



UNIDADE V

Linguagem de Programação C#



ALGORITMO – C# (C sharp)

Sumário

1	UNIDADE 5 – FERRAMENTA VISUAL STUDIO 2013	3
1.1	Visual studio	3
1.2	As principais novidades do Visual Studio 2013	3
1.2.1	Depuração de aplicações	3
1.2.3	Team Foundation (Controle de versão)	4
1.2.3	Testes.....	5
1.2.4	ASP.NET 4.5.1	6
1.2.5	Windows Store Apps.....	6
1.2.6	Windows Azure Mobile Services	6
1.2.7	SharePoint.....	7
1.2.8	LightSwitch	7
1.3	Conhecendo o ide.....	8
1.4	CRIAR UM APLICATIVO SIMPLES.....	9
1.4.1	Criando um projeto	9
1.4.2	Gerenciando suas soluções e projetos	10
1.4.3	Ferramentas	13
1.4.4	Debug.....	15
1.4.4	Debug (Passo a passo)	16
1.4.4.1	Janelas Importantes.....	17
1.4.5	Erros.....	19
1.4.5.1	Janela de lista de erros	20
3	Conclusão	21

1 UNIDADE 5 – FERRAMENTA VISUAL STUDIO 2013

Nessa unidade veremos a ferramenta Microsoft Visual Studio 2013 como um todo e conhecer a sintaxe, comandos e operadores da linguagem C#.

1.1 VISUAL STUDIO

O Microsoft Visual Studio permite que os desenvolvedores criem com muita rapidez aplicativos, proporcionar experiências de usuário com a mais alta qualidade e riqueza. O Visual Studio 2013, junta ferramentas com as quais as organizações sentirão maior facilidade em capturar e analisar informações, o que significa a melhor tomada de decisões de negócios. O Visual Studio 2013 possibilita que organizações de todos os tamanhos criem aplicativos mais seguros, gerenciáveis e confiáveis. O Visual Studio 2013 se baseia em três pilares para proporcionar melhor experiência para os programadores: Melhorias na produtividade do desenvolvedor; Gerenciamento do ciclo de vida do aplicativo; e Utilização das mais recentes tecnologias.

1.2 AS PRINCIPAIS NOVIDADES DO VISUAL STUDIO 2013

Ainda no início do mês de Junho/2013, a Microsoft anunciaria durante o TechEd norte-americano planos para o lançamento da versão 2013 do Visual Studio, juntamente com a versão 4.5.1 do .NET Framework. Desde o final deste mesmo mês, o Visual Studio 2013 Preview encontra-se disponível para download e testes por meio do link: <http://www.microsoft.com/visualstudio/ptb/2013-downloads>

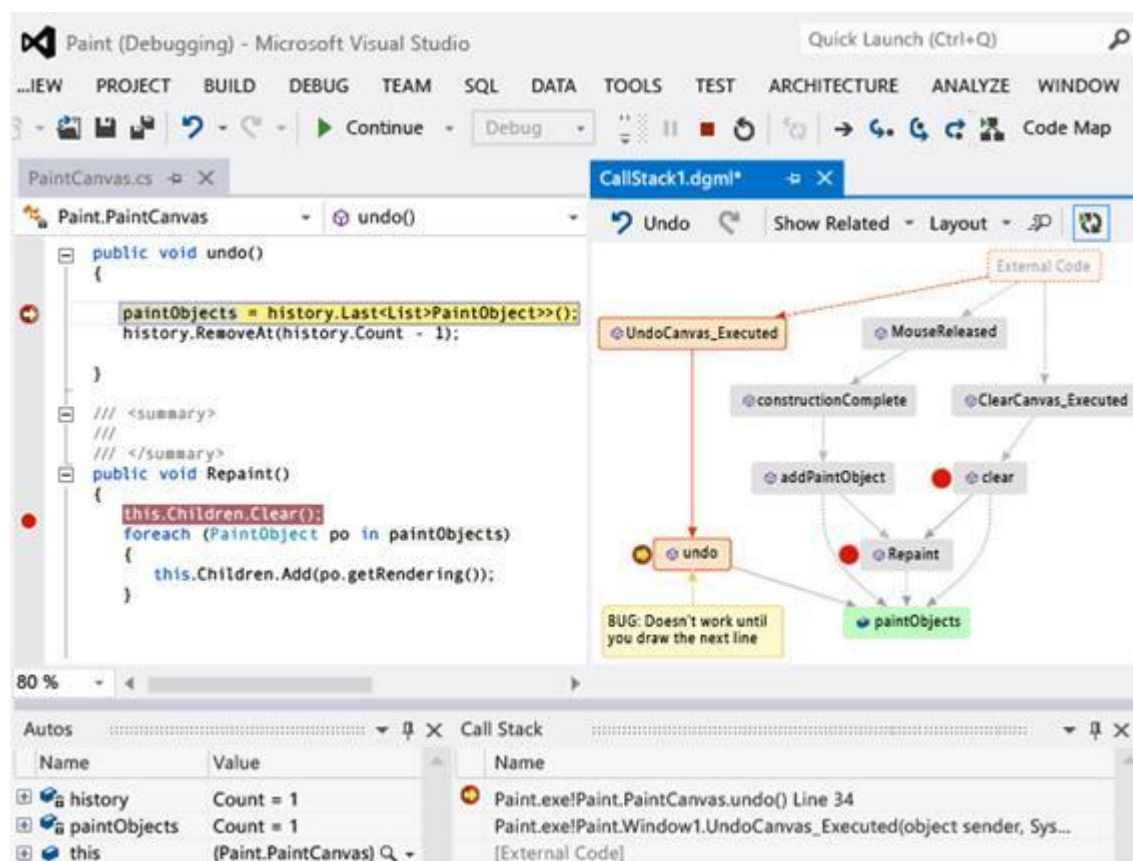
1.2.1 Depuração de aplicações

Diversas novidades também foram introduzidas no mecanismo de debug nesta nova versão do Visual Studio. Dentre as atualizações, é possível destacar:

A depuração de soluções do tipo Windows Store que fazem uso de técnicas de programação assíncrona foi melhorada;

Novas melhorias também foram incorporadas no que se refere ao debug de trechos de código em C++ e JavaScript;

A análise do conteúdo do call stack permite agora a geração de mapas analisando o código (code maps), possibilitando assim uma visualização mais detalhada do que está acontecendo durante a depuração mostrada na figura abaixo. Notas inclusive podem ser adicionadas aos itens destes mapas, constituindo-se em um importante auxílio durante a verificação de problemas que estão ocorrendo em uma aplicação.



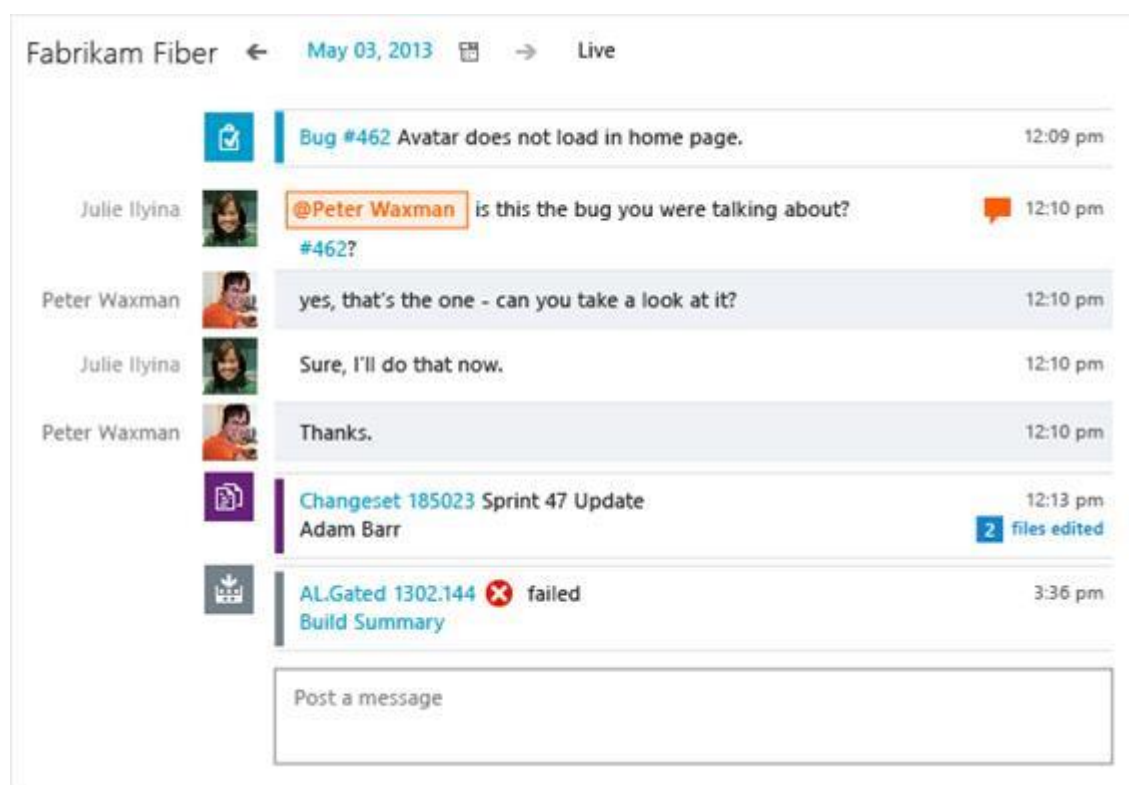
1.2.3 Team Foundation (Controle de versão)

O Team Foundation é uma solução colaborativa da Microsoft com forte ênfase no controle das atividades de equipes desenvolvimento. A fim de cumprir tais objetivos, essa ferramenta conta com funcionalidades para o controle de versão do código de aplicações, gerenciamento dos usuários que estarão envolvidos neste processo, além de relatórios e estatísticas que detalham o desempenho dos membros de um time no dia-a-dia.

A integração do Visual Studio com o Team Foundation também passou por diversas melhorias:

A possibilidade de acesso a repositórios na Web como o CodePlex e o GitHub;
O gerenciamento de atividades de backlog através de portfólios (recurso conhecido como “Portfolio Backlogs”), permitindo assim um melhor controle das tarefas de times de desenvolvedores dentro de uma organização;

Um dispositivo para conferências online chamado Team Room. A partir deste mecanismo, diferentes profissionais envolvidos em um projeto podem discutir aspectos relativos ao desenvolvimento, questionar uns aos outros a respeito de soluções técnicas ou, mesmo, reportar o status de suas atividades num determinado instante. O exemplo abaixo mostra um grupo de desenvolvedores conversando pela Team Room.



1.2.3 Testes

O Visual Studio 2013 Preview também contempla funcionalidades que viabilizam o teste automatizado interfaces gráficas construídas em SharePoint, voltadas ao Internet Explorer 11 ou, até mesmo, aplicações Windows Store baseadas no padrão XAML.

1.2.4 ASP.NET 4.5.1

O anúncio do Visual Studio 2013 Preview foi acompanhado também pelo lançamento da versão 4.5.1 do ASP.NET. Dentre os novos recursos para o desenvolvimento Web incorporados a este release, é possível destacar:

O ASP.NET MVC 5, com um novo template para a construção de sites;

O ASP.NET Web API 2, com novos melhoramentos no que se refere à construção de soluções baseadas no modelo REST;

O recurso ASP.NET Scaffolding para a implementação de aplicações Web Forms. Semelhante ao mesmo conjunto de funcionalidades já existentes no ASP.NET MVC, este mecanismo busca oferecer uma maior produtividade durante o desenvolvimento de sites em Web Forms. Graças ao ASP.NET Scaffolding, é possível a geração automatizada de interfaces gráficas (formulários .aspx) para a consulta, inclusão, exclusão ou atualização de registros de uma base de dados;

A possibilidade de integrar soluções baseadas em ASP.NET MVC e ASP.NET Web API a mecanismos de autenticação disponibilizados por provedores externos (como Facebook, Twitter, Microsoft ou Google).

1.2.5 Windows Store Apps

O Visual Studio 2013 também oferece suporte ao desenvolvimento de soluções voltadas ao Windows 8.1 Preview. Para a nova versão deste sistema operacional (ainda em avaliação) o .NET Framework conta com APIs que facilitam a implementação de aplicações que façam uso de recursos de programação assíncrona e do DirectX 11.2, além de novos controles baseados em HTML/Javascript e XAML.

1.2.6 Windows Azure Mobile Services

O Windows Azure Mobile Services é uma plataforma na nuvem que viabiliza a implementação de aplicações para o Windows Phone 8, Windows Store e o ambiente iOS. Assim como acontece com outras tecnologias já citadas aqui, o Visual Studio 2013 Preview também oferece suporte ao desenvolvimento de soluções baseadas nestes serviços.

1.2.7 SharePoint

Projetos que façam uso de SharePoint também podem se beneficiar de melhoramentos introduzidos com o Visual Studio 2013. O SharePoint é a solução da Microsoft destinada ao gerenciamento de documentos e outras formas de conteúdo. Isto acontece através da implementação de sites, os quais podem ser acessados dentro de uma rede corporativa ou, até mesmo, a partir da Internet.

Até antes desta nova versão do Visual Studio, o desenvolvimento de projetos em SharePoint estava obrigatoriamente atrelado ao uso de recursos de Web Forms para a implementação de soluções.

Com este novo release, existe agora a possibilidade de desenvolvimento de projetos SharePoint que façam uso do framework ASP.NET MVC, além da publicação de tais aplicações sob a forma de sites na plataforma Windows Azure.

1.2.8 LightSwitch

O Microsoft Visual Studio LightSwitch é um ambiente de desenvolvimento criado para simplificar e encurtar o tempo de desenvolvimento de aplicativos de negócios e de serviços de dados. O LightSwitch facilita a criação de aplicativos de negócios centrados em dados que podem consumir uma grande variedade de fontes de dados e criar clientes executáveis em inúmeros dispositivos. Com o LightSwitch, você pode:

- Criar aplicativos baseados em HTML5 que podem ser executados em todos os dispositivos modernos;

- Consumir e agregar várias fontes de dados, como bancos de dados, SharePoint e OData;

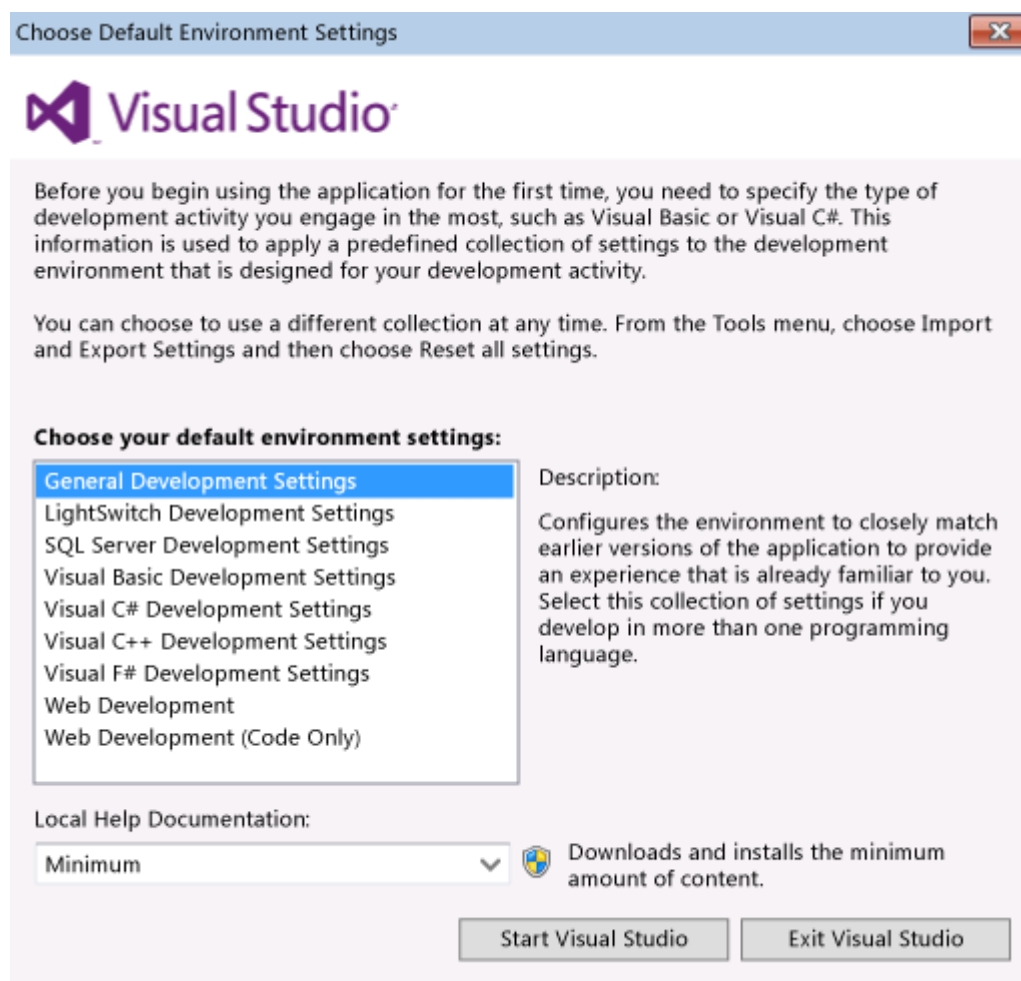
- Eliminar o código repetitivo e se concentrar naquilo que torna seu aplicativo único;

- Ter opções de implantação flexível e de hospedagem na nuvem que incluem o Azure e o Office 365;

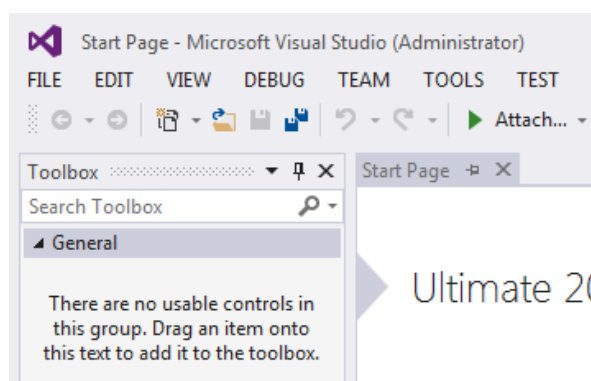
- Confiar que sua solução tem como base tecnologia de ponta e as práticas recomendadas do mercado;

1.3 CONHECENDO O IDE

Quando você inicia o Visual Studio pela primeira vez, deve escolher uma combinação de configurações que aplica um conjunto de personalizações predefinidas ao IDE. Cada combinação de configurações foi criada para facilitar o desenvolvimento de aplicativos para você. No nosso caso, selecionamos Visual C#:



A imagem abaixo mostra a página inicial do Visual Studio 2013. Com seu menu superior.



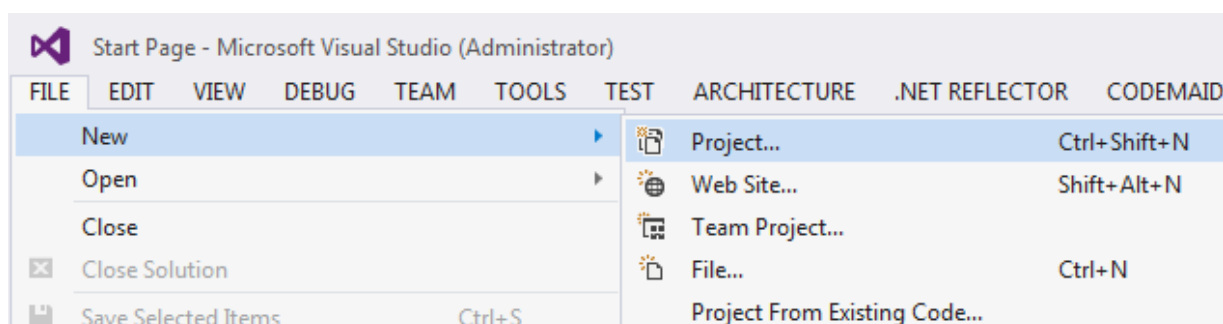
1.4 CRIAR UM APLICATIVO SIMPLES

Esse capítulo mostrará como criar o seu primeiro aplicativo utilizando o Visual Studio 2013.

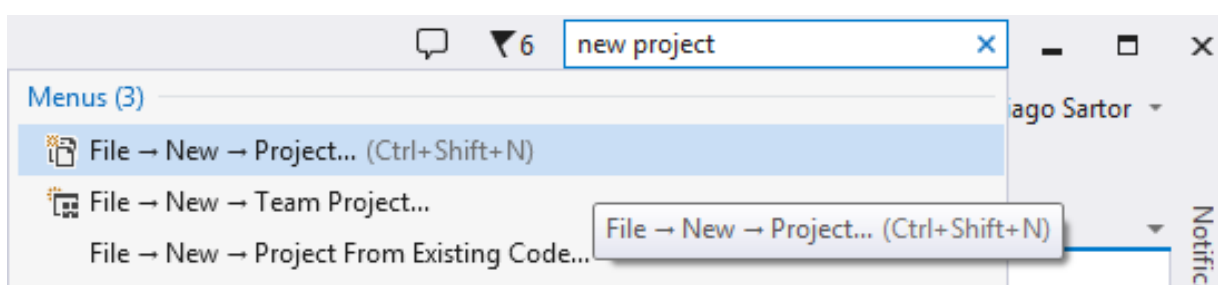
1.4.1 Criando um projeto

Quando você cria um aplicativo no Visual Studio, primeiro cria um projeto e uma solução. Para este exemplo, você criará uma solução do ExerciciosAula.

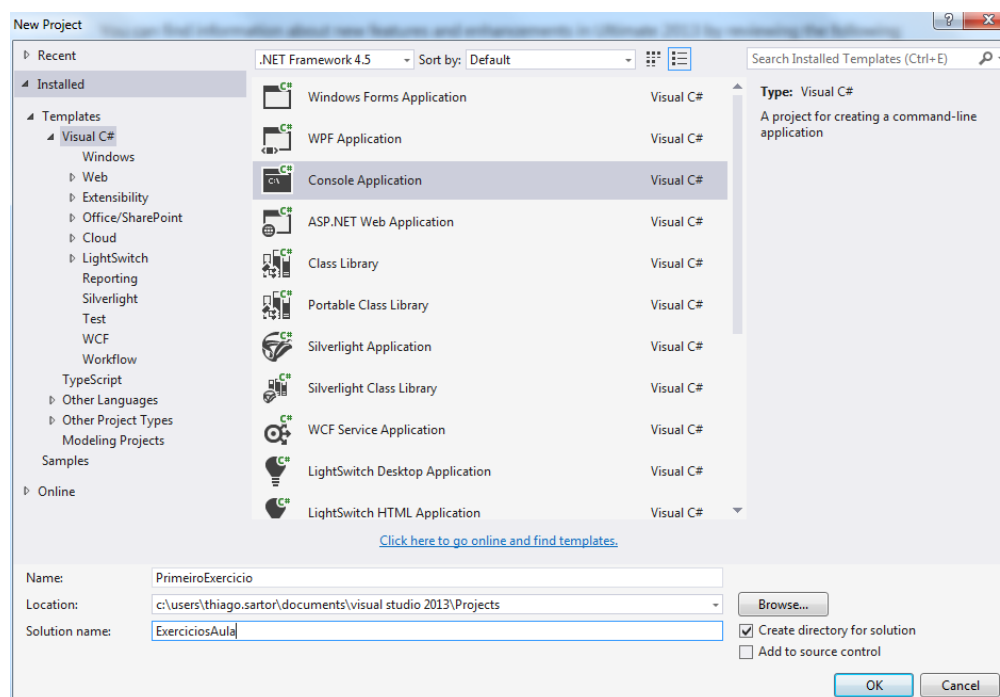
Para criar um novo projeto vá até a barra de menus selecione: **File, New, Project.**



Você também pode digitar New Project na caixa **Quick Launch**, traduzindo para o português inicialização rápida, e fazer a mesma coisa. Essa opção normalmente fica no canto superior direito. Como a imagem abaixo mostra:

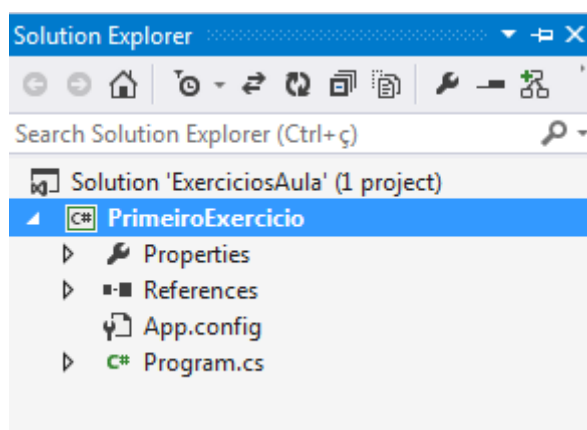


Escolha Visual C#, Console Application e então nomeie o projeto como 'Primeiro Exercício', veja que o nome da solução deve ser mudado para ExerciciosAula, para evitar tipos de conflitos futuramente:



1.4.2 Gerenciando suas soluções e projetos

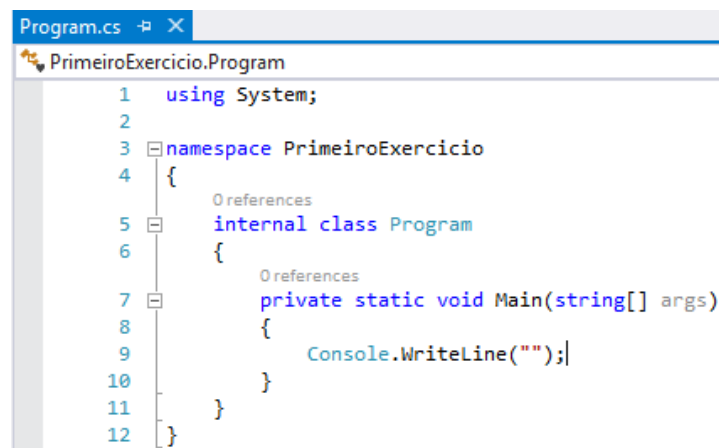
O projeto e a solução foram criados e os arquivos diferentes aparecem no **Solution Explorer**. Caso não esteja aparecendo vá até o menu superior e selecione: **View**, **Solution Explorer**. Ou pelo **Quick Launch** e digite *Solution Explorer*. O projeto Console Application tem uma classe principal, que se chama Program.cs.



Essa classe possui um método principal, que é responsável pela execução do teu projeto. Por que esse método é estático? E qualquer variável que eu declare para ele tem que ser estática?

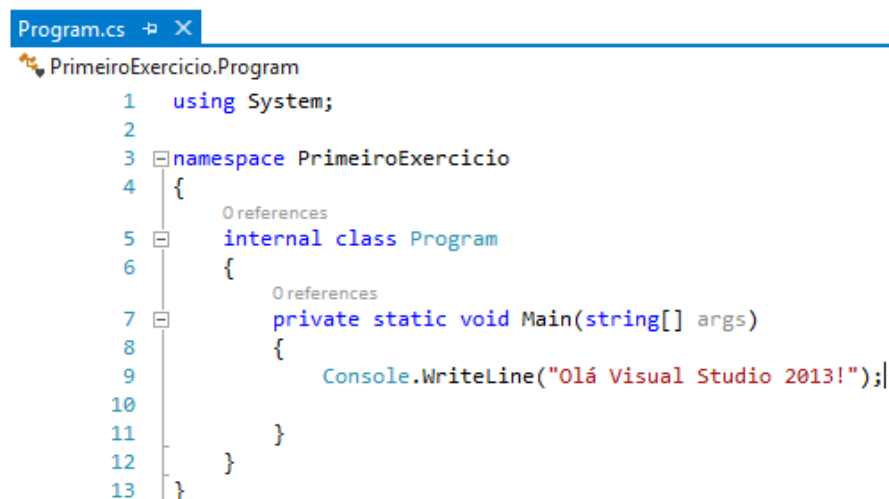
O método Main é estático porque ele é pré-definido pelo compilador C# como o ponto de entrada no código para o assembly. Não há nenhuma instância de nenhum objeto num primeiro momento, apenas o modelo da classe é carregado na memória e seus membros estáticos, incluindo o método de ponto de entrada.

Um método estático só pode chamar outros métodos estáticos (a menos que haja um identificador de instância de algo criado para uso). É por isso que o método Main chama outros métodos estáticos e porque você recebe um erro de compilação se você tentar chamar um método não-estático (de uma instancia).



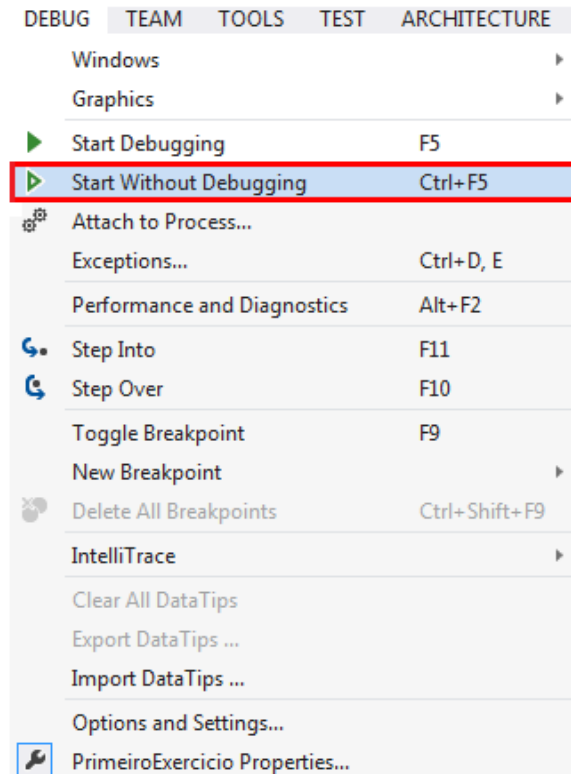
```
Program.cs
PrimeiroExercicio.Program
1  using System;
2
3  namespace PrimeiroExercicio
4  {
5      internal class Program
6      {
7          private static void Main(string[] args)
8          {
9              Console.WriteLine("");
10         }
11     }
12 }
```

Com o comando de saída de dados "Console.WireteLine("");" faremos nosso primeiro programa em C# no Visual Studio 2013. Digite o código como a imagem abaixo:

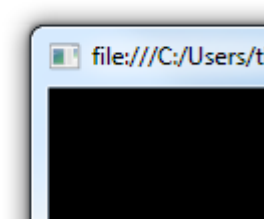
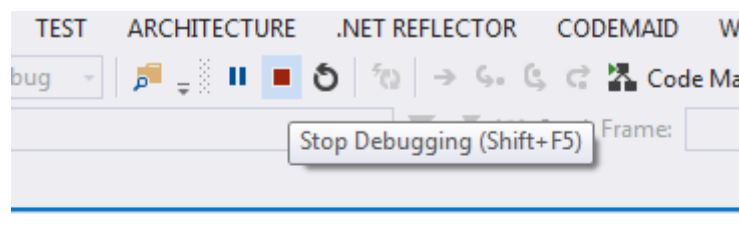


```
Program.cs
PrimeiroExercicio.Program
1  using System;
2
3  namespace PrimeiroExercicio
4  {
5      internal class Program
6      {
7          private static void Main(string[] args)
8          {
9              Console.WriteLine("Olá Visual Studio 2013!");
10         }
11     }
12 }
13 }
```

Se você pressionar a tecla F5 (Executar), o programa irá abrir o console e fechar rapidamente. Para isso temos a opção CTRL+F5 (Imagem abaixo), ou se preferirem utilizar o comando "Console.ReadKey();" no final do método principal.



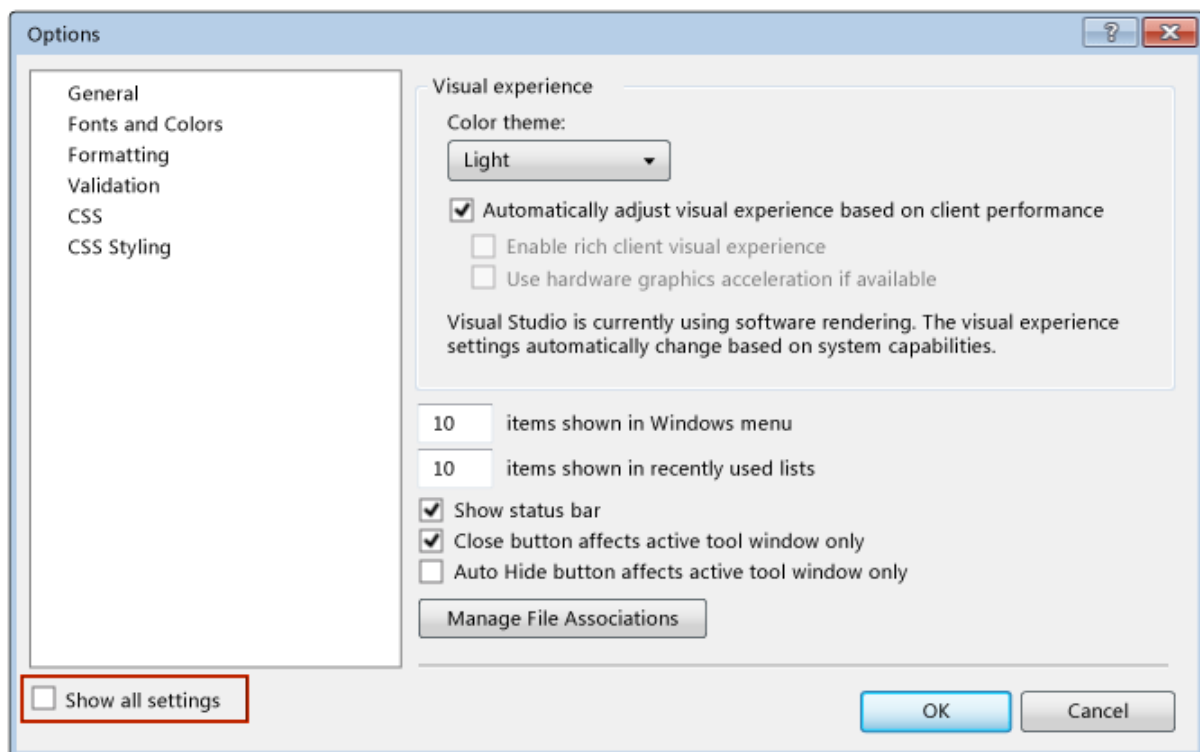
Como paro minha aplicação? Quando seu aplicativo estiver rodando aparecerá um botão de pausa e stop. O de Stop, interrompe na hora. O seu atalho é Shift + F5.



1.4.3 Ferramentas

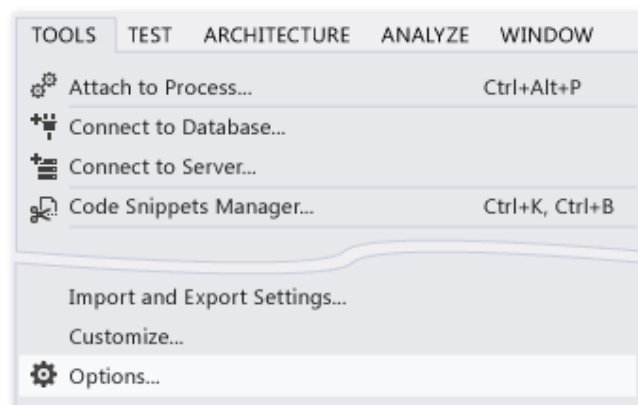
Você pode fazer personalizações adicionais no Visual Studio, como alterar a fonte e o tamanho do texto no editor ou o tema de cores do IDE, usando a caixa de diálogo **Opções**. Dependendo da combinação de configurações que você tiver aplicado, alguns itens na caixa de diálogo poderão não aparecer automaticamente. Você pode garantir que todas as opções possíveis apareçam escolhendo a caixa de seleção **Show all settings** que mostrar todas as configurações.

Logo abaixo a figura exibe a caixa de diálogo **Opções**:

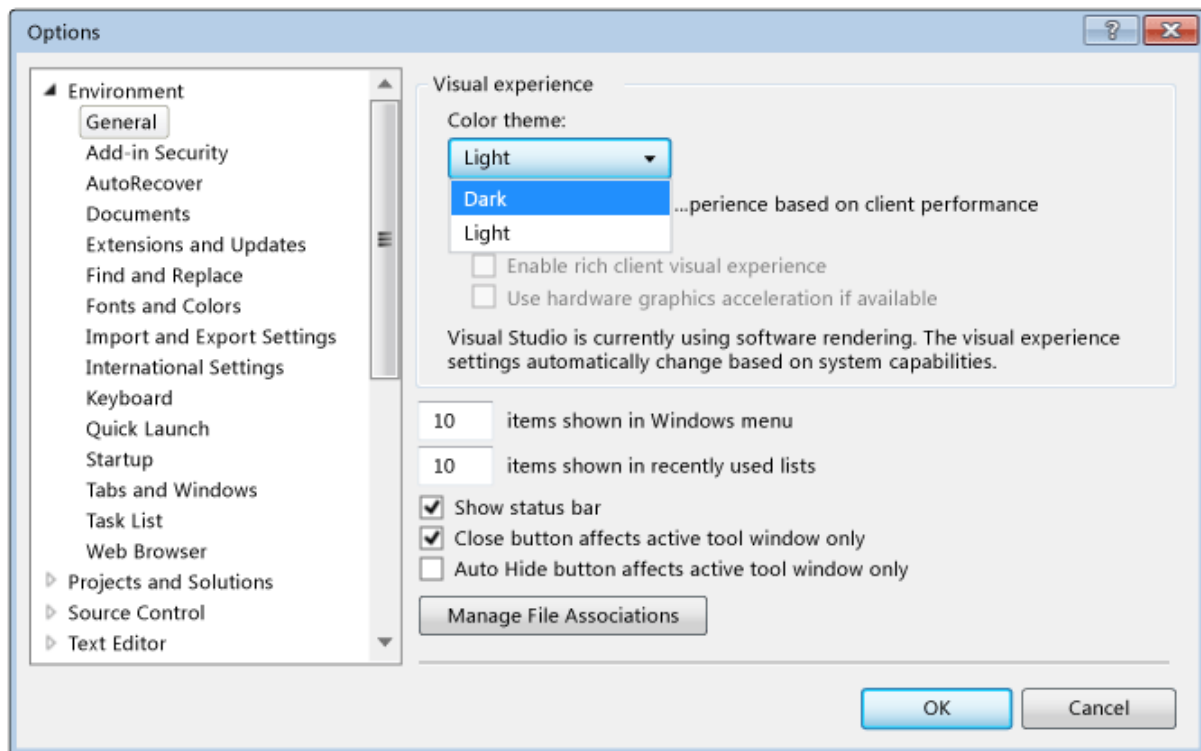


No exemplo abaixo, você irá alterar o tema de cor da IDE entre claro e escuro. Para alterar o tema de cores do IDE:

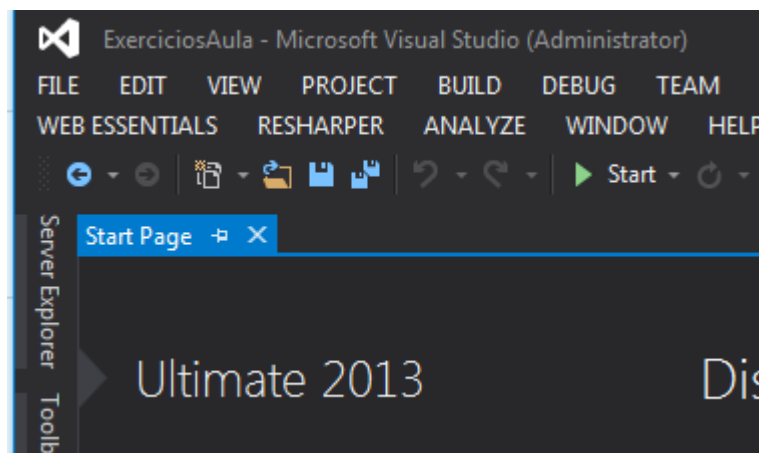
1. Abra a caixa de diálogo **Opções**:



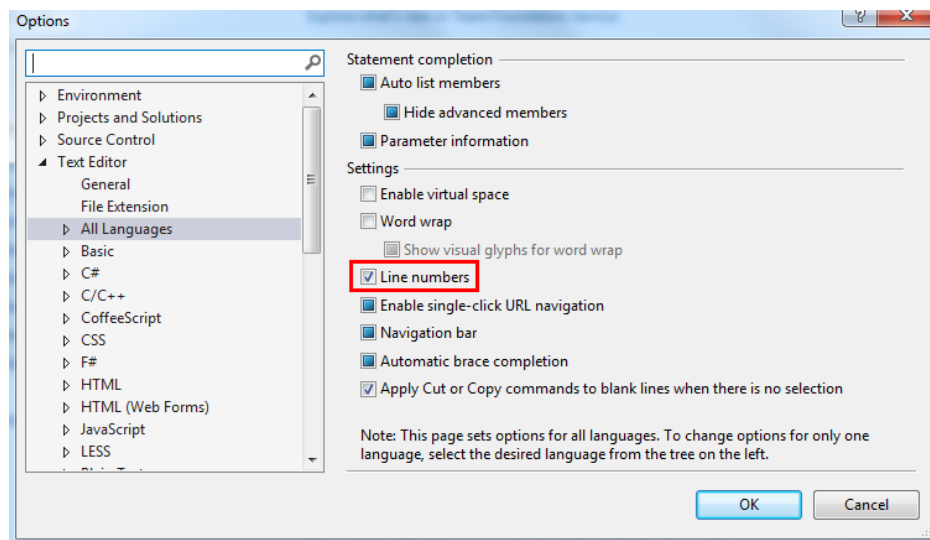
2. Altere Tema de cores para Escuro e então clique em OK.



As cores no Visual Studio devem corresponder à seguinte imagem:

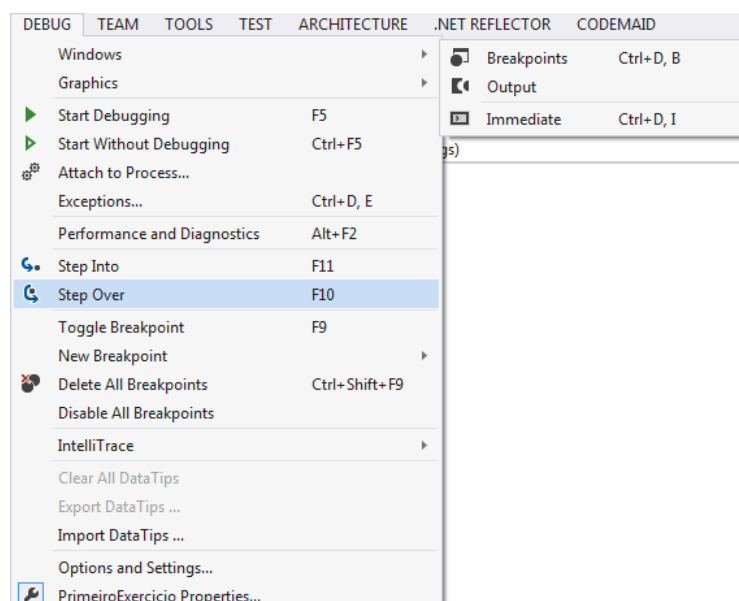


3. Na figura abaixo mostra como colocar número de linhas no código. Isso é muito importante, caso aconteça algum erro em tempo de compilação ou execução e mostre que o erro foi em determinada linha, você poderá achar com as linhas numeradas: **Tools, Options, Text Editor, All Languages, Selecione Line numbers** e clica em **OK**.



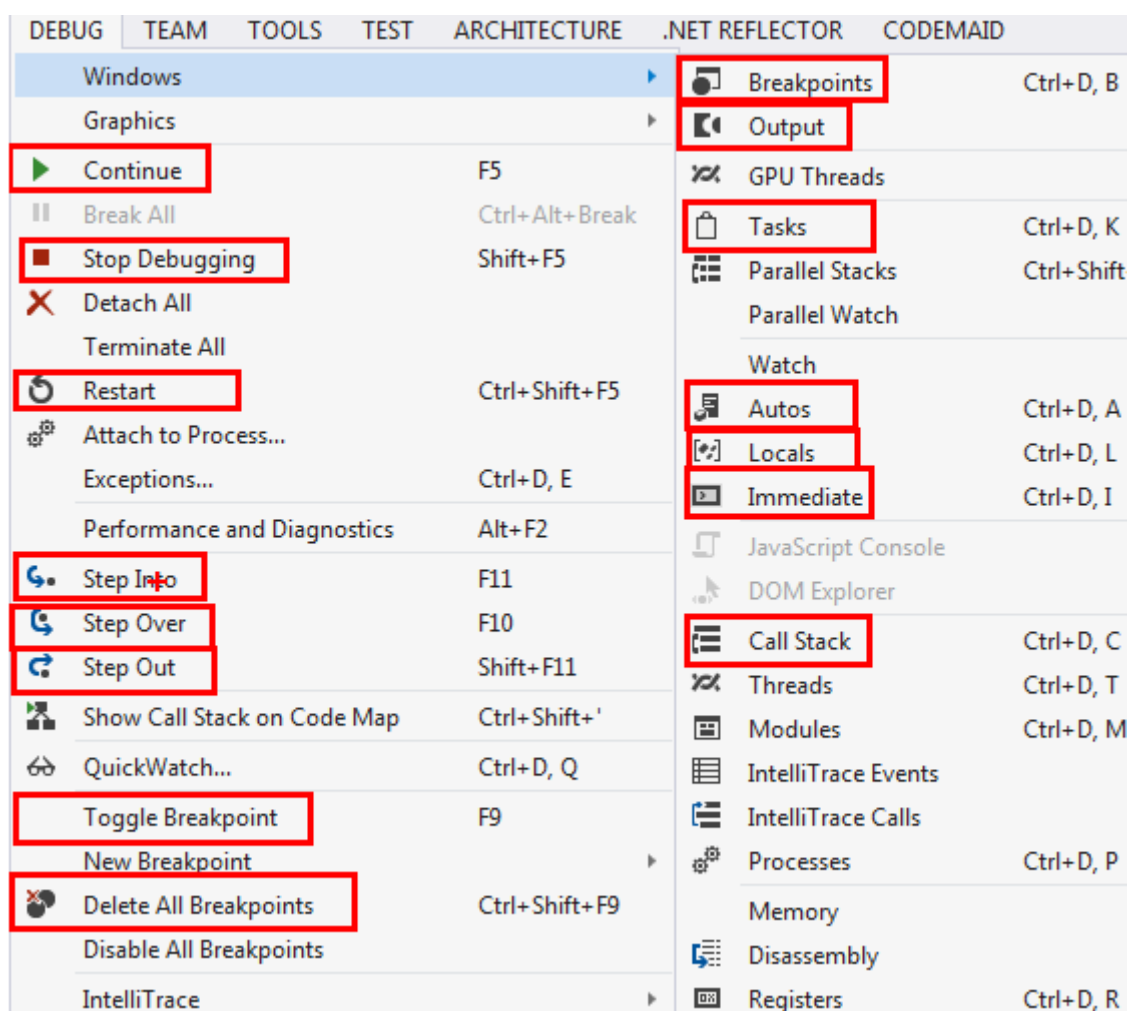
1.4.4 Debug

No menu superior os comando de passo a passo ficam na aba 'Debug'. Na imagem abaixo mostra todas as opções que você tem.



1.4.4 Debug (Passo a passo)

Quando fazemos o passo a passo de uma aplicação, colocamos sempre um ponto de parada ou breakpoints. Executamos a aplicação, e ela irá parar aonde você marcou. A imagem abaixo exibe as opções do Debug quando a aplicação está parada em um breakpoint:



Continue(F5): Continuará executar até o próximo breakpoint, caso não tenha mais nenhum ou não esteja em um laço de repetição ele termina o debugging ou a execução;

Stop Debugging(Shift+F5): Irá parar o debugging ou a execução;

Restart(Ctrl+Shift+F5): Ele irá reiniciar seu debugging ou sua execução;

Step Into(F11): Passo a passo, caso esteja passando em um método ou classe ele entrará nele e fará o passo a passo dentro dele;

Step Over(F10): Com ele passamos pelas linhas normalmente. Só que se estiver um método ou uma classe, ele passará sem podemos ver o que ele faz dentro desse método ou classe;

Step Out(Shift+F11): Às vezes sem perceber você entra em uma classe reservada do C#, normalmente utilizando o Step Into(F11). E você não quer passar pelas 2.000 linhas da aquela classe. Então usa-se esse comando. Para pular para o próximo passo;

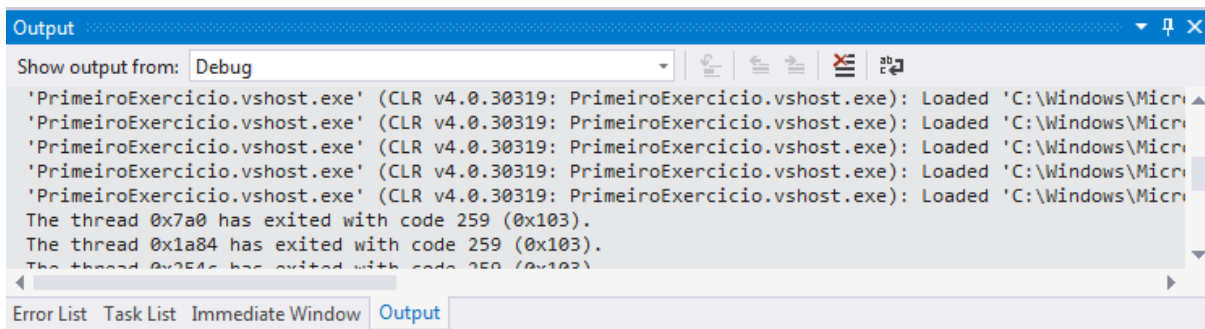
Tougggle Breakpoint(F9): Coloca a linha selecionada como ponto de parada;

Delete All Breakpoints(CTRL + ALT + O): Deleta todos os Breakpoint existentes;

1.4.4.1 Janelas Importantes

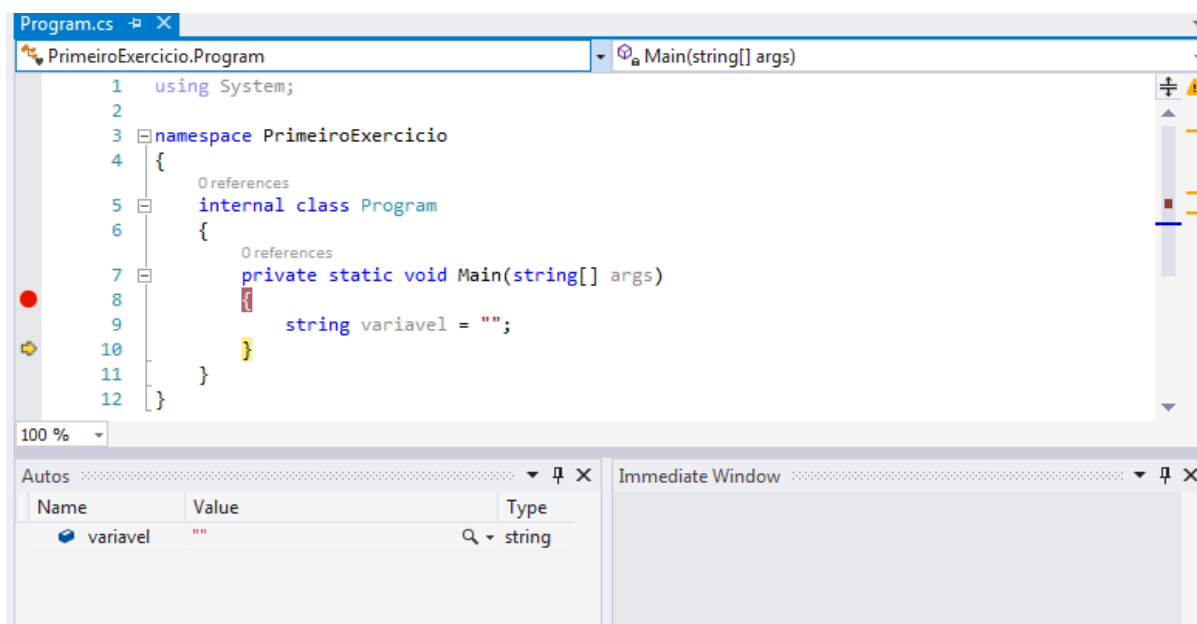
No Debug temos várias janelas muitas delas serviram para nos ajudar resolver problemas de nossas aplicações. Elas servem como ajudantes dos desenvolvedores. São elas:

Output: A output pode exibir mensagens de status para vários recursos no ambiente de desenvolvimento integrado (IDE). Para abrir a output na barra de menu, escolha **Exibir, Output** (ou clique em CTRL + ALT + O).



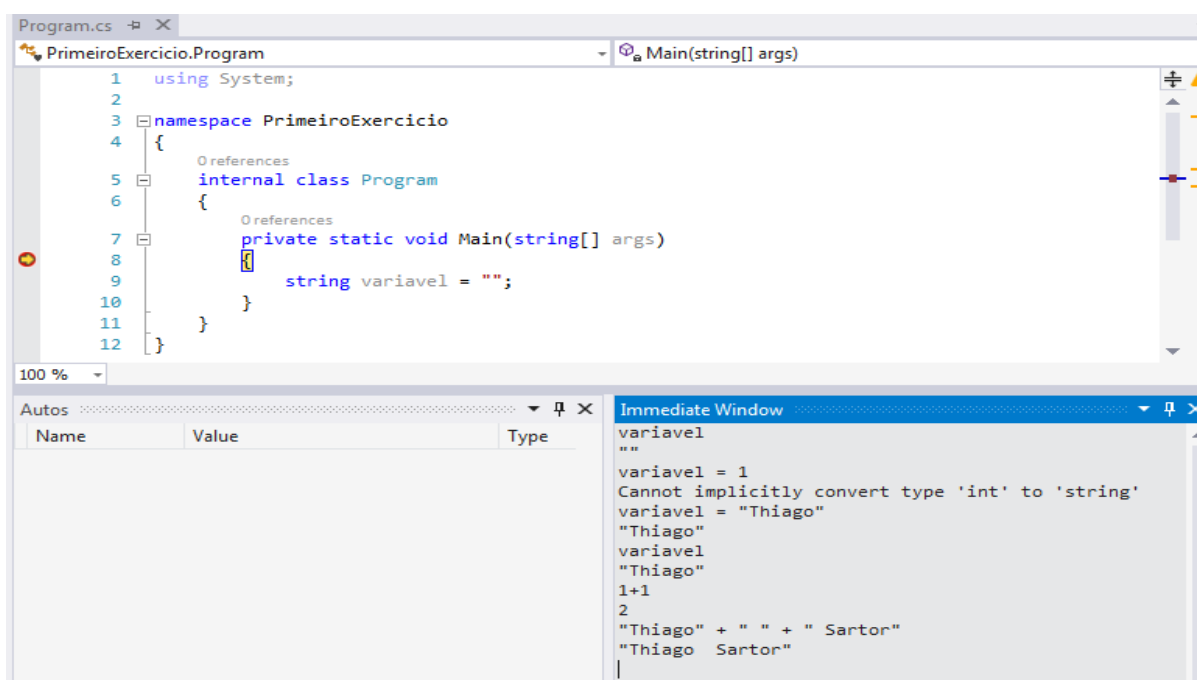
Immediate: é usada para depurar e avaliar expressões, executar instruções, os valores de variáveis de impressão, e assim por diante. Permite que você insira expressões a serem avaliadas ou executadas pela linguagem de desenvolvimento durante a depuração. Para exibir a janela **Immediate**, selecione **Debug** no menu superior, **Windows, Immediate**, ou pressione CTRL+ALT+I.

A imagem abaixo mostra a funcionalidade dessa janela:



Com o passo a passo, declaramos a variável do tipo string ou caracter. E ela está vazia. Vá para o menu Debug e selecione a janela immediate window como na imagem acima.

Ela funciona como uma janela de comandos. A imagem abaixo mostra as funcionalidades dessa janela:



Digite o nome da variável e pressione enter. Abaixo ela mostrará o valor que ela tem. Nesse caso ela está vazia. Então digitamos que essa variável agora vai receber o número 1.

Na próxima linha mostrará uma mensagem de erro. "Não é possível atribuir um valor inteiro para uma variável string" ou seja, devemos atribuir um valor string para ela. Na outra

linha a variável recebe “Thiago”. Agora se você digita o nome da variável e ela não está mais vazia. Está como valor que você atribuiu a ela. Nessa janela você pode fazer qualquer tipo de cálculo não só manipular suas variáveis. A imagem acima mostra outros exemplos de cálculos.

1.4.5 Erros

Mesmo os programadores mais experientes cometem erros, e saber como depurar um aplicativo e encontrar esses erros é uma parte importante da programação. E depurando nos ajuda a saber os tipos de erros que você precisará localizar e corrigir.

Erros de programação se encaixam em três categorias: erros de compilação, erros em tempo de execução e erros de lógica.

Erros de compilação: Erros de compilação, também conhecido como erros de compilador, são erros que impedem seu programa de executar. Quando você pressionar F5 para executar um programa, Visual C# compila o código em um idioma binário que o computador compreende. Se o compilador Visual C# encontra código que ele não entende, ele emite um erro de compilador.

A maioria dos erros de compilador são causados por erros que você faz ao digitar o código. Por exemplo, você pode errar uma palavra-chave, deixar sem algumas pontuações necessárias ou tentar usar uma instrução **pubic**. O certo seria **public**.

Erros de Tempo de Execução: Erros em tempo de execução são erros que ocorrem enquanto o programa é executado. Elas normalmente ocorrem quando o programa tenta uma operação que é impossível executar. Um exemplo disso é a divisão por zero. Suponha que você tinha a instrução a seguir:

$\text{Speed} = \text{Miles} / \text{Hours}$

Se a variável Hours possui um valor de 0, a operação de divisão falha e faz com que ocorra um erro em tempo de execução. O programa deve ser executado de modo a esse erro ser detectado, e se Hours contiver um valor válido, ele não ocorrerá.

Quando ocorrer um erro de tempo de execução, você pode usar as ferramentas de depuração no Visual C# para determinar a causa. Você aprenderá a localizar e corrigir erros em tempo de execução na lição Não funciona! Localizando e eliminando erros em tempo de execução.

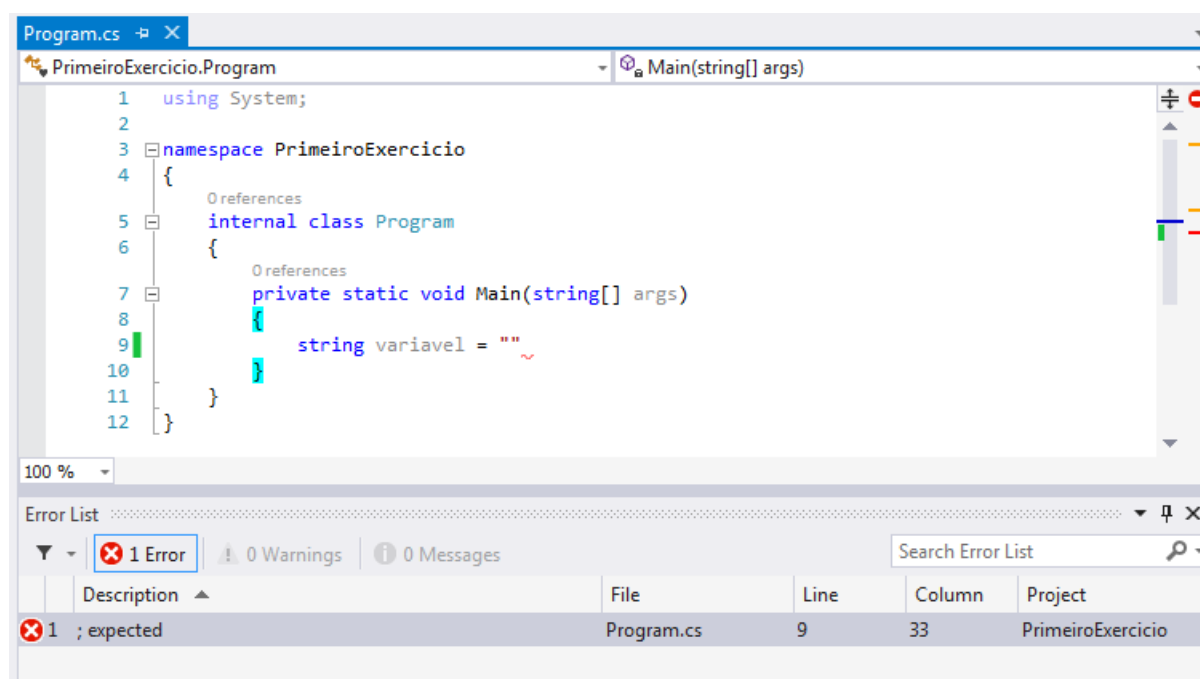
Erros de lógica: Erros lógicos são erros que impedem seu programa de fazer o que você pretendia fazer. Seu código pode ser compilado e executado sem erros, mas o resultado de uma operação pode produzir um resultado que você não esperava.

Por exemplo, você pode ter uma variável chamada *PrimeiroNome* que é inicialmente definida como uma sequência de caracteres em branco. Posteriormente no seu programa, você pode concatenar *PrimeiroNome* com outra variável chamada *SobreNome* para exibir um nome completo. Se você esqueceu atribuir um valor para *PrimeiroNome*, apenas o último nome seria exibido, não o nome completo como você pretendia.

Os erros lógicos são o mais difícil localizar e corrigir, mas Visual C# possui as ferramentas que facilitam, também este trabalho de depuração. Você aprenderá a localizar e corrigir os erros lógicos. Palavra que você vai usar muito quando descobrir um erro desses: **“Não acredito que eu fiz isso!”**

1.4.5.1 Janela de lista de erros

Erro List(Ctrl+W+E): Essa é uma das janelas muito utilizadas, porque nela é onde vemos os erros de execução e compilação. Na imagem abaixo na linha 9, “esquecemos” de colocar o ponto e vírgula no final da linha.



A janela descreve muito bem, que temos um erro e esse erro, é que se espera um ponto e vírgula na linha 9. Super descritivo!

3 CONCLUSÃO

Essa ferramenta é magnífica e a última versão na qual estamos utilizando é ainda mais surpreendente. Creio que todos os itens citados acima são os principais utilizados na programação que veremos no curso preparatório. Conforme as dúvidas surgirem, iremos aperfeiçoando esse manual. Espero que consiga ajudar vocês nesse primeiro contato com o Visual Studio 2013.

Bons estudos!