Programming in C: How to Get Things Done!

Jan Wigginton

Biostatistics 615/815

Last Lecture

- Anatomy of a C program
  - A collection of short functions
- Built-in data types available in C
- The C standard library

Executing C Code

- C is a high level language
  - Relatively easy to understand
- Computer CPUs execute much more detailed, "lower-level" instructions
- A compiler performs the necessary translation…

Working in a UNIX Environment

- GCC
  - Compile code
- GDB
  - Debug and test code
- GPROF
  - Collect profiling information

GCC

- GCC is a free C compiler
  - GNU C Compiler
  - G++ is a companion tool for C++
- Versions available for
  - Linux
  - Unix
  - Mac
  - Windows
- Developed by Free Software Foundation

Simple Usage

```
gcc my_program.c –o my_program -lm
```

- Uses text file with C/C++ code as input
- Produces executable program as output

Compiling with multiple source files

- All ".cpp" files in current directory

```
gcc *.cpp –o my_program –lm
```

- All ".cpp" files in "my_directory"

```
gcc –Imy_directory my_program.cpp
my_directory/*.cpp –o my_program –lm
```

A shortcut

- On Unix, it can be a time saver to create a "build.csh" script

```
#!/bin/tcsh
gcc *.c –o my_program –lm
```

- Make it executable

```
chmod +x build.csh
```

- To compile

```
./build.csh
```

Programming errors

- Compile time
  - Program does not compile.
  - Compiler reports a “best guess” of the problem
- Run time
  - Executable crashes or has unexpected behavior
  - May not appear for all conditions or all data sets

Compiler output

- Errors or warnings
- Executable isn’t created until all errors are corrected
- Program will still compile if warnings are not dealt with
- Good practice to correct compiler warnings as well

Common compiler errors

- Undeclared variables or functions
- Missing semicolon or brace
- Typos
- Missing files or libraries
- Type ambiguities

Common compiler errors

Undeclared variable

```
#include <stdio.h>

int main(int argc, char * argv[])
   {
   printf("I am a computer program\n");

   printf(“Value for variable %d", var);

   return 0;
   }
```

Compiler message

```
./build.csh
basic.cpp: In function `int
   main(int, char**)':
basic.cpp:14: error: `var'
   undeclared (first use this
   function)
basic.cpp:14: error: (Each
   undeclared identifier is
   reported only once for each
   function it appears in.)
```

Common compiler errors

- Missing semicolon

```
#include <stdio.h>

void print_something()

int main(int argc, char * argv[])
   {
   printf("I am a computer
    program\n");

   print_something();

   return 0;
   }

void print_something()
  {
  printf("Printing from a
   subroutine\n\n");
  }
```

- Compiler message

```
./build.csh
print_something.cpp: In function `void
    print_something()':
print_something.cpp:5: error: parse
    error before `int'
print_something.cpp: In function `void
    print_something()':
print_something.cpp:17: error:
    redefinition of `void
    print_something()'
print_something.cpp:5: error: `void
    print_something()' previously
    defined here
print_something.cpp:17: error:
    redefinition of `void
    print_something()'
print_something.cpp:5: error: `void
    print_something()' previously
    defined here
```

Common compiler errors

- Missing brace

```
 #include <stdio.h>

void print_something();
void print_something_else();

int main(int argc, char * argv[])
   {
   printf("I am a computer program\n");
   return 0;
   }

void print_something()
  {
  printf("Printing from a subroutine\n");
 // }

void print_something_else()
  {
  printf("Printing from subroutine II\n");
  }
```

- Compiler message

```
./build.csh
basic.cpp: In function `void
   print_something()':
basic.cpp:18: error: parse error
   before `{' token
```

Common compiler errors

- Missing library

```
//include <stdio.h>

void print_something();

int main(int argc, char * argv[])
   {
   printf("I am a computer program\n");
   return 0;
   }

void print_something()
  {
  printf("Printing from a subroutine\n");
  }
```

- Compiler message

```
./build.csh
basic.cpp: In function `int
   main(int, char**)':
basic.cpp:7: error: `printf'
   undeclared (first use this
   function)
basic.cpp:7: error: (Each
   undeclared identifier is
   reported only once for each
   function it appears in.)
basic.cpp: In function `void
   print_something()':
basic.cpp:14: error: `printf'
   undeclared (first use this
   function)
```

Common compiler errors

- Type ambiguity

```
#include <stdio.h>
#include <math.h>

int main(int argc, const char *argv[])
   {
   printf("I am a computer program");

   int var = log(10);
   return 0;
   }
```

- Compiler message

Visual C++

```
c:\Documents and Settings\Visual
tudioProjects\Example\MyApplication.cp
   p
(9): error C2668: 'log' : ambiguous
   call to overloaded function
```

GCC

```
./build.csh
overload.cpp: In function `int
   main(int, const char**)':
overload.cpp:9: warning: converting to
   `int' from `double'
```

Tips for working through compiler errors

- Start at the top
- Compiler may be “confused” by missing semicolons or braces and produce many error messages that don’t seem logical.
- Recompile as you work out errors
- Check for missing header file declarations and code libraries

Common run-time errors

- Uninitialized variables
- Memory errors
- Numeric errors
- Type errors in print statements
- Closing a NULL file pointer
- Accessing a NULL pointer
- Variables out of scope

Tips for tracking down run-time errors

- Use a debugger
- Debug data not code
- Isolate a small case
- Understand what's happening before making changes
- For updated code with new run-time errors
  - Suspect recent changes
  - Can also be a pre-existing bug that was uncovered by code changes

Avoiding errors

- Declare and initialize new variables as you code
- Write paired coding constructs at the same time
  - Open paren/ close paren
  - Allocate/deallocate
  - Constructor / destructor
  - fopen/fclose
- Always consider end cases
- Pay attention to variable scope
- Use error check code

GDB

- A simple debugger
- Helps test and evaluate programs by:
  - Stopping program at specific points
  - Running program one line at a time
  - Displaying the values of specific variables

Simple Usage

- Compile program with debug information

```
gcc –pg my_program.c –o my_program  -lm
```

- Load program into debugger

```
gdb my_program
```

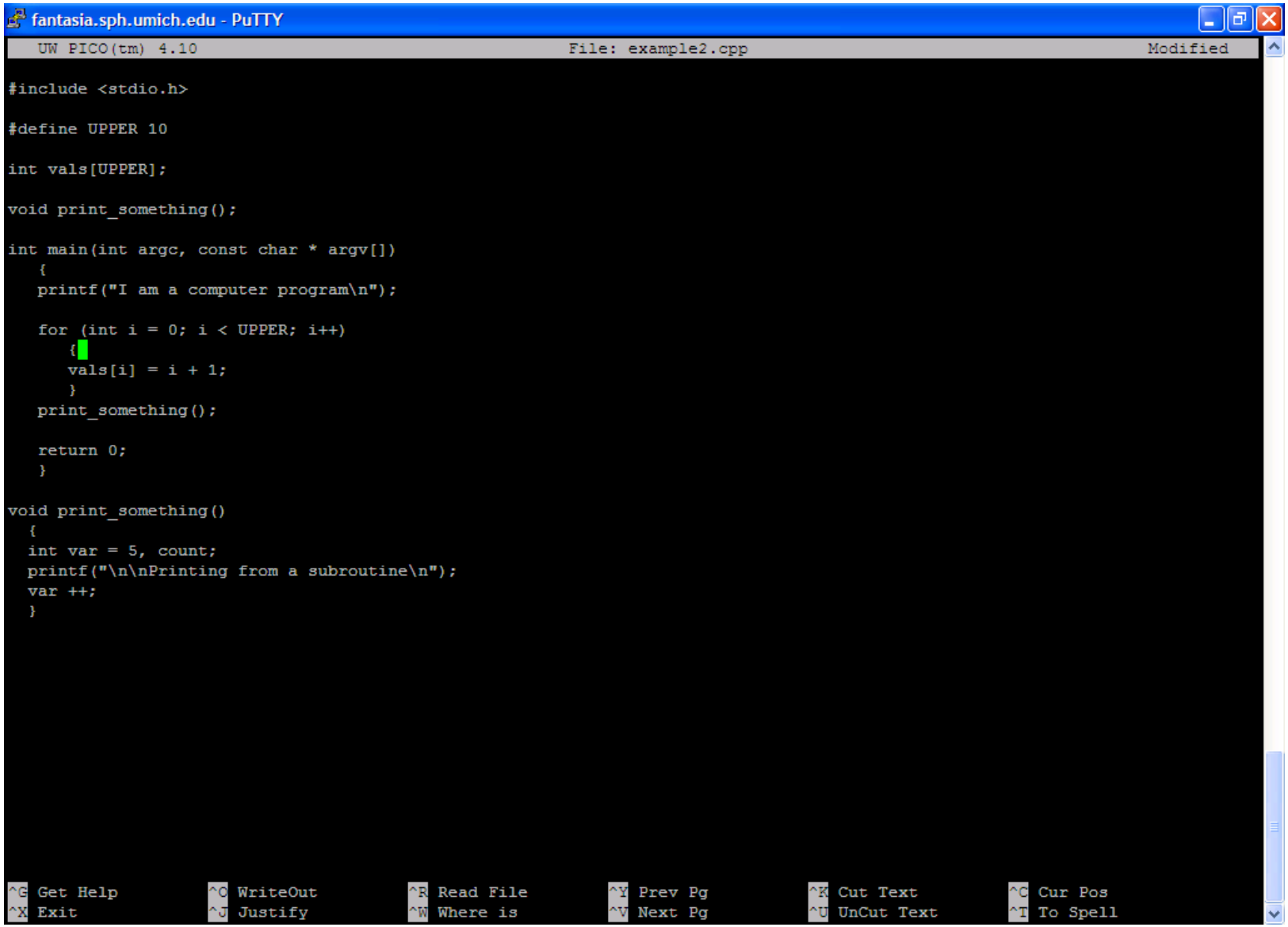
- Use debugger commands to control execution of program …

Essential GDB Commands

- `run`
  - Start execution of the program
- `next`
  - Execute a single line of code
- `step`
  - Execute a single line of code, stop at entry point of any called functions
- `continue`
  - Execute program until next *breakpoint*
- `where`
  - Reports line of code where crash occurs

More Essential GDB Commands

- `break main`
  - Stop program when `main` function is called
- `break n`
  - Stop program when line `n` is reached
- `print x`
  - Print contents of variable `x`
- `info locals`
  - Print all local variables


fantasia.sph.umich.edu - PuTTY
UW PICO(tm) 4.10
File: example2.cpp
Modified
#include <stdio.h>
#define UPPER 10
int vals[UPPER];
void print_something();
int main(int argc, const char * argv[])
{
printf("I am a computer program\n");
for (int i = 0; i < UPPER; i++)
{
vals[i] = i + 1;
}
print_something();
return 0;
}
void print_something()
{
int var = 5, count;
printf("\n\nPrinting from a subroutine\n");
var ++;
}
^G Get Help
^O WriteOut
^R Read File
^Y Prev Pg
^K Cut Text
^C Cur Pos
^X Exit
^J Justify
^W Where is
^V Next Pg
^U UnCut Text
^T To Spell

```
(gdb) break 1
Breakpoint 1 at 0x8048444: file example2.cpp, line
     1.
(gdb) list
2
3       #define UPPER 10
4
5       int vals[UPPER];
6
7       void print_something();
8
9       int main(int argc, const char * argv[])
10         {
11         printf("I am a computer program\n");

(gdb) run
Starting program:
     /home/wiggie/powerpoint/biostat851/example

Breakpoint 1, main (argc=134513732, argv=0x1) at
     example2.cpp:10
10         {
(gdb) n
11         printf("I am a computer program\n");
(gdb) n
I am a computer program
13         for (int i = 0; i < UPPER; i++)
(gdb) n
15            vals[i] = i + 1;
. (gdb) info locals
i = 0
(gdb) print vals[i]
$1 = 0
(gdb) s
13         for (int i = 0; i < UPPER; i++)
(gdb) print vals[i]
$2 = 1
(gdb) s
15            vals[i] = i + 1;
(gdb) break print_something()
Breakpoint 2 at 0x8048496: file example2.cpp, line
     24.
(gdb) cont
Continuing.

Breakpoint 2, print_something () at example2.cpp:24
24        int var = 5, count;
(gdb) list
19         return 0;
20         }
21
22      void print_something()
23        {
24        int var = 5, count;
25        printf("\n\nPrinting from a
     subroutine\n");
26        var ++;
27        }
28
(gdb) n
25        printf("\n\nPrinting from a
     subroutine\n");
(gdb) print count
$3 = -1076624204
(gdb) print var var
A syntax error in expression, near `var'.
(gdb) print var
$4 = 5
(gdb) n


Printing from a subroutine
26        var ++;
(gdb) n
```

Debugging with printf

```
int printf(char * format, …);
```

- Writes formatted output
- Help localize problems, particularly for tight loops or when array variables are suspect
- Output is buffered, so printf should be followed call to fflush(stdout)

printf

```
int printf(char * format, …);
```

- Format string controls how arguments are converted to text
- Parameters are printed as specified in % fields
- %[flags][width][precision]type
- Otherwise, string is quoted

printf fields

`%[flags][width][precision]type`

Flags:

- -” to left justify result • “+” to show sign in positive numbers

Width

- Minimum number of characters to print

Precision

- Number of digits after decimal (for floating point) • Maximum number of characters (for strings)

Type

- “s” for strings
- “d” for integers, “x” to print hexadecimal integers •
- “f” for floating point, “e” for exponential notation, “g” for automatic

About compiler flags

- Running GDB on executable that isn't compiled with debug flags, will produce confusing output

```
(gdb) run
Starting program: /home/wiggie/powerpoint/biostat851/basic

Breakpoint 1, 0x0804844a in main ()
(gdb) list
1       in ../sysdeps/i386/elf/start.S
(gdb) print
The history is empty.
.
(gdb) info locals
No symbol table info available.
```

- Recompile finished code without debug flags
- Optimization flags (-O2 –O3) can make executable faster
- -Wall for "picky" compilation

GPROF

- Works with GCC to collect information about a program's execution
  - How often is each function called?
  - How much time is spent in each function?
- Works in three steps
  - Compile code
  - Execute program
  - Tabulate profile information

Simple Usage

- Compile program with profile information

```
gcc –pg my_program.c –o my_program -lm
```

- Execute program

```
./my_program
```

- Summarize profile information

```
gprof my_program
```

time

```
fantasia:~/powerpoint/biostat851> time ./example
I am a computer program
I can count

1 potato
2 potato
3 potato
4 potato
5 potato
6 potato
7 potato
8 potato
9 potato
10 potato


Printing from a subroutine
0.004u 0.000s 0:00.02 0.0%      0+0k 0+0io 0pf+0w
fantasia:~/powerpoint/biostat851>
```

Getting Help

- From the UNIX prompt, you can use the `man` command to get help …

- For example:
  - `man gcc` - info on how to run `gcc`
  - `man printf` - info on using `printf()`

Working in a Windows Environment

- Good integrated toolsets exist
- Good options include:
  - Microsoft Visual Studio / Visual C++
    - Discounted version available through the University
  - Turbo C++ Explorer
    - Free C/C++ compiler, www.turboexplorer.com

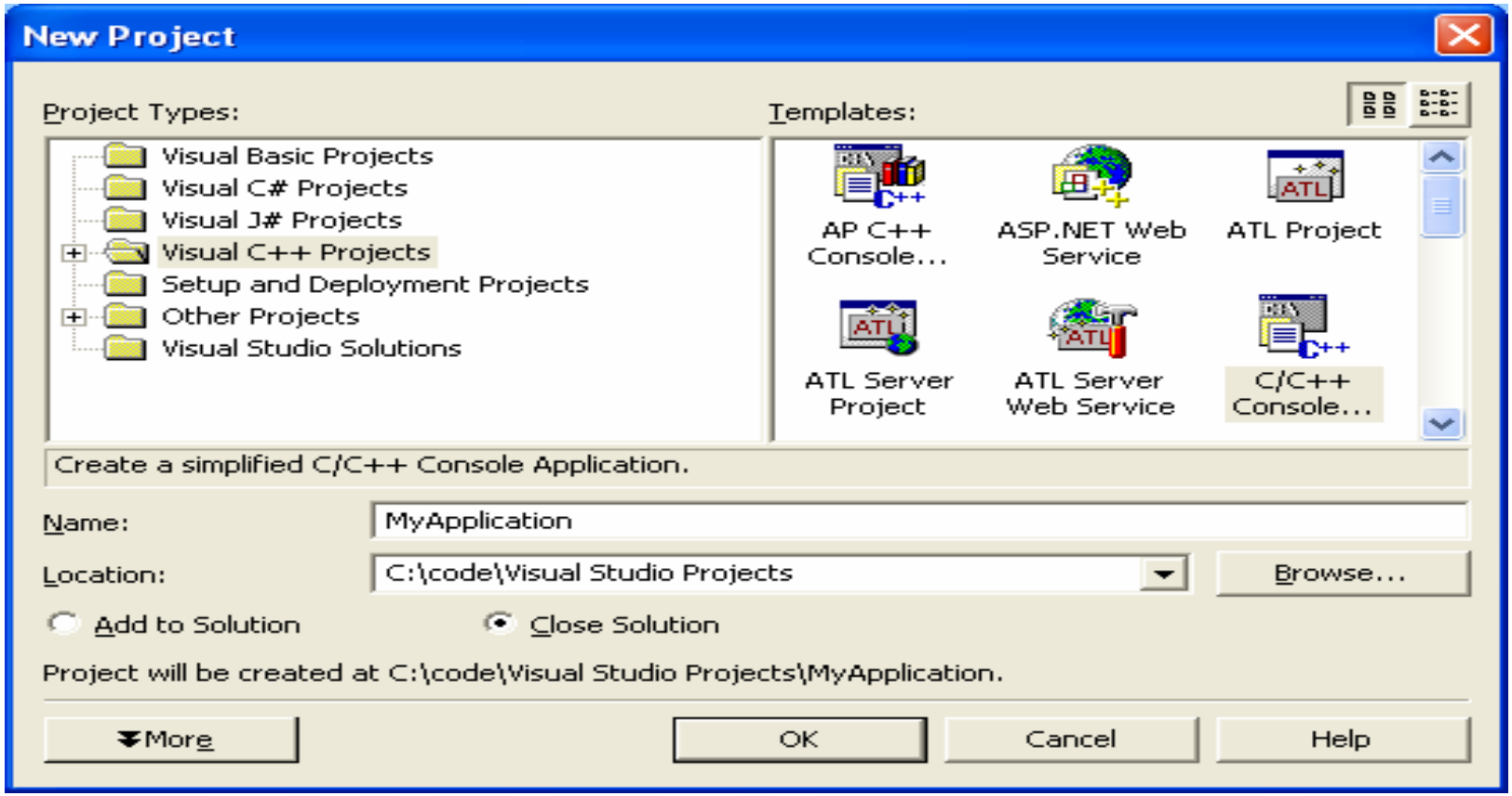
Microsoft Visual Studio

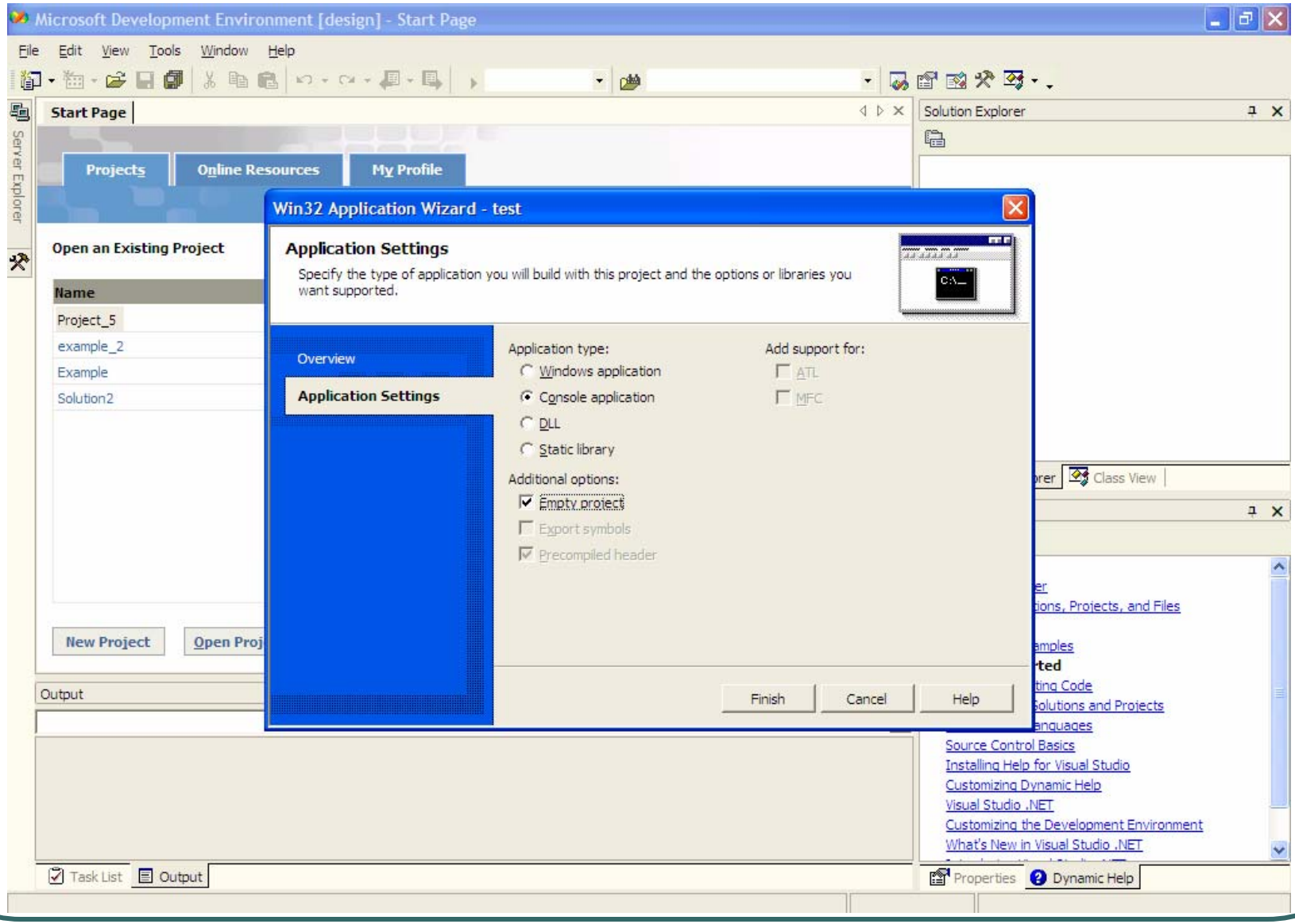
- "Student Tools for Visual Studio"
  - Included in Academic Edition
  - Simplifies environment slightly
- Each application you develop is a project
  - Can include a collection of source files
  - But only one for most of our examples
- Many types of applications possible …
  - Our examples will be "Console Applications"

“All-In-One”

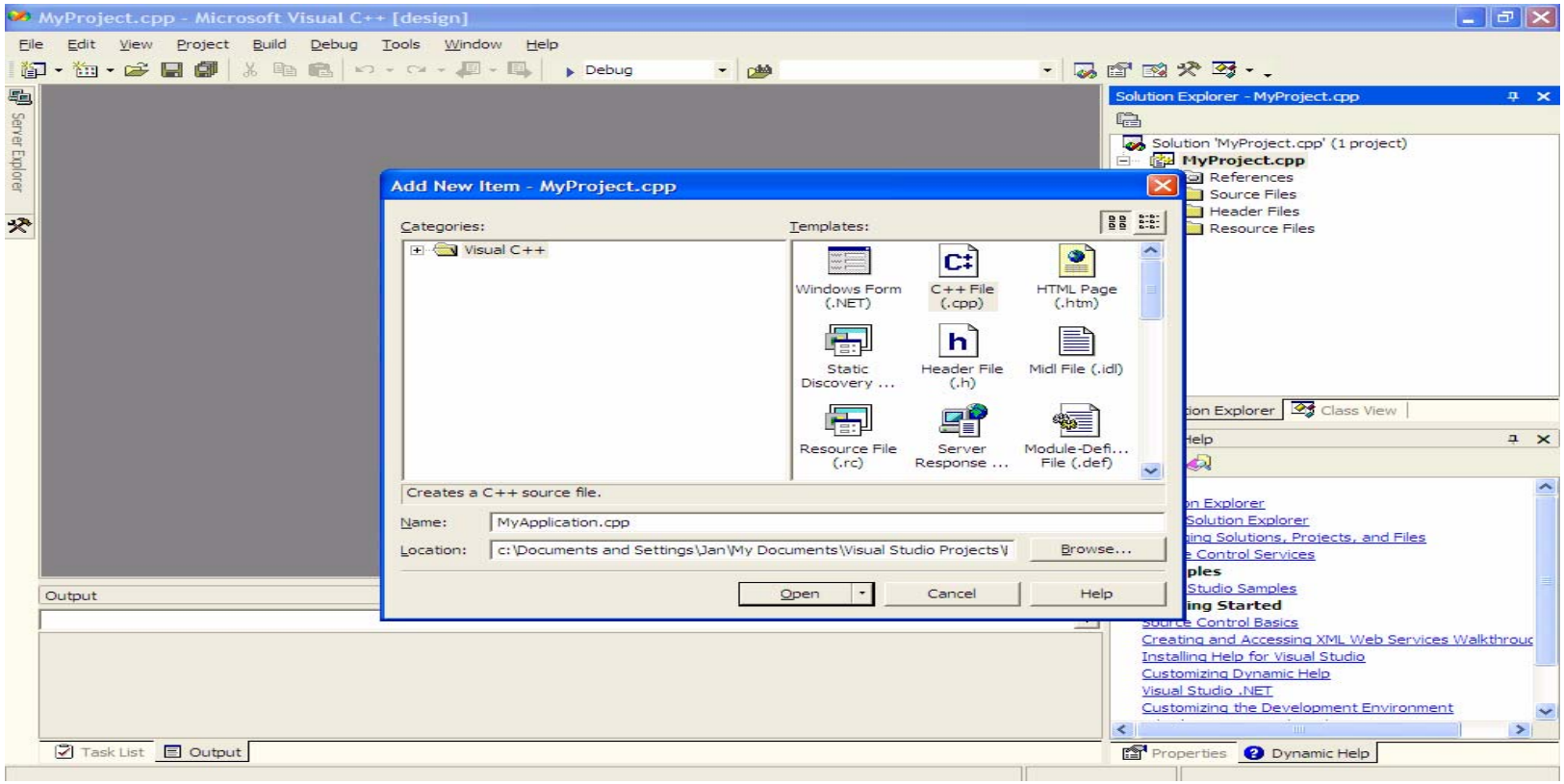
- Edit and manage source files
- Compile code
- Run, debug and test

File | New | Project ...



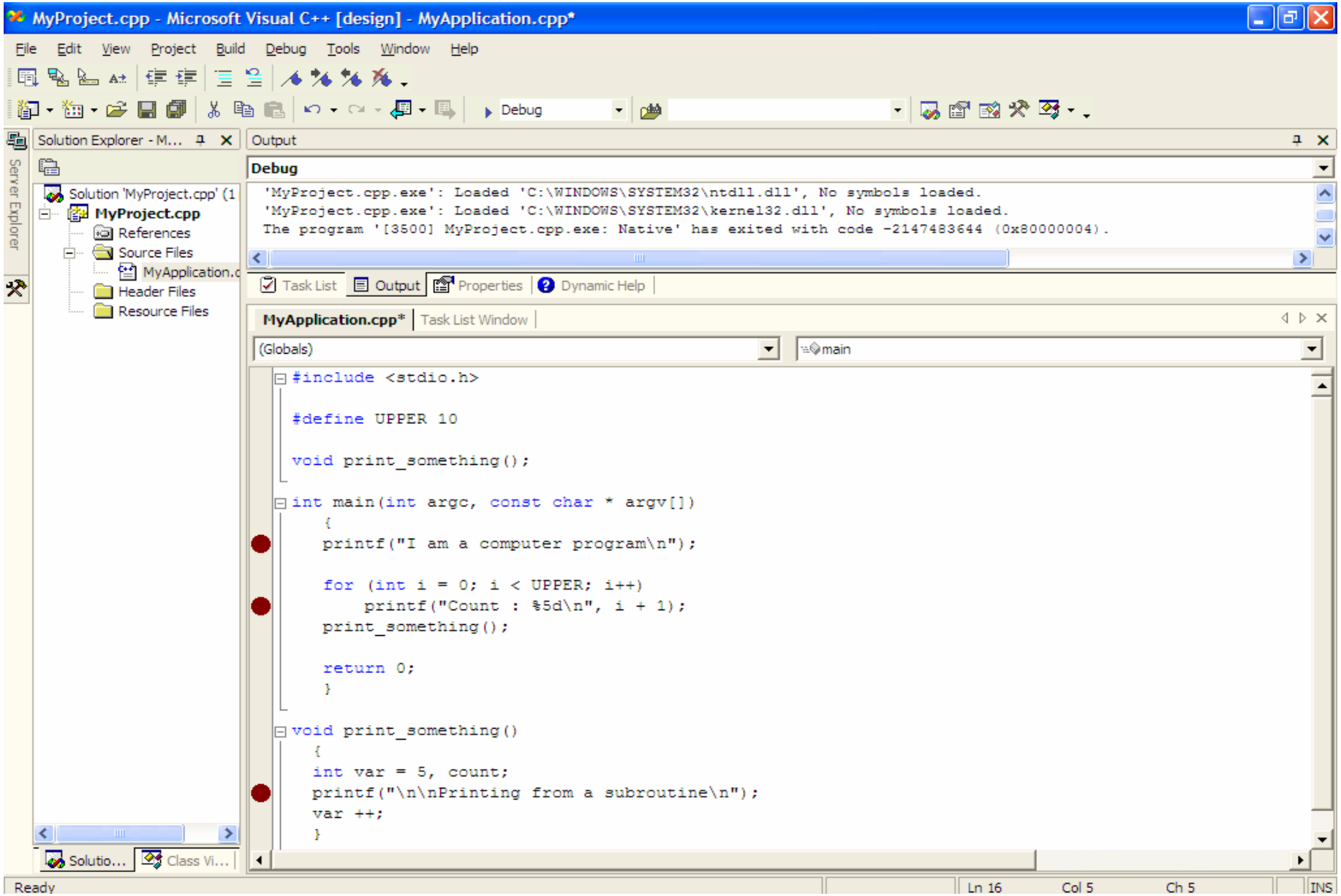

Microsoft Development Environment [design] - Start Page
File Edit View Tools Window Help
Start Page
Projects
Online Resources
My Profile
Open an Existing Project
Name
Project_5
example_2
Example
Solution2
New Project
Open Proj
Server Explorer
Win32 Application Wizard - test
Application Settings
Specify the type of application you will build with this project and the options or libraries you want supported.
Overview
Application Settings
Application type:
Windows application
Console application
DLL
Static library
Additional options:
Empty project
Export symbols
Precompiled header
Add support for:
ATL
MFC
Finish
Cancel
Help
Solution Explorer
Class View
Output
Task List
Output
Properties
Dynamic Help
Source Control Basics
Installing Help for Visual Studio
Customizing Dynamic Help
Visual Studio .NET
Customizing the Development Environment
What's New in Visual Studio .NET

Project | Add New Item | C++ File



F1: Online Help System

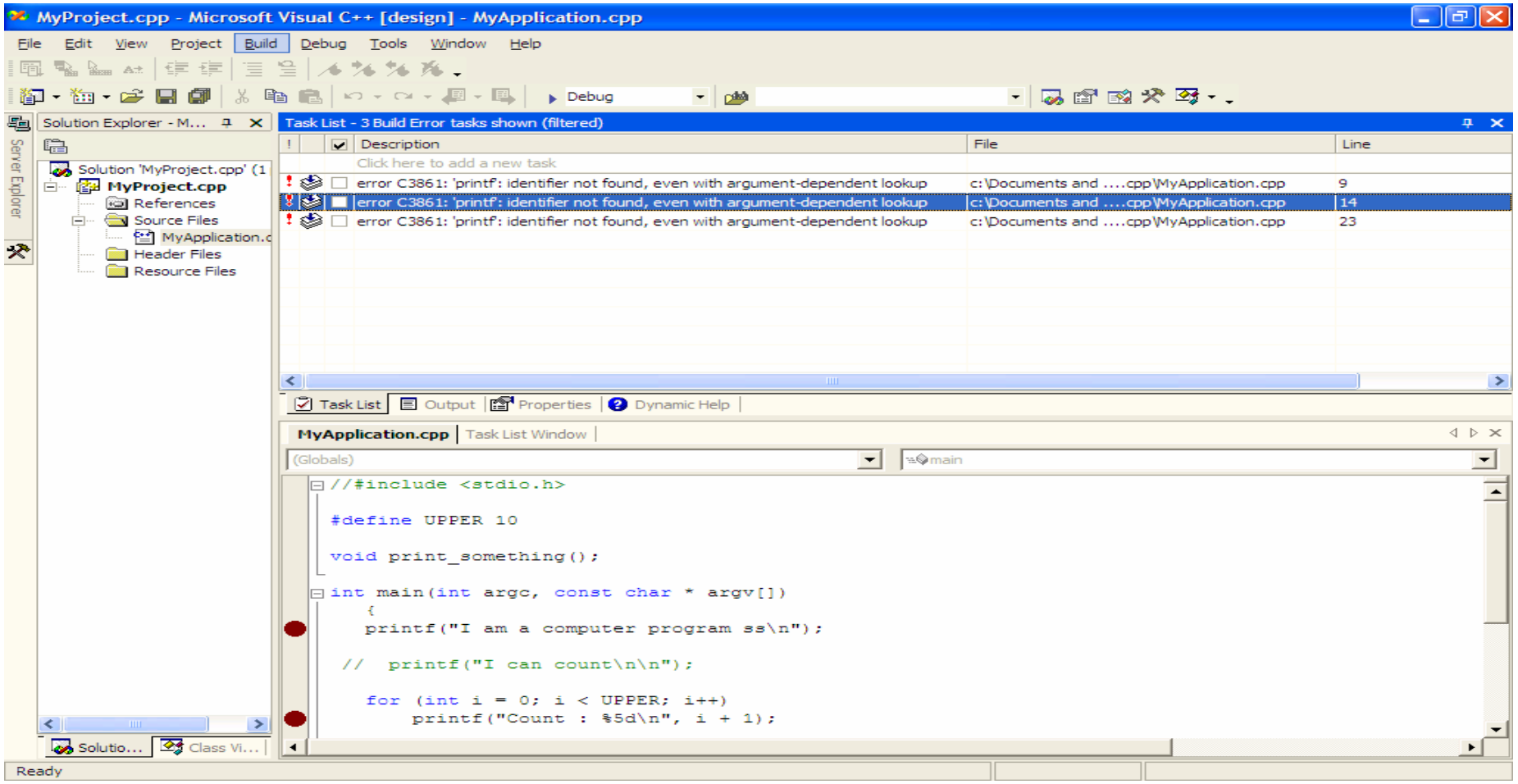
- Visual Studio includes an extensive integrated help system
- Using the F1 key, you can get info on standard library functions:
  - Including examples of how the function should be used.


MyProject.cpp - Microsoft Visual C++ [design] - MyApplication.cpp*
File Edit View Project Build Debug Tools Window Help
Debug
Solution Explorer - M...
Solution 'MyProject.cpp' (1
MyProject.cpp
References
Source Files
MyApplication.c
Header Files
Resource Files
Output
Debug
'MyProject.cpp.exe': Loaded 'C:\WINDOWS\SYSTEM32\ntdll.dll', No symbols loaded.
'MyProject.cpp.exe': Loaded 'C:\WINDOWS\SYSTEM32\kernel32.dll', No symbols loaded.
The program '[3500] MyProject.cpp.exe: Native' has exited with code -2147483644 (0x80000004).
Task List
Output
Properties
Dynamic Help
MyApplication.cpp*
Task List Window
(Globals)
main
#include <stdio.h>
#define UPPER 10
void print_something();
int main(int argc, const char * argv[])
{
printf("I am a computer program\n");
for (int i = 0; i < UPPER; i++)
printf("Count : %5d\n", i + 1);
print_something();
return 0;
}
void print_something()
{
int var = 5, count;
printf("\n\nPrinting from a subroutine\n");
var ++;
}
Solutio...
Class Vi...
Ready
Ln 16
Col 5
Ch 5
INS

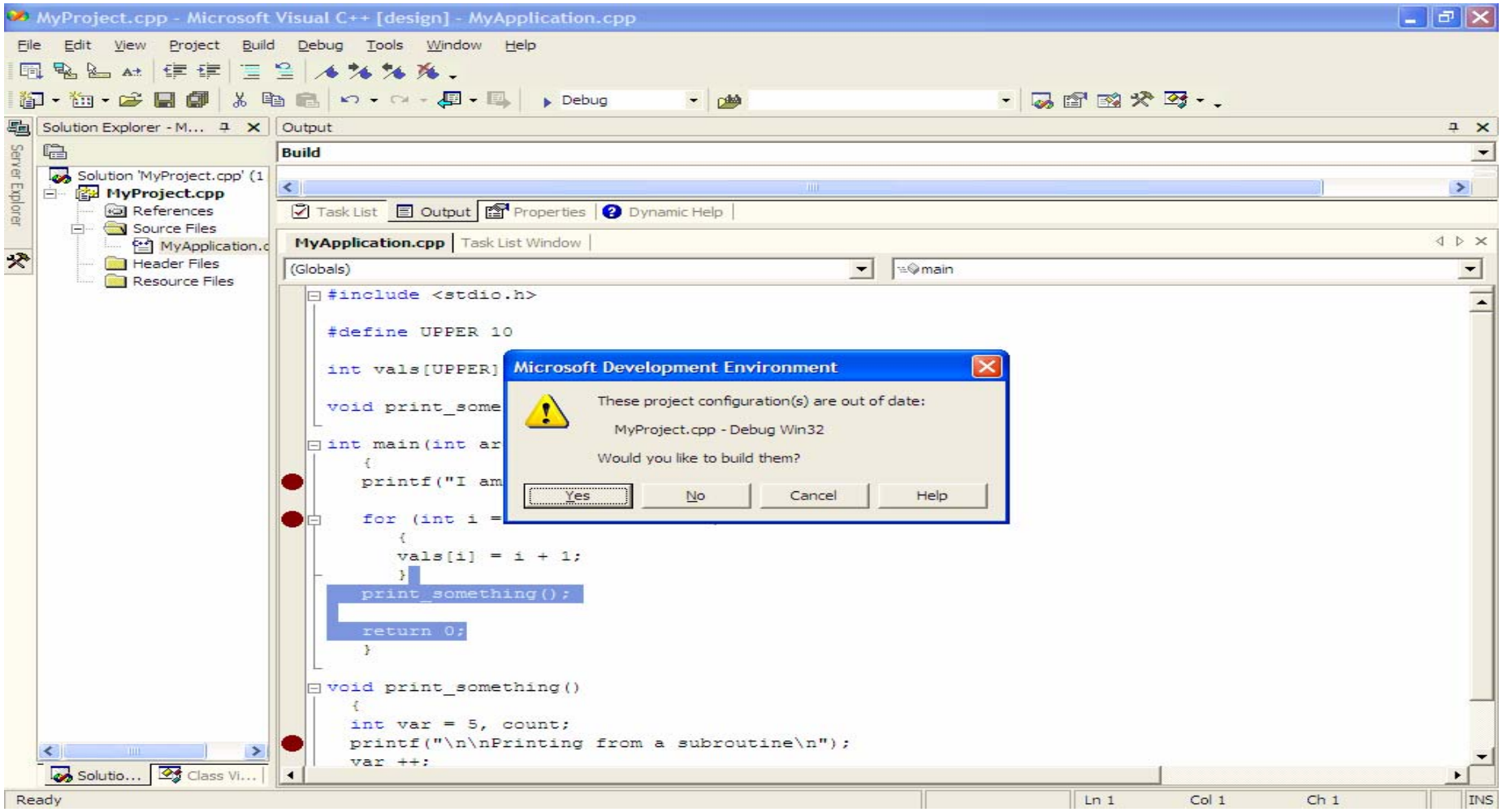
Important Commands

- Build | Build MyApplication
  - Compiles code, identifies problems with source
- Debug | Start, Step Into, Step Over
  - Manage execution of program
- Debug | QuickWatch
  - Examine contents of variables as program runs
- Click to the left of particular source lines to set breakpoints

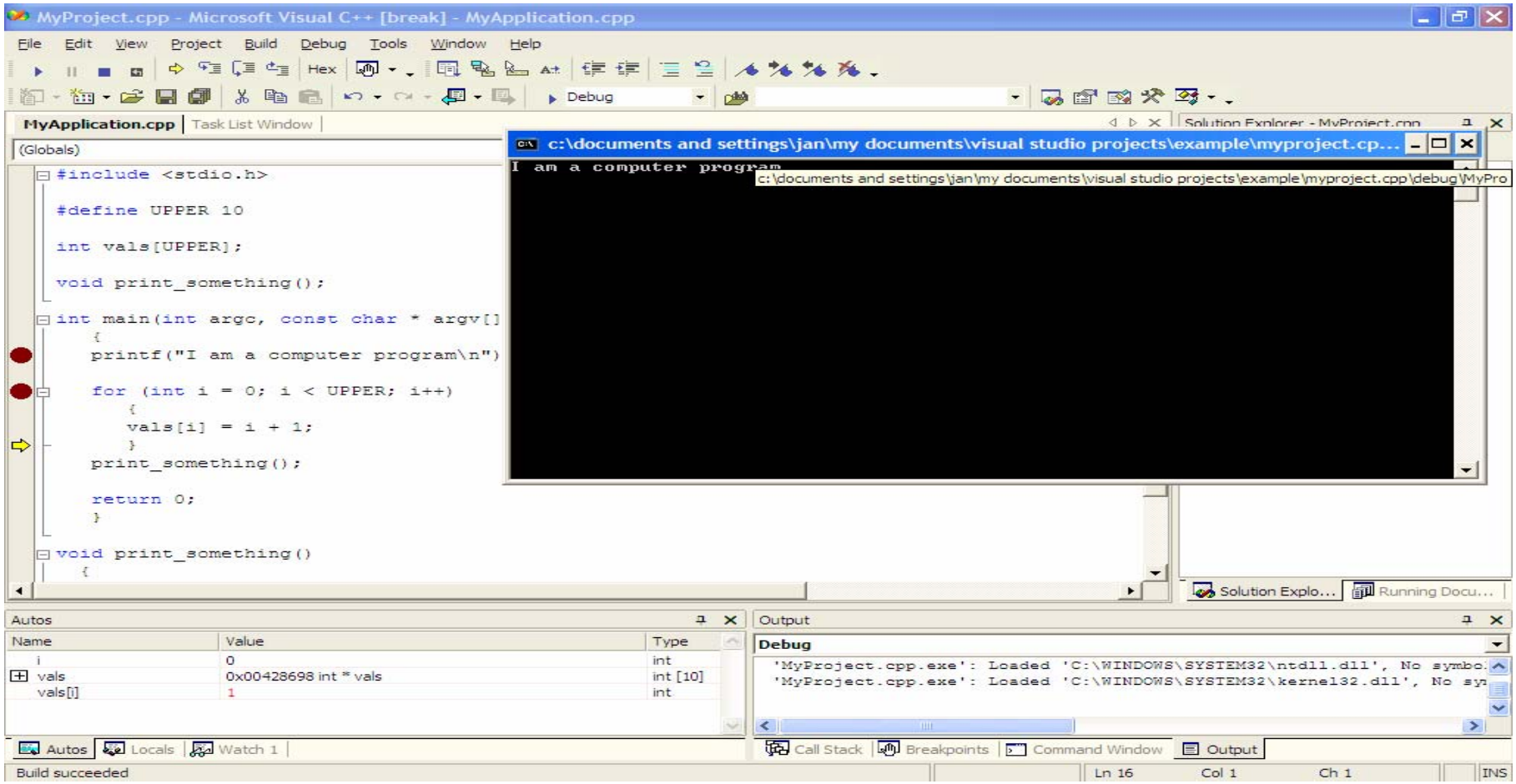
Build | Build MyProject.cpp



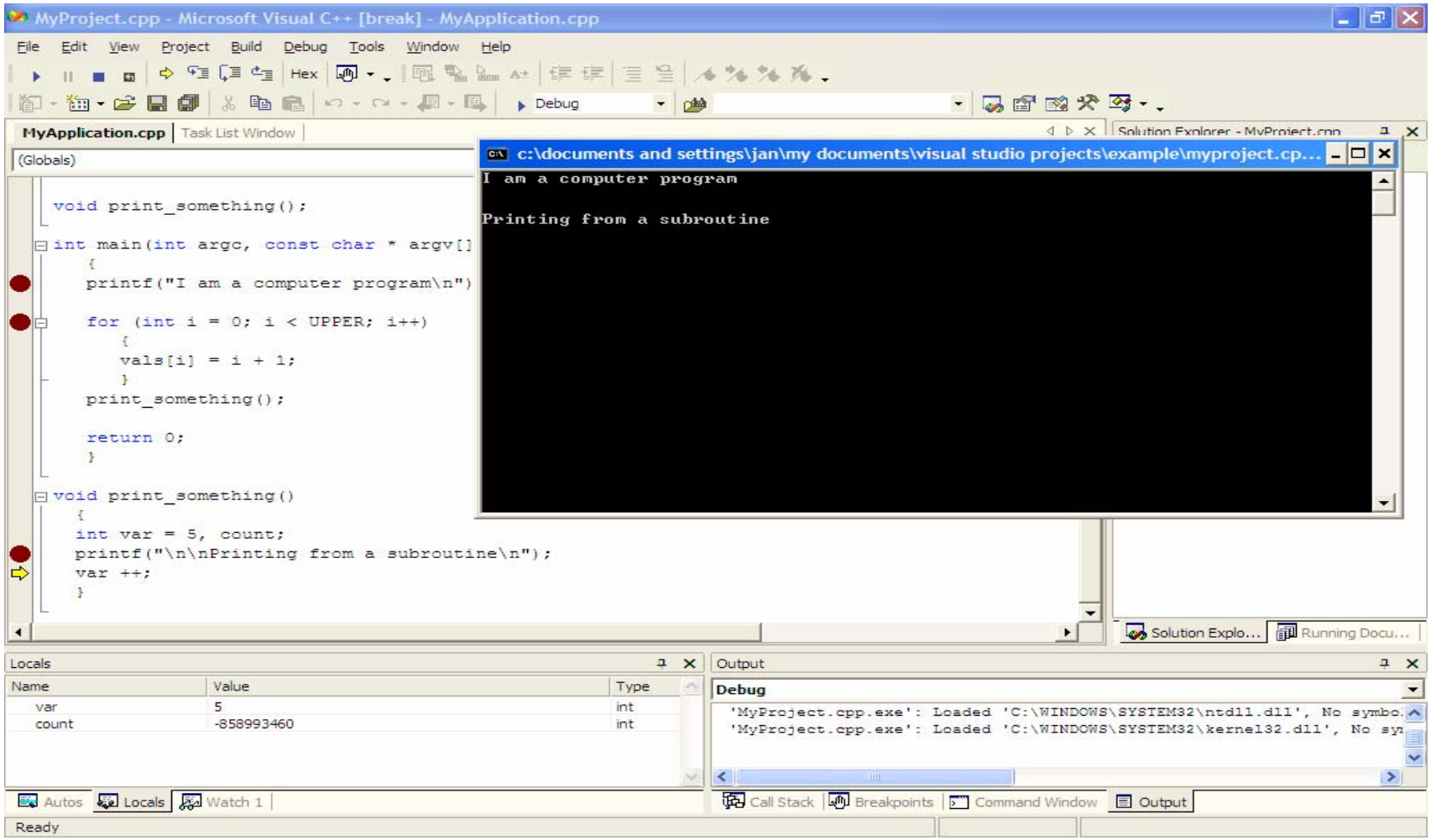
Debug | Start


MyProject.cpp - Microsoft Visual C++ [design] - MyApplication.cpp
File Edit View Project Build Debug Tools Window Help
Debug
Solution Explorer - M...
Solution 'MyProject.cpp' (1
MyProject.cpp
References
Source Files
MyApplication.c
Header Files
Resource Files
Output
Build
Task List Output Properties Dynamic Help
MyApplication.cpp Task List Window
(Globals)
main
#include <stdio.h>
#define UPPER 10
int vals[UPPER]
void print_some
int main(int ar
{
printf("I am
for (int i =
{
vals[i] = i + 1;
}
print_something();
return 0;
}
void print_something()
{
int var = 5, count;
printf("\n\nPrinting from a subroutine\n");
var ++;
Microsoft Development Environment
These project configuration(s) are out of date:
MyProject.cpp - Debug Win32
Would you like to build them?
Yes
No
Cancel
Help
Ready
Ln 1
Col 1
Ch 1
INS

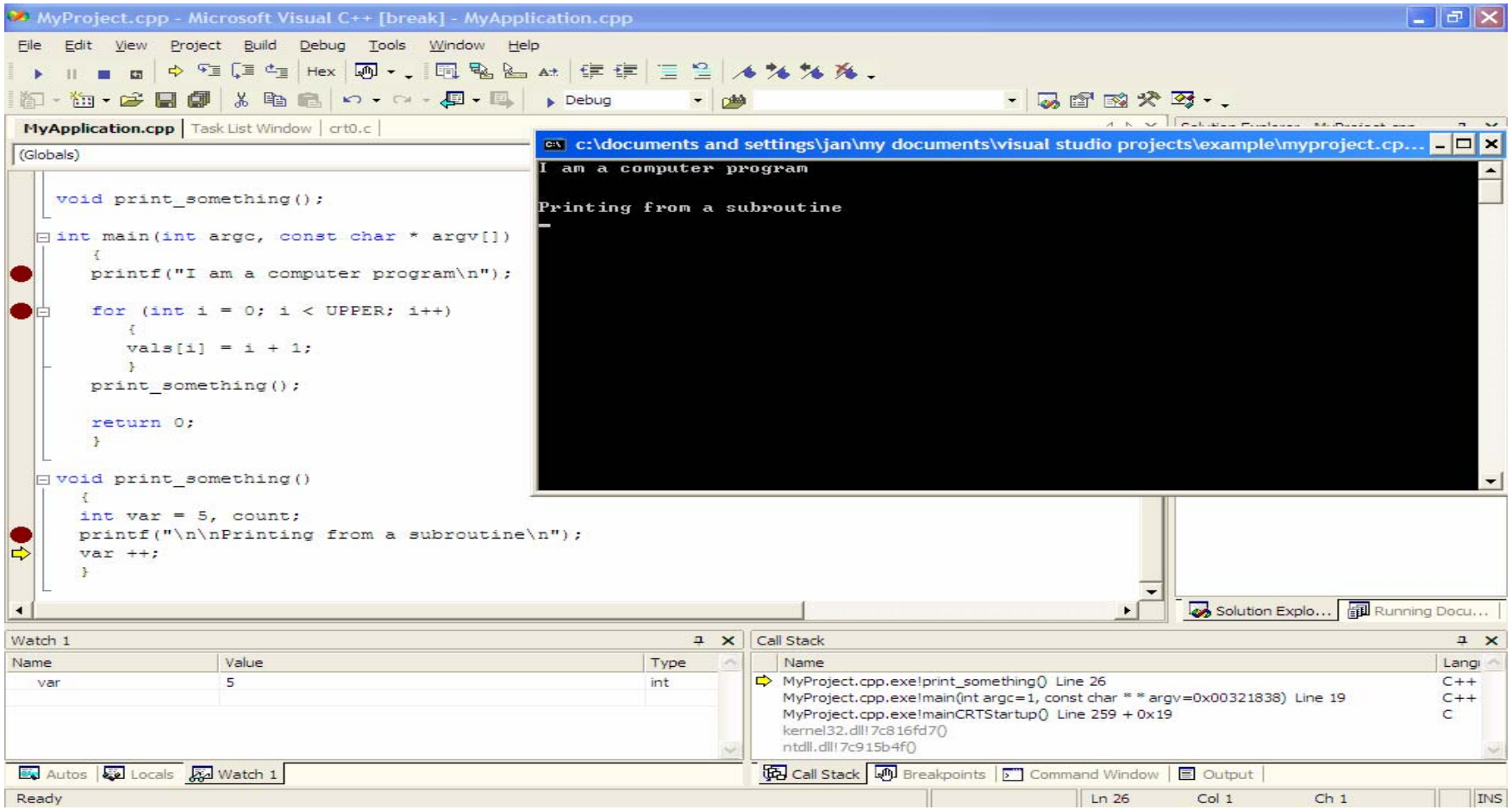
Debug | Start


MyProject.cpp - Microsoft Visual C++ [break] - MyApplication.cpp
File Edit View Project Build Debug Tools Window Help
Debug
MyApplication.cpp | Task List Window
(Globals)
#include <stdio.h>
#define UPPER 10
int vals[UPPER];
void print_something();
int main(int argc, const char * argv[]
{
printf("I am a computer program\n")
for (int i = 0; i < UPPER; i++)
{
vals[i] = i + 1;
}
print_something();
return 0;
}
void print_something()
{
c:\documents and settings\jan\my documents\visual studio projects\example\myproject.cp...
I am a computer program
c:\documents and settings\jan\my documents\visual studio projects\example\myproject.cpp\debug\MyPro
Solution Explo...
Running Docu...
Autos
Name
Value
Type
i
0
int
vals
0x00428698 int * vals
int [10]
vals[i]
1
int
Autos
Locals
Watch 1
Output
Debug
'MyProject.cpp.exe': Loaded 'C:\WINDOWS\SYSTEM32\ntdll.dll', No symbo
'MyProject.cpp.exe': Loaded 'C:\WINDOWS\SYSTEM32\kernel32.dll', No sy
Call Stack
Breakpoints
Command Window
Output
Build succeeded
Ln 16
Col 1
Ch 1
INS

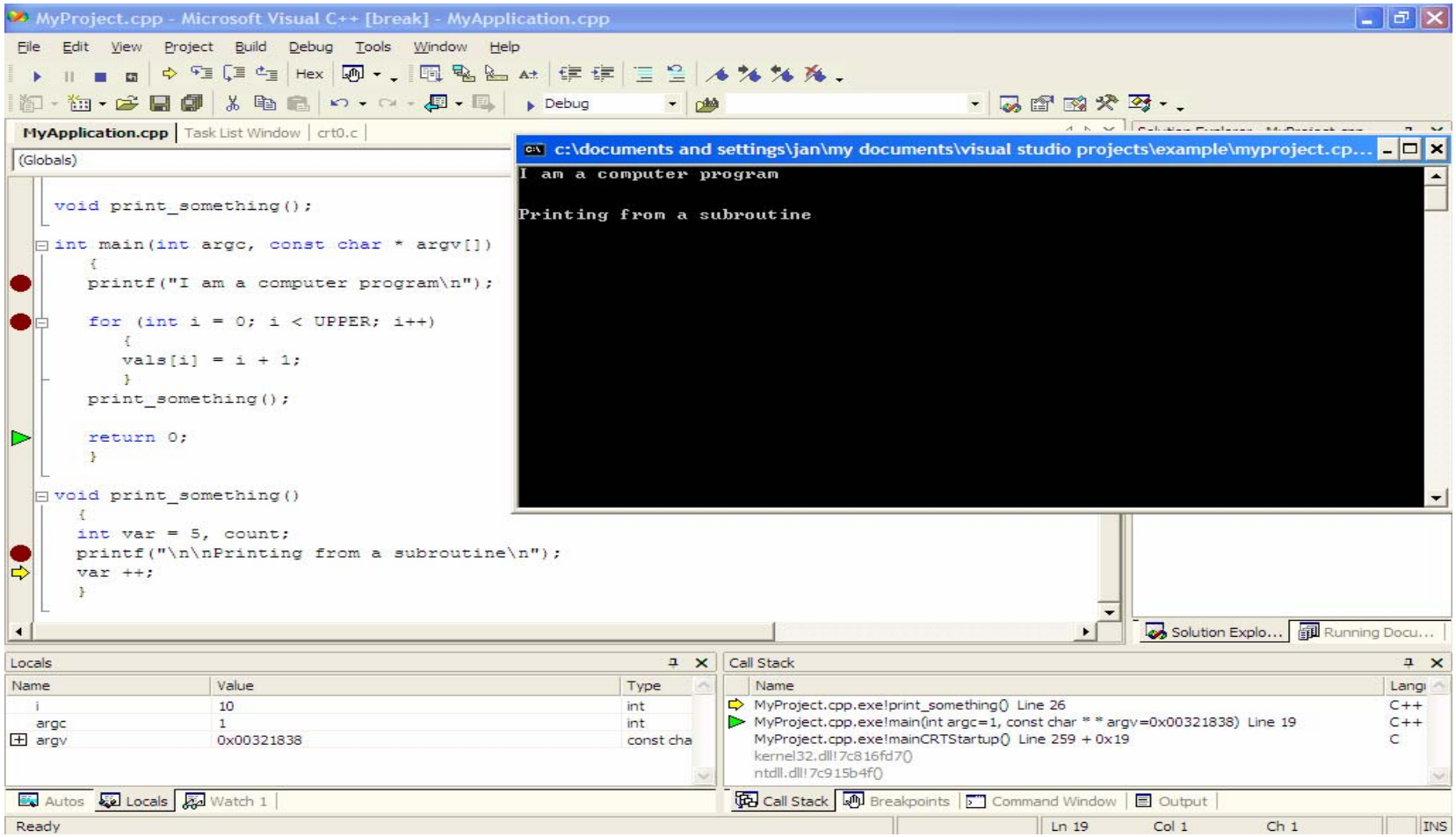
Stepping through code



Adding a watch variable



Call stack


MyProject.cpp - Microsoft Visual C++ [break] - MyApplication.cpp
File Edit View Project Build Debug Tools Window Help
Debug
MyApplication.cpp | Task List Window | crt0.c
(Globals)
void print_something();
int main(int argc, const char * argv[])
{
printf("I am a computer program\n");
for (int i = 0; i < UPPER; i++)
{
vals[i] = i + 1;
}
print_something();
return 0;
}
void print_something()
{
int var = 5, count;
printf("\n\nPrinting from a subroutine\n");
var ++;
}
c:\documents and settings\jan\my documents\visual studio projects\example\myproject.cp...
I am a computer program
Printing from a subroutine
Solution Explo... | Running Docu...
Locals
Name | Value | Type
i | 10 | int
argc | 1 | int
argv | 0x00321838 | const cha
Autos | Locals | Watch 1
Call Stack
Name | Lang
MyProject.cpp.exe!print_something() Line 26 C++
MyProject.cpp.exe!main(int argc=1, const char * * argv=0x00321838) Line 19 C++
MyProject.cpp.exe!mainCRTStartup() Line 259 + 0x19 C
kernel32.dll!7c816fd7()
ntdll.dll!7c915b4f()
Call Stack | Breakpoints | Command Window | Output
Ready
Ln 19 Col 1 Ch 1
INS

Today

- How to compile, run and debug C code
- GNU tool collection
  - Unix / Linux / newer Macs
- Integrated environments, such as Visual Studio
  - Windows

Questions/Clarification?

- Feel free to drop me an e-mail :

Janis Wigginton
wiggie@umich.edu