Embedded computing for scientific and industrial imaging applications

Lecture 5 - Visual studio, Newton method,
Binary storage, Integer, floating point number

Microsoft Visual Studio

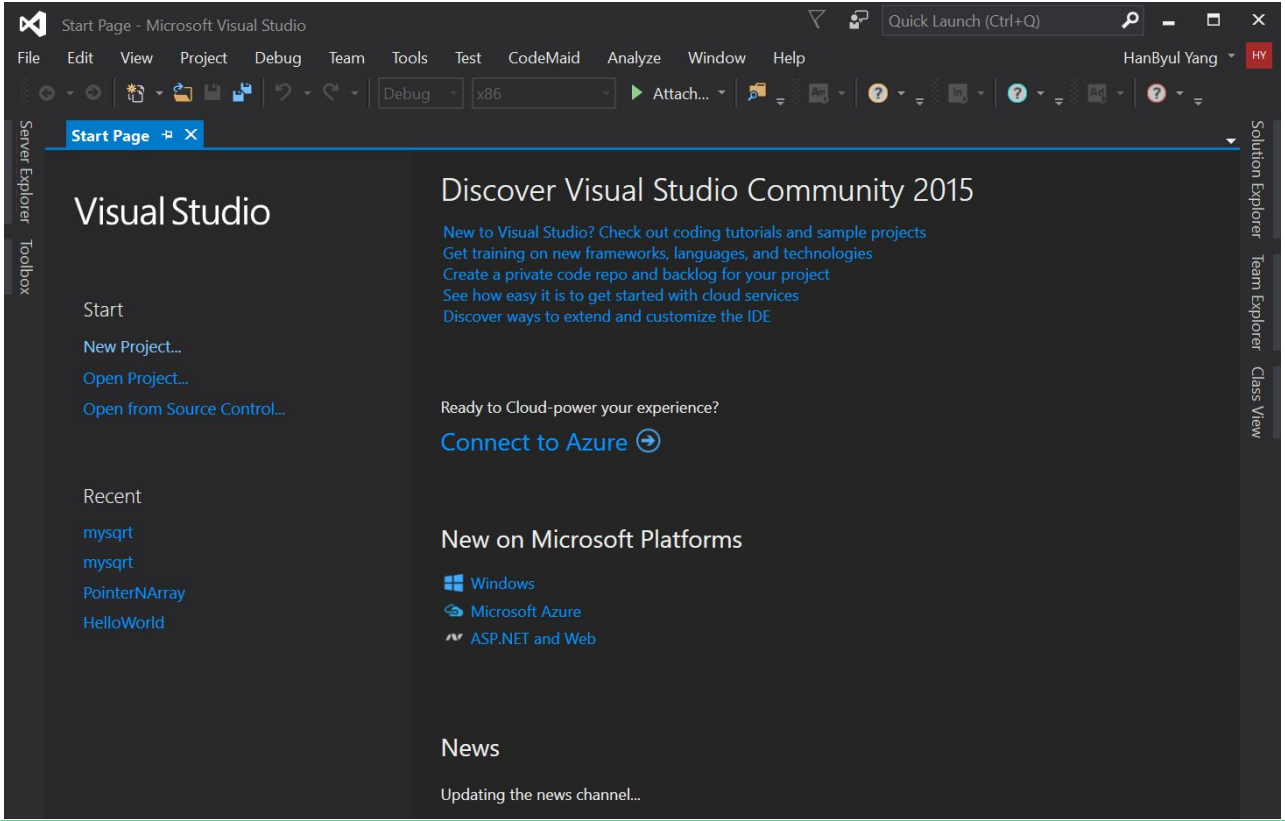
- Used to develop computer program for MS Windows
- IDE - Integrated development environment
  - Code Editor - syntax highlighting, code completion (InteliSense)
  - Debugger - breakpoints, step over
  - Version control system
- Support languages
  - C/C++
  - C++/CLI, Visual C++
  - Visual Basic, Visual Basic .NET
  - C#, F#
  - Python, Ruby, Javascript
- Community edition - free for all users.

Microsoft Visual Studio - terminology

- Solution
- Project
- Properties Editor
- Solution Explorer
- Team Explorer
- Watch
- Call stack
- Output

Demo : visual studio

Start “New Project”

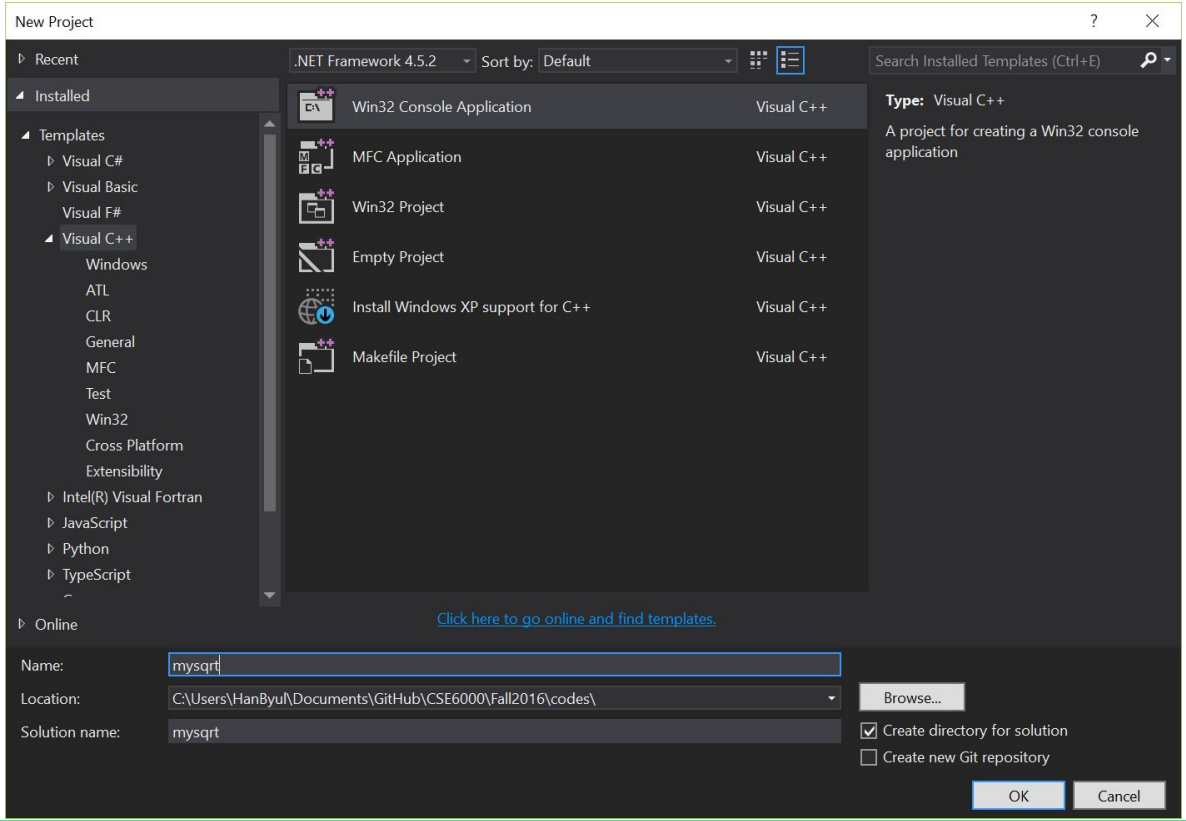


Demo : new project

Visual c++

Win32 Console application

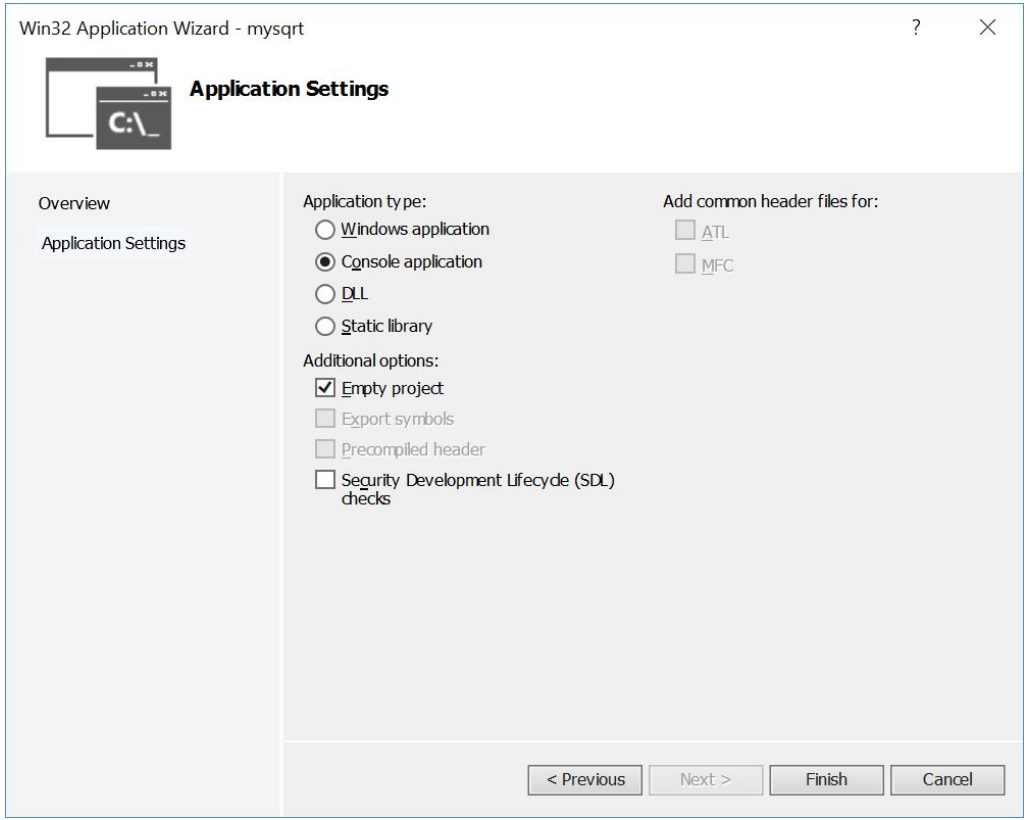
“mysqrt”



Demo : application setting

Console application

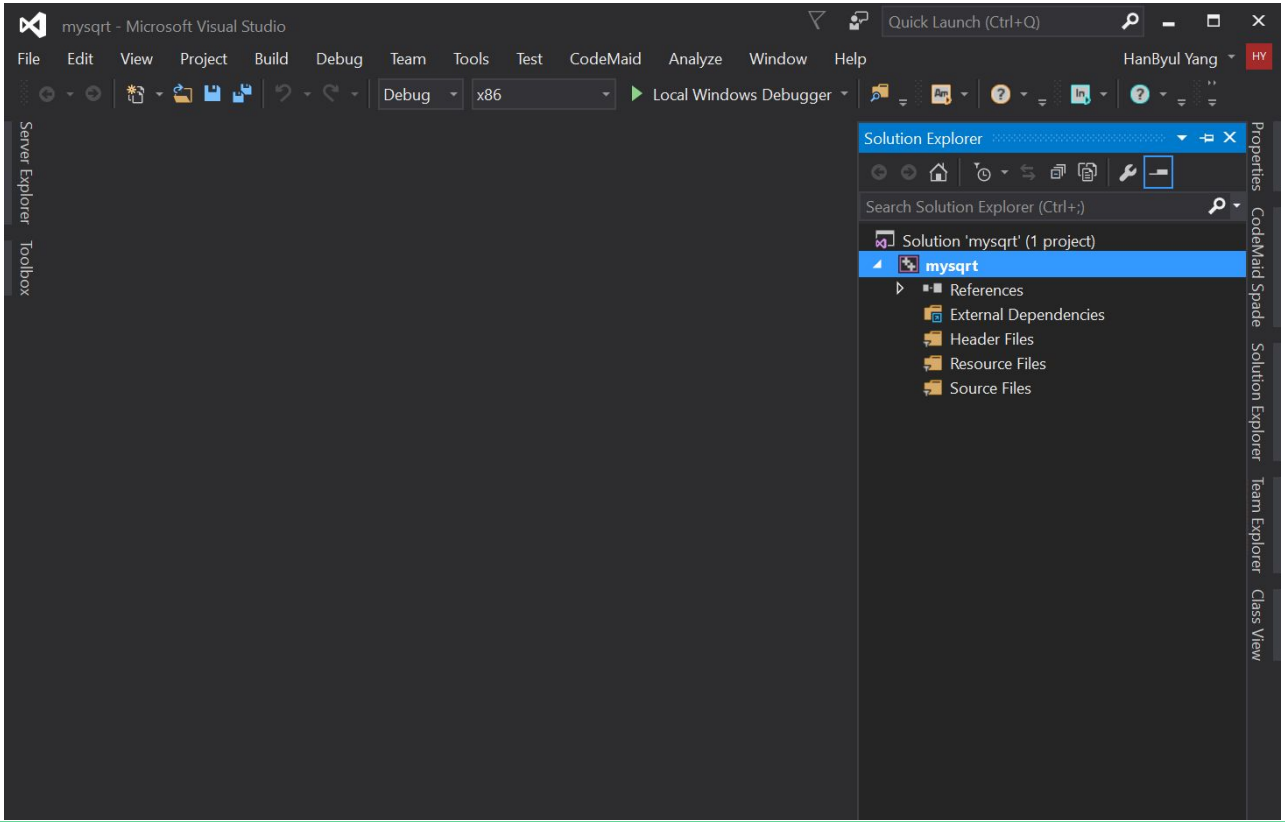
Empty project



Demo : Solution Explorer

Solution

Project

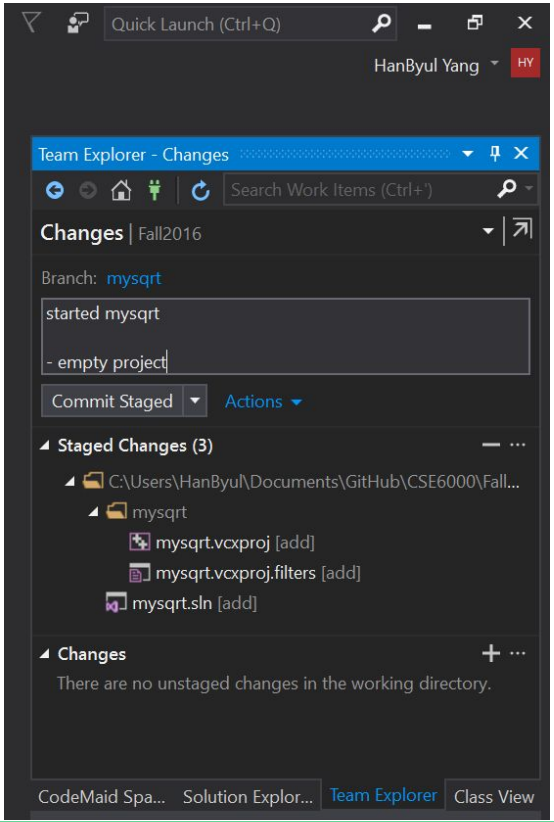


Demo : Team Explorer

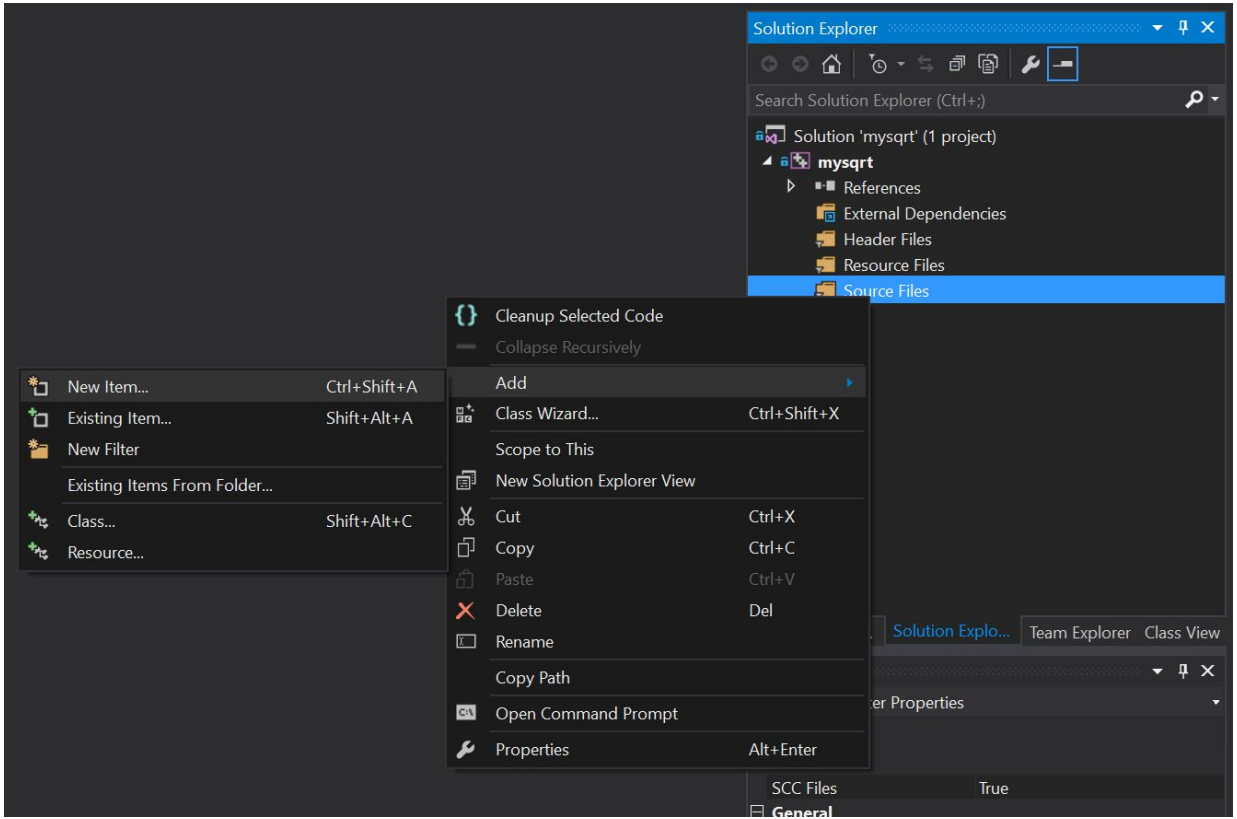
Branch : mysqrt

Commit 3 files

- mysqrt.sln
- mysqrt.vcxproj
- mysqrt.vcxproj.filters



Add “New Item”


Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'mysqrt' (1 project)
mysqrt
References
External Dependencies
Header Files
Resource Files
Source Files
Cleanup Selected Code
Collapse Recursively
Add
Class Wizard... Ctrl+Shift+X
Scope to This
New Solution Explorer View
Cut Ctrl+X
Copy Ctrl+C
Paste Ctrl+V
Delete Del
Rename
Copy Path
Open Command Prompt
Properties Alt+Enter
New Item... Ctrl+Shift+A
Existing Item... Shift+Alt+A
New Filter
Existing Items From Folder...
Class... Shift+Alt+C
Resource...
Solution Explo...
Team Explorer
Class View
er Properties
SCC Files True
General

Demo : First version of mysqrt

mysqrt - Microsoft Visual Studio

File Edit View Project Build Debug Team Tools Test CodeMaid Analyze Window Help

Debug x86 Local Windows Debugger

mysqrt.cpp History - mysqrt History - mysqrt

Miscellaneous Files (Global Scope) main()

```
#include <stdio.h>

int main()
{
    double x = 2.;
    double s = 1.;

    for (int i = 0; i < 6; i++)
    {
        s = 0.5 * (s + x / s);
    }
    printf("mysqrt(%f) = (%f)\n", x, s);

    return 0;
}
```

100 %

Output

Demo : Print each iteration

mysqrt - Microsoft Visual Studio

File Edit View Project Build Debug Team Tools Test CodeMaid Analyze Window Help

Debug x86 Local Windows Debugger

mysqrt.cpp History - mysqrt

Miscellaneous Files (Global Scope) main()

```c
#include <stdio.h>

int main()
{
    double x = 2.;
    double s = 1.;

    int i = 0;
    for (i = 0; i < 6; i++)
    {
        printf("Before iteration %d, s = %f\n", i, s);
        s = 0.5 * (s + x / s);
    }
    printf("After %d iterations, s = %f\n", i, s);

    printf("\nmysqrt(%f) = (%f)\n", x, s);

    return 0;
}
```

100 %

Output

Demo : Breakpoint and debugging

Demo : Add convergence test

mysqrt - Microsoft Visual Studio

File Edit View Project Build Debug Team Tools Test CodeMaid Analyze Window Help

Debug x86 Local Windows Debugger

mysqrt.cpp

mysqrt (Global Scope) main()

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

int main()
{
    double x = 2.;
    double s = 1.;

    int maxiter = 100;
    double tol = 1.e-14;
    int i = 0;
    for (i = 0; i < maxiter; i++)
    {
        printf("Before iteration %d, s = %f\n", i, s);
        double s0 = s;
        s = 0.5 * (s + x / s);

        double delta_s = s - s0;
        if (fabs(delta_s / x) < tol)
            break;
    }
    printf("After %d iterations, s = %f\n", i, s);

    printf("\nmysqrt(%f) = (%f)\n", x, s);

    return 0;
}
```

100 %

Output

Demo : Refinement of printing floating point number

mysqrt - Microsoft Visual Studio

File Edit View Project Build Debug Team Tools Test CodeMaid Analyze Window Help

Debug x86 Local Windows Debugger

History - mysqrt

C:\Users\Ha...e49c0660.cpp

Miscellaneous Files (Global Scope)

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

int main()
{
    double x = 2.;
    double s = 1.;

    int maxiter = 100;
    double tol = 1.e-14;
    int i = 0;
    for (i = 0; i < maxiter; i++)
    {
        printf("Before iteration %d, s = %20.15f\n", i, s);
        double s0 = s;
        s = 0.5 * (s + x / s);

        double delta_s = s - s0;
        if (fabs(delta_s / x) < tol)
            break;
    }
    printf("After %d iterations, s = %20.15f\n", i, s);

    printf("\nmysqrt(%f) = (%f)\n", x, s);

    return 0;
}
```

Demo : mysqrt() function

mysqrt - Microsoft Visual Studio

File Edit View Project Build Debug Team Tools Test CodeMaid Analyze Window Help

Debug x86 Local Windows Debugger

History - mysqrt

C:\Users\Han...1861c2f8.cpp

Miscellaneous Files (Global Scope)

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

void mysqrt(double x)
{

    double s = 1.;

    int maxiter = 100;
    double tol = 1.e-14;
    int i = 0;
    for (i = 0; i < maxiter; i++)
    {
        printf("Before iteration %d, s = %20.15f\n", i, s);
        double s0 = s;
        s = 0.5 * (s + x / s);

        double delta_s = s - s0;
        if (fabs(delta_s / x) < tol)
            break;
    }
    printf("After %d iterations, s = %20.15f\n", i, s);

}

int main()
{
    double x = 2.;
    mysqrt(x);
    //    printf("\nmysqrt(%f) = (%f)\n", x, s);

    return 0;
}
```

Demo : mysqrt() function with debug flag

mysqrt - Microsoft Visual Studio

File Edit View Project Build Debug Team Tools Test CodeMaid Analyze Window Help

Debug x86 Local Windows Debugger

mysqrt.cpp

mysqrt (Global Scope)

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

double mysqrt(double x, bool debug = false)
{

    double s = 1.;

    int maxiter = 100;
    double tol = 1.e-14;
    int i = 0;
    for (i = 0; i < maxiter; i++)
    {
        if (debug)
            printf("Before iteration %d, s = %20.15f\n", i, s);
        double s0 = s;
        s = 0.5 * (s + x / s);

        double delta_s = s - s0;
        if (fabs(delta_s / x) < tol)
            break;
    }
    if (debug)
        printf("After %d iterations, s = %20.15f\n", i, s);

    return s;
}

int main()
{
    double x = 2.;
    double s = mysqrt(x, true);
    printf("\nmysqrt(%f) = (%f)\n", x, s);

    return 0;
}
```

Team Explorer - Changes

Changes | Fall2016

Branch: mysqrt

added debug flag to mysqrt()

Commit All Actions

Changes (1)

C:\Users\HanByul\Documents\GitHub\CSE...

mysqrt.cpp

Demo : separates mysqrt()

New files

- mymath.h
- mymath.cpp

mysqrt - Microsoft Visual Studio

File Edit View Project Build Debug Team Tools Test CodeMaid Analyze Window Help

Debug x86 Local Windows Debugger

mymath.cpp mymath.h mysqrt.cpp

mysqrt (Global Scope)

```
#include <stdio.h>
#include "mymath.h"

int main()
{
    double x = 2.;
    double s = mysqrt(x, true);
    printf("\nmysqrt(%f) = (%f)\n", x, s);

    return 0;
}
```

Team Explorer - Changes

Changes | Fall2016

Branch: mysqrt

seperates mysqrt() function with new file

Commit All Actions

Changes (5)

C:\Users\HanByul\Documents\GitHub\CSE...

mymath.cpp [mysqrt.cpp][edit, rename]

mymath.h [add]

mysqrt.cpp [add]

mysqrt.vcxproj

mysqrt.vcxproj.filters

Demo : Call stack

Links

- Getting Started with C++ in Visual Studio
- Visual C++ in Visual Studio 2015
- C++ Language Reference

Binary storage,
Integer, floating point number

Big matrix

Recall: Approximating the heat equation on a 100×100 grid gives a linear system with $10,000$ equations, $Au = b$ where the matrix A is $10,000 \times 10,000$.

Question: How much disk space is required to store a $10,000 \times 10,000$ matrix of real numbers?

It depends on how many bytes are used for each real number.

1 byte = 8 bits, bit = “binary digit”

Assuming 8 bytes (64 bits) per value:

A $10,000 \times 10,000$ matrix has 10^8 elements,

so this requires 8×10^8 bytes = 800 MB.

Units

- Kilo = thousand (10^3)
- Mega = million (10^6)
- Giga = billion (10^9)
- Tera = trillion (10^{12})
- Peta = 10^{15}
- Exa = 10^{18}

Computer memory

Memory is subdivided into bytes, consisting of 8 bits each.
One byte can hold $2^8 = 256$ distinct numbers:

```
00000000 = 0
00000001 = 1
00000010 = 2
...
11111111 = 255
```

Might represent integers, characters, colors, etc.
Usually programs involve integers and real numbers that require more than 1 byte to store.

Often 4 bytes (32 bits) or 8 bytes (64 bits) used for each.

Integers

To store integers, need one bit for the sign (+ or −) In one byte this would leave 7 bits for binary digits.
Advantage: Binary addition works directly.

Two's complement representation used:

```
10000000 = -128
10000001 = -127
10000010 = -126
…
11111110 = -2
11111111 = -1
00000000 = 0
00000001 = 1
00000010 = 2
…
01111111 = 127
```

Integers

Integers are typically stored in 4 bytes (32 bits). Values between roughly -2^{31} and 2^{31} can be stored.

Note: special software for arithmetic, may be slower!

```
$ python

    >>> 2**30

    1073741824

    >>> 2**100

    1267650600228229401496703205376L
```

Note L on end!

Fixed point notation

Use, e.g. 64 bits for a real number but always assume N bits in integer part and M bits in fractional part.

Analog in decimal arithmetic, e.g.:

5 digits for integer part and

6 digits in fractional part

Could represent, e.g.:

```
00003.141592 = (pi)
00000.000314 = (pi / 10000)
31415.926535 = (pi * 10000)
```

Disadvantages:

Precision depends on size of number

Often many wasted bits (leading 0's)

Limited range; often scientific problems involve very large or small numbers.

Floating point numbers

Base 10 scientific notation:

$0.2345e{-}18 = 0.2345 \times 10^{-18} = 0.000000000000000002345$

Mantissa: 0.2345, Exponent: −18

Binary floating point numbers:

Example: Mantissa: 0.101101, Exponent: −11011 means:

$$0.101101 = 1(2^{-1}) + 0(2^{-2}) + 1(2^{-3}) + 1(2^{-4}) + 0(2^{-5}) + 1(2^{-6})$$
$$= 0.703125 \text{ (base 10)}$$
$$-11011 = -1(2^{4}) + 1(2^{3}) + 0(2^{2}) + 1(2^{1}) + 1(2^{0})$$
$$= -27 \text{ (base 10)}$$

So the number is

$0.703125 \times 2^{-27} \approx 5.2386894822120667 \times 10^{-9}$

Floating point numbers

Visual C++ double is 8 bytes with IEEE standard representation.

53 bits for mantissa and 11 bits for exponent (64 bits = 8 bytes).
We can store 52 binary bits of precision.

$2^{-52} \approx 2.2 \times 10^{-16}$ ⇒ roughly 15 decimal digits of precision.

Floating point numbers

Roughly 15 decimal digits of precision.

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

int main()
{
    printf("pi = %9.15lf\n", M_PI);
    printf("pi * 1000 = %9.15lf\n", M_PI * 1000.0);
    printf("pi / 1000 = %9.15lf\n", M_PI / 1000.0);
    return 0;
}
```

```
pi = 3.141592653589793
pi * 1000 = 3141.592653589792917
pi / 1000 = 0.003141592653590
```