = k + 1*. Это принятая сокращенная форма записи таких инструкций, не требующая повторения имени изменяемой переменной.

5. Сохраните и запустите приложение. Введите в текстовом поле текст Ока и щелкните на кнопке Подсчитать гласные. После некоторого ожидания текст на кнопке не изменится, хотя это должно было произойти. Программа продолжает работать и не останавливается. Это признак того, что произошло заикливание. Остановите программу нажатием клавиш *Ctrl+Break* или командой *Debug, Stop Debugging*.

6. Чтобы выяснить причину неправильной работы программы задайте точку останова на строке, в которой начинается цикл. Для этого щелкните слева от этой строки в серой вертикальной полосе. Строка будет выделена красным цветом (рис. 3).

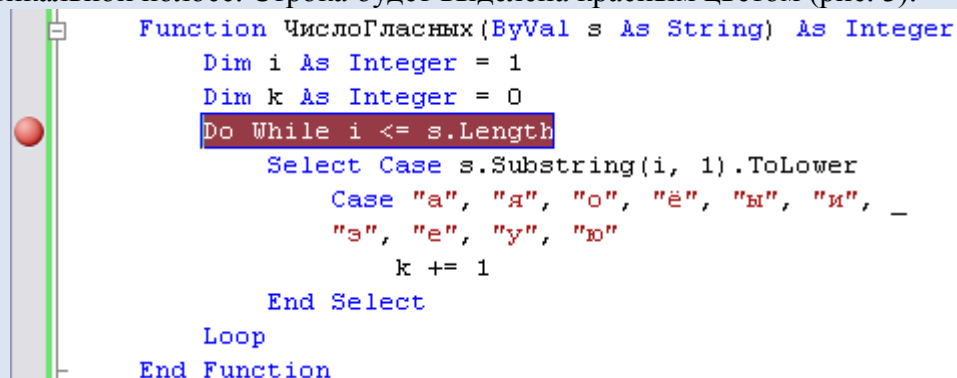



Рис. 3. Задание точки останова

7. Запустите программу. Снова введите в текстовом поле текст Ока и нажмите на кнопку Подсчитать гласные. Программа начала выполнять код и остановилась перед инструкцией, в которой установлена точка останова. Инструкция, которая будет

выполнена следующей, выделена желтым цветом. Подвигайте курсор мыши над программным кодом. Если задержать курсор над переменной выполняемой процедуры или над выделенным выражением, то появиться окно подсказки, в котором будет отображено текущее значение этой переменной или выражения. Дальнейшие строки программы Вы выполняйте по шагам, выполняя за один шаг одну инструкцию и контролируя после каждого шага изменения переменных. Для выполнения шага следует нажать кнопку  Step into на стандартной панели инструментов или клавишу F8. Обратите внимание на переменную i. По назначению эта переменная – номер обрабатываемого символа. Каждый раз после завершения цикла ее значение должно увеличиваться на 1. Но этого не происходит. Пропущен код увеличения этой переменной.

8. Устраните эту ошибку. Для этого остановите работу приложения и добавьте строку кода, выделенную ниже полужирным шрифтом:

```
Do While i <= s.Length
    Select Case s.Substring(i, 1).ToLower
        Case "a", "я", "о", "ё", "ы", "и", _
            "э", "е", "у", "ю"
            k += 1
    End Select
    i += 1
Loop
```

9. Ошибка исправлена. Снимите точку останова и запустите программу. Введите в текстовом поле текст Ока и нажмите на кнопку Подсчитать гласные. Работа программы завершится сообщением о возникшем исключении, показанным на рис. 4.

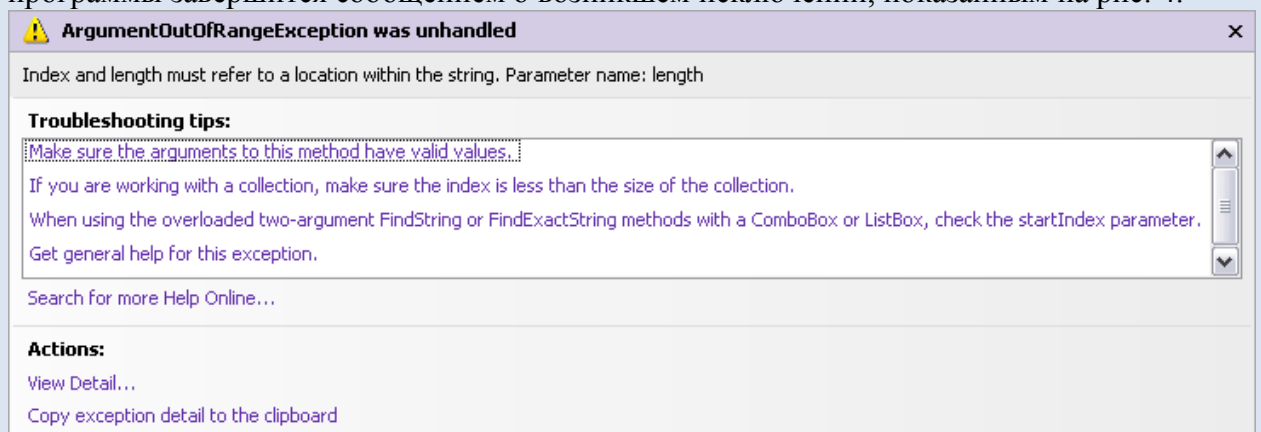


Рис. 4. Сообщение об аварийном завершении работы программы

10. Сообщение говорит о попытке извлечь из строки символ, номер которого превышает длину строки. Закройте окно этого сообщения. Среда находится в режиме Debugging (отладки). Об этом говорит слово, которое в прямоугольных скобках отображено в заголовке проекта справа от его имени. Желтым цветом выделена строка кода, выполнение которой привело к исключению. Вы можете убедиться, что ошибка произошла при извлечении символа № 3 строки ($i = 3$), а также в том, что длина строки ($s.Length$) равна 3. В чем же дело? А дело в том, что нумерация символов начинается не с 1, а с 0. Это означает, что при длине строки, равной 3, последний ее символ находится в позиции № 2.

11. Устраните эту ошибку. Для этого остановите работу приложения и исправьте две строки кода функции:

Ошибочная инструкция

```
Dim i As Integer = 1
Do While i <= s.Length
```

Исправленная инструкция

```
Dim i As Integer = 0
Do While i <= s.Length - 1
```


12. И эта ошибка исправлена. Снова запустите программу. Введите в текстовом поле текст Ока и нажмите на кнопку Подсчитать гласные. Программа работает неправильно. На кнопке отобразился 0, а в строке Ока две гласные.

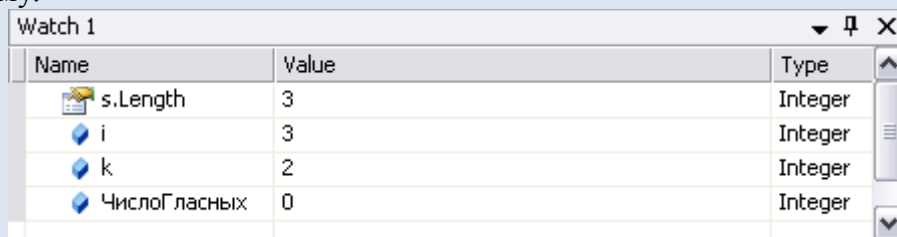
13. Чтобы выяснить причину неправильной работы программы задайте точку останова на строке, в которой начинается цикл, и запустите приложение. Введите в текстовом поле текст Ока и нажмите на кнопку Подсчитать гласные.

14. После перехода среды в режим отладки откройте окно контрольных значений Watch. Для этого следует выполнить команду меню Debug, Window, Watch, Watch1.

В нижней части экрана отобразится окно Watch1. В это окно можно поместить для контроля те переменные и выражения, изменение значений которых по ходу выполнения программы нас будут интересовать. Перетащите в это окно те данные, за изменениями которых Вы хотите наблюдать. Для этого в окне программного кода данное следует выделить, затем навести на него курсор мыши, нажать левую кнопку мыши и, не отпуская ее, отбуксировать в окно Watch1. Прodelайте это для данных: s.Length, i, k и ЧислоГласных.

15. Выполните по шагам код и следите за изменениями данных. Обратите внимание, переменная k при выполнении тела функции изменяется правильно. Остановите пошаговое выполнение кода, когда желтым цветом будет выделена последняя строка функции End Function. Содержание окна при этом должно соответствовать рис. 5. При следующем шаге тело функции будет покинуто, а ее имя до сих пор имеет значение, отличающееся от значения переменной k. Почему?

Все дело в том, что в функции пропущена инструкция присвоения значения результата вычислений k имени функции для возврата этого значения в вызывающую подпрограмму.



Name	Value	Type
s.Length	3	Integer
i	3	Integer
k	2	Integer
ЧислоГласных	0	Integer

Рис. 5. Вид окна контрольных значений Watch

16. Устраните эту ошибку. Для этого остановите работу приложения и ниже инструкции Loop добавьте строку кода Return (k).

17. Снимите точку останова и снова запустите программу. Введите в текстовом поле текст Ока и нажмите на кнопку Подсчитать гласные. Теперь программа работает правильно. На кнопке отобразилась цифра 2 (рис. 6). В строке Ока действительно две гласные буквы.



Рис. 6. Пример работы программы

18. Протестируйте программу на других строках, чтобы убедиться в ее правильной работе. Закончив тестирование, остановите приложение.

19. Попробуйте ответить на вопросы для контроля.
20. Покажите преподавателю результаты своей работы.
21. Удалите на диске *d* свою рабочую папку.

5. Вопросы для контроля

1. Каковы типы ошибок программы и методы их устранения?
2. В каких состояниях может находиться среда VB.NET?
3. Что является сигналом наличия в программном коде синтаксической ошибки?
4. Как можно убедиться, что программный код не содержит логических ошибок?
5. Какие средства предусмотрены для локализации логической ошибки?
6. Как можно установить и снять точку останова?
7. Чем различаются шаг с заходом и шаг с обходом?
8. Как в режиме Debugging среды IDE можно наблюдать за изменениями значений данных?
9. О чем можно узнать с помощью метода Length строки?
10. Что возвращает метод строки Substring(*x*, *k*)?
11. Что возвращает метод ToLower строки?
12. Каково назначение операции +=?
13. Какой номер имеет первый символ строки?
14. В каком состоянии среды может быть открыто одно из окон наблюдений?