fundamentals of programming I

Lab 1

What is C++?

- C++ is a cross-platform language that can be used to create high-performance applications.
- C++ was developed by Bjarne Stroustrup, as an extension to the C language.
- C++ gives programmers a high level of control over system resources and memory.

Why Use C++?

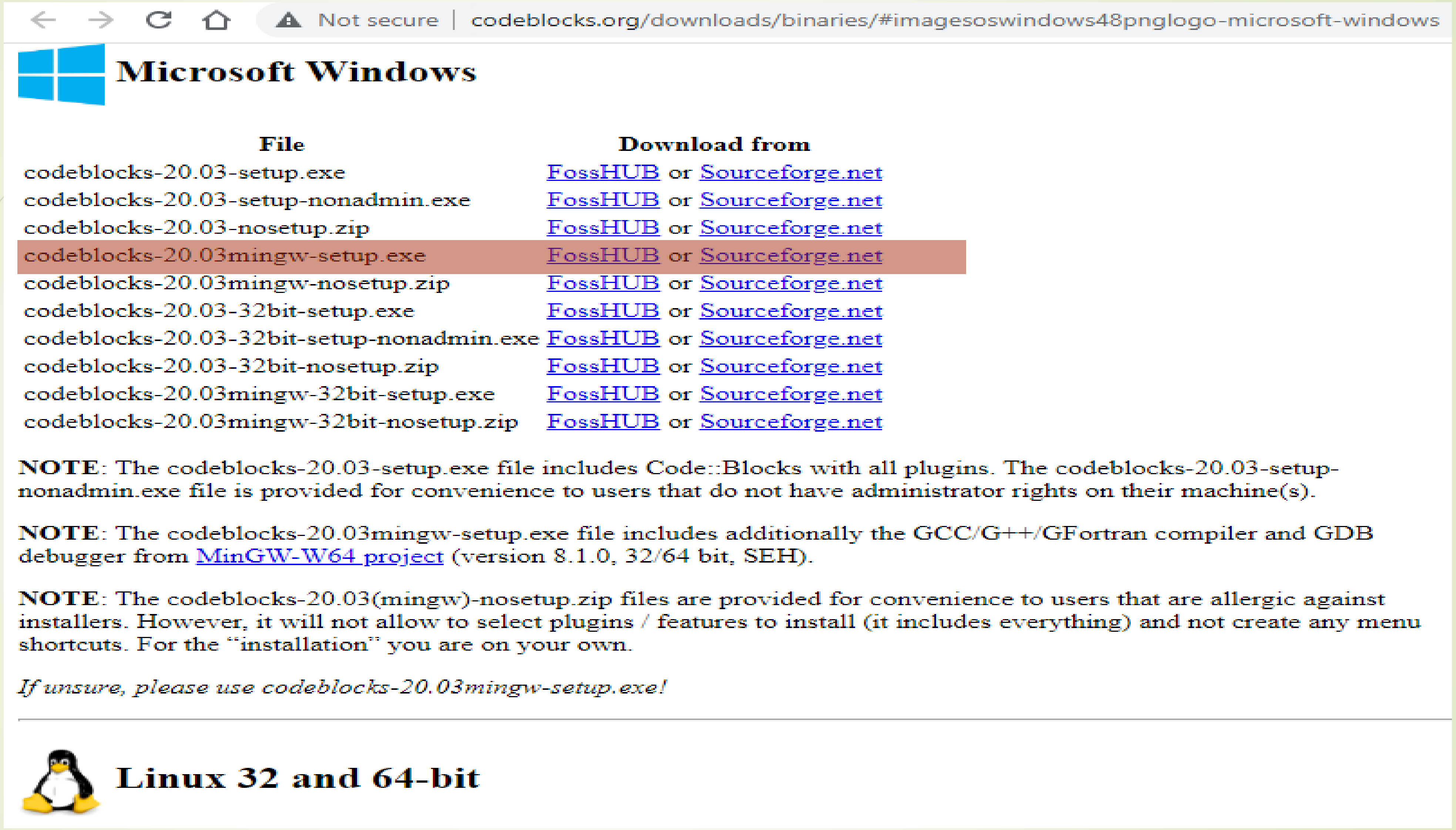
- C++ is one of the world's most popular programming languages.
- C++ can be found in today's operating systems, Graphical User Interfaces, and embedded systems.
- C++ is an object-oriented programming language which gives a clear structure to programs and allows code to be reused, lowering development costs.
- C++ is portable and can be used to develop applications that can be adapted to multiple platforms.
- C++ is fun and easy to learn!
- As C++ is close to C, C# and Java, it makes it easy for programmers to switch to C++ or vice versa

C++ Get Started

- To start using C++, you need two things:
- A text editor, like Notepad, to write C++ code
- A compiler, like GCC, to translate the C++ code into a language that the computer will understand

C++ Install IDE

- We will use **Code::Blocks** in our tutorial, which we believe is a good place to start.
- You can find the latest version of Codeblocks at http://www.codeblocks.org/.
- Download the `mingw-setup.exe` file, which will install the text editor with a compiler.



Not secure | codeblocks.org/downloads/binaries/#imagesoswindows48pnglogo-microsoft-windows

Microsoft Windows

File	Download from
codeblocks-20.03-setup.exe	FossHUB or Sourceforge.net
codeblocks-20.03-setup-nonadmin.exe	FossHUB or Sourceforge.net
codeblocks-20.03-nosetup.zip	FossHUB or Sourceforge.net
codeblocks-20.03mingw-setup.exe	FossHUB or Sourceforge.net
codeblocks-20.03mingw-nosetup.zip	FossHUB or Sourceforge.net
codeblocks-20.03-32bit-setup.exe	FossHUB or Sourceforge.net
codeblocks-20.03-32bit-setup-nonadmin.exe	FossHUB or Sourceforge.net
codeblocks-20.03-32bit-nosetup.zip	FossHUB or Sourceforge.net
codeblocks-20.03mingw-32bit-setup.exe	FossHUB or Sourceforge.net
codeblocks-20.03mingw-32bit-nosetup.zip	FossHUB or Sourceforge.net

NOTE: The codeblocks-20.03-setup.exe file includes Code::Blocks with all plugins. The codeblocks-20.03-setup-nonadmin.exe file is provided for convenience to users that do not have administrator rights on their machine(s).

NOTE: The codeblocks-20.03mingw-setup.exe file includes additionally the GCC/G++/GFortran compiler and GDB debugger from MinGW-W64 project (version 8.1.0, 32/64 bit, SEH).

NOTE: The codeblocks-20.03(mingw)-nosetup.zip files are provided for convenience to users that are allergic against installers. However, it will not allow to select plugins / features to install (it includes everything) and not create any menu shortcuts. For the "installation" you are on your own.

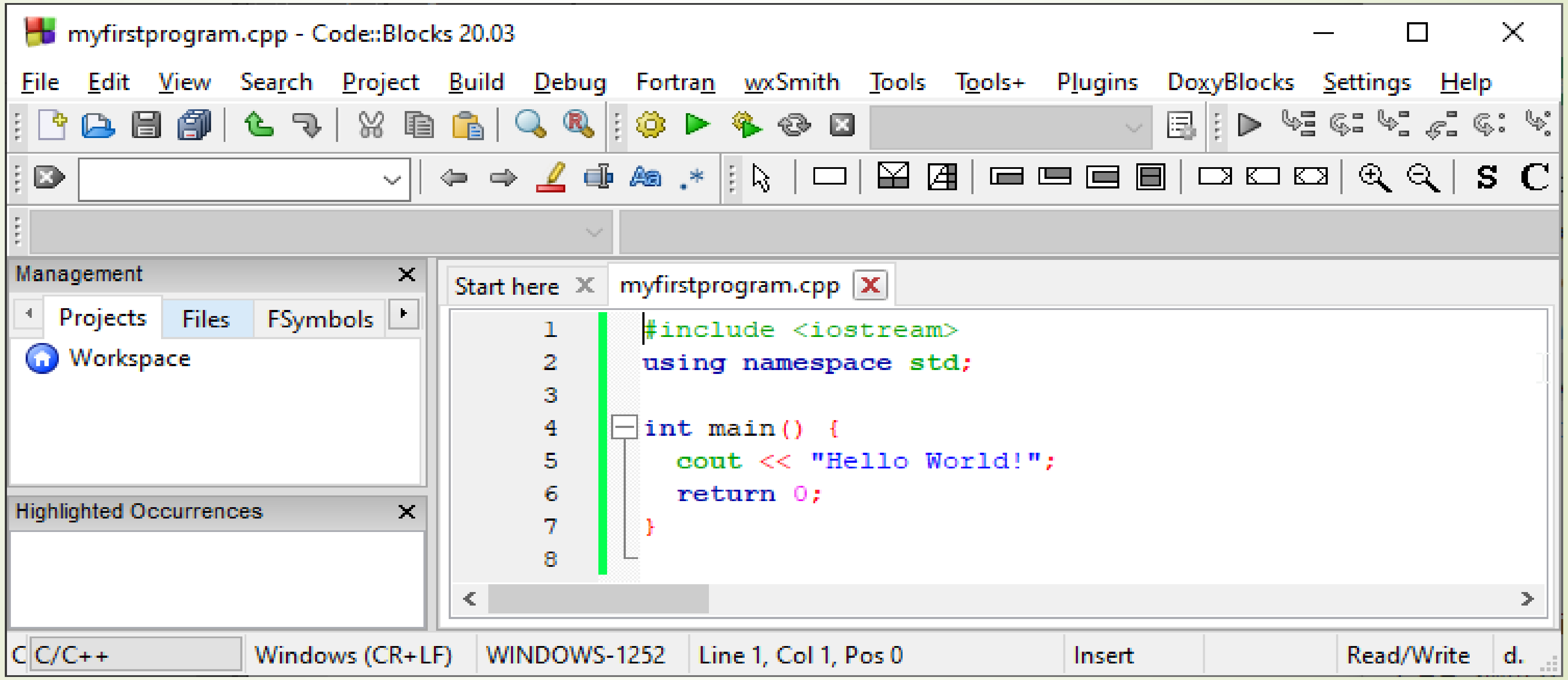
If unsure, please use codeblocks-20.03mingw-setup.exe!

Linux 32 and 64-bit

C++ Quickstart

- Open Code::Blocks and go to **File** > **New** > **Empty File**.
- Write the following C++ code and save the file as myfirstprogram.cpp (**File** > **Save File as**):

Your First Program


myfirstprogram.cpp - Code::Blocks 20.03
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
Management
Projects Files FSymbols
Workspace
Highlighted Occurrences
Start here
myfirstprogram.cpp
C/C++ Windows (CR+LF) WINDOWS-1252 Line 1, Col 1, Pos 0 Insert Read/Write d.

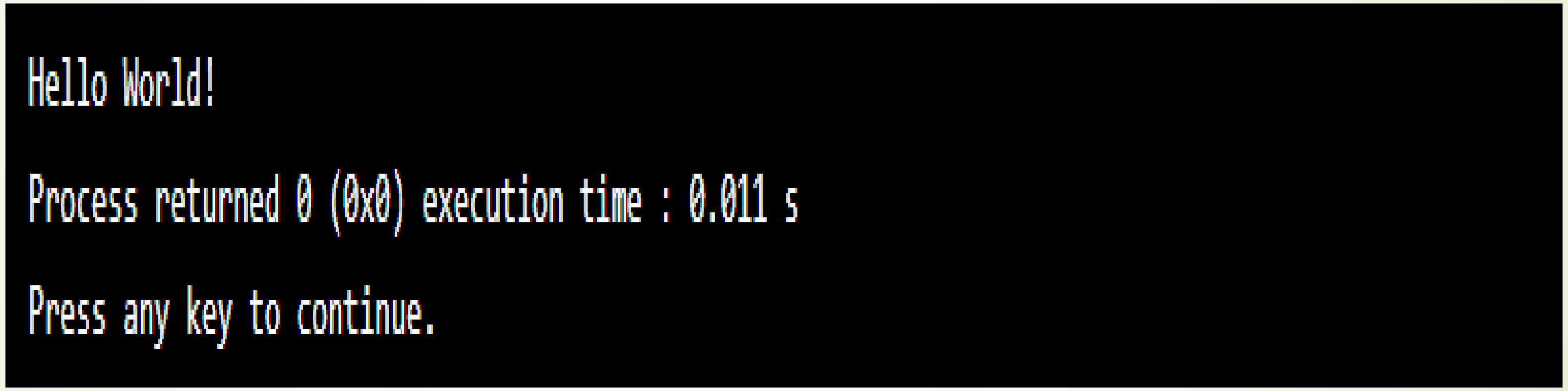
```cpp
#include <iostream>
using namespace std;

int main() {
  cout << "Hello World!";
  return 0;
}

```

Your First Program

- Then, go to **Build** > **Build and Run** to run (execute) the program. The result will look something to this:



```
Hello World!

Process returned 0 (0x0) execution time : 0.011 s

Press any key to continue.
```

- **Congratulations**! You have now written and executed your first C++ program.

C++ Syntax

1. Line 1: #include <iostream> is a **header file library** that lets us work with input and output objects, such as cout (used in line 5). Header files add functionality to C++ programs.

2. Line 2: using namespace std means that we can use names for objects and variables from the standard library.
3. Line 3: A blank line. C++ ignores white space. But we use it to make the code more readable.
4. Line 4: Another thing that always appear in a C++ program, is int main(). This is called a function. Any code inside its curly brackets { } will be executed.

C++ Syntax

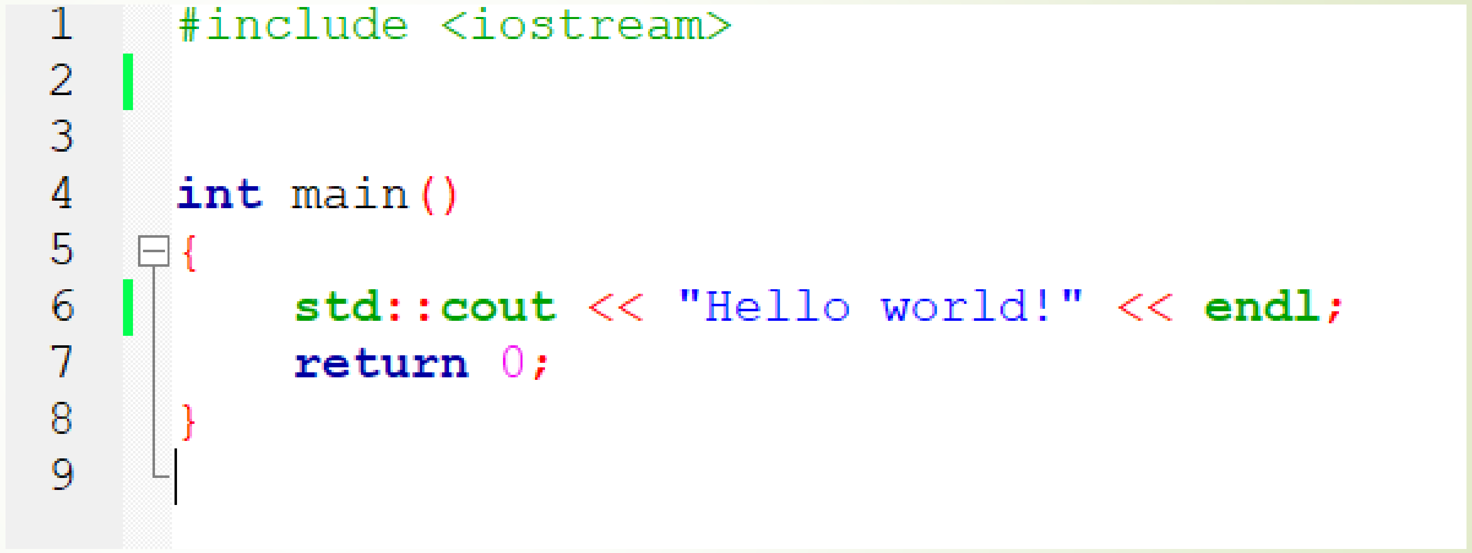
5. Line 5: cout (pronounced "see-out") is an object used together with the insertion operator (<<) to output/print text. In our example it will output "Hello World".

- Note: Every C++ statement ends with a semicolon ;
- Note: The body of int main() could also been written as:
- int main () { cout << "Hello World! "; return 0; }
- Remember: The compiler ignores white spaces. However, multiple lines makes the code more readable.

5. Line 6: return 0 ends the main function.
6. Line 7: Do not forget to add the closing curly bracket } to actually end the main function.

Omitting Namespace

- You might see some C++ programs that runs without the standard namespace library. The `using namespace std` line can be omitted and replaced with the std keyword , followed by the :: operator for some objects:



```
#include <iostream>


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

```

C++ Output (Print Text)

- The cout object, together with the << operator, is used to output values/print text:

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

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

C++ Output (Print Text)

- You can add as many cout objects as you want. However, note that it does not insert a new line at the end of the output:

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

  int main() {
    cout << "Hello World!";
    cout << "I am learning C++";
    return 0;
  }
  ```

C++ New Lines

- To insert a new line, you can use the \n character:

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

  int main() {
    cout << "Hello World! \n";
    cout << "I am learning C++";
    return 0;
  }
  ```

C++ New Lines

- Two \n characters after each other will create a blank line:

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

  int main() {
    cout << "Hello World! \n\n";
    cout << "I am learning C++";
    return 0;
  }
  ```

C++ New Lines

- Another way to insert a new line, is with the endl manipulator:

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

  int main() {
    cout << "Hello World!"<<endl;
    cout << "I am learning C++";
    return 0;
  }
  ```

escape sequence

- The newline character (\n) is called an **escape sequence**, and it forces the cursor to change its position to the beginning of the next line on the screen. This results in a new line.
- Examples of other valid escape sequences are

escape sequence	Description
\t	Creates a horizontal tab
\\	Inserts a backslash character (\)
\”	Inserts a double quote character

C++ Comments

- Single-line Comments
- Single-line comments start with two forward slashes (//).

-
  ```
  // This is a comment
  cout << "Hello World!";
  ```
- C++ Multi-line Comments
- Multi-line comments start with /* and ends with */

-
  ```
  /* The code below will print the words Hello World!
  to the screen, and it is amazing */
  cout << "Hello World!";
  ```

C++ Variables

- Variables are containers for storing data values. In C++, there are different **types** of variables (defined with different keywords), for example:

type	description
Int	stores integers (whole numbers), without decimals
Float	stores floating point numbers, with decimals
Double	stores Double precision floating point number
Char	stores single characters, such as 'a' or 'B'. Char values are surrounded by single quotes
String	stores text, such as "Hello World". String values are surrounded by double quotes
bool	stores values with two states: true or false

Declaring (Creating) Variables

- To create a variable, specify the type and assign it a value:
- Syntax:
- type variableName = value;

- Where type is one of C++ types (such as int), and variableName is the name of the variable (such as **x** or **myName**). The **equal sign** is used to assign values to the variable.

- Create a variable called **myNum** of type int and assign it the value **15**:

```
int myNum = 15;
cout << myNum;
```

Declaring (Creating) Variables

- You can also declare a variable without assigning the value, and assign the value later:

```
int myNum;
myNum = 15;
cout << myNum;
```

- Note that if you assign a new value to an existing variable, it will overwrite the previous value:

```
int myNum = 15;  // myNum is 15
myNum = 10;  // Now myNum is 10
cout << myNum;  // Outputs 10
```

Data types

```
int myNum = 5;                 // Integer (whole number without decimals)
double myFloatNum = 5.99;      // Floating point number (with decimals)
char myLetter = 'D';           // Character
string myText = "Hello";       // String (text)
bool myBoolean = true;         // Boolean (true or false)
```

Display Variables

- The cout object is used together with the << operator to display variables.
- To combine both text and a variable, separate them with the << operator:

```
int myAge = 35;
cout << "I am " << myAge << " years old.";
```

- Add Variables Together
- To add a variable to another variable, you can use the + operator:

```
int x = 5;
int y = 6;
int sum = x + y;
cout << sum;
```

Declare Many Variables

- To declare more than one variable of the **same type**, use a comma-separated list:

-
  ```
  int x = 5, y = 6, z = 50;
  cout << x + y + z;
  ```

C++ Identifiers

- All C++ variables must be identified with unique names.
- These unique names are called identifiers.
- Identifiers can be short names (like x and y) or more descriptive names (age, sum, totalVolume).
- Note: It is recommended to use descriptive names in order to create understandable and maintainable code:

```
// Good
int minutesPerHour = 60;

// OK, but not so easy to understand what m actually is
int m = 60;
```

C++ Identifiers

- The general rules for naming variables are:
- Names can contain letters, digits and underscores.
- Names must begin with a letter or an underscore (_).
- Names are case sensitive (myVar and myvar are different variables).
- Names cannot contain whitespaces or special characters like !, #, %, etc.
- Reserved words (like C++ keywords, such as int) cannot be used as names.

Constants

- When you do not want others (or yourself) to override existing variable values, use the const keyword (this will declare the variable as "constant", which means **unchangeable and read-only**):

```
const int myNum = 15;  // myNum will always be 15
 myNum = 10;  // error: assignment of read-only variable 'myNum'
```

- You should always declare the variable as constant when you have values that are unlikely to change:

```
const int minutesPerHour = 60;
const float PI = 3.14;
```

C++ User Input

- You have already learned that cout is used to output (print) values. Now we will use cin to get user input.
- cin is a predefined variable that reads data from the keyboard with the extraction operator (>>).
- In the following example, the user can input a number, which is stored in the variable x hen we print the value of x:

-
  ```cpp
  int x;
  cout << "Type a number: "; // Type a number and press enter
  cin >> x; // Get user input from the keyboard
  cout << "Your number is: " << x; // Display the input value
  ```

Thanks