# Survey of Scientific Computing

(SciComp 301)

Dave Biersach
Brookhaven National Laboratory
dbiersach@bnl.gov

## Session 07

Creating a New Project

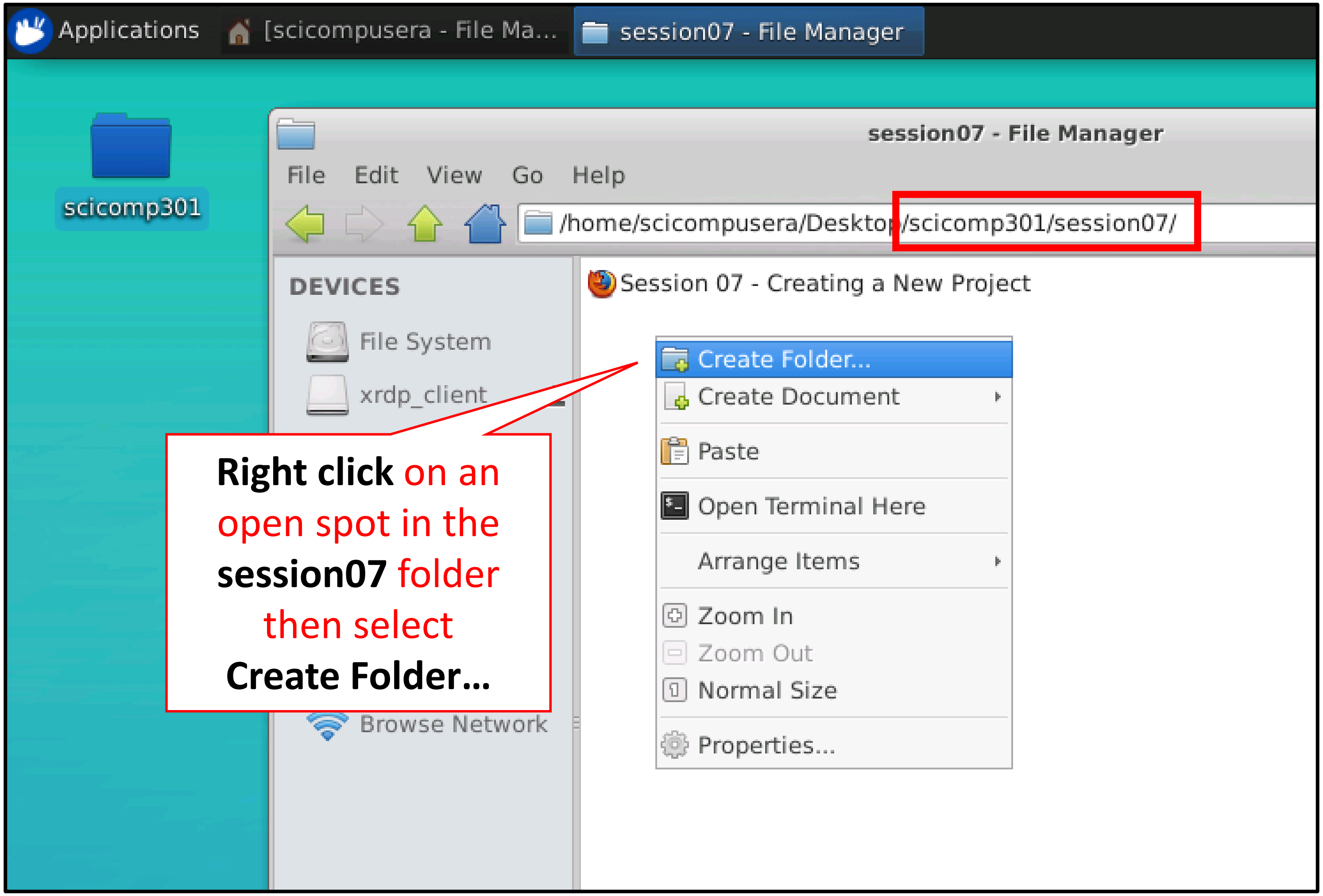
4th Annual SCI-ED Day hosted at BNL on November 4, 2014, provides a professional development opportunity for elementary and secondary science teachers.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace SimpleEvents
{
    public partial class Form1 : Form
    {
        Person person = new Person();

        public Form1()
        {
            InitializeComponent();
            person.FirstName = "Christian";
            person.LastName = "Nano";
        }

        private void button1_Click(object sender, EventArgs e)
        {
            person.HairColor = textBox1.Text;
        }
    }
}
```

# Session Goals

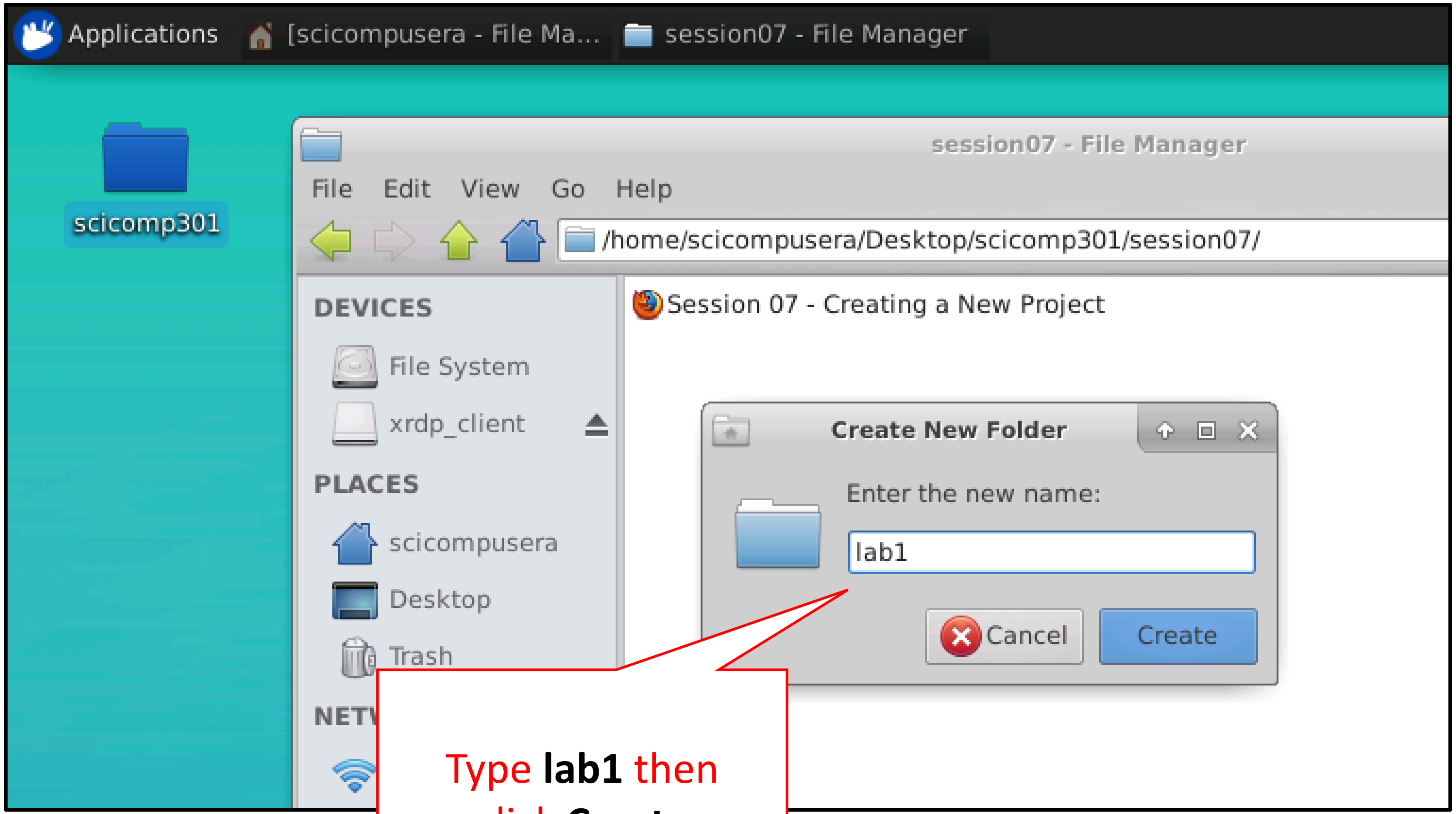
- Demonstrate step-by-step how to create a “starter” (blank) **C++/14** console application using the **Code::Blocks** IDE
- Explain the purpose of the **stdafx.h** file and how to add it to a Code::Blocks project
- Review the **bubble sort** algorithm and how it can order the elements of a given vector
- Introduce Euler’s **Totient** function which returns the number of positive integers less than a given integer that are relatively prime to that integer
- Write code to implement **Heron’s Formula** and to calculate basic **statistics** of a vector of random integers

# Create *lab1* subfolder in *session07* folder



**Right click** on an open spot in the **session07** folder then select **Create Folder…**

# Create *lab1* subfolder in *session07* folder



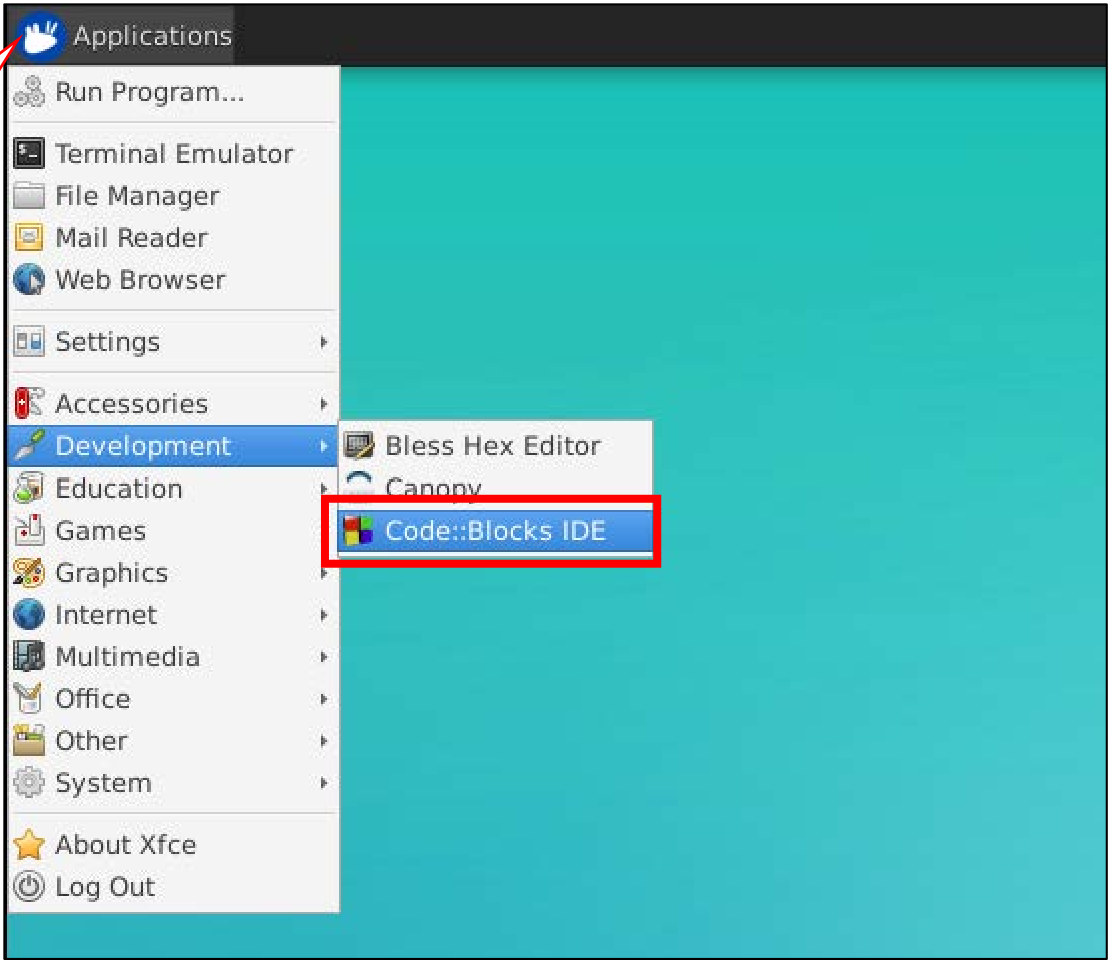
/home/scicompusera/Desktop/scicomp301/session07/

Enter the new name: lab1

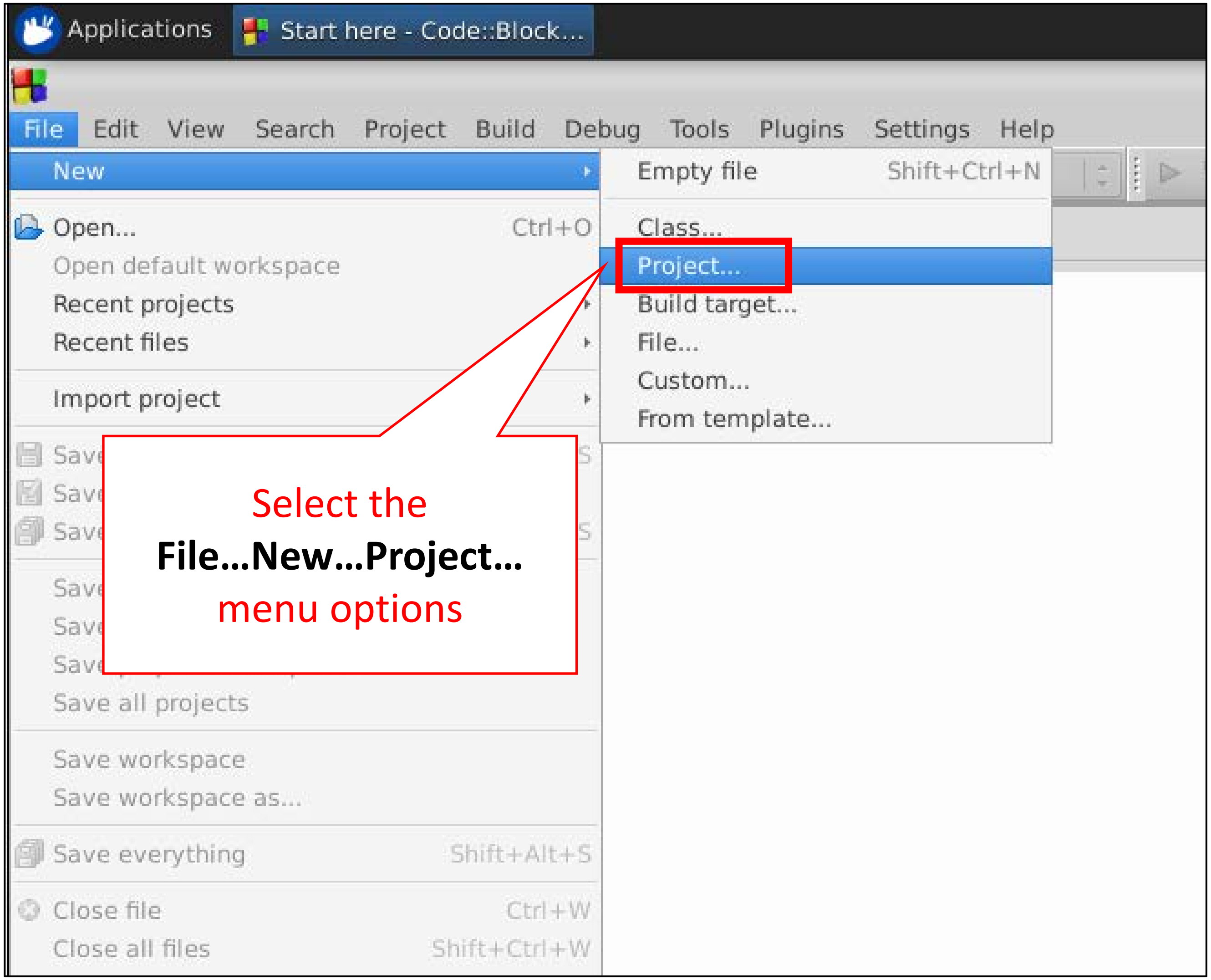
Type **lab1** then click **Create**

# Launch the Code::Blocks Integrated Development Environment (IDE)

From **Applications** select **Development** and then **Code::Blocks IDE**

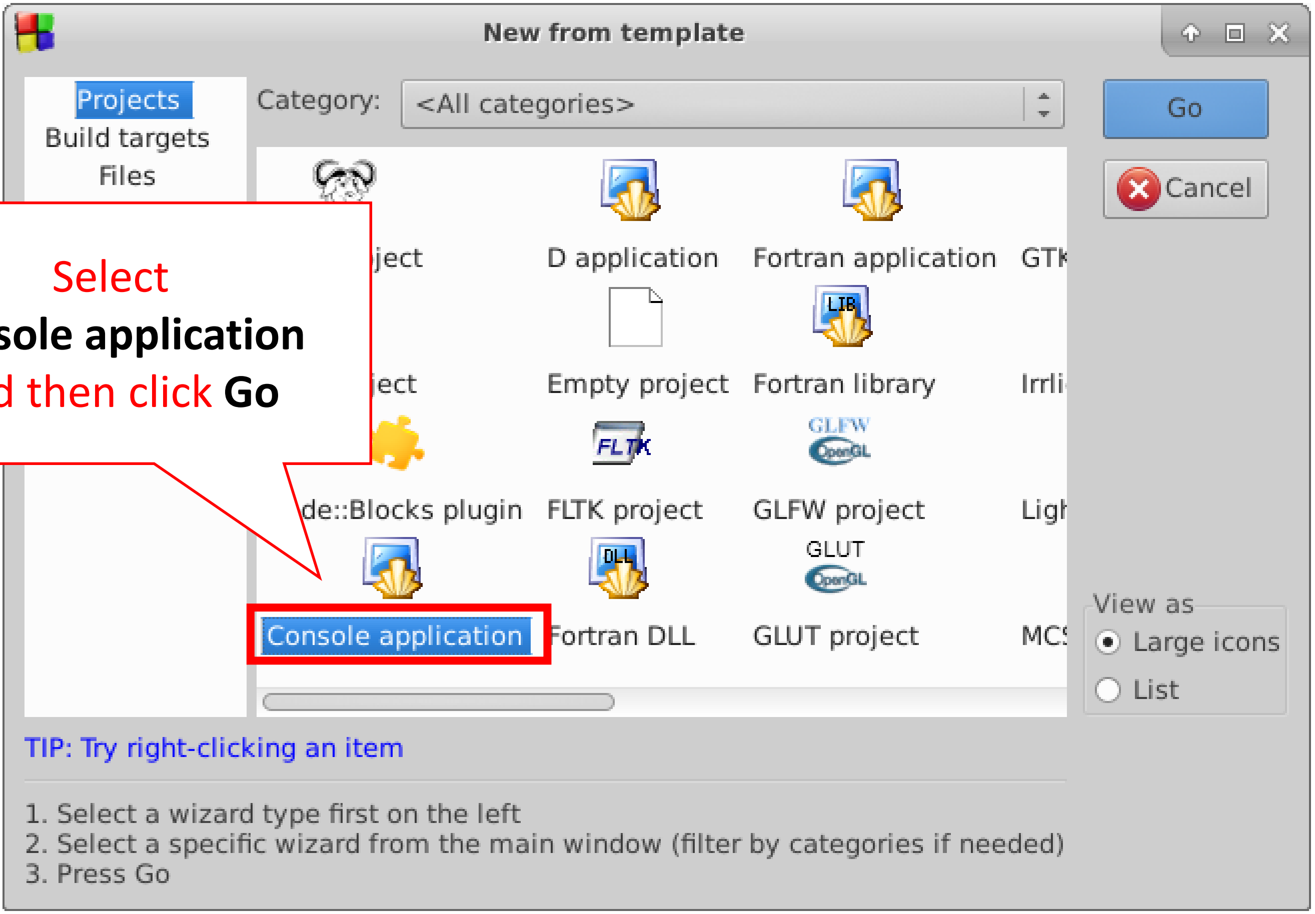


# Create a New Project in Code::Blocks



Select the **File…New…Project…** menu options

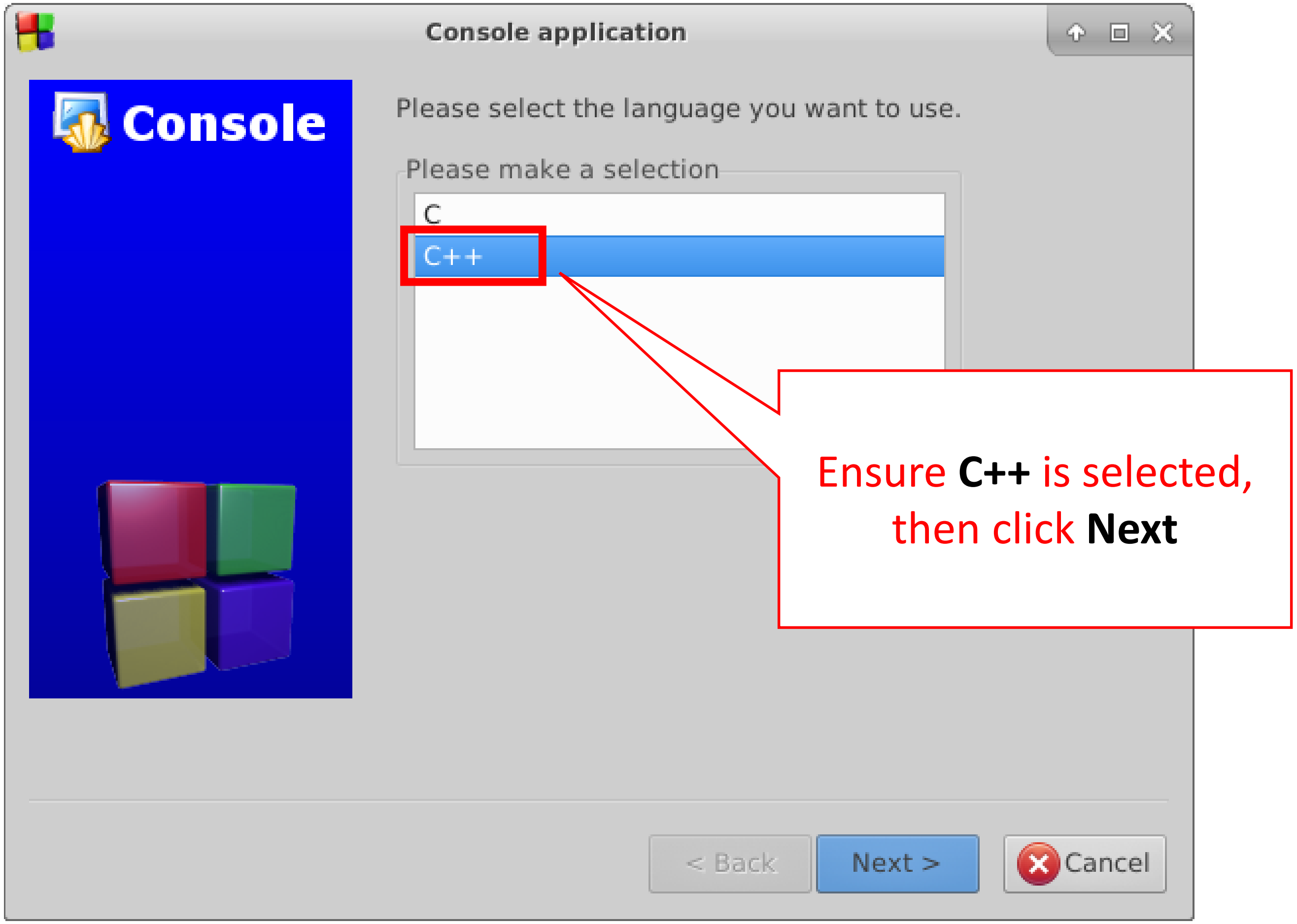
# Create a New Project in Code::Blocks



1. Select a wizard type first on the left
2. Select a specific wizard from the main window (filter by categories if needed)
3. Press Go

Select **Console application** and then click **Go**

# Create a New Project in Code::Blocks



Please select the language you want to use.

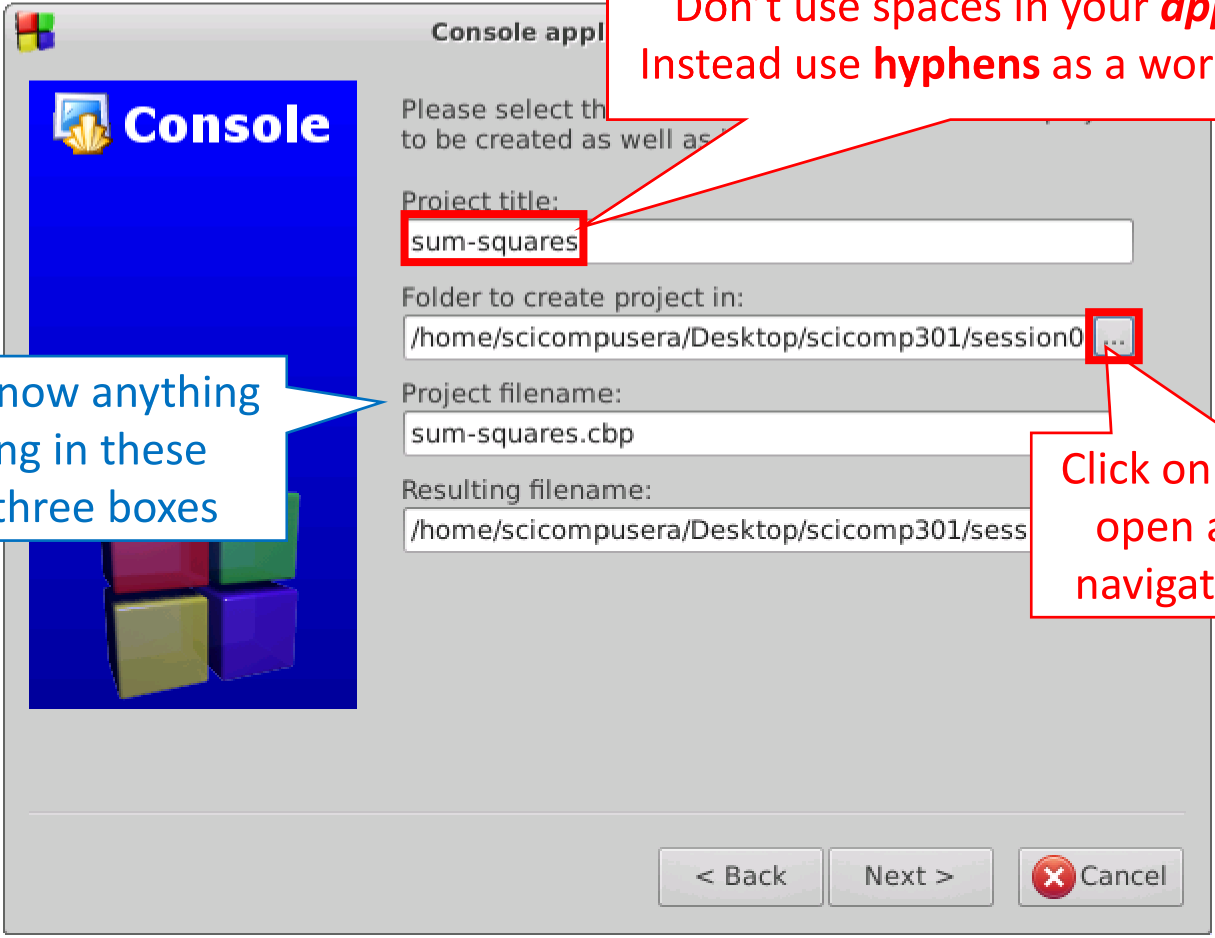
Please make a selection

C

C++

Ensure **C++** is selected, then click **Next**

# Set ***appname*** and project folder location



Please select th
to be created as well as

Project title:
sum-squares

Folder to create project in:
/home/scicompusera/Desktop/scicomp301/session0

Project filename:
sum-squares.cbp

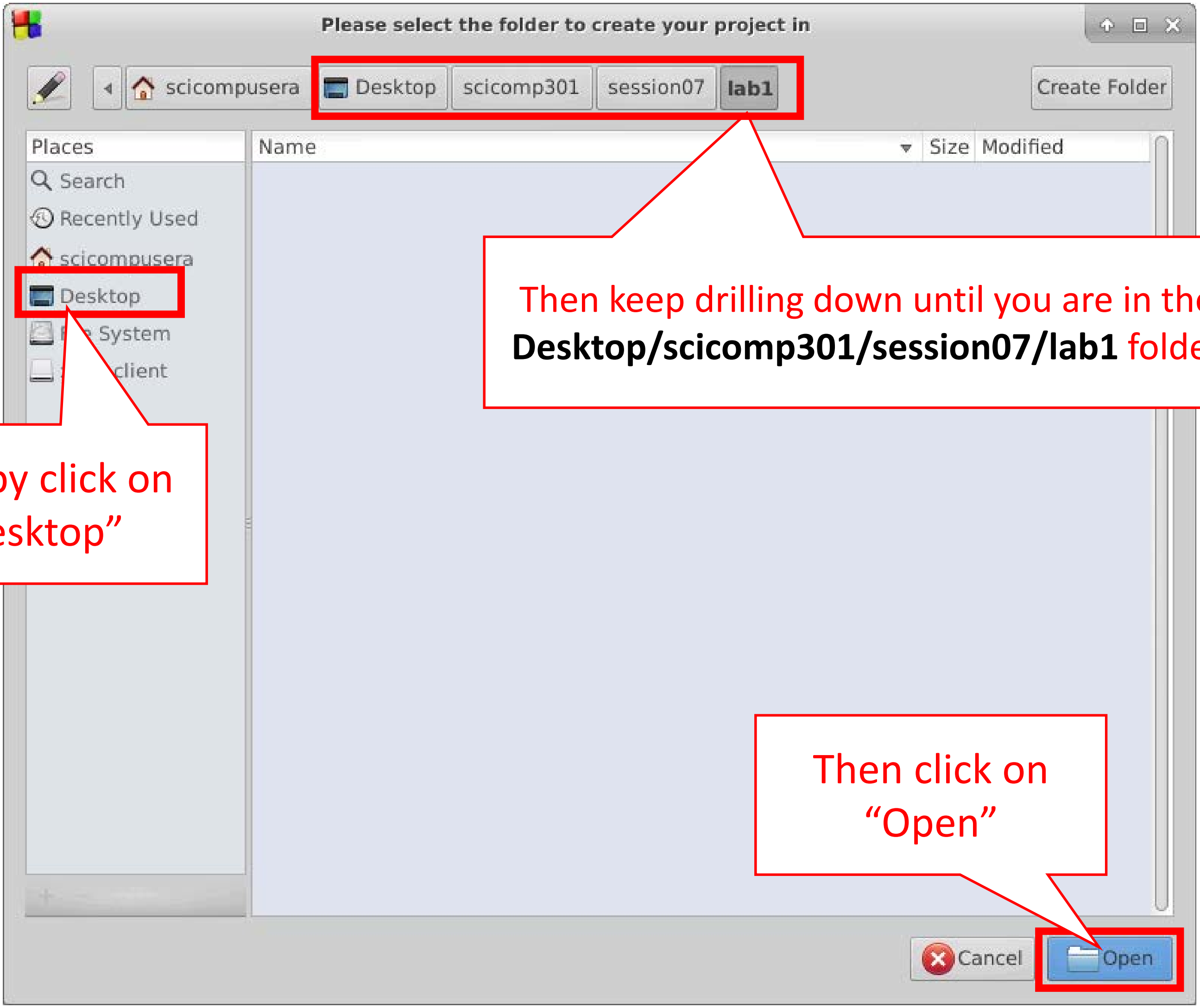
Resulting filename:
/home/scicompusera/Desktop/scicomp301/sess

Don’t use spaces in your ***appname***!
Instead use **hyphens** as a word breaker

Ignore for now anything appearing in these bottom three boxes

Click on the ellipsis to open a file system navigator dialog box

# Set ***appname*** and project folder location

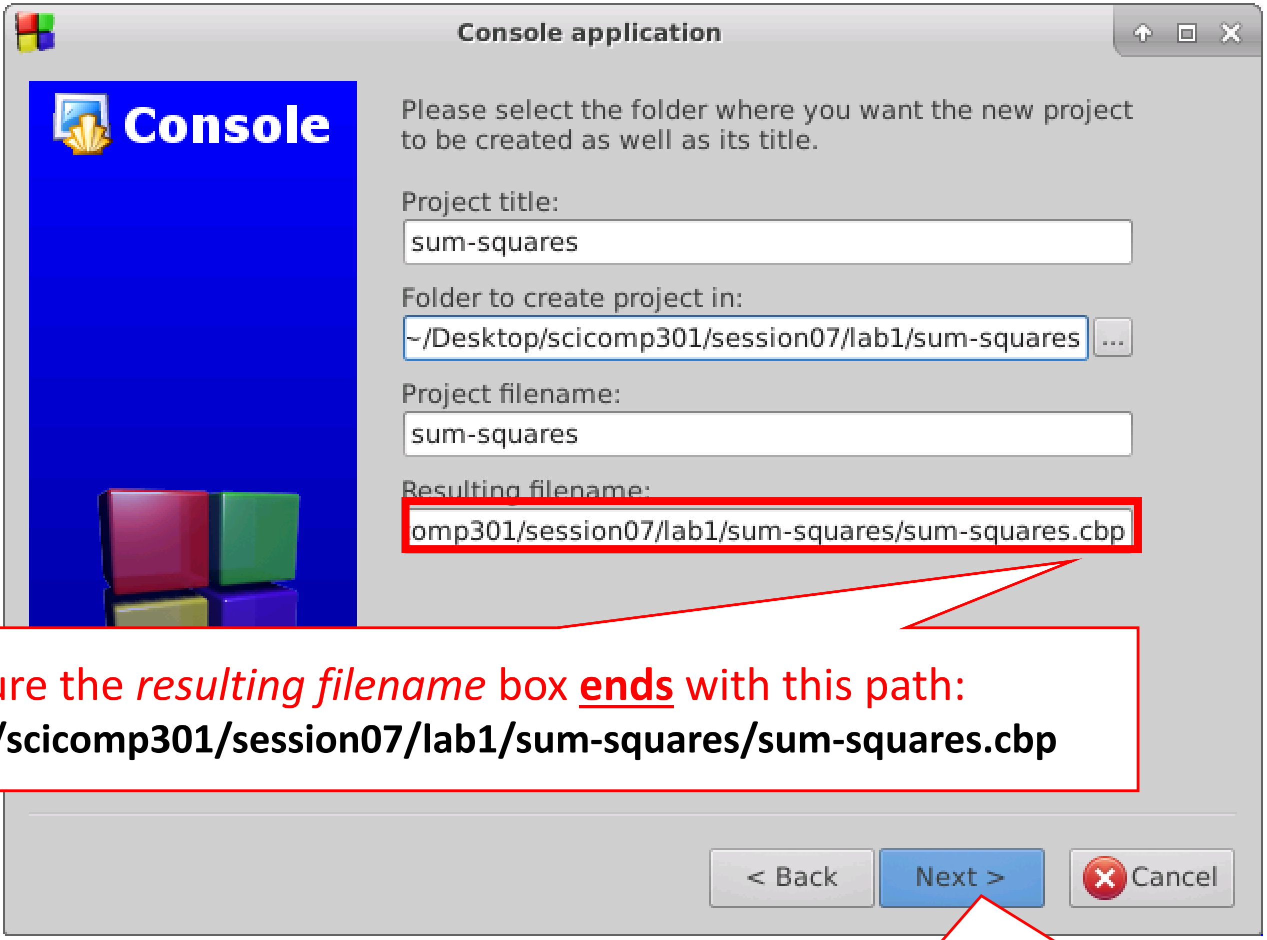


Start by click on “Desktop”

Then keep drilling down until you are in the **Desktop/scicomp301/session07/lab1** folder

Then click on “Open”

# Set ***appname*** and project folder location



Please select the folder where you want the new project to be created as well as its title.

Project title: sum-squares

Folder to create project in: ~/Desktop/scicomp301/session07/lab1/sum-squares

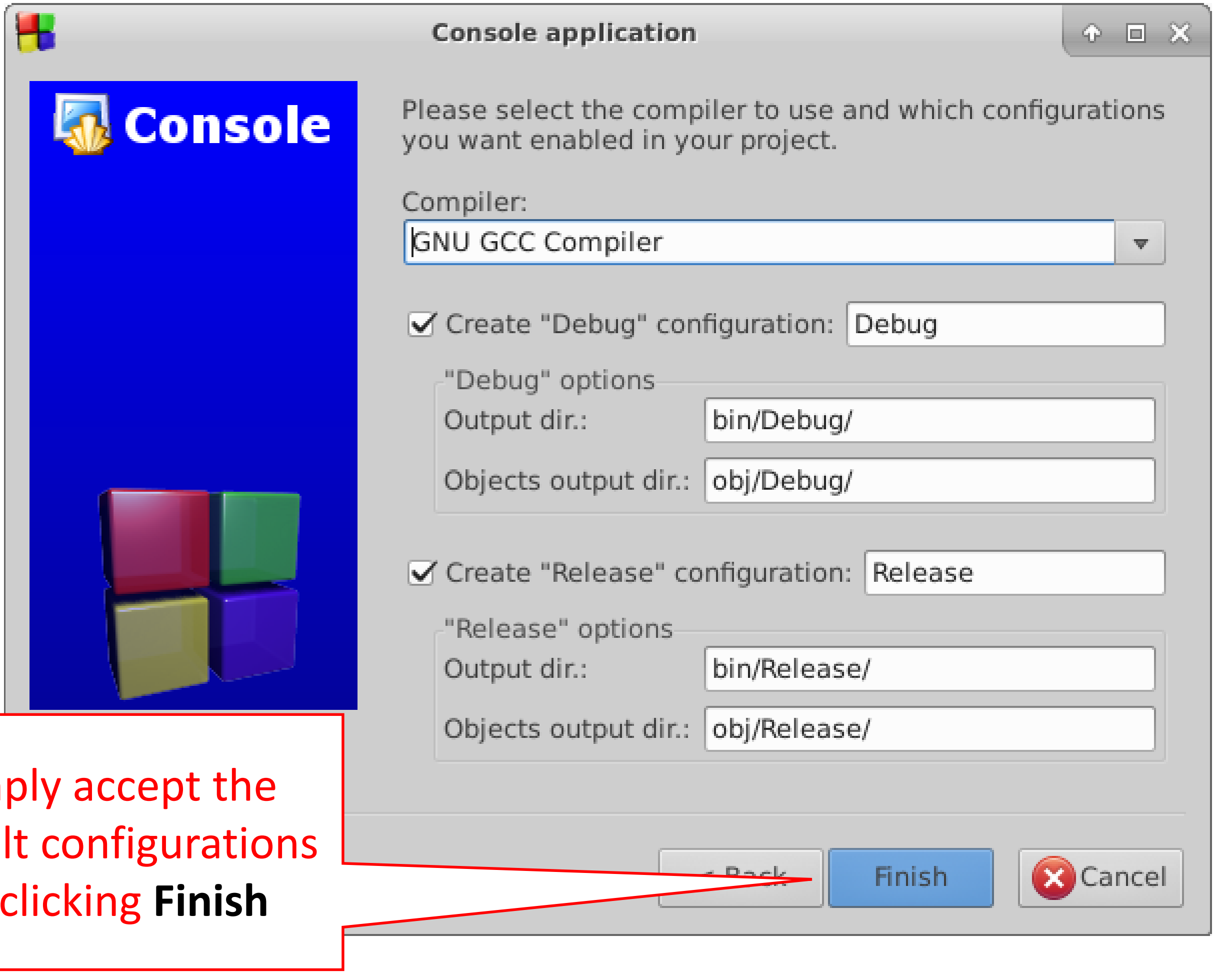
Project filename: sum-squares

Resulting filename: omp301/session07/lab1/sum-squares/sum-squares.cbp

Ensure the *resulting filename* box **ends** with this path:
**/Desktop/scicomp301/session07/lab1/sum-squares/sum-squares.cbp**

Then click on "Next"

# Accept Default Configurations



Console application

Console

Please select the compiler to use and which configurations you want enabled in your project.

Compiler: GNU GCC Compiler

☑ Create "Debug" configuration: Debug

"Debug" options

Output dir.: bin/Debug/

Objects output dir.: obj/Debug/

☑ Create "Release" configuration: Release

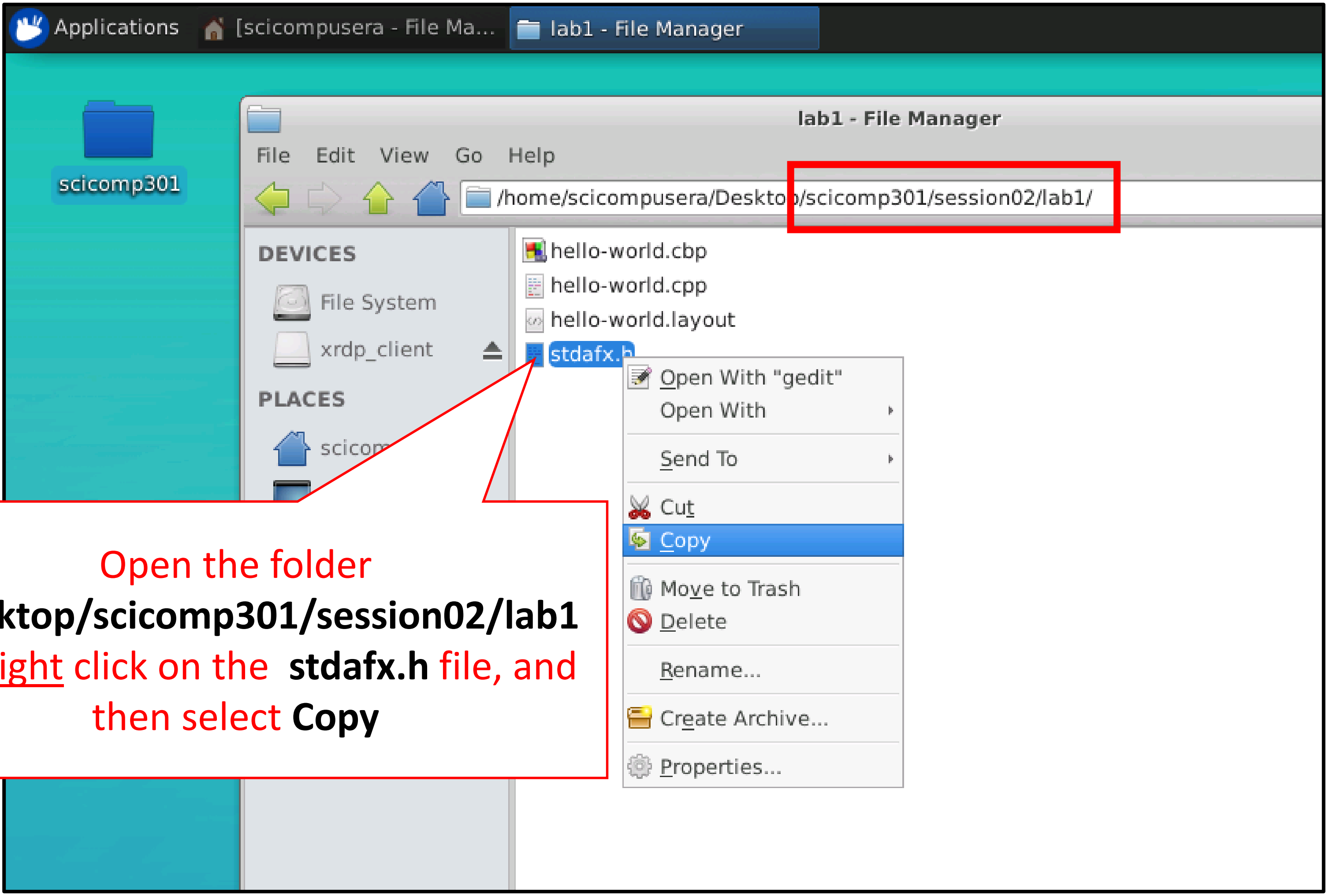
"Release" options

Output dir.: bin/Release/

Objects output dir.: obj/Release/

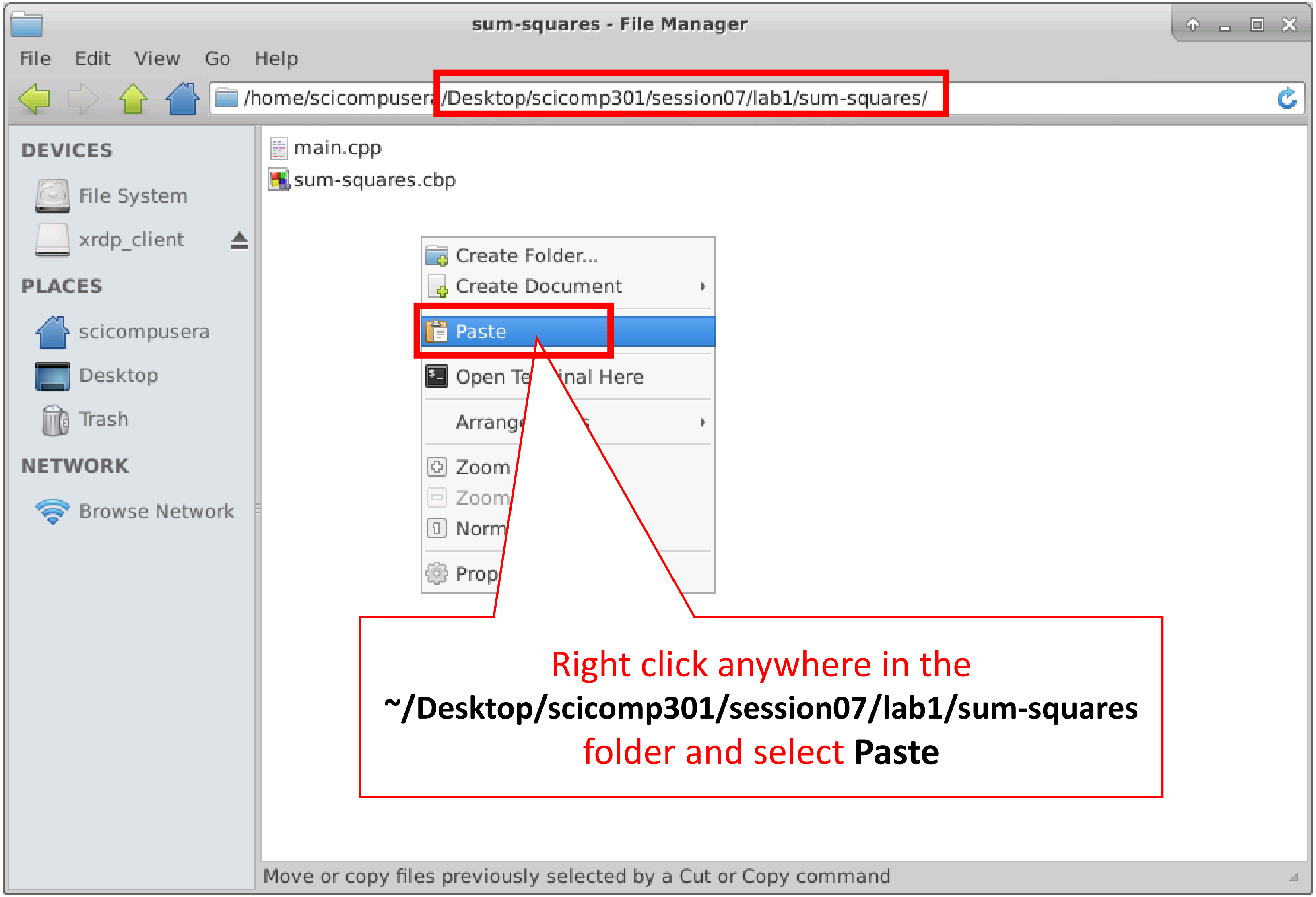
Simply accept the default configurations by clicking **Finish**

# Copy "stdafx.h" file to your ***appname*** folder



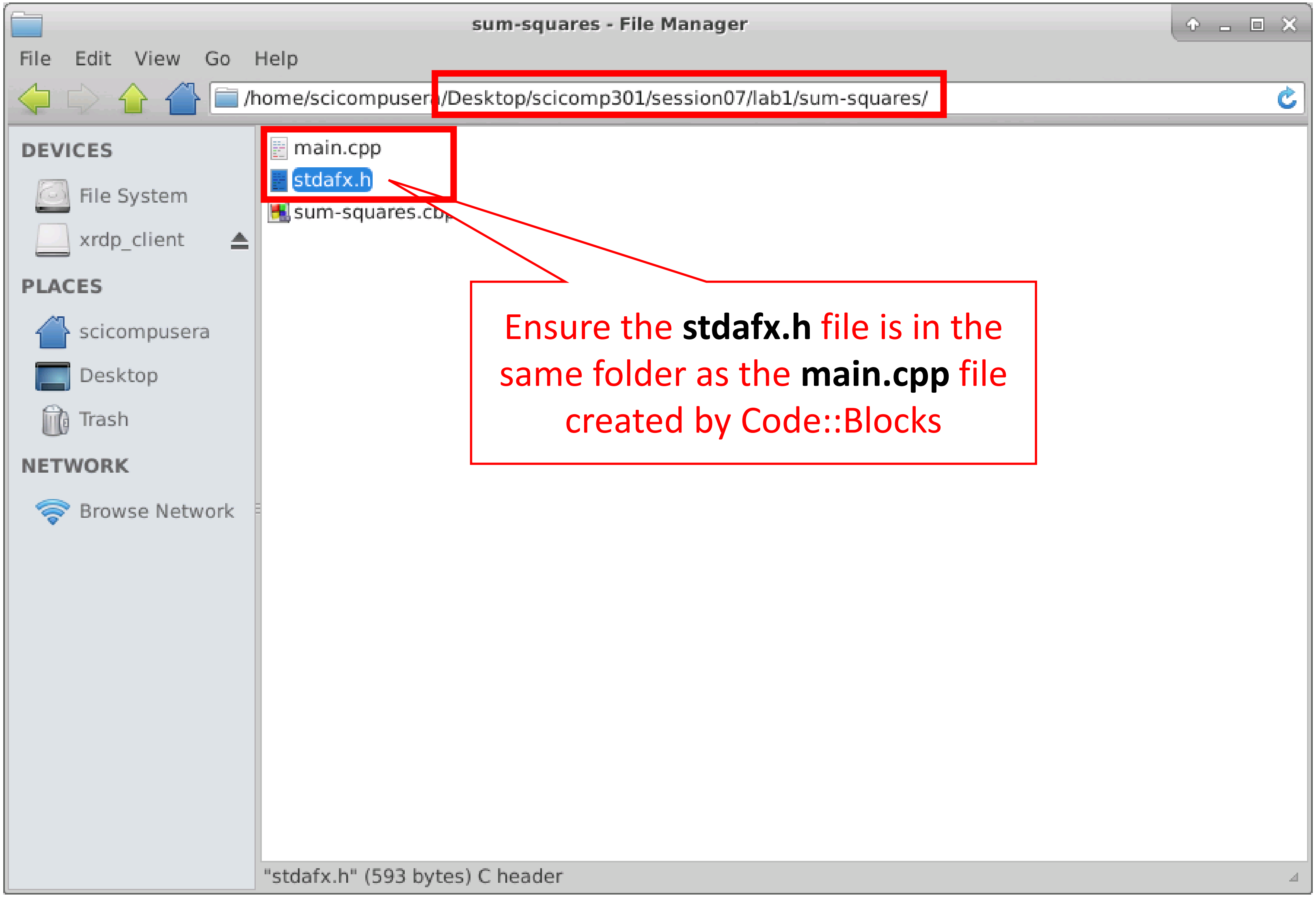
Open the folder
**~/Desktop/scicomp301/session02/lab1**
then right click on the **stdafx.h** file, and
then select **Copy**

# Copy “**stdafx.h**” file to your ***appname*** folder



Right click anywhere in the **~/Desktop/scicomp301/session07/lab1/sum-squares** folder and select **Paste**

# Copy “**stdafx.h**” file to your ***appname*** folder



sum-squares - File Manager

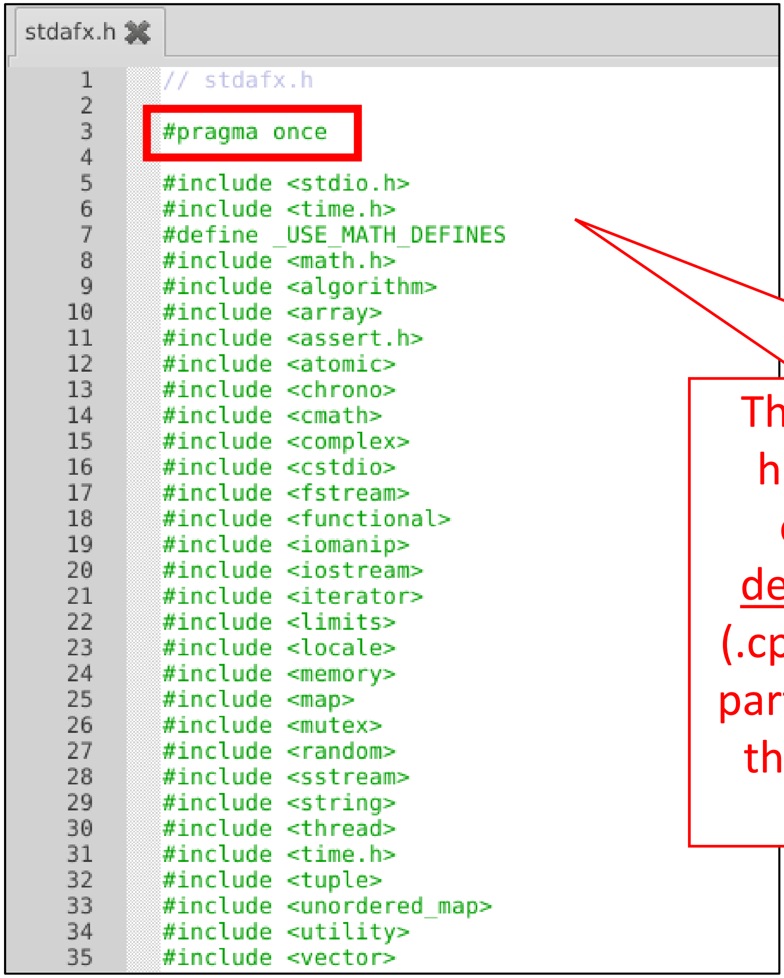
/home/scicompusera/Desktop/scicomp301/session07/lab1/sum-squares/

main.cpp
stdafx.h
sum-squares.cbp

"stdafx.h" (593 bytes) C header

Ensure the **stdafx.h** file is in the same folder as the **main.cpp** file created by Code::Blocks

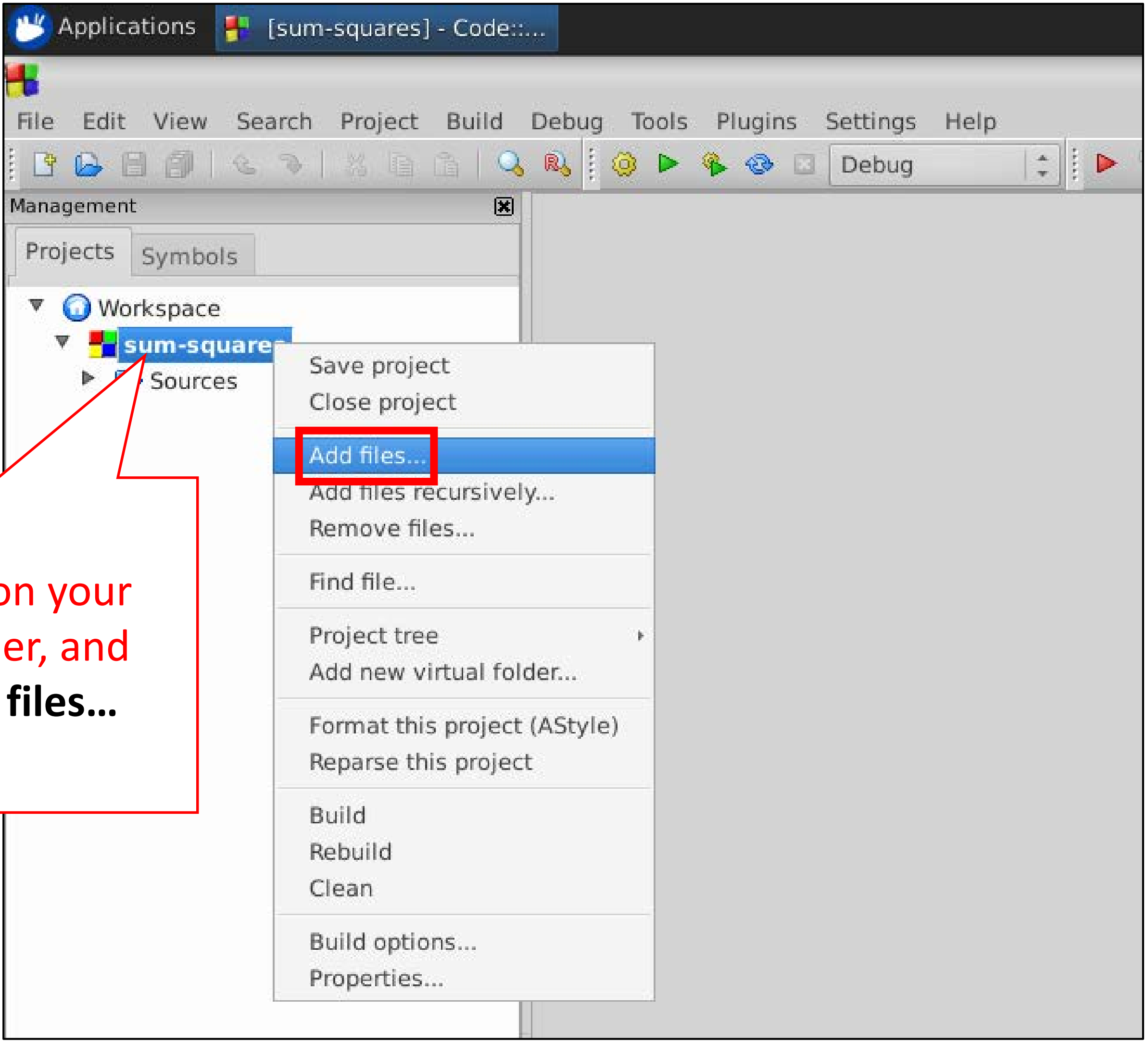
# **stadafx.h** – The Standard Application Framework Header File



```
// stdafx.h

#pragma once

#include <stdio.h>
#include <time.h>
#define _USE_MATH_DEFINES
#include <math.h>
#include <algorithm>
#include <array>
#include <assert.h>
#include <atomic>
#include <chrono>
#include <cmath>
#include <complex>
#include <cstdio>
#include <fstream>
#include <functional>
#include <iomanip>
#include <iostream>
#include <iterator>
#include <limits>
#include <locale>
#include <memory>
#include <map>
#include <mutex>
#include <random>
#include <sstream>
#include <string>
#include <thread>
#include <time.h>
#include <tuple>
#include <unordered_map>
#include <utility>
#include <vector>
```

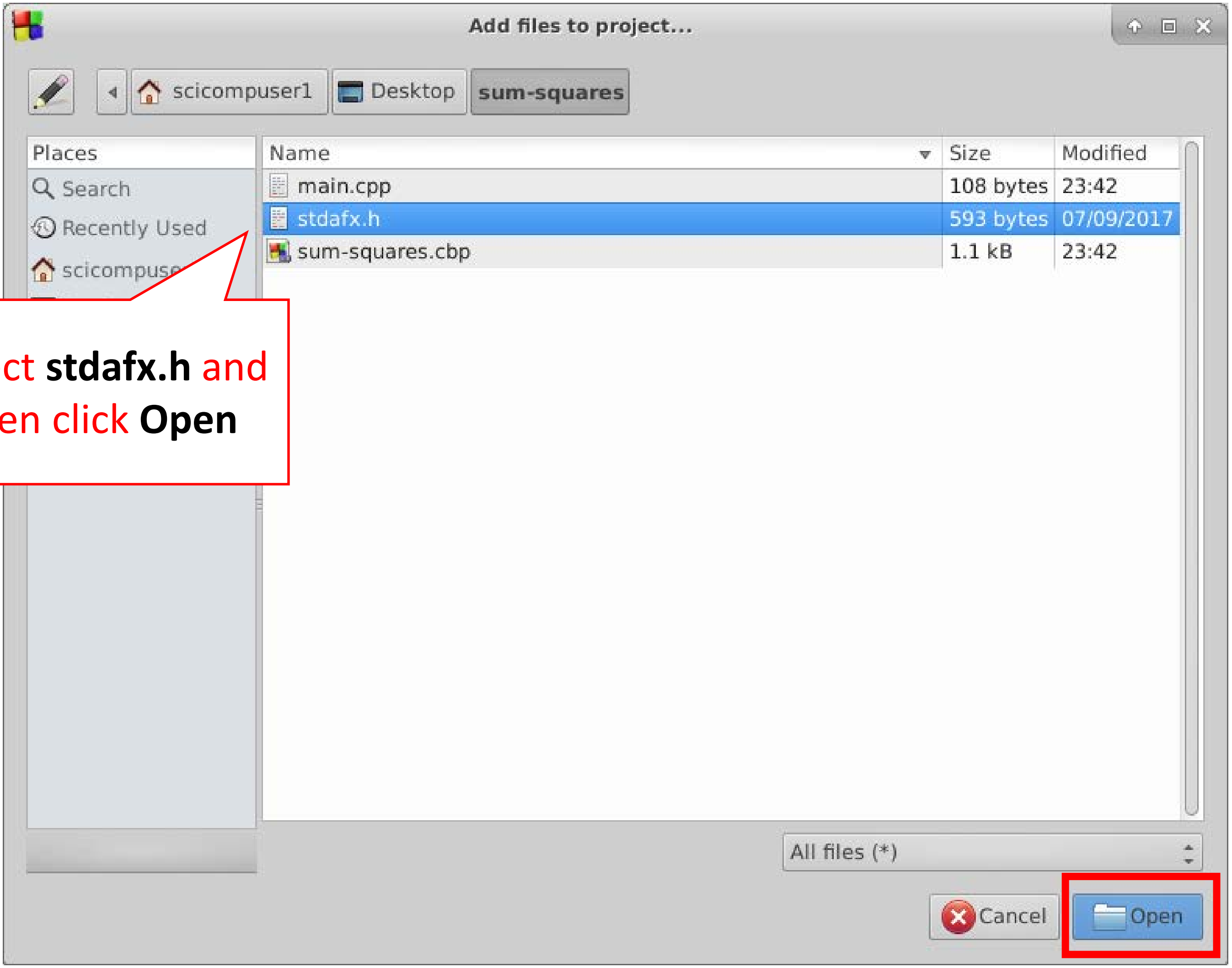
The **stdafx.h** file ***includes*** other header files. Header files (.h) contain variable & function declarations used in the source (.cpp) files in your project. These particular header files cover all of the C++ functions we will likely need for our labs

# Add “stdafx.h” file to your project



**Right click** on your project folder, and select **Add files…**

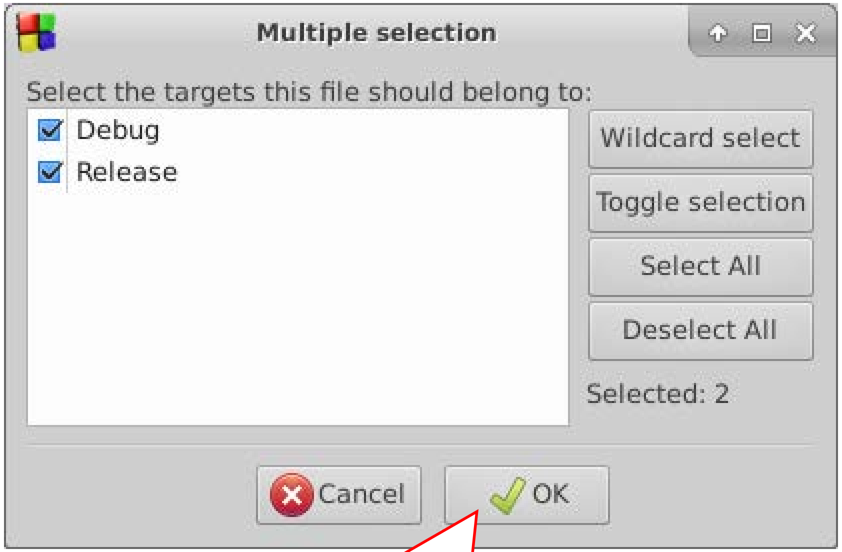
# Add “**stdafx.h**” file to your project



| Name | Size | Modified |
|---|---|---|
| main.cpp | 108 bytes | 23:42 |
| stdafx.h | 593 bytes | 07/09/2017 |
| sum-squares.cbp | 1.1 kB | 23:42 |

Select **stdafx.h** and then click **Open**

# Add "stdafx.h" file to your project



Multiple selection

Select the targets this file should belong to:

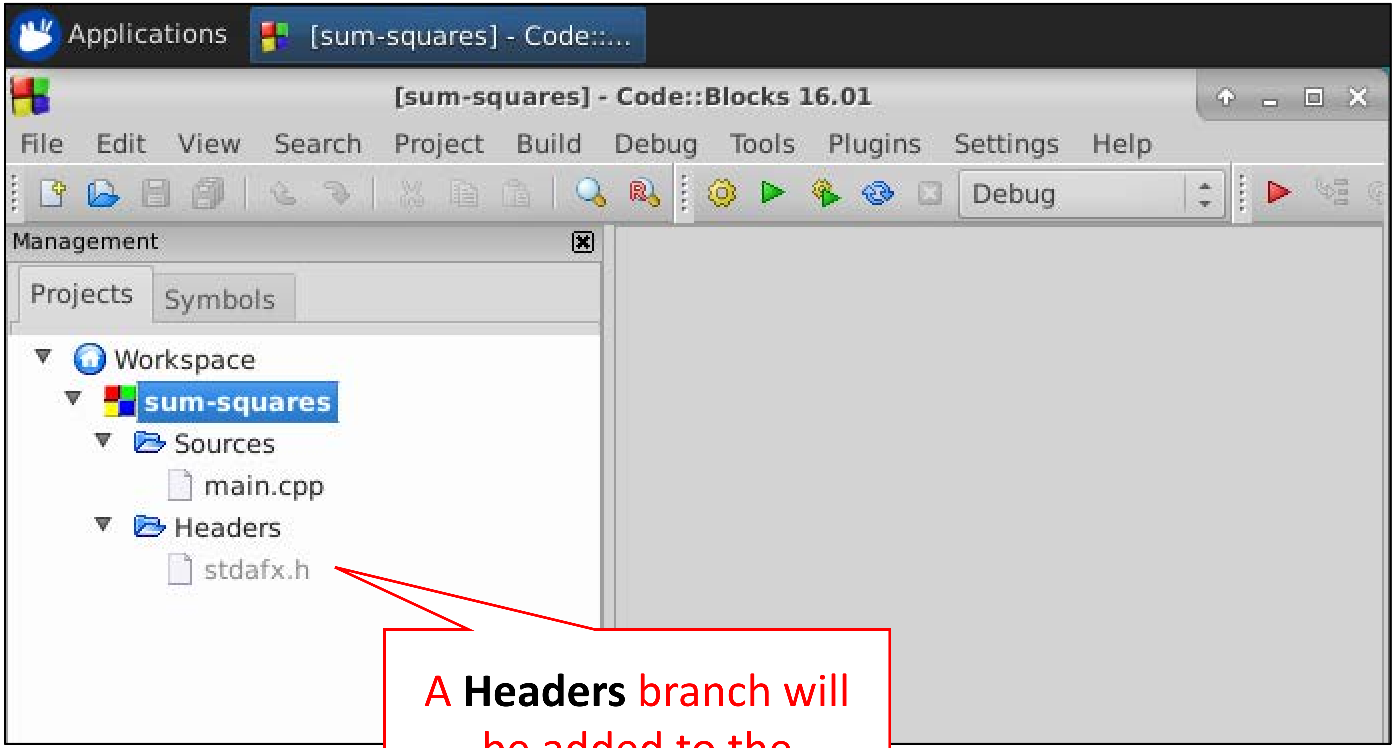
☑ Debug

☑ Release

Selected: 2

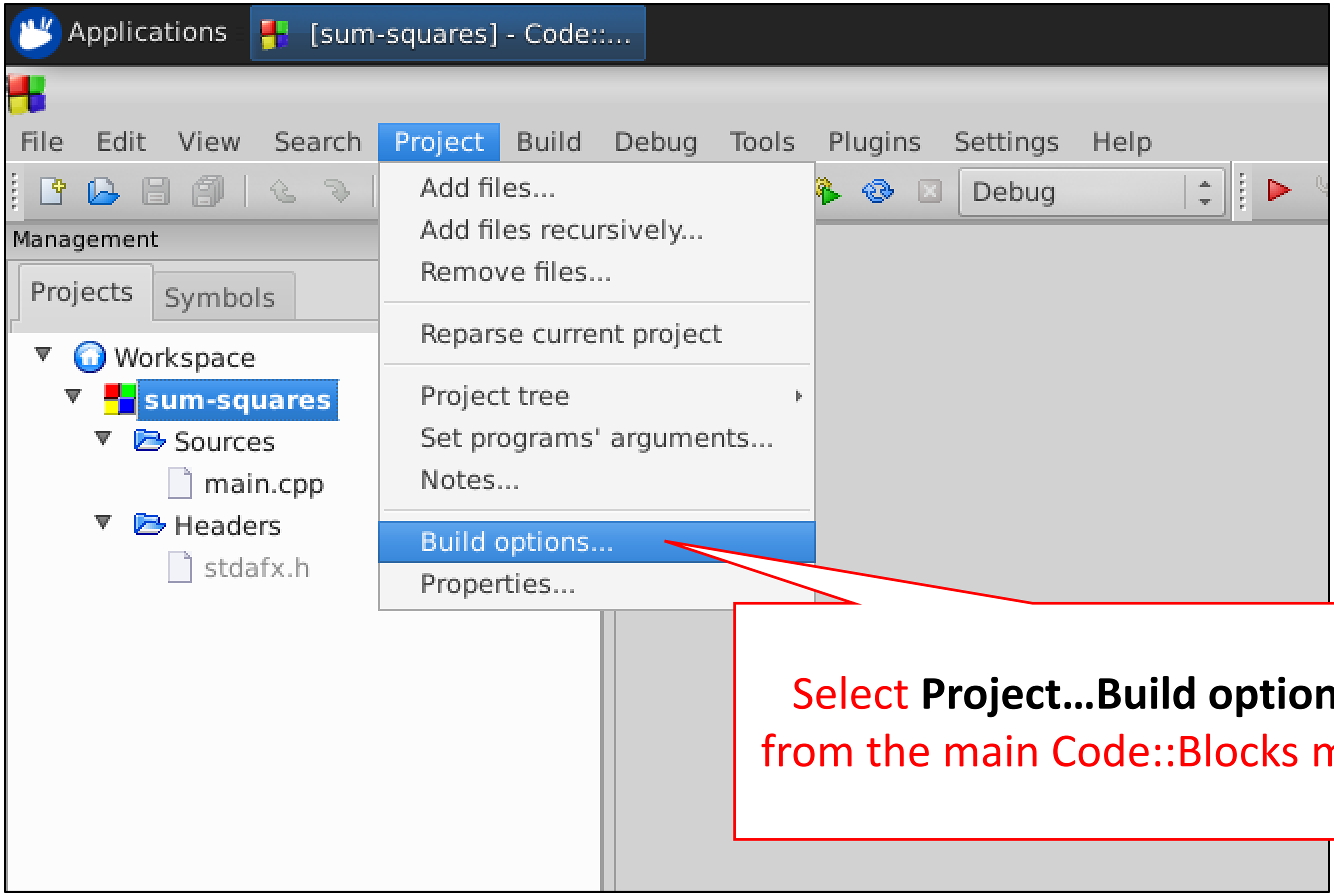
Accept the default targets by clicking **OK**

# Add “**stdafx.h**” file to your project


Applications
[sum-squares] - Code::...
[sum-squares] - Code::Blocks 16.01
File Edit View Search Project Build Debug Tools Plugins Settings Help
Debug
Management
Projects
Symbols
Workspace
sum-squares
Sources
main.cpp
Headers
stdafx.h

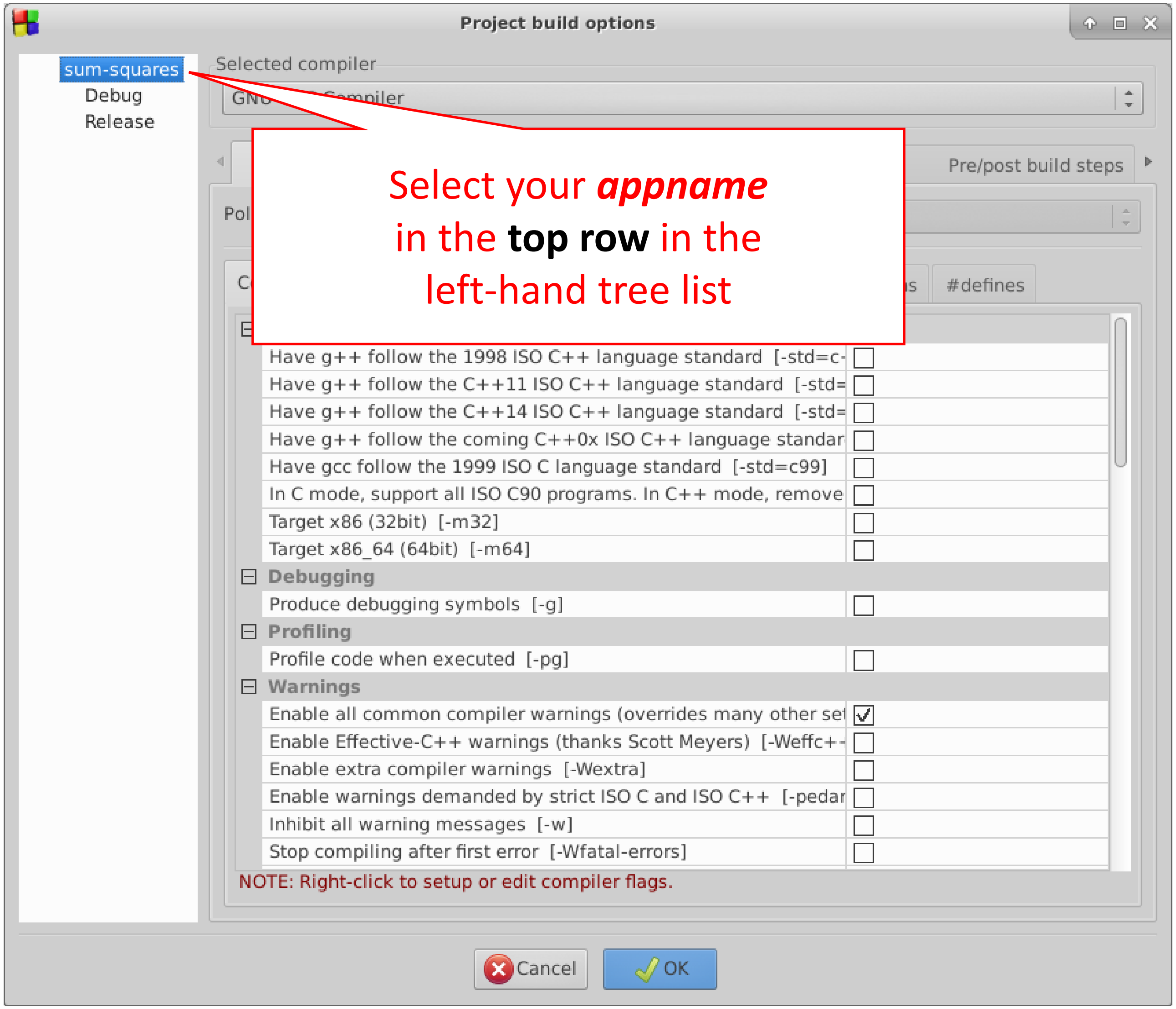
A **Headers** branch will be added to the **Management** pane

# Set Project Build Options



Select **Project...Build options...** from the main Code::Blocks menu

# Set Project Build Options



Select your ***appname*** in the **top row** in the left-hand tree list

- Have g++ follow the 1998 ISO C++ language standard [-std=c- □
- Have g++ follow the C++11 ISO C++ language standard [-std= □
- Have g++ follow the C++14 ISO C++ language standard [-std= □
- Have g++ follow the coming C++0x ISO C++ language standar □
- Have gcc follow the 1999 ISO C language standard [-std=c99] □
- In C mode, support all ISO C90 programs. In C++ mode, remove □
- Target x86 (32bit) [-m32] □
- Target x86_64 (64bit) [-m64] □

Debugging

- Produce debugging symbols [-g] □

Profiling

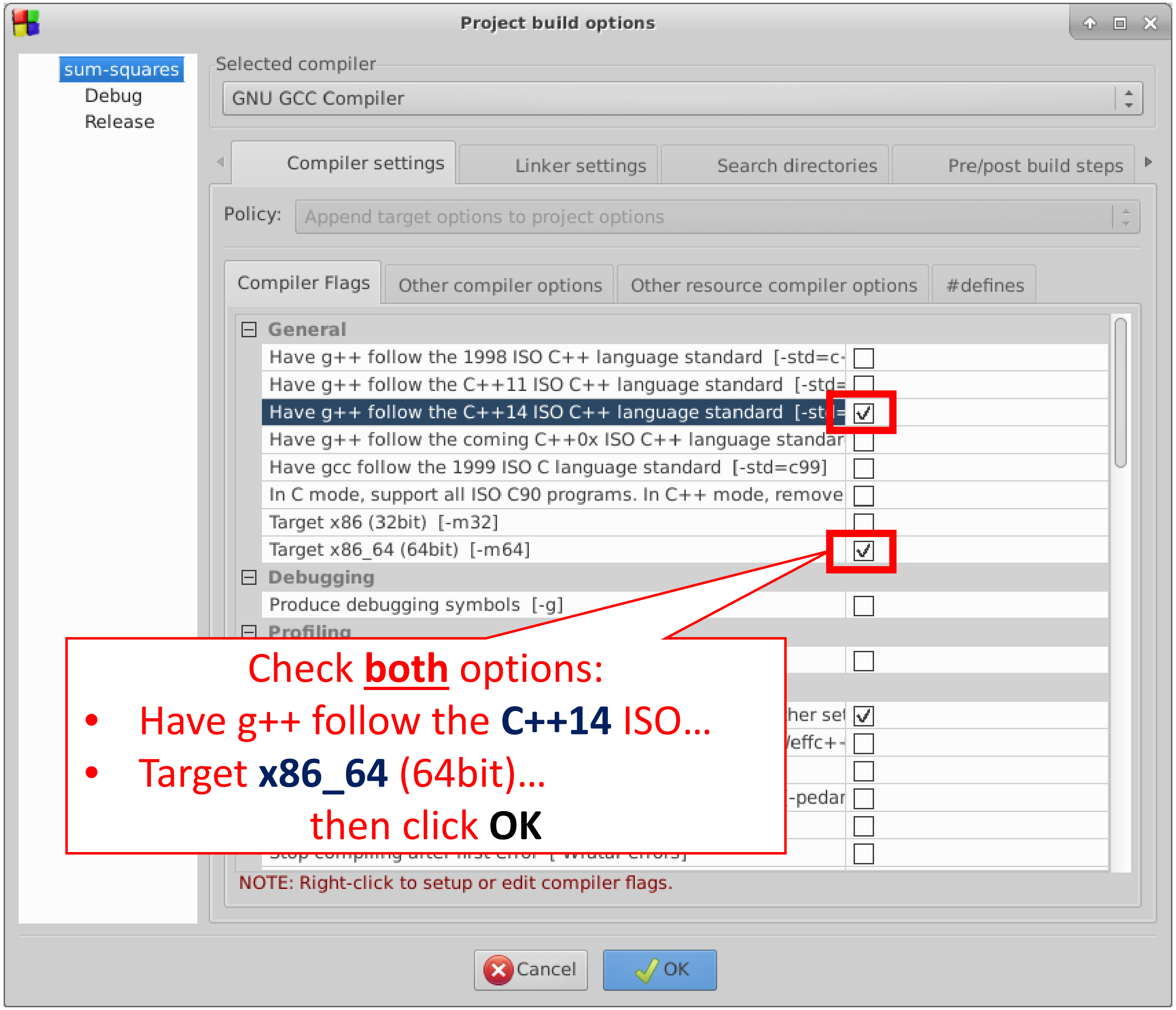
- Profile code when executed [-pg] □

Warnings

- Enable all common compiler warnings (overrides many other set ☑
- Enable Effective-C++ warnings (thanks Scott Meyers) [-Weffc+- □
- Enable extra compiler warnings [-Wextra] □
- Enable warnings demanded by strict ISO C and ISO C++ [-pedar □
- Inhibit all warning messages [-w] □
- Stop compiling after first error [-Wfatal-errors] □

NOTE: Right-click to setup or edit compiler flags.

# Set Project Build Options



Selected compiler: GNU GCC Compiler

Policy: Append target options to project options

General
- Have g++ follow the 1998 ISO C++ language standard [-std=c- □
- Have g++ follow the C++11 ISO C++ language standard [-std= □
- Have g++ follow the C++14 ISO C++ language standard [-st ☑
- Have g++ follow the coming C++0x ISO C++ language standar □
- Have gcc follow the 1999 ISO C language standard [-std=c99] □
- In C mode, support all ISO C90 programs. In C++ mode, remove □
- Target x86 (32bit) [-m32] □
- Target x86_64 (64bit) [-m64] ☑

Debugging
- Produce debugging symbols [-g] □

Profiling
- □
- her set ☑
- /effc+- □
- □
- -pedar □
- □
- Stop compiling after first error [-Wfatal-errors] □

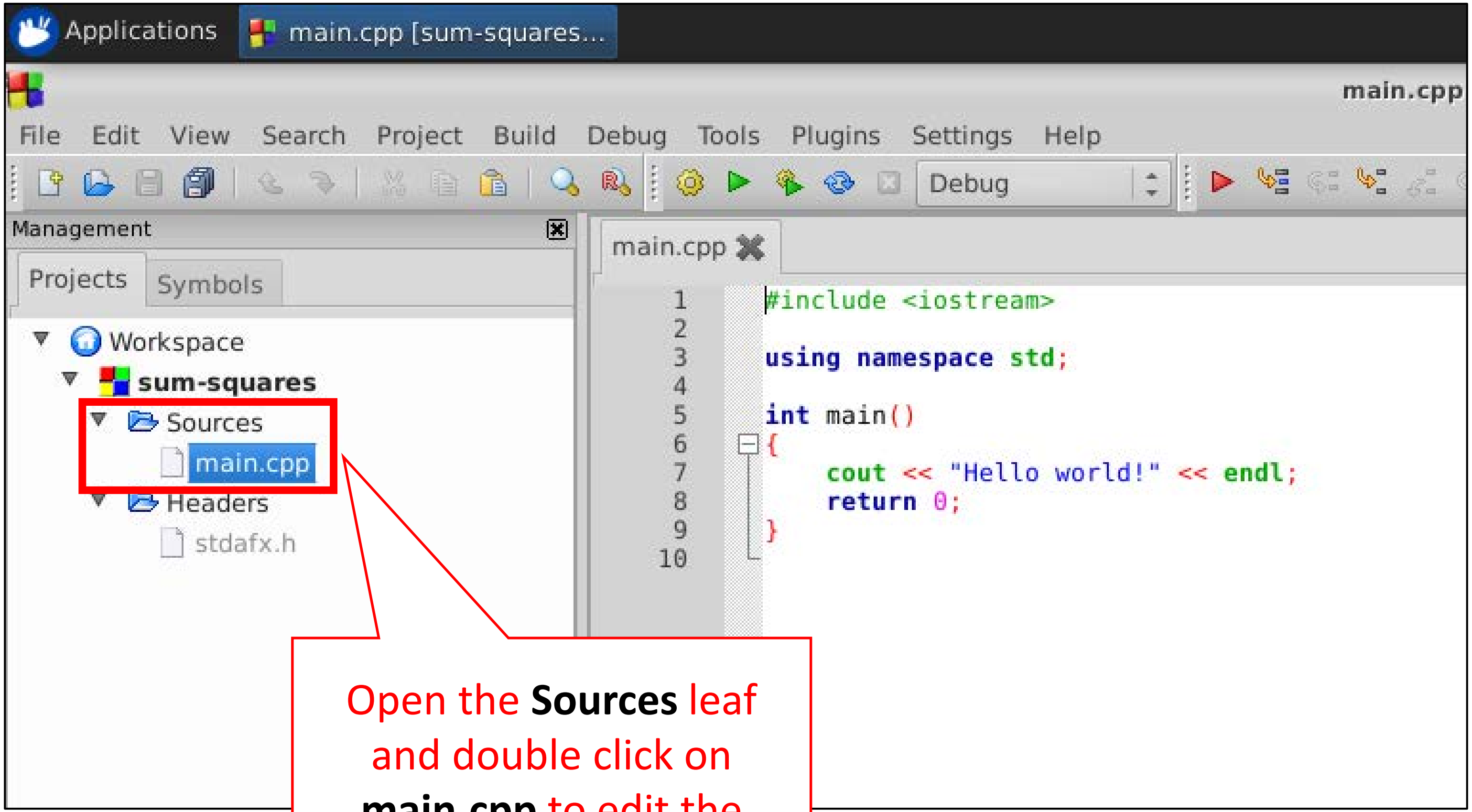
NOTE: Right-click to setup or edit compiler flags.

Check **both** options:

- Have g++ follow the **C++14** ISO…
- Target **x86_64** (64bit)…

then click **OK**

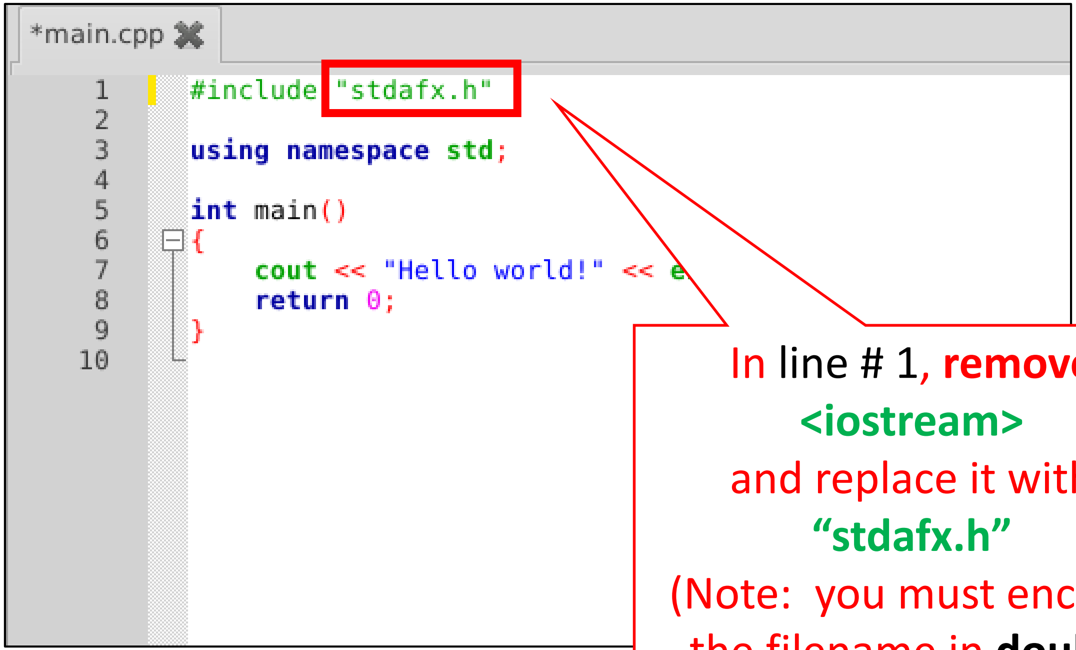
# Edit “main.cpp”



```
#include <iostream>

using namespace std;

int main()
{
    cout << "Hello world!" << endl;
    return 0;
}
```

Open the **Sources** leaf and double click on **main.cpp** to edit the primary source code file

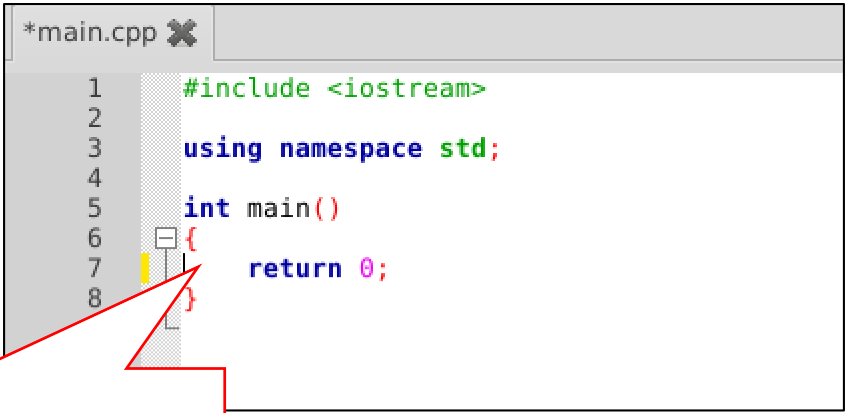
# Edit “main.cpp”



```
1   #include "stdafx.h"
2
3   using namespace std;
4
5   int main()
6   {
7       cout << "Hello world!" << e
8       return 0;
9   }
10
```

In line # 1, **remove** **<iostream>**
and replace it with
**“stdafx.h”**
(Note: you must enclose
the filename in **double**
quotation marks)

# Edit “main.cpp”

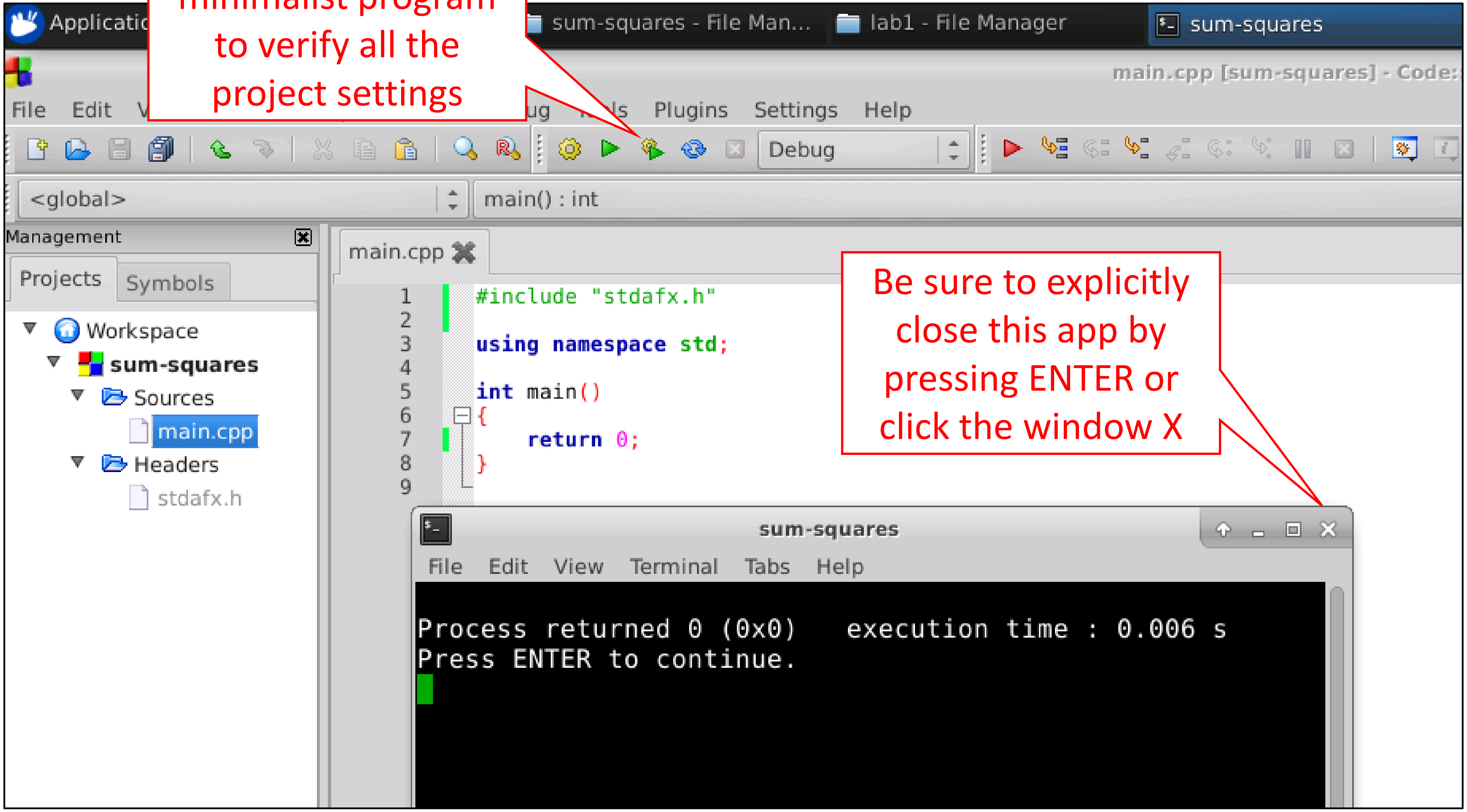


```
1  #include <iostream>
2
3  using namespace std;
4
5  int main()
6  {
7      return 0;
8  }
```

**Delete line # 7**
**cout << “Hello World”**

# Edit “main.cpp”

Build and run this minimalist program to verify all the project settings



```
#include "stdafx.h"

using namespace std;

int main()
{
    return 0;
}
```

```
Process returned 0 (0x0)   execution time : 0.006 s
Press ENTER to continue.
```

Be sure to explicitly close this app by pressing ENTER or click the window X

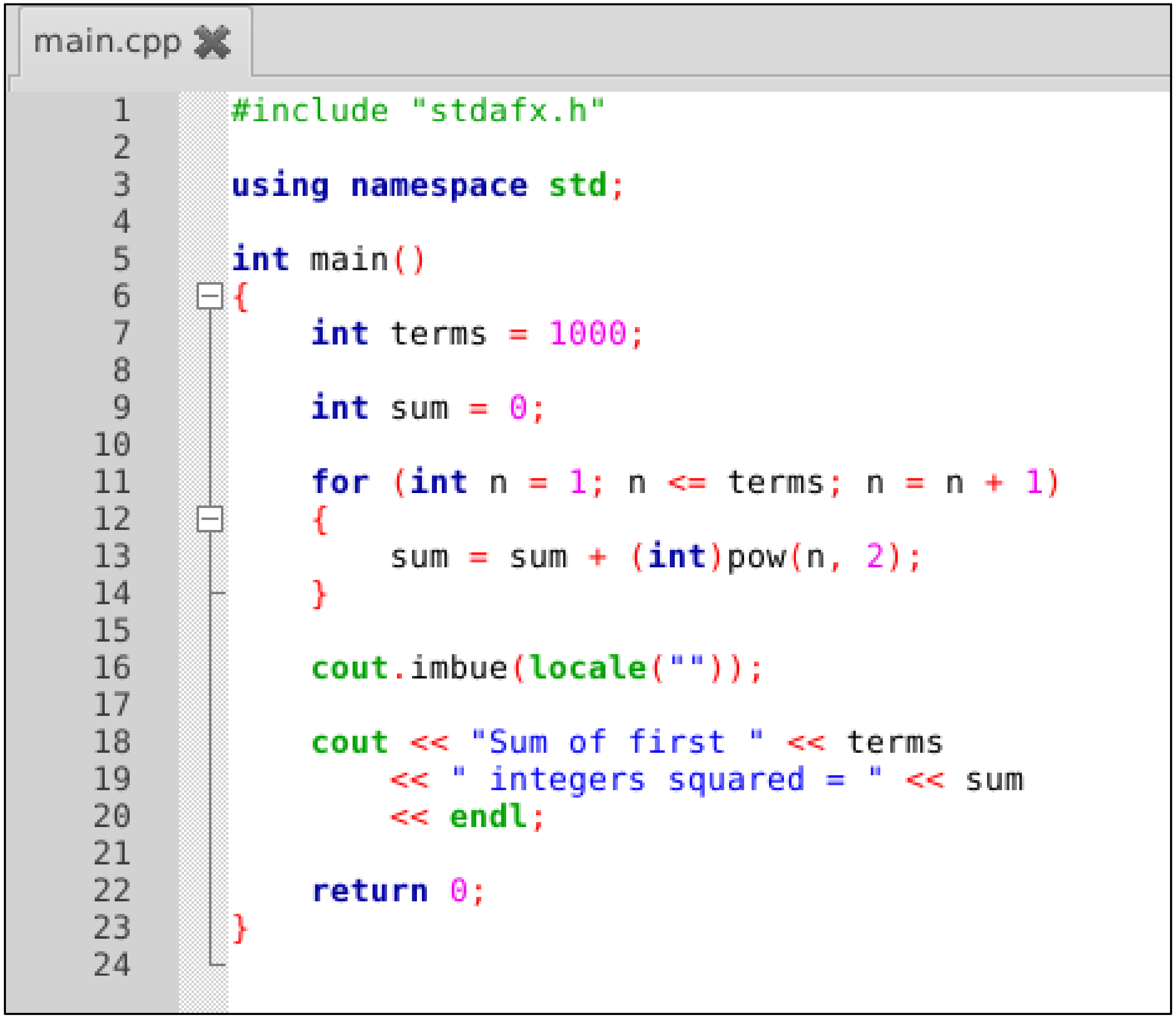
# Lab 1 – Sum of Squares

- Write a program to calculate the sum of the squares of the first 1000 natural numbers

| n | n^2 | Sum |
|---:|---:|---:|
| 1 | 1 | 1 |
| 2 | 4 | 5 |
| 3 | 9 | 14 |
| 4 | 16 | 30 |
| 5 | 25 | 55 |
| 6 | 36 | 91 |
| 7 | 49 | 140 |
| 8 | 64 | 204 |
| 9 | 81 | 285 |
| 10 | 100 | 385 |

$$P_n = \sum_{k=1}^{n} k^2 = \frac{n(n+1)(2n+1)}{6} = \boxed{\frac{2n^3+3n^2+n}{6}}.$$

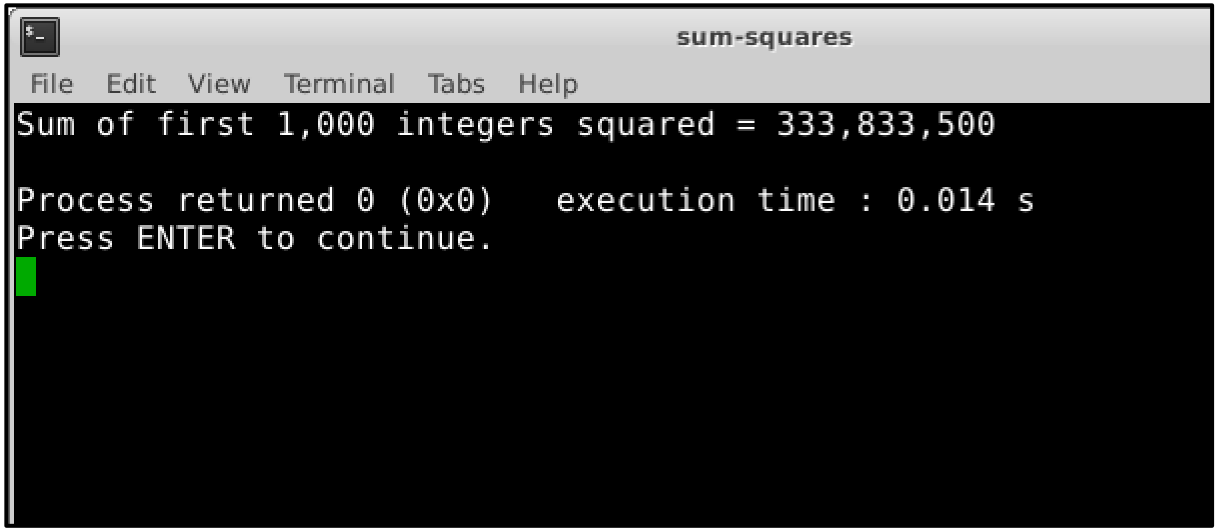
# Lab 1 – Sum of Squares



```cpp
#include "stdafx.h"

using namespace std;

int main()
{
    int terms = 1000;

    int sum = 0;

    for (int n = 1; n <= terms; n = n + 1)
    {
        sum = sum + (int)pow(n, 2);
    }

    cout.imbue(locale(""));

    cout << "Sum of first " << terms
        << " integers squared = " << sum
        << endl;

    return 0;
}
```

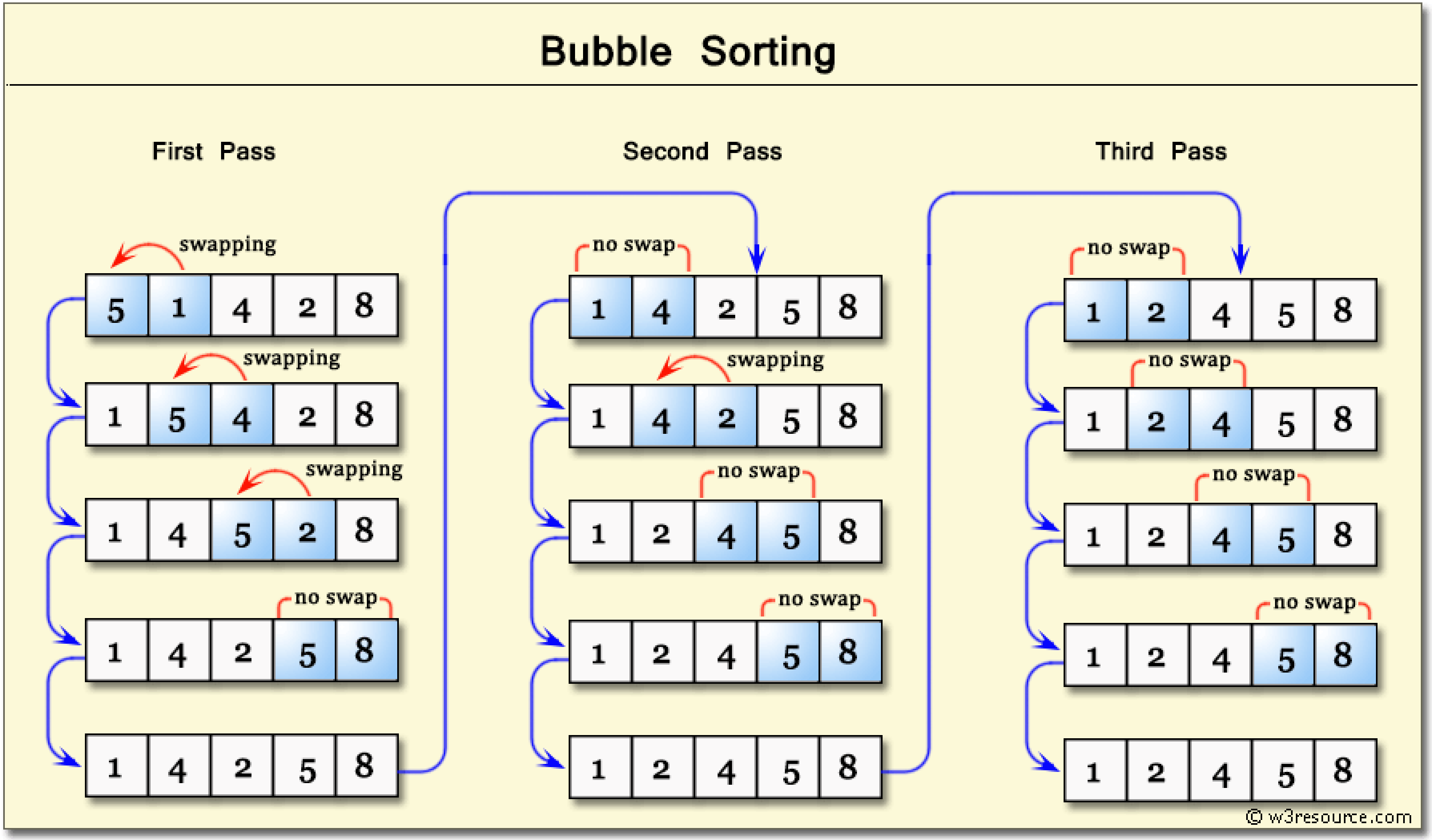
# Lab 1 – Sum of Squares

- Write a program to calculate the sum of the squares of the first 1000 natural numbers



```
Sum of first 1,000 integers squared = 333,833,500

Process returned 0 (0x0)   execution time : 0.014 s
Press ENTER to continue.
```

# Lab 2 – Bubble Sort



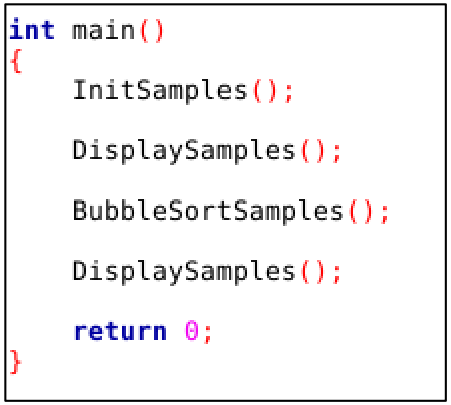
# Lab 2 – Bubble Sort

- Create a new Code::Blocks C++/14 console application project called **bubble-sort** located at this path:

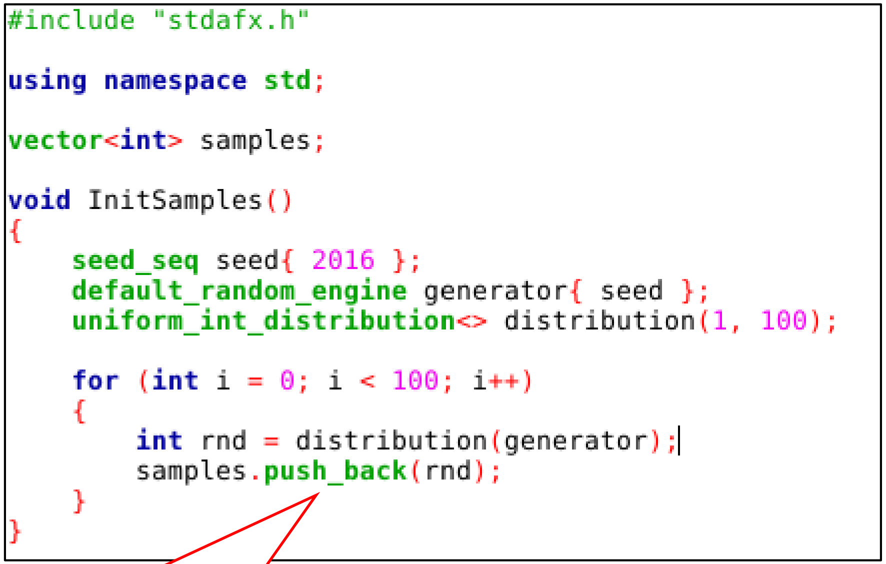
```
~/Desktop/scicomp301/session07/lab2/bubble-sort/bubble-sort.cpb
```

- Your code should initialize a vector of **100** random integers (uniform distribution) each having a value between **1** and **100** inclusive with an initial seed value of **2016**
  - Your code should first display all the elements in the unsorted vector
  - Your code should then **bubble sort** the vector in increasing order (lowest items move the front of the vector)
  - Finally your code should display all the elements in the sorted vector

# Lab 2 – Bubble Sort



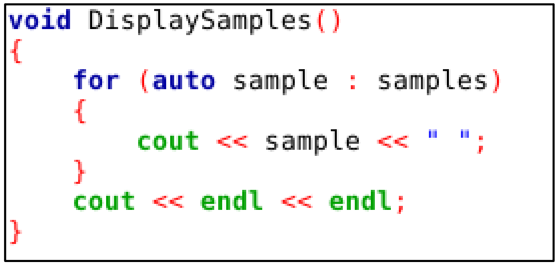
```
int main()
{
	InitSamples();

	DisplaySamples();

	BubbleSortSamples();

	DisplaySamples();

	return 0;
}
```



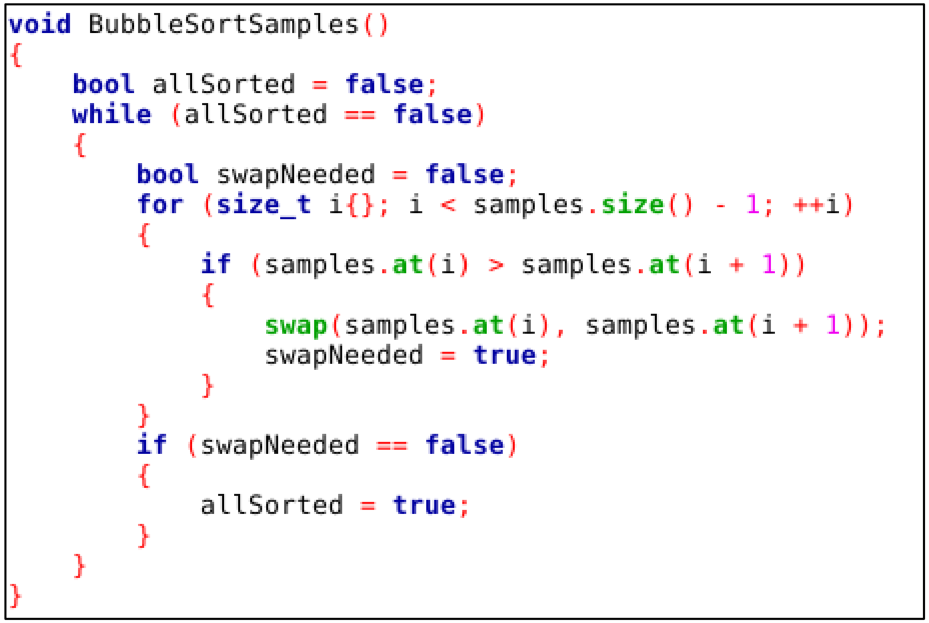
```
#include "stdafx.h"

using namespace std;

vector<int> samples;

void InitSamples()
{
	seed_seq seed{ 2016 };
	default_random_engine generator{ seed };
	uniform_int_distribution<> distribution(1, 100);

	for (int i = 0; i < 100; i++)
	{
		int rnd = distribution(generator);
		samples.push_back(rnd);
	}
}
```

**push_back()** appends the value in parenthesis to the end of the given **vector**

# Lab 2 – Bubble Sort

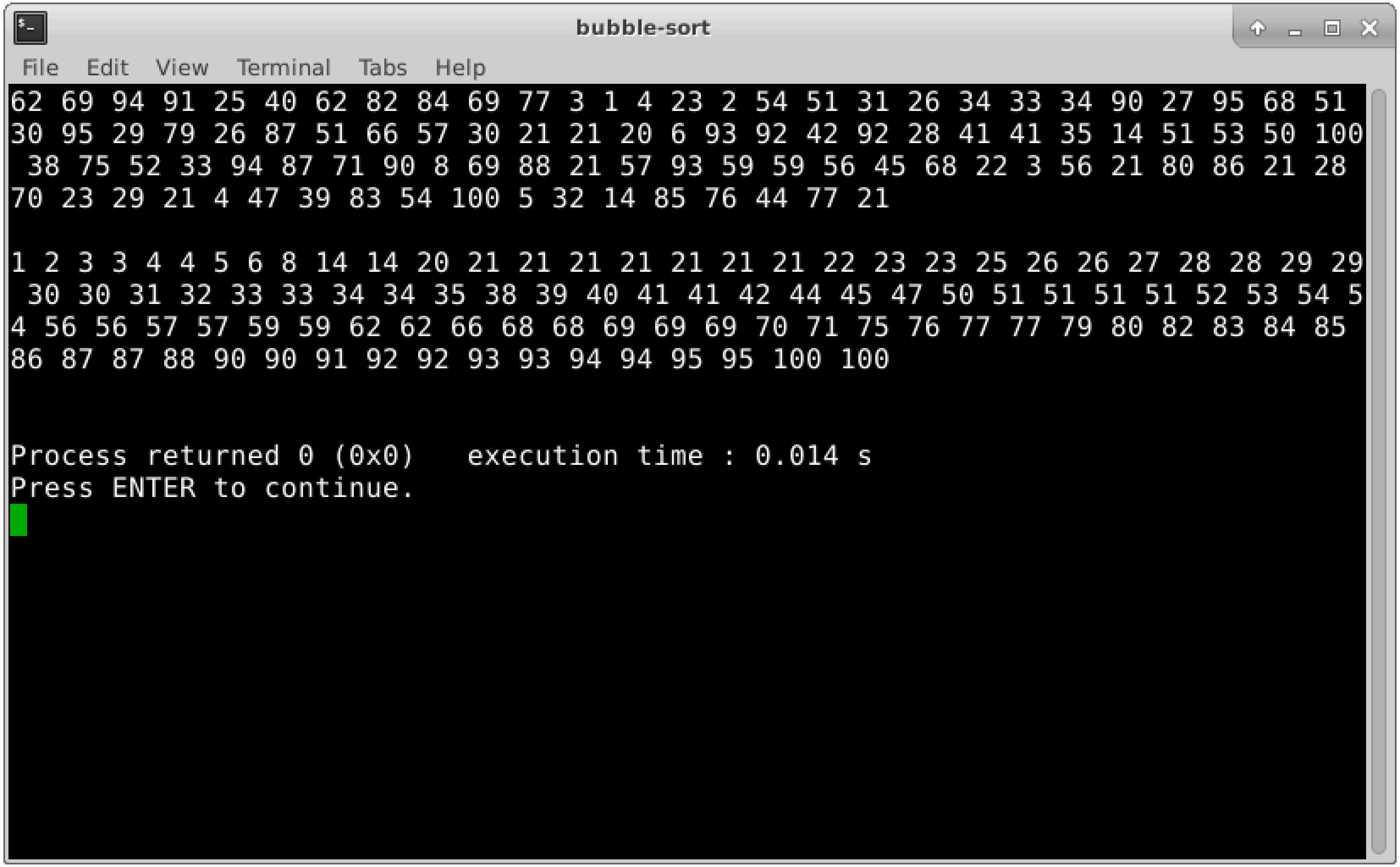


```
void DisplaySamples()
{
    for (auto sample : samples)
    {
        cout << sample << " ";
    }
    cout << endl << endl;
}
```



```
void BubbleSortSamples()
{
    bool allSorted = false;
    while (allSorted == false)
    {
        bool swapNeeded = false;
        for (size_t i{}; i < samples.size() - 1; ++i)
        {
            if (samples.at(i) > samples.at(i + 1))
            {
                swap(samples.at(i), samples.at(i + 1));
                swapNeeded = true;
            }
        }
        if (swapNeeded == false)
        {
            allSorted = true;
        }
    }
}
```

# Lab 2 – Bubble Sort



```
62 69 94 91 25 40 62 82 84 69 77 3 1 4 23 2 54 51 31 26 34 33 34 90 27 95 68 51
30 95 29 79 26 87 51 66 57 30 21 21 20 6 93 92 42 92 28 41 41 35 14 51 53 50 100
 38 75 52 33 94 87 71 90 8 69 88 21 57 93 59 59 56 45 68 22 3 56 21 80 86 21 28
70 23 29 21 4 47 39 83 54 100 5 32 14 85 76 44 77 21

1 2 3 3 4 4 5 6 8 14 14 20 21 21 21 21 21 21 21 22 23 23 25 26 26 27 28 28 29 29
 30 30 31 32 33 33 34 34 35 38 39 40 41 41 42 44 45 47 50 51 51 51 51 52 53 54 5
4 56 56 57 57 59 59 62 62 66 68 68 69 69 69 70 71 75 76 77 77 79 80 82 83 84 85
86 87 87 88 90 90 91 92 92 93 93 94 94 95 95 100 100



Process returned 0 (0x0)   execution time : 0.014 s
Press ENTER to continue.
```

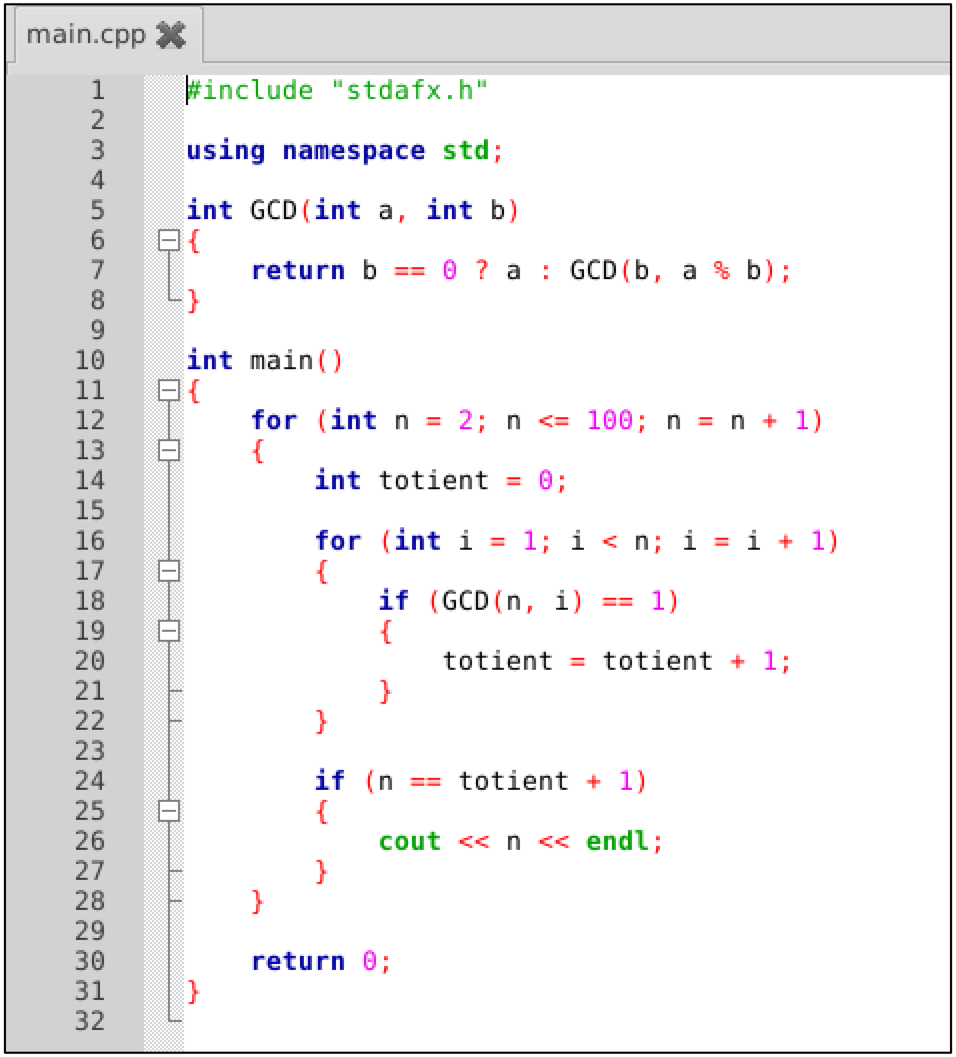
# Lab 3 – Euler's Totient

- Create a new Code::Blocks C++/14 console application project called **euler-totient** located at this path:

```
~/Desktop/scicomp301/session07/lab3/euler-totient/euler-totient.cpb
```

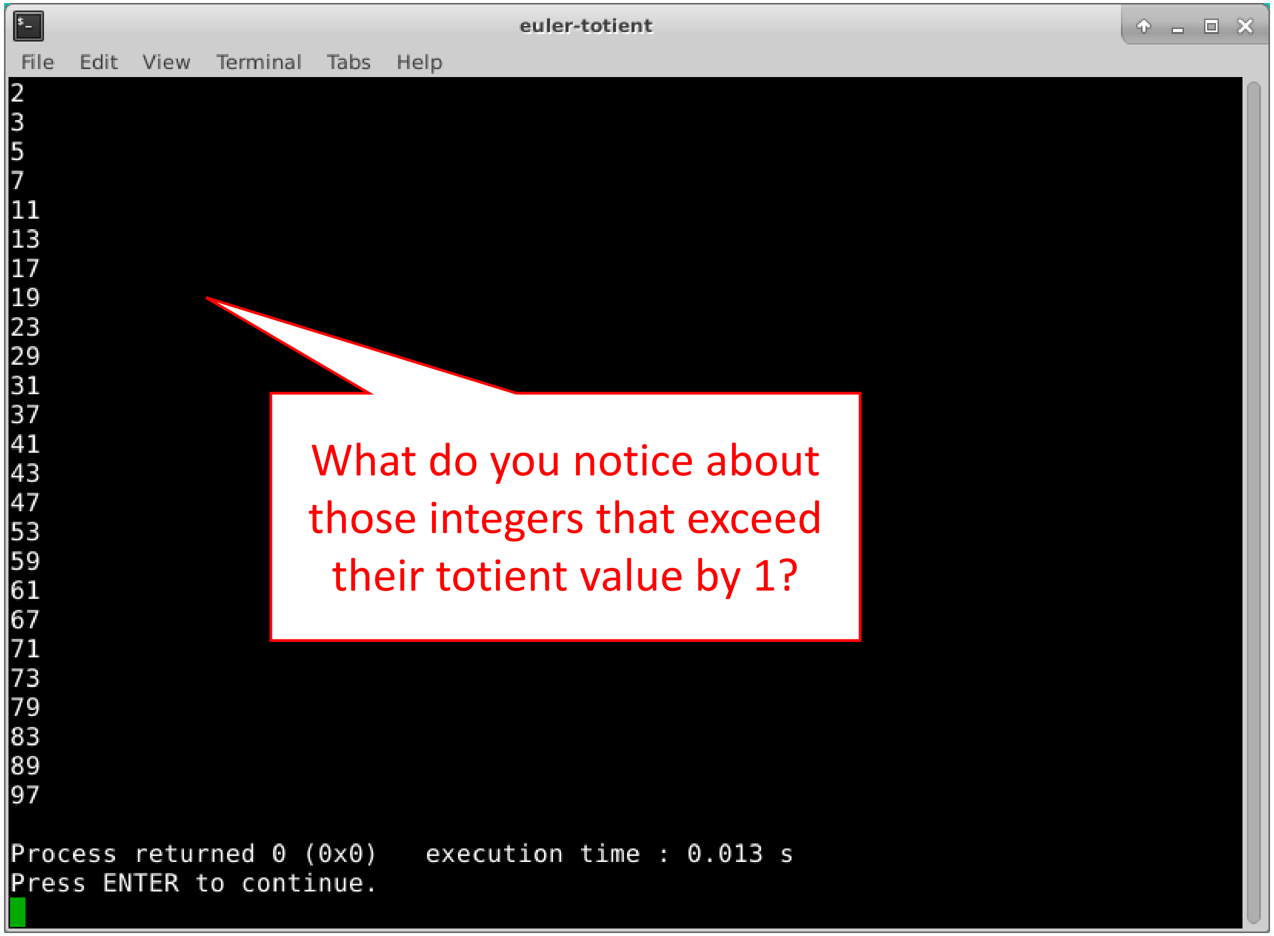
- Using the **GCD** function, your code should calculate Euler's Totient for all integers between **2** and **100** inclusive
    - The totient of an integer is the "number of positive integers" less than the given integer that are relatively prime to that integer
    - Totient(12) = 4  because only {1, 5, 7, 11} are relatively prime to 12
    - http://www.wolframalpha.com/input/?i=totient(12)
- Your program should display **only** those integers whose **value exceeds its own totient value by 1 exactly**
    - What do you notice about those integers that are displayed?

# Lab 3 – Euler's Totient



```cpp
#include "stdafx.h"

using namespace std;

int GCD(int a, int b)
{
    return b == 0 ? a : GCD(b, a % b);
}

int main()
{
    for (int n = 2; n <= 100; n = n + 1)
    {
        int totient = 0;

        for (int i = 1; i < n; i = i + 1)
        {
            if (GCD(n, i) == 1)
            {
                totient = totient + 1;
            }
        }

        if (n == totient + 1)
        {
            cout << n << endl;
        }
    }

    return 0;
}

```

# Lab 3 – Euler's Totient



```
2
3
5
7
11
13
17
19
23
29
31
37
41
43
47
53
59
61
67
71
73
79
83
89
97

Process returned 0 (0x0)   execution time : 0.013 s
Press ENTER to continue.
```

What do you notice about those integers that exceed their totient value by 1?

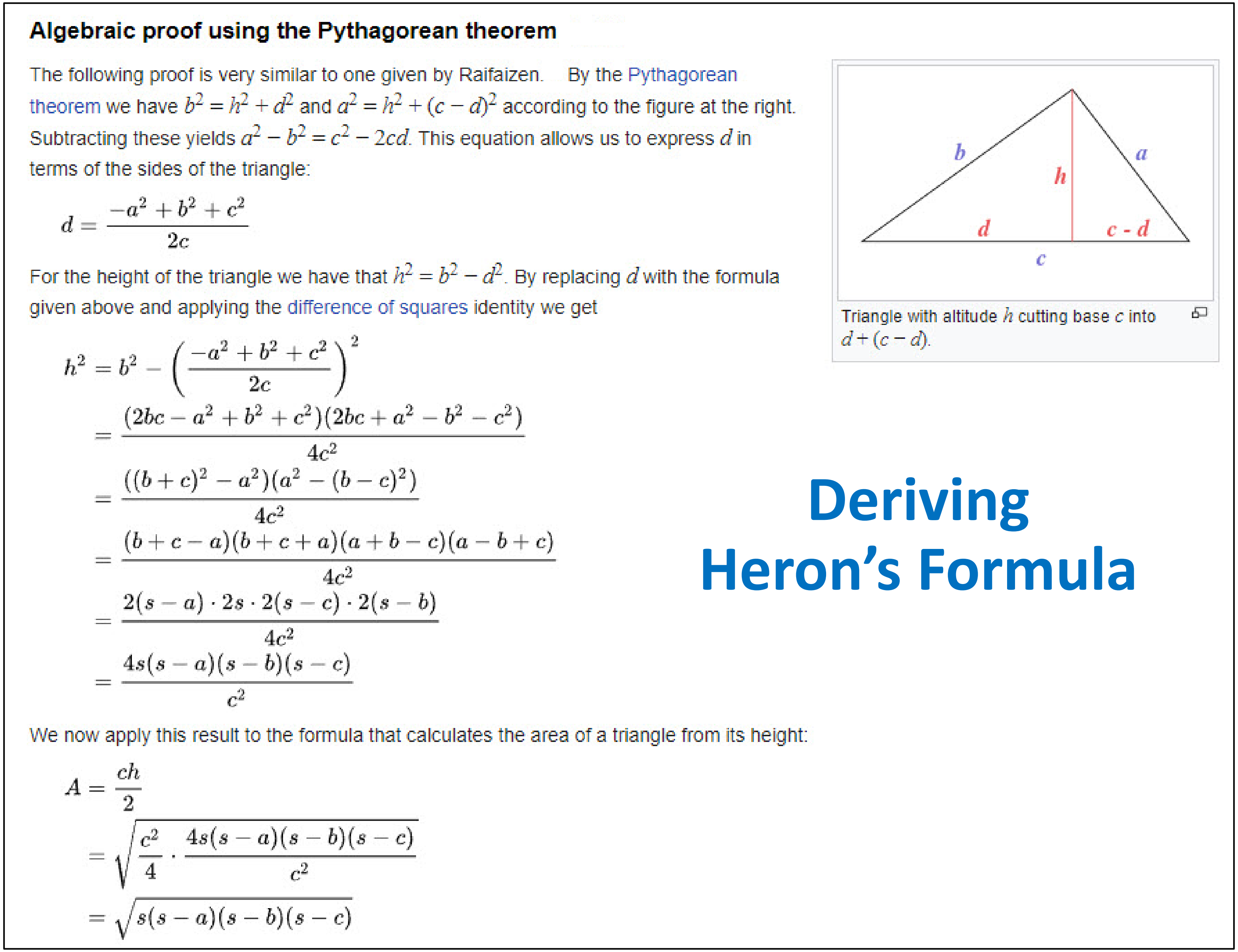
# Lab 4 – Heron's Formula

- Create a new Code::Blocks C++/14 console application project called **herons-formula** located at this path:

```
~/Desktop/scicomp301/session07/lab4/herons-formula/herons-formula.cpb
```

- Use **Heron's Formula** to calculate and display the lengths of every side and the **area** of **10** random triangles
- The length of each side of each triangle should be a uniformly distributed random integer within **[1, 100]**
- Use the **triangle inequality theorem** to exclude any *invalid* triangle from your list of 10
- Include any **degenerate triangles** in your list of 10

# Deriving Heron’s Formula



### Algebraic proof using the Pythagorean theorem

The following proof is very similar to one given by Raifaizen. By the Pythagorean theorem we have $b^2 = h^2 + d^2$ and $a^2 = h^2 + (c - d)^2$ according to the figure at the right. Subtracting these yields $a^2 - b^2 = c^2 - 2cd$. This equation allows us to express $d$ in terms of the sides of the triangle:

$$d = \frac{-a^2 + b^2 + c^2}{2c}$$

For the height of the triangle we have that $h^2 = b^2 - d^2$. By replacing $d$ with the formula given above and applying the difference of squares identity we get

$$\begin{aligned} h^2 &= b^2 - \left(\frac{-a^2 + b^2 + c^2}{2c}\right)^2 \\ &= \frac{(2bc - a^2 + b^2 + c^2)(2bc + a^2 - b^2 - c^2)}{4c^2} \\ &= \frac{((b+c)^2 - a^2)(a^2 - (b-c)^2)}{4c^2} \\ &= \frac{(b+c-a)(b+c+a)(a+b-c)(a-b+c)}{4c^2} \\ &= \frac{2(s-a) \cdot 2s \cdot 2(s-c) \cdot 2(s-b)}{4c^2} \\ &= \frac{4s(s-a)(s-b)(s-c)}{c^2} \end{aligned}$$

We now apply this result to the formula that calculates the area of a triangle from its height:

$$\begin{aligned} A &= \frac{ch}{2} \\ &= \sqrt{\frac{c^2}{4} \cdot \frac{4s(s-a)(s-b)(s-c)}{c^2}} \\ &= \sqrt{s(s-a)(s-b)(s-c)} \end{aligned}$$

Triangle with altitude $h$ cutting base $c$ into $d + (c - d)$.

# Lab 5 – Statistics

- Create a new Code::Blocks C++/14 console application project called **statistics** located at this path:

  `~/Desktop/scicomp301/session07/lab5/statistics/statistics.cpb`

- Generate and display a **vector** of **10** uniformly distributed random integers all between 0 and 100 inclusive, using a PRNG seed of **2016**

- Calculate and display the following five (5) **population** statistics that describe the vector:

1. Mean
2. Median
3. **Mode**
4. Variance
5. Standard Deviation

# Now you know…

- How to create a “starter” (blank) C++/14 console application using the Code::Blocks IDE
- The purpose of the **stdafx.h** file and how to add it to a Code::Blocks project
- How the **bubble sort** algorithm can order the elements of a given vector
- Euler’s **Totient** function returns the number of integers less than a given integer that are relatively prime to that integer
- The median is a tricky statistic depending if you have an **odd** or **even** number of elements in your set
- Finding the mode requires counting element **occurrences**