C++ Basics
Part Three

Sai-Keung Wong

National Yang Ming Chiao Tung University

Hsinchu, Taiwan

Intended Learning Outcomes

- Define a class
- Define the meaning of a member function that has const at its declaration
- Implement functions to show intermediate results for debugging

How to define a class?

```
class CLASS_NAME {
        public:
         CLASS_NAME ( int score ) {
                // this is a constructor, initializing data members.
                this->score = score ;
        }
        void foo( ) {
                // this is a method.
                cout << score << endl;
        }
        public:
        int score;        // this is a data member; declaration
};
```

How to define a class?

```
class CLASS_NAME {
        public:
         CLASS_NAME ( int score ) {
                // this is a constructor, initializing data members.
                this->score = score ; // what does this mean?
        }
        void foo( ) {
                // this is a method.
                cout << score << endl;
        }
        public:
        int score;              // this is a data member; declaration
};
```

The formal parameter: score

How to define a class?

```
class CLASS_NAME {
        public:
         CLASS_NAME ( int score ) {
                // this is a constructor, initializing data members.
                this->score = score ; // what does this mean?
        }
        void foo( ) {
                // this is a method.
                cout << score << endl;
        }
        public:
        int score;          // this is a data member; declaration
};
```

Data member: score

The formal parameter: score

How to define a class?

```
class CLASS_NAME {
        public:
         CLASS_NAME ( int s ) {
                // this is a constructor, initializing data members.
                 score = s;
        }
        void foo( ) {
                // this is a method.
                cout << score << endl;
        }
        public:
        int score;            // this is a data member; declaration
};
```

The parameter name is not the same as the data member name.
Readability problem.

```
CLASS_NAME ( int score ) {
   score = score;   // assign the parameter to itself.
}
```

```
CLASS_NAME ( int score ) {
   this->score = score;   // assign the parameter
                          to data member score.
}
```

How to define a class?

```
class CLASS_NAME {
        public:
         CLASS_NAME ( int score ) {
                // this is a constructor, initializing data members.
                 this->score = score;
        }
        void foo( ) {
                // this is a method.
                cout << score << endl;
        }
        public:
        int score;              // this is a data member; declaration
};
```

(A) A1

(B) A2

(C) A3

How to define a class?

```
class CLASS_NAME {
        public:
         CLASS_NAME ( int s ) {
                // this is a constructor, initializing data members.
                 score = s;
        }
        void foo( ) {
                // this is a method.
                cout << score << endl;
        }
        public:
        int score;          // this is a data member; declaration
};
```

```
void test( ) {
  CLASS_A a(2);
  CLASS_A c(20);

  a.score = 99;
  c.score = 2;

  a.foo( );
  c.foo( );
}
```

What are the outputs?

How to define a class?

```
class CLASS_NAME {
        public:
         CLASS_NAME ( int s ) {
                // this is a constructor, initializing data members.
                 score = s;
        }
        void foo( ) {
                // this is a method.
                cout << score << endl;
        }
        public:
        int score;          // this is a data member; declaration

};
```

```
void test( ) {
  CLASS_A a(2);
  CLASS_A c(20);

  a.score = 99;
  c.score = 2;

  a.foo( );
  c.foo( );
}
```

A1

A2

What are the outputs?

How to define a class?

```
class CLASS_A {
            public:
             CLASS_A( ) { //default constructor; no argument
                        // this is a constructor, initializing data members.
                        this->score =score; // what does this mean?
            }
            void foo( ) {
                        // this is a method.
            }
            public:
            int score;            // this is a data member

};
```

assign the parameter to data member score.

How to define a class?

```
class CLASS_A {
        public:
         CLASS_A( ) { //default constructor; no argument
                // this is a constructor, initializing data members.
                this->score =score; // what does this mean?
        }
        void foo( ) {
                // this is a method.
        }
        public:
        int score;          // this is a data member

};
```

```
void test( ) {
  CLASS_A a(2);
  CLASS_A c(20);

  a.score = 99;
  c.score = 2;

  a.foo( );
  c.foo( );
}
```

Any errors?

How to define a class?

```
class CLASS_A {
        public:
         CLASS_A( ) { //default constructor; no argument
                // this is a constructor, initializing data members.
                this->score =score; // what does this mean?
        }
        void foo( ) {
                // this is a method.
        }
        public:
        int score;              // this is a data member
};
```

```
void test( ) {
  CLASS_A a(2);
  CLASS_A c(20);

  a.score = 99;
  c.score = 2;

  a.foo( );
  c.foo( );
}
```

Any errors?

How to define a class?

```
class CLASS_A {
        public:
         CLASS_A( ) { //default constructor; no argument
                // this is a constructor, initializing data members.
                this->score =score; // what does this mean?
        }
        void foo( ) {
                // this is a method.
        }
        public:
        int score;        // this is a data member
};
```

```
void test( ) {
  CLASS_A a(2);
  CLASS_A c(20);
```

Any errors?

CLASS_A a(2); A1

CLASS_A c(20); A2

// A3 A4

```
  a.foo( );
  c.foo( );
}
```

Exercise

```
class CLASS_A {
        public:
         CLASS_A( int score ) { //
                this->score =score;
        }
        void foo( ) {
                // this is a method.
        }
        public:
        int score;        // this is a data
member
};
```

```
void test( ) {
  CLASS_A a;
  CLASS_A c;
  ……
}
```

Any errors?

Exercise

```
class CLASS_A {
        public:
         CLASS_A( int score ) { //
                this->score =score;
        }
        void foo( ) {
                // this is a method.
        }
        public:
        int score;        // this is a data
member
};
```

```
void test( ) {
  CLASS_A a;
  CLASS_A c;
  ……
}
```

Any errors?

```
CLASS_A a;  // A1
CLASS_A c; // A2
// A3 A4
```

Exercise

```
class CLASS_A {
        public:
        CLASS_A( ) { ……}
        CLASS_A( int score ) { //
                this->score =score;
        }
        void foo( ) {
                // this is a method.
        }
        public:
        int score;          // this is a data member
};
```

```
void test( ) {
  CLASS_A a;
  CLASS_A c;
  ……
}
```

Any errors?

```
CLASS_A a;  // A1
CLASS_A c; // A2
// A3 A4
```

Exercise

```
class CLASS_A {
        public:
        CLASS_A( ) {  // No-arg constructor
           score = 0;
        }
        CLASS_A( int score ) { //
          this->score =score;
        }
        void printf( ) const {
          cout << "score:" << score << endl;
        }
        public:
        int score;   // this is a data member
};
```

```
void test( ) {
  CLASS_A a;
  CLASS_A c;
  CLASS_A d(2);
  CLASS_A e(2023);

  a.printf( );
  c.printf( );
  d.printf( );
  e.printf( );
}
```

Any errors?

No error

Exercise

```
class CLASS_A {
        public:
        CLASS_A( ) {  // No-arg constructor
            score = 0;
        }
        CLASS_A( int score ) { //
           this->score =score;
        }
        void printf( ) const {
           cout << “score:” << score << endl;
        }
        public:
        int score;   // this is a data member
};
```

```
void test( ) {
  CLASS_A a;
  CLASS_A c;
  CLASS_A d(2);
  CLASS_A e(2023);

  a.printf( );        A1
  c.printf( );        A2
  d.printf( );        A3
  e.printf( );        A4
}
```

Any errors?

What are the outputs?

Example One: Quadratic equation solver

- Requirement specification: Write a program to ask the user to input a quadratic equation and solve the quadratic equation.
- If there are real roots, show them. Otherwise, show “no real roots”.

$a\,x^2 + b\,x + c = 0$

root1 = (-b - sqrt(b^2 – 4ac)) / 2a
root2 = (-b + sqrt(b^2 – 4ac)) / 2a

$x^2 + 5\,x + 6 = 0$. a = 1, b = 5, c = 6
root1 = (-5 – sqrt(5*5 – 4*6))/2 = -3
root2 = (-5 + sqrt(5*5 – 4*6))/2 = -2

Example One: Quadratic equation solver

- Requirement specification: Write a program to ask the user to input a quadratic equation and solve the quadratic equation.
- If there are real roots, show them. Otherwise, show “no real roots”.

$a\,x^2 + b\,x + c = 0$

$root1 = (-b - sqrt(\,b^2 – 4ac\,)\,)\,/\,2a$
$root2 = (-b + sqrt(\underline{\,b^2 – 4ac\,})\,)\,/\,2a$

$x^2 + 5\,x + 6 = 0.\ a = 1,\ b = 5,\ c = 6$
$root1 = (-5 – sqrt(\underline{5*5 – 4*6})\,)/2 = -3$
$root2 = (-5 + sqrt(\underline{5*5 – 4*6})\,)/2 = -2$

Example One: Quadratic equation solver

- Requirement specification: Write a program to ask the user to input a quadratic equation and solve the quadratic equation.
- If there are real roots, show them. Otherwise, show “no real roots”.

$a\,x^2 + b\,x + c = 0$

root1 = $(-b - \mathrm{sqrt}(\,b^2 – 4ac\,)\,)\,/\,2a$
root2 = $(-b + \mathrm{sqrt}(\,b^2 – 4ac\,)\,)\,/\,2a$

Determinant = $b^2 – 4ac$

```
void q_solver( ) {
   double a, b, c;
   cin >> a >> b >> c;
   double d2;            // determinant
   d2 = b*b – 4*a*c;

   if (d2 < 0) {
      cout << “no real root” << endl;
      return;
   }
   double r1 = ( - b - sqrt(d2))/(2*a);
   double r2 = ( - b + sqrt(d2))/(2*a);
   cout << “root:” << r1 << “\t” << r2 << endl;
}
```

Example One: Quadratic equation solver

- Requirement specification: Write a program to ask the user to input a quadratic equation and solve the quadratic equation.
- If there are real roots, show them. Otherwise, show “no real roots”.

$$a x^2 + b x + c = 0$$

$$root1 = (-b - sqrt(b^2 – 4ac)) / 2a$$
$$root2 = (-b + sqrt(b^2 – 4ac)) / 2a$$

$$Determinant = b^2 – 4ac$$

input

compute determinant

if there is no real root, show a message and return

compute the two roots

output

```
void q_solver( ) {
  double a, b, c;
  cin >> a >> b >> c;
  double d2;          // determinant
  d2 = b*b – 4*a*c;

  if (d2 < 0) {
    cout << “no real root” << endl;
    return;
  }
  double r1 = ( - b - sqrt(d2))/(2*a);
  double r2 = ( - b + sqrt(d2))/(2*a);
  cout << “root:” << r1 << “\t” << r2 << endl;
}
```

Example One: Quadratic equation solver

- Requirement specification: Write a program to ask the user to input a quadratic equation and solve the quadratic equation.
- If there are real roots, show them. Otherwise, show “no real roots”.

$a\,x^2 + b\,x + c = 0$

root1 = $(-b - \mathrm{sqrt}(\,b^2 - 4ac\,)\,) / 2a$
root2 = $(-b + \mathrm{sqrt}(\,b^2 - 4ac\,)\,) / 2a$

Determinant = $b^2 - 4ac$

input {

compute determinant {

if there is no real root, show a message and return {

compute the two roots {

output {

Example One: Quadratic equation solver

- Requirement specification: Write a program to ask the user to input a quadratic equation and solve the quadratic equation.
- If there are real roots, show them. Otherwise, show “no real roots”.

$a x^2 + b x + c = 0$

$root1 = (-b - sqrt(b^2 – 4ac)) / 2a$
$root2 = (-b + sqrt(b^2 – 4ac)) / 2a$

$Determinant = b^2 – 4ac$

input {

compute determinant {

if there is no real root, show a message and return {

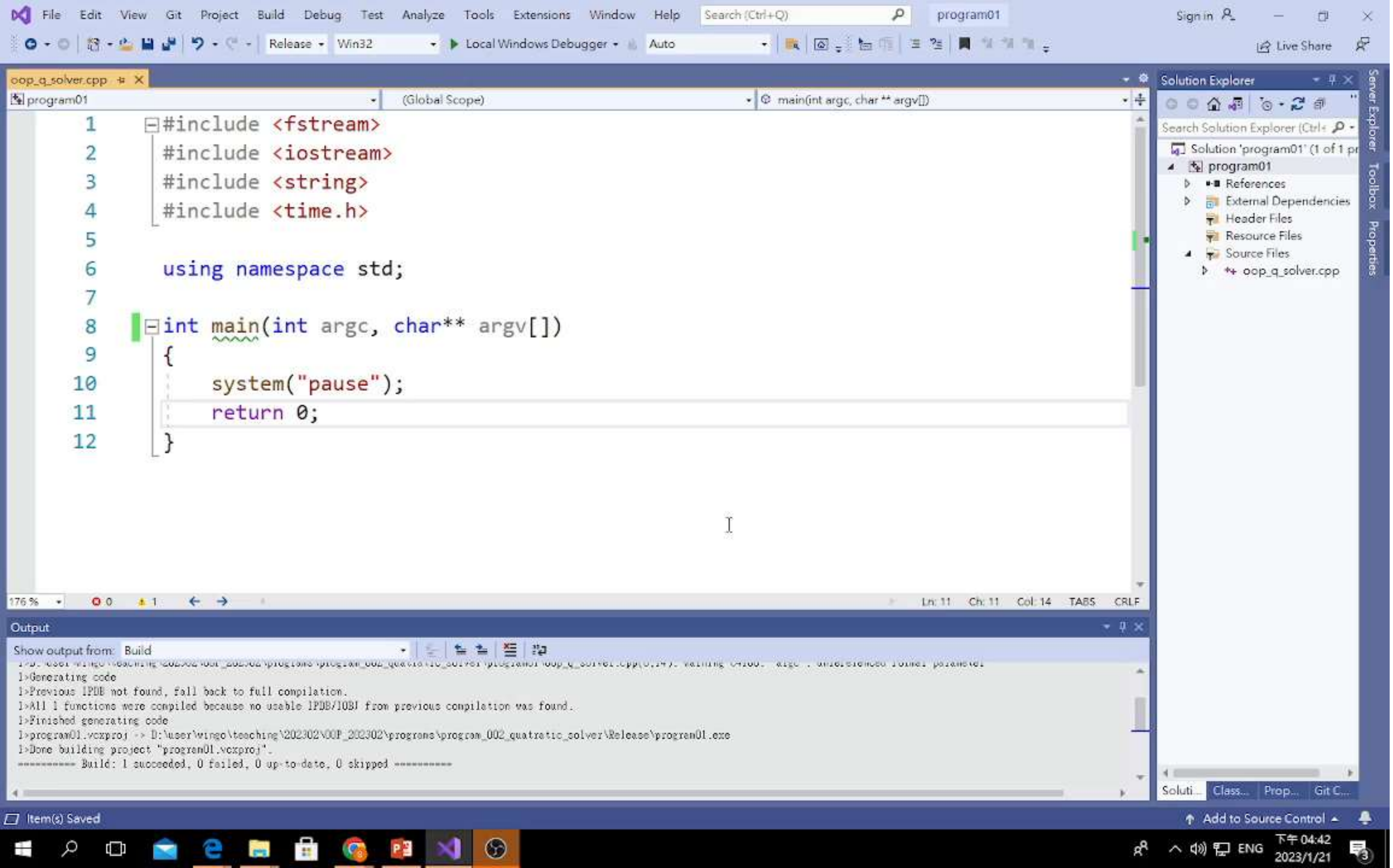
compute the two roots {

output {

```
void q_solver( ) {
   double a, b, c;
   cin >> a >> b >> c;
   double d2;            // determinant
   d2 = b*b – 4*a*c;

   if (d2 < 0) {
      cout << “no real root” << endl;
      return;
   }
   double r1 = ( - b - sqrt(d2))/(2*a);
   double r2 = ( - b + sqrt(d2))/(2*a);
   cout << “root:” << r1 << “\t” << r2 << endl;
}
```

~01:03

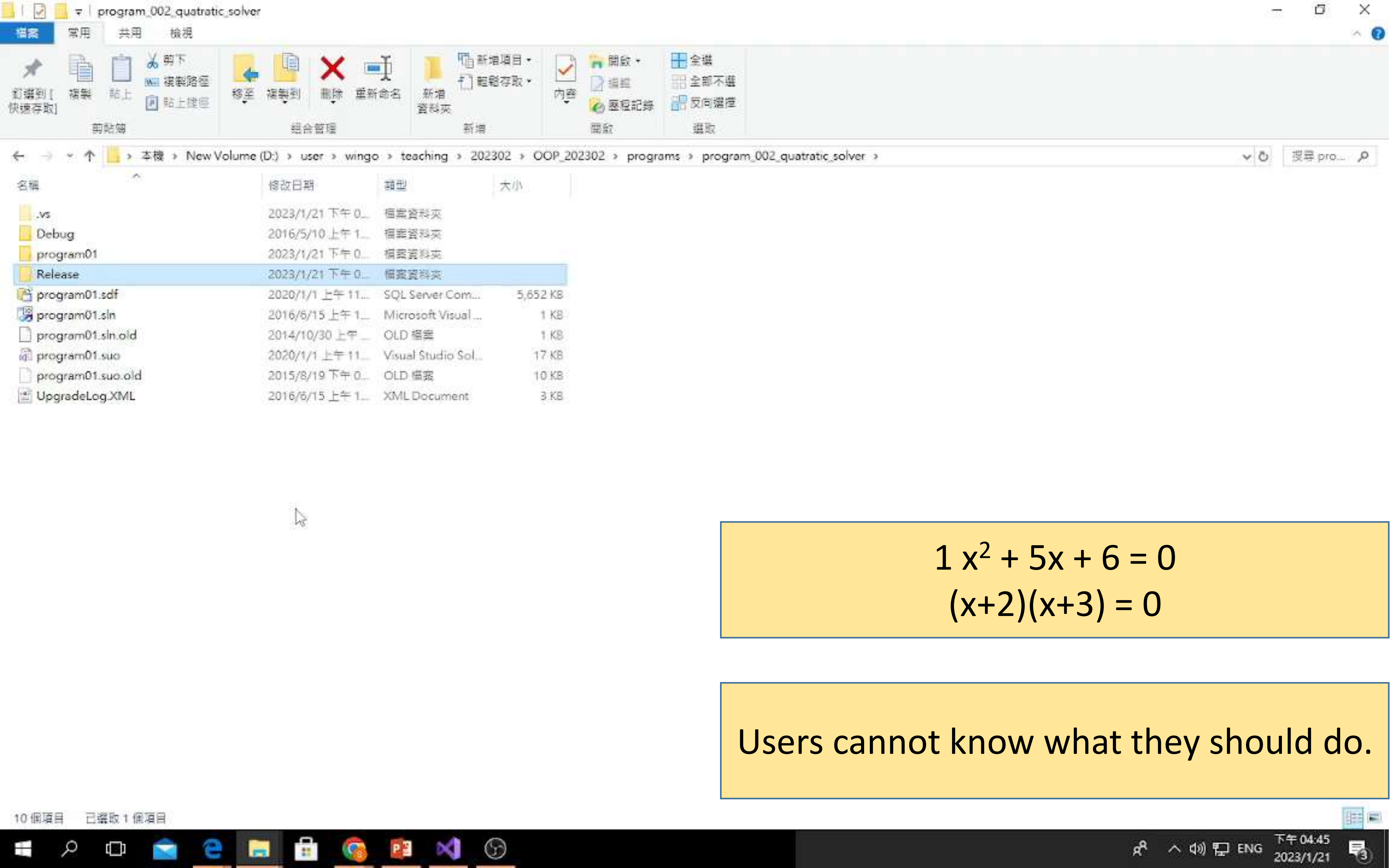


```cpp
#include <fstream>
#include <iostream>
#include <string>
#include <time.h>

using namespace std;

int main(int argc, char** argv[])
{
	system("pause");
	return 0;
}
```

```
1>Generating code
1>Previous IPDB not found, fall back to full compilation.
1>All 1 functions were compiled because no usable IPDB/IOBJ from previous compilation was found.
1>Finished generating code
1>program01.vcxproj -> D:\user\wingo\teaching\202302\OOP_202302\programs\program_002_quatratic_solver\Release\program01.exe
1>Done building project "program01.vcxproj".
========== Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped ==========
```



program_002_quatratic_solver

本機 › New Volume (D:) › user › wingo › teaching › 202302 › OOP_202302 › programs › program_002_quatratic_solver ›

名稱	修改日期	類型	大小
.vs	2023/1/21 下午 0...	檔案資料夾	
Debug	2016/5/10 上午 1...	檔案資料夾	
program01	2023/1/21 下午 0...	檔案資料夾	
Release	2023/1/21 下午 0...	檔案資料夾	
program01.sdf	2020/1/1 上午 11...	SQL Server Com...	5,652 KB
program01.sln	2016/6/15 上午 1...	Microsoft Visual ...	1 KB
program01.sln.old	2014/10/30 上午 ...	OLD 檔案	1 KB
program01.suo	2020/1/1 上午 11...	Visual Studio Sol...	17 KB
program01.suo.old	2015/8/19 下午 0...	OLD 檔案	10 KB
UpgradeLog.XML	2016/6/15 上午 1...	XML Document	3 KB

10 個項目　已選取 1 個項目

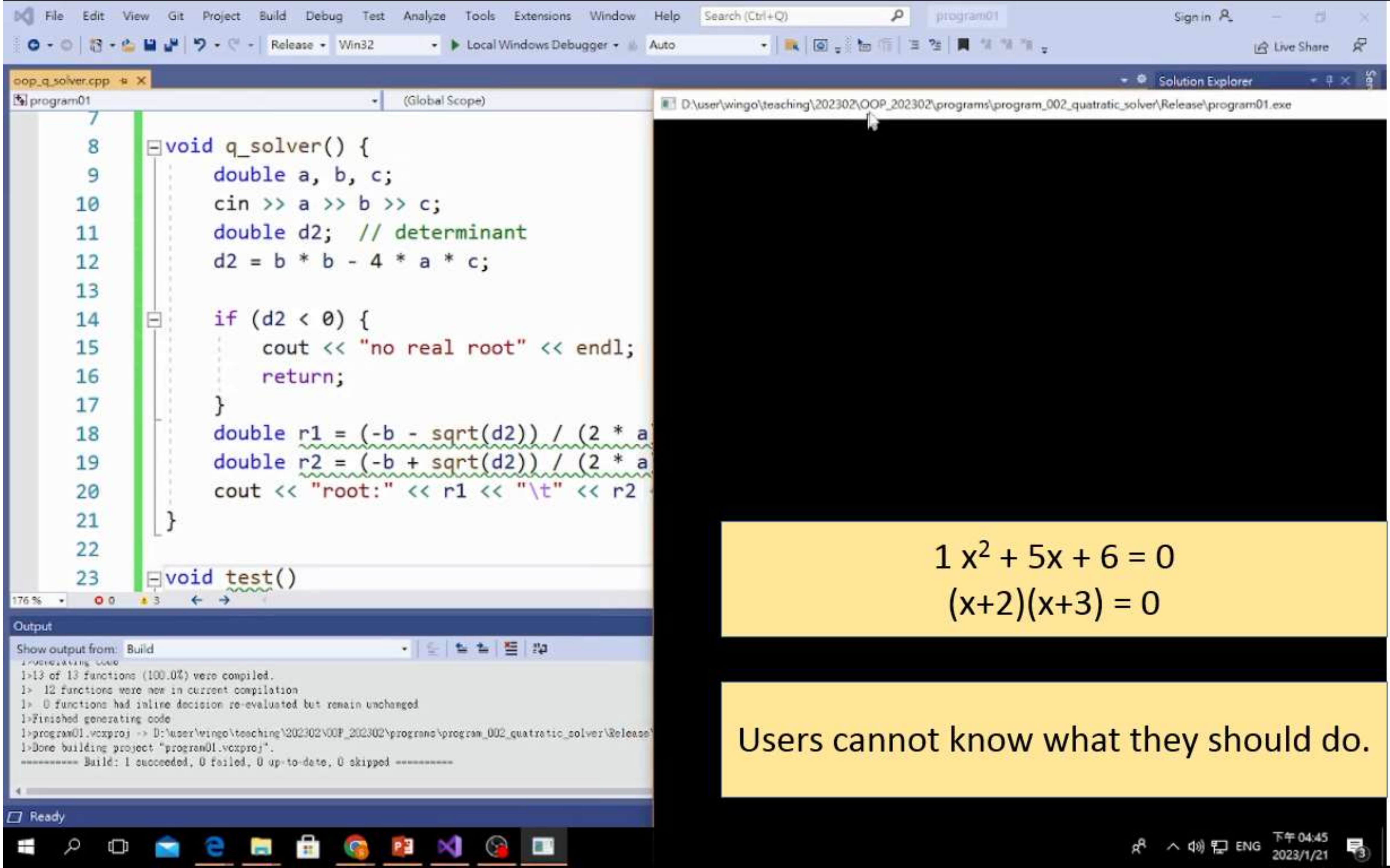
ENG 下午 04:45 2023/1/21

~00:33

$$1\,x^2 + 5x + 6 = 0$$
$$(x+2)(x+3) = 0$$

Users cannot know what they should do.

~00:33



D:\user\wingo\teaching\202302\OOP_202302\programs\program_002_quatratic_solver\Release\program01.exe

```cpp
void q_solver() {
    double a, b, c;
    cin >> a >> b >> c;
    double d2;  // determinant
    d2 = b * b - 4 * a * c;

    if (d2 < 0) {
        cout << "no real root" << endl;
        return;
    }
    double r1 = (-b - sqrt(d2)) / (2 * a
    double r2 = (-b + sqrt(d2)) / (2 * a
    cout << "root:" << r1 << "\t" << r2
}

void test()
```

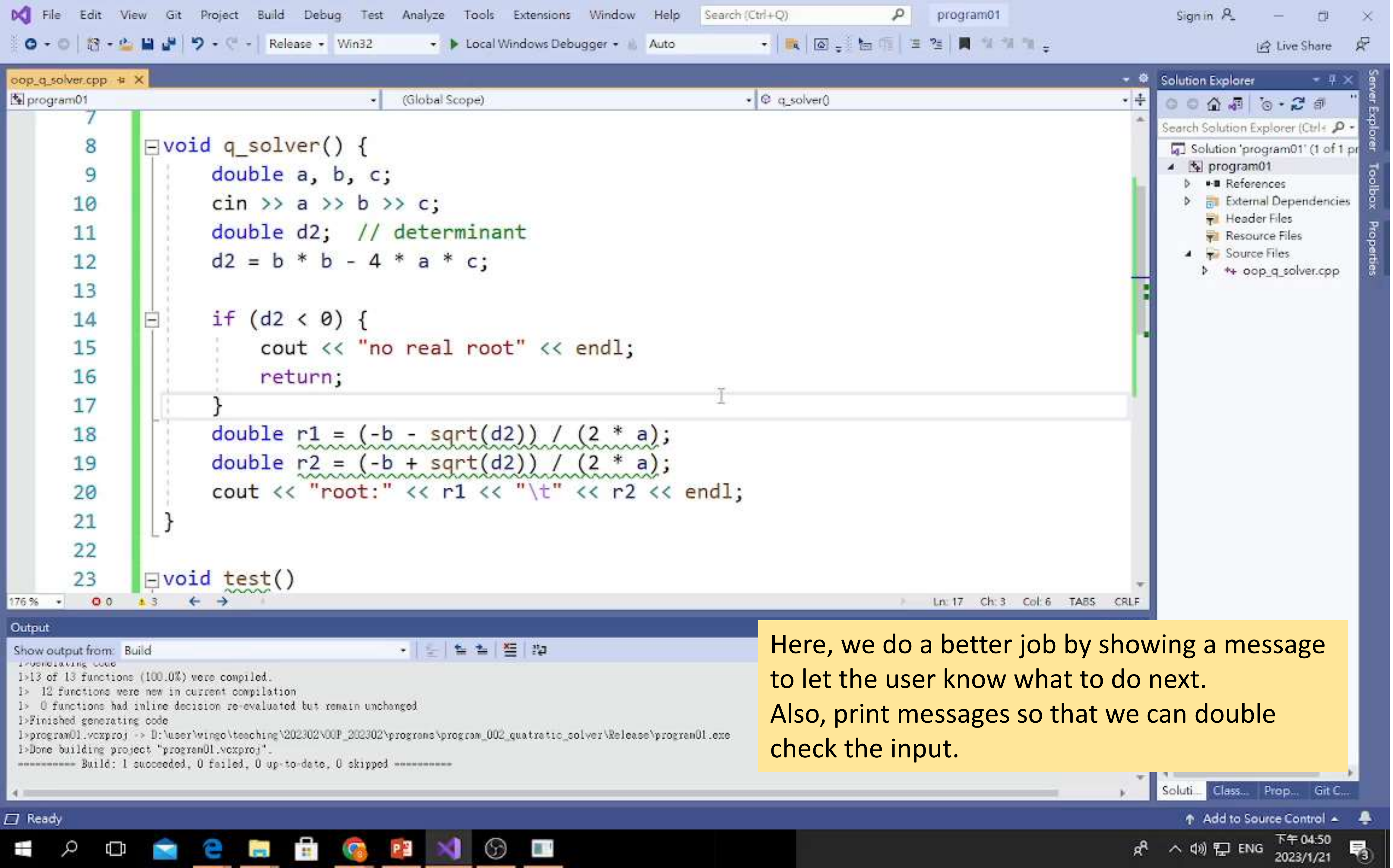
Show output from: Build

```
1>13 of 13 functions (100.0%) were compiled.
1>  12 functions were new in current compilation
1>  0 functions had inline decision re-evaluated but remain unchanged
1>Finished generating code
1>program01.vcxproj -> D:\user\wingo\teaching\202302\OOP_202302\programs\program_002_quatratic_solver\Release
1>Done building project "program01.vcxproj".
========== Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped ==========
```

$$1\,x^2 + 5x + 6 = 0$$
$$(x+2)(x+3) = 0$$

Users cannot know what they should do.

~01:38



```cpp
void q_solver() {
    double a, b, c;
    cin >> a >> b >> c;
    double d2;  // determinant
    d2 = b * b - 4 * a * c;

    if (d2 < 0) {
        cout << "no real root" << endl;
        return;
    }
    double r1 = (-b - sqrt(d2)) / (2 * a);
    double r2 = (-b + sqrt(d2)) / (2 * a);
    cout << "root:" << r1 << "\t" << r2 << endl;
}

void test()
```

```
1>13 of 13 functions (100.0%) were compiled.
1>  12 functions were new in current compilation
1>  0 functions had inline decision re-evaluated but remain unchanged
1>Finished generating code
1>program01.vcxproj -> D:\user\wingo\teaching\202302\OOP_202302\programs\program_002_quatratic_solver\Release\program01.exe
1>Done building project "program01.vcxproj".
========== Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped ==========
```

Here, we do a better job by showing a message to let the user know what to do next.
Also, print messages so that we can double check the input.

Convert the program into a class

- Then, create an object of the class.
- Use the object to do the tasks.

e.g.,

```
QuadraticEquationSolver solver;

// ask for input, solve the equation,
// and show the result
solver.foo( );
```

Convert the program into a class

- Then, create an object of the class.
- Use the object to do the tasks.

e.g.,

```
QuadraticEquationSolver solver;

// ask for input, solve the equation,
// and show the result
solver.foo( );
```

Check the correctness
of input determinant

```
void q_solver( ) {
  double a, b, c;
  cout << "Input a, b, and c:";
  cin >> a >> b >> c;
  // double check the input
  cout << "a:" << a << endl;
  cout << "b:" << b << endl;
  cout << "c:" << c << endl;
  double d2;                // determinant
  d2 = b * b - 4 * a * c;
  if (d2 < 0) {
    cout << "no real roots" << endl;
    return;
  }
  double r1 = (-b - sqrt(d2)) / (2 * a);
  double r2 = (-b + sqrt(d2)) / (2 * a);
  cout << "root:" << r1 << "\t" << r2 << endl;
}
```

Using a Class

```
class QuadraticEquationSolver {
protected:
   double a, b, c;
public:
   QuadraticEquationSolver() { … }
   void foo( ) {          // give a good name
               …
   }
};
```

```
void q_solver( ) {
  double a, b, c;
  cout << "Input a, b, and c:";
  cin >> a >> b >> c;
  // double check the input
  cout << "a:" << a << endl;
  cout << "b:" << b << endl;
  cout << "c:" << c << endl;
  double d2;                  // determinant
  d2 = b * b - 4 * a * c;
  if (d2 < 0) {
    cout << "no real roots" << endl;
    return;
  }
  double r1 = (-b - sqrt(d2)) / (2 * a);
  double r2 = (-b + sqrt(d2)) / (2 * a);
  cout << "root:" << r1 << "\t" << r2 << endl;
}
```

Using a Class

```
class QuadraticEquationSolver {
protected:
   double a, b, c;
public:
   QuadraticEquationSolver( );
   void solve( );
};
```

```
void QuadraticEquationSolver::solve( ) {
  cout << "Input a, b, and c:";
  cin >> a >> b >> c;
  // double check the input
  cout << "a:" << a << endl;
  cout << "b:" << b << endl;
  cout << "c:" << c << endl;
  double d2;                // determinant
  d2 = b * b - 4 * a * c;
  if (d2 < 0) {
    cout << "no real roots" << endl;
    return;
  }
  double r1 = (-b - sqrt(d2)) / (2 * a);
  double r2 = (-b + sqrt(d2)) / (2 * a);
  cout << "root:" << r1 << "\t" << r2 << endl;
}
```

```
void q_solver( ) {
  double a, b, c;
  cout << "Input a, b, and c:";
  cin >> a >> b >> c;
  // double check the input
  cout << "a:" << a << endl;
  cout << "b:" << b << endl;
  cout << "c:" << c << endl;
  double d2;                // determinant
  d2 = b * b - 4 * a * c;
  if (d2 < 0) {
    cout << "no real roots" << endl;
    return;
  }
  double r1 = (-b - sqrt(d2)) / (2 * a);
  double r2 = (-b + sqrt(d2)) / (2 * a);
  cout << "root:" << r1 << "\t" << r2 << endl;
}
```

Using a Class

```
class QuadraticEquationSolver {
protected:
   double a, b, c;



public:
   QuadraticEquationSolver( );
   void solve( );
};
```

```
void QuadraticEquationSolver::solve( ) {
  cout << "Input a, b, and c:";
  cin >> a >> b >> c;
  // double check the input
  cout << "a:" << a << endl;
  cout << "b:" << b << endl;
  cout << "c:" << c << endl;
  double d2;              // determinant
  d2 = b * b - 4 * a * c;
  if (d2 < 0) {
     cout << "no real roots" << endl;
     return;
  }
  double r1 = (-b - sqrt(d2)) / (2 * a);
  double r2 = (-b + sqrt(d2)) / (2 * a);
  cout << "root:" << r1 << "\t" << r2 << endl;
}
```

Use data members to store the determinant and the two roots

```
void q_solver( ) {
  double a, b, c;
  cout << "Input a, b, and c:";
  cin >> a >> b >> c;
  // double check the input
  cout << "a:" << a << endl;
  cout << "b:" << b << endl;
  cout << "c:" << c << endl;
  double d2;              // determinant
  d2 = b * b - 4 * a * c;
  if (d2 < 0) {
     cout << "no real roots" << endl;
     return;
  }
  double r1 = (-b - sqrt(d2)) / (2 * a);
  double r2 = (-b + sqrt(d2)) / (2 * a);
  cout << "root:" << r1 << "\t" << r2 << endl;
}
```

Using a Class

```
class QuadraticEquationSolver {
protected:
   double a, b, c;
   double d2;          // determinant
   double r1, r2;      // roots
public:
   QuadraticEquationSolver( );
   void solve( );
};
```

Use data members to store the determinant and the two roots

```
void QuadraticEquationSolver::solve( ) {
  cout << "Input a, b, and c:";
  cin >> a >> b >> c;
  // double check the input
  cout << "a:" << a << endl;
  cout << "b:" << b << endl;
  cout << "c:" << c << endl;
  ~~double d2;              // determinant~~
  d2 = b * b - 4 * a * c;
  if (d2 < 0) {
    cout << "no real roots" << endl;
    return;
  }
  ~~double~~ r1 = (-b - sqrt(d2)) / (2 * a);
  ~~double~~ r2 = (-b + sqrt(d2)) / (2 * a);
  cout << "root:" << r1 << "\t" << r2 << endl;
}
```

```
void q_solver( ) {
  double a, b, c;
  cout << "Input a, b, and c:";
  cin >> a >> b >> c;
  // double check the input
  cout << "a:" << a << endl;
  cout << "b:" << b << endl;
  cout << "c:" << c << endl;
  double d2;              // determinant
  d2 = b * b - 4 * a * c;
  if (d2 < 0) {
    cout << "no real roots" << endl;
    return;
  }
  double r1 = (-b - sqrt(d2)) / (2 * a);
  double r2 = (-b + sqrt(d2)) / (2 * a);
  cout << "root:" << r1 << "\t" << r2 << endl;
}
```

Using a Class

```
class QuadraticEquationSolver {
protected:
   double a, b, c;
   double d2;          // determinant
   double r1, r2;      // roots
public:
   QuadraticEquationSolver( );
   void solve( );
};
```

```
void QuadraticEquationSolver::solve( ) {
  cout << "Input a, b, and c:";
  cin >> a >> b >> c;
  // double check the input
  cout << "a:" << a << endl;
  cout << "b:" << b << endl;
  cout << "c:" << c << endl;

  d2 = b * b - 4 * a * c;
  if (d2 < 0) {
    cout << "no real roots" << endl;
    return;
  }
  r1 = (-b - sqrt(d2)) / (2 * a);
  r2 = (-b + sqrt(d2)) / (2 * a);
  cout << "root:" << r1 << "\t" << r2 << endl;
}
```

```
void q_solver( ) {
  double a, b, c;
  cout << "Input a, b, and c:";
  cin >> a >> b >> c;
  // double check the input
  cout << "a:" << a << endl;
  cout << "b:" << b << endl;
  cout << "c:" << c << endl;
  double d2;              // determinant
  d2 = b * b - 4 * a * c;
  if (d2 < 0) {
    cout << "no real roots" << endl;
    return;
  }
  double r1 = (-b - sqrt(d2)) / (2 * a);
  double r2 = (-b + sqrt(d2)) / (2 * a);
  cout << "root:" << r1 << "\t" << r2 << endl;
}
```

Using a Class

```
class QuadraticEquationSolver {
protected:
   double a, b, c;
   double d2;          // determinant
   double r1, r2;      // roots
public:
   QuadraticEquationSolver( );
   void solve( );
};
```

```
void QuadraticEquationSolver::solve( ) {
  cout << "Input a, b, and c:";
  cin >> a >> b >> c;
  // double check the input
  cout << "a:" << a << endl;
  cout << "b:" << b << endl;
  cout << "c:" << c << endl;

  d2 = b * b - 4 * a * c;
  if (d2 < 0) {
    cout << "no real roots" << endl;
    return;
  }
  r1 = (-b - sqrt(d2)) / (2 * a);
  r2 = (-b + sqrt(d2)) / (2 * a);
  cout << "root:" << r1 << "\t" << r2 << endl;
}
```

A — A. Ask for input

B — B. Show coefficients (Show input)

C — C. Compute determinant

D — D. Check if there are no real roots.

E — E. Calculate real roots

F — F. Show roots

Using a Class

```
class QuadraticEquationSolver {
protected:
   double a, b, c;
   double d2;          // determinant
   double r1, r2;      // roots
public:
   QuadraticEquationSolver( );
   void solve( );
protected:
   void askForInput( );
   void showInput( );
   void computeDeterminant( );
   void checkForRealRoots( );
   void computeRealRoots( );
   void showRoots( );
};
```

Implement more methods to complete the class behaviors

```
void QuadraticEquationSolver::solve( ) {
  cout << "Input a, b, and c:";
  cin >> a >> b >> c;
  // double check the input
  cout << "a:" << a << endl;
  cout << "b:" << b << endl;
  cout << "c:" << c << endl;

  d2 = b * b - 4 * a * c;
  if (d2 < 0) {
    cout << "no real roots" << endl;
    return;
  }
  r1 = (-b - sqrt(d2)) / (2 * a);
  r2 = (-b + sqrt(d2)) / (2 * a);
  cout << "root:" << r1 << "\t" << r2 << endl;
}
```

A — A. Ask for input

B — B. Show coefficients (Show input)

C — C. Compute determinant

D — D. Check if there are no real roots.

E — E. Calculate real roots

F — F. Show roots

Using a Class

```
class QuadraticEquationSolver {
protected:
   double a, b, c;
   double d2;           // determinant
   double r1, r2;       // roots
public:
   QuadraticEquationSolver( );
   void solve( );
protected:
   void askForInput( );
   void showInput( ) const;
   void computeDeterminant( );
   bool checkForRealRoots( );
   void computeRealRoots( );
   void showRoots( ) const;
};
```

```
void QuadraticEquationSolver::solve( ) {
  cout << "Input a, b, and c:";
  cin >> a >> b >> c;
  // double check the input
  cout << "a:" << a << endl;
  cout << "b:" << b << endl;
  cout << "c:" << c << endl;

  d2 = b * b - 4 * a * c;
  if (d2 < 0) {
    cout << "no real roots" << endl;
    return;
  }
  r1 = (-b - sqrt(d2)) / (2 * a);
  r2 = (-b + sqrt(d2)) / (2 * a);
  cout << "root:" << r1 << "\t" << r2 << endl;
}
```

```
void QuadraticEquationSolver:::askForInput( ) {
  cout << "Input a, b, and c:";   cin >> a >> b >> c;
}
void QuadraticEquationSolver ::showInput( ) const {
  cout << "a:" << a << endl;
  cout << "b:" << b << endl;
  cout << "c:" << c << endl;
}
void QuadraticEquationSolver :: computeDeterminant( ) {
   d2 = b * b - 4 * a * c;
}
bool QuadraticEquationSolver :: checkForRealRoots( ) {
   if (d2 < 0) {
      cout << "no real roots" << endl;
   }
  return d2 >= 0;
}
void QuadraticEquationSolver :: computeRealRoots( ) {
  r1 = (-b - sqrt(d2)) / (2 * a);
  r2 = (-b + sqrt(d2)) / (2 * a);
}
void QuadraticEquationSolver :: showRoots( )  const {
  cout << "root:" << r1 << "\t" << r2 << endl;
}
```

Using a Class

```
void QuadraticEquationSolver::solve( ) {
  cout << "Input a, b, and c:";
  cin >> a >> b >> c;
  // double check the input
  cout << "a:" << a << endl;
  cout << "b:" << b << endl;
  cout << "c:" << c << endl;

  d2 = b * b - 4 * a * c;
  if (d2 < 0) {
    cout << "no real roots" << endl;
    return;
  }
  r1 = (-b - sqrt(d2)) / (2 * a);
  r2 = (-b + sqrt(d2)) / (2 * a);
  cout << "root:" << r1 << "\t" << r2 << endl;
}
```

A

A. Ask for input

B

B. Show coefficients (Show input)

C

C. Compute determinant

D

D. Check if there are no real roots.

E

E. Calculate real roots

F

F. Show roots

Using a Class

```
void QuadraticEquationSolver::solve( ) {
  cout << "Input a, b, and c:";
  cin >> a >> b >> c;
  // double check the input
  cout << "a:" << a << endl;
  cout << "b:" << b << endl;
  cout << "c:" << c << endl;

  d2 = b * b - 4 * a * c;
  if (d2 < 0) {
    cout << "no real roots" << endl;
    return;
  }
  r1 = (-b - sqrt(d2)) / (2 * a);
  r2 = (-b + sqrt(d2)) / (2 * a);
  cout << "root:" << r1 << "\t" << r2 << endl;
}
```

A

A. Ask for input

B

B. Show coefficients (Show input)

C

C. Compute determinant

D

D. Check if there are no real roots.

E

E. Calculate real roots

F

F. Show roots

```
void
QuadraticEquationSolver::solve( ) {
  askForInput( );
  showInput( );
  computeDeterminant( );
  if (  checkForRealRoots( ) ) {
    computeRealRoots( );
    showRoots( );
  }
}
```

Using a Class

```
void QuadraticEquationSolver::solve( ) {
  cout << "Input a, b, and c:";
  cin >> a >> b >> c;
  // double check the input
  cout << "a:" << a << endl;
  cout << "b:" << b << endl;
  cout << "c:" << c << endl;

  d2 = b * b - 4 * a * c;
  if (d2 < 0) {
    cout << "no real roots" << endl;
    return;
  }
  r1 = (-b - sqrt(d2)) / (2 * a);
  r2 = (-b + sqrt(d2)) / (2 * a);
  cout << "root:" << r1 << "\t" << r2 << endl;
}
```

```
void
QuadraticEquationSolver::solve( ) {
  askForInput( );
  showInput( );
  computeDeterminant( );

  if (  checkForRealRoots( ) ) {
    computeRealRoots( );
    showRoots( );
  }
}
```

Using a Class

```
void QuadraticEquationSolver::solve( ) {
  cout << "Input a, b, and c:";
  cin >> a >> b >> c;
  // double check the input
  cout << "a:" << a << endl;
  cout << "b:" << b << endl;
  cout << "c:" << c << endl;

  d2 = b * b - 4 * a * c;
  if (d2 < 0) {
    cout << "no real roots" << endl;
    return;
  }
  r1 = (-b - sqrt(d2)) / (2 * a);
  r2 = (-b + sqrt(d2)) / (2 * a);
  cout << "root:" << r1 << "\t" << r2 << endl;
}
```

```
void
QuadraticEquationSolver::solve( ) {
  askForInput( );
  showInput( );
  computeDeterminant( );

  if (  checkForRealRoots( ) ) {
    computeRealRoots( );
    showRoots( );
  }
}
```

Good style

```
void
QuadraticEquationSolver::solve( ) {
  askForInput( );
  showInput( );
  computeDeterminant( );

  if (  !checkForRealRoots( ) ) return;

  computeRealRoots( );
  showRoots( );
}
```

Using a Class

```
class QuadraticEquationSolver {
protected:
   double a, b, c;
   double d2;          // determinant
   double r1, r2;      // roots
public:
   QuadraticEquationSolver( );
   void solve( );
protected:
   void askForInput( );
   void showInput( ) const;
   void computeDeterminant( );
   bool checkForRealRoots( );
   void computeRealRoots( );
   void showRoots( ) const;
};
```

```
class QuadraticEquationSolver {
protected:
   double a, b, c;
   double d2;          // determinant
   double r1, r2;      // roots
public:
   QuadraticEquationSolver( );
   void solve( );
protected:
   void askForInput( );
   void showInput( ) const;
   void computeDeterminant( );
   bool checkForRealRoots( );
   void computeRealRoots( );
   void showRoots( ) const;
   void showDeterminant( ) const;
};
```

What should we **show** to **help us debug** and check **correctness of the process**?

Using a Class

```
class QuadraticEquationSolver {
protected:
   double a, b, c;
   double d2;           // determinant
   double r1, r2;       // roots
public:
   QuadraticEquationSolver( );
   void solve( );
protected:
   void askForInput( );
   void showInput( ) const;
   void computeDeterminant( );
   bool checkForRealRoots( );
   void computeRealRoots( );
   void showRoots( ) const;
};
```

```
class QuadraticEquationSolver {
protected:
   double a, b, c;
   double d2;           // determinant
   double r1, r2;       // roots
public:
   QuadraticEquationSolver( );
   void solve( );
protected:
   void askForInput( );
   void showInput( ) const;
   void computeDeterminant( );
   bool checkForRealRoots( );
   void computeRealRoots( );
   void showRoots( ) const;
   void showDeterminant( ) const;
};
```

What should we **show** to **help us debug** and check **correctness of the process**?

A1 A2 that can help us debug and A3 the A4 of the steps of the process.

Example Two: Requirement Specification

- Write a program to ask the user to input a quadratic equation

Example Two: Requirement Specification

- Write a program to ask the user to input a quadratic equation
- Solve the quadratic equation.
- Show the roots (real or A1).

Example Two: Requirement Specification

- Write a program to ask the user to input a quadratic equation
- Solve the quadratic equation.
- Show the roots (real or complex).

$$a\,x^2 + b\,x + c = 0$$

$$\text{root1} = (\boxed{\quad A1 \quad}) \,/\, \boxed{A2}$$

$$\text{root2} = (\boxed{\quad A3 \quad}) \,/\, \boxed{A4}$$

Requirement Specification

$x^2 + 1 = 0$

$a = 1, b = 0, c = 1$

Requirement Specification

$x^2 + 1 = 0$

$a = 1, b = 0, c = 1$

determinant = $b^2 - 4ac = -4$, sqrt(-4) = $\boxed{\begin{matrix}A\\1\end{matrix}}$

Requirement Specification

$x^2 + 1 = 0$

$a = 1, b = 0, c = 1$

determinant = $b^2 - 4ac = -4$, sqrt(-4) = 2i

sqrt(-1) = i

$i^2 = \boxed{\text{A1}}$

i: the $\boxed{\text{A2}}$ unit

Requirement Specification

$x^2 + 1 = 0$

$a = 1, b = 0, c = 1$

determinant = $b^2 – 4ac = -4$, sqrt(-4) = 2i

root1 = (-b - sqrt($b^2 – 4ac$)) / 2a

root2 = (-b + sqrt($b^2 – 4ac$)) / 2a

root1 = root2 = A
1

sqrt(-1) = i

$i^2 = -1$

i: the imaginary unit

Requirement Specification

- Write a program to ask the user to input a quadratic equation and solve the quadratic equation.
- Show the roots (real or complex).

$a\,x^2 + b\,x + c = 0$

determinant $D = b^2 – 4ac$

root1 = $(-b - \mathrm{sqrt}(\, b^2 – 4ac\,)\,) / 2a$

root2 = $(-b + \mathrm{sqrt}(\, b^2 – 4ac\,)\,) / 2a$

Requirement Specification

- Write a program to ask the user to input a quadratic equation and solve the quadratic equation.
- Show the roots (real or complex).

$$a\,x^2 + b\,x + c = 0$$

$$\text{determinant } D = b^2 - 4ac$$

$$\text{root1} = (-b - \text{sqrt}(\, b^2 - 4ac\,)\,) / 2a$$
$$\text{root2} = (-b + \text{sqrt}(\, b^2 - 4ac\,)\,) / 2a$$

-How do we compute sqrt(D) if D is A1 ?

Requirement Specification

- Write a program to ask the user to input a quadratic equation and solve the quadratic equation.
- Show the roots (real or complex).

$$a\,x^2 + b\,x + c = 0$$

$$\text{determinant } D = b^2 - 4ac$$

$$\text{root1} = (-b - \text{sqrt}(\, b^2 - 4ac \,)\,) / 2a$$

$$\text{root2} = (-b + \text{sqrt}(\, b^2 - 4ac \,)\,) / 2a$$

-How do we compute sqrt(D) if D is negative?

Requirement Specification

Consider a simple question first

-How do we compute sqrt(D) if D is negative?

What do we want?

Requirement Specification

Consider a simple question first

-How do we compute sqrt(D) if D is negative?

What do we want?

E.g., sqrt(-4), we show sqrt(4) i,

Requirement Specification

Consider a simple question first

-How do we compute sqrt(D) if D is negative?

What do we want?

E.g., sqrt(-4), we show sqrt(4) i,

i.e., sqrt(-D) **i**

Requirement Specification

- Write a program to ask the user to input a quadratic equation and solve the quadratic equation.
- Show the roots (real or complex).

$a\, x^2 + b\, x + c = 0$

determinant $D = b^2 - 4ac$

root1 = $(-b - \text{sqrt}(b^2 - 4ac)) / 2a$

root2 = $(-b + \text{sqrt}(b^2 - 4ac)) / 2a$

-How do we compute sqrt(D) if D is negative?

Example:

D = -4

Show

sqrt(D) as

2i

sqrt(-D) i

Requirement Specification

- Write a program to ask the user to input a quadratic equation and solve the quadratic equation.
- Show the roots (real or complex).

$a\ x^2 + b\ x + c = 0$

determinant $D = b^2 - 4ac$

root1 = $(-b - \text{sqrt}(\ b^2 - 4ac\ )\ )\ /\ 2a$

root2 = $(-b + \text{sqrt}(\ b^2 - 4ac\ )\ )\ /\ 2a$

Example:

D = -4

Show

sqrt(D) as

2i

-How do we compute sqrt(D) if D is negative?

sqrt(-D) i

So, cout << A1 **<<** A2 **<< endl;**

Requirement Specification

- Write a program to ask the user to input a quadratic equation and solve the quadratic equation.
- Show the roots (real or complex).

$$a\,x^2 + b\,x + c = 0$$

$$\text{determinant } D = b^2 - 4ac$$

$$\text{root1} = (-b - \text{sqrt}(\, b^2 - 4ac\,)\,) / 2a$$

$$\text{root2} = (-b + \text{sqrt}(\, b^2 - 4ac\,)\,) / 2a$$

-Need a condition check

```
if ( D < 0 ) cout << sqrt(-D) << “i” << endl;
else cout << A1 << endl;
```

Requirement Specification

-How do we show sqrt(D)?

```
if (D < 0 ) cout << sqrt(-D) << “i” << endl;
else cout << sqrt(D) << endl;

// if D = -4, the output is
```

A1

```
// if D = 4, the output is
```

A2

Requirement Specification

-How do we show sqrt(D)?

```
if (D < 0 ) cout << sqrt(-D) << “i” << endl;
else cout << sqrt(D) << endl;

// if D = -4, the output is
2i

// if D = 4, the output is
2
```

Intended Learning Outcomes

- Define a class
- Define the meaning of a member function that has const at its declaration
- Implement functions to show intermediate results for debugging

Supplemental Material

Exercise

```
class CLASS_A {
        public:
        void foo( ) {

        }
        public:
        int score;
};
```

```
void test( ) {
  CLASS_A a( );        //error?
  CLASS_A c;           //error?
  a.score = 5;         //error?
  c.score = 7;         //error?
}
```

Exercise

```
class CLASS_A {
        public:
        void foo( ) {

        }
        public:
        int score;
};
```

```
void test( ) {
  CLASS_A a( );        //no error
  CLASS_A c;           //no error
  a.score = 5;         //error
  c.score = 7;         //no error
}
```

Exercise

```
class CLASS_A {
        public:
        void foo( ) {

        }
        public:
        int score;
};
```

```
void test( ) {
  CLASS_A a( );        //no error
  CLASS_A c;           //no error
  a.score = 5;         //error
  c.score = 7;         //no error
}
```

```
CLASS_A a( );     //Declare a function which returns an object of CLASS_A
CLASS_A c;        //Use the default constructor to create object c
a.score = 5;      //a is a function name. Should call a( ).score = 5;
c.score = 7;      //score is public. So c.score can be accessed by client test
```

How to define a class?

```
void f( ) {
  CLASS_A x( );  // no error
   ……
   x.score = 5;     // error
}
```

We declare a function x which does not have a parameter; and it returns an object of CLASS_A. x is not an object.

```
void f( ) {
  CLASS_A x;       // no error
   ……
   x.score = 5;     // no error
}
```

We declare an object x of CLASS_A. x is initialized by the default constructor.

Invoke the function.
x( ).score = 5. // if score is public, it works.
It reads as follows: invoke function x that returns an object and then assign 5 to score of the object.

How to define a class?

```
class CLASS_A {
        public:
        void foo( ) {
                // this is a method.
        }
        public:
        int score;      // this is a data member
};
```

```
CLASS_A a;     // use default constructor
CLASS_A c;     // use default constructor
```

System Requirement

What are the requirements for running our software?

Hardware? (Audio? AI Computation?)

Software? (Maya, 3D Max, Unity)

Libraries (OpenCV? Tensor Flow)

Graphics processing units? (Rendering? Dynamic textures?)

Example One: Quadratic equation solver

- Requirement specification: Write a program to ask the user to input a quadratic equation and solve the quadratic equation.
- If there are real roots, show them. Otherwise, show "no real roots".

$$a x^2 + b x + c = 0$$

$$\text{root1} = (-b - \text{sqrt}(\ b^2 - 4ac\ )\ )\ /\ 2a$$
$$\text{root2} = (-b + \text{sqrt}(\ b^2 - 4ac\ )\ )\ /\ 2a$$

Show a message to let the user know what to do next

Print messages so that we can double check the input.

```
void q_solver( ) {
  double a, b, c;
  cout << "Input a, b, and c:";
  cin >> a >> b >> c;
  // double check the input
  cout << "a:" << a << endl;
  cout << "b:" << b << endl;
  cout << "c:" << c << endl;
  double d2;              // determinant
  d2 = b * b - 4 * a * c;
  if (d2 < 0) {
    cout << "no real roots" << endl;
    return;
  }
  double r1 = (-b - sqrt(d2)) / (2 * a);
  double r2 = (-b + sqrt(d2)) / (2 * a);
  cout << "root:" << r1 << "\t" << r2 << endl;
}
```