CSCI 1540 Introduction to Computing Using C++

Tutorial 5

Xinyi Hu

SHB 1005

xyhu@cse.cuhk.edu.hk

Outline

Debugging

- Why debugging?
- General debug procedure
- Debug with IDE (Visual Studio 2017)

Outline

Debugging

- Why debugging?
- General debug procedure
- Debug with IDE (Visual Studio 2019)

Types of Error

- Syntax error…
  - *Your program won't even compile…*
    - *Missing semicolon ;*
    - Unbalanced parenthesis () {}
    - Missing include statements… (#include <xyz>)

- Logical error
  - Your program can be compiled, but does not work as expected…
    - What's going on?

I have written a "buggy" program, what should I do?

- **1. Locate the bug**
  - What we are going to talk about…
- **2. Fix it!**
  - This part is up to you…

Debugging

- **Debugging** is a methodical process of finding and reducing the number of **bugs**, or **defects**, in a computer program or a piece of electronic hardware, thus making it behave as expected.

Before Debugging Your Program

- **Make sure it is indented!!!**

Indentation

- Indent code according to program structure
- Inner structure => indent deeper
  - Spaces : usually multiple of 2 / 4
  - Tabs: e.g. Visual Studio

- Use the same type of indentation throughout the whole program
  - Don’t mix spaces and tabs

```
int main() {
  int x = 0;
  while (x != -1) {
    cin >> x;
    if ((x % 2) == 0) {
      cout << x;
    }
  }
  return 0;
}
```

Indentation is Important

- There is an error in both code below
  - Which one is easier to spot?

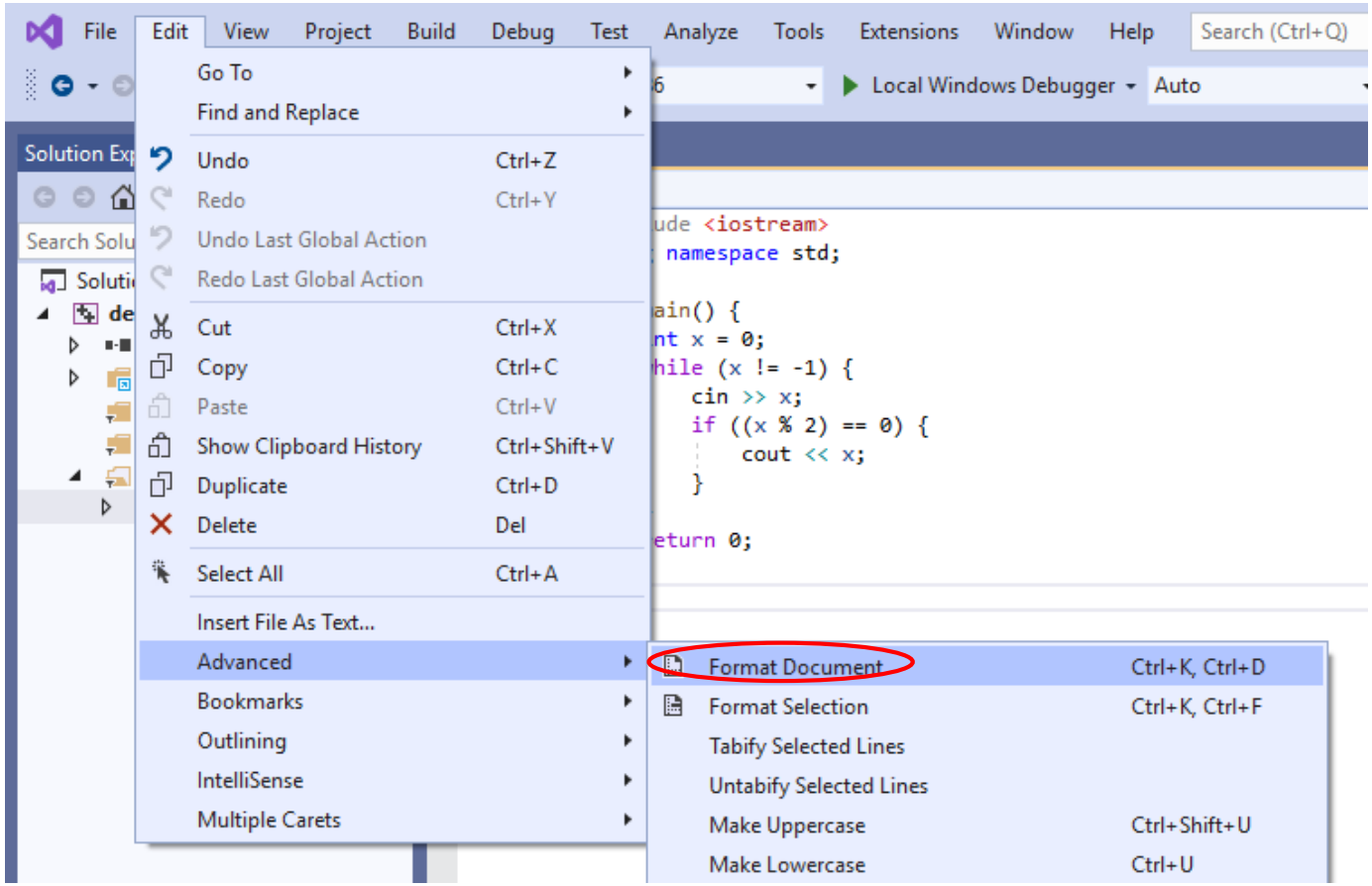
```
int main() {
  int x;
  cin >> x;
  while (x > 0) {
    if ((x % 2) == 0){
      cout << x;
    x--;
  }
}
```

```
int main() {
int x;
cin >> x;
while (x > 0) {
if ((x % 2) == 0){
cout << x;
x--;
}
}
```

Easier to recognize Unbalanced parenthesis

Auto Correction in VS Community 2019

- VS Community 2019 can auto correct indentation
  - EDIT -> Advanced -> Format Document



Auto Correction in VS Community 2019

- VS Community 2019 can auto correct indentation
  - EDIT -> Advanced -> Format Document

```
#include <iostream>
using namespace std;

int main()
{
int x;
cin >> x;
while (x > 0) {
if ((x % 2) == 0) {
cout << x;
}
x--;
}
return 0;
}
```

```
#include <iostream>
using namespace std;

int main()
{
    int x;
    cin >> x;
    while (x > 0) {
        if ((x % 2) == 0) {
            cout << x;
        }
        x--;
    }
    return 0;
}
```

Outline

Debugging

- Why debugging?
- General debug procedure
- Debug with IDE (Visual Studio 2019)

Locating the Bug - Part I

- **General procedure to debug**

Basic Debugging

- **Basic Approaches To Locate Bugs**
  - Select some input values which you know what the output is
  - Decide which variables to **inspect** and where to inspect them
  - **Output the value** of the variables (to see if they match your expected values)

Basic Debugging

- **Typically, a program consists of 3 parts**
  - Input, Calculation and Output

```
// A program to compute interest
double rate, principal, interest;
int months;

// Input
   ...


// Calculation
   ...


// Output
   ...
```

Basic Debugging

- **Print out the values for inspection**

```
// A program to compute interest
double rate, principal, interest;
int months;

// Input
  ...
cout << "Debug (Input): " << rate << " "
     << principal << " " << months << endl;

// Calculation
  ...
cout << "Debug (Cal): " << interest << endl;

// Output
  ...
```

Locating the Bug - Part II

- **Getting help from your IDE**

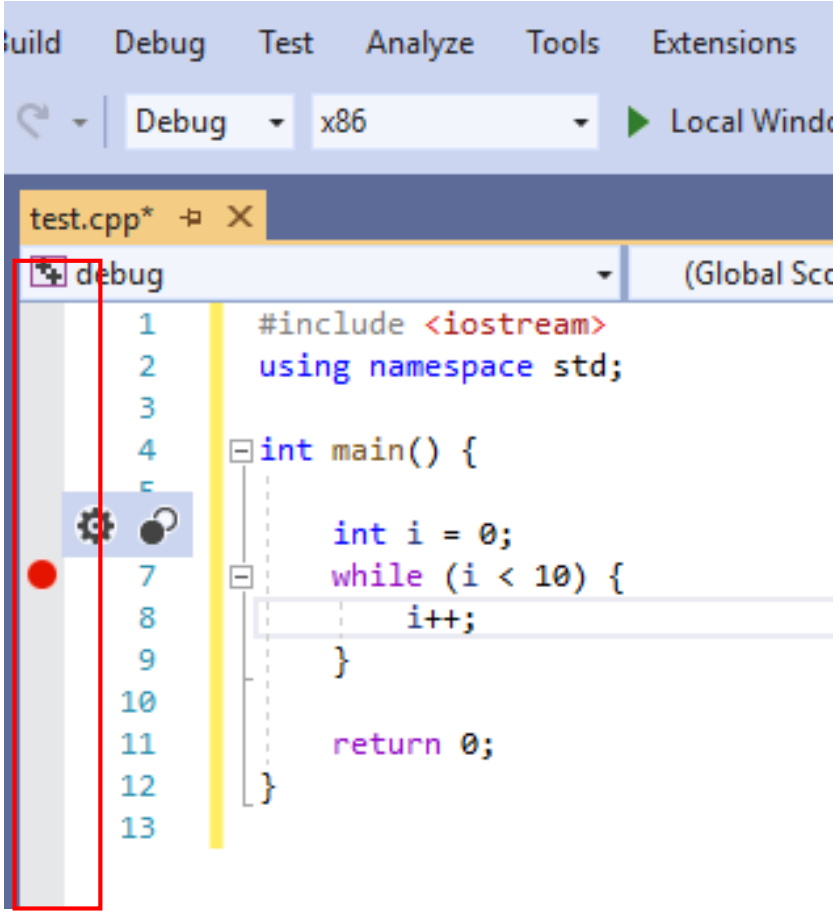
Outline

Debugging

- Why debugging?
- General debug procedure
- Debug with IDE (Visual Studio 2019)

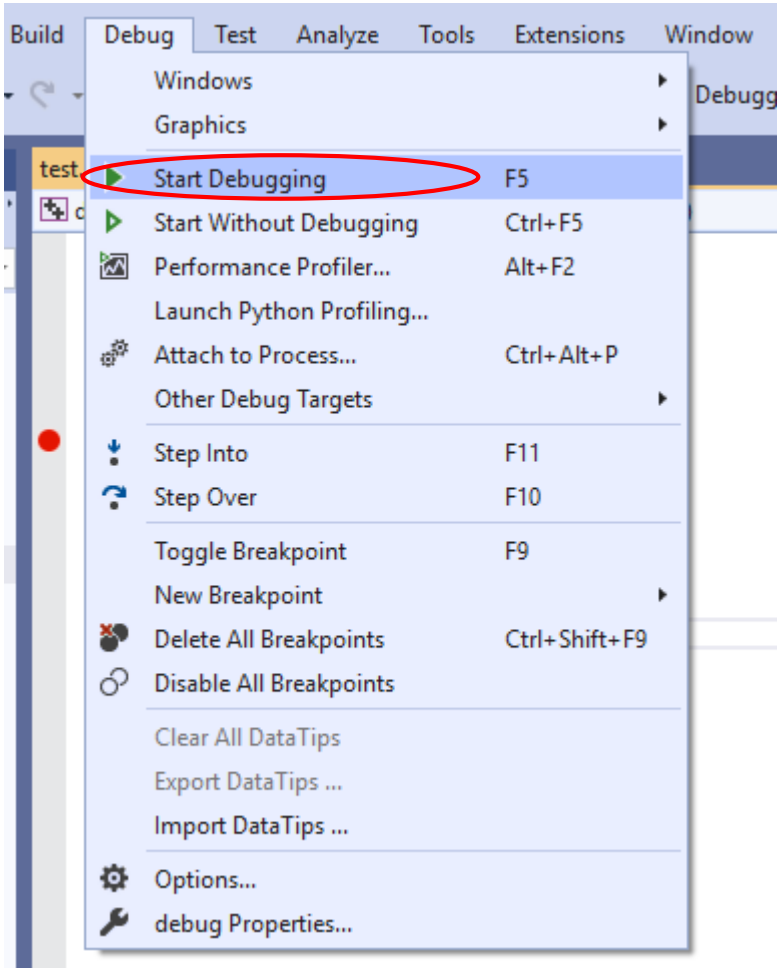
Introduction to "Debugger"

- **Adding break points**
  - Left click in the grey area
  - Left click on a red dot to remove a break point



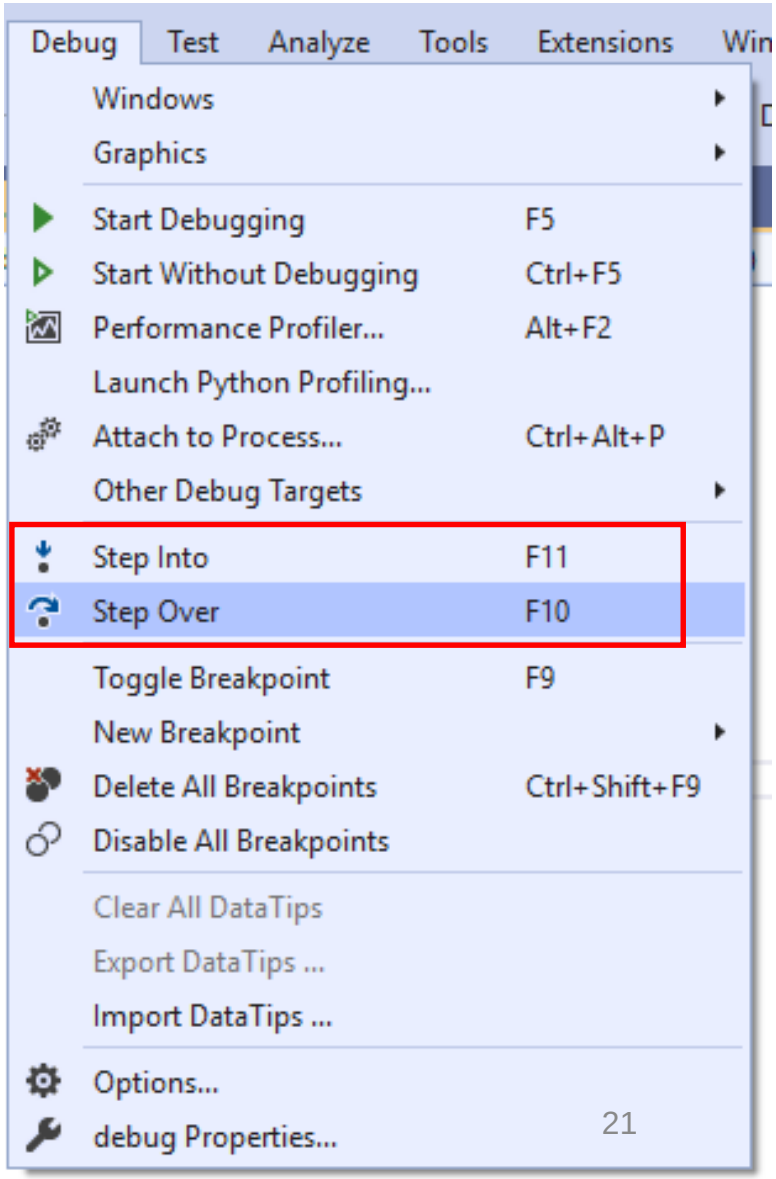
Introduction to "Debugger"

- **Run your program in "Debug mode"!!!**
  - Debug -> Start Debugging, OR
  - Press F5



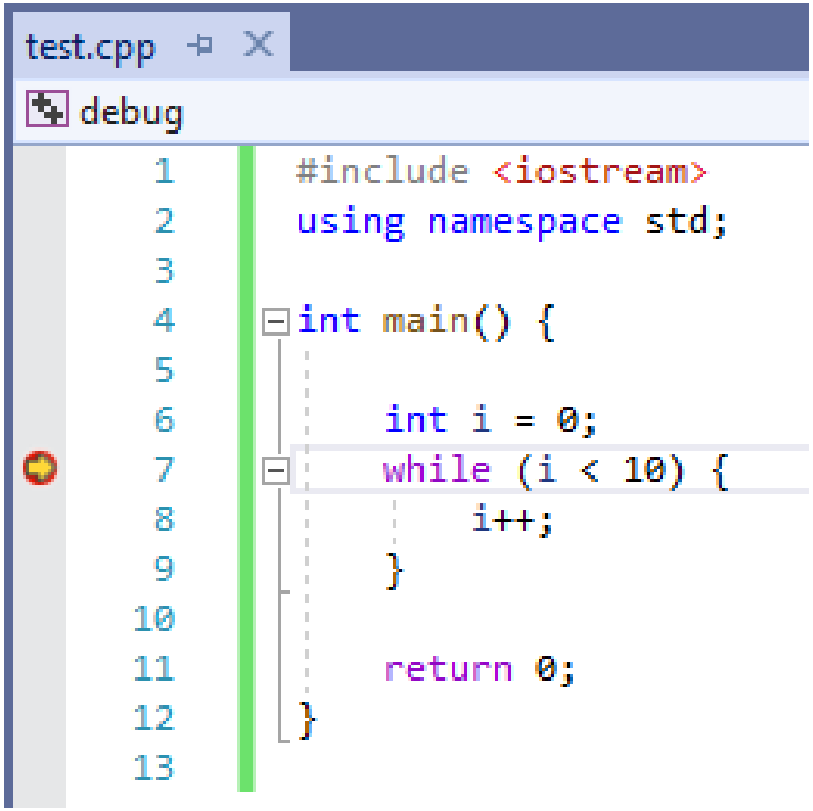
Introduction to "Debugger"

- **Stepping: run program line by line**
  - Step Into (F11)
  - Step Over (F10)
  - Step Out (Shift+F11)



Introduction to "Debugger"

- **Program is paused here**
  - Yellow arrow



Introduction to "Debugger"

- **Press F10 (Step over) to execute the statements**

debug

```
1  #include <iostream>
2  using namespace std;
3
4  int main() {
5
6      int i = 0;
7      while (i < 10) {
8          i++;
9      }
10
11     return 0;
12 }
13
```

debug (Global Sco

```
1  #include <iostream>
2  using namespace std;
3
4  int main() {
5
6      int i = 0;
7      while (i < 10) {
8          i++;  ≤1ms elapsed
9      }
10
11     return 0;
12 }
13
```

debug (Global S

```
1  #include <iostream>
2  using namespace std;
3
4  int main() {
5
6      int i = 0;
7      while (i < 10) {
8          i++;
9      }  ≤1ms elapsed
10
11     return 0;
12 }
13
```

Breakpoints

- Pause your program execution
- You can add many breakpoints in a program(at least one)

- **Usages**
  - Check variable values
  - View call stack
  - Trace execution

Example…

- Rabbit population problem (fibonacci number)
  - A newly born pair of rabbits (1 male + 1 female) are put in a field
  - It takes 1 month for any newly born rabbit to grow up
  - Every grown up rabbit pair produce another pair in every month
  - Assume that no rabbit will die
  - How many pairs of rabbit are there at the end of the n-th month?

- $a_n = a_{n-1} + a_{n-2}$
  - $a_1=a_2=1$

- 1, 1, 2, 3, 5, 8, 13, 21, ….

Code for this problem

test.cpp*

debug (Global Scope)

```cpp
#include <iostream>

using namespace std;

int main() {

    int n;

    //Ask user for input
    cout << "Enter n: ";
    cin >> n;

    int newly_born = 1; //number of rabbit pairs born in this month
    int grown_up = 0; //number of rabbit pairs that have already grown up

    //simulate the whole rabbit population procedure
    for (int i = 1; i < n; i++)
    {
        int born = grown_up; //"born" pairs of rabbit are born in this month
        grown_up = grown_up + newly_born; //"newly_born" pairs of rabbit grown up in this month
        newly_born = born;
    }

    cout << "There is(are)" << newly_born + grown_up
        << "pair(s) of rabbit at the end of the"
        << n << "-th month." << endl;

    return 0;
}
```

This is a good example: add proper comments to enhance readability

Insert a breakpoint

test.cpp*

debug | (Global Scope) | main()

```
1   #include <iostream>
2
3   using namespace std;
4
5   int main() {
6
7       int n;
8
9       //Ask user for input
10      cout << "Enter n: ";
11      cin >> n;
12
13      int newly_born = 1; //number of rabbit pairs born in this month
14      int grown_up = 0; //number of rabbit pairs that have already grown up
15
16      //simulate the whole rabbit population procedure
17      for (int i = 1; i < n; i++)
18      {
19          int born = grown_up; //"born" pairs of rabbit are born in this month
20          grown_up = grown_up + newly_born; //"newly_born" pairs of rabbit grown up in this month
21          newly_born = born;
22      }
23
24      cout << "There is(are)" << newly_born + grown_up
25          << "pair(s) of rabbit at the end of the"
26          << n << "-th month." << endl;
27
28      return 0;
29  }
30
```

Press F5 to debug

D:\CSCI 1540\Tutorial 05 Debugging\debug\Debug\debug.exe

```
Enter n: 10
```

test.cpp

debug | (Global Scope)

```cpp
#include <iostream>

using namespace std;

int main() {

    int n;

    //Ask user for input
    cout << "Enter n: ";
    cin >> n;

    int newly_born = 1; //number of rabbit pairs born in this month
    int grown_up = 0; //number of rabbit pairs that have already grown up

    //simulate the whole rabbit population procedure
    for (int i = 1; i < n; i++)
    {
        int born = grown_up; //"born" pairs of rabbit are born in this month
        grown_up = grown_up + newly_born; //"newly_born" pairs of rabbit grown up in this month
        newly_born = born;
    }

    cout << "There is(are)" << newly_born + grown_up
        << "pair(s) of rabbit at the end of the"
        << n << "-th month." << endl;

    return 0;
}
```

100 % No issues found

Pause at this line

Different debug windows

Autos

Search (Ctrl+E) Search Depth: 3

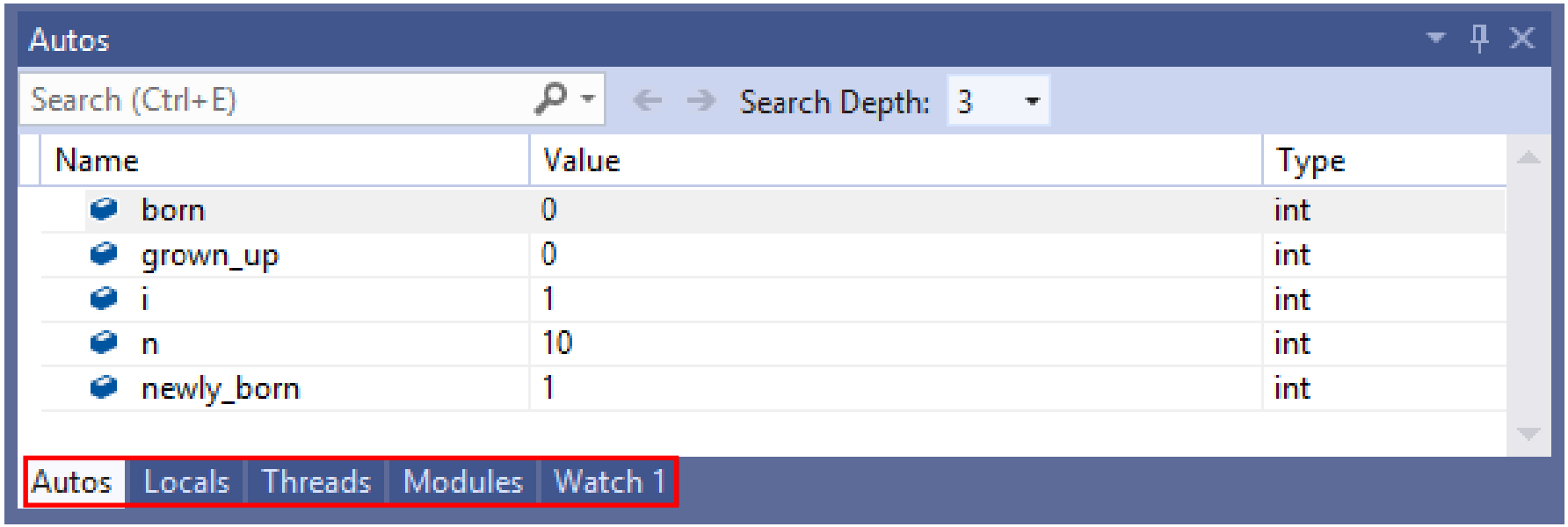
Name	Value	Type
born	0	int
grown_up	0	int
i	1	int
n	10	int
newly_born	1	int

Call Stack

Name
debug.exe!main() Line 20
[External Code]
kernel32.dll![Frames below may be incorrect and/or missing, no symbols loaded for kernel32.dll]

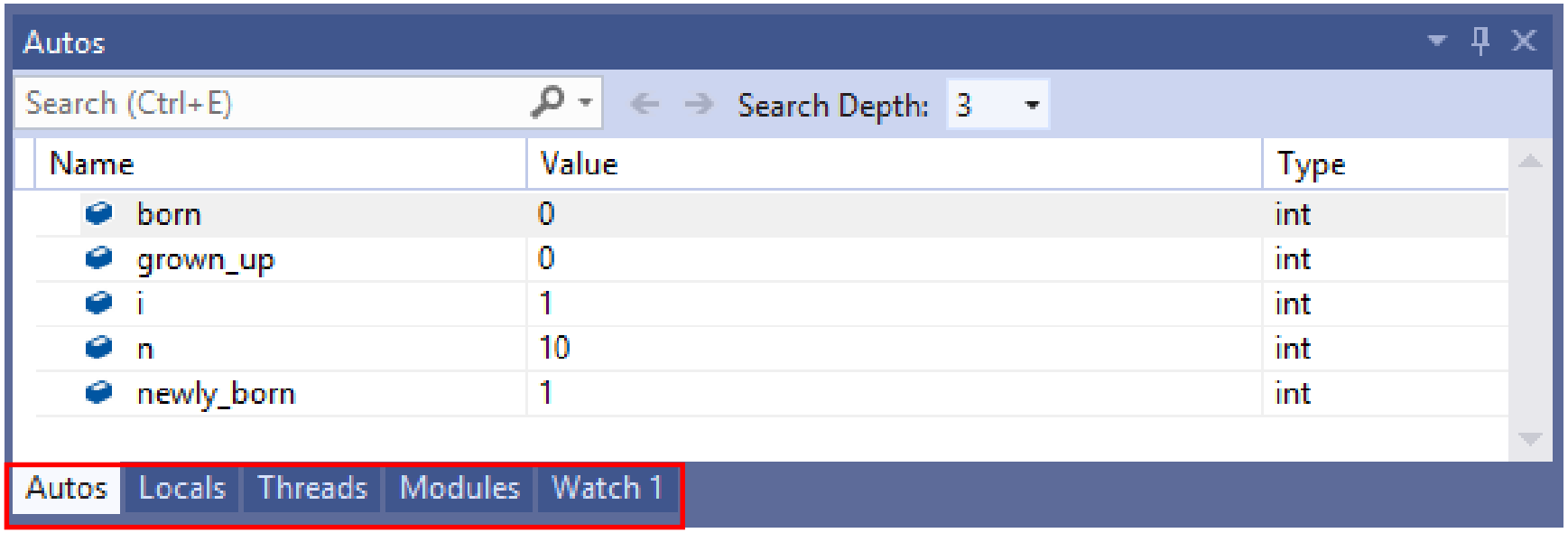
Debug Window

- Debug Window: Autos, Locals, Threads, Modules, Watch1



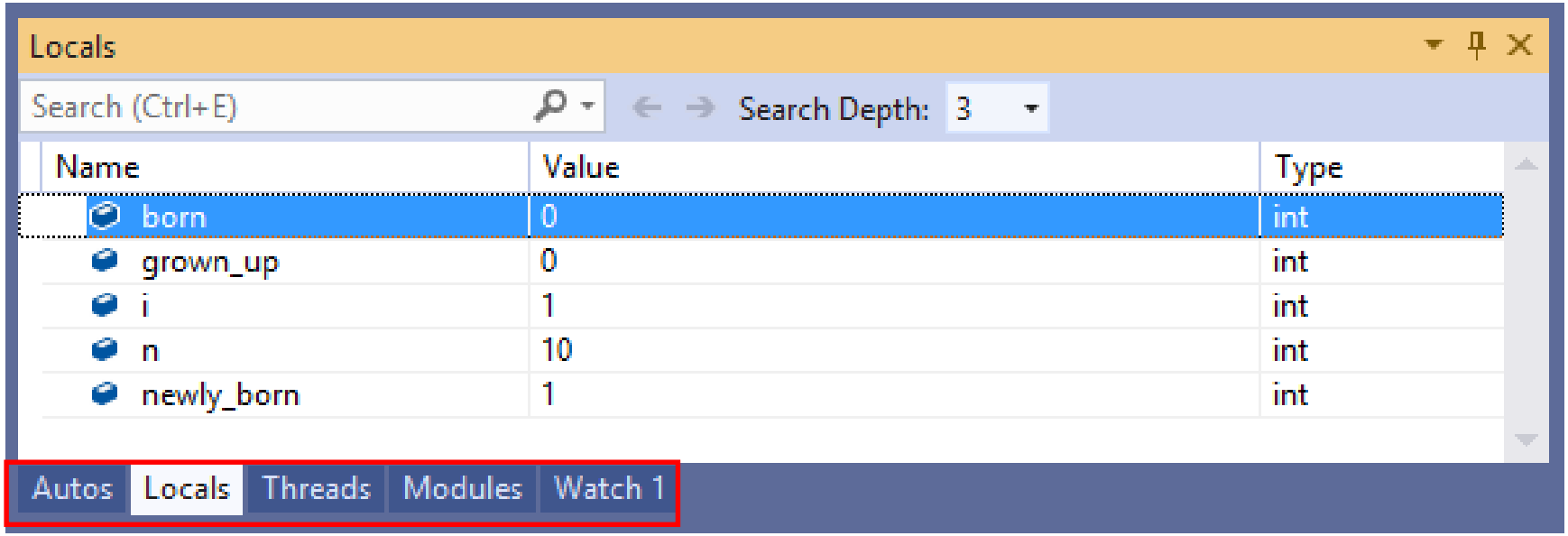
Debug Window

- Autos: see variables used near your instruction pointer.



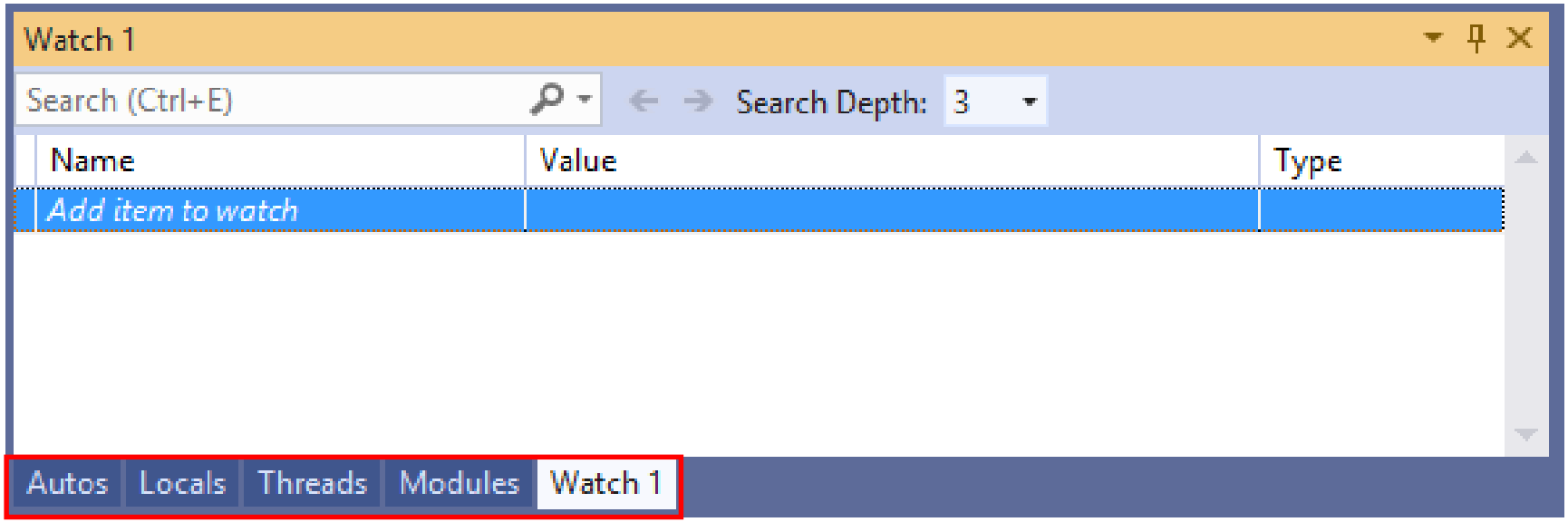
Debug Window

- Locals: see variables that exist in the local scope of your current stack frame



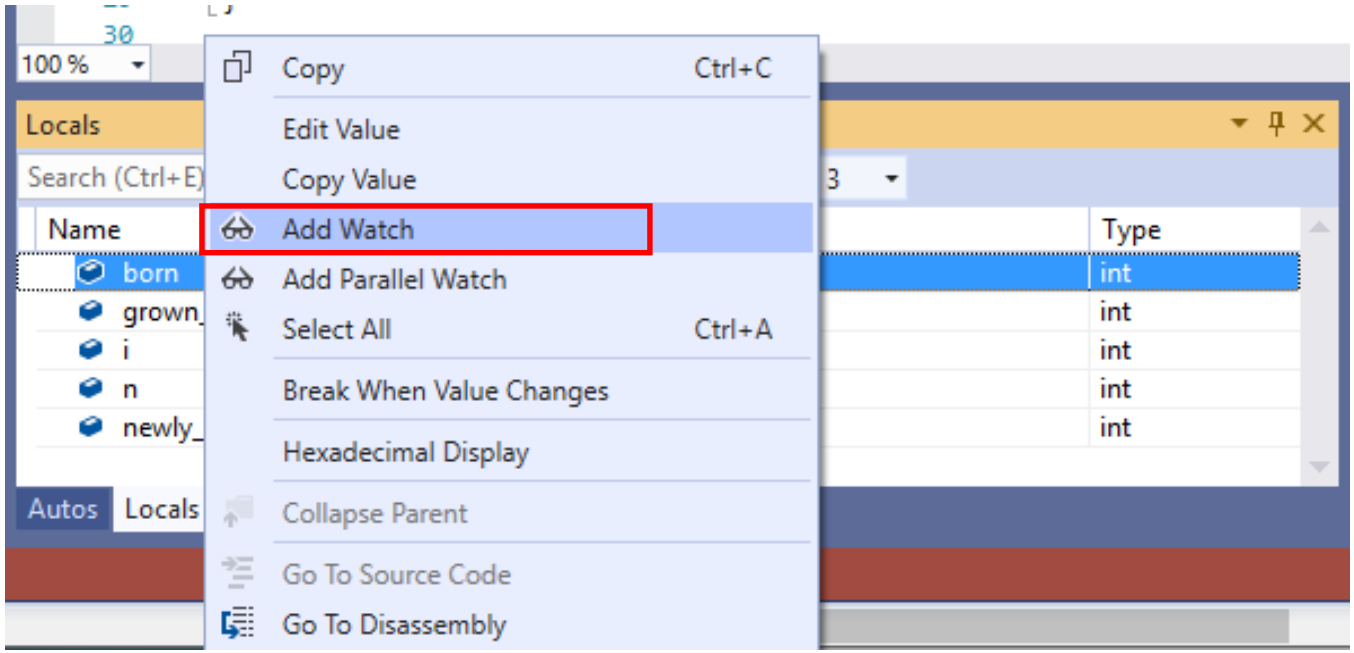
Debug Window

- Watches: Monitor the values of specified variable or expression
- E.g. type “born” and “grown_up”



Debug Window

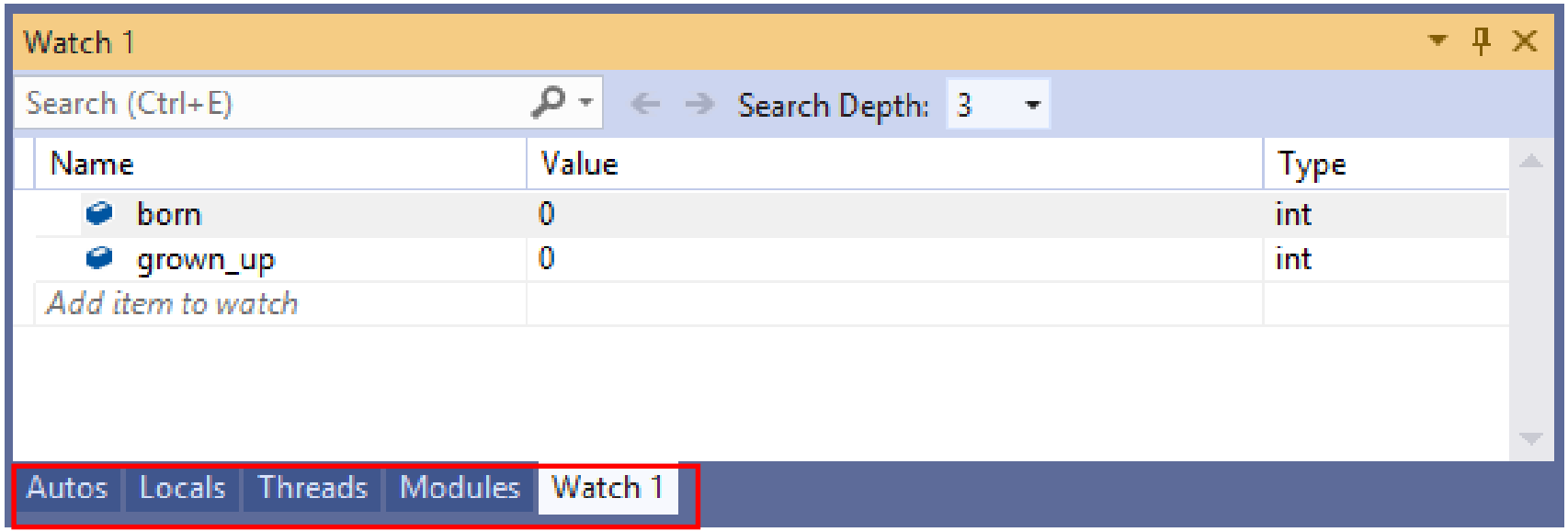
- Watches: Right-click the variables at Locals/Autos to add it to Watch



- Monitor any variable by adding it into "Watch"

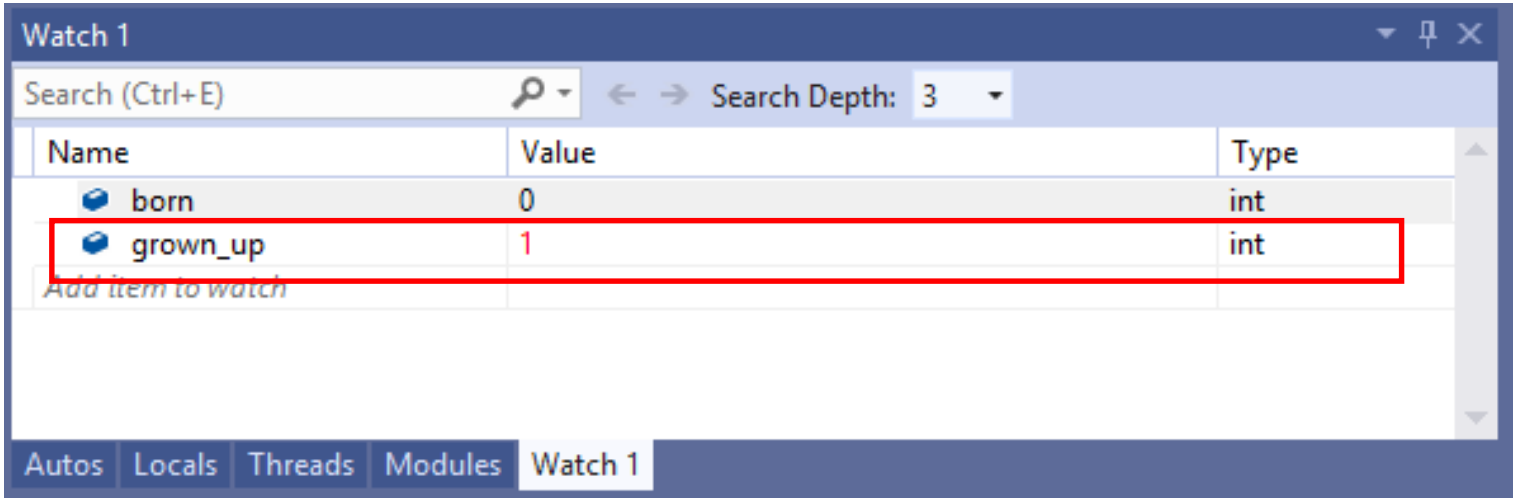
Debug Window

- Watches: Monitor the values of specified variable or expression
- E.g. type “born” and “grown_up”



Debug Window

- Watches: Recent changes in RED



Step Over (F10)

- **Run the program line by line**
- **Skip functions being called**
  - The underlying functions are still executed
  - But not displayed

Run line by line, from the breakpoint

test.cpp

debug (Global Scope)

```
1   #include <iostream>
2
3   using namespace std;
4
5   int main() {
6
7       int n;
8
9       //Ask user for input
10      cout << "Enter n: ";
11      cin >> n;
12
13      int newly_born = 1; //number of rabbit pairs born in this month
14      int grown_up = 0; //number of rabbit pairs that have already grown up
15
16      //simulate the whole rabbit population procedure
17      for (int i = 1; i < n; i++)
18      {
19          int born = grown_up; //"born" pairs of rabbit are born in this month
20          grown_up = grown_up + newly_born; //"newly_born" pairs of rabbit grown up in this month
21          newly_born = born;
22      }
23
24      cout << "There is(are)" << newly_born + grown_up
25          << "pair(s) of rabbit at the end of the"
26          << n << "-th month." << endl;
27
28      return 0;
29  }
```

Step Into (F11)

- **Look into all functions codes being called**

```
5   int countRabbits(int n) {
6       int newly_born = 1; // number of rabbit pairs bo
7       int grown_up = 0;   // number of rabbit pairs th
8
9       // simulate the whole rabbit population procedur
10      for(int i=1;i<n;i++) {
11          int born = grown_up; // "born" pairs of rabb
12          grown_up = grown_up + newly_born; // "newly_
13          newly_born = born;
14      }
15
16      return newly_born + grown_up;
17  }
18
19  int main() {
20      int n, total;
21
22      // Ask user for input
23      cout << "Enter n: ";
24      cin >> n;
25
26      total = countRabbits(n);
27
28      cout << "There is(are) " << total
29           << " pair(s) of rabbit at the end of the "
30           << n << "-th month." << endl;
```

```
5   int countRabbits(int n) {
6       int newly_born = 1; // number of rabbit pairs bo
7       int grown_up = 0;   // number of rabbit pairs th
8
9       // simulate the whole rabbit population procedur
10      for(int i=1;i<n;i++) {
11          int born = grown_up; // "born" pairs of rabb
12          grown_up = grown_up + newly_born; // "newly_
13          newly_born = born;
14      }
15
16      return newly_born + grown_up;
17  }
18
19  int main() {
20      int n, total;
21
22      // Ask user for input
23      cout << "Enter n: ";
24      cin >> n;
25
26      total = countRabbits(n);
27
28      cout << "There is(are) " << total
29           << " pair(s) of rabbit at the end of the "
30           << n << "-th month." << endl;
```

Step Out (Shift+F11)

- **Get out of the current function**

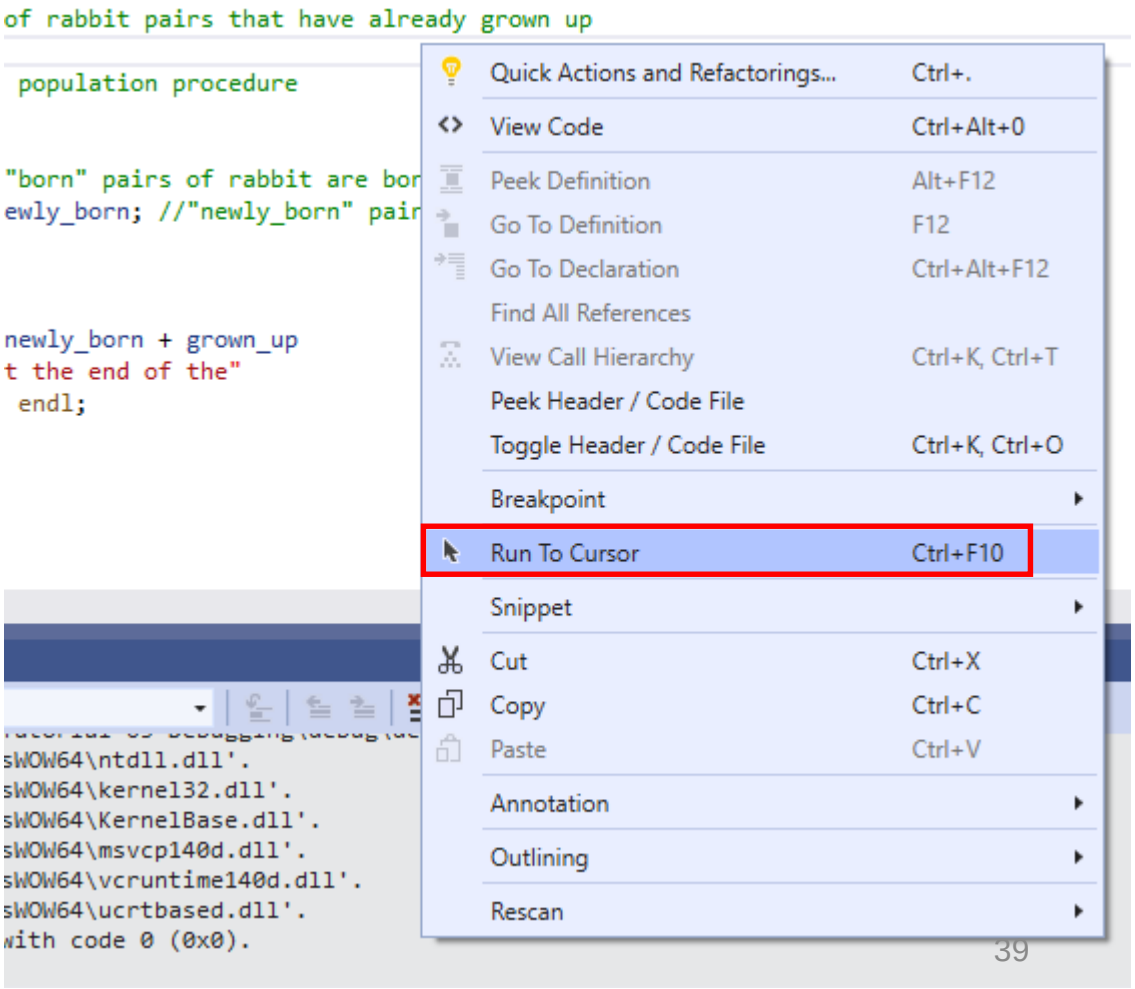
```
5   int countRabbits(int n) {
6       int newly_born = 1; // number of rabbit pairs bo
7       int grown_up = 0;   // number of rabbit pairs th
8
9       // simulate the whole rabbit population procedur
10      for(int i=1;i<n;i++) {
11          int born = grown_up; // "born" pairs of rabb
12          grown_up = grown_up + newly_born; // "newly_
13          newly_born = born;
14      }
15
16      return newly_born + grown_up;
17  }
18
19  int main() {
20      int n, total;
21
22      // Ask user for input
23      cout << "Enter n: ";
24      cin >> n;
25
26      total = countRabbits(n);
27
28      cout << "There is(are) " << total
29           << " pair(s) of rabbit at the end of the "
30           << n << "-th month." << endl;
```

```
5   int countRabbits(int n) {
6       int newly_born = 1; // number of rabbit pairs bo
7       int grown_up = 0;   // number of rabbit pairs th
8
9       // simulate the whole rabbit population procedur
10      for(int i=1;i<n;i++) {
11          int born = grown_up; // "born" pairs of rabb
12          grown_up = grown_up + newly_born; // "newly_
13          newly_born = born;
14      }
15
16      return newly_born + grown_up;
17  }
18
19  int main() {
20      int n, total;
21
22      // Ask user for input
23      cout << "Enter n: ";
24      cin >> n;
25
26      total = countRabbits(n);
27
28      cout << "There is(are) " << total
29           << " pair(s) of rabbit at the end of the "
30           << n << "-th month." << endl;
```

Run to cursor

- **Right-click on your code, run to cursor**

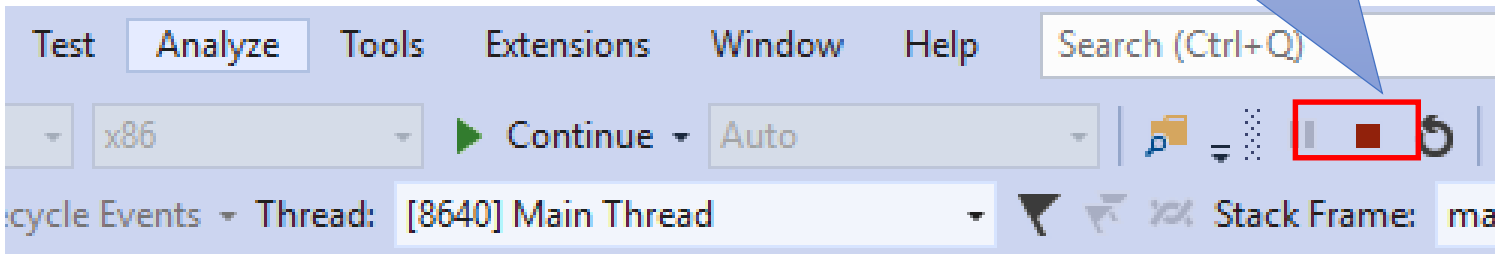
- Help you to skip part of the code quickly
- An alternative of running line by line



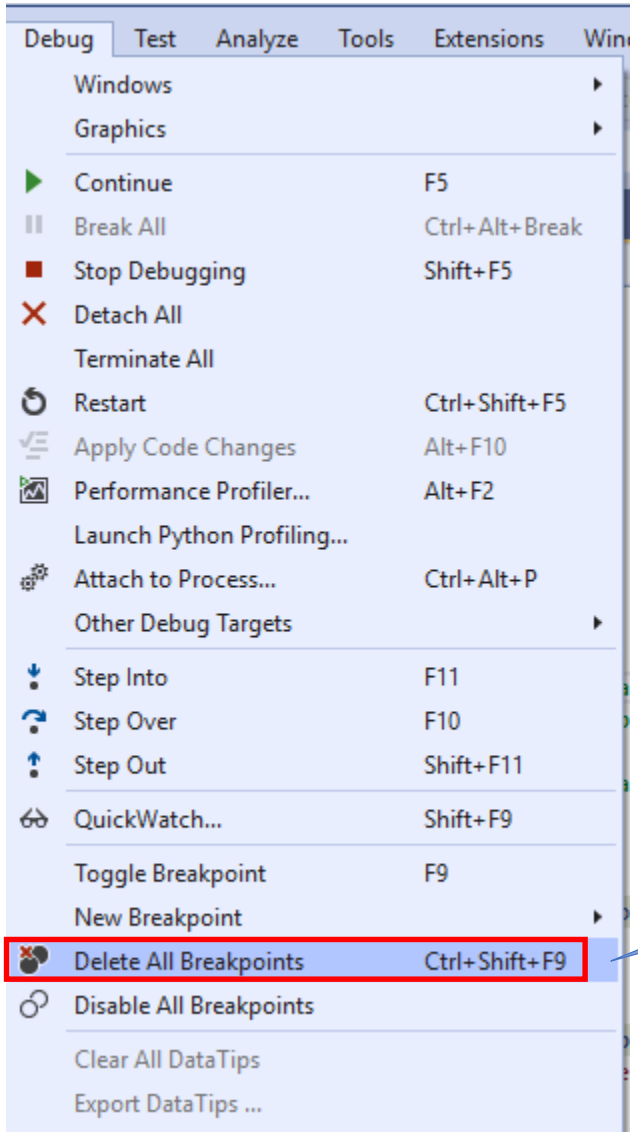
Stopping the Debugger

- **End the program and debug session**

Click the stop button or press Shift + F5



Delete the breakpoints



Tips: you can also click on the breakpoint to remove it

Debugging without “Debugger”

- If program has bug, the variable values will be different from what we expect
  - Use codes to print out the variables in the running time
  - No breakpoints
  - No need to control debugging procedure

Debugging without “Debugger”

- Example: simple program to calculate product from 1 to 5 with loop
- Code with bug:

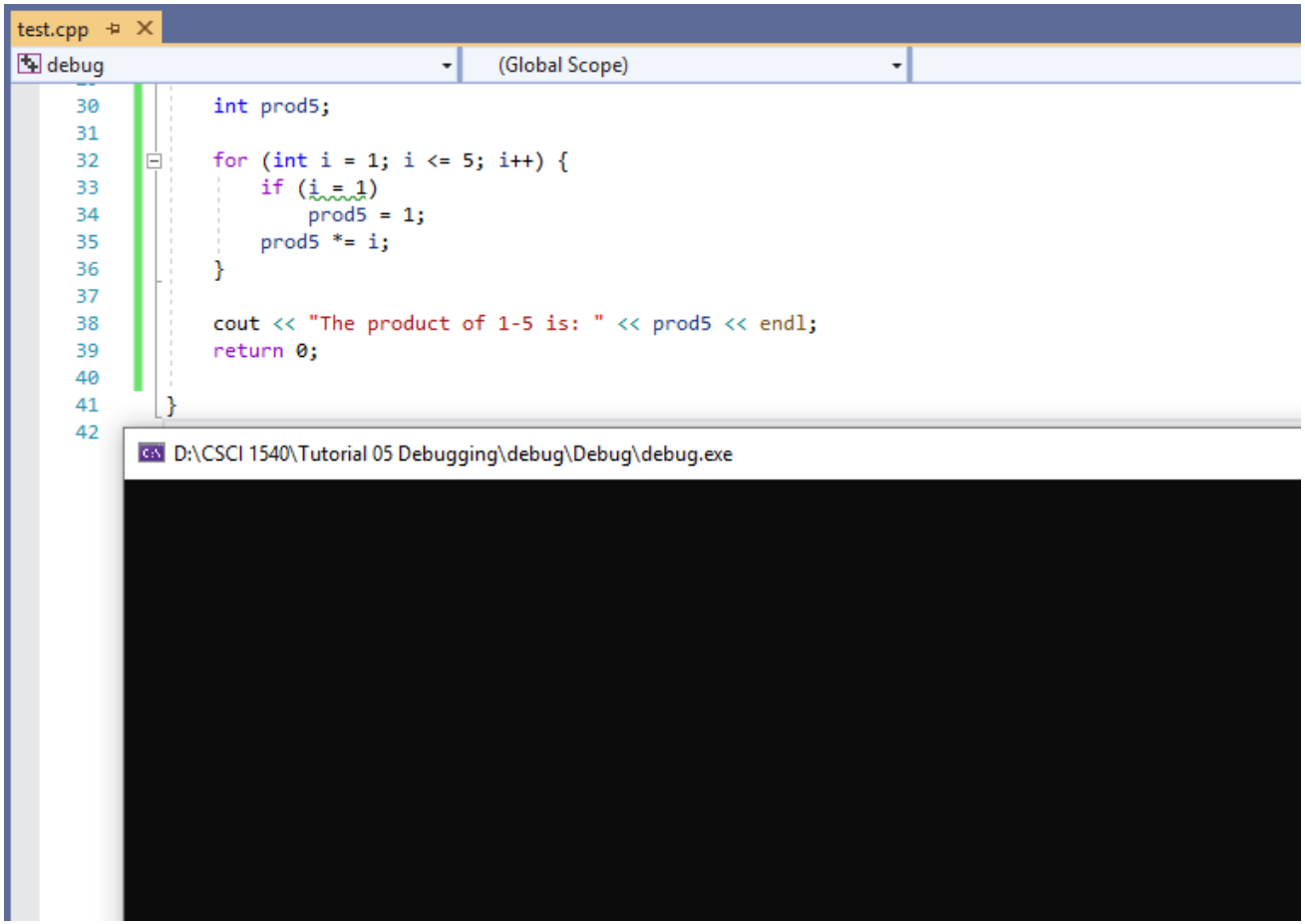
```
int prod5;

for (int i = 1; i <= 5; i++) {
        if (i = 1)
                prod5 = 1;
        prod5 *= i;
}

cout << "The product of 1-5 is: " << prod5 << endl;
return 0;
```

Debugging without “Debugger”

- Ctrl+F5
- No reaction in the CLI.



Debugging without “Debugger”

- cout ‘prod5’ and ‘i’ in the loop

```
int prod5;

for (int i = 1; i <=
        if (i = 1)
                prod5 = 1;
        prod5 *= i;
        cout << "i:" << i << "  product:" << prod5 << endl;
}

cout << "The product of 1-5 is: " << prod5 << endl;
return 0;
```

‘==‘ (equal to) instead of ‘=‘ (assign to)

Hope to see:

```
i:1  product:1
i:2  product:2
i:3  product:6
i:4  product:24
i:5  product:120
```

The fact:

```
i:1  product:1
i:1  product:1
i:1  product:1
i:1  product:1
i:1  product:1
i:1  product:1
i:1  product:1
i:1  product:1
i:
```

Endless loop
Wrong value of ‘i’

Debugging without “Debugger”

- ‘==‘ (equal to) instead of ‘=‘ (assign to)

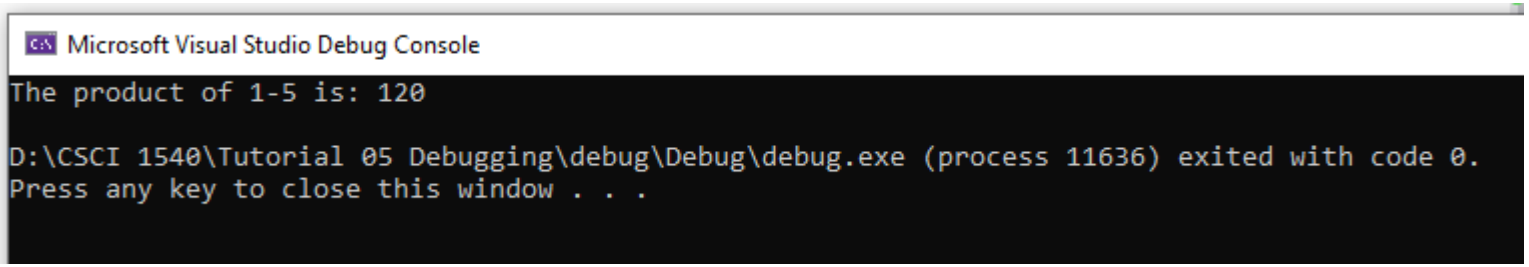
```
int prod5;

for (int i = 1; i <= 5; i++) {
        if (i == 1)
                prod5 = 1;
        prod5 *= i;
}

cout << "The product of 1-5 is: " << prod5 << endl;
return 0;
```

Don’t forget to delete the “Debugging” codes

- Fix the bug, and the output:



Q & A