**Практическая работа**

**Знакомство с отладчиком Visual С#**

**Теоретические сведения.**

Независимо от обстоятельств код, создаваемый разработчиками программного обеспечения, далеко не всегда работает так, как задумано. В некоторых случаях все идет совершенно не по плану! В подобных ситуациях необходимо выяснить, почему так происходит. При этом вместо многочасового изучения кода в поисках ошибок гораздо проще и эффективнее будет использовать средство отладки (отладчик).

К сожалению, отладчик не является той волшебной палочкой, по мановению которой будут выявлены абсолютно все проблемы в коде. Процесс *отладки* подразумевает пошаговое выполнение кода в средстве отладки (например, в Visual Studio) в поисках точки, в которой вы допустили ошибку при написании программы. Таким образом, вы получите возможность проанализировать код и внести в него необходимые исправления. При этом средства отладки часто допускают внесение временных изменений, благодаря чему вы можете продолжать выполнение программы.

Эффективное использование отладчика также требует определенных навыков, которые вырабатываются только с практикой, однако умение работать с ним является основополагающим требованием к любому разработчику программного обеспечения. В этой работе мы познакомим вас с основными принципами отладки и поделимся рекомендациями о том, с чего можно начать.

**Проанализируйте проблему, задавая себе правильные вопросы**

Это поможет вам выяснить, в чем состоит проблема, прежде чем приступать к ее решению. Мы полагаем, что вы уже сталкивались с проблемами в коде, иначе вряд ли читали бы сейчас эту статью в поисках советов по его отладке! Итак, прежде чем начать отладку, проанализируйте проблему, которую вы пытаетесь решить:

* Что именно должен был выполнить код?
* Что произошло вместо этого?

Если во время выполнения приложения вы столкнулись с ошибкой или исключением, это может быть хорошим признаком! Исключение возникает в том случае, если при выполнении кода происходит непредвиденное событие (как правило, это ошибка какого-либо рода). С помощью средства отладки вы можете перейти точно к тому месту в коде, где возникло исключение, и исследовать возможные способы исправления ситуации.

Если произошло что-то еще, каковы признаки проблемы? Есть ли у вас предположения относительно того, в каком месте кода возникла проблема? Например, если код должен выводить какой-то текст, но при этом текст содержит ошибки, вы можете сделать вывод, что в этом случае используются неверные данные или код вывода текста содержит ошибки другого рода. При пошаговом выполнении кода в отладчике вы можете изучить каждое изменение переменных и точно определить, когда и каким образом были присвоены неверные значения.

**Проверьте свои предположения**

Прежде чем исследовать причины возникновения ошибки, проверьте предположения, на основании которых вы ожидаете тот или иной результат. Скрытые или неизвестные предположения могут помешать выявлению проблемы, даже если вы изучаете непосредственно место ее возникновения в отладчике. Перечень возможных предположений может быть достаточно большим. Чтобы проверить их, задайте себе несколько вопросов.

* Используете ли вы нужный API (то есть соответствующие объект, функцию, метод или свойство)?Возможно, используемый вами API работает не так, как вы ожидаете. (После проверки вызова API в отладчике для исправления проблемы и выявления нужного API вам может потребоваться обратиться к документации.)
* Правильно ли вы используете API? Даже если вы выбрали нужный API, он может использоваться неправильно.
* Нет ли в вашем коде опечаток? Некоторые опечатки, например ошибки в написании имени переменной, могут быть незаметными, особенно при работе с языками, в которых не требуется объявление переменных перед их использованием.
* Вносили ли вы изменения в код и могут ли они быть связаны с возникшей проблемой?
* Должны ли объект или переменная содержать определенное значение (или определенный тип значения) и соответствует ли это действительности?
* Известно ли назначение кода? Как правило, отладка кода, написанного другим разработчиком, дается гораздо сложнее. Если это не ваш код, возможно, для его эффективной отладки вам потребуется изучить, что он делает.

**Совет**

При написании кода старайтесь начинать с небольших и гарантированно работающих фрагментов! (В этом вам может помочь хороший пример кода.) В некоторых случаях для исправления большого сложного фрагмента кода можно начать с маленького сегмента, демонстрирующего основную задачу, которую требуется выполнить. Затем вы можете последовательно изменять или добавлять код в поисках точки возникновения ошибки.

Анализируя собственные предположения, вы можете сократить время, необходимое на поиск проблем в коде. Кроме того, вы можете ускорить решение самих проблем.

**Используйте режим пошагового выполнения во время отладки для поиска места возникновения проблемы.**

Очевидно, что ошибки и неверные результаты можно увидеть только после выполнения кода приложения. Кроме того, работа программы может завершиться неожиданно без каких-либо сообщений.

При выполнении приложения в отладчике (в так называемом *режиме отладки*) осуществляется активный мониторинг всего, что происходит во время работы программы. Кроме того, вы можете в любой точке приостановить работу приложения, исследовать его состояние и при необходимости перейти в режим пошагового выполнения, чтобы изучить необходимые строки кода более детально.

Чтобы перейти в режим отладки в Visual Studio, необходимо нажать клавишу **F5** (также вы можете выбрать пункт меню **Отладка** > **Начать отладку** или нажать кнопку **Начать отладку** Начать отладку в панели инструментов "Отладка"). Если возникает исключение, помощник по исправлению ошибок Visual Studio направит вас к точке его появления и предоставит другую необходимую информацию.

Если исключение не возникает, возможно, вам следует проанализировать определенные места в коде, которые могут являться источником проблемы. На этом этапе следует использовать *точки останова* в отладчике, благодаря которым вы сможете исследовать код более внимательно. Точки останова — это один из самых простых и важных компонентов надежной отладки. Точка останова указывает, где Visual Studio следует приостановить выполнение кода, чтобы вы могли проверить значения переменных, поведение памяти или последовательность выполнения кода.

Чтобы задать точку останова в Visual Studio, достаточно щелкнуть в левом поле рядом с интересующей вас строкой кода. Также для этого можно поместить указатель мыши в нужную строку и нажать клавишу **F9**.

Чтобы продемонстрировать это, мы рассмотрим пример кода, который уже содержит несколько ошибок. В этом случае мы используем C#, однако функции отладки также работают для Visual Basic, C++, JavaScript, Python и других поддерживаемых языков.

**Создание образца приложения с ошибками**

Сейчас мы создадим приложение, которое содержит некоторые ошибки.

1. Для этого вам потребуется установленная среда Visual Studio с рабочими нагрузками **Разработка классических приложений .NET** или **Кроссплатформенная разработка .NET Core** (в зависимости от того, приложение какого типа вы создаете).

Установите Visual Studio бесплатно со страницы  [скачиваемых материалов Visual Studio](https://visualstudio.microsoft.com/downloads/) , если вы еще не сделали этого.

Если у вас уже установлена среда Visual Studio без нужных рабочих нагрузок, щелкните **Сервис** > **Получить средства и компоненты**. Запускается Visual Studio Installer. Выберите рабочую нагрузку **Разработка классических приложений .NET** или **Кроссплатформенная разработка .NET Core**, а затем щелкните **Изменить**.

1. Запустите Visual Studio.

На начальном экране выберите **Создать проект**. Введите **консоль** в поле поиска и выберите **Консольное приложение (.NET Core)** или **Консольное приложение (.NET Framework)** . Нажмите кнопку **Далее**.Введите имя проекта, например **ConsoleApp-FirstApp**, и нажмите кнопку **Создать**.

Если шаблон проекта **Консольное приложение (.NET Framework)** или **Консольное приложение (.NET Core)** отсутствует, перейдите в меню **Сервис** > **Получить средства и компоненты**, после чего запустится VisualStudioInstaller. Выберите рабочую нагрузку **Разработка кроссплатформенных приложений .NET Core** или **Разработка классических приложений .NET Core**, а затем выберите **Изменить**.

Visual Studio создаст консольный проект и откроет его в обозревателе решений (правая область).

1. Откройте файл *Program.cs* и замените все его содержимое по умолчанию следующим кодом.

using System;

using System.Collections.Generic;

namespaceConsoleApp\_FirstApp

{

classProgram

{

staticvoidMain(string[] args)

{

Console.WriteLine("Welcome to Galaxy News!");

IterateThroughList();

Console.ReadKey();

}

privatestaticvoidIterateThroughList()

{

var theGalaxies = new List<Galaxy>

{

new Galaxy() { Name="Tadpole", MegaLightYears=400, GalaxyType=new GType('S')},

new Galaxy() { Name="Pinwheel", MegaLightYears=25, GalaxyType=new GType('S')},

new Galaxy() { Name="Cartwheel", MegaLightYears=500, GalaxyType=new GType('L')},

new Galaxy() { Name="Small Magellanic Cloud", MegaLightYears=.2, GalaxyType=new GType('I')},

new Galaxy() { Name="Andromeda", MegaLightYears=3, GalaxyType=new GType('S')},

new Galaxy() { Name="Maffei 1", MegaLightYears=11, GalaxyType=new GType('E')}

};

foreach (Galaxy theGalaxy in theGalaxies)

{

Console.WriteLine(theGalaxy.Name + " " + theGalaxy.MegaLightYears + ", " + theGalaxy.GalaxyType);

}

// Expected Output:

// Tadpole 400, Spiral

// Pinwheel 25, Spiral

// Cartwheel, 500, Lenticular

// Small Magellanic Cloud .2, Irregular

// Andromeda 3, Spiral

// Maffei 1, 11, Elliptical

}

}

publicclassGalaxy

{

publicstring Name { get; set; }

publicdouble MegaLightYears { get; set; }

publicobject GalaxyType { get; set; }

}

publicclassGType

{

publicGType(char type)

{

switch(type)

{

case'S':

MyGType = Type.Spiral;

break;

case'E':

MyGType = Type.Elliptical;

break;

case'l':

MyGType = Type.Irregular;

break;

case'L':

MyGType = Type.Lenticular;

break;

default:

break;

}

}

publicobject MyGType { get; set; }

privateenum Type { Spiral, Elliptical, Irregular, Lenticular}

}

}

Этот код выводит список, содержащий название галактики, расстояние до нее, а также тип галактики. При отладке важно учитывать предназначение кода. Ниже показан формат одной строки списка, который должен выводить код.

*название галактики*, *расстояние*, *тип галактики*.

**Запуск приложения**

1. Нажмите клавишу **F5** или кнопку **Начать отладку** Начать отладку на панели инструментов "Отладка", которая расположена над редактором кода.

По результатам запуска приложения отладчик не демонстрирует никаких исключений. Тем не менее данные, выводимые в окно консоли, не соответствуют ожиданиям. Вот, что мы должны были увидеть.

Tadpole 400, Spiral

Pinwheel 25, Spiral

Cartwheel, 500, Lenticular

SmallMagellanicCloud .2, Irregular

Andromeda 3, Spiral

Maffei1, Elliptical

Вместо этого выводятся следующие данные.

Tadpole 400, ConsoleApp\_FirstApp.GType

Pinwheel 25, ConsoleApp\_FirstApp.GType

Cartwheel, 500, ConsoleApp\_FirstApp.GType

Small Magellanic Cloud .2, ConsoleApp\_FirstApp.GType

Andromeda 3, ConsoleApp\_FirstApp.GType

Maffei 1, 11, ConsoleApp\_FirstApp.GType

Проанализировав выходные данные и код, мы можем сделать вывод, что в классе GType содержится тип галактики. Нам необходимо выводить на экран тип галактики (например, Spiral), а не название класса.

**Отладка приложения**

1. Во время выполнения приложения щелкните в левом поле рядом с вызовом метода Console.WriteLine, чтобы задать точку останова в этой строке кода.

foreach (Galaxy theGalaxy in theGalaxies)

{

Console.WriteLine(theGalaxy.Name + " " + theGalaxy.MegaLightYears + ", " + theGalaxy.GalaxyType);

}

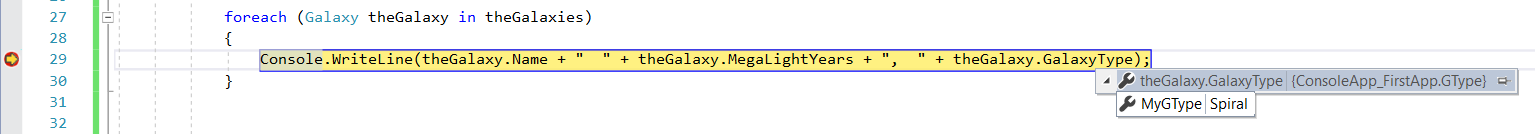
В месте установки точки останова в левом поле появится красный круг.

Поскольку возникшая проблема связана с выходными данными, необходимо начать отладку с проверки кода, который выполняется перед установкой выходных данных.

1. Нажмите кнопку **Перезапустить** Перезапустить приложение на панели инструментов отладки (**CTRL** + **SHIFT** + **F5**).

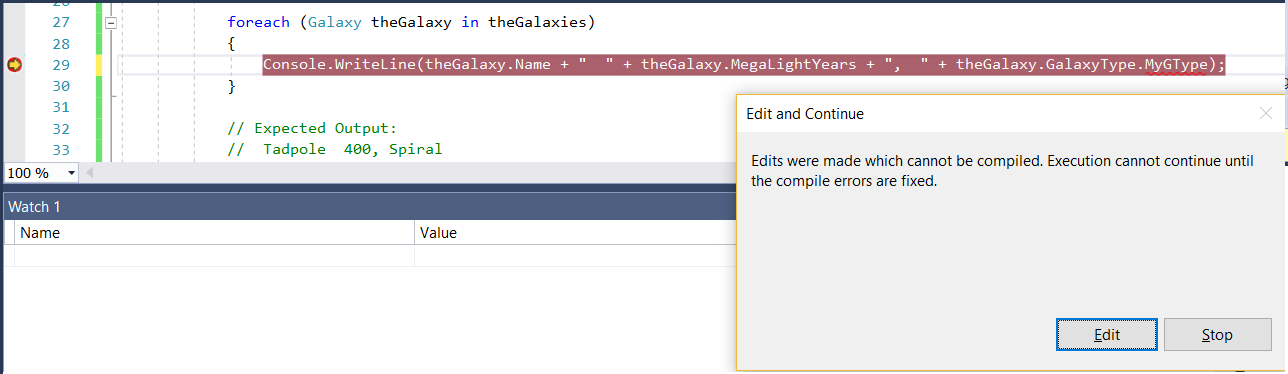
Выполнение приложения приостановится в заданной точке останова. Место приостановки отладчика будет выделено желтым цветом (при этом желтая строка кода на этот момент не выполнена).

1. Наведите указатель мыши на переменную GalaxyType справа. После этого разверните theGalaxy.GalaxyType слева от значка гаечного ключа. Как вы можете видеть, GalaxyType содержит свойство MyGType, которому присваивается значение Spiral.

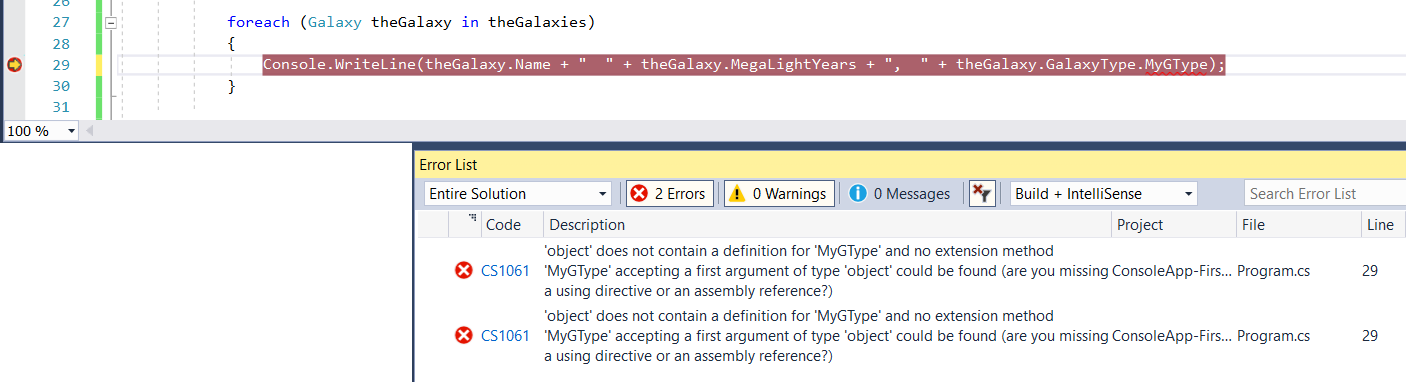


Изначально вы ожидаете, что в окно консоли будет выведено именно значение Spiral. Таким образом, вам следует проверить это значение в коде во время выполнения приложения. В этом сценарии мы используем неправильный API. Посмотрим, сможем ли мы исправить проблему во время выполнения кода в отладчике.

1. Во время отладки этого же кода установите указатель мыши в конец элемента theGalaxy.GalaxyType и измените его на theGalaxy.GalaxyType.MyGType. Несмотря на допустимость такого изменения, в редакторе кода отображается ошибка, свидетельствующая о невозможности выполнить компиляцию.



1. Щелкните **Изменить** в окне сообщения **Изменить и продолжить**. Сообщение об ошибке будет выведено в окне **Список ошибок**. Эта ошибка указывает, что 'object' не содержит определение для MyGType.



Несмотря на то что мы задаем каждую галактику с использованием объекта GType (который содержит свойство MyGType), отладчик не распознает theGalaxy как объект типа GType. Что происходит? В этом случае необходимо проверить код, который задает тип галактики. Вы увидите, что класс GType содержит свойство MyGType, однако все же что-то работает не так. Разгадка кроется в сообщении об ошибке для object. Интерпретатор языка воспринимает его как объект object вместо ожидаемого типа GType.

1. Анализируя код, в котором задается тип галактики, вы можете увидеть, что свойство GalaxyType класса Galaxy задается как object вместо GType.

Public object GalaxyType { get; set; }

1. Измените предшествующий код на следующий.

public GType GalaxyType { get; set; }

1. Нажмите кнопку **Перезапустить** Перезапустить приложение на панели инструментов отладки (**CTRL** + **SHIFT** + **F5**), чтобы выполнить повторную компиляцию кода и снова запустить его.

Когда отладчик приостановит выполнение на строке Console.WriteLine, вы можете навести указатель мыши на theGalaxy.GalaxyType.MyGType и убедиться, что значение задано правильно.

1. Удалите точку останова, щелкнув ее кружок в левом поле (также для этого можно выбрать команду **Точка останова** > **Удалить точку останова**). После этого нажмите клавишу **F5** для продолжения.

Приложение запускается и отображает выходные данные. Теперь все работает правильно, однако вы можете заметить еще одну неточность. Для галактики SmallMagellanicCloud, которая имеет тип Irregular, в консоли не выводится никакой тип.

Tadpole 400, Spiral

Pinwheel 25, Spiral

Cartwheel, 500, Lenticular

Small Magellanic Cloud .2,

Andromeda 3, Spiral

Maffei 1, Elliptical

1. Задайте точку останова на этой строке кода.

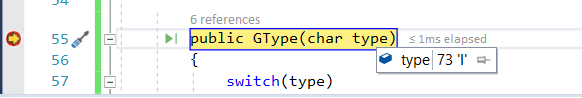
Public GType(char type)

Здесь задается тип галактики, поэтому нам необходимо изучить эту строку более пристально.

1. Нажмите кнопку **Перезапустить** Перезапустить приложение на панели инструментов отладки (**CTRL** + **SHIFT** + **F5**) для перезапуска.

Отладчик приостановит работу в строке кода, где вы задали точку останова.

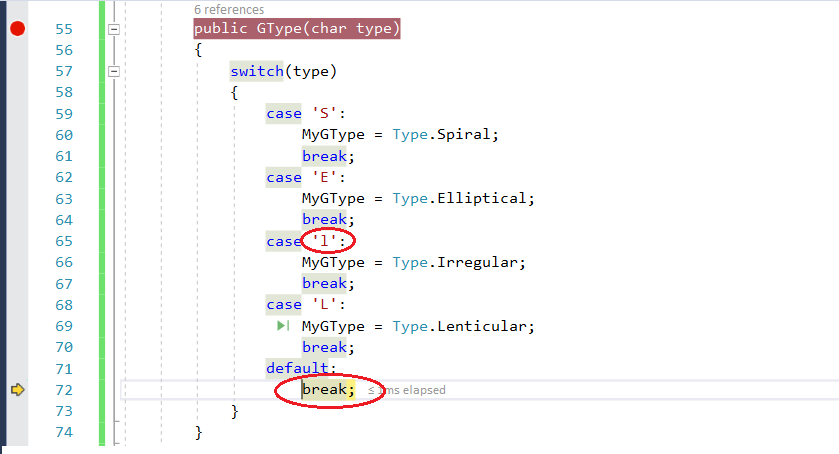
1. Наведите указатель мыши на переменную type. Отображается значение S (после кода символа). Вас интересует значение I, поскольку эта галактика должна иметь тип Irregular.
2. Нажмите клавишу **F5** и снова наведите указатель мыши на переменную type. Повторяйте этот шаг, пока в переменной type не появится значение I.



1. Нажмите клавишу **F11** (выберите **Отладка** > **Шаг с заходом** или нажмите кнопку **Шаг с заходом** в панели инструментов "Отладка").

При нажатии клавиши **F11** отладчик переходит на одну инструкцию вперед и выполняет соответствующий код. Клавиша **F10** (**Шаг с обходом**) имеет схожую функцию и также полезна при работе с отладчиком.

1. Нажимайте клавишу **F11** до тех пор, пока вы не остановитесь в строке кода с инструкцией switch для значения I. Здесь вы увидите очевидную ошибку, связанную с опечаткой. Вы ожидали, что код будет выполнен дальше до места, где для MyGType задается тип Irregular, однако вместо этого отладчик полностью пропускает этот код и приостанавливает работу в разделе default инструкции switch.



Взглянув на код, вы заметите опечатку в инструкции case 'l'. Вместо этого она должна иметь вид case 'I'.

1. Щелкните код case 'l' и замените его на case 'I'.
2. Удалите точку останова и нажмите кнопку **Перезапустить**, чтобы запустить приложение снова.

Теперь все ошибки исправлены и приложение выдает ожидаемые результаты.

Нажмите любую клавишу, чтобы завершить работу приложения.

**Замечания**

Если вы сталкиваетесь с проблемой, воспользуйтесь отладчиком и [командами пошагового выполнения](https://docs.microsoft.com/ru-ru/visualstudio/debugger/navigating-through-code-with-the-debugger?view=vs-2019), такими как **F10** и **F11**, для поиска области кода, в которой возникают ошибки.

Если выявить проблемную область кода не удается, задайте точку останова в коде, который выполняется перед возникновением ошибки, после чего используйте команды пошагового выполнения, пока проблема не проявится. Также вы можете использовать [**точки трассировки**](https://docs.microsoft.com/ru-ru/visualstudio/debugger/using-breakpoints?view=vs-2019#BKMK_Print_to_the_Output_window_with_tracepoints) для вывода сообщений в **окно вывода**. Анализируя сообщения, которые выводятся в окно или отсутствуют в нем, нередко можно изолировать область кода, где возникает проблема. Для дальнейшего сужения области поиска этот процесс может потребоваться повторить несколько раз.

Выявив проблемную область кода, используйте отладчик для ее детального анализа. Чтобы определить причину возникновения проблемы, проверьте код во время выполнения приложения в отладчике.

* [Проверьте переменные](https://docs.microsoft.com/ru-ru/visualstudio/debugger/view-data-values-in-data-tips-in-the-code-editor?view=vs-2019) и убедитесь, что они содержат значения того типа, который вы ожидаете. Если переменная содержит недопустимое значение, найдите, где оно было задано. Для этого может потребоваться перезапустить отладчик, проверить [стек вызовов](https://docs.microsoft.com/ru-ru/visualstudio/debugger/how-to-use-the-call-stack-window?view=vs-2019) или выполнить одновременно оба этих действия.
* Проверьте, выполняет ли ваше приложение код, который вы ожидаете. (Так, в примере приложения должна была выполняться инструкция switch, устанавливающая тип галактики Irregular, однако нужный код был пропущен из-за опечатки.)