

# Embedded computing for scientific and industrial imaging applications

---

Lecture 4 - C demo, Newton's method

# Outline

C demo with visual studio

- Computing square roots by Newton's method
- Incremental developing by using git

# Computing square roots

Hardware arithmetic units can add, subtract, multiply, divide. Other mathematical functions usually take some software.

$$\sqrt{2} \approx 1.4142135623730951$$

In most languages, `sqrt(2)` computes this.

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

```
sqrt(2);
```

# Newton's method

**Problem:** Find a solution of  $f(s) = 0$  (zero or root of  $f$ )

**Idea:** Given approximation  $s^{[k]}$ ,  
approximate  $f(s)$  by a linear function,  
the tangent line at  $(s^{[k]}, f(s^{[k]}))$ .

Find unique zero of this function and use as  $s^{[k+1]}$ .

**Updating formula:**

$$s^{[k+1]} = s^{[k]} - \frac{f(s^{[k]})}{f'(s^{[k]})}$$

# Approximate $s = \sqrt{x}$

Newton's method to find root of  $s^2 - x = 0$ .

```
double s = 1.;  
for (int i = 0; i < maxiter; i++)  
{  
    s = 0.5 * (s + x / s);  
}
```

where `maxiter` is some maximum number of iterations.

# Demo

## Goals:

- Develop our own version of sqrt function.
- Start simple and add complexity in stages.
- Illustrate some C programming.
- Illustrate use of git to track our development
- Get familiar with visual studio IDE

We will do this in codes/mysqrt directory so you can examine the various versions later.

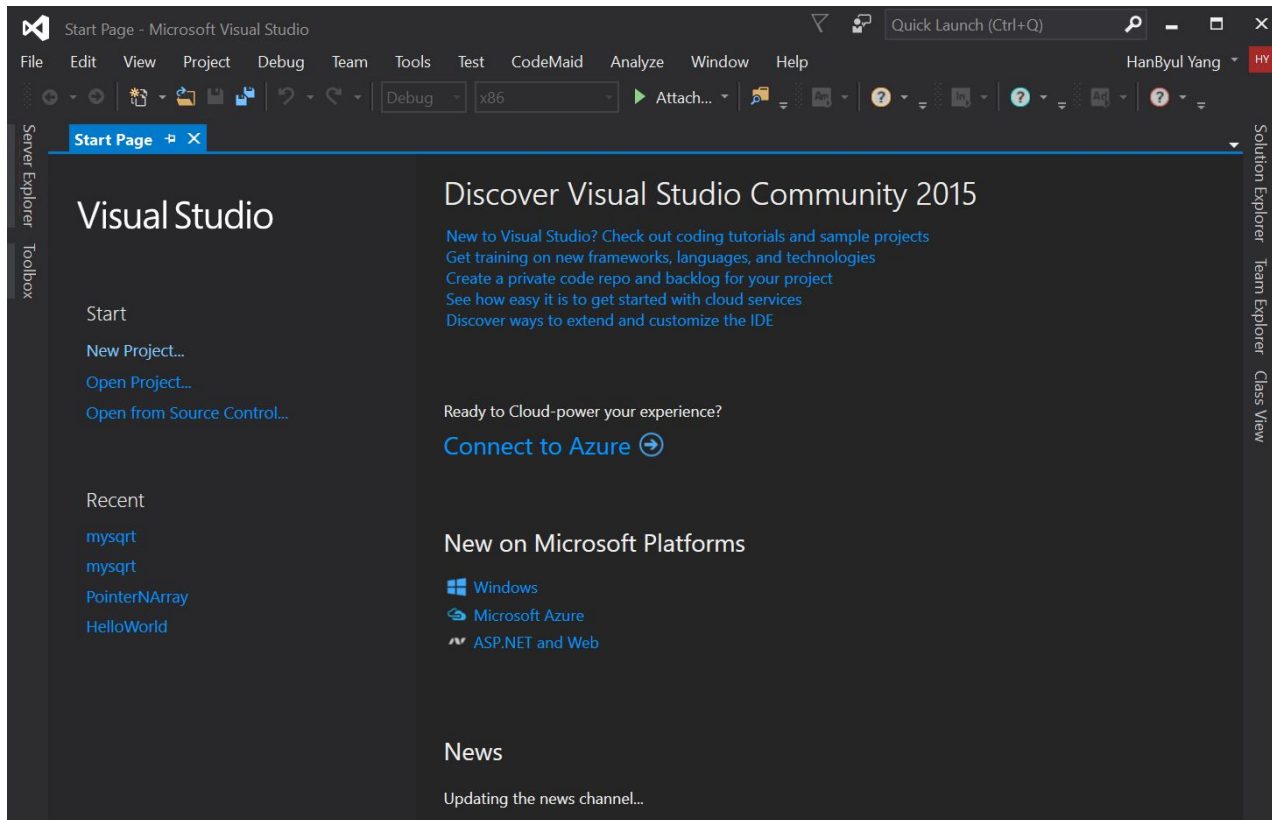
# Demo : mysqrt.c

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

int main()
{
    double x = 2;
    double s = sqrt(x);
    printf("sqrt(%f) = %f\n", x, s);
}
```

# Demo : visual studio

Start “New Project”



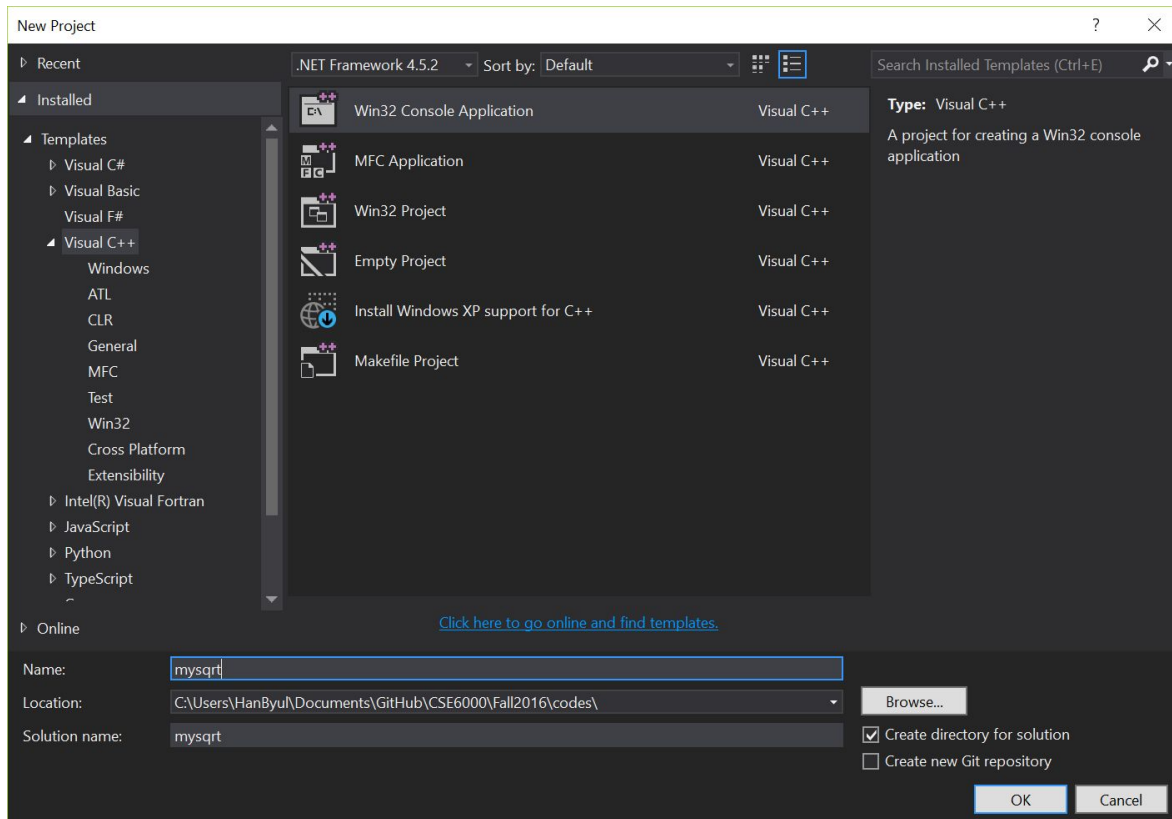


# Demo : new project

Visual c++

Win32 Console application

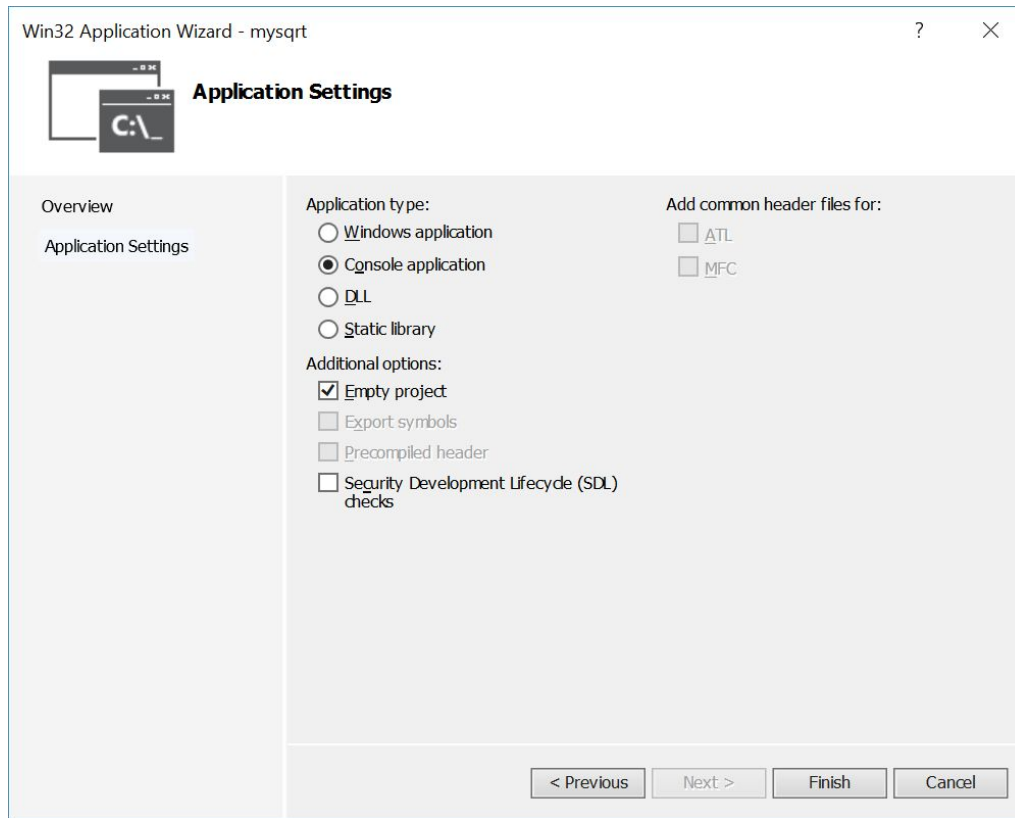
“mysql”



# Demo : application setting

Console application

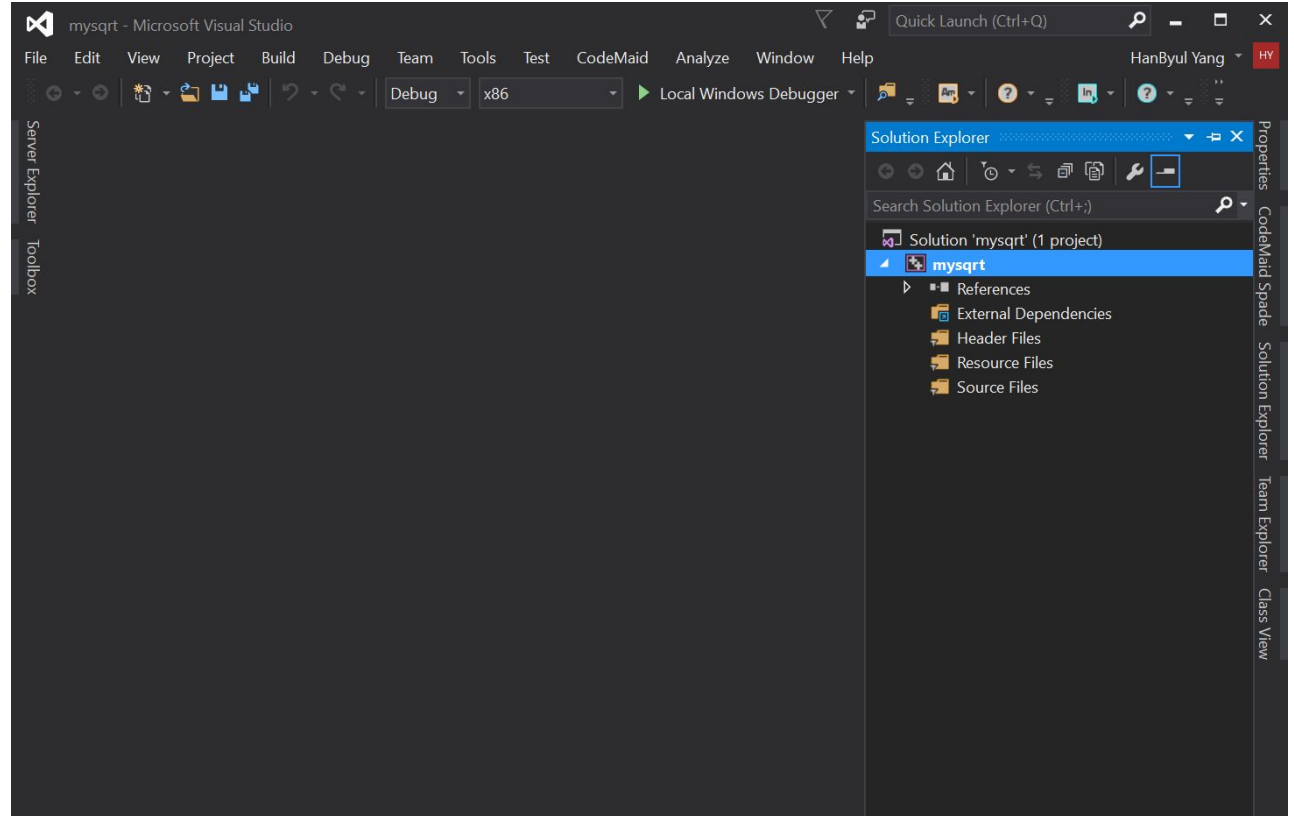
Empty project



# Demo : Solution Explorer

Solution

Project

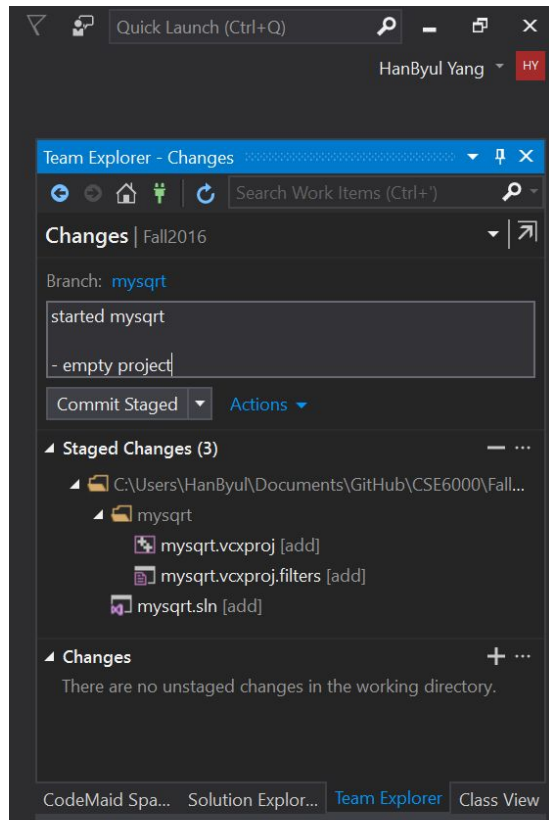


# Demo : Team Explorer

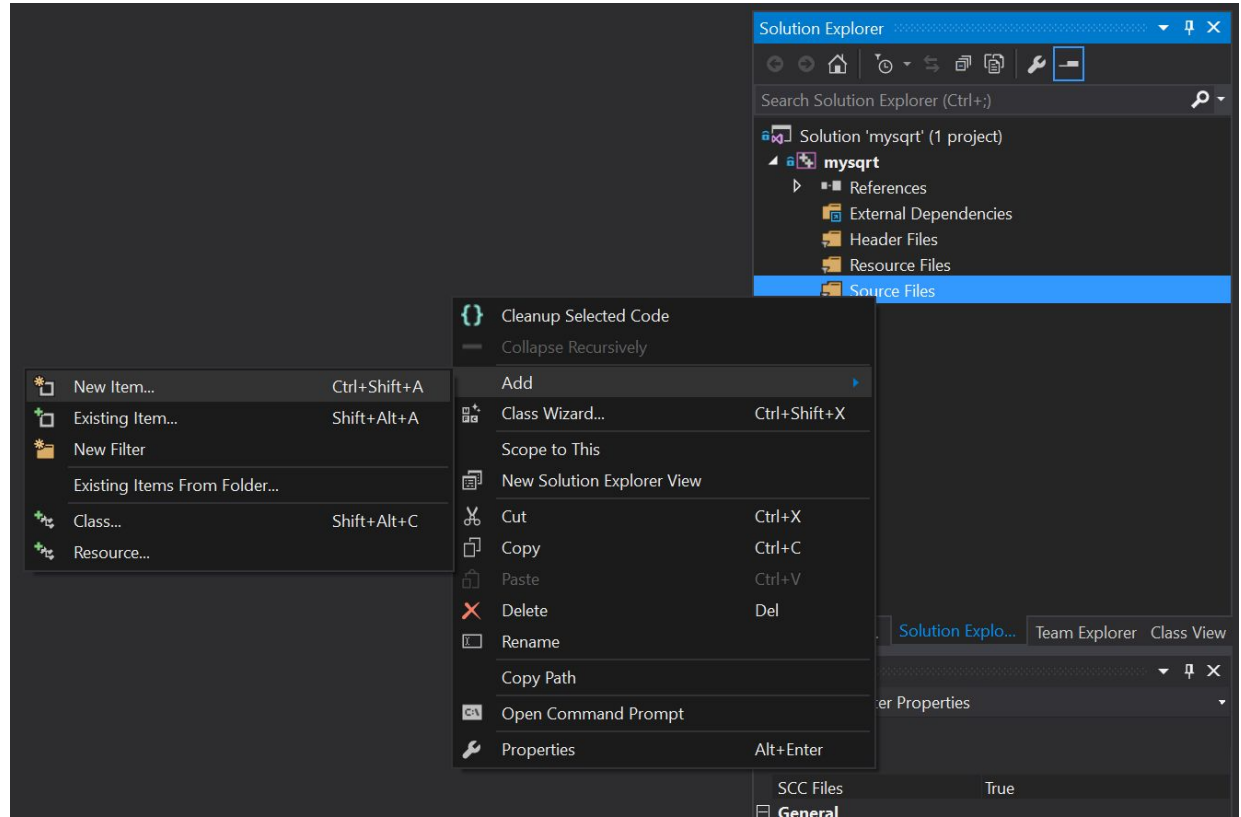
Branch : mysqlt

Commit 3 files

- mysqlt.sln
- mysqlt.vcxproj
- mysqlt.vcxproj.filters



# Add “New Item”



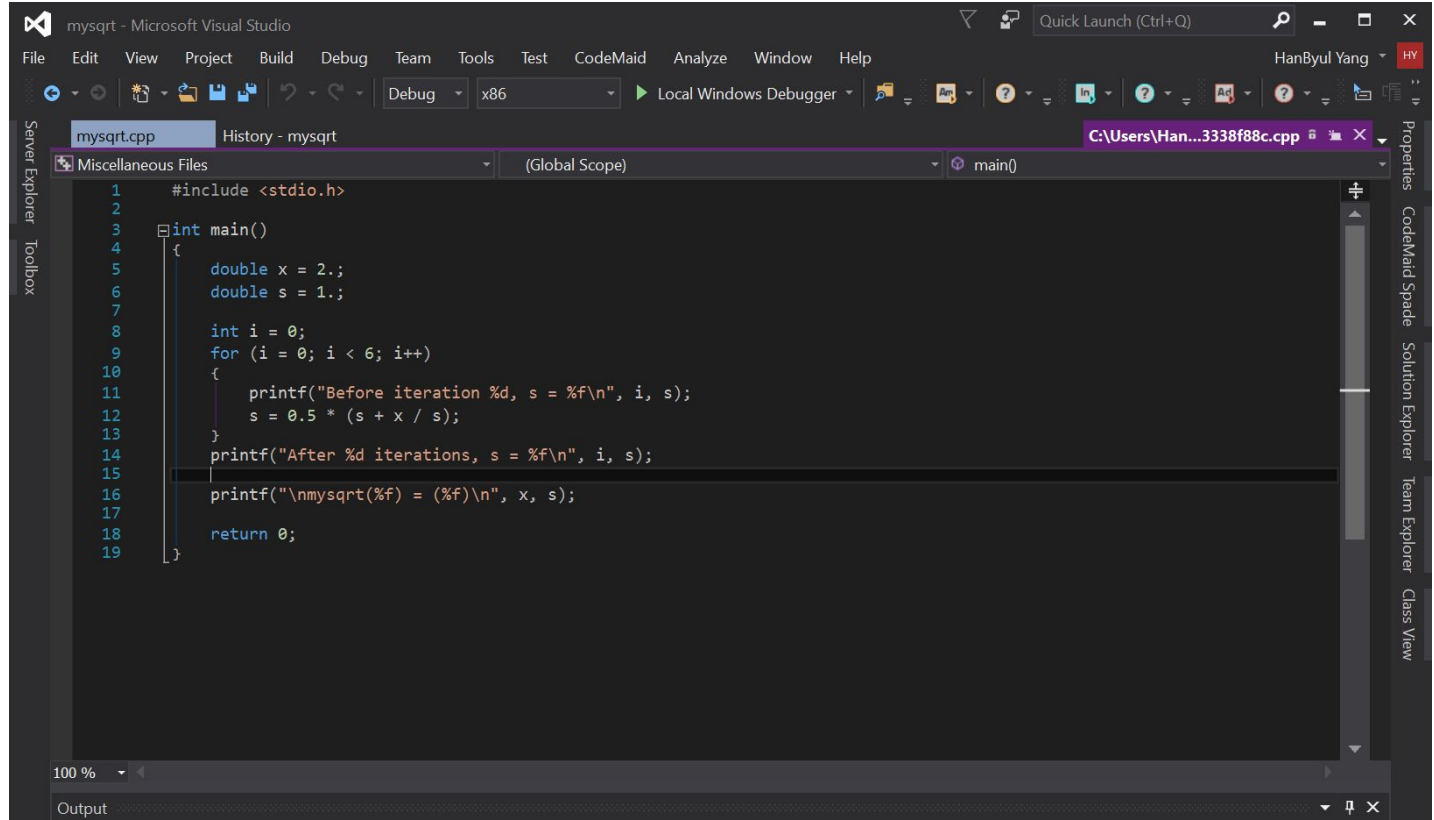
# Demo : First version of mysqlt

```
mysqlt - Microsoft Visual Studio
File Edit View Project Build Debug Team Tools Test CodeMaid Analyze Window Help
Debug x86 Local Windows Debugger
mysqlt.cpp History - mysqlt History - mysqlt C:\Users\Ha...ef292399.cpp
Miscellaneous Files (Global Scope) main()
1 #include <stdio.h>
2
3 int main()
4 {
5     double x = 2.;
6     double s = 1.;
7
8     for (int i = 0; i < 6; i++)
9     {
10         s = 0.5 * (s + x / s);
11     }
12     printf("mysqlt(%f) = (%f)\n", x, s);
13
14     return 0;
15 }
```

100 %

Output

# Demo : Print each iteration

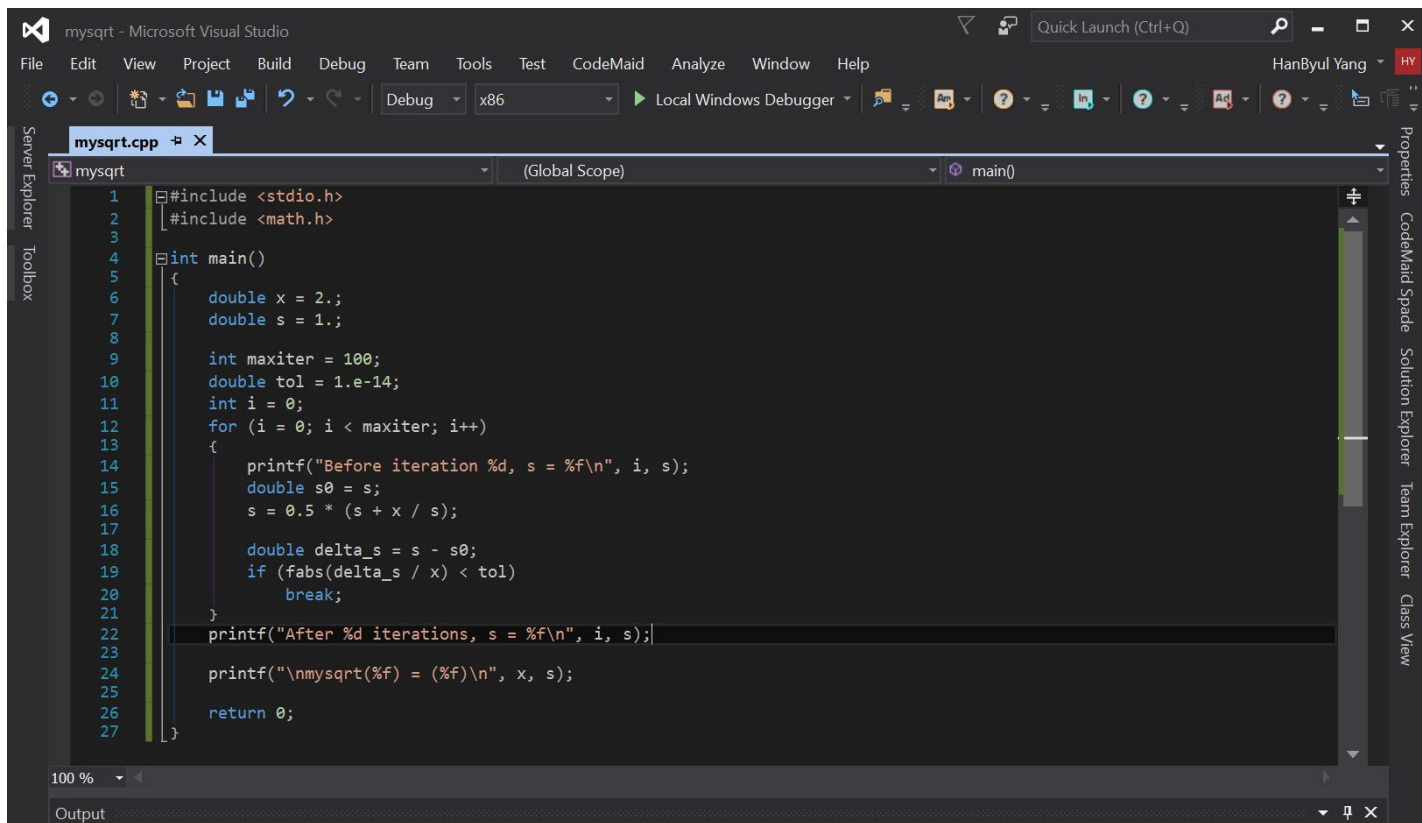


The screenshot shows the Microsoft Visual Studio IDE with a C++ file named `mysqrt.cpp` open. The code implements a function to calculate the square root of 2 using the Newton-Raphson method. The program prints the value of `s` before and after each iteration of the loop.

```
1  #include <stdio.h>
2
3  int main()
4  {
5      double x = 2.;
6      double s = 1.;
7
8      int i = 0;
9      for (i = 0; i < 6; i++)
10     {
11         printf("Before iteration %d, s = %f\n", i, s);
12         s = 0.5 * (s + x / s);
13     }
14     printf("After %d iterations, s = %f\n", i, s);
15
16     printf("\nmysqrt(%f) = (%f)\n", x, s);
17
18     return 0;
19 }
```

The output window at the bottom of the IDE is currently empty, showing the text "Output".

# Demo : Add convergence test

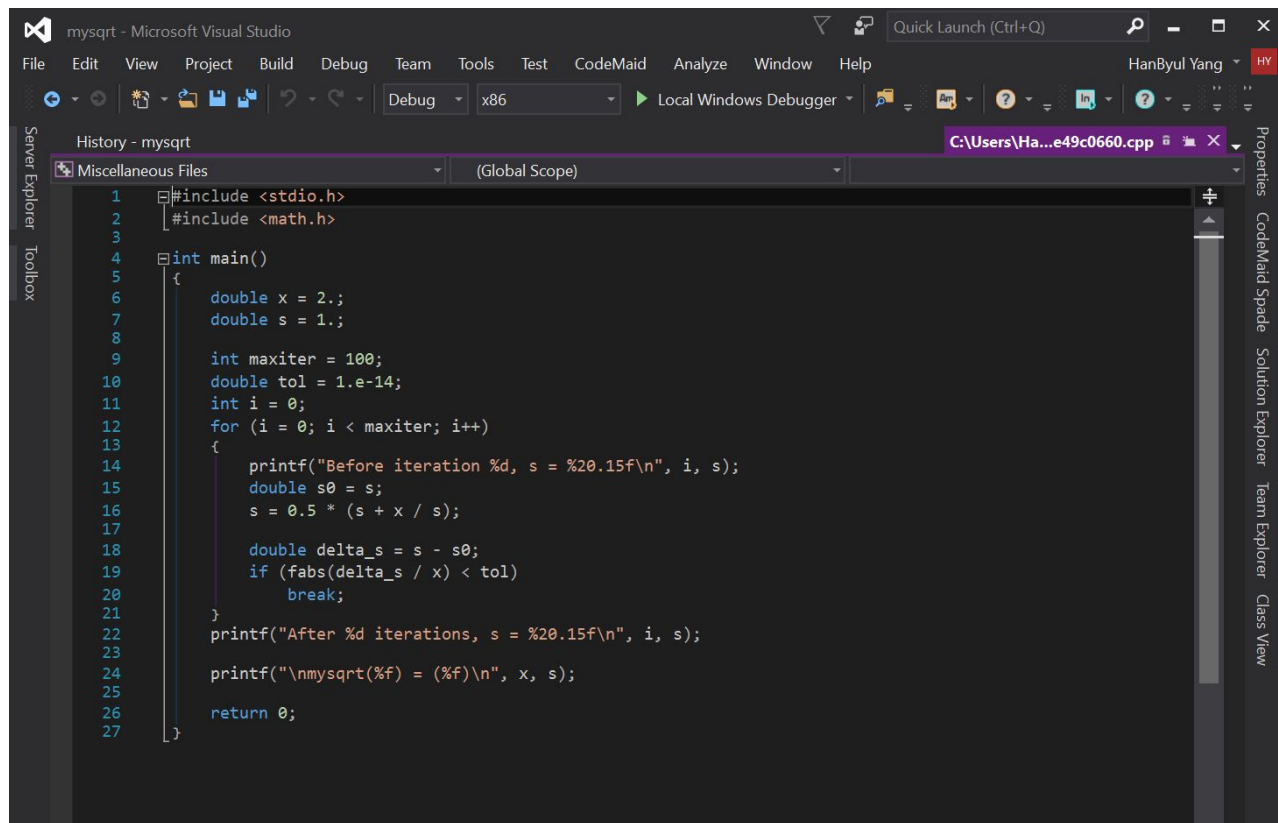


The screenshot shows the Microsoft Visual Studio IDE with a C++ project named 'mysqrt'. The code in 'mysqrt.cpp' implements a Newton-Raphson method for finding the square root of a number. It includes a convergence test that stops the iteration when the relative error is small enough.

```
1  #include <stdio.h>
2  #include <math.h>
3
4  int main()
5  {
6      double x = 2.;
7      double s = 1.;
8
9      int maxiter = 100;
10     double tol = 1.e-14;
11     int i = 0;
12     for (i = 0; i < maxiter; i++)
13     {
14         printf("Before iteration %d, s = %f\n", i, s);
15         double s0 = s;
16         s = 0.5 * (s + x / s);
17
18         double delta_s = s - s0;
19         if (fabs(delta_s / x) < tol)
20             break;
21     }
22     printf("After %d iterations, s = %f\n", i, s);
23
24     printf("\nmysqrt(%f) = (%f)\n", x, s);
25
26     return 0;
27 }
```



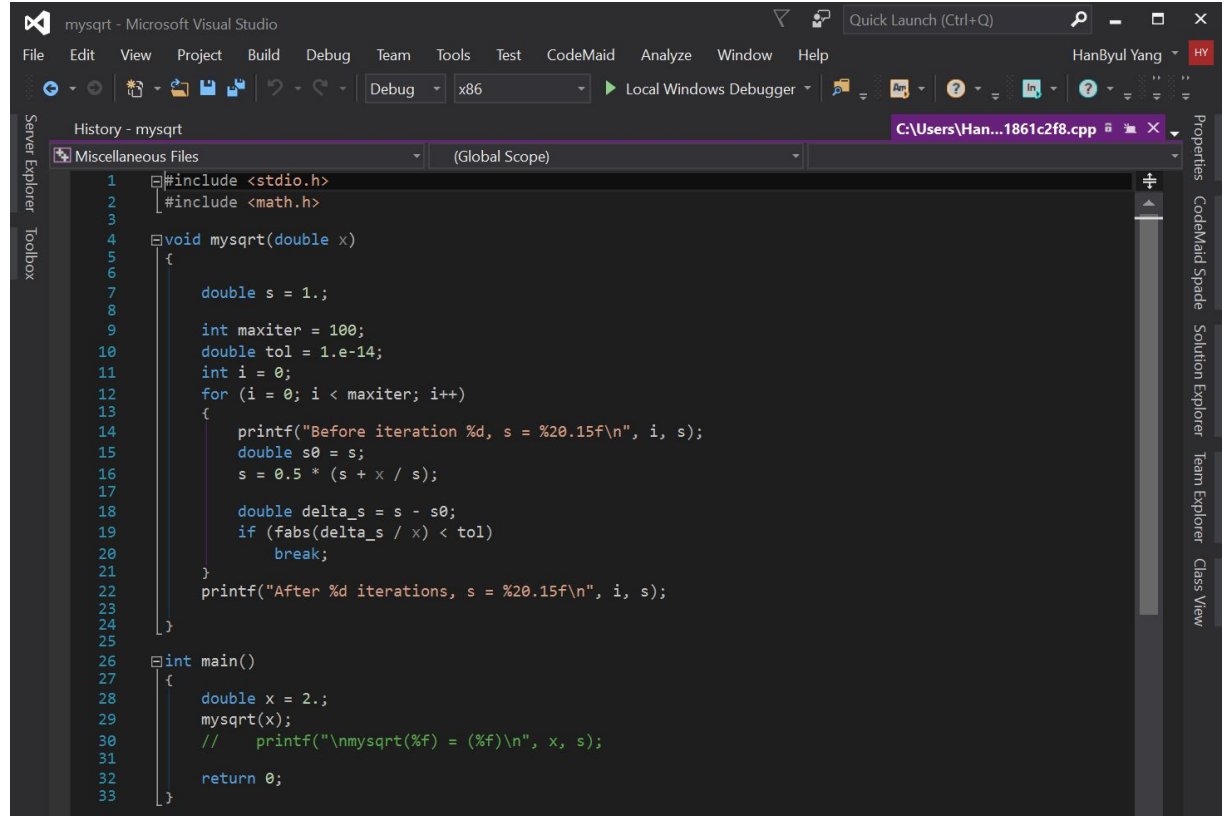
# Demo : Refinement of printing floating point number



The screenshot shows the Microsoft Visual Studio IDE with a C++ project named 'mysqrt'. The code is written in a dark-themed editor and implements a function to calculate the square root of a number using the Newton-Raphson method. The code includes standard headers, defines constants for maximum iterations and tolerance, and uses `printf` to display the results at each iteration and the final result.

```
1  #include <stdio.h>
2  #include <math.h>
3
4  int main()
5  {
6      double x = 2.;
7      double s = 1.;
8
9      int maxiter = 100;
10     double tol = 1.e-14;
11     int i = 0;
12     for (i = 0; i < maxiter; i++)
13     {
14         printf("Before iteration %d, s = %20.15f\n", i, s);
15         double s0 = s;
16         s = 0.5 * (s + x / s);
17
18         double delta_s = s - s0;
19         if (fabs(delta_s / x) < tol)
20             break;
21     }
22     printf("After %d iterations, s = %20.15f\n", i, s);
23
24     printf("\nmysqrt(%f) = (%f)\n", x, s);
25
26     return 0;
27 }
```

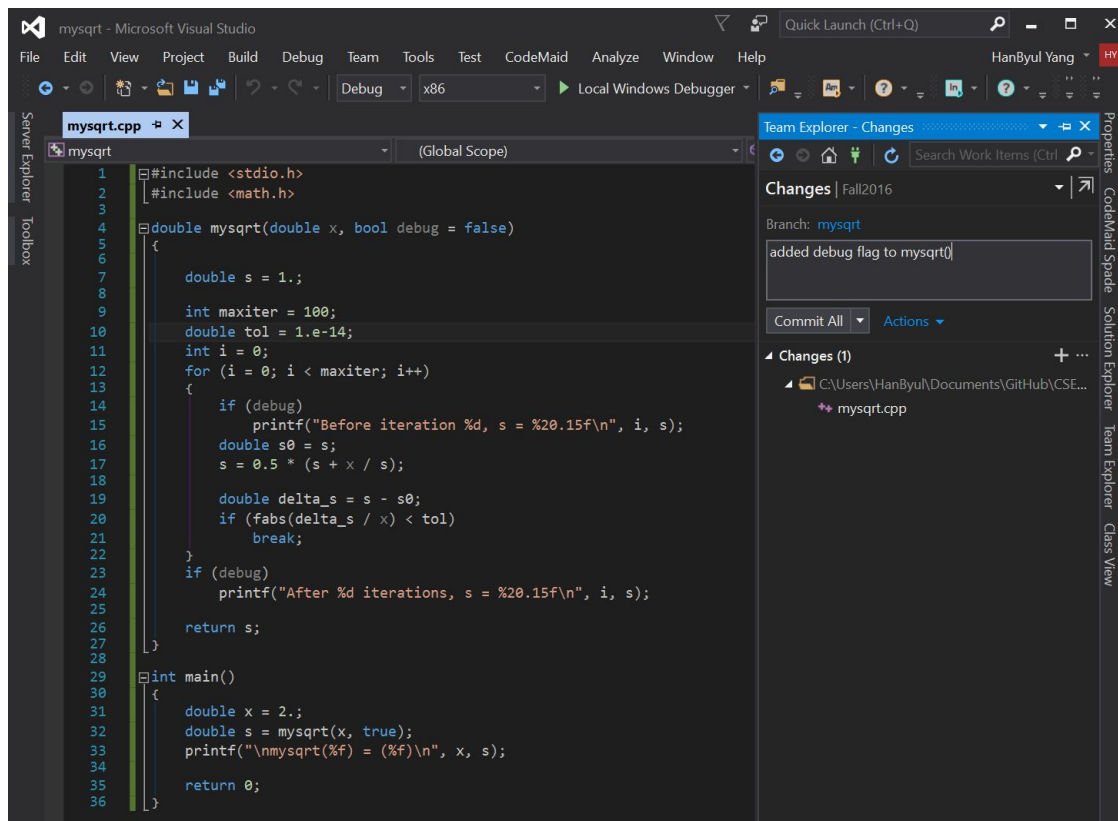
# Demo : mysqrt() function



The screenshot shows the Microsoft Visual Studio IDE with a C++ project named 'mysqrt'. The code is written in a dark-themed editor. The file explorer on the left shows 'Miscellaneous Files' and '(Global Scope)'. The code defines a 'mysqrt' function that uses the Newton-Raphson method to calculate the square root of a number 'x'. It includes headers for <stdio.h> and <math.h>. The function 'mysqrt' takes a 'double x' as input and returns a 'double'. It initializes 's' to 1.0, sets a maximum number of iterations to 100, and a tolerance to 1.e-14. It then enters a loop where it calculates the next value of 's' using the formula  $s = 0.5 * (s + x / s)$  and checks if the absolute relative error is less than the tolerance. The 'main' function calls 'mysqrt' with 'x = 2.0' and prints the result.

```
1 #include <stdio.h>
2 #include <math.h>
3
4 void mysqrt(double x)
5 {
6     double s = 1.;
7
8     int maxiter = 100;
9     double tol = 1.e-14;
10    int i = 0;
11    for (i = 0; i < maxiter; i++)
12    {
13        printf("Before iteration %d, s = %20.15f\n", i, s);
14        double s0 = s;
15        s = 0.5 * (s + x / s);
16
17        double delta_s = s - s0;
18        if (fabs(delta_s / x) < tol)
19            break;
20    }
21    printf("After %d iterations, s = %20.15f\n", i, s);
22
23 }
24
25
26 int main()
27 {
28     double x = 2.;
29     mysqrt(x);
30     // printf("\nmysqrt(%f) = (%f)\n", x, s);
31
32     return 0;
33 }
```

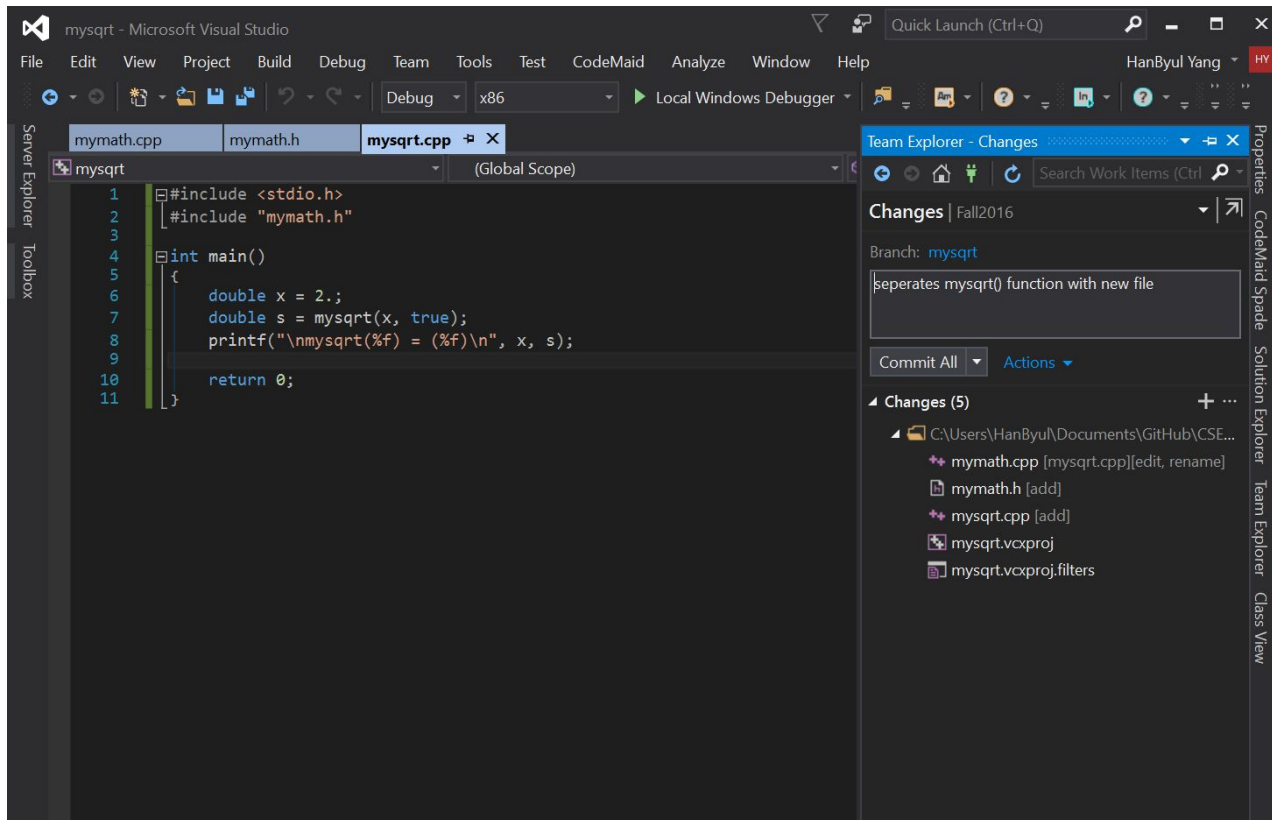
# Demo : mysqrt() function with debug flag



# Demo : separates mysqrt()

## New files

- mymath.h
- mymath.cpp



# Links

- [Getting Started with C++ in Visual Studio](#)
- [Visual C++ in Visual Studio 2015](#)
- [C++ Language Reference](#)