

Visual Studio .NET Academic Student Tools Guide

Information in this document, including URL and other Internet Web site references, is subject to change without notice. Unless otherwise noted, the example companies, organizations, products, domain names, e-mail addresses, logos, people, places and events depicted herein are fictitious, and no association with any real company, organization, product, domain name, e-mail address, logo, person, place or event is intended or should be inferred. Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Microsoft Corporation.

Microsoft may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from Microsoft, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

Microsoft, MS-DOS, Windows, Windows NT, MSDN, Visual Basic, Visual C++, Visual C#, Visual FoxPro, Visual SourceSafe, Visual Studio, the Visual Studio logo and Win32 are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

The names of actual companies and products mentioned herein may be the trademarks of their respective owners.

© 2001 Microsoft Corporation. All rights reserved.

11/01 Part No. X08-44749

Contents

CHAPTER 1 Getting Started	1
Student Tools Features	2
Installing Visual Studio .NET Academic	3
Visual Studio .NET Academic Integrated Development Environment	4
Solution Explorer	6
Dynamic Help Window	7
Class View Window	8
Task List	9
Output Window	10
Walkthrough: Creating a Console Application	11
Creating a Console Application	11
Opening a Source Code File	13
Adding Code to the Application	13
Building the Application and Fixing Errors	16
Running the Application	18
Student Resources	18
CHAPTER 2 Managing Courses and Assignments	21
Adding a Course	21
Working with Courses and Assignments	22
Downloading an Assignment Starter Project	23
Submitting an Assignment	25
Checking Assignment Status	27
Viewing Course Information	28
Reading Messages	29

Changing Your User Password	30
Deleting a Course	30
CHAPTER 3 Creating an Advanced Placement Console Application	31
Creating an AP Console Application	32
Adding AP Files to an Existing Application	33
CHAPTER 4 Working With Visual Studio .NET	35
Navigating Help	35
Using Dynamic Help	37
Accessing F1 Help	38
Setting Compiler Warning Levels	38
Debugging an Application	39
Debugging Overview	41
Breakpoints	42
Step Features	43
Variable Values	44
Call Stack	45
Walkthrough: Debugging a Sample Application	45
Printing the Output Window	47
Viewing Line Numbers	47
CHAPTER 5 Reference	49
Visual Studio .NET Quick Reference	50
Visual Studio .NET File Extensions	51
Compiler Warning Levels	54
Console Application Overview	54
Advanced Placement Classes	55
User Interface Reference	56
Student Resources, Visual Studio Start Page	56
Student Course Tools, Visual Studio Start Page	57
Add Course Page	57
Work With Courses Page	57
Download Starter Project Dialog Box	58

Delete Course Page	59
Course, Assignment Manager	59
Course Info Page	60
Course Assignments Page	60
Download Starter Project Page	61
Upload Submission Page	61
Messages, Assignment Manager	62
Read Page	62
Password, Assignment Manager	62
Change Password Page	63
Errors for Visual Studio .NET Academic Student Tools	63
Course assignment file is not valid. Check with instructor.	64
Course file <file name> already exists. Delete the existing course before adding the new course.	64
Course information is not available. Check with instructor.	64
Course file is not valid.	64
Enter a course URL.	65
Select course(s) to delete.	65
Starter project directory already exists. Specify another directory.	65
Student Add-in is not loaded.	65
Unable to copy the assignment files from the server path <server path>.	66
Unable to load course information from server or local copy.	66
Unable to open the startup file <file name>. Check with instructor.	66
Download location did not exist on local file system.	66
Invalid credentials. Please try again.	66
Root upload location not available.	67
The CourseID is missing or invalid. Please use the Core Academic Tools when accessing this page.	67
You are not registered for this course.	67
You have exceeded the maximum upload size for this project of <xx> MB. No more files may be uploaded.	67

CHAPTER 6 Samples	69
Hello World Sample	70
Tic-Tac-Toe Sample	70
Diff Tool Sample	71
Elementary Data Structures Sample	74
Expression Parser Sample	75
Sorting Sample	76
Tile Puzzle Sample	77
Towers of Hanoi Sample	78
Network Chat Sample	79
Debugging Sample	79

Getting Started

Microsoft® Visual Studio® .NET is the complete suite of tools for rapidly building Web applications and high-performance desktop applications. This suite includes powerful component-based development tools, such as Microsoft® Visual Basic® .NET, Microsoft® Visual C++® .NET, and Microsoft® Visual C#® .NET (pronounced C sharp). You will find a number of additional technologies that make it easier for you to design, develop, and deploy your applications. Also included is the Microsoft® MSDN® Library, which contains all the documentation for these development tools.

Microsoft® Visual Studio® .NET Academic incorporates everything in the Visual Studio .NET Professional Edition with a set of tools specifically designed for students, referred to as the Student Tools.

In This Section

Student Tools Features

Provides instructions on how to view your instructor's courses and assignments on your computer, create console applications, and submit completed assignments to your instructor or teaching assistant.

Installing Visual Studio .NET Academic

Provides instructions on how to install Visual Studio .NET Academic Student Tools.

Visual Studio .NET Academic Integrated Development Environment

Discusses some of the development tools that are provided with Visual Studio .NET.

Walkthrough: Creating a Console Application

Provides instructions on how to create a console application using Visual Studio .NET.

Student Resources

Provides instructions on how to locate the latest information from Microsoft for student developers.

Related Sections

Visual Studio .NET

Introduces Visual Studio and the Visual Basic, Visual C++, and Visual C# languages.

Locating Information

Provides instructions on how to use Help in Visual Studio through the table of contents (TOC), keyword index, and full-text search.

Navigating Code and Text

Provides instructions on how to move around in Visual Studio using various methods.

Student Tools Features

Visual Studio .NET Academic provides a set of course and assignment tools designed specifically for students, referred to as the Student Tools. With the Student Tools, you can easily view your instructor's courses and assignments on your computer, create console applications, and submit completed assignments to your instructor or teaching assistant.

Student Tools includes the following set of features:

Assignment tools


Access course and assignment information for the programming courses that you are enrolled in from within Visual Studio. Receive and submit assignments, monitor course or assignment changes, and check grades. For more information, see [Managing Courses and Assignments](#).

Console Application Wizard

Create a C or C++ console application using a wizard that is specifically designed for the student developer. For more information, see [Walkthrough: Creating a Console Application](#).

Advanced Placement Application Wizard

Write a console application that includes the standard C++ classes for the Advanced Placement test in computer science. For more information, see [Creating an Advanced Placement Console Application](#).

Note  This content is applicable only to high school students in the United States.

Student Resources page

Stay current with the latest articles and information for computer science students from Microsoft. For more information, see [Student Resources](#).

Samples

Improve your programming skills using the wide variety of code samples provided with the Student Tools. For more information, see [Samples](#).

Installing Visual Studio .NET Academic

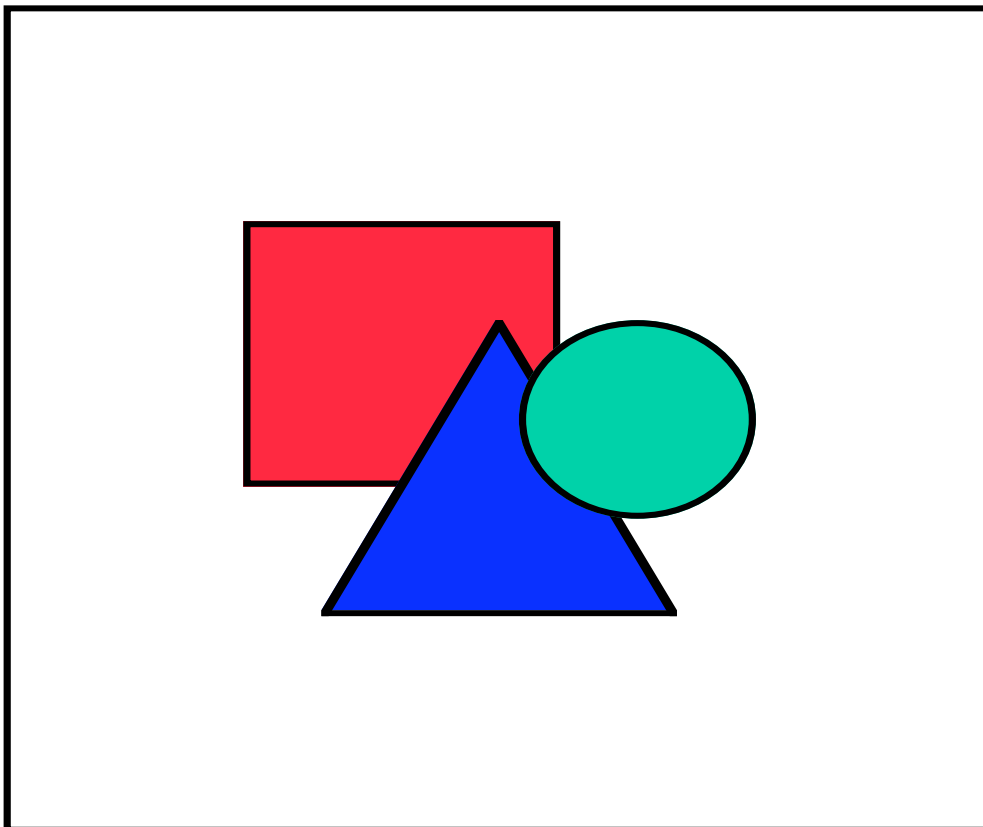
For general installation issues, refer to the Readme files, located in the root folder of the installation compact disc. These files:

- Contain detailed information on installation issues for all of the products in Visual Studio.
- Are in HTML format and can be viewed with an Internet browser, such as Microsoft® Internet Explorer.

For more information, see [Locating Readme Files and Installation and Setup](#) in the MSDN Library.

Visual Studio .NET Academic Integrated Development Environment

When you open Visual Studio .NET Academic for the first time, you see a version of the Visual Studio integrated development environment (IDE) called the Student Window Layout. The IDE is the group of tools, including a compiler, code editor, and debugger that you can use to develop software. These tools all run from a single user interface. The integration of all of these tools into a single application greatly simplifies development. In addition to these tools, Visual Studio features a number of development tools, including Solution Explorer, Dynamic Help window, Class View window, Task List, and Output window, as shown in the following screen shot.



In This Section

Solution Explorer

Discusses Solution Explorer, which is used to display and access the contents of a solution.

Dynamic Help Window

Discusses the Dynamic Help window, which provides a set of Help links that relate to the task you are performing in Visual Studio.

Class View Window

Discusses the Class View window, which displays the list of programming classes in a project.

Task List

Discusses the Task List, which displays errors, warnings, and specialized user comments in the active file.

Output Window

Discusses the Output window, which displays status messages for different features in the IDE.

Related Sections

Walkthrough: Creating a Console Application

Provides instructions on how to use the IDE to create a simple console application.

Student Resources

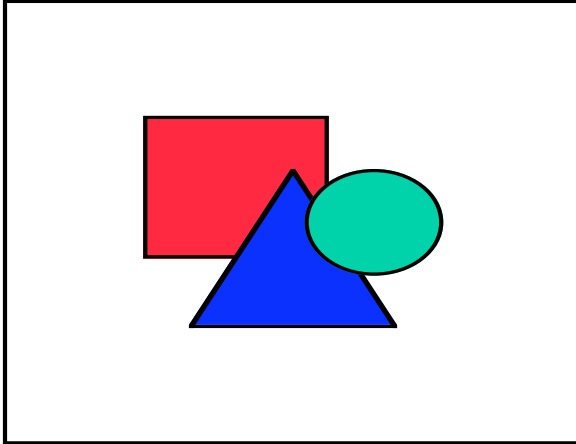
Provides instructions on how to access the latest articles and features from Microsoft for students.

My Profile, Start Page

Reviews the different window layouts available in Visual Studio .NET.

Solution Explorer

Solution Explorer displays an organized list of projects, project files, and directories that comprise the current solution. An example of Solution Explorer is shown in the following screen shot.

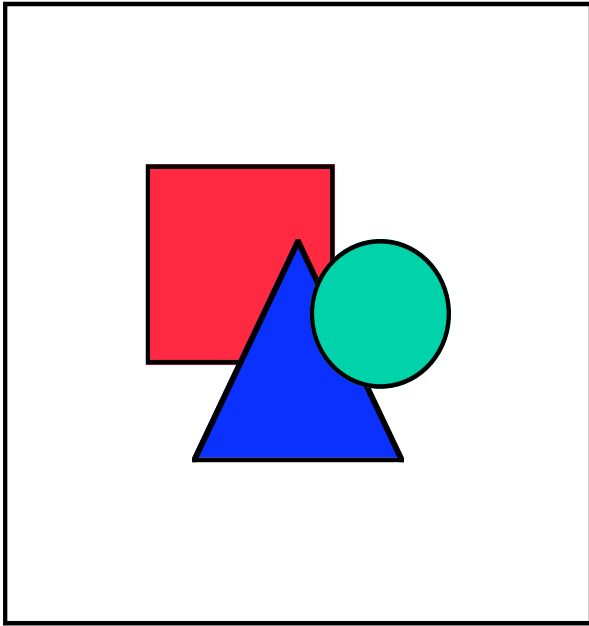


A solution is essentially a container for the projects and other items (for example, code, graphics, text files, and so on) from which an application is built. A single solution can contain multiple projects. Projects typically contain source code files, code header files, resource files, and other items, such as Readme files. For more information, see [Introduction to Solutions, Projects, and Items](#) in the MSDN Library.

Solution Explorer provides current information about the projects and items in the open solution. For example, you can monitor the status of items that are under source code control, drag items from one project to another, add items, and display or set solution, project, and file properties. For more information, see [Managing File Storage and Solution Explorer](#) in the MSDN Library.

Dynamic Help Window

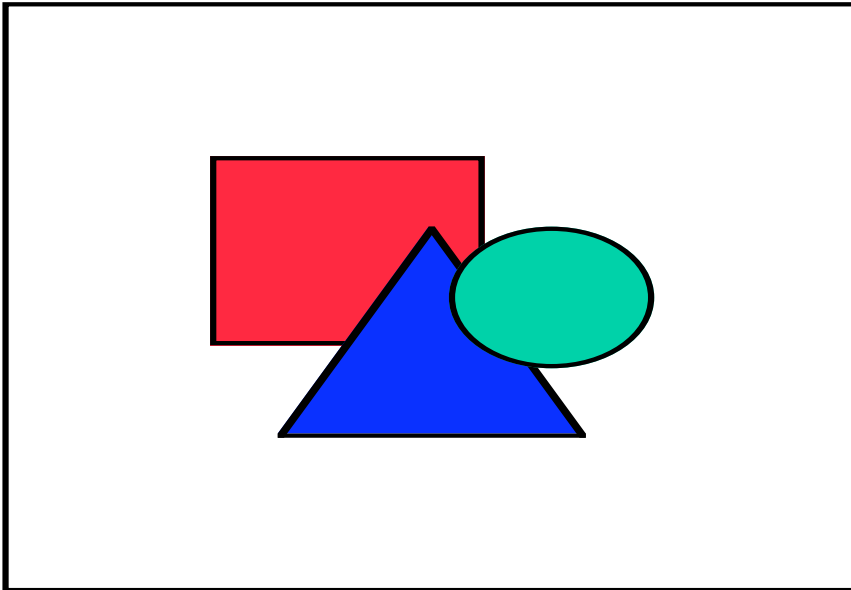
The Dynamic Help window provides a running list of links to Help topics that are available in the MSDN Library, as shown in the following screen shot. The topics pertain to the particular area of the integrated development environment (IDE) in which you are working currently and for language keywords in the Code Editor.



By tracking the selections you make, the placement of your cursor and the items selected (that is, the items with focus) within the IDE, Dynamic Help filters through topics available in the MSDN Library and provides pointers to relevant information specific to the development task currently being performed. For more information, see [Customizing Dynamic Help](#) in the MSDN library and [Navigating Help](#) in [Working With Visual Studio .NET](#).

Class View Window

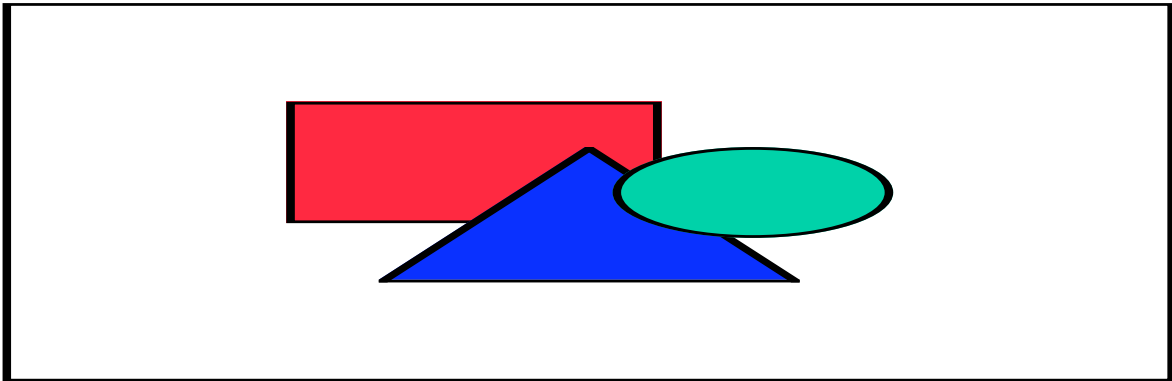
The Class View window displays the classes in each project, as shown in the following screen shot.



A class is a specialized data type used to create programming objects. The Class View window displays the list of global functions and variables, the list of classes, and class constructors, destructors, and methods. You can use the Class View window to view your source code in terms of the objects you create and to navigate through the classes in your code without needing to know which file contains each class. This makes it possible for you to view your code from an architectural standpoint rather than from a linear, or file-based, view. For more information, see [Class View in the MSDN Library](#).

Task List

The Task List makes it possible for you to mark your code with specialized comments that are then displayed as individual items in the Task List. In addition to showing user-defined tasks, the Task List also displays the list of compiler errors and warnings generated when building a file, as shown in the following screen shot.

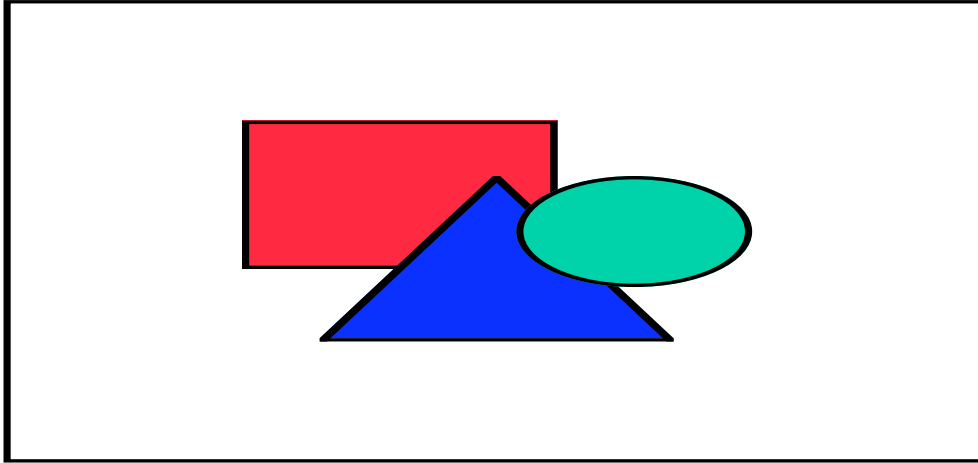


You can use the Task List to locate any of the items displayed in it. For example, if you double-click a compiler warning in the Task List, Visual Studio opens the appropriate editor and moves the insertion point to the specified location in the code. For more information, see [Task List Window](#) in the MSDN library.

You can also sort items in the task list by their type (for example, comment or compiler error), or you can filter the Task List to show only items of a certain type (for example, build errors). For more information, see [Task List Views](#) in the MSDN Library.

Output Window

The Output window, shown in the following screen shot, displays status messages for various integrated development environment (IDE) features, including compilation errors.



The IDE features provide information to the Output window. You can switch between the information from each feature using the Output list pane. For example, the Output window displays build information, as well as output from external tools, such as batch (.bat) files, which are typically displayed in the command window. A batch file is an ASCII text file that contains a sequence of commands for the operating system.

You can print the contents of the Output window. For more information, see [Printing the Output Window](#) in [Working with Visual Studio .NET](#) and [Output Window](#) in the MSDN Library.

Walkthrough: Creating a Console Application

This section guides you in creating your first simple C++ application with Visual Studio .NET. Use this walkthrough to become familiar with some of the development tools provided by the Visual Studio .NET integrated development environment (IDE), including the Task List, Solution Explorer, and the Code Editor. You will use these tools to develop your own applications as you become a more experienced programmer and as you gain experience with Visual Studio .NET. This walkthrough provides a basic introduction to Visual Studio and shows you how to create a Hello World application using Visual C++ .NET. The walkthrough involves the following steps:

- Creating a Console Application
- Opening a Source Code File
- Adding Code to the Application
- Building the Application and Fixing Errors
- Running the Application

Creating a Console Application

The first step in writing your application is to create a console application. A console application is a program that runs from the operating system's command line, rather than from a graphical user interface (GUI). The output of the console application is an executable (.exe) file, in other words, a program file that can be run. Use the following procedure to create a console application.

To create a new console application

1. From the Visual Studio start page, click the **Get Started** link, and choose **New Project**.

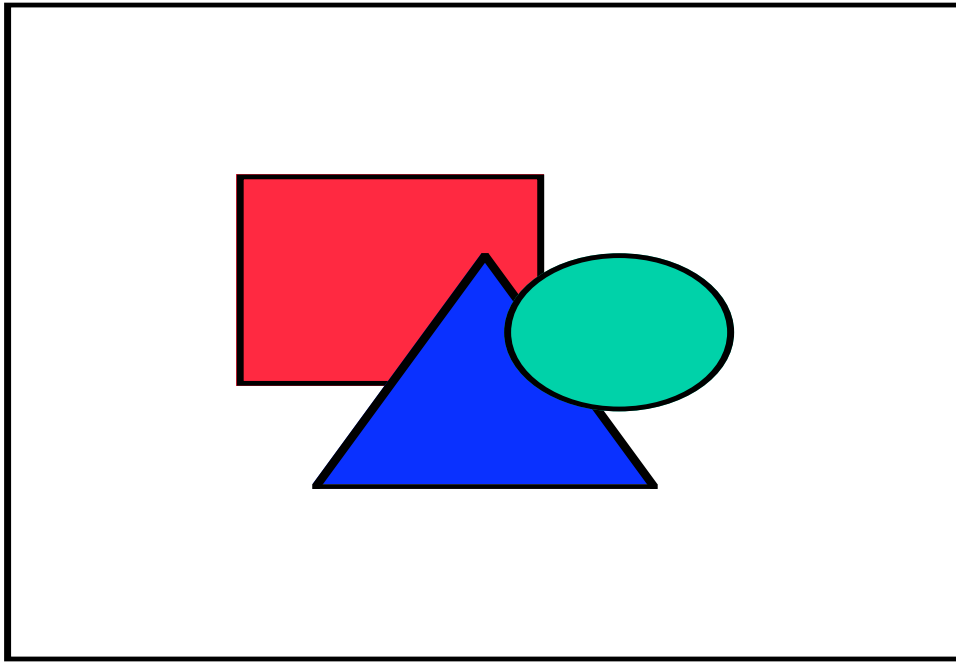
-or-

From the **File** menu, choose **New**, and then choose **Project**.

The **New Project** dialog box appears.

2. In the **Project Types** pane, choose the **Visual C++ Projects** folder.
3. In the **Templates** pane, choose the **C/C++ Console Application** template.
4. In the **Name** text box, type **MyFirstConsoleApp**. Visual Studio .NET automatically appends the correct file extension to the name based on the type of project you are creating. In this case, a .vcproj file extension is appended, which indicates that you are creating a Visual C++ project.
5. Click **Browse** to find an appropriate file location, or use the default location shown in the **Location** drop-down box. This is the location where the project files are stored.

The **New Project** dialog box is shown in the following screen shot.



6. Click **OK**.

Now, you have created a new C/C++ console application. The files in this project are shown in Solution Explorer. Using Solution Explorer, you can work with all projects that are open in Visual Studio currently, regardless of whether these projects are written in the same programming language. For more information, see [Introduction to Solutions, Projects, and Items in the MSDN Library](#).

The source code file opens automatically in the Code Editor. For more information, see [Opening a Source Code File](#).

Now, you are ready to add code to your application. For more information, see [Adding Code to the Application](#).

Opening a Source Code File

If your source code file, `MyFirstConsoleApp.cpp`, did not open automatically in the Visual Studio .NET Code Editor after you created your project, then use the following procedure to open your source code file.

To open a source code file

1. In Solution Explorer, expand the node for the **MyFirstConsoleApp** project by selecting the plus (+) sign to the left of the project name.

Three folders, entitled **Source Files**, **Header Files**, and **Resource Files**, appear. These folders contain the various files in your project.

2. Select the plus (+) sign located to the left of the **Source Files** folder.
3. Double-click the `MyFirstConsoleApp.cpp` file.

The Visual Studio Code Editor opens.

Now, you are ready to add code to your application. For more information, see [Adding Code to the Application](#).

Adding Code to the Application

Now that you have created a new console application and opened your code (.cpp) file in the Visual Studio Code Editor, you must add code to your application. The code you add will create a simple Hello World application.

To add code to the application

1. The console application template automatically adds some code to your application to get you started. The first line of code in your application is a comment line that reads as follows:

```
// MyFirstConsoleApp : Defines the entry point for the console application.
```

Comments are non-executable statements in code that make it possible for you to add information about your code. The Visual Studio Code Editor identifies all text following two forward slashes (//) and preceding the next carriage return as a comment in the C++ language. By default, comments in the C and C++ languages are shown in green in the Visual Studio Code Editor. For more information, see [Coding Techniques in the MSDN Library](#).

2. Replace this comment with the following text:

```
// Hello World Application
```

3. After the comment line, type the following lines of code, exactly as shown:

```
#include <iostream>
using namespace std;
```

This code is case-sensitive, so be sure to type it exactly as it is written. Notice that as you type `#include` and `using namespace`, they appear in blue. This identifies them as keywords in the Visual Studio Code Editor. These words have a special significance within the C++ language. If keywords are not colored when you type them into the Code Editor, then the keywords have been entered incorrectly.

4. The next two lines of code are already included in your application:

```
int main(int argc, char* argv[])
{
```

This code is always provided as part of the console application and is used to write your complete application. All C++ console programs contain the function `main`, inside which the body of the program is written. In this case, the `main` statement references parameters (`argc` and `argv[]`), which are used to receive command line arguments. For more information, see *The main Function and Program Execution* in the MSDN Library.

The `main` function is preceded by the keyword `int`, which indicates that this `main` function returns an integer value. The `main` function statement is followed by a left curly brace (`{`), which will be matched by a right curly brace (`}`) at the end of the `main` function.

5. On the line immediately following the left curly brace and immediately before `return 0`; type the following code:

```
cout<<"Hello World"<<endl
```

Note The last line of code contains the keyword `endl`, which will execute a carriage return (that is, an advance to the next line). The last character in this keyword is the lowercase letter `l`, not the number `1`.

The Code Editor automatically indents the code based on the indenting conventions of the Visual C++ language.

When you are done typing, your code should look like the following.

Note This code intentionally contains an error, which you will find and remove in subsequent steps in this walkthrough. For more information, see [Debugging an Application in Working with Visual Studio .NET](#).

```
// Hello World Application
#include <iostream>
using namespace std;

int main(int argc, char* argv[])
{
    cout<<"Hello World"<<endl
    return 0;
}
```

For more information about the colorizing features of the Visual Studio Code Editor, see [Editing Code, HTML, and Text in the MSDN Library](#). This colorizing feature, along with other features of the Code Editor, such as word completion and automatic brace matching, are referred to as IntelliSense[®] features. For more information, see [Using IntelliSense in the MSDN Library](#). IntelliSense features can be accessed by choosing the IntelliSense command from the Edit menu.

Now, you are ready to move onto the next step, which is building your application. For more information, see [Building the Application and Fixing Errors](#).

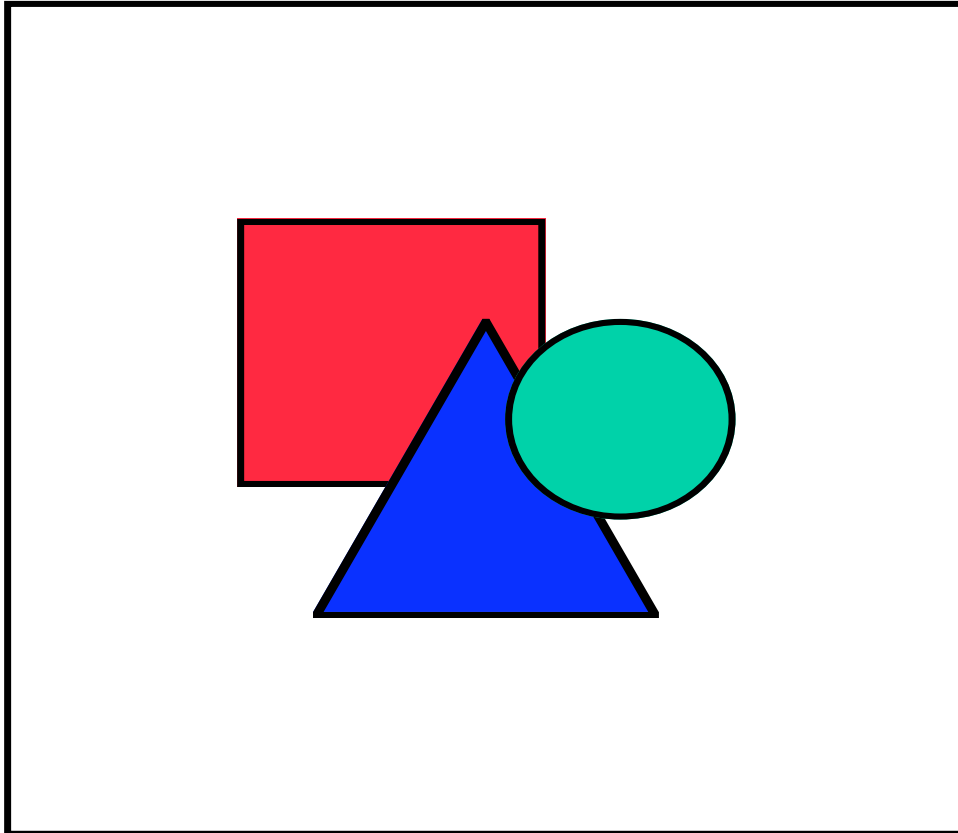
Building the Application and Fixing Errors

After you have typed the code into your source file, you are ready to build the executable file from your application.

To build the application

1. From the **Build** menu, choose **Build Solution**.

The results of your build are displayed in the **Task List**, as shown in the following screen shot.



- When you build your application, the following error is shown in the **Task List**.

error C2143L syntax error : missing ';' before 'return'

This compiler error indicates that a semicolon (;) is missing from the end of the line preceding the return 0; statement. This semicolon was left out of the code on purpose to demonstrate using the **Task List** to find build errors.

Note To access Help for this error, select the error, and press **F1**.

- The following information is shown in the **Output** window. This window is tabbed together with the **Task List** and therefore is not visible in the previous screen shot.

Build: 0 succeeded, 1 failed, 0 skipped

This statement indicates that the project did not build successfully. When a project or group of projects is built, the **Output** window displays the number of projects that were built successfully, the number of projects that failed to build, and the number of projects that were not selected for building.

To fix build errors

- Double-click the error in the **Task List**, and Visual Studio inserts a marker indicating where the error occurs.
- To fix this error, add a semicolon (;) immediately after <<endl, as shown in the following code snippet:

```
cout<<"Hello World"
<<endl;
```

- From the **Build** menu, choose the **Build** command.

The following information is shown in the **Output** window.

Build: 1 succeeded, 0 failed, 0 skipped

Your project now builds successfully. The final step is to run the executable file you built with your application. For more information, see [Running the Application](#).

Note An executable file is built by compiling and linking your console application to the object code from statically-linked libraries and other object files.

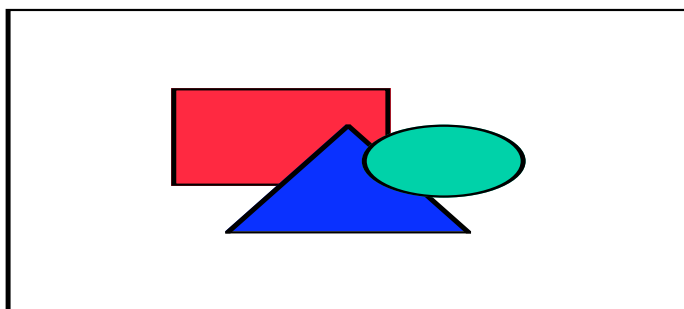
Running the Application

Now, it is time to run your Hello World program and see what it does. Use the following procedure to run your program.

To run the application

- From the **Debug** menu, choose **Start Without Debugging**, or press **CTRL+F5**.

Your program output is displayed in a command window, as shown in the following screen shot.



As you can see from the screen shot, the output of this program is simply the statement "Hello World". The "Press any key to continue" statement is added by Visual Studio and prevents the command window from closing before you are able to view the program output. If, instead, you select Start from the Debug menu, then the application runs, but the command window automatically closes at the end of program execution unless you set a breakpoint on the last line of the program. For more information, see [Breakpoints in Working With Visual Studio .NET](#).

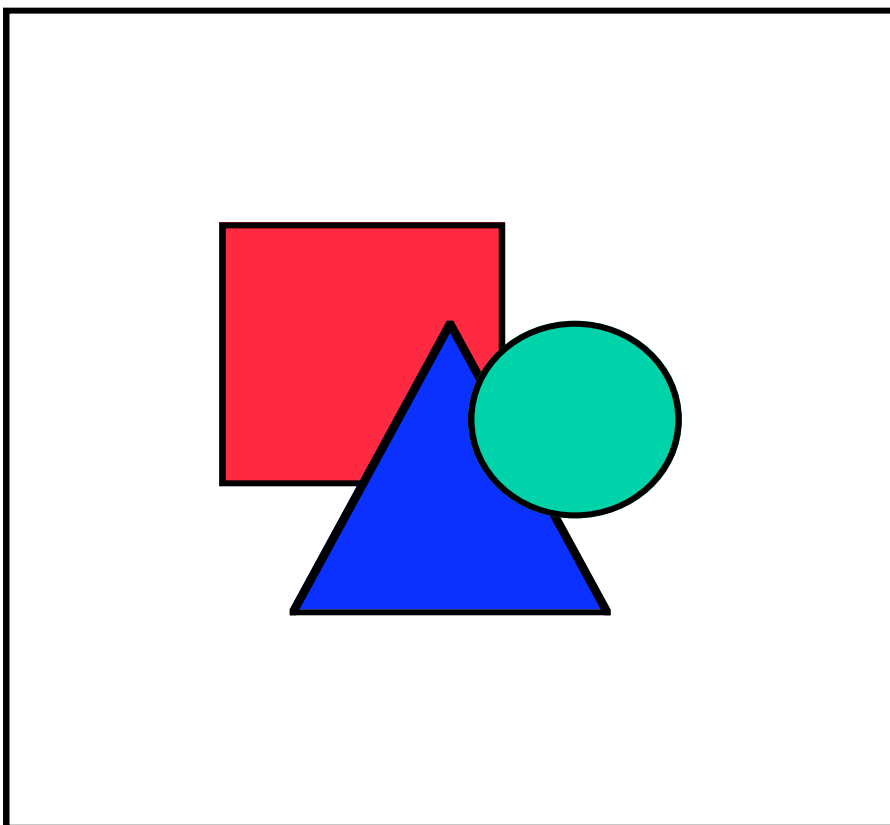
Student Resources

The Student Resources page provides links to the latest news, technical articles, samples and other information for students from Microsoft using the tabs described in the following table.

Tab	Content
News	Latest news from Microsoft for students.
Technical Articles	Current technical articles, including links to case studies and presentations.
Samples	Links to interesting samples for students in various languages.
Cool Stuff	Web sites and newsgroups for student developers.

The Student Resources page is updated with new content each time you click it. A sample of the content provided is shown in the following screen shot.

Student Resources page



BLANK PAGE

IMPORTANT: This text will appear on screen, but will not print on a PostScript printer.

This page should be the last one in this file; it was inserted by running the InsertBlankPage macro.

Do not type any additional text on this page!

Managing Courses and Assignments

This section guides you through using the assignment tools that are provided with Microsoft® Visual Studio® .NET Academic Student Tools. These tools make it possible for you to access course and assignment information provided by your instructor from within Microsoft® Visual Studio® .NET. Using Visual Studio, you can download starter projects for your homework assignments, submit the completed assignments to your instructor, and check your grade.

In This Section

Adding a Course

Provides instructions on how to access course information from within Visual Studio .NET.

Working with Courses and Assignments

Provides instructions on how to view course information, download new assignment starter projects, and submit completed assignments.

Deleting a Course

Provides instructions on how to delete a course in which you are no longer enrolled.

Related Sections

Working With Visual Studio .NET

Covers the basics of using Visual Studio.

Adding a Course

Before you can access the course information that your instructor has provided, you must add your instructor's course using the Student Tools. Your instructor will provide you with the path or URL that you must enter to access the course. Use the following procedure to add a course.

To add a course

1. From the Visual Studio .NET start page, choose **Student Course Tools**.
2. Click the **Add Course** link.
The **Add Course** page is displayed.
3. Type the path or URL that your instructor provided for the course in the text box and click **Add**.

4. If your instructor is using Assignment Manager, you also must enter your user name and password when prompted and click **Log On** to connect to the Assignment Manager server before the course is added. You will receive your user name and password from your instructor.

When you successfully add an Assignment Manager course, the Course Assignments page is displayed. For courses created using the assignment publishing tools, a message indicates whether your course was added successfully. For more information, see *Working with Courses and Assignments*.

Working with Courses and Assignments

After you have added a course using the Student Tools, you can access the information and assignments for that course by clicking the link for the course name on the Student Course Tools page. Then, you can view course information, download assignment starter projects, and submit completed assignments.

Your instructor can provide course and assignment information using either of two tools:

1. Assignment publishing tools
2. Assignment Manager

Different procedures are provided in this section for each tool. Ask your instructor which tool was used for your course.

In This Section

Downloading an Assignment Starter Project

Provides instructions on how to download a starter project for an assignment.

Submitting an Assignment

Provides instructions on how to submit a completed assignment using Assignment Manager.

Checking Assignment Status

Provides instructions on how to check the status and grade for an Assignment Manager assignment.

Viewing Course Information

Provides instructions on how to find information for your course.

Reading Messages

Provides instructions on how to read messages sent from your instructor.

Changing Your User Password

Provides instructions on how to change your Assignment Manager password.

Related Sections

Adding a Course

Provides instructions on how to access course information from within Visual Studio .NET.

Deleting a Course

Provides instructions on how to remove course information from Visual Studio .NET.

Downloading an Assignment Starter Project

When you add a course, all assignments currently available for that course are shown underneath the course listing on the Work With Courses page if the course was created using the assignment publishing tools or on the Course Assignments page if the course was created using Assignment Manager. As your instructor adds new assignments to this list, or updates existing assignments, these updates automatically appear when you open Visual Studio .NET and connect to the course server. The Student Tools automatically connect to the course server each time you open Visual Studio if the server is accessible.

Starter projects are provided by your instructor for you to use as a starting point for coding your assignment. Starter projects can be a blank project, or they can contain an incomplete solution to the assignment that has key sections of functional code removed. In this last case, your instructor will replace the sections that are removed with TODO comments that provide instructions for adding the code. These comments appear in the Task List when you open the starter project in Visual Studio.

Use the following procedure to download the starter project for an assignment created using the assignment publishing tools.

To download a starter project created using the assignment publishing tools

1. From the Visual Studio start page, choose **Student Course Tools**, and then click the course name link under **Work With Courses**.
The **Work With Courses** page is displayed.
2. Expand the **Course Assignments** node for the course that contains the assignment with which you want to work. A course can contain multiple assignments.
3. Expand the node of the assignment for which you want to download the starter project.
4. Click **Download Starter Project**.
The **Download Starter Project** dialog box appears.
5. In the **Project Name** text box, you can enter a new name for the starter project. By default, this text box contains the name that your instructor assigned to the starter project.
6. In the **Location** text box, enter the full path to where you want to create the project, or click **Browse** to find the location.

7. Click **OK**.

After the currently open files are saved, the start page for the assignment is opened, and the project is opened in a new solution in Solution Explorer.

8. Use **Solution Explorer** to view the code files contained in the starter project.

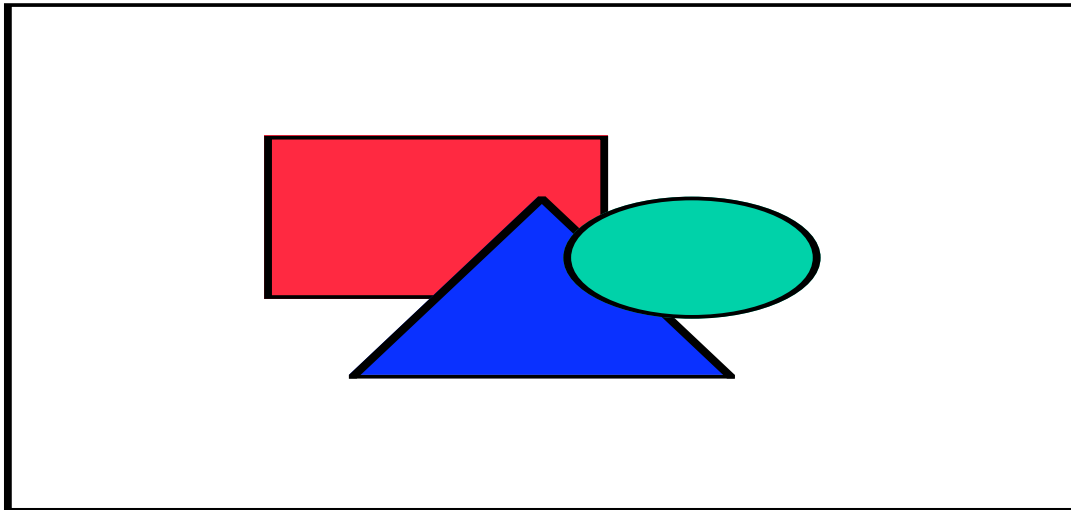
For more information, see Using Solution Explorer in the Microsoft® MSDN® Library.

Use the following procedure to download an assignment created using the Assignment Manager server.

To download a starter project created using the Assignment Manager server

1. From the Visual Studio start page, choose **Student Course Tools**, and then click the course name link under **Work with Courses**.
2. Log on to Assignment Manager, providing your user name and password.

The **Course Assignments** page appears showing the list of assignments in the course, the date each assignment is due, and so on, as shown in the following screen shot.



3. Click the icon in the **Get Starter** field for the assignment starter project you want to download, as shown in the following screen shot.



4. A default path for downloading the starter project is provided in the **Location to place downloaded files** text box. If this path is acceptable, click **Download**.

-or-

Browse to find or type the location of where you want to download the starter project to in this text box, and click **Download**.

If your instructor provides an assignment home page, then this page is opened in Visual Studio, and the project is opened in a new solution. The assignment home page can be used to provide additional information about the assignment. If no home page is provided, then the assignment details are displayed.

Use Solution Explorer to view the code files contained in the starter project; use the **Task List** to view your instructor's TODO comments. For more information, see Task List Window in the MSDN Library.

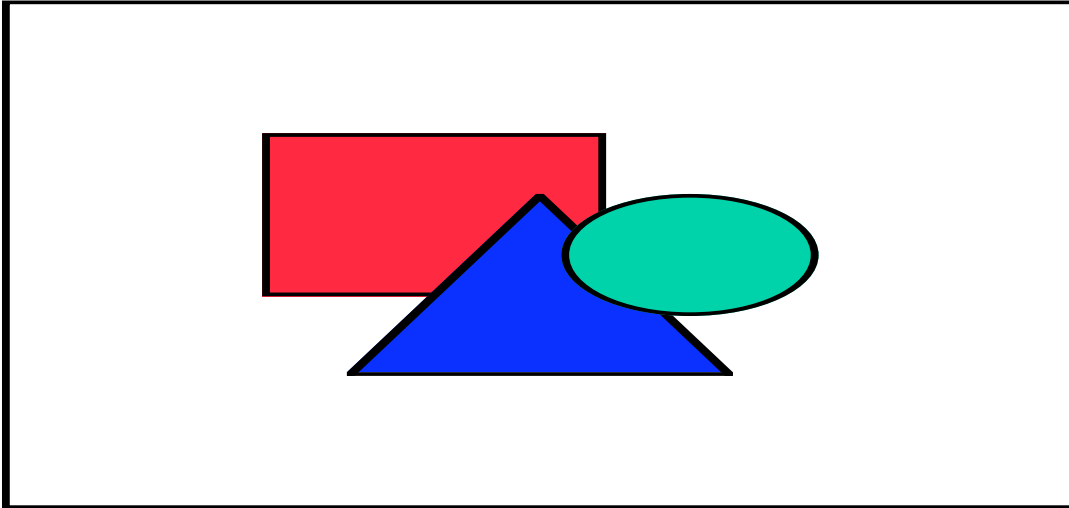
Submitting an Assignment

If your instructor uses Assignment Manager to create courses, then you can also submit your completed assignments using the Assignment Manager user interface. If your instructor does not use Assignment Manager, then submit your assignments by whatever method your instructor specifies.

To submit an assignment using Assignment Manager

1. Open your completed project in Solution Explorer.
2. From the Visual Studio start page, choose **Student Course Tools**, and then click the link for the course name under the **Work With Courses** heading.
3. Log on to the Assignment Manager server, providing your user name and password.

The **Course Assignments** page appears showing the list of assignments in the course, the date each assignment is due, and so on, as shown in the following screen shot.



4. Click the icon in the **Submit Project** field for the completed assignment that you want to submit to your instructor, as shown in the following screen shot.



The **Upload Submission** page appears. The name of the assignment you are working with is displayed in the **Assignment Name** text box.

5. In the **Select Project to Upload** drop-down list, choose the project that you are submitting for your completed assignment.

Note Your completed project must be open in Solution Explorer to be shown as an option under **Select Project to Upload**. If you access the **Select Project to Upload** page before opening your project in Solution Explorer, click the **Refresh Browser** button on the toolbar to refresh the screen and show the project in the drop-down list.

6. Click **Browse** to find additional files to include with your submission in the **Extra Files** field, and then click **Add Files** to include these files with your submitted assignment.

The list of extra files is displayed in the text box located above the **Upload Project** button. Examples of extra files include an executable file for the working assignment, or a Word document containing answers to a problem set.

7. Click **Upload Project** to submit your completed assignment and all associated files.

When you submit an assignment, the date and time of submission is shown in the Date Last Submitted field on the Course Assignments page. If Auto Build and Auto Check features are enabled on the Assignment Manager server that your instructor is using, then the Auto Build and

Auto Check fields display icons indicating that these tests are pending. Otherwise, solid lines are displayed in these fields. Check your assignment before the due date to determine whether your completed assignment passed auto build and auto check. For more information, see [Checking Assignment Status](#). If your assignment failed either of these tests and your instructor permits multiple submissions, then you can resubmit a fixed version. Each new submission replaces the previous submission. Check with your instructor to find out whether they allow multiple submissions on their assignments.

Checking Assignment Status

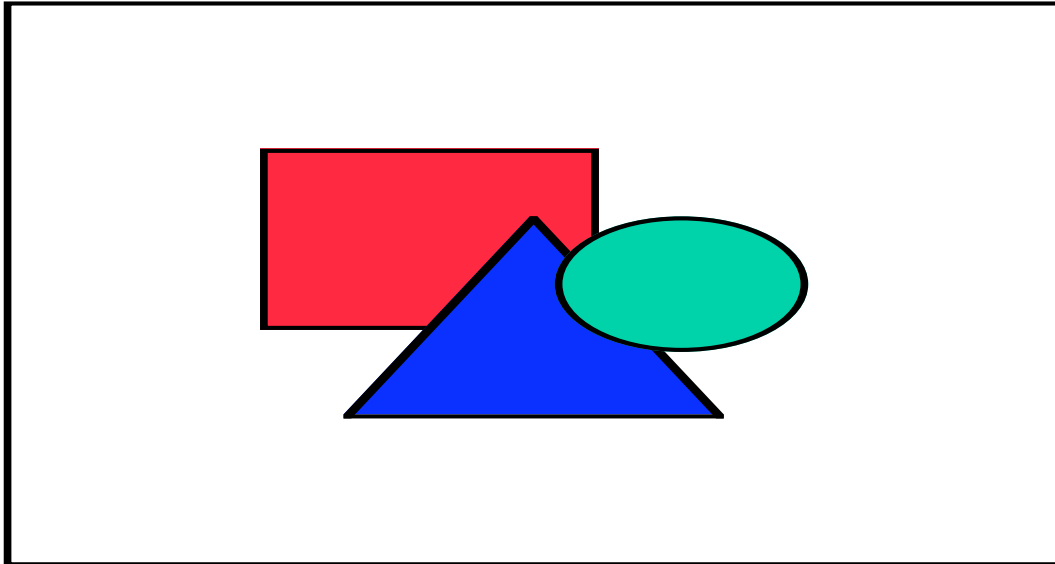
If you submitted your assignment using Assignment Manager, then you can also use Assignment Manager to check the status of your assignment. This includes checking the assignment grade and whether the assignment passed auto build and auto check. Auto build is a feature that makes it possible for your instructor to build your file automatically using a make file. Auto check makes it possible for the instructor to compare the output of your built executable (.exe) file to the output of their executable file, while specifying the same input for both files.

Use the following procedure to check assignment status.

To check assignment status

1. From the Visual Studio start page, choose **Student Course Tools**, and then click the link for the course name under the **Work With Courses** heading.
2. Log on to the Assignment Manager, providing your user name and password.

The **Course Assignments** page appears with a table of assignment information including assignment name, due date, date last submitted, grade, and so on, as shown in the following screen shot.



3. Your assignment status is displayed in the **Auto Build**, **Auto Check**, and **Grade** fields.
4. After you submit your assignment, the **Auto Build** and **Auto Check** fields will display an icon indicating whether the action is completed, pending, or has failed if your instructor has enabled the Auto Build and Auto Check features on the Assignment Manager server. Otherwise, a solid line is displayed in these fields. A key to the icons used is provided at the bottom of the **Course Assignments** page.
5. The **Grade** field will display a value of **N/A** if your instructor has not yet assigned a grade. A grade value is displayed in their field after your instructor has entered a grade for your assignment. Click the **Grade** field to display the grade details.

Viewing Course Information

You can view course information to find a description of the course, the course home page, or a list of resources or course links that are available for the course.

Use the following procedure to view information for an assignment publishing tools course.

To view assignment publishing tools course information

1. From the Visual Studio start page, choose **Student Course Tools**, and then click the link for the course name under the **Work With Courses** heading.

The **Work With Course** page appears.

2. Expand the **Course Description** node to view the description of the course.
3. Expand the **Course Links** node to view the list of e-mail addresses, Web sites, newsgroups, and so on that your instructor has provided as resources for this course.
4. Expand the **Course Assignments** node to view the list of course assignments.

Use the following procedure to view information for an Assignment Manager course.

To view Assignment Manager course information

1. From the Visual Studio start page, choose **Student Course Tools**, and then click the link for the course name under the **Work With Courses** heading.

2. Log on to the Assignment Manager, providing your user name and password.

The **Course Assignments** page appears.

3. Click the **Course Info** tab.

The **Course Info** page is displayed, and lists the **Course Name** and **Description**.

4. Expand the **Course Resources** node to view the list of e-mail addresses, Web sites, newsgroups, and so on that your instructor has provided as resources for this course.

Reading Messages

In Assignment Manager, you can read messages from your instructor or teaching assistant regarding course or assignment changes.

To read a message

1. In the left pane of Assignment Manager, click **Messages**.
2. Click the **Read** tab.
3. Click the subject of the message you want to read.

The message is displayed in a new window.

Messages cannot be deleted, but they will expire at the time and date set by your instructor.

Changing Your User Password

The Password page makes it possible for you to change the password you use to log on to Assignment Manager.

Note Passwords are case-sensitive.

To change a user password

1. Click **Password** in the left pane of Assignment Manager.
2. In the **Current Password** text box, enter your current password.
3. In the **Enter a New Password** text box, enter your new password.
4. In the **Confirm New Password** text box, enter your new password again.
5. Click **OK**.

Deleting a Course

Use the following procedure to delete a course.

To delete a course

1. From the Visual Studio .NET start page, choose **Student Course Tools**.
2. Click the **Delete Courses** link.

The **Delete Course** page is displayed with the list of currently installed courses.

3. To mark one or more courses for deletion, select the individual course(s) under **Currently Installed Courses**.

-or-

To mark all courses for deletion, click **Select All**.

4. Click **Delete Selected Courses**.

A message box appears and prompts you to confirm whether you want to delete the selected courses.

5. Click **OK** to complete the deletion, or click **Cancel** to stop.

Creating an Advanced Placement Console Application

The Advanced Placement (AP) test for computer science tests a high school student's basic understanding of object-oriented programming using the C++ programming language. The test evaluates a student's ability to write and analyze programs and to provide appropriate documentation for code. Microsoft® Visual Studio® .NET Academic provides an AP console application that includes the five different C++ classes commonly used to study for the AP test. C++ contains a number of pre-defined data types, such as int, char, and float. A class is a custom data type that you define. For more information about the Advanced Placement test for computer science, see the C++/AP Resources page (<http://www.mainfunction.com/resources/ap.asp>).

Note This content is applicable only to high school students in the United States.

In This Section

Creating an AP Console Application

Provides instructions on how to create a new Advanced Placement Console Application.

Adding AP Files to an Existing Application

Provides instructions on how to add Advanced Placement files to an application.

Related Sections

Advanced Placement Classes

Documents the C++ classes used in the AP test.

Console Application Overview

Discusses console applications.

Creating an AP Console Application

Use the following procedure to create an Advanced Placement (AP) console application with AP classes.

To create a new Advanced Placement console application

- Create a console application. For more information, see [Creating a Console Application in Getting Started](#).

Note Instead of choosing the **C/C++ Console Application** template, choose the **AP C++ Console Application** template to create a console application that contains Advanced Placement classes.

The following Visual C++ source code (.cpp) files make up the Advanced Placement Console Application:

main.cpp

Base file that you modify to be the primary source code file for the project.

apstring.cpp

String class. Compile this file to get the definition of the apstring class. All other classes are added using `#include` statements in **main.cpp**.

apmatrix.h, apqueue.h, apstack.h, apstring.h, apvector.h

Header files for the AP classes. These classes are added to **main.cpp** using **#include** statements.

readme.txt

Readme file that you modify to be the informational file for the project. This file provides a description of the files included in your project. Use this file to add general information regarding individual files or your application as a whole.

Adding AP Files to an Existing Application

The Student Tools make it easier for you to add Advanced Placement (AP) classes to your existing applications. Information about each of the AP classes is provided at the AP Computer Science page (<http://www.collegeboard.org/ap/computer-science/>). You can use this information to guide you in the use of these classes.

To add Advanced Placement files to an existing application

1. Open your project in Visual Studio .NET.
2. In **Solution Explorer**, right-click your project, and choose **Properties**.

The **ProjectName Property Pages** dialog box appears.

3. In the left pane, click **C/C++**, and then click **General**.
4. Under **Additional Include Directories**, type the following path:

C:\Program Files\Microsoft Visual Studio .NET\VC7\include\AP

Note C: represents the drive location where Visual Studio is installed and should be replaced as appropriate.

5. Click **Apply**, and then click **OK**.
6. In **Solution Explorer**, double-click your main application file. This is a Visual C++ source code (.cpp) file.

The file opens the Code Editor.

7. Type the following lines of code at the beginning of the file:

```
#include "apvector.h"
#include "apstring.h"
#include "apqueue.h"
#include "apstack.h"
#include "apmatrix.h"
```

You have created an application that references the Advanced Placement classes. If you want to have copies of these files in your application, then copy all of the files from C:\Program Files\Microsoft Visual Studio .NET\VC7\include\AP to the local directory where your main application file is located, and then add the include statements as specified earlier.

BLANK PAGE

IMPORTANT: This text will appear on screen, but will not print on a PostScript printer.

This page should be the last one in this file; it was inserted by running the InsertBlankPage macro.

Do not type any additional text on this page!

Working With Visual Studio .NET

This section covers some of the basic tasks that you can perform with Microsoft® Visual Studio® .NET.

In This Section

Navigating Help

Provides instructions on how to use the different types of Help available in Visual Studio .NET.

Setting Compiler Warning Levels

Provides instructions on how to set the level of warnings supplied by the Microsoft® Visual C++® compiler.

Debugging an Application

Discusses basic debugging features in Visual Studio.

Printing the Output Window

Provides instructions on how to print the contents of the Output window.

Viewing Line Numbers

Provides instructions on how to view line numbers in the Code Editor.

Related Sections

Developing with Visual Studio .NET (Microsoft® MSDN® Library)

Provides instructions on how to develop applications in Visual Studio.

Navigating Help

In Visual Studio .NET, Help is integrated fully within the development environment. You can access Help topics in Visual Studio .NET by means of the F1 button or the Dynamic Help, Contents, Index, and Search windows. This section focuses on how to obtain Help using the Dynamic Help window and F1 Help. Links to topics on accessing Help through the Search, Index, and Contents windows are also provided along with a link on how to use Help filters. Filters reduce and focus the Help content displayed in the various Help windows.

In This Section

Using Dynamic Help

Learn how to use the Dynamic Help window to access Help for the task you are currently performing in the environment.

Accessing F1 Help

Learn how to use F1 Help to obtain Help for a window, dialog box, or language keyword.

Related Sections

Strategies for Using Help (MSDN Library)

Learn more about the different options for accessing Help and when to use each option.

Navigating with the Table of Contents (MSDN Library)

Learn how to use the Help table of contents (TOC) to find information.

Finding Information with the Index (MSDN Library)

Learn how to use the list of keywords provided in the index to locate Help topics.

Finding Information with Full-Text Search (MSDN Library)

Learn how to use the search feature to locate Help based on the occurrence of a particular word or phrase in the text of Help topics.

Creating and Using Filters (MSDN Library)

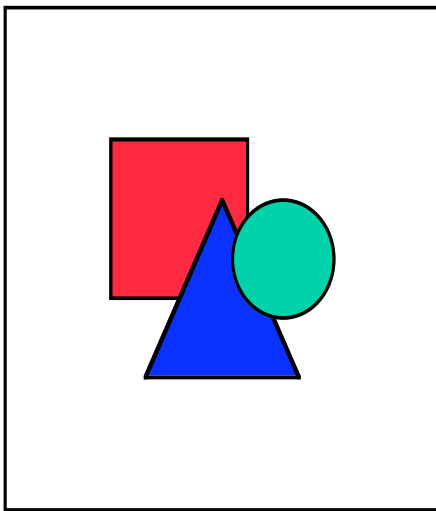
Learn how to use Help filters to reduce the list of individual topics displayed in the TOC, keywords shown in the index, or topics queried during a full-text search.

Help on Help (MSDN Library)

Learn how to use the Help viewer integrated into Visual Studio .NET.

Using Dynamic Help

The Dynamic Help window provides immediate access to Help topics pertinent to any task you are performing in Visual Studio. By tracking the selections that you make, the placement of your cursor, and the items in focus within the integrated development environment (IDE), Dynamic Help provides you with a list of pointers to related topics in the MSDN Library. For example, when you first open Microsoft® Visual Studio® .NET Academic, the Dynamic Help window provides links to sample, walkthrough, and other introductory topics, as shown in the following screen shot.



The Dynamic Help window separates the information it displays into different categories, including Help, Actions, Miscellaneous, and Samples. Links within these categories are displayed according to their relevance, with the most relevant topics listed first. You can choose to display Help within the IDE (as a separate window), or outside of the IDE in a stand-alone version of the MSDN Library.

Use the following procedure to display the Dynamic Help window, if it is not showing currently.

To view the Dynamic Help window

- From the **Help** menu, choose **Dynamic Help**.
- or-
- Press **CTRL+F1** on your keyboard.

If no relevant information can be found, the Dynamic Help window displays the following text:
"No links are available for the current selection."

Accessing F1 Help

F1 Help is available for user interface (UI) elements, such as windows, dialog boxes, message boxes, menu items, or toolbar buttons, and for key terms, such as keywords in the language editor.

Use the following procedure to view F1 Help in the IDE.

To view F1 Help

1. Select the UI element or language keyword for which you want to view F1 Help.
2. Press **F1**.

The contents of the Help topic are displayed in a new document window.

The list of topics that are either displayed in the Dynamic Help window or displayed when you press F1 are retrieved by Visual Studio from the available Help collections. For more information about Help collections, see *Specifying a Preferred Help Collection in the MSDN Library*.

Setting Compiler Warning Levels

A compiler is a program that translates the code in your source files to machine code that can be executed by the system's central processing unit (CPU). You can set the level of warnings supplied by the Visual C++ compiler to make it more or less sensitive to possible coding errors in your files, or you can disable specific compiler warnings (but not compiler errors). The warning level is a numeric value that specifies the highest level of warning generated by the compiler. Valid warning levels range from 0 to 4. For more information, see *Compiler Warning Levels in Reference*.

Note For additional compiler options to set from the command line, click the Command Line property page, and type the compiler option in the Additional Options text box. For more information, see *Setting Compiler Options and C/C++ Building Reference in the MSDN Library*.

You can change compiler sensitivity or disable specific compiler warnings and errors on a project-by-project basis using the following procedure.

To set compiler warning levels

1. Open the project's **Property Pages** dialog box.
 - a. In Solution Explorer, choose a project.
 - b. From the **View** menu, click **Property Pages**, or press **SHIFT+F4**.

The **Property Pages** dialog box for your project appears.

- c. Using the **Configuration** list, select the type of configuration for this project's properties.

Configuration properties include **Active**, **Debug**, **Release**, and **All Configurations**. For more information, see *Debug and Release Configurations* in the MSDN Library.

Note Using this procedure, you can also change properties for individual files within your project by selecting a file in your project instead of selecting the entire project. In this case, it is important to be aware of the differences among individual file warning levels when viewing compiler errors in **Task List**.

For more information, see *Specifying Project Settings with Property Pages* in the MSDN Library.

2. Click the **C/C++** folder.
3. Click the **General** property page, and modify the **Warning Level** property to the desired warning level.
4. Click the **Advanced** property page, and modify the **Disable Specific Warnings** property if you want to disable individual warnings.

A specific letter and number combination identifies each compiler warning. For example, compiler warning C4013 indicates that the compiler encountered a call to an undefined function. **The Disable Specific Warnings** option makes it possible for you to disable a compiler warning by specifying that warning's letter and number combination. Thus, if you specified a value of C4013, then you would prevent the compiler from listing this warning in **Task List** if you had any undefined functions in your project. For a list of compiler warnings, see *C/C++ Build Errors* in the MSDN library and view the sections on compiler warnings.

Debugging an Application

Debugging is a very complex topic with entire books written on how to approach debugging applications. This section is intended to provide only a basic overview of debugging and an introduction to debugging in Visual Studio.

In This Section

Debugging Overview

Includes an overview of the steps that make up the debugging process.

Breakpoints

Provides an overview of how breakpoints are used in code to instruct the program in when or where to pause during execution.

Step Features

Provides instructions on how to step through your code by unit of code or by breakpoint.

Variable Values

Provides instructions on how to watch the values of variables change as the program progresses.

Call Stack

Provides instructions on how to view the function or procedure calls that are currently on the stack.

Walkthrough: Debugging a Sample Application

Provides instructions on the how to debug a simple application using Visual Studio .NET.

Related Sections

Debugging Sample

Provides a sample to be used in conjunction with the topic Walkthrough: Debugging a Sample Application.

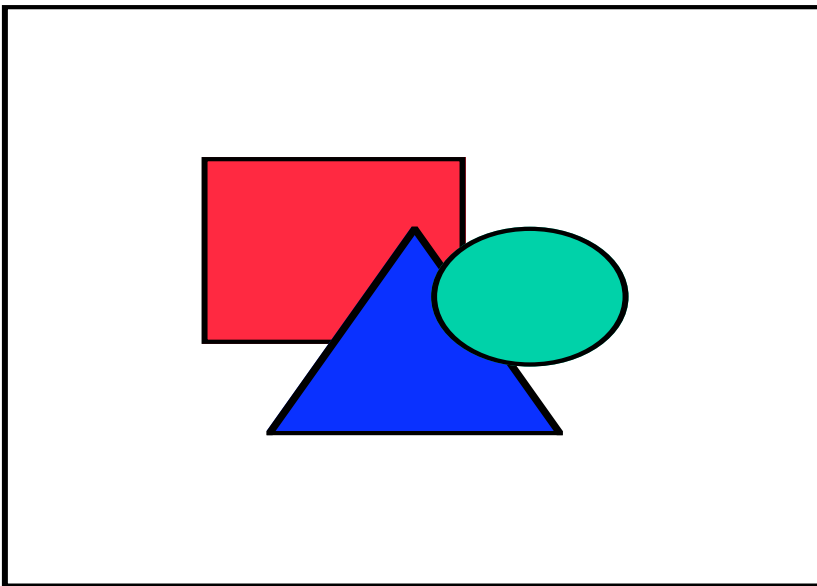
Debugging (MSDN Library)

Discusses, in depth, the debugging features provided with Visual Studio.

Debugging Overview

After you have created your application and resolved the build errors shown in the Task List, it is time to find and correct the code and logic errors that prevent your application from running correctly. You can do this with the debugger in Visual Studio, which is available for the Visual C++, Visual Basic, and Visual C# programming languages. The debugging process is shown in the following diagram.

Overview of the debugging process



Debugging is an iterative process that can be divided into five steps:

1. Testing your application to find errors (bugs) in how it functions.
2. Identifying bugs in the application.
3. Locating the bugs found during testing in code using the debugger.
4. Fixing the bugs in your code.
5. Rebuilding and compiling your application and then verifying the bug fix.

Repeat the process until you have located and resolved all bugs.

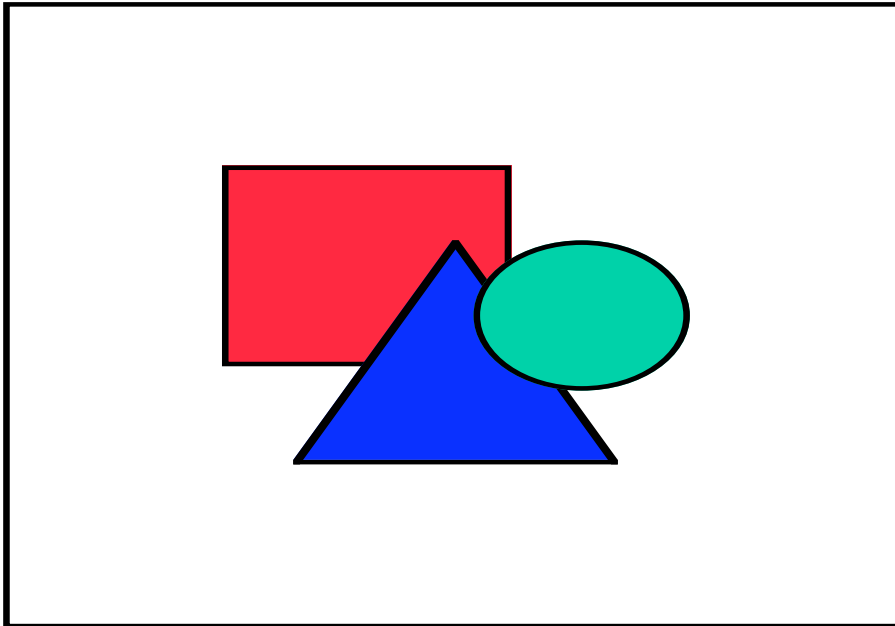
The first three steps of this process are frequently combined using the debugger. For example, using the Visual Studio debugger, you can both run your application and step through its execution

in code incrementally. Doing this makes it possible for you to see the order that code statements are called during execution, how variable values change, how input is processed, and what output is created. When the application functions incorrectly or the wrong output is generated, you can use the debugger to see the exact line of code that is executed when this problem occurs.

Breakpoints

A breakpoint tells the Visual Studio debugger when or where a program should pause during execution. You can set breakpoints on a specific line or statement in code, or you can set them to occur when a certain condition is met in the code. Use breakpoints to evaluate your program logic. For example, you can use breakpoints to determine the order in which program statements are executed or to view the value of a variable at a certain point in program execution. As an example, you could set a breakpoint to pause the program when a variable's value equals 10.

Breakpoints are set in the left, gray margin in the Code Editor. For more information, see [Inserting a New Breakpoint from the Debug Menu](#) in the MSDN library. Several examples of breakpoints are shown in the following screen shot.



When a breakpoint is reached, your program and the debugger go into break mode, where program execution is halted until you press F5, choose Continue from the Debug menu, or step through the program.

Step Features

In addition to breakpoints, other debugging features are available that make it possible for you to step through your code in the following ways:

Step Into (F11)

Makes it possible for you to execute code one unit, that is, one line, statement, or machine instruction at a time. From the Debug menu, choose the Step By command to select one of these options.

Step Over (F10)

Instructs the debugger to execute the next unit of code (for example, statement or function). If the next unit of code contains a function call, Step Over executes the entire function and then halts at the first unit outside the function. By contrast, Step Into only executes the function call and then halts at the first unit of code inside the function. Use Step Into if you want to step inside the function call. Use Step Over if you want to avoid stepping into functions.

Step Out (SHIFT+F11)

Resumes execution of your code until a function returns and then breaks at the return point in the function. Use Step Out when you are inside a function call and you want to return to the calling function.

Note You can access the Step commands only if your application is in break mode or before you start the application.

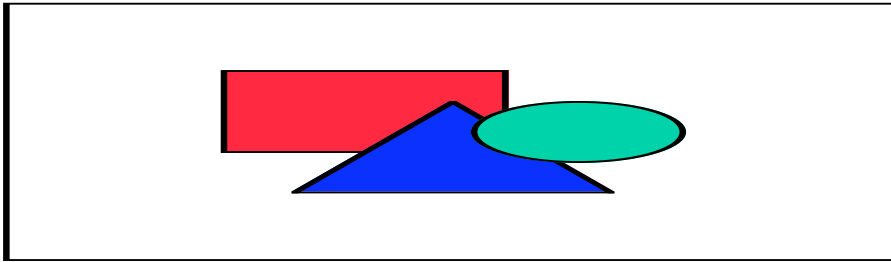
Run to Cursor (CTRL+F10)

Runs your application until it reaches the location where the cursor is set in the code. For more information, see [Running to the Cursor Location in the MSDN Library](#).

The Step Over, Step Out, and Run to Cursor commands will stop at a breakpoint if the code execution reaches one prior to completing the requested command.

Variable Values

While you are debugging your application, you can use the Autos and Locals windows to watch how the values of variables change as the program progresses. The Autos window displays variables used in the current and previous statements. The current statement is the next statement that will be executed if program execution continues. The variable name, value, and type are displayed in the Autos window, as shown in the following screen shot.

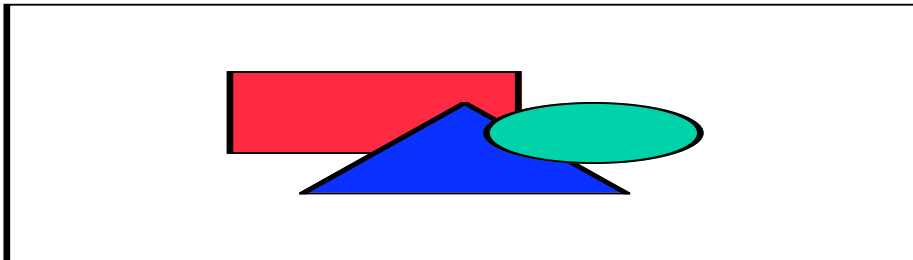


The value of the variable that changed in the last step executed in the application code is shown in red. Using the Autos window, you can edit the value of a variable while you are debugging your application. When you edit a variable value, the variable will also be shown in red. For more information, see [Using the Autos Window](#) in the MSDN Library. The name of the Autos window is taken from the fact that this window automatically identifies the values of these variables.

The Locals window displays variables that are local to the current context. The default context is the function containing the current execution location in the debugger. However, you can choose to display an alternative context in the Locals window. For example, you could choose to display the context for a specific function.

Call Stack

The Call Stack window is used to view the function or procedure calls that are currently on the stack. The stack is a region of memory reserved by programs for storing data, such as procedure and function call addresses. The Call Stack window displays the name of each function and the programming language in which it is written, as shown in the following screen shot.



In the call stack, KERNEL32 refers to the DLL that contains the entry point for your program; this is what actually launches your program. Additional function and procedure information can also be displayed in the Call Stack window. For more information, see *Using the Call Stack Window* in the MSDN Library.

Walkthrough: Debugging a Sample Application

This section guides you through debugging a simple C application for converting from Fahrenheit to Celsius. For temperatures of 0, 15, 30, 45, 60, 75, and 90 degrees Fahrenheit, this program displays the corresponding temperature in degrees Celsius. However, this application has an error in it that results in the display of temperatures far beyond this value. Use the following procedure to find the bug in this application.

To find the bug

1. Load the FahrenheitToCelsius.cpp file in Solution Explorer. For more information, see *Debugging Sample*.
2. View line numbers in FahrenheitToCelsius.cpp. For more information, see *Viewing Line Numbers*.
3. Set a breakpoint at Line 18 by clicking once in the gray margin to the left of this line.

Lines 17 through 18 contain a for loop that increases the value of the integer variable *i* in increments of 15, beginning with a value of zero (0). At each new increment, the value of *i* is converted to the equivalent Celsius temperature and printed to the screen. The program will execute the loop until *i* equals a value of "100".

4. From the **Debug** menu, choose **Start**.

-or-

Press **F5**.

5. Press **F5** again, while observing the value of *i* in the **Autos** window.

As you press **F5**, you will see that the value of *i* changes from "0" to "15" to "30", and so on. Eventually the value of *i* reaches **105**. Because *i* never equals a value of "100" the program never exits the loop.

6. Press **SHIFT+F5** to stop debugging.

Now that you have located the bug in the application, fix it.

To fix the bug

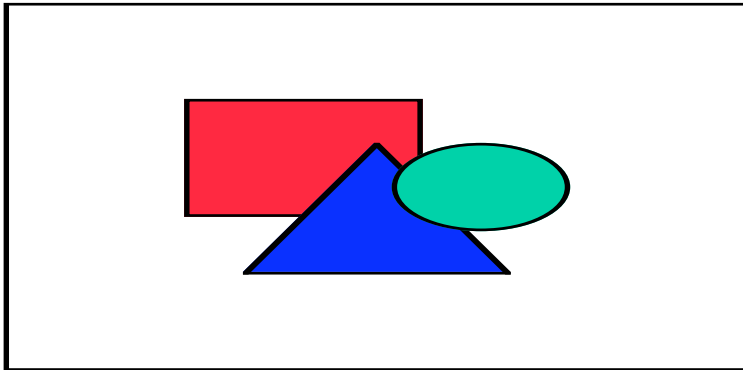
1. On line 17, change the code to read as follows:

```
for (int i = 0; i <= 100; i += 15) {
```

This involves replacing `!=` (not equal to) with `<=` (less than or equal to), so the program will exit when *i* is greater than 100. Before and during the time that *i* equals 100, the program continues to execute. After *i* is greater than 100, the program exits.

2. From the **File** menu, choose **Save**.
3. From the **Build** menu, choose **Build**.
4. From the **Debug** menu, choose **Start Without Debugging**.

The program now runs, displaying a list of Fahrenheit and corresponding Celsius temperatures, up to a maximum temperature of 90 degrees F (30 degrees C), as shown in the following screen shot.



Printing the Output Window

The Output window displays status messages at run time, such as build errors and build status. You can also use the Output window to display the results of external tools, such as .bat and .com files, that usually display output in the command window. A batch file is an ASCII text file that contains a sequence of commands for the operating system.

Use the following procedure to display the Output window, if it does not appear with the Task List currently.

To display the Output window

- From the **View** menu, choose **Other Windows**, and then choose **Output**.
- or-
- Press **CTRL+ALT+O**.

Use the following procedure to print the contents of the Output window.

To print the Output window

1. Place the cursor in the **Output** window to give this window focus.
2. From the **File** menu, choose **Print**.
The **Print** dialog box appears.
3. Select the appropriate printer, print range, and number of copies, and then click **OK**.

Viewing Line Numbers

Use the line numbers feature when viewing code to help find the location of build errors shown in the task list.

To view line numbers in code

1. From the **Tools** menu, select **Options**.
2. Under **Options**, click **Text Editor**.
3. Under **Text Editor**, click **All Languages**.
4. Under **Display**, select **Line Numbers**.

BLANK PAGE

IMPORTANT: This text will appear on screen, but will not print on a PostScript printer.

This page should be the last one in this file; it was inserted by running the InsertBlankPage macro.

Do not type any additional text on this page!

Reference

This section documents reference information for Microsoft® Visual Studio® .NET Academic Student Tools.

In This Section

Visual Studio .NET Quick Reference

Contains a quick introduction to important features and commands needed to effectively write code in Microsoft® Visual Studio® .NET.

Visual Studio .NET File Extensions

Provides a list of file extensions used in Visual Studio .NET.

Compiler Warning Levels

Provides a list of warning levels that can be applied in the compiler with a description of each level's severity.

Console Application Overview

Provides a general discussion of console applications.

Advanced Placement Classes

Lists the classes used in the Advanced Placement (AP) test.

Note This content is applicable only to high school students in the United States.

User Interface Reference

Contains topics that explain the dialog boxes and windows in Visual Studio .NET Academic.

Errors for Visual Studio .NET Academic Student Tools

Contains topics that explain the error messages used in Visual Studio .NET Academic.

Related Sections

Reference

Provides reference information for the Visual Studio .NET commands, command line arguments, user interface, and error messages.

Visual Studio .NET Quick Reference

This topic provides a quick reference for some of the things that you must know to write code in Visual Studio .NET. For more information, see *Developing with Visual Studio .NET and Editor Convenience Commands and Features* in the Microsoft® MSDN® Library.

Note All topics referenced in the third column are in the MSDN Library.

Feature (key command)	Description	For more information
Clipboard Cycling (CTRL+SHIFT+V)	Cycles through the last 20 copied or cut items.	Using the Clipboard Ring
Incremental Search (CTRL+I or CTRL+SHIFT+I)	Searches for a string as characters are typed.	Navigating Code and Text
Line Numbers	Displays line numbers in code.	Managing the Code Editor and View
Go To line number (CTRL+G)	Jumps to the specified line of code.	Navigating Code and Text
Outlining Code	Expands and collapses sections of code.	Outlining and Hiding Code
Full Screen Mode (SHIFT+ALT+ENTER)	Increases the viewing area for editing code.	Managing the Code Editor and View
Store Text on the Toolbox	Stores code in the Toolbox for use in other applications.	Toolbox and Managing Tabs and Items in the Toolbox
Insert File as Text	Inserts text from a file at the current position in the code.	Managing File Storage
Virtual Space	Continues code lines off to the side of the visible screen area.	Managing the Code Editor and View
Create Custom Comments Tokens	Makes it possible for you to create custom comment tokens for the task list, in addition to the default tokens: TODO, HACK, and UNDONE.	Creating Custom Comment Tokens

Feature (key command)	Description	For more information
Indent Selected Code CTRL+K, CTRL+F	Makes it possible for you to determine whether your statements are nested correctly.	Managing Code Formatting
Automatic Brace Matching CTRL+]]	Makes it possible for you to determine whether your statements are nested correctly.	Automatic Brace Matching

Visual Studio .NET File Extensions

The following is a list of general file extensions for Visual Studio .NET including Microsoft Visual C++ .NET, Microsoft Visual Basic .NET, and Microsoft Visual C# .NET.

Extension	Description	Language
*.aspx	Used to write code that handles such global ASP.NET application-level events as Session_OnStart and Application_OnStart. The file has a name of Global.aspx, which you cannot change.	ASP
*.asmx	References a class that exposes methods by means of HTTP requests. It provides a graphical design surface.	ASP
*.asp	Active server page file that is used to define both the HTML formatting and the server-side script.	ASP
*.aspx	Form used in a Web application, analogous to a Windows form in a local application. It provides a visual designer, that is, a graphical design surface.	ASP
*.bmp	Bitmap file.	
*.c	C source code file.	Visual C++
*.cpp	Visual C++ source code file.	Visual C++
*.cs	C# form or control.	C#
*.csproj	C# project file.	C#
*.css	Cascading style sheet file, which is used with HTML Web pages to define styles.	HTML
*.dbp	Database project file.	
*.def	If the project is for a dynamic-link library (DLL), this is the	Visual C++

Extension	Description	Language
	module definition file. For a control, this file provides the name and description of the control, and the size of the run-time heap.	
*.disco	Sometimes called a disco file. Provides a means to enumerate all Web Services and all schemas in a Web project.	
*.dll	Dynamic-link library file.	Visual C++
*.exe	Executable file.	Visual C++
*.h	Header file.	Visual C++
*.htm	Hypertext Markup Language (HTML) file, or a typical Web page file.	HTML
*.ico	Icon file.	
*.js	Microsoft® JScript® .NET code file. For more information, see What is JScript .NET? in the MSDN Library.	JScript
*.lic	User license file.	Visual C++
*.ncb	No compile browser file. Contains information generated by the parser, which is used by Class View to display class information. If this file is deleted, it is regenerated automatically.	Visual C++
*.odl	Object description language file. This file contains the Object Description Language source code for a control type library and is used by Visual C++ to generate a type library. The library generated exposes the control's interface to other Automation clients.	Visual C++
*.rc	Resource script file. For more information, see Resource Files in the MSDN Library.	Visual C++
*.rc2	Resource file for more than one project.	Visual C++
*.rct	Resource template file. For more information, see Using Resource Templates in the MSDN Library.	Visual C++
*.reg	Registration file.	Visual C++
*.res	Resource file.	Visual C++
*.rpt	Crystal Report file, a third-party reporting tool for databases. When you open the file, it opens in the Crystal Report Designer.	Visual Basic
*.sln	Visual Studio Solution file. Organizes all of the elements of a project or multiple projects into a single solution.	
*.suo	Visual Studio Solution options file. Stores all user options for a solution.	

Extension	Description	Language
*.txt	Text file.	
*.vb	Visual Basic form or control.	Visual Basic
*.vbproj	Visual Basic project file.	Visual Basic
*.vbs	Visual Basic Scripting Edition (VBScript) code file. For more information, see What is VBScript? in the MSDN Library.	VBScript
*.vcproj	Visual C++ project file. Stores information specific to the project. A separate *.vcproj file is created for each Visual C++ project in a solution. *.vcproj files are not compatible with NMAKE.	Visual C++
*.vdp	Visual Studio Setup and Deployment file.	
*.vfpproj	Microsoft Visual FoxPro project file.	Visual FoxPro
*.web	ASP.NET uses this file to configure the Web settings for a Web project. The file has a name of Config.web, which you cannot change.	ASP
*.wsf	Windows Script Host code file. For more information, see Microsoft Windows Scripting Host: A Universal Scripting Host for Scripting Languages in the MSDN Library	
*.xml	Extensible Markup Language (XML) document file. For more information, see XML Technology Background in the MSDN Library.	XML
*.xsd	XSD schema file. For more information, see XML Schema Reference (XSD) in the MSDN Library.	XML
*.xslt	Used to transform XML and XSD documents. For more information, see Great Ways to Use XSLT in the MSDN Library.	XML

Compiler Warning Levels

The following table shows the list of warning levels that can be applied and provides a description of each level's severity.

Warning Level	Description
0	Disables all warnings.
1	Displays severe warnings.
2	Displays all level-1 warnings and warnings less severe than level 1. Level 2 is the default warning level at the command line.
3	Displays all level-2 warnings and all other warnings recommended for production purposes.
4	Displays all level-3 warnings plus informational warnings. Some of these warnings can be ignored safely, and some represent genuine bugs in the application. This option should be used to search for probable bugs in code and to detect code features that are likely to be errors or wasteful.

Console Application Overview

A console application generally has no graphical user interface (GUI), compiles into an executable (.exe) file, and can be run as a stand-alone application from the command line. You can use console applications to learn the C and C++ programming languages, and because console applications typically have no user interface, you will avoid the added complexity of concurrently learning Windows GUI programming.

Visual Studio .NET Academic Student Tools provides two types of console applications with which you can work. In addition to the standard console application, a console application designed specifically for beginning student developers is included. For more information, see [Creating a Console Application in Getting Started](#).

The standard and student console applications differ in that the student console application does not include precompiled header (.pch) files, which increase the complexity of simple applications. Precompiled headers are used to support faster compilation times by restricting compilation to only those files that require it. They are, therefore, used in large, complex applications to achieve faster build times. For more information, see [Creating Precompiled Header Files in the MSDN library](#).

The following files make up the student console application:

.vcproj file

Visual C++ project file. Stores the information specific to your project. Each project has a separate .vcproj file.

.cpp file

Visual C++ source code file.

Readme.txt

Readme file that you modify to be the informational file for the project. This file provides a description of the files included in your project. Use this file to add general information regarding individual files or your application as a whole.

Advanced Placement Classes

The following table shows the list of AP classes and a brief description of each.

Note This content is applicable only to high school students in the United States.

Class	Files	Description
apmatrix	apmatrix.h and apmatrix.cpp	Resizable, two-dimensional array with safe subscripting.
apqueue	apqueue.h and apqueue.cpp	Queue data structure. Items placed in the queue can only be accessed from the front of the queue. A queue is an example of a first-in, first-out data structure.
apstack	apstack.h and apstack.cpp	Stack object. A stack is like a stack of plates. As more plates are added to the stack, they go on top of the stack. The last plate added is the first removed. A stack is an example of a last-in, first-out data structure.
apstring	apstring.h and apstring.cpp	String object. Used to store a series of characters (that is, a string) in place of the standard char (character) data type. This class maintains the current string length, thus eliminating the need for the terminating null character.
apvector	apvector.h and apvector.cpp	Vector object. Similar to a two-dimensional array but dynamically resizable.

Note For the latest version of the AP C++ classes, please visit the AP Web site (<http://www.collegeboard.com/ap/computer-science/html/classes.html>).

User Interface Reference

User interface (UI) reference topics explain the options that appear on various dialog boxes, windows, and other user interface elements. These topics generally appear when you press F1 in a dialog box or wizard.

In This Section

Student Resources, Visual Studio Start Page

Used to display current headlines and news articles from the MSDN Library that are provided specifically for students.

Student Course Tools, Visual Studio Start Page

Used to add or delete a course, access an installed course, download a starter project for an assignment, or view course resources.

Course, Assignment Manager

Used to view course information, download a new assignment, submit a completed assignment, and change user information.

Messages, Assignment Manager

Used to read course messages.

Password, Assignment Manager

Used to change your Assignment Manager password.

Related Sections

Reference

Documents reference information for Visual Studio .NET Academic Student Tools.

Student Resources, Visual Studio Start Page

Use the **Student Resources** page to display current headlines and news articles from the MSDN Library that are provided specifically for students. To access this page, click **Student Resources** from the Visual Studio .NET start page.

To update the Student Resources page

- Select the **Student Resources** page.
The current list of resource links is shown on the page.

Student Course Tools, Visual Studio Start Page

Use the **Student Course Tools** page to add or delete a course, to access an installed course to download a starter project for an assignment, or to view course resources. To access this page, click **Student Course Tools** from the Visual Studio .NET start page.

Add Course

Click to add a course. For more information, see [Add Course Page](#).

Delete Course

Click to delete a course. For more information, see [Delete Course Page](#).

The list of currently installed courses is also shown on this page. To access the **Work With Courses** page for a specific course, click the course name. For more information, see [Work With Courses Page](#).

In This Section

Add Course Page

Used to install course information on your local machine.

Work With Courses Page

Used to download starter projects for assignments and to view the course description and course resources.

Delete Course Page

Used to delete a course.

Related Section

User Interface Reference

Provides UI reference topics for the Student Tools.

Add Course Page

Use the **Add Course** page to install course information on your local machine. To access this page, click the **Add Course** link from the **Student Course Tools** page.

In the text box, type the URL location provided by your instructor for this course, then click **Add**.

Work With Courses Page

Use the **Work With Courses** page to download starter projects for assignments and to view the course description and course resources. To access this page, click the name of the course you want to work with from the **Student Course Tools** page.

Full Course Description

Click to display your instructor's description of the course with which you are currently working.

Course Resources

Click to display your instructor's list of resources for this course. Resources can include links to e-mail addresses, newsgroups, and Web sites.

Course Assignments

Click to display the current list of assignments provided by your instructor for this course. Use this link to download the starter project for a given assignment. For more information, see Download Starter Project Dialog Box.

In This Section**Download Starter Project Dialog Box**

Used to open the starter project in Solution Explorer in Visual Studio .NET.

Related Section**Student Course Tools, Visual Studio Start Page**

Used to add or delete a course, access an installed course, download a starter project for an assignment, or view course resources.

Download Starter Project Dialog Box

Use the **Download Starter Project** dialog box to open the starter project in Solution Explorer in Visual Studio .NET. To access this dialog box, click the name of the course for which you want to download a starter project on the **Student Course Tools** page. Next, click **Course Assignments**, and then click the **Download Starter Project** link for the desired assignment.

Project Name

The name of the assignment starter project to download is displayed in this text box.

Location

A default location for the starter project is displayed in this text box. Click Browse to find an alternate location for the starter project.

Project will be created at

Displays the location where the starter project will be downloaded.

Delete Course Page

Use the **Delete Course** dialog box to delete one or more courses from your local machine. To access this dialog box, click **Delete Course** from the **Student Course Tools** page.

Currently Installed Courses

Displays the list of courses that you have installed to your local machine. Choose which course(s) to delete by selecting the check box to the left of each course name, or, to delete all courses, click **Select All**.

Click **Delete Selected Courses** to delete the course(s) from your local machine.

Course, Assignment Manager

Access the **Course** page to view course information, download a new assignment, submit a completed assignment, and change user information.

In This Section

Course Info Page

Used to view the course description and resources for an Assignment Manager course.

Course Assignments Page

Used to download a starter project, submit an assignment, view the status of auto build and auto check, and check your grade.

Upload Submission Page

Used to download the starter project for an Assignment Manager assignment.

Download Starter Project Page

Used to submit the project for your completed assignment to your instructor.

Related Section

Working with Courses and Assignments

Discusses how to download an assignment starter project, submit an assignment, check assignment status, view course information, read messages, and change your user password.

Course Info Page

Use the **Course Info** page to view the course description and resources for an Assignment Manager course. To access this page, click **Add Course** on the **Student Course Tools** page. Enter the course URL, and click **Add**. Enter your user name and password, and click **Log On** to log onto Assignment Manager. Click the **Course Info** page.

Course Name

Displays the course name. This field cannot be modified.

Description

Displays the course description. This field cannot be modified.

Course Resources

Provides a list of links to the resources for this course. Click the link of the resource you want to access.

Course Assignments Page

Use the **Course Assignments** page to download a starter project, submit an assignment, view the status of auto build and auto check, and check your grade. To access this page, click **Add Course** on the **Student Course Tools** page. Enter the course URL, and click **Add**. Enter your user name and password, and click **Log On** to log onto Assignment Manager. The **Course Assignments** page appears.

Assignment

Displays the assignment name. This field cannot be modified.

Due Date

Displays the assignment due date. This field cannot be modified.

Get Starter

Click to download the starter project for the assignment, if one is provided. For more information, see Download Starter Project Page.

Submit Project

Click to submit a completed project to your instructor. For more information, see Upload Submission Page.

Date Last Submitted

Indicates the last date that you submitted this assignment. If you have never submitted this assignment, then this field is blank. Your instructor can make it possible for you to submit an assignment multiple times. In this case, each new submission replaces the previous submission.

Auto Build

After you submit your assignment, the **Auto Build** field will display an icon indicating whether the action is completed, pending, or has failed if your instructor has enabled the Auto

Build feature on the Assignment Manager server. Otherwise, a solid line is displayed in these fields. A key to the icons used is provided at the bottom of the **Course Assignments** page.

Auto Check

After you submit your assignment, the **Auto Check** field will display an icon indicating whether the action is completed, pending, or has failed if your instructor has enabled the Auto Check feature on the Assignment Manager server. Otherwise, a solid line is displayed in these fields. A key to the icons used is provided at the bottom of the **Course Assignments** page.

Grade

Display a value of **N/A** if your instructor has not assigned a grade to date or a grade value if your instructor has entered a grade for your assignment.

Download Starter Project Page

Use the **Download Starter Project** page to download the starter project for an Assignment Manager assignment. To access this page, click **Get Starter** from the **Course Assignments** page.

Location to place downloaded files

Provides a default path for downloading the starter project. If this path is acceptable, click **Download**, or browse to find or type the location of where you want to download the starter project to in this text box, and click **Download**.

Upload Submission Page

Use the **Upload Submission** page to submit the project for your completed assignment to your instructor. To access this page, click **Submit Project** from the **Course Assignments** page.

Assignment Name

Displays the name of this assignment for which the project is being uploaded. This field cannot be modified.

Select Project to Upload

The list of projects open in Solution Explorer is displayed in this list box. Select the project to submit as your completed assignment from this list box.

Extra Files

Browse to find any additional files to include with your project submission, and click **Add Files**.

Upload Project

Click to submit your completed assignment to your instructor.

Messages, Assignment Manager

Access the Messages page to read course messages.

In This Section

Read Page

Used to read messages sent from the internal messaging feature in Assignment Manager.

Related Section

Reading Messages

Discusses how to read internal messages sent from your instructor through Assignment Manager.

Read Page

Use the **Read** page to read messages sent from the internal messaging feature in Assignment Manager. To access this page, click **Messages** in the left pane of Assignment Manager. Click the **Read** page.

From

Indicates the sender of the message.

Subject

Indicates the message subject. Click the subject link to read the message.

Date Sent

Indicates the date that the message was sent.

Date Read

Indicates the date that the message was read.

Password, Assignment Manager

Access the **Password** page to change your Assignment Manager password.

In This Section

Password Page

Used to change your password.

Related Section

Changing Your User Password

Discusses how to change your Assignment Manager user password.

Change Password Page

Use the **Change Password** page to change your password. To access this page, click **Password** in the left pane of Assignment Manager.

User

Displays the current user for whom the password is being changed.

Current Password

Enter your current password.

Enter a New Password

Enter your new password.

Confirm New Password

Re-enter your new password.

Errors for Visual Studio .NET Academic Student Tools

This section contains information about the following errors:

- Course assignment file is not valid. Check with instructor.
- Course file <file name> already exists. Delete the existing course before adding the new course.
- Course information is not available. Check with instructor.
- Course file is not valid.
- Enter a course URL.
- Select course(s) to delete.
- Starter project directory already exists. Specify another directory.
- Student Add-in is not loaded.
- Unable to copy the assignment files from the server path <server path>.
- Unable to load course information from server or local copy.
- Unable to open the startup file <file name>. Check with instructor.
- Download location did not exist on local file system.
- Invalid credentials. Please try again.
- Root upload location not available.

- The CourseID is missing or invalid. Please use the Core Academic Tools when accessing this page.
- You are not registered for this course.
- You have exceeded the maximum upload size for this project of <xx> MB. No more files may be uploaded.

Course assignment file is not valid. Check with instructor.

The assignment file is formed incorrectly.

To correct this error

- Contact your instructor or teaching assistant. The course description or the course <name> tag might be incorrect in the Assignments.xml file.

Course file <file name> already exists. Delete the existing course before adding the new course.

A course of the same name already exists locally on your hard disk.

To correct this error

- Delete the existing course from your hard disk before adding the new course. For more information, see Deleting a Course.

Course information is not available. Check with instructor.

The course file is formed incorrectly, that is, there is an incorrect entry in the Courses.xml file.

To correct this error

- Contact your instructor or teaching assistant.

Course file is not valid.

The local copy of the course file is formed incorrectly.

To correct this error

1. Delete the existing course file (Courses.xml) from your hard disk.
2. Add the course file from the course server. For more information, see Adding a Course.

Enter a course URL.

You must have a Uniform Resource Locator (URL) to add a course.

To correct this error

- Enter a course URL in the dialog box. The course URL is provided by the instructor or the teaching assistant for this course.

Select course(s) to delete.

You must select a course to be able to delete a course.

To correct this error

- Select a course to delete.

Starter project directory already exists. Specify another directory.

This error occurs when you attempt to download a starter project to a directory that already contains another project of the same name.

To correct this error

- Delete the existing project in the directory.
- or-
- Download the starter project to a different directory.

Student Add-in is not loaded.

The student add-in that contains the assignment tools did not load with Visual Studio .NET.

To correct this error

1. Open the Visual Studio .NET Academic.
2. On the **Tools** menu, click **Add-In Manager**.
The **Add-In Manager** dialog box appears.
3. In the **Add-In Manager** dialog box, select **Student Course Tools** under **Available Add-ins**, and select **Startup** on the same line to have the add-in load on startup.
4. Click **OK**.

-or-

- Reinstall Visual Studio .NET Student Academic if the add-in still does not load or if the entry for **Student Course Tools** does not appear in the **Add-In Manager** dialog box.

Unable to copy the assignment files from the server path <server path>.

This error occurs when you are downloading a starter project and either you do not have the correct file permissions on the server or the file does not exist at the specified location on the server.

To correct this error

- Contact your instructor or teaching assistant for this course to determine if the file location is correct, if the file is missing, or if your file permissions are incorrect.

Unable to load course information from server or local copy.

The course file at the server location is inaccessible, and the local copy of the course file is formed incorrectly.

To correct this error

1. Delete the local copy of the existing course file. For more information, see [Deleting a Course](#).
2. Add the course file from the course server.

Unable to open the startup file <file name>. Check with instructor.

The URL for the assignment startup page associated is corrupt.

To correct this error

- Contact your instructor or teaching assistant, and inform them that you are unable to connect to the startup page associated with the assignment.

Download location did not exist on local file system.

Occurs when the file cannot be downloaded to the specified location.

To correct this error

- Be sure you have permissions to write to the specified location.

Invalid credentials. Please try again.

Occurs when the user ID or password provided are not valid or do not match.

To correct this error

- Enter a valid user ID and password.

Root upload location not available.

Occurs when the root upload location is not available.

To correct this error

- Contact your instructor or teaching assistant, and inform them that you are unable to connect to the upload location for the assignment.

The CourseID is missing or invalid. Please use the Core Academic Tools when accessing this page.

User attempted to navigate directly to the Assignment Manager Web page from outside the IDE rather than from within the Student Tools.

To correct this error

- Navigate to Assignment Manager from within the Student Tools. For more information, see Adding a Course.

You are not registered for this course.

Occurs when a user is not associated with a course.

To correct this error

- Contact your instructor or teaching assistant, and inform them that you are unable to access this course.

You have exceeded the maximum upload size for this project of <xx> MB. No more files may be uploaded.

Occurs when the maximum upload size for a project is exceeded.

To correct this error

- Contact your instructor or teaching assistant, and inform them that you are unable to upload your completed project to the Assignment Manager server due to file size constraints.

BLANK PAGE

IMPORTANT: This text will appear on screen, but will not print on a PostScript printer.

This page should be the last one in this file; it was inserted by running the InsertBlankPage macro.

Do not type any additional text on this page!