CSE 1062 **Fundamentals of Programming**

Lecture #7

Spring 2016

Computer Science & Engineering Program
The School of EE & Computing
Adama Science & Technology University

- Branching Structure
  - Selection Criteria
  - Relational and Logical Operators
  - The if-else statement
  - Nested if statement
  - The switch statement
- Looping Structure
  - Basic loop structures
  - while loops and for loops
  - Nested Loops
  - do while loops

Case study: [General Math]: Solving Quadratic Equations

[Physics]: Flight of a Ball

Reading assignment

- Chapter 4 of the textbook
- Chapter 5 of the textbook

Selection Criteria

- **`if-else`** statement: Implements a decision structure for two alternatives

  Syntax:

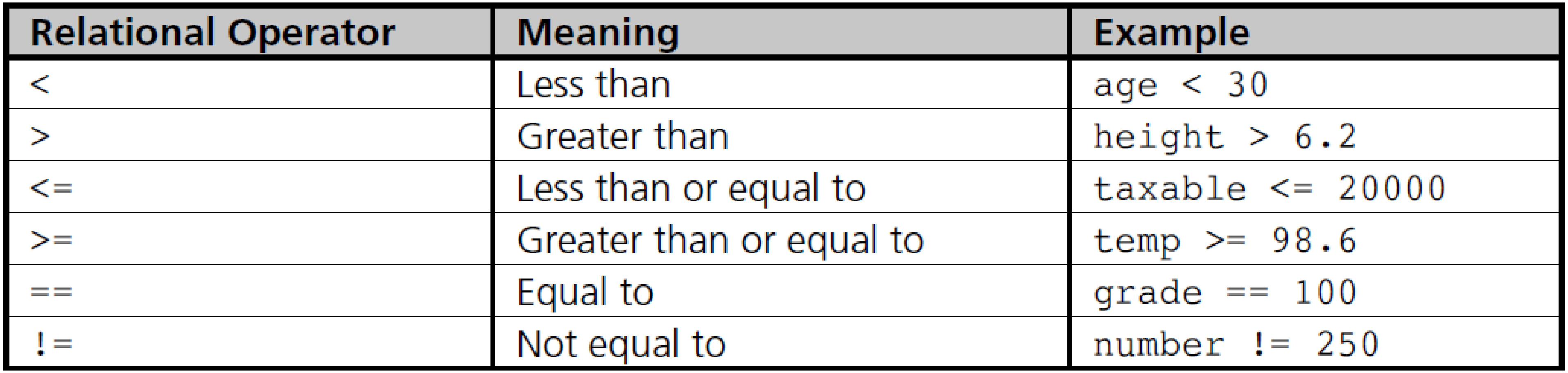
  ```
  if (condition)
      statement executed if condition is true;
  else
      statement executed if condition is false;
  ```

Selection Criteria

- The condition is evaluated to its numerical value:
  - A non-zero value is considered to be true
  - A zero value is considered to be false
- The **`else`** portion is optional
  - Executed only if the condition is false
- The condition may be any valid C++ expression

Relational Operators

- **Relational expression:** Compares two operands or expressions using **relational operators**



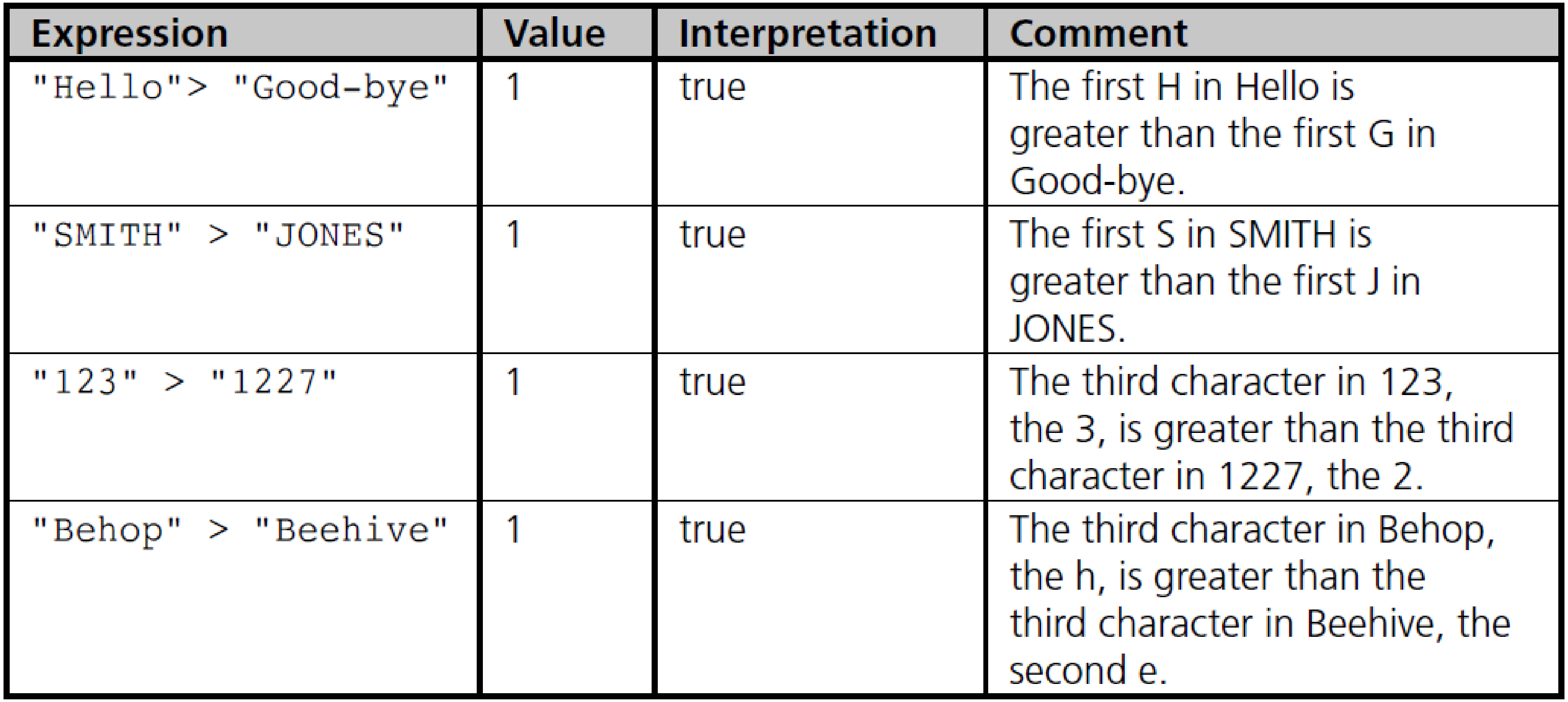
Relational Operator	Meaning	Example
<	Less than	`age < 30`
>	Greater than	`height > 6.2`
<=	Less than or equal to	`taxable <= 20000`
>=	Greater than or equal to	`temp >= 98.6`
==	Equal to	`grade == 100`
!=	Not equal to	`number != 250`

Relational Operators

- Relational expressions are evaluated to a numerical value of 1 or 0 only:
  - If the value is 1, the expression is true
  - If the value is 0, the expression is false
- **`char`** values are automatically coerced to **`int`** values for comparison purposes
- Strings are compared on a character by c haracter basis
  - The string with the first lower character is c onsidered smaller

Relational Operators

- Examples of string comparisons



Expression	Value	Interpretation	Comment
`"Hello"> "Good-bye"`	1	true	The first H in Hello is greater than the first G in Good-bye.
`"SMITH" > "JONES"`	1	true	The first S in SMITH is greater than the first J in JONES.
`"123" > "1227"`	1	true	The third character in 123, the 3, is greater than the third character in 1227, the 2.
`"Behop" > "Beehive"`	1	true	The third character in Behop, the h, is greater than the third character in Beehive, the second e.

Logical Operators

- AND (**&&**): Condition is true only if both expressions are true
- OR (**||**): Condition is true if either one or both of the expressions is true
- NOT (**!**): Changes an expression to its opposite state; true becomes false, false becomes true

Operator Precedence

Operator	Associativity
! unary - ++ --	Right to left
* / %	Left to right
+ -	Left to right
< <= > >=	Left to right
== !=	Left to right
&&	Left to right
||	Left to right
= += -= *= /=	Right to left

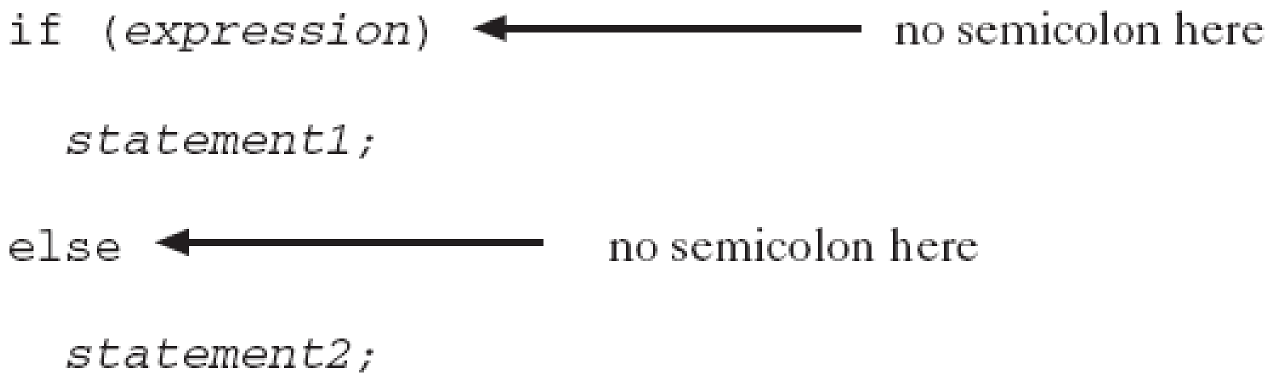
A Numerical Accuracy Problem

- Comparing single and double precision values for equality (==) can lead to errors because values are stored in binary
- Instead, test that the absolute value of the difference is within an acceptable range
- Example:

  abs(operandOne - operandTwo) < 0.000001

The if-else Statement

- **`if-else`** performs instructions based on the result of a comparison
- Place statements on separate lines for readability
- Syntax:



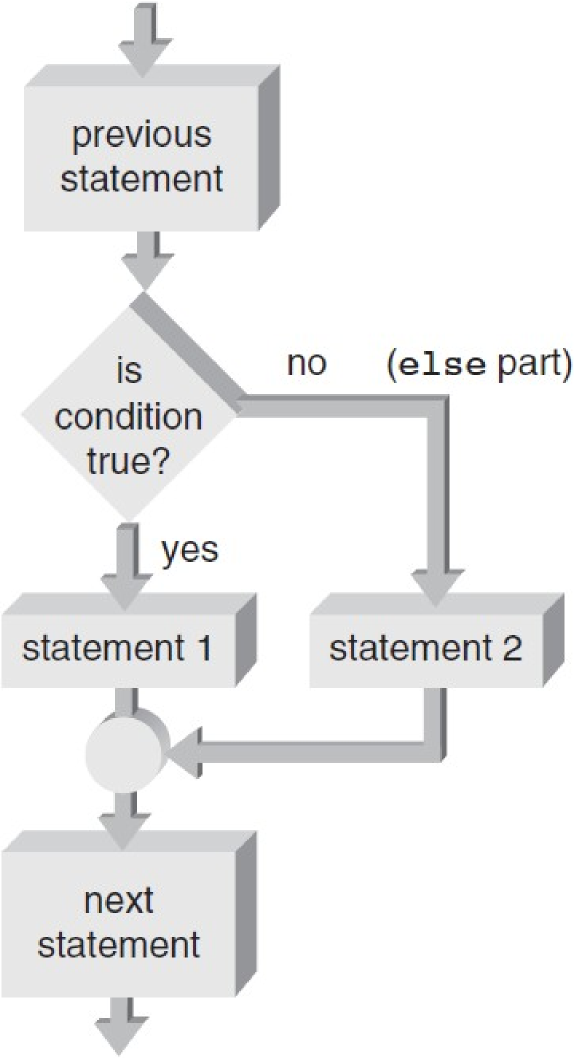
```
if (expression)
  statement1;
else
  statement2;
```

no semicolon here

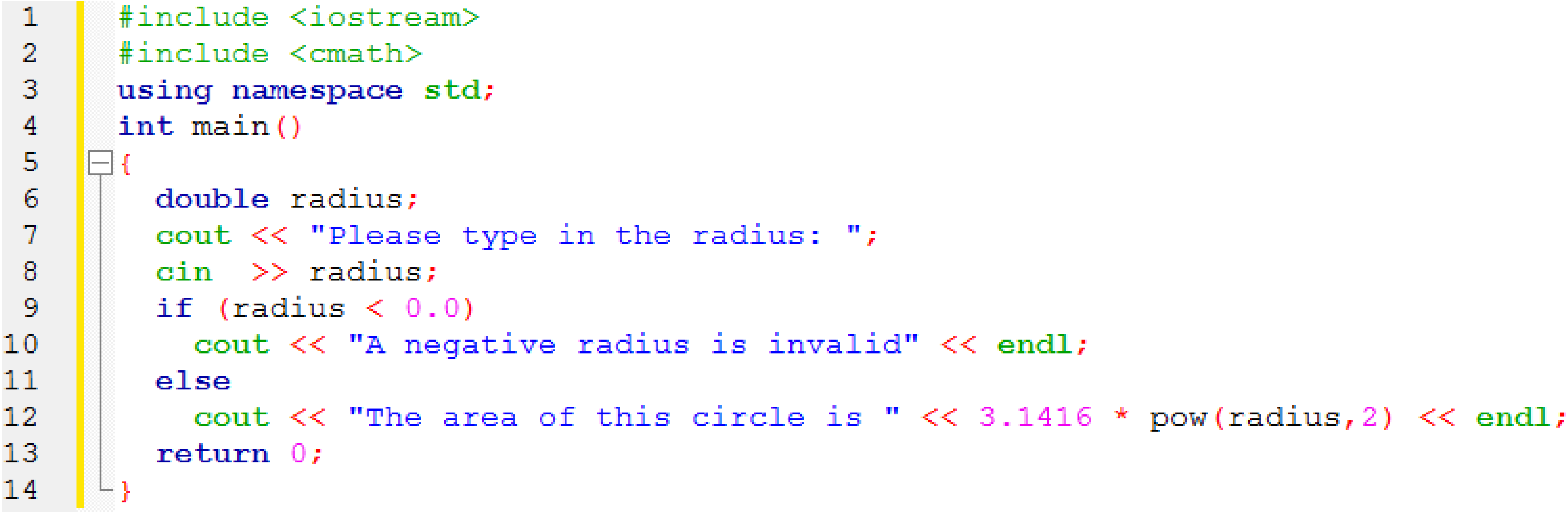
no semicolon here

The if-else Statement

ASTU



The if-else Statement

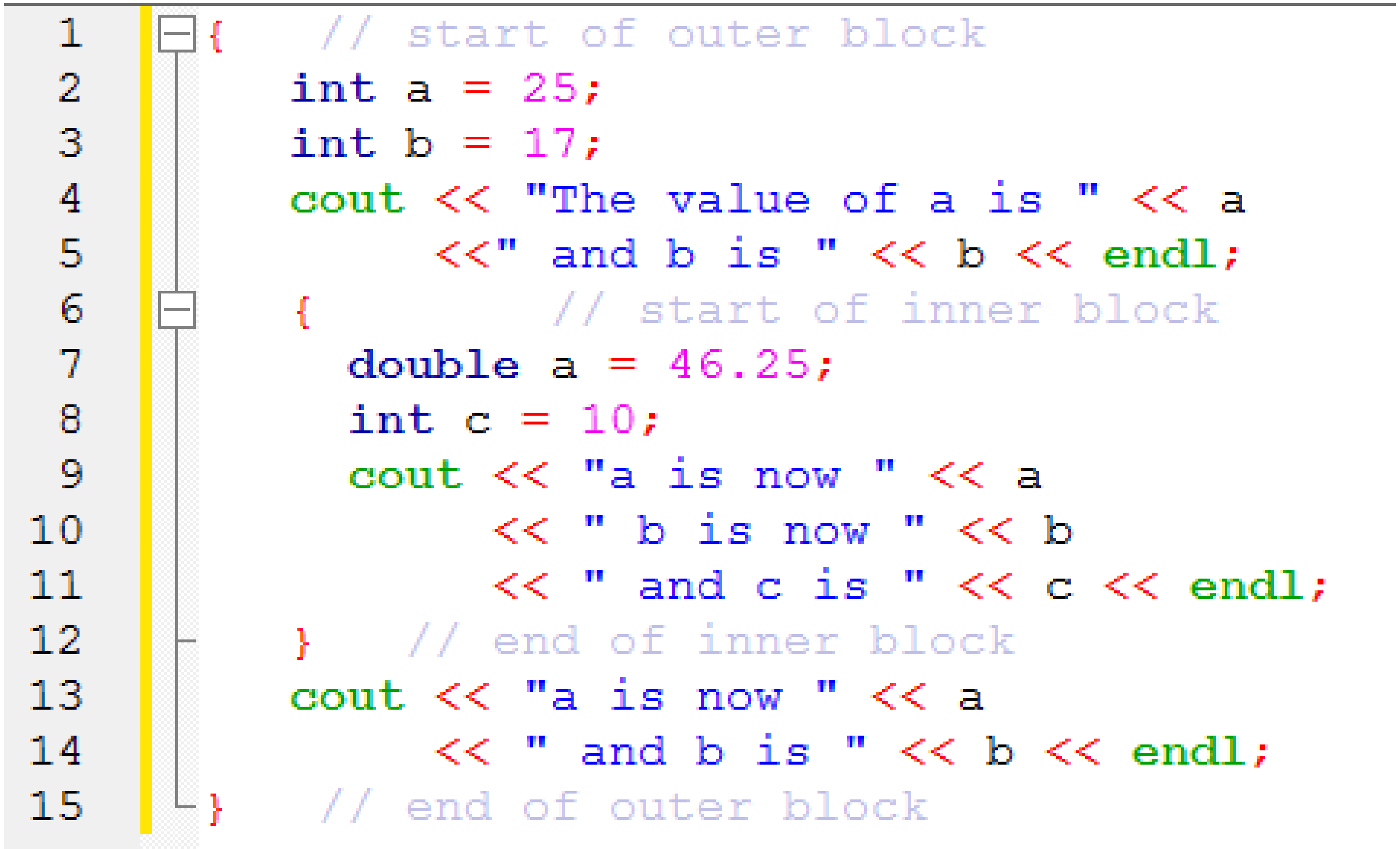


```
#include <iostream>
#include <cmath>
using namespace std;
int main()
{
  double radius;
  cout << "Please type in the radius: ";
  cin  >> radius;
  if (radius < 0.0)
    cout << "A negative radius is invalid" << endl;
  else
    cout << "The area of this circle is " << 3.1416 * pow(radius,2) << endl;
  return 0;
}
```

Compound Statements

- **Compound statement:** A sequence of single statements contained between braces
  - Creates a block of statements
  - A block of statements can be used anywhere that a single statement is legal
  - Any variable declared within a block is usable only within that block
- **Scope:** The area within a program where a variable can be used
  - A variable's scope is based on where the variable is declared

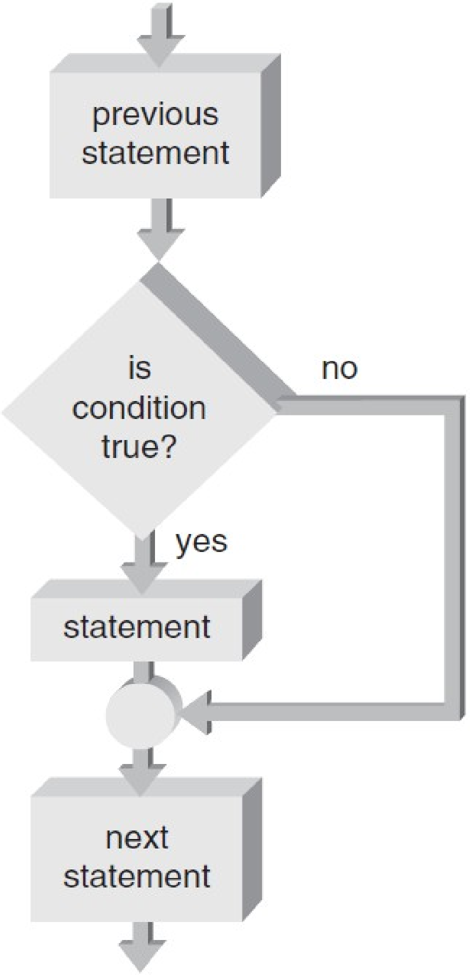
Compound Statements



```
{   // start of outer block
   int a = 25;
   int b = 17;
   cout << "The value of a is " << a
        <<" and b is " << b << endl;
   {        // start of inner block
     double a = 46.25;
     int c = 10;
     cout << "a is now " << a
          << " b is now " << b
          << " and c is " << c << endl;
   }   // end of inner block
   cout << "a is now " << a
        << " and b is " << b << endl;
}   // end of outer block
```

One-Way Selection

- **One-way selection:** An **`if`** statement without the optional **`else`** portion



Common Problems with if-else

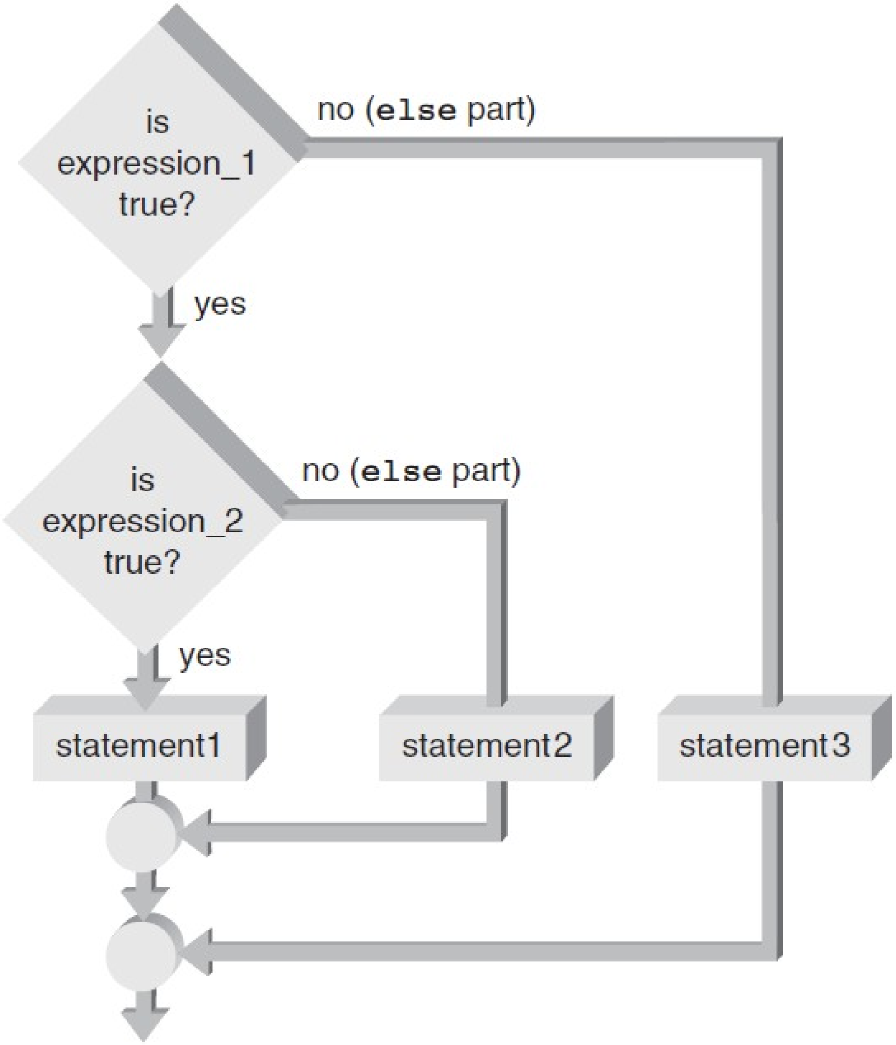
- Misunderstanding what an expression is
- Using the assignment operator (**=**) instead of the relational operator (**==**)

Nested if Statements

- **if-else** statement can contain any valid C++ statement, including another **if-else**
- Nested **if** statement: an **if-else** statement completely contained within another **if-else**
- Use braces to block code, especially when inner **if** statement does not have its own **else**

Nested if Statements

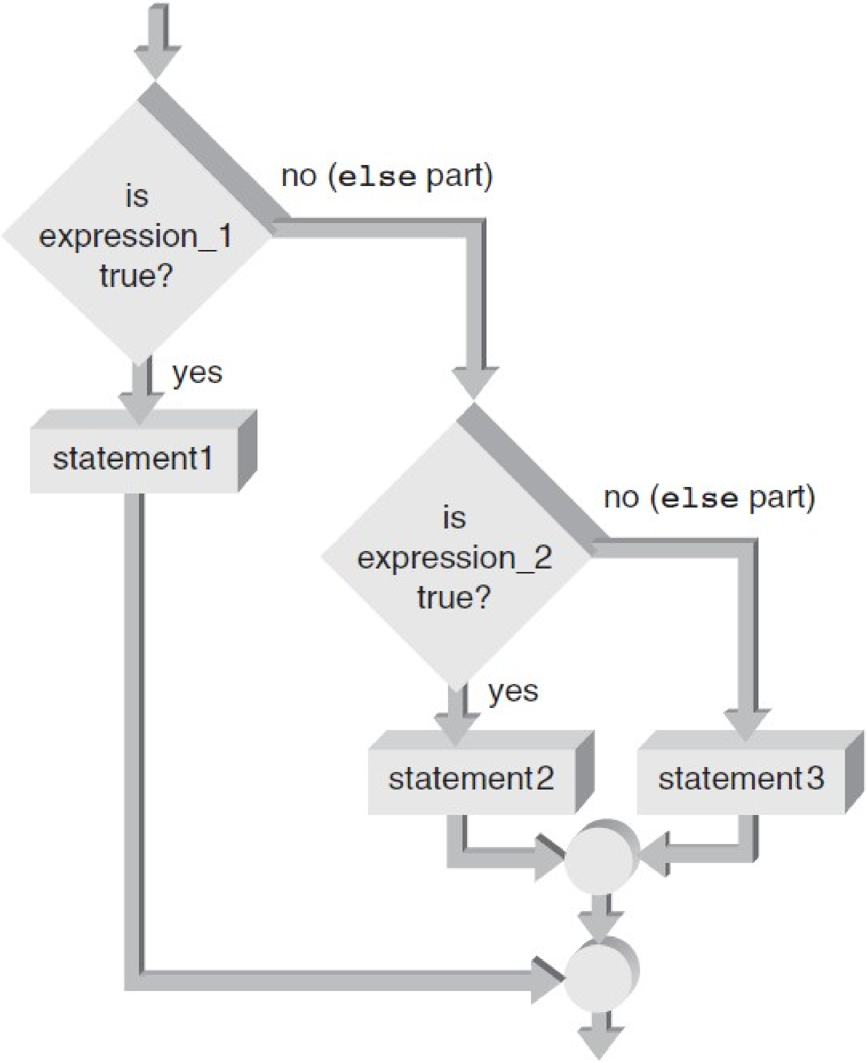
ASTU



The if-else Chain

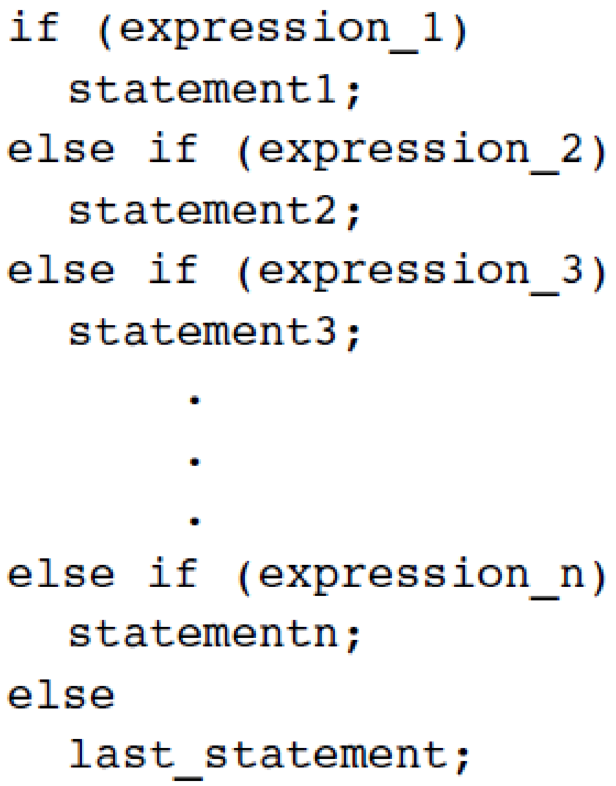
- **if-else chain:** A nested if statement occurring in the else clause of the outer if-else
- If any condition is true, the corresponding statement is executed and the chain terminates
- Final else is only executed if no conditions were true
  - Serves as a catch-all case
- if-else chain provides one selection from many possible alternatives

The if-else Chain



The if-else Chain

- General form of an **if-else** chain



```
if (expression_1)
   statement1;
else if (expression_2)
   statement2;
else if (expression_3)
   statement3;
         .
         .
         .
else if (expression_n)
   statementn;
else
   last_statement;
```

The switch Statement

- **switch statement:** Provides for one selection from many alternatives
- switch keyword starts the statement
  - Is followed by the expression to be evaluated
- **case** keyword identifies a value to be compared to the switch expression
  - When a match is found, statements in this case block are executed
- All further cases after a match is found are executed unless a **break** statement is found

The switch Statement

- **default** case is executed if no other case value matches were found
- **default** case is optional

Case Study: Solving Quadratic Equations

- **Data validation:** Use defensive programming techniques to validate user input
  - Includes code to check for improper data before an attempt is made to process it further
- **Solving quadratic equations:** Use the software development procedure to solve for the roots of a quadratic equation

Step 1: Analyze the Problem

- The problem requires
  - Accepting three **inputs**: the coefficients a, b, and c of a quadratic equation.
  - The **outputs** are the roots of the equation, found by using the given formulas.

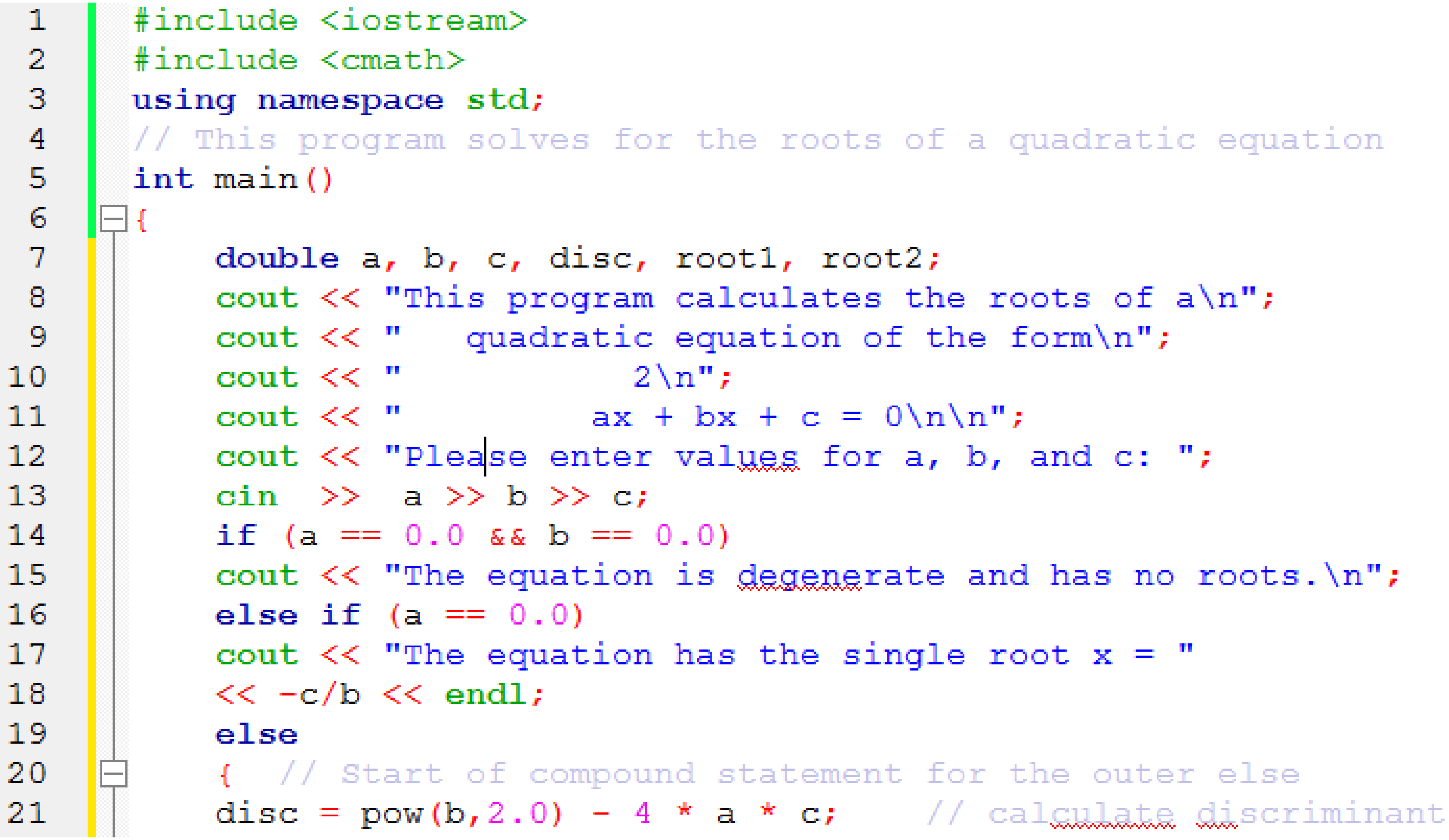
Step 2: Develop a Solution

- Display a program purpose message
- Accept user-input values for a, b, and c
- Calculate the two roots
- Display the values of the calculated roots

Step 2: Develop a Solution(Refining)

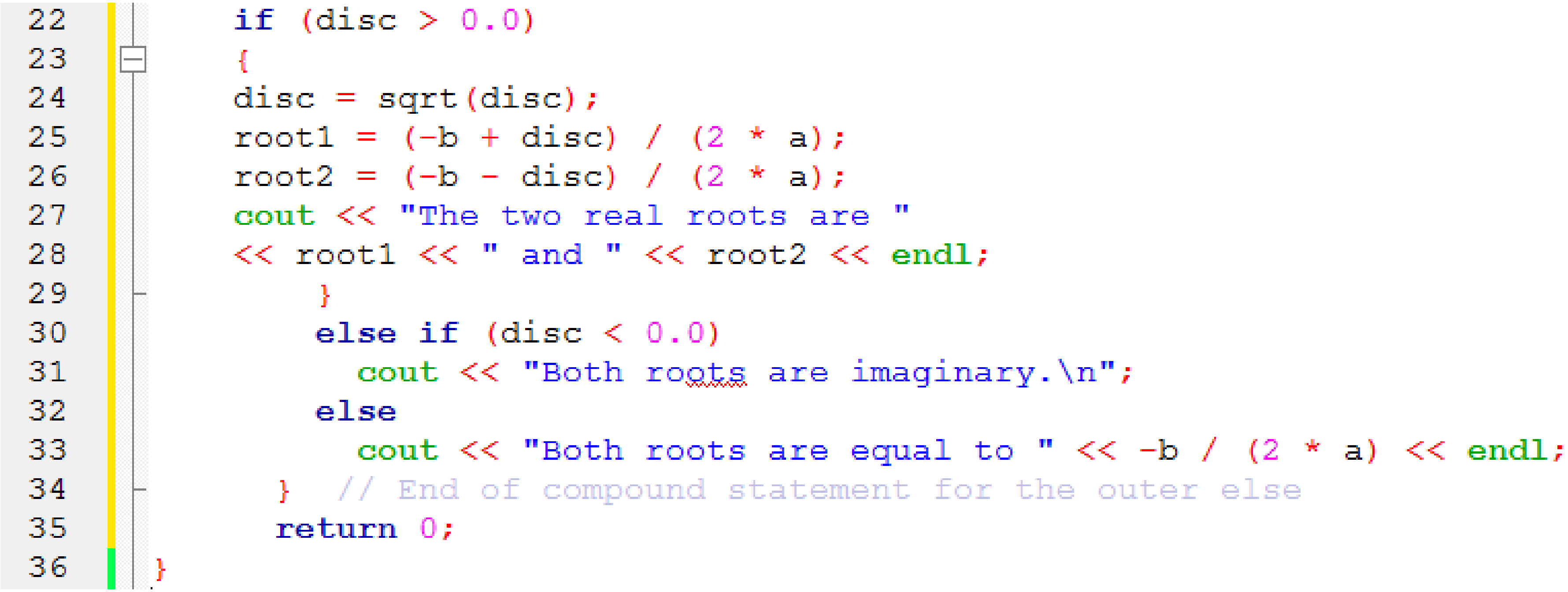
```
Display a program purpose message
Accept user-input values for a, b, and c
if a = 0 and b = 0 then
    Display a message saying that the equation has no solution
else if a = zero then
    Calculate the single root equal to -c/b
    Display the single root
else
    Calculate the discriminant
    If the discriminant > 0 then
        Solve for both roots using the given formulas
        Display the two roots
    Else If the discriminant < 0 then
        Display a message that there are no real roots
    Else
        Calculate the repeated root equal to -b/(2a)
        Display the repeated root
    End If
end if
```

Step 3: Code the Solution



```
#include <iostream>
#include <cmath>
using namespace std;
// This program solves for the roots of a quadratic equation
int main()
{
    double a, b, c, disc, root1, root2;
    cout << "This program calculates the roots of a\n";
    cout << "   quadratic equation of the form\n";
    cout << "           2\n";
    cout << "         ax + bx + c = 0\n\n";
    cout << "Please enter values for a, b, and c: ";
    cin  >>  a >> b >> c;
    if (a == 0.0 && b == 0.0)
    cout << "The equation is degenerate and has no roots.\n";
    else if (a == 0.0)
    cout << "The equation has the single root x = "
    << -c/b << endl;
    else
    {  // Start of compound statement for the outer else
    disc = pow(b,2.0) - 4 * a * c;    // calculate discriminant
```

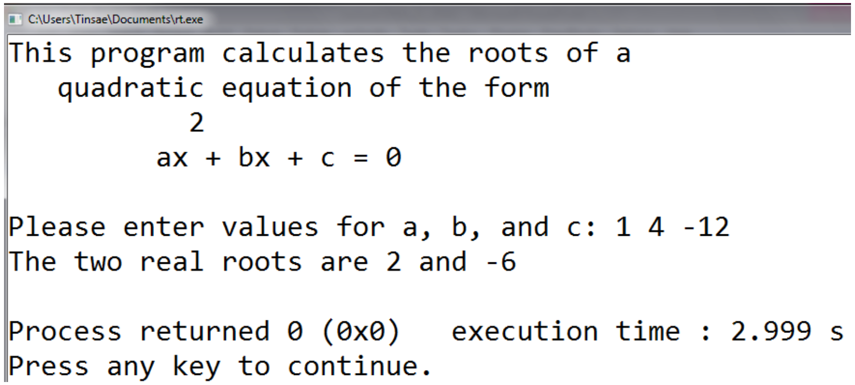
Step 3: Code the Solution



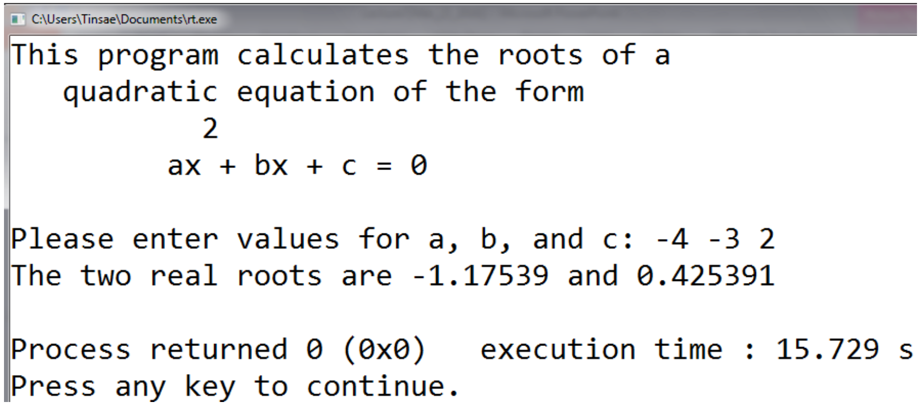
```
    if (disc > 0.0)
    {
    disc = sqrt(disc);
    root1 = (-b + disc) / (2 * a);
    root2 = (-b - disc) / (2 * a);
    cout << "The two real roots are "
    << root1 << " and " << root2 << endl;
        }
        else if (disc < 0.0)
          cout << "Both roots are imaginary.\n";
        else
          cout << "Both roots are equal to " << -b / (2 * a) << endl;
      }  // End of compound statement for the outer else
      return 0;
}
```

Step 4: Test and Correct the Program

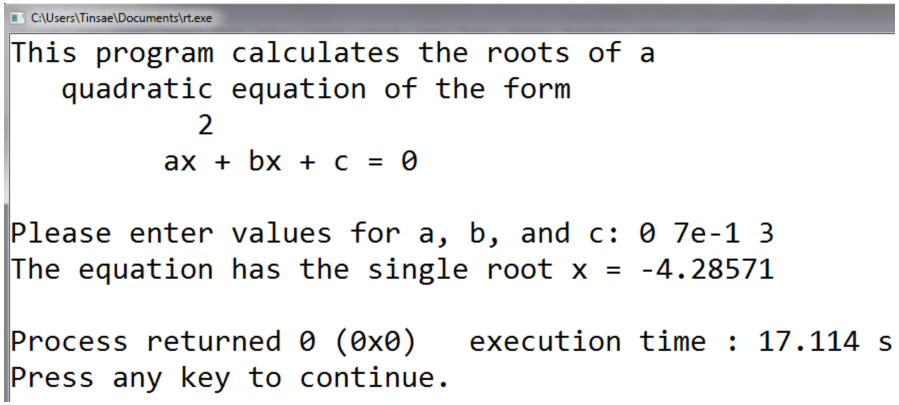
- Test it with different inputs
- Modify the program to show imaginary r oots



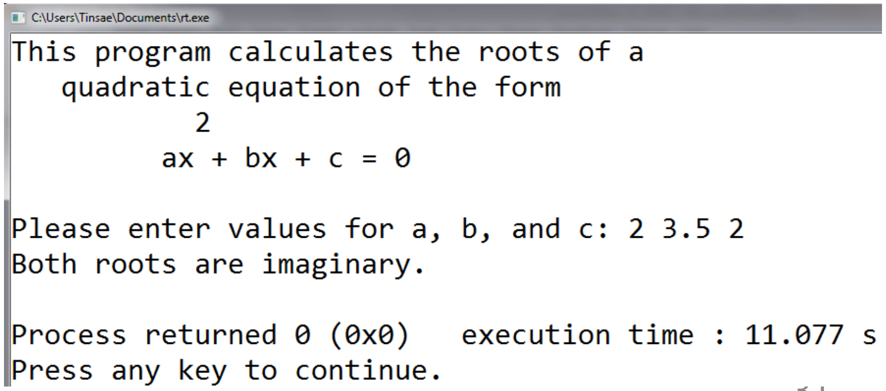
```
This program calculates the roots of a
   quadratic equation of the form
           2
         ax + bx + c = 0

Please enter values for a, b, and c: 1 4 -12
The two real roots are 2 and -6

Process returned 0 (0x0)   execution time : 2.999 s
Press any key to continue.
```



```
This program calculates the roots of a
   quadratic equation of the form
           2
         ax + bx + c = 0

Please enter values for a, b, and c: -4 -3 2
The two real roots are -1.17539 and 0.425391

Process returned 0 (0x0)   execution time : 15.729 s
Press any key to continue.
```



```
This program calculates the roots of a
   quadratic equation of the form
           2
         ax + bx + c = 0

Please enter values for a, b, and c: 0 7e-1 3
The equation has the single root x = -4.28571

Process returned 0 (0x0)   execution time : 17.114 s
Press any key to continue.
```



```
This program calculates the roots of a
   quadratic equation of the form
           2
         ax + bx + c = 0

Please enter values for a, b, and c: 2 3.5 2
Both roots are imaginary.

Process returned 0 (0x0)   execution time : 11.077 s
Press any key to continue.
```

Basic Loop Structures

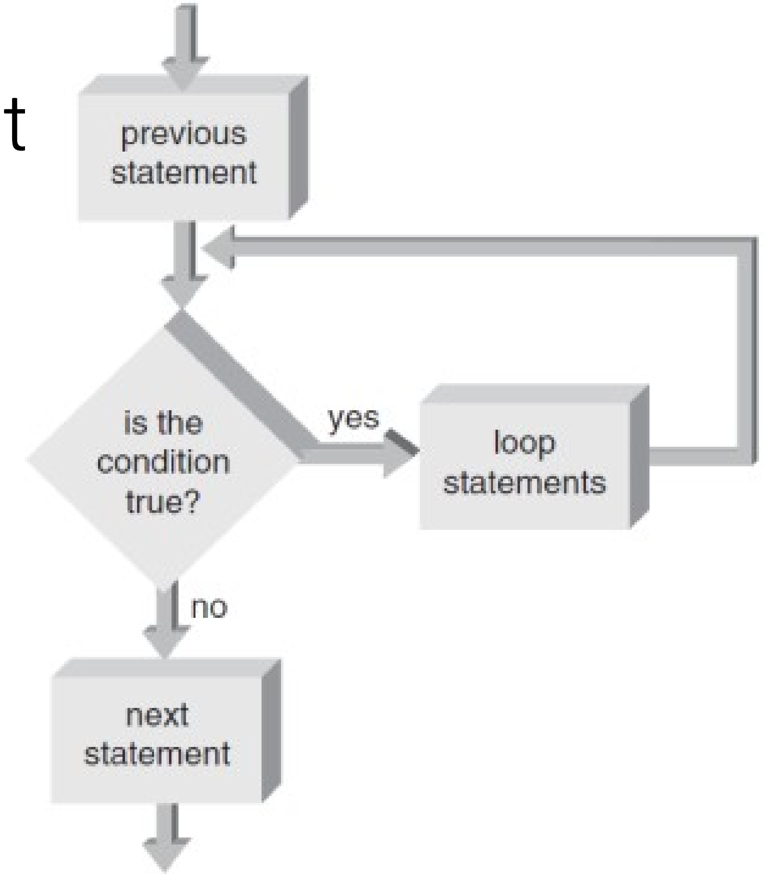
- Repetition structure has four required elements:
  - Repetition statement
  - Condition to be evaluated
  - Initial value for the condition
  - Loop termination
- Repetition statements include:
  - while
  - for
  - do while

Basic Loop Structures

- The condition can be tested
  - At the beginning or Pretest
  - At the end or Posttest
- Something in the loop body must cause the condition to change
  - to avoid an infinite loop , which never t erminates

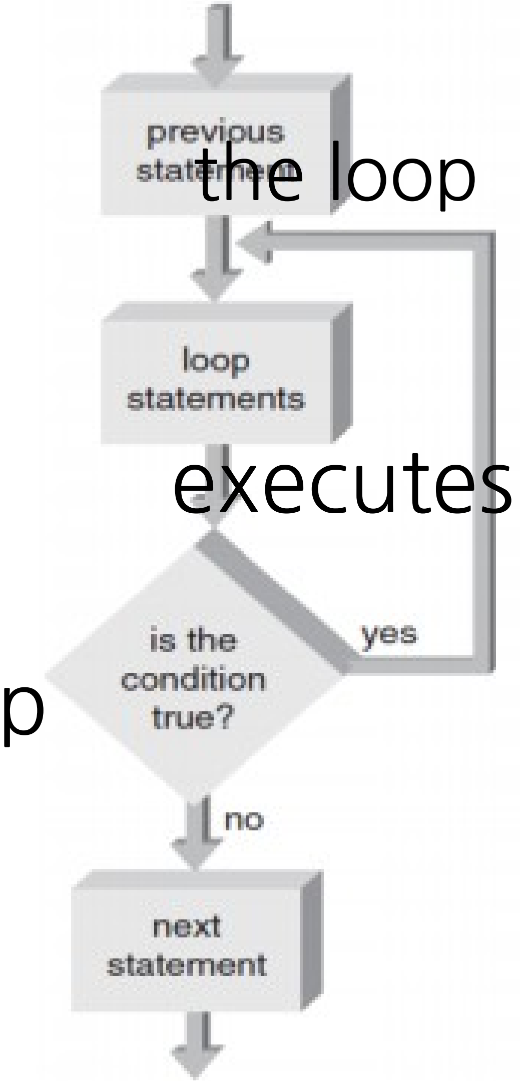
Pretest and Posttest Loops

- Pretest loop:
  - Condition is tested first
  - if false,
    - statements in the loop e never executed
- while and for loops st loops



Pretest and Posttest Loops

- Posttest loop:
  - Condition is tested after the loop body statements are executed;
- loop body always executes at least once
- do while is a posttest loop



Fixed-Count versus Variable Condition Loops

- Fixed-count loop: Loop is processed for a fixed number of repetitions
- Variable-condition loop: Number of repetitions depends on the value of a variable

while loops

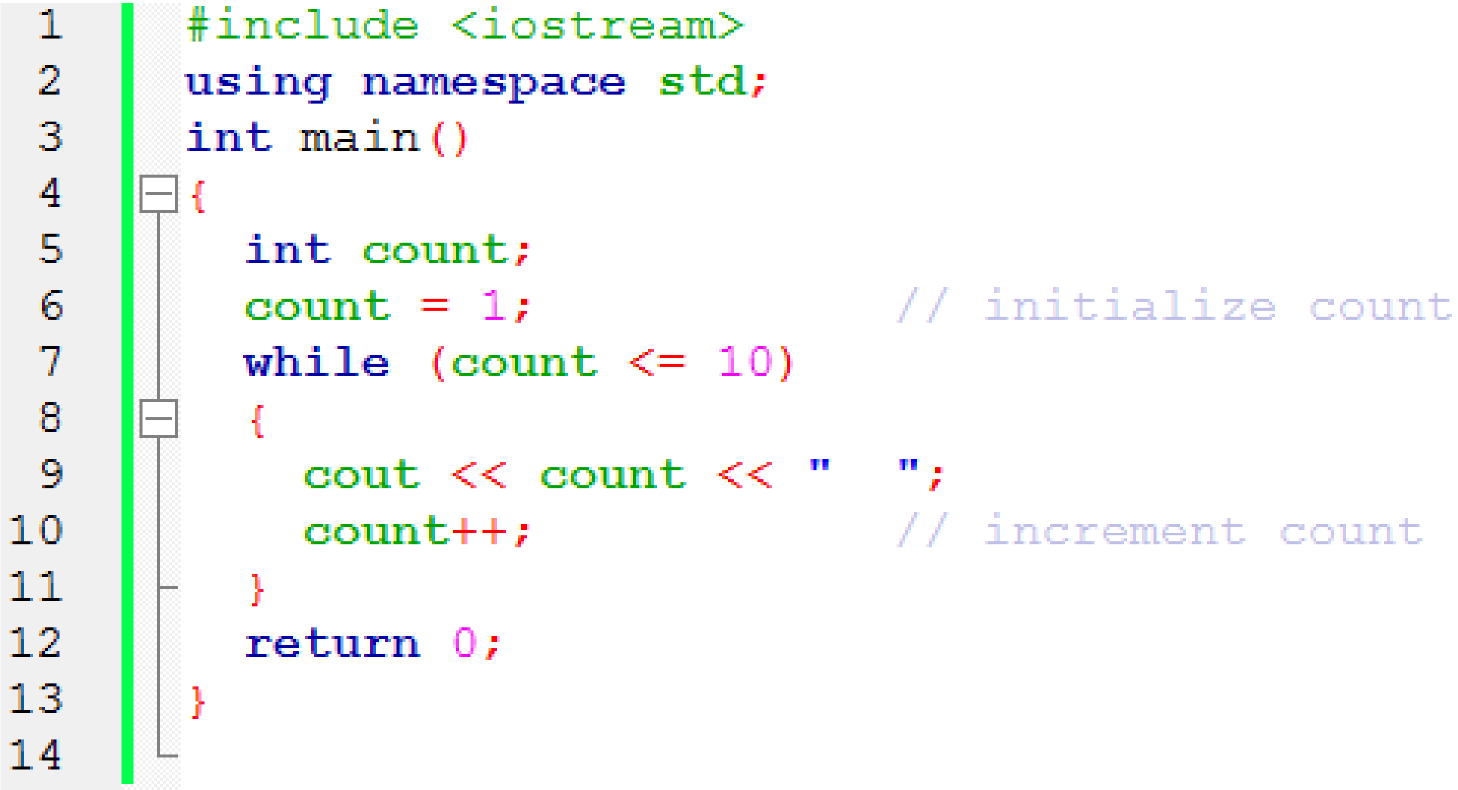
- while **statement** is used to create a while loop

- Syntax:

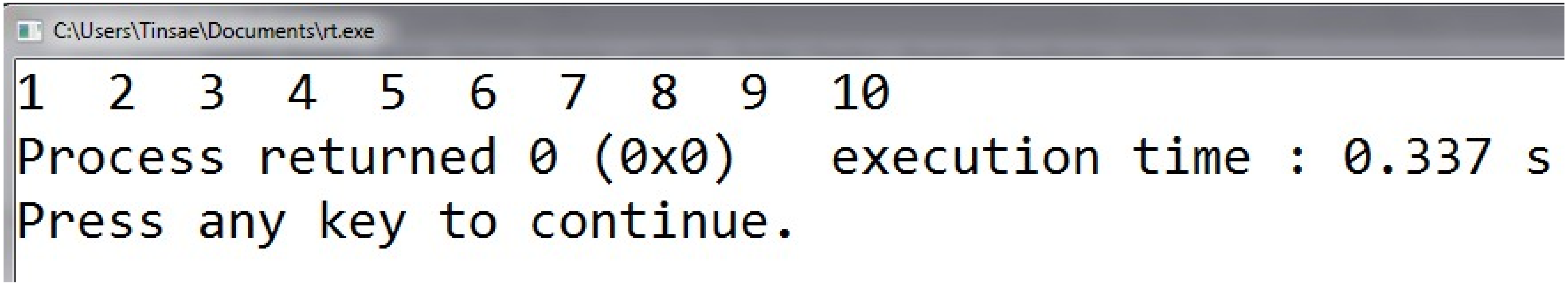
```
while (expression)
        statement;
```

- Statements following the expressions are executed as long as the expression condition remains true
- A non-zero value is true

while loop example



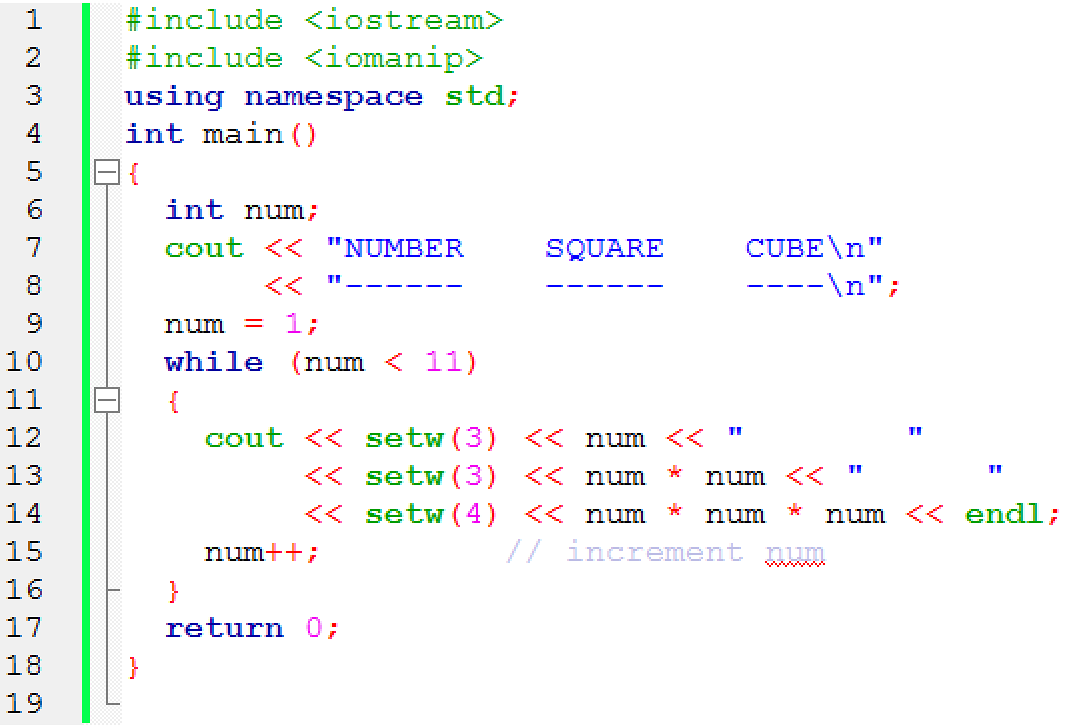
```
#include <iostream>
using namespace std;
int main()
{
   int count;
   count = 1;                 // initialize count
   while (count <= 10)
   {
      cout << count << "   ";
      count++;                // increment count
   }
   return 0;
}
```



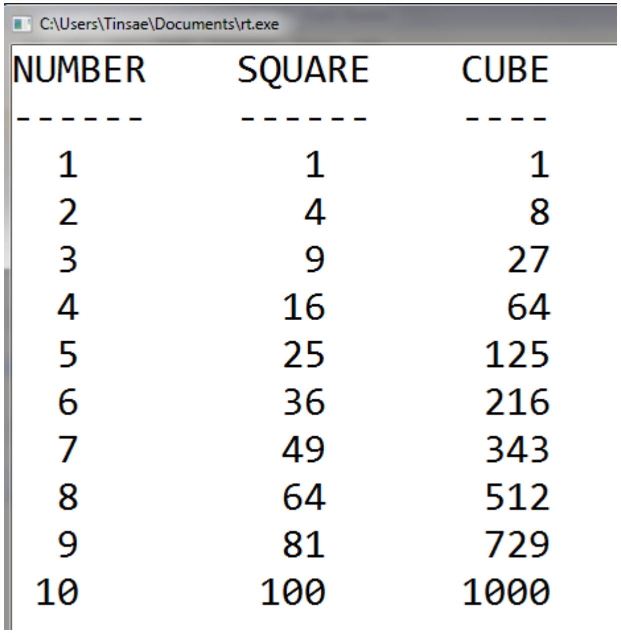
```
1   2   3   4   5   6   7   8   9   10
Process returned 0 (0x0)   execution time : 0.337 s
Press any key to continue.
```

while loop example

- A program to display the square and cube of numbers 1 up to 10



```cpp
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
   int num;
   cout << "NUMBER     SQUARE     CUBE\n"
        << "------     ------     ----\n";
   num = 1;
   while (num < 11)
   {
      cout << setw(3) << num << "         "
           << setw(3) << num * num << "       "
           << setw(4) << num * num * num << endl;
      num++;              // increment num
   }
   return 0;
}
```



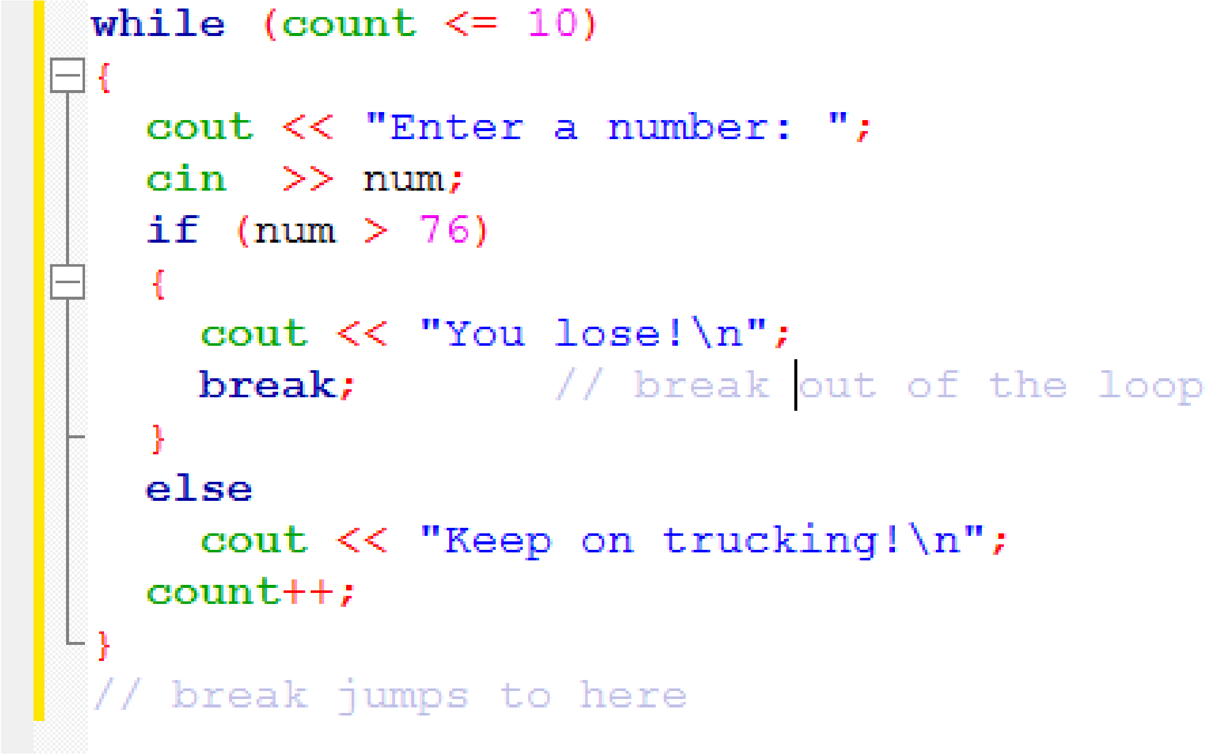
C:\Users\Tinsae\Documents\rt.exe

```
NUMBER      SQUARE      CUBE
------      ------      ----
  1            1           1
  2            4           8
  3            9          27
  4           16          64
  5           25         125
  6           36         216
  7           49         343
  8           64         512
  9           81         729
 10          100        1000
```

break statement

- Forces an immediate break, or exit, from switch, while, for, and do-while statements
- Violates pure structured programming
  - but is useful for breaking out of loops whe n an unusual condition is detected

break statement example



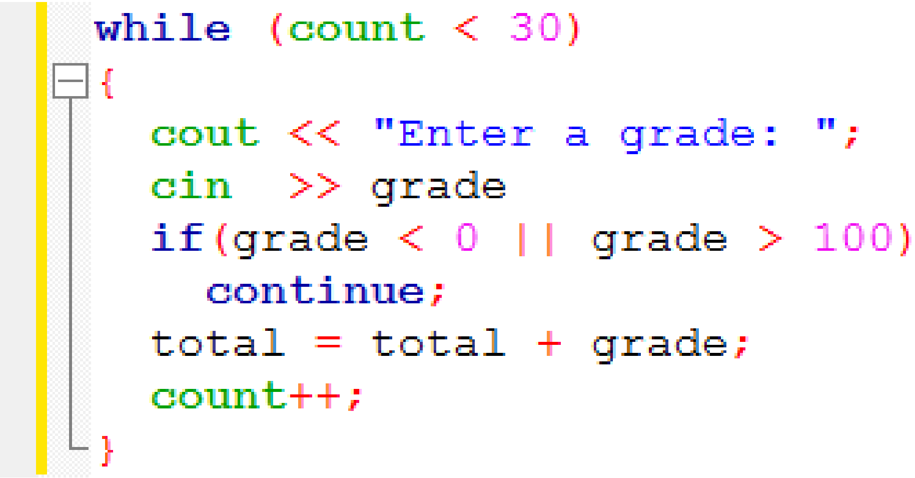
```
while (count <= 10)
{
  cout << "Enter a number: ";
  cin  >> num;
  if (num > 76)
  {
    cout << "You lose!\n";
    break;          // break out of the loop
  }
  else
    cout << "Keep on trucking!\n";
  count++;
}
// break jumps to here
```

continue statement

- Applies to while, do-while , and for statements;
- causes the next iteration of the loop to begin immediately
- Useful for skipping over data that should not be processed in this iteration
  - while staying within the loop

continue statement example

- A continue statement where invalid grades are ignored, and only valid grades are added to the total:



```
while (count < 30)
{
    cout << "Enter a grade: ";
    cin   >> grade
    if(grade < 0 || grade > 100)
      continue;
    total = total + grade;
    count++;
}
```

for loops

- A loop with a fixed count condition that handles alteration of the condition
- Syntax:

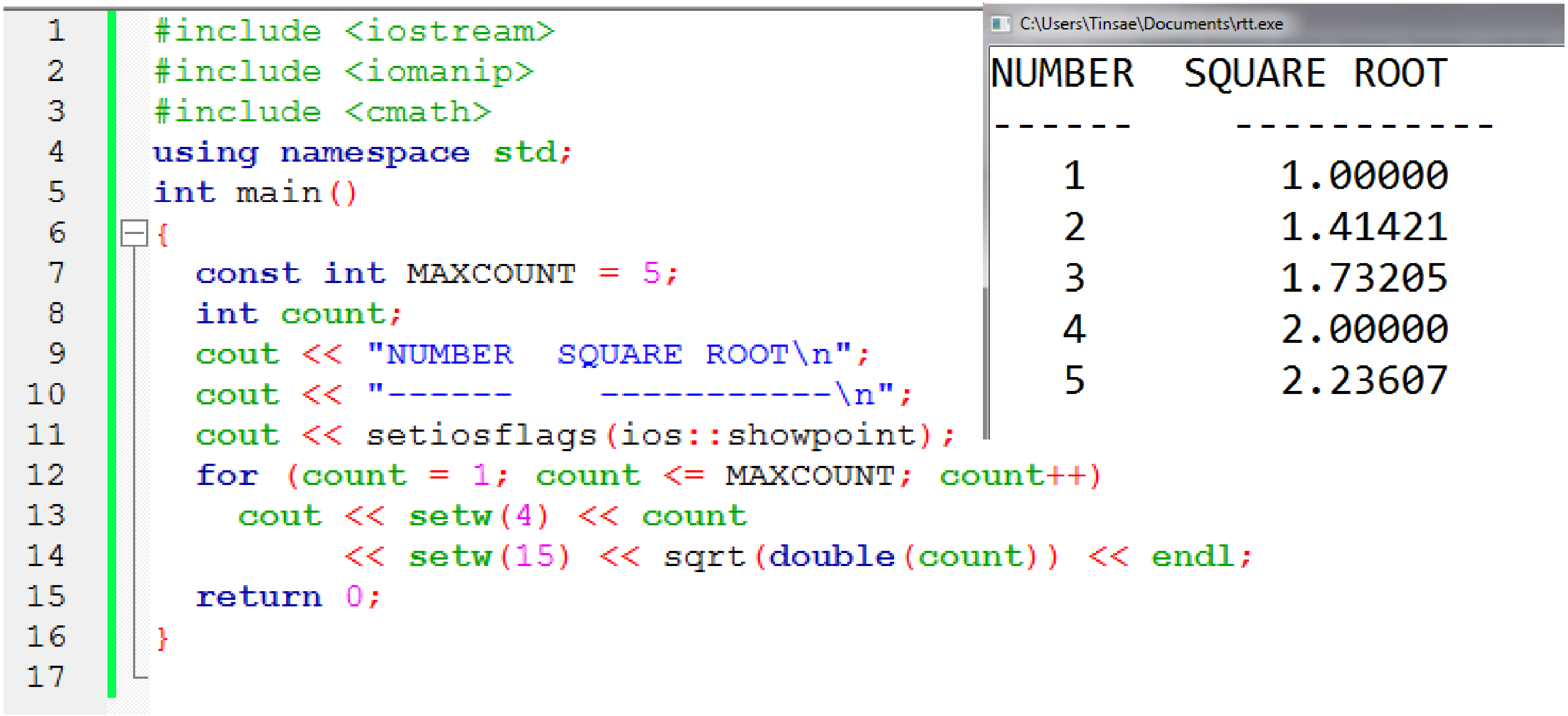
  for (initializing list; expression; altering list)

  statement;
- **Initializing list**
  - Sets the starting value of a counter
- **Expression**
  - Contains the maximum or minimum value the counter can have;
  - determines when the loop is finished
- **Altering list**
  - Provides the increment value that is added or subtracted from the counter in each iteration of the loop

for loops

- If initializing list is missing
  - the counter initial value must be provided p rior to entering the for loop
- If altering list is missing
  - the counter must be altered in the loop   b ody
- Omitting the expression will result in an infinite loop

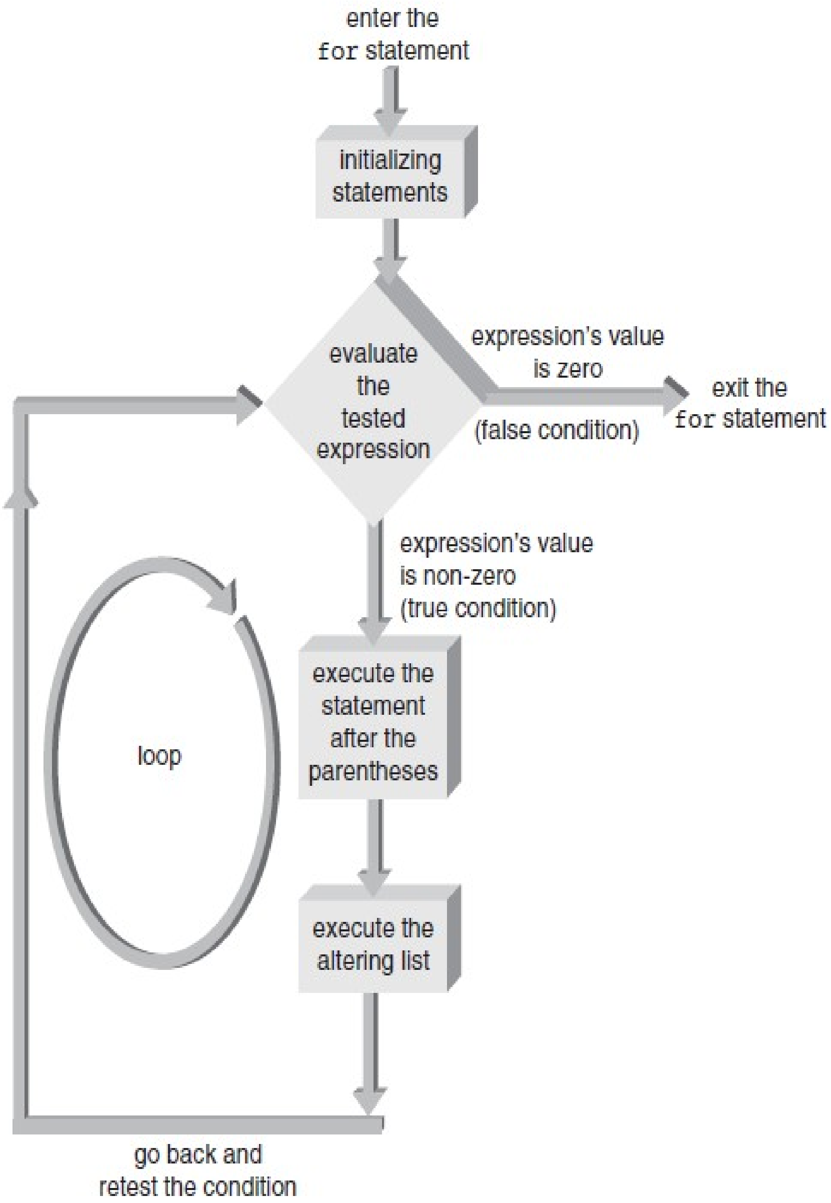
for loop example



```cpp
#include <iostream>
#include <iomanip>
#include <cmath>
using namespace std;
int main()
{
  const int MAXCOUNT = 5;
  int count;
  cout << "NUMBER  SQUARE ROOT\n";
  cout << "------    -----------\n";
  cout << setiosflags(ios::showpoint);
  for (count = 1; count <= MAXCOUNT; count++)
    cout << setw(4) << count
         << setw(15) << sqrt(double(count)) << endl;
  return 0;
}
```

```
NUMBER  SQUARE ROOT
------    -----------
   1        1.00000
   2        1.41421
   3        1.73205
   4        2.00000
   5        2.23607
```

For Loop Flow of Control

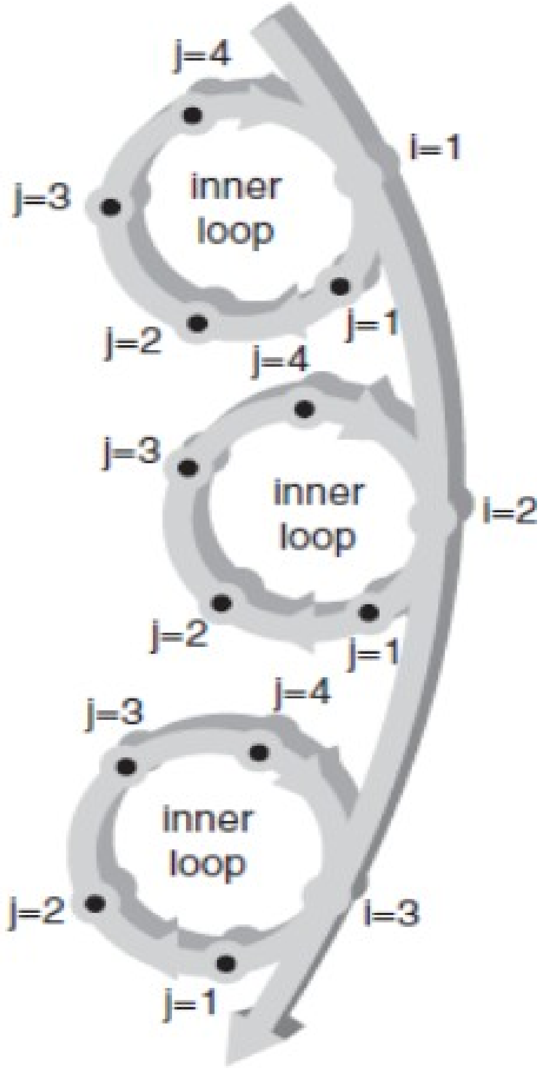


Nested Loops

- A loop contained within another loop
  - All statements of the inner loop must be completely contained within the outer loop; no overlap allowed
  - Different variables must be used to control each loop
  - For each single iteration of the outer loop, the inner loop runs through all of its iterations

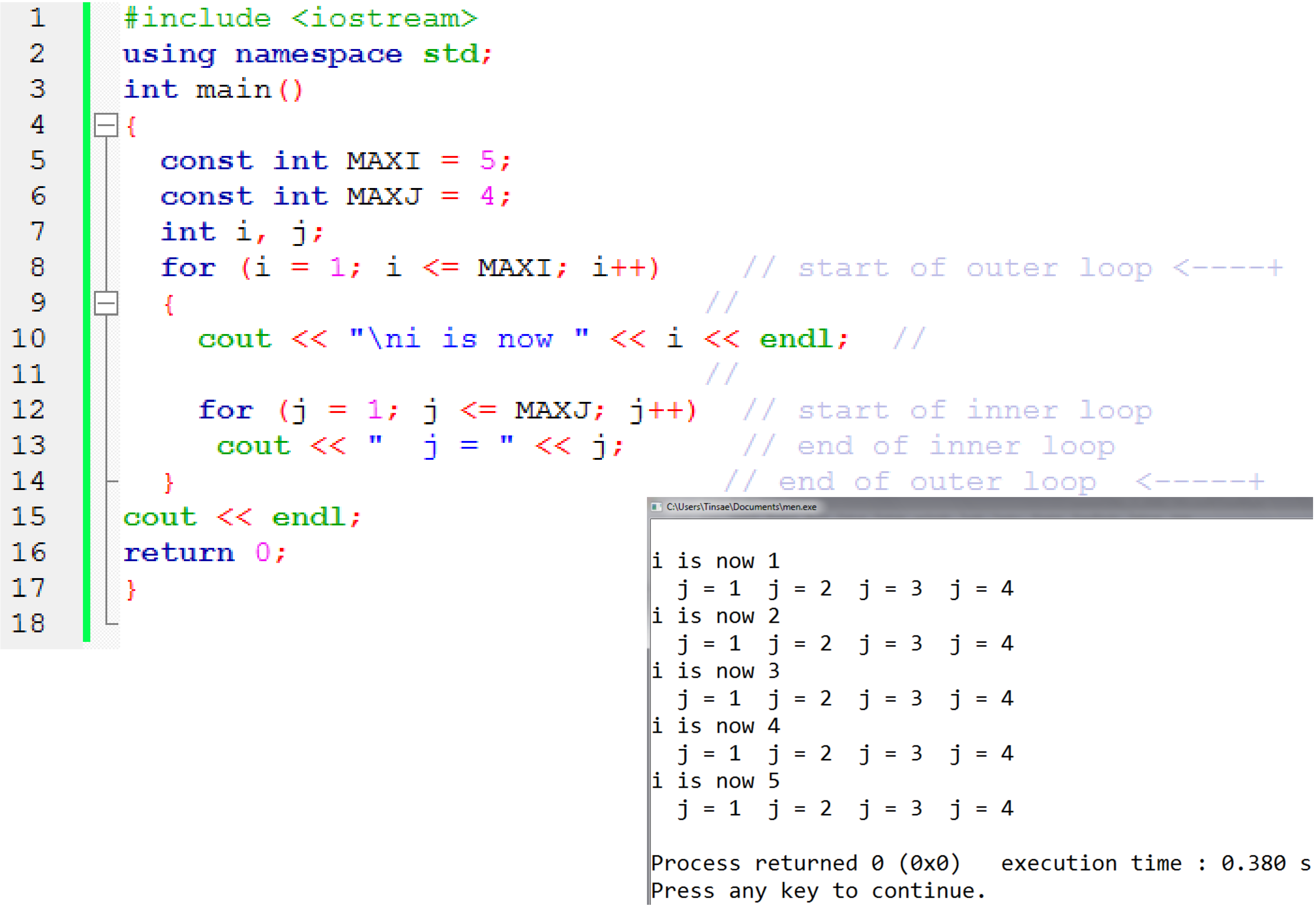
Nested Loops Diagram

- For each i, j loops
- i controls the outer loop. Range is 1-3
- j controls the inner loop. Range is 1-4



Nested Loops Example

ASTU



```
#include <iostream>
using namespace std;
int main()
{
   const int MAXI = 5;
   const int MAXJ = 4;
   int i, j;
   for (i = 1; i <= MAXI; i++)      // start of outer loop <----+
   {                              //
      cout << "\ni is now " << i << endl;  //
                                  //
      for (j = 1; j <= MAXJ; j++)  // start of inner loop
       cout << "  j = " << j;      // end of inner loop
   }                              // end of outer loop  <----+
cout << endl;
return 0;
}
```

C:\Users\Tinsae\Documents\men.exe

```

i is now 1
  j = 1  j = 2  j = 3  j = 4
i is now 2
  j = 1  j = 2  j = 3  j = 4
i is now 3
  j = 1  j = 2  j = 3  j = 4
i is now 4
  j = 1  j = 2  j = 3  j = 4
i is now 5
  j = 1  j = 2  j = 3  j = 4

Process returned 0 (0x0)   execution time : 0.380 s
Press any key to continue.
```

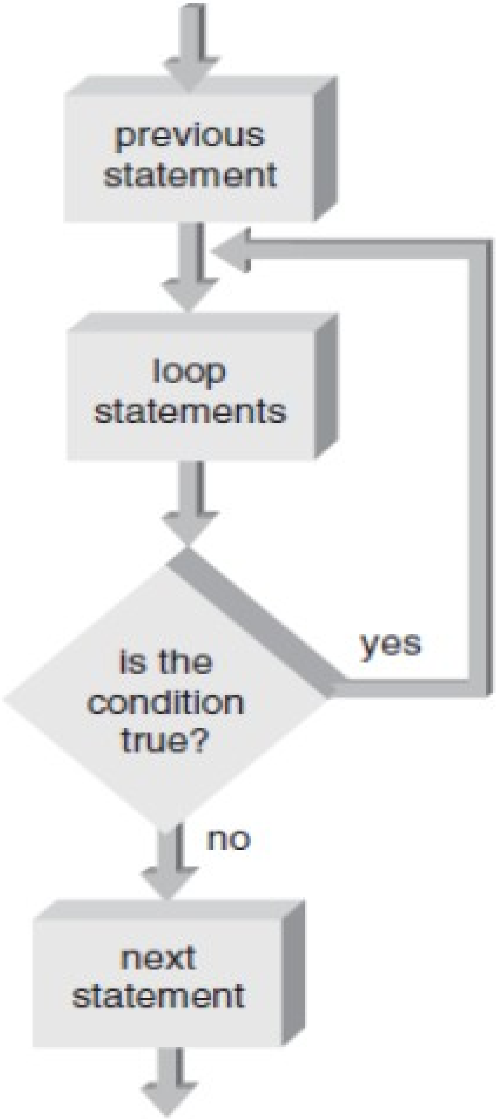
do while loop

- Loop continues while the condition is true
- Condition is tested at the end of the loop
- Syntax:

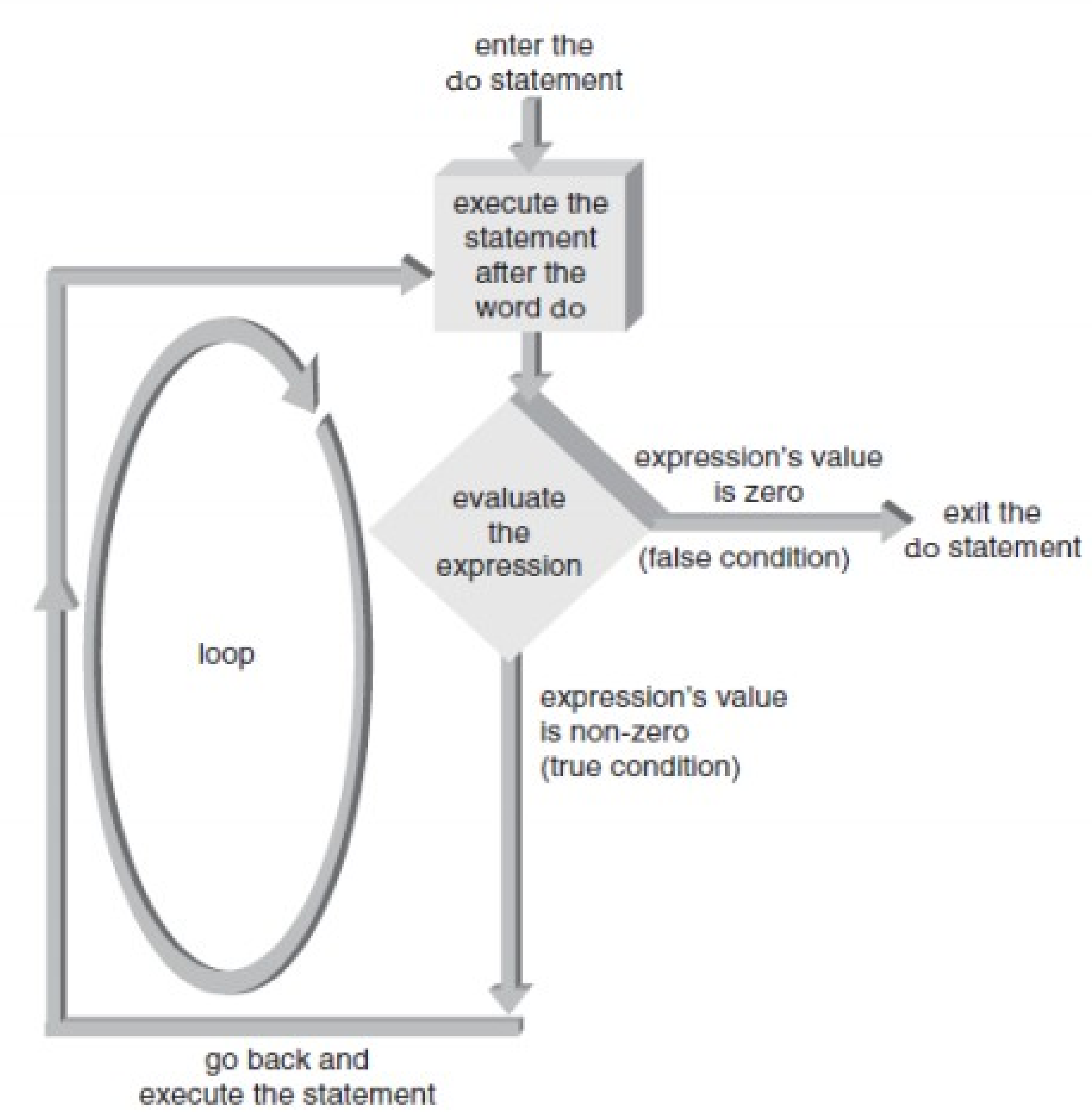
```
do
    statement;
while (expression);
```

- All statements are executed at least once in a posttest loop

do while loop flowchart

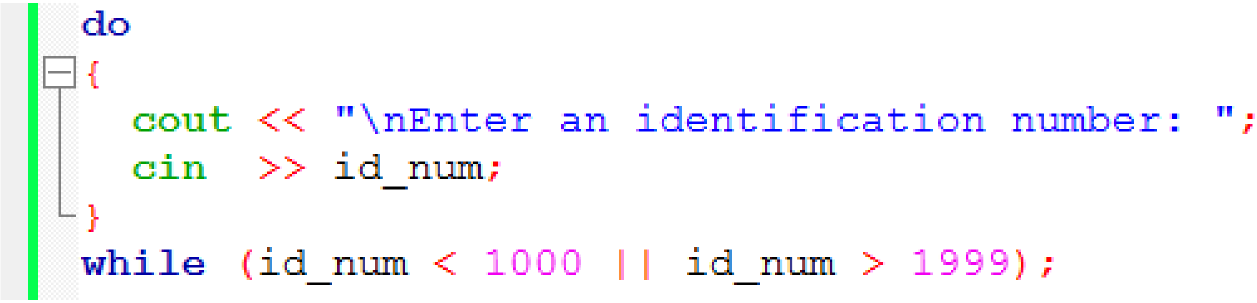


do while loop flow of control



common use of do while loop

- Useful in filtering user-entered input and providing data validation checks

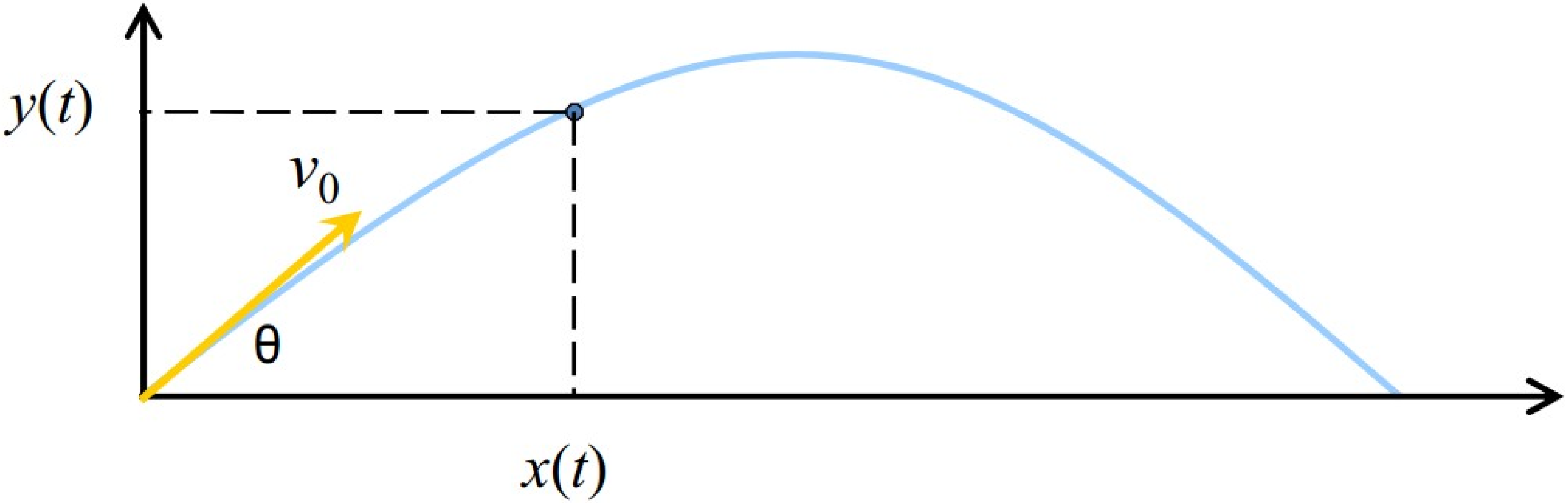


```
do
{
   cout << "\nEnter an identification number: ";
   cin  >> id_num;
}
while (id_num < 1000 || id_num > 1999);
```

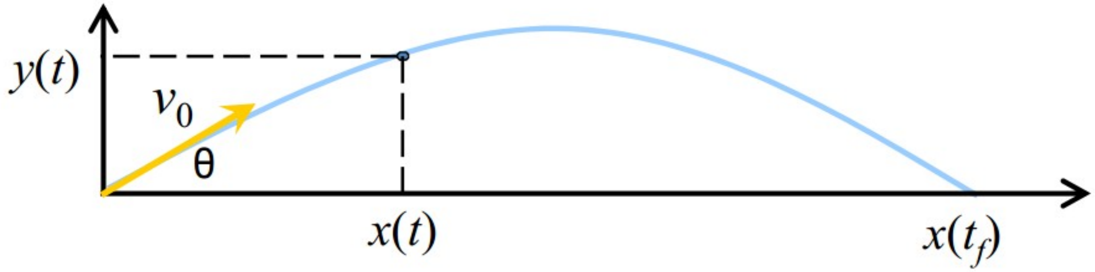
- Can enhance with if-else statement

Case Study: Flight of a ball

- Given a throwing velocity and angle θ , find the flying range.
- What is the throwing angle which results in maximum flying range?
- Without using calculus



Case Study: Flight of a ball



- Coordinate of the flying trajectory

$$x(t) = v_{0xt}$$

$$y(t) = v_{0y}\mathrm{t} + \frac{1}{2}gt^2$$

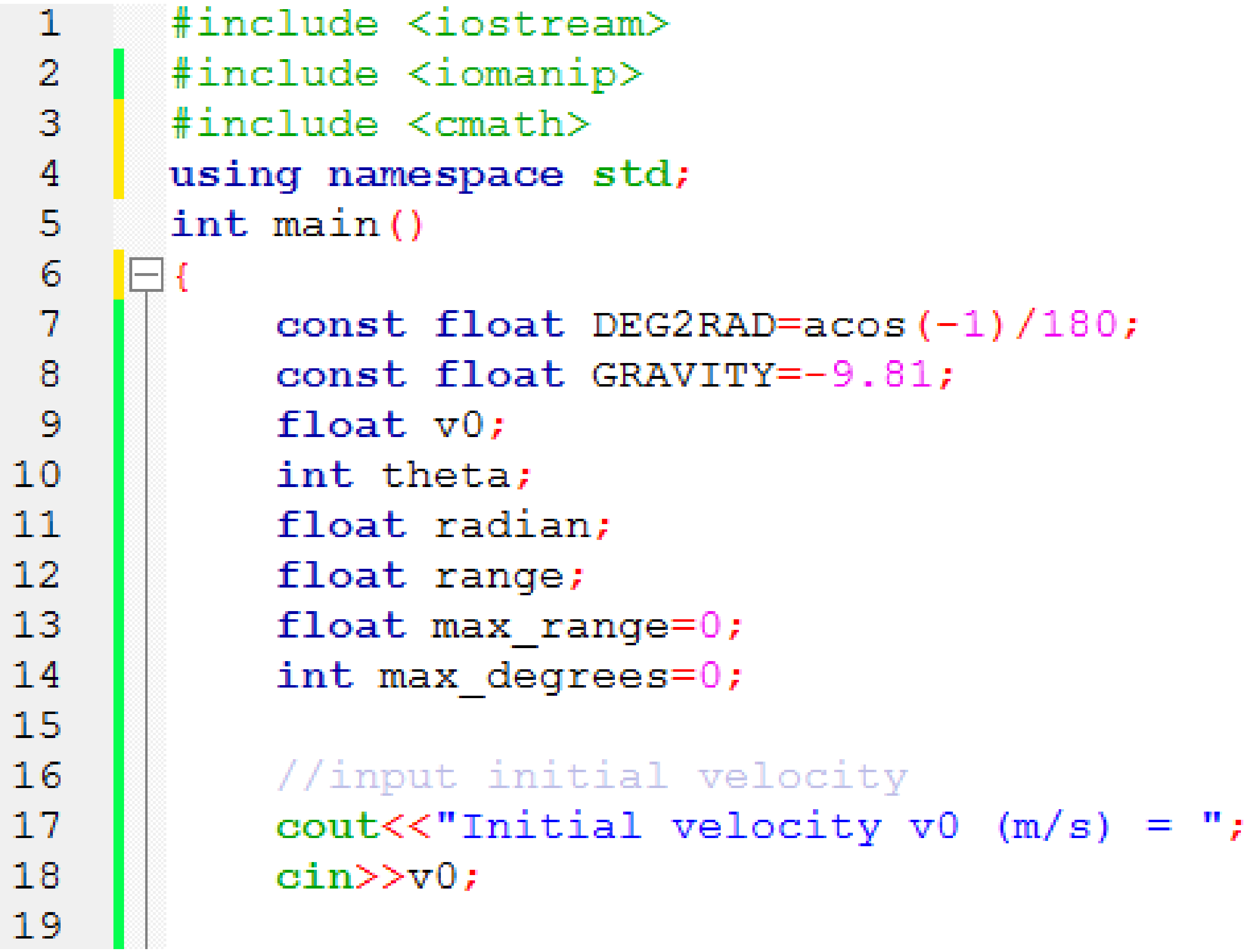
- The ball will hit the ground when $y(t)$ =0

$$0 = v_{0y}\mathrm{t} + \frac{1}{2}gt^2$$

- The time interval of the flight
- $t_f = -\frac{2v_{0y}}{g}$

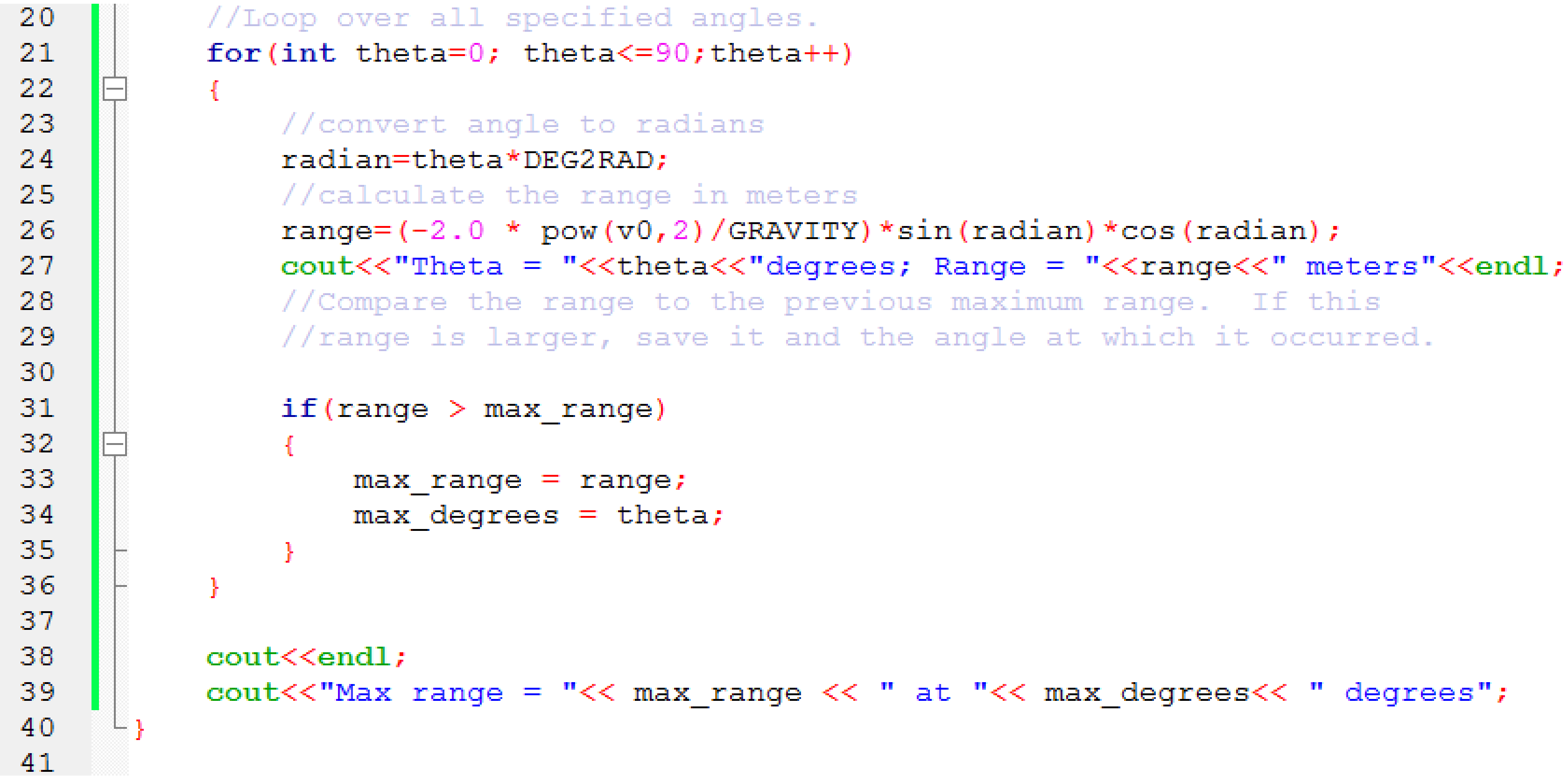
- The time interval of the flight
- $x(t_f) = \mathrm{x}\left(-\frac{v_{oy}}{g}\right) = v_{0x}\left(-\frac{2v_{oy}}{g}\right) = \frac{2{v_0}^2}{g}\cos 2\theta = 0$

Case Study: Flight of a ball



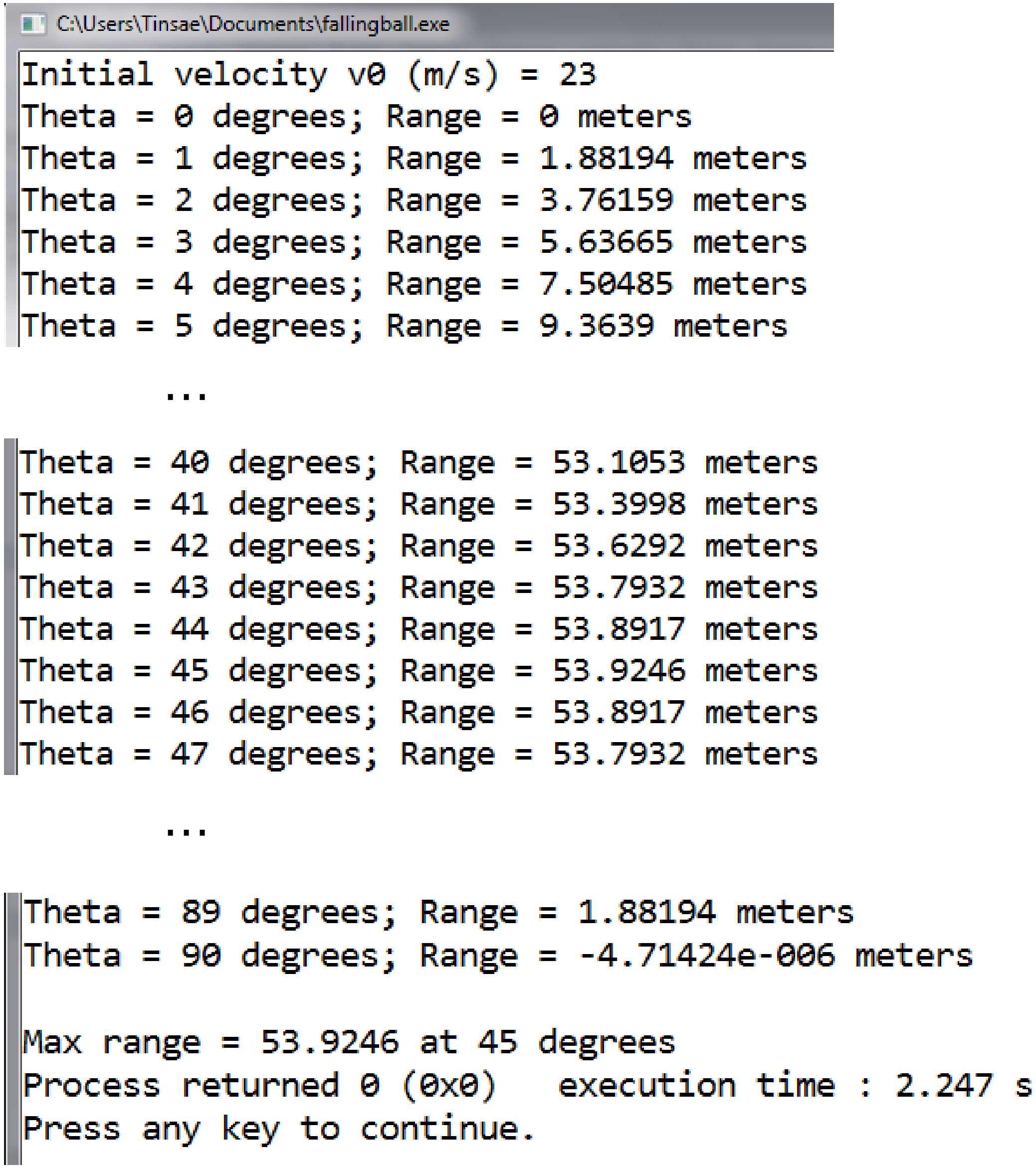
```
#include <iostream>
#include <iomanip>
#include <cmath>
using namespace std;
int main()
{
    const float DEG2RAD=acos(-1)/180;
    const float GRAVITY=-9.81;
    float v0;
    int theta;
    float radian;
    float range;
    float max_range=0;
    int max_degrees=0;

    //input initial velocity
    cout<<"Initial velocity v0 (m/s) = ";
    cin>>v0;

```

Case Study: Flight of a ball



```cpp
    //Loop over all specified angles.
    for(int theta=0; theta<=90;theta++)
    {
        //convert angle to radians
        radian=theta*DEG2RAD;
        //calculate the range in meters
        range=(-2.0 * pow(v0,2)/GRAVITY)*sin(radian)*cos(radian);
        cout<<"Theta = "<<theta<<"degrees; Range = "<<range<<" meters"<<endl;
        //Compare the range to the previous maximum range.  If this
        //range is larger, save it and the angle at which it occurred.

        if(range > max_range)
        {
            max_range = range;
            max_degrees = theta;
        }
    }

    cout<<endl;
    cout<<"Max range = "<< max_range << " at "<< max_degrees<< " degrees";
}

```

Case Study: Flight of a ball



C:\Users\Tinsae\Documents\fallingball.exe

```
Initial velocity v0 (m/s) = 23
Theta = 0 degrees; Range = 0 meters
Theta = 1 degrees; Range = 1.88194 meters
Theta = 2 degrees; Range = 3.76159 meters
Theta = 3 degrees; Range = 5.63665 meters
Theta = 4 degrees; Range = 7.50485 meters
Theta = 5 degrees; Range = 9.3639 meters

       ...

Theta = 40 degrees; Range = 53.1053 meters
Theta = 41 degrees; Range = 53.3998 meters
Theta = 42 degrees; Range = 53.6292 meters
Theta = 43 degrees; Range = 53.7932 meters
Theta = 44 degrees; Range = 53.8917 meters
Theta = 45 degrees; Range = 53.9246 meters
Theta = 46 degrees; Range = 53.8917 meters
Theta = 47 degrees; Range = 53.7932 meters

       ...

Theta = 89 degrees; Range = 1.88194 meters
Theta = 90 degrees; Range = -4.71424e-006 meters

Max range = 53.9246 at 45 degrees
Process returned 0 (0x0)   execution time : 2.247 s
Press any key to continue.
```