**Lab 6: C Program Reversing**

**What You Need**

* A Windows machine, real or virtual. I used Windows Server 2008.

**Purpose**

You will write a small C program and compile it.

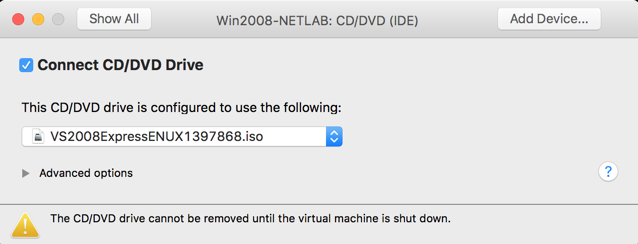
**Downloading and Installing Visual Studio Express 2008**

Open this page:

[Downloading the Virtual Machines](https://games.samsclass.info/secret/download-vms.htm)

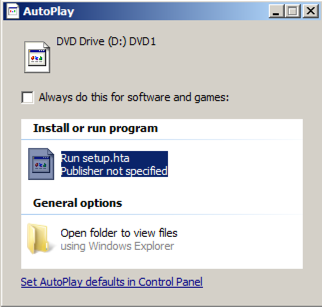
Enter the credentials your instructor gave you in class. In the "CNIT 126" section, download **VS2008ExpressENUX1397868.iso**

In VM settings, point the virtual CD drive to the **VS2008ExpressENUX1397868.iso** file, as shown below.



A box pops up, as shown below.

Click "**Run setup.hta**".



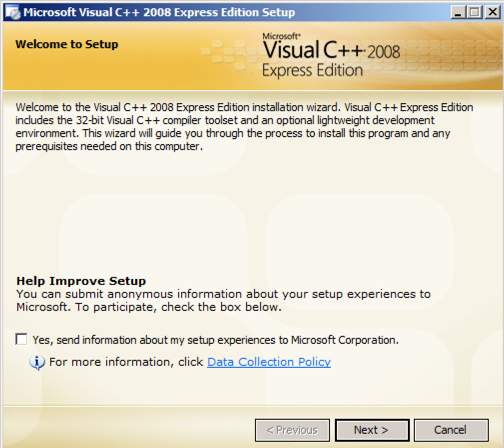
A Visual Studio Express window opens, as shown below.

Click "**Visual C++ 2008 Express Edition**".



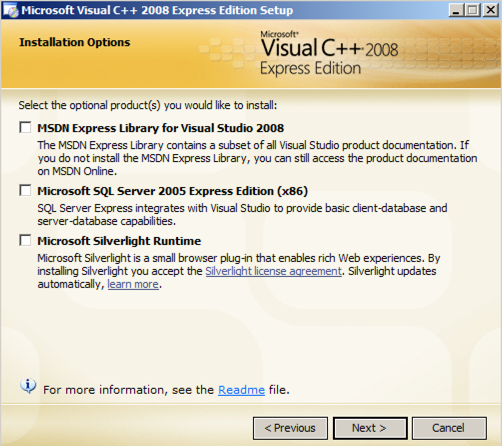
A "Welcome to Setup" box opens, as shown below.

Click **Next**.



Accept the license terms and click **Next**.

At the Installation Options box, clear all the checkboxes, as shown below, and click **Next**.



Click **Install**. When you see the "Setup complete" message, click **Exit**.

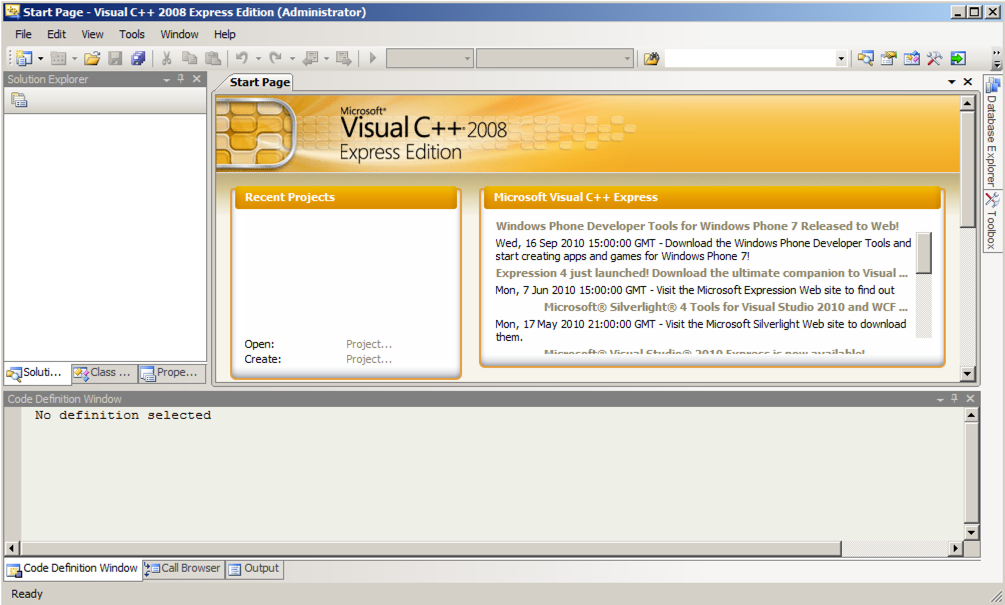
Close the "Visual Studion Express Edition Setup" box. Go into VM settings and disconnect the virtual CD.

**Launching Visual Studio Express 2008**

Click **Start**. Type **VISUAL**

In the search results, click "**Microsoft Visual C++ 2008 Express Edition**"

Visual C++ 2008 Express launches, as shown below:



**Creating a Simple C Program**

From the menu bar, click **File**, **New**, **Project...**.

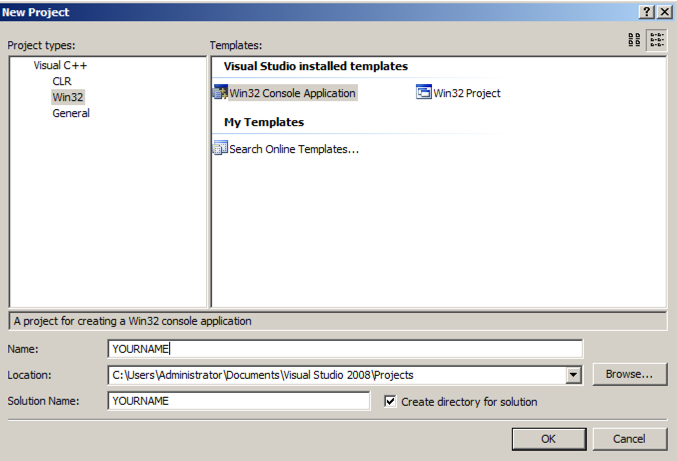
In the "New Project" window, on the left, click **Win32**, as shown below.

In the right pane, accept the default selection of "**Win32 Console Application**"

At the bottom of the "New Project" window, type a Name of **YOURNAME**, replacing "YOURNAME" with your own name. Do not use any spaces in the name.

In the "Location" line, notice the location files will be saved in--it's a subfolder of your Documents folder.

In the "New Project" window, click **OK**.



A box opens, titled "Welcome to the Win32 Application Wizard".

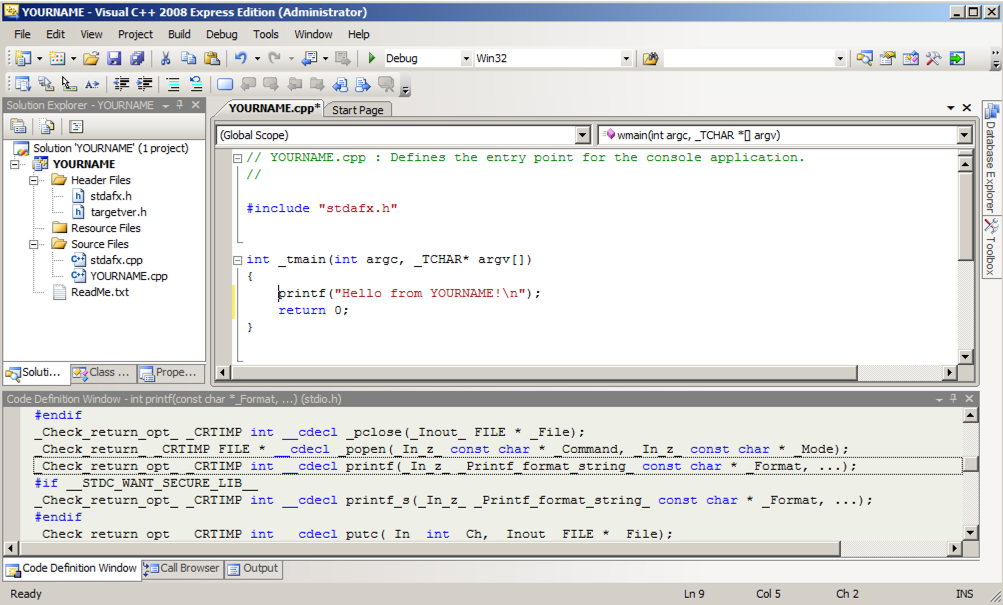
Click **Next**. In the next screen, accept the default settings and click **Finish**.

A window opens, showing a simple C program.

Add this line to the program, as shown below.

Do not use the literal string "YOURNAME"--replace it with your own name.

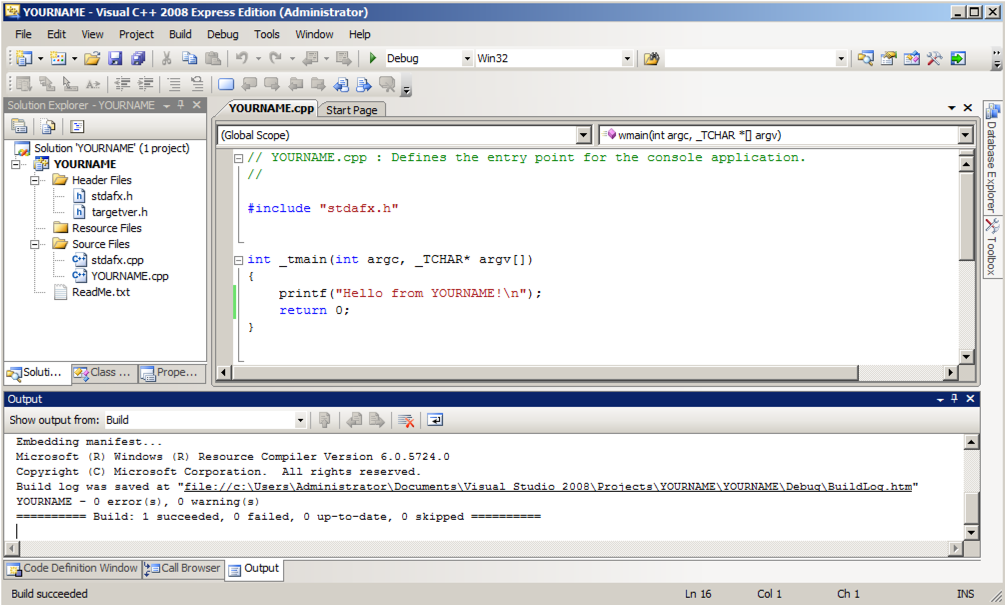
**printf("Hello from YOURNAME!\n");**



**Compiling your Program**

Click **BUILD**, "**Build Solution**".

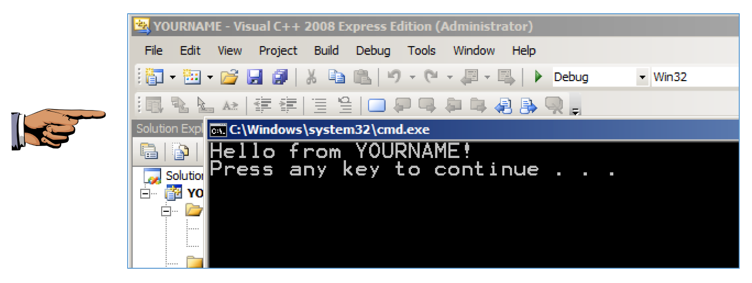
You should see the message "Build: 1 succeeded" at the bottom of the window, as shown below.



**Running your Program**

Click **DEBUG**, "**Start Without Debugging**".

A Command Prompt window opens, showing the message with your name in it, as shown below:



**Saving the Screen Image**

Make sure you can see the windows showing "**Hello from YOURNAME!**", as shown above.

On your keyboard, press the PrntScrn key.

Click **Start**, type in **PAINT**, and open Paint.

Press **Ctrl+V** to paste in the image of your desktop.

Save the image with a filename of "**Proj 7 from YOUR NAME**".

**Global and Local Variables**

From the menu bar, click **File**, **New**, **Project...**.

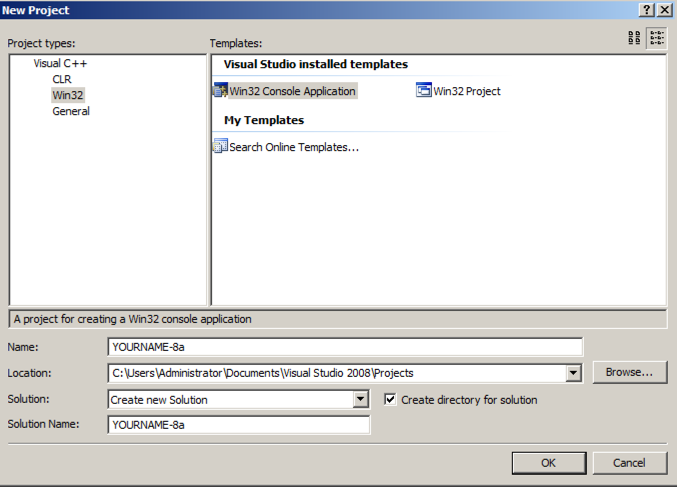
In the "New Project" window, on the left, click **Win32**, as shown below.

In the right pane, accept the default selection of "**Win32 Console Application**"

At the bottom of the "New Project" window, type a Name of **YOURNAME-8a**, replacing "YOURNAME" with your own name. Do not use any spaces in the name.

In the "Location" line, notice the location files will be saved in--it's a subfolder of your Documents folder.

In the "New Project" window, click **OK**.



A box opens, titled "Welcome to the Win32 Application Wizard".

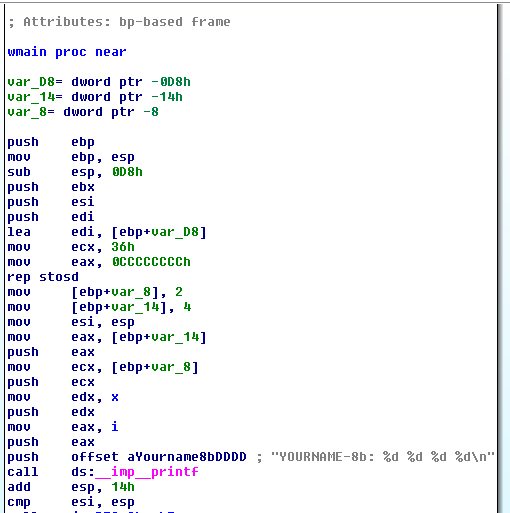
Click **Next**. In the next screen, accept the default settings and click **Finish**.

A window opens, showing a simple C program.

Modify this program to match the code shown in text and the image below.

Do not use the literal string "YOURNAME"--replace it with your own name.

**// YOURNAME-8a.cpp : Defines the entry point for the console application.  
//  
  
#include "stdafx.h"  
  
int i=1; // GLOBAL VARIABLE  
  
int \_tmain(int argc, \_TCHAR\* argv[])  
{  
int j=2; // LOCAL VARIABLE  
printf("YOURNAME-8a: %d %d\n", i, j);  
return 0;  
}**



**Compiling your Program**

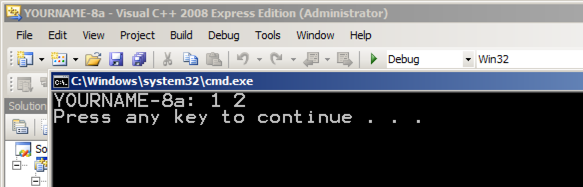
Click **BUILD**, "**Build Solution**".

You should see the message "Build: 1 succeeded" at the bottom of the window. If you see errors, you need to correct them and re-compile the program.

**Running your Program**

Click **DEBUG**, "**Start Without Debugging**".

A Command Prompt window opens, showing the output of "1 2", as shown below:



**Disassembling the EXE**

Click in the Command Prompt window, and press Enter to close it.

Minimize the Visual Studio Community Edition window.

Start IDA Pro Free.

In the "About" box, click **OK**.

Agree to the license.

Close the Help window.

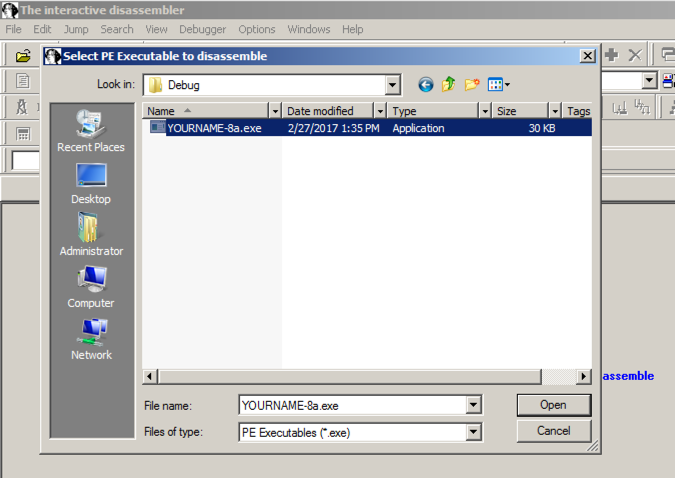
In the "Welcome to IDA!" box, click the **New** button.

In the "New disassembly database" box, double-click "**PE Executable**".

In the "Select PE Executable to disassemble" box, navigate to the folder you used to save your program. The default location is in the Administator home folder, with this relative path:

**Documents\Visual Studio 2008\Projects\YOURNAME-8a\Debug**

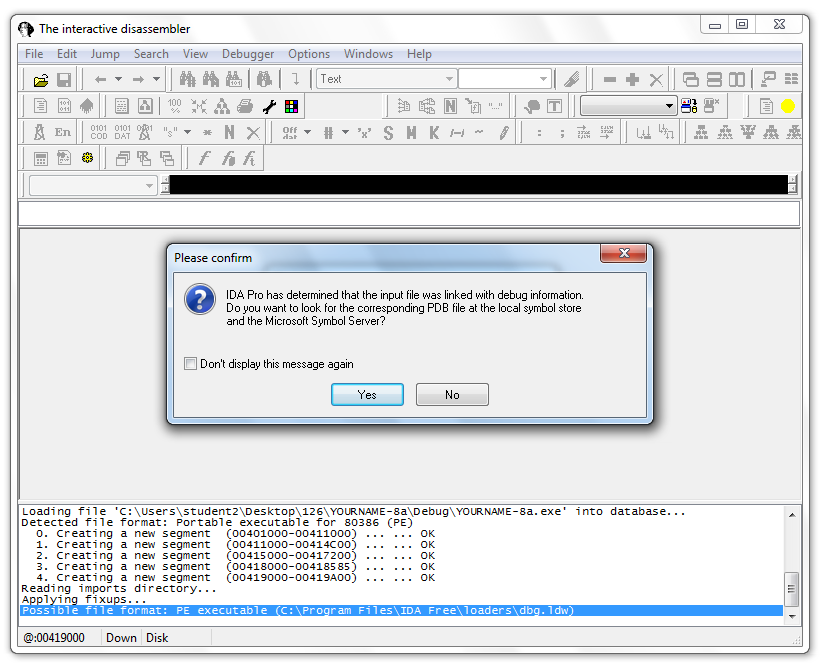
Double-click the **YOURNAME-8a.exe** file.



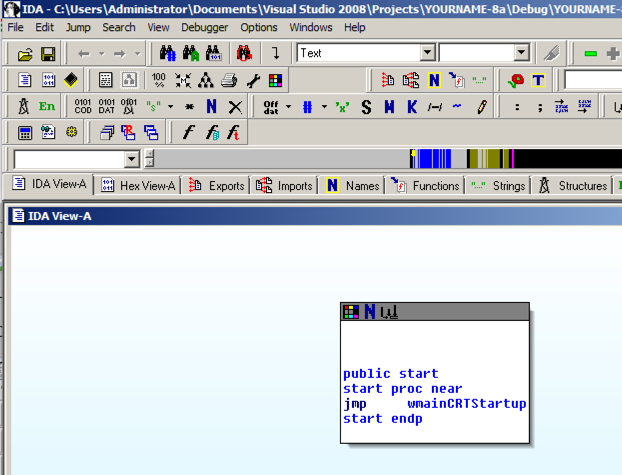
In the "PE Executable file loading Wizard", click **Next**, **Next**, **Finish**.

A box appears, saying this file was linked with debug information, as shown below. This is a luxury you won't often have with malware, but it's nice for this project.

Click **Yes**

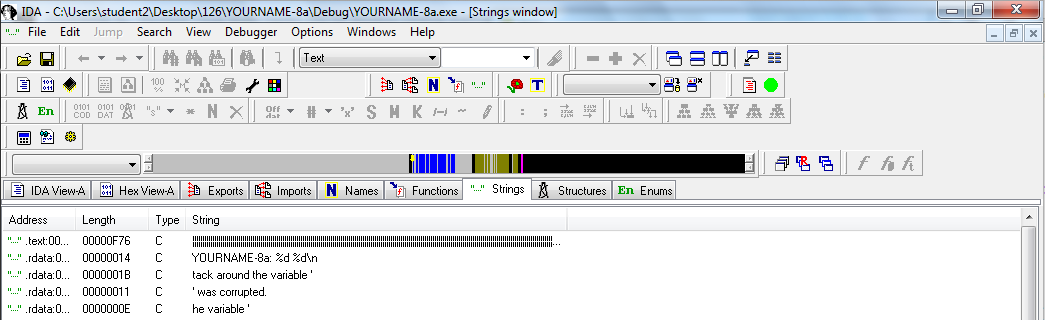


IDA Pro loads the file. Unfortunately, the graph mode isn't much use, as shown below.



However, we can still find the code. Expand the **Strings** window and find "**YOURNAME-8a %d %d\n**", as shown below.

Double-click "**YOURNAME-8a %d %D\n**".

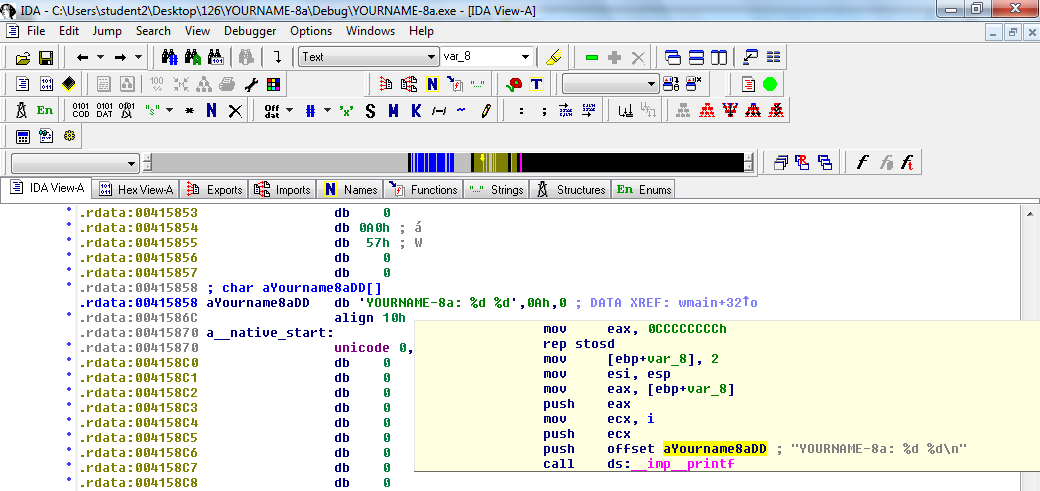


The location containing the string appears, as shown below.

This is in the .rdata section of the file, which contains data but not executable instructions.

To the right of "YOURNAME-8a" there is a "DATA XREF" comment. Hover over the address to the right of "DATA XREF", which was "**wmain+32**" when I did it.

The instructions that use this string appear in a yellow pop-up box, as shown below.



Double-click "**wmain+32**".

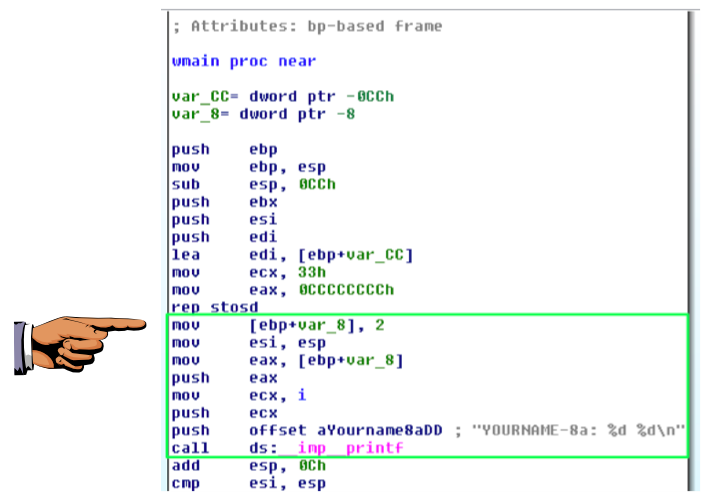
Now the assembly code that performs the task you wrote in C appears, as shown below.

Notice the region in the green box in the figure below.

These commands perform these C statements:

**int j=2;  
printf("YOURNAME-8a: %d %d\n", i, j);**

The **call** at the end jumps into the printf() function.



**Saving the Screen Image**

Make sure you can see the command showing

**push offset aYourname8aDD ; "YOURNAME-8a: %d %d\n**

as shown above. The offset value may be different, but it should contain **push** and **YOURNAME**.

On your keyboard, press the PrntScrn key.

Click **Start**, type in **PAINT**, and open Paint.

Press **Ctrl+V** to paste in the image of your desktop.

Save the image with a filename of "**Proj 8a from YOUR NAME**".

**Understanding Global and Local Variables**

Before the **call**, the three arguments are pushed onto the stack in reverse order: first j, then i, then the string "YOURNAME-8a: %d %d\n", as detailed below.

|  |
| --- |
| mov [ebp+var\_8], 2 **; PUT 2 into j**  mov esi, esp  mov eax, [ebp+var\_8] **; PUT j into eax**  push eax **; PUSH j onto the stack**  mov ecx, i **; PUT i into ecx**  push ecx **; PUSH i onto the stack**  push offset aYourname8aDD ; "YOURNAME-8a: %d %d\n" **; PUSH the address of the string onto the stack**  call ds:\_\_imp\_\_printf **; CALL printf()** |

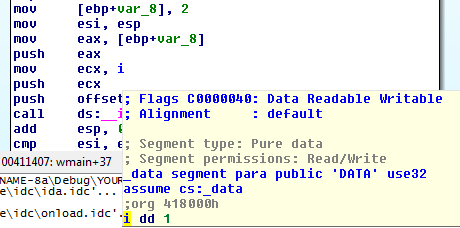
j is a local variable, so it is simply stored on the stack at the location **ebp+var\_8**. It's temporary, only available to the function it's defined in.

i is a global variable, and in this case IDA was able to refer to it by name in the "mov ecx, i" instruction.

To see where i is stored, hover the mouse over it.

A yellow box pops up showing where it is stored. When I did it, it was stored at location 418000, as shown below.

This variable will be available everywhere in the program, to any function.



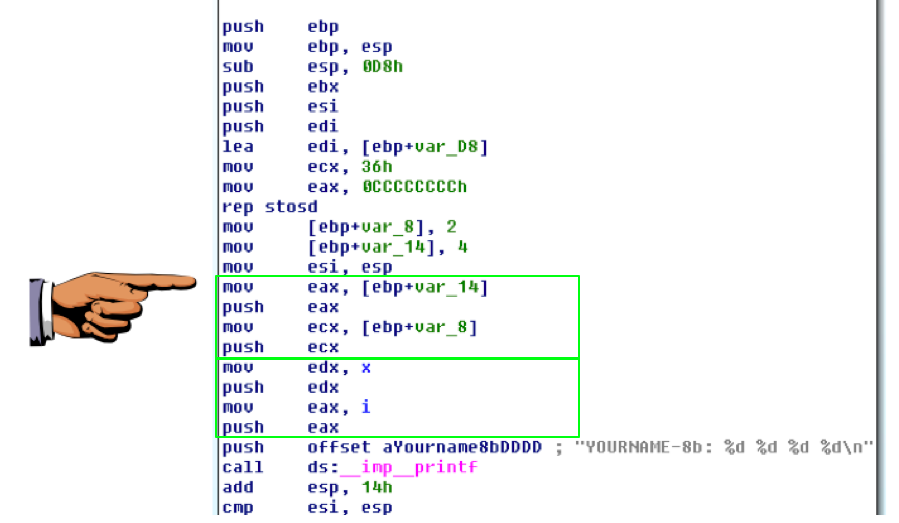
**CHALLENGE**

Modify the C program to contain a second global variable named x and a second local variable named y.

Compile it and disassemble it.

It must show these features, as shown below:

* **Two local variables** as shown in the top green box in the figure below: two **mov** instructions referencing stack locations such as **[ebp+var\_14]**, each followed by a **push** instruction.
* **Two global variables** as shown in the lower green box in the figure below: two **mov** instructions referencing named variables such as **x**, each followed by a **push** instruction.
* **YOUR NAME** in the string.
* A **call** operation to **printf**.



**Making a New C Program**

From the "VVisual C++ 2008 Express Edition" menu, click **FILE**, **New**, **Project...**.

In the "New Project" window, on the left, click **Win32**, as shown below.

In the right pane, accept the default selection of "**Win32 Console Application**"

At the bottom of the "New Project" window, type a Name of **YOURNAME-9a**, replacing "YOURNAME" with your own name. Do not use any spaces in the name.

In the "Location" line, click the **Browse** button and navigate to a folder you have permission to save files in, such as your desktop.

Click the "**Select folder**" button.

In the "New Project" window, click **OK**.

A box opens, titled "Welcome to the Win32 Application Wizard".

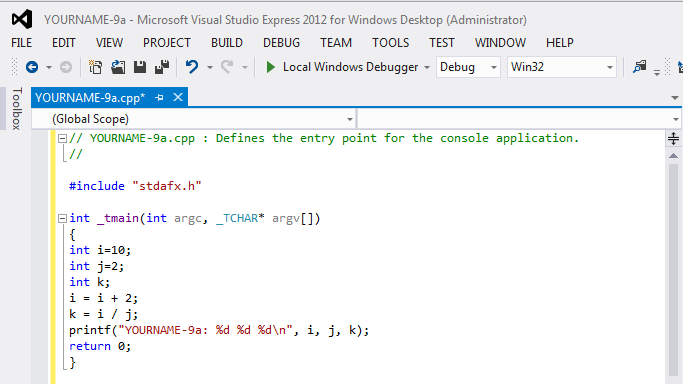
Click **Next**. In the next screen, accept the default settings and click **Finish**.

A window opens, showing a simple C program.

Modify this program to match the code shown in text and the image below.

Do not use the literal string "YOURNAME"--replace it with your own name.

**// YOURNAME-9a.cpp : Defines the entry point for the console application.  
//  
  
#include "stdafx.h"  
  
int \_tmain(int argc, \_TCHAR\* argv[])  
{  
int i=10;  
int j=2;  
int k;  
i = i + 2;  
k = i / j;  
printf("YOURNAME-9a: %d %d %d\n", i, j, k);  
return 0;  
}**



**Compiling your Program**

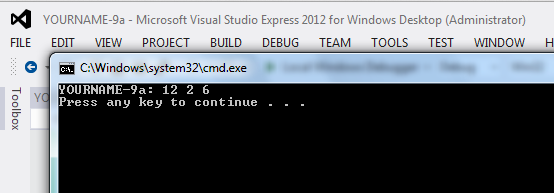
Click **BUILD**, "**Build Solution**".

You should see the message "Build: 1 succeeded" at the bottom of the window. If you see errors, you need to correct them and re-compile the program.

**Running your Program**

Click **DEBUG**, "**Start Without Debugging**".

A Command Prompt window opens, showing the output of "YOURNAME-9a: 12 2 6", as shown below:



**Disassembling the EXE**

Click in the Command Prompt window, and press Enter to close it.

Minimize the Visual Studio Express window.

Start IDA Pro Free.

In the "About" box, click **OK**.

Agree to the license.

Close the Help window.

In the "Welcome to IDA!" box, click the **New** button.

In the "New disassembly database" box, double-click "**PE Executable**".

In the "Select PE Executable to disassemble" box, navigate to the folder you used to save your program in Visual Studio Express, probably your desktop.

Double-click the "YOURNAME-9a" folder.

Double-click the **Debug** folder.

Double-click the **YOURNAME-9a.exe** file.

In the "PE Executable file loading Wizard", click **Next**, **Next**, **Finish**.

A box appears, saying this file was linked with debug information.

Click **Yes**

IDA Pro loads the file. As before, the graph mode doesn't show the interesting part of the program.

Expand the **Strings**. Double-click "**YOURNAME-9a %d %d %d\n**".

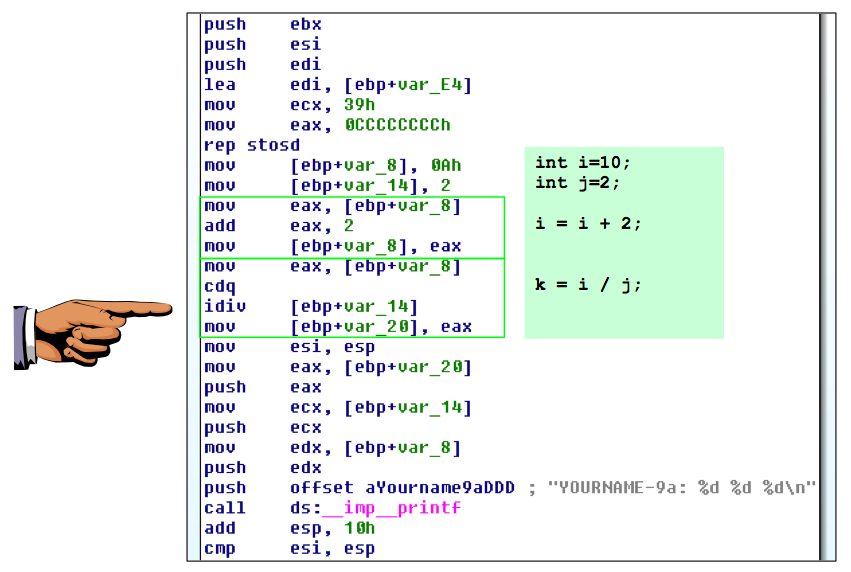
The location containing the string appears.

To the right of "YOURNAME-9a" there is a "DATA XREF" comment. To the right of the "XREF", double-click "**wmain**".

Now the assembly code that performs the task you wrote in C appears, as shown below.

Find the commands listed below, and see how they work. The explanations refer to the C code added to the figure below in the box with green shading.

|  |  |  |
| --- | --- | --- |
| **ASM Code** | **Explanation** | **C Code** |
|  |  |  |
| **mov [ebp+var\_8], 0Ah** | Put the number 10 into a local variable (i) | **int i=10;** |
| **mov [ebp+var\_14], 2** | Put the number 2 into a local variable (j) | **int j=2;** |
| **mov eax, [ebp+var\_8]** | Put i into eax |  |
| **add eax, 2** | Add 2 to eax | **i = i + 2;** |
| **mov [ebp+var\_8], eax** | Put eax (the result) into a local variable (i) |  |
| **mov eax, [ebp+var\_8]** | Put i into eax |  |
| **cdq** | Convert double to quad (required for division) | **k = i / j;** |
| **idiv [ebp+var\_14]** | Divide the value in eax by a local variable (j) |  |
| **mov [ebp+var\_20], eax** | Put eax (the result) into a local variable (k) |  |



**Saving the Screen Image**

Make sure you can see the commands listed above, and YOURNAME at the bottom.

On your keyboard, press the PrntScrn key.

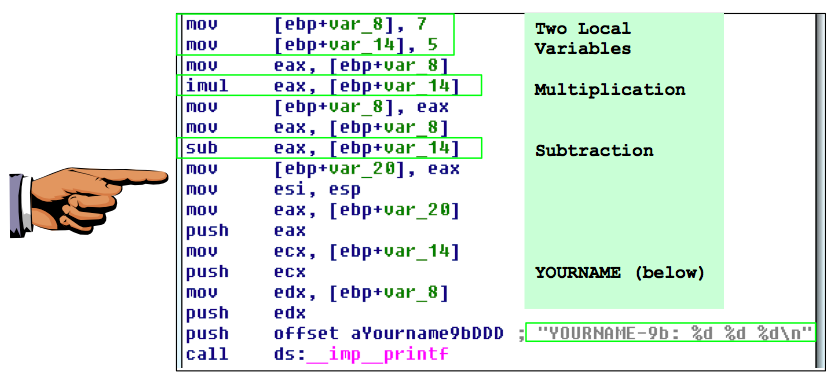
Click **Start**, type in **PAINT**, and open Paint.

Press **Ctrl+V** to paste in the image of your desktop.

Save the image with a filename of "**Proj 9a from YOUR NAME**".

**CHALLENGE**

Modify the C program to use multiplication and subtraction, compile it and disassemble it, producing the assembly code shown below.



It must show these features, as labelled in the image above:

* **Two local variables**: two **mov** instructions referencing stack locations such as **[ebp+var\_14]**, setting the variables to the values **7** and **5**.
* An **imul** operation
* A **sub** operation
* **YOUR NAME** in the string.