The background features a large, faint watermark of the Simon Fraser University (SFU) logo, which consists of a stylized tree with four main branches and the letters 'SFU' below it.

CIS 129

# Advanced Computer Programming

Chapter 1: An Overview of Computers and Programming Languages

Mr. Horence Chan

# Object-oriented language

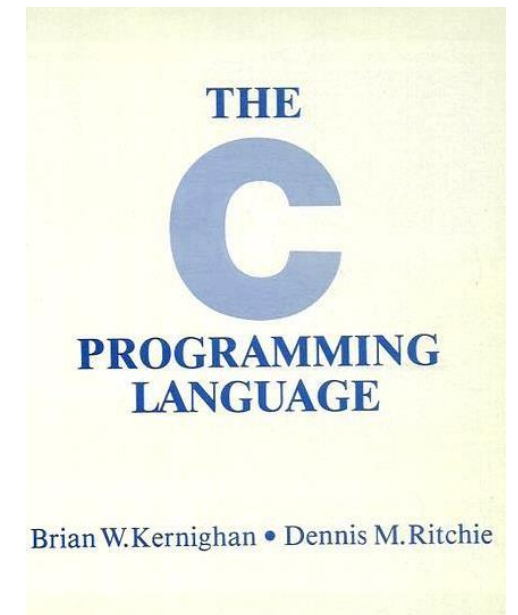
- Most popular programming language (Updated on Jan 2024)

Jan 2024	Jan 2023	Change	Programming Language	Ratings	Change
1	1		 Python	13.97%	-2.39%
2	2		 C	11.44%	-4.81%
3	3		 C++	9.96%	-2.95%
4	4		 Java	7.87%	-4.34%
5	5		 C#	7.16%	+1.43%
6	7	▲	 JavaScript	2.77%	-0.11%
7	10	▲	 PHP	1.79%	+0.40%
8	6	▼	 Visual Basic	1.60%	-3.04%
9	8	▼	 SQL	1.46%	-1.04%

<https://www.tiobe.com/tiobe-index/>

# What is C?

- C is a general purpose, procedural, imperative language developed in 1972 by Dennis Ritchie at Bell Labs for the Unix Operating System
  - Low-level access to memory
  - Language constructs close to machine instructions
  - Used as a “machine-independent assembler”



# My first C Program

A preprocessor directive

Include standard io declarations

```
#include <stdio.h>

int main(void)
{
    printf("hello, world\n");
    return 0;
}
```

Write to standard output

char array

Indicate correct termination

# What is C++?



Programming  
with python



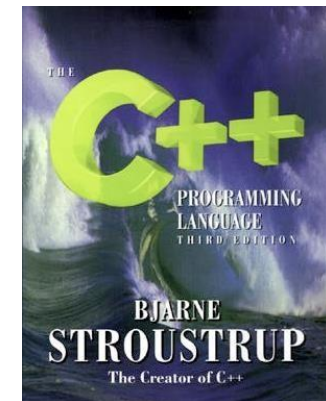
Programming  
with C++



Programming  
directly to  
binary

# What is C++?

- A “*better C*” that supports:
- Systems programming
- Object-oriented programming (classes & inheritance)
- Programming-in-the-large (namespaces, exceptions)
- Generic programming (templates)
- Reuse (large class & template libraries)



# C++ vs C

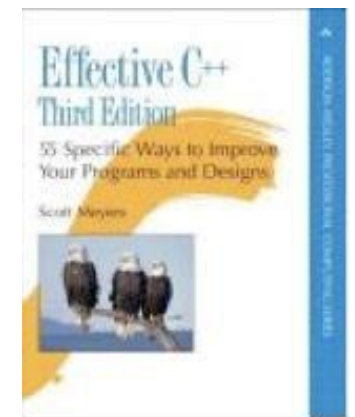
- Most C programs are also C++ programs.

*Nevertheless, good C++ programs usually do not resemble C:*

- avoid macros (use `inline`)
- avoid pointers (use references)
- avoid `malloc` and `free` (use `new` and `delete`)
- avoid arrays and `char*` (use vectors and strings) ...
- avoid structs (use classes)

*C++ encourages a different style of programming:*

- avoid procedural programming
  - **model your domain** with classes and templates



# “Hello World” in C++



```
print("Hello World")
```

Python



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

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

C++



# “Hello World” in C++

Use the standard namespace

Include standard  
iostream classes

A C++ comment

cout is an  
instance of  
ostream

```
#include <iostream>
using namespace std;
// My first C++ program!
int main(void)
{
    cout << "hello world!" << endl;
    return 0;
}
```

insertion operator

# “Hello World” in C++

**Adding another programming language to my resume after learning how to write Hello World in it.**



# Makefiles / Managed Make in CDT

You could compile it all together by hand:

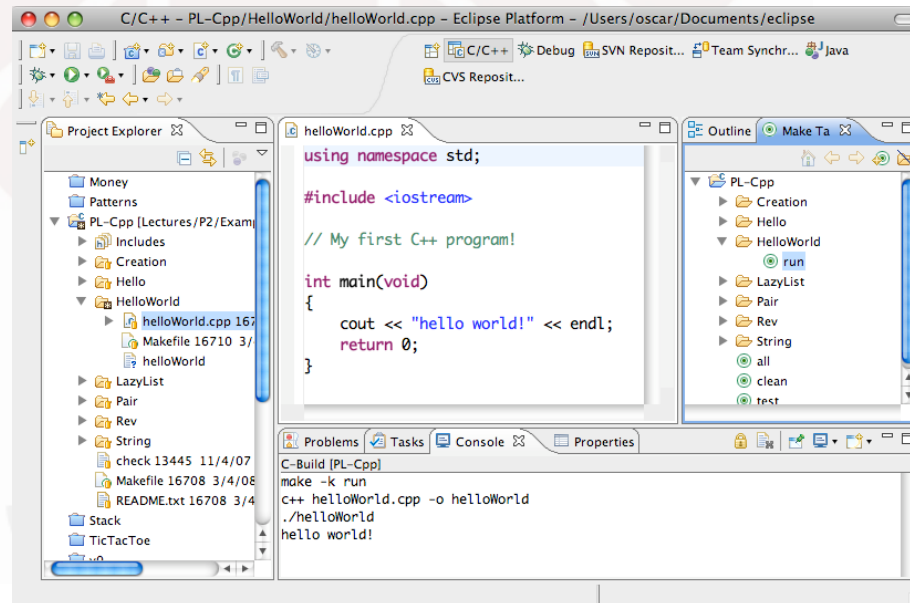
```
c++ helloWorld.cpp -o helloWorld
```

Or you could use a *Makefile* to manage dependencies:

```
helloWorld : helloWorld.cpp  
c++ $@.cpp -o $@
```

```
make helloWorld
```

Or you could use *cdt with eclipse* to create a standard managed make project



# C++ Design Goals

*“C with Classes” designed by Bjarne Stroustrup in early 1980s:*

- Originally a translator to C
  - Initially difficult to debug and inefficient
- Mostly *upward compatible* extension of C
  - “As close to C as possible, but no closer”
  - Stronger type-checking
  - Support for object-oriented programming
- Run-time efficiency
  - Language primitives close to machine instructions

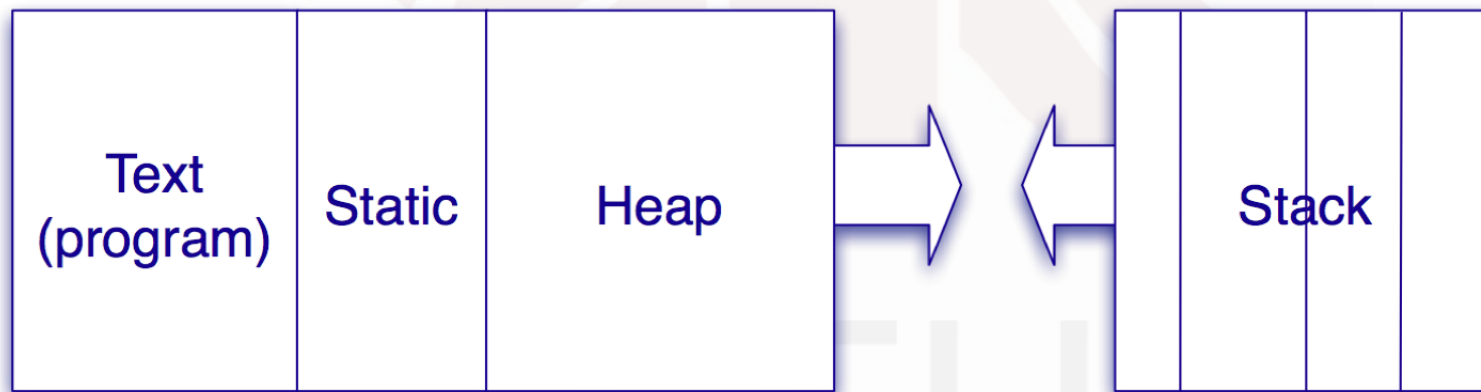
# C++ Features

<b><i>C with Classes</i></b>	Classes as structs Inheritance; virtual functions Inline functions
<b><i>C++ 1.0 (1985)</i></b>	Strong typing; function prototypes new and delete operators
<b><i>C++ 2.0</i></b>	Local classes; protected members Multiple inheritance
<b><i>C++ 3.0</i></b>	Templates Exception handling
<b><i>ANSI C++ (1998)</i></b>	Namespaces RTTI (Runtime Type Information)

# Memory Layout

*The address space consists of (at least):*

<b>Text:</b>	executable program text (not writable)
<b>Static:</b>	static data
<b>Heap:</b>	dynamically allocated global memory (grows upward)
<b>Stack:</b>	local memory for function calls (grows downward)



# C++ Classes

C++ classes may be instantiated either *automatically* (on the stack):

```
MyClass oVal;    // constructor called  
                // destroyed when scope ends
```

or *dynamically* (in the heap)

```
MyClass *oPtr;    // uninitialized pointer  
  
oPtr = new MyClass; // constructor called  
                  // must be explicitly deleted
```

# Pointers in C++

```
int i;  
int *iPtr; // a pointer to an integer  
  
iPtr = &i; // iPtr contains the address of I  
*iPtr = 100;
```

variable	value	Address in hex
i	100	456FD4
iPtr	456FD4	456FD0
	...	



# References

A reference is an **alias** for another variable:

```
int i = 10;
int &ir = i; // reference (alias)
ir = ir + 1; // increment i
```

i,ir

10
10
10

*Once initialized, references cannot be changed.*

References are especially useful in **procedure calls** to avoid the overhead of passing arguments by value, without the clutter of explicit pointer dereferencing (`y = *ptr;`)

```
void refInc(int &n)
{
    n = n+1; // increment the variable n refers to
}
```

# References vs Pointers

*References should be preferred to pointers*

***except when:***

- manipulating dynamically allocated objects
  - **new** returns an object pointer
- a variable must range over a set of objects
  - use a **pointer** to walk through the set

SFU

The background features a large, faint watermark of the SFU logo, which consists of a cross with stylized leaves in each quadrant and the letters 'SFU' at the bottom.

# Introduction of Visual Studio

# Visual Studio (For Windows user)

- <https://visualstudio.microsoft.com/>

## It's how you make software

What do you want to [code, build, debug, deploy, collaborate on, analyze, learn] today?

Visual Studio can do that.

## Meet the Visual Studio family



### Visual Studio

Windows | Version 17.0

The best comprehensive IDE for .NET and C++ developers on Windows. Fully packed with a sweet array of tools and features to elevate and enhance every stage of software development.

[Download Visual Studio](#) ▾



### Visual Studio for Mac

macOS | Version 8.10

A comprehensive IDE for .NET developers that's native to macOS. Includes top-notch support for web, cloud, and game development —plus ridiculously good tools for making cross-platform mobile apps.

Read more about [activating your license](#)

[Download Visual Studio for Mac](#)



### Visual Studio Code

Windows, macOS, Linux | Version 1.62

A standalone source code editor that runs on Windows, macOS, and Linux. The top pick for Java and web developers, with tons of extensions to support just about any programming language.

By using Visual Studio Code you agree to its [license](#) & [privacy statement](#)

[Download Visual Studio Code](#) ▾

# Visual Studio (For Windows user)

- During installation, please choose “Desktop development with C++”

Modifying — Visual Studio Community 2019 — 16.8.4

Workloads Individual components Language packs Installation locations

Web & Cloud (4)

- ASP.NET and web development  
Build web applications using ASP.NET Core, ASP.NET, HTML/JavaScript, and Containers including Docker support.
- Python development  
Editing, debugging, interactive development and source control for Python.
- Azure development  
Azure SDKs, tools, and projects for developing cloud apps and creating resources using .NET Core and .NET
- Node.js development  
Build scalable network applications using Node.js, an asynchronous event-driven JavaScript runtime.

Desktop & Mobile (5)

- .NET desktop development  
Build WPF, Windows Forms, and console applications using C#, Visual Basic, and F# with .NET Core and .NET
- Desktop development with C++**   
Build modern C++ apps for Windows using tools of your choice, including MSVC, Clang, CMake, or MSBuild.
- Universal Windows Platform development  
Create applications for the Universal Windows Platform with C#, VB, or optionally C++.
- Mobile development with .NET  
Build cross-platform applications for iOS, Android or Windows using Xamarin.

Installation details

> Visual Studio core editor

✓ Desktop development with C++ \*  
Included

- ✓ C++ core desktop features
- ✓ IntelliCode

Optional

- ✓ MSVC v142 - VS 2019 C++ x64/x86 build tools (...)
- ✓ Windows 10 SDK (10.0.18362.0)
- ✓ Just-In-Time debugger
- ✓ C++ profiling tools
- ✓ C++ CMake tools for Windows
- ✓ C++ ATL for latest v142 build tools (x86 & x64)
- ✓ Test Adapter for Boost.Test
- ✓ Test Adapter for Google Test
- ✓ Live Share
- ✓ C++ AddressSanitizer (Experimental)
- C++ MFC for latest v142 build tools (x86 & x64)
- C++/CLI support for v142 build tools (14.28)
- C++ Modules for v142 build tools (x64/x86 – ex...
- C++ Clang tools for Windows (10.0.0 - x64/x86)

Location  
C:\Program Files (x86)\Microsoft Visual Studio\2019\Community

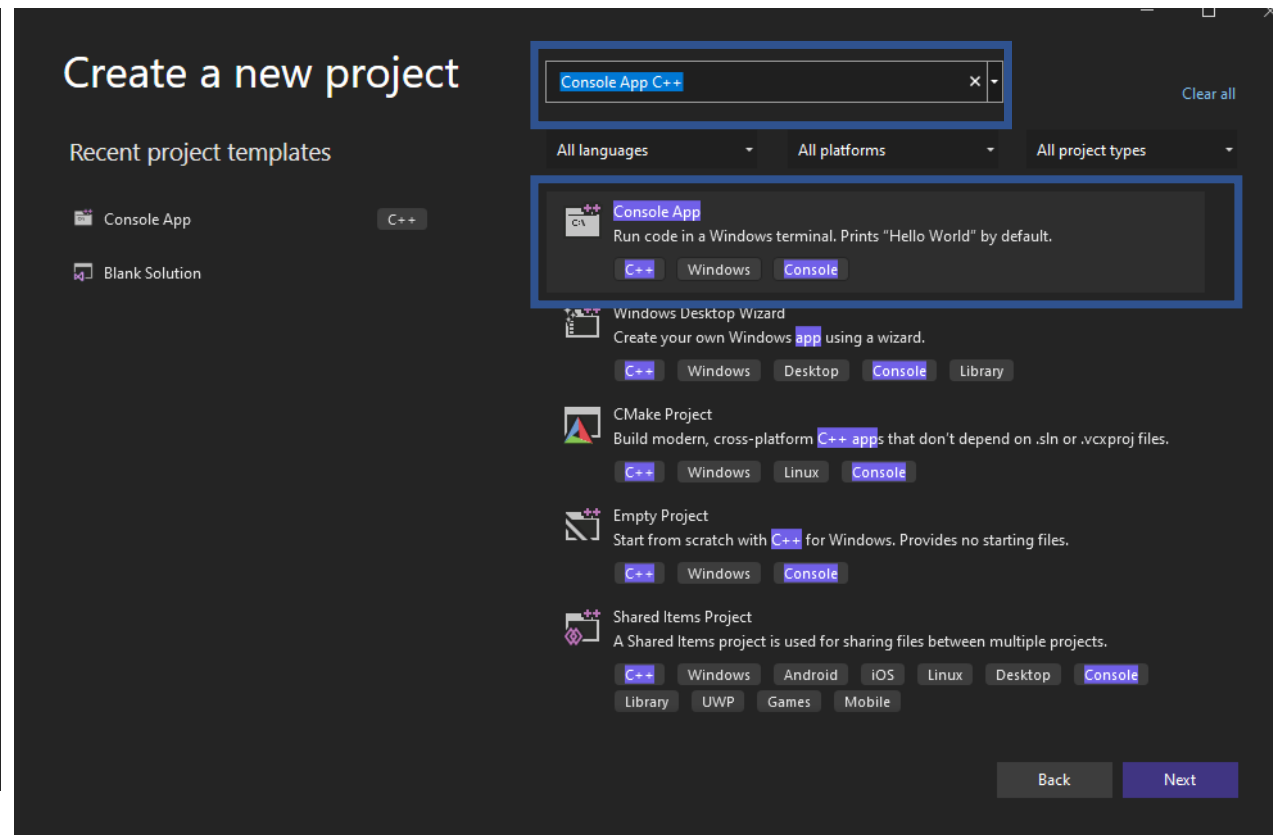
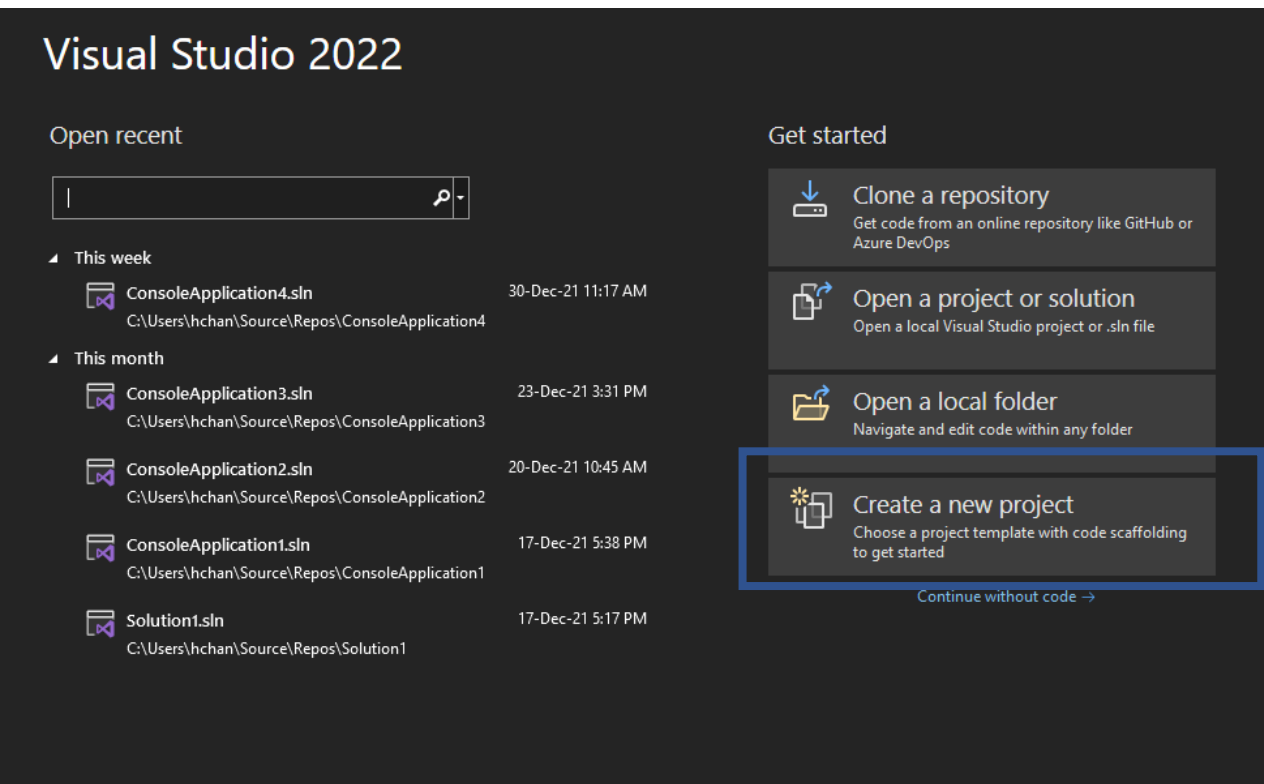
Total space required 3.96 GB

By continuing, you agree to the [license](#) for the Visual Studio edition you selected. We also offer the ability to download other software with Visual Studio. This software is licensed separately, as set out in the [3rd Party Notices](#) or in its accompanying license. By continuing, you also agree to those licenses.

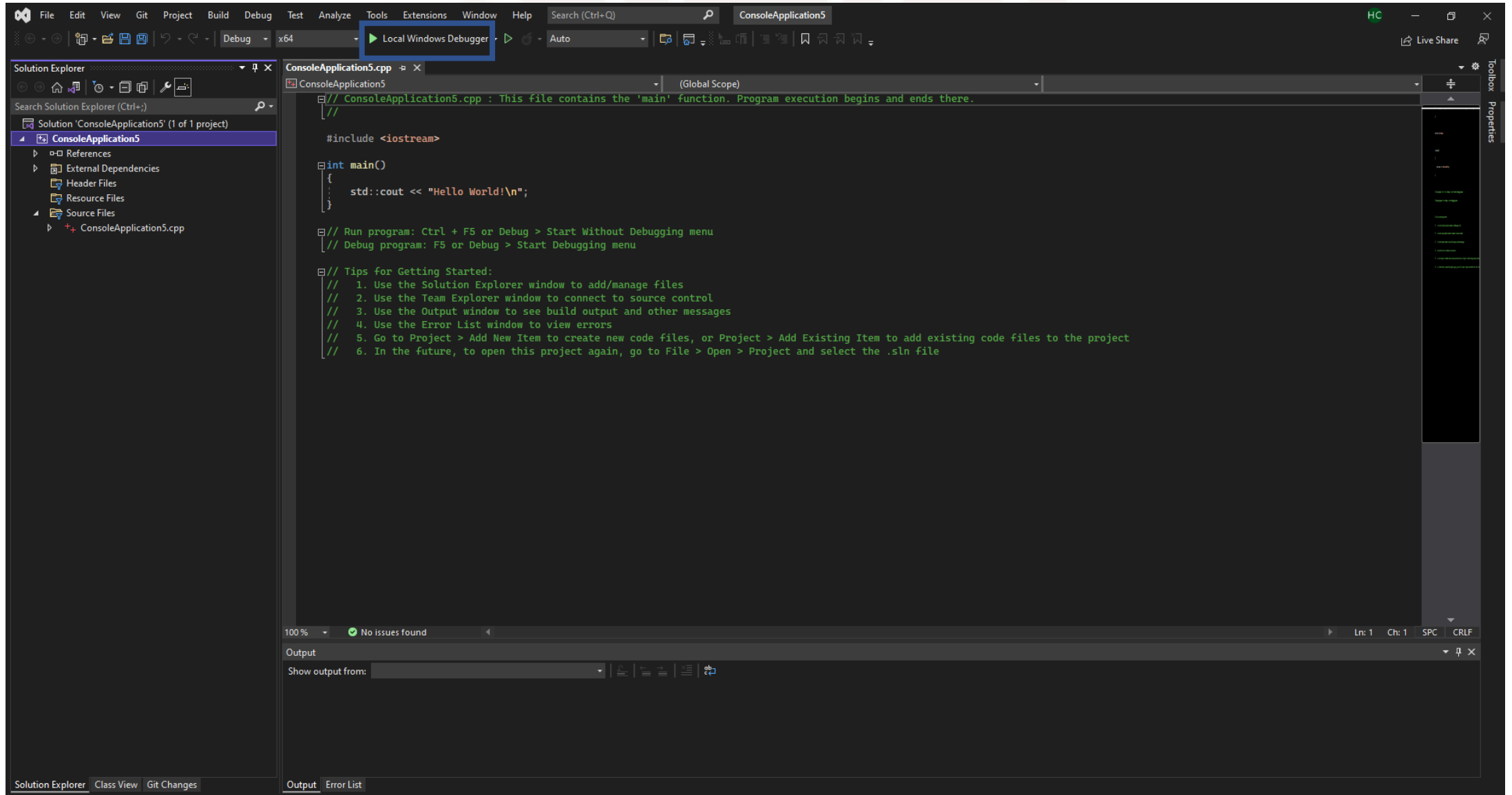
Install while downloading ▾ Modify

# Create your first C++ program

- Select “Create a new project”
- Type “Console App C++” in the search bar
- Select “Console App”



# Try to run the default “Hello World” program



# “Hello World” program output

```
Microsoft Visual Studio Debug Console
Hello World!
C:\Users\hchan\Source\Repos\ConsoleApplication5\x64\Debug\ConsoleApplication5.exe (process 7652) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```

when you didn't code in a language  
for too long and successfully write  
a hello world program





# Online GDB (For Windows / Mac user)

- [https://www.onlinegdb.com/online\\_c++\\_compiler#](https://www.onlinegdb.com/online_c++_compiler#)



The screenshot displays the Online GDB web interface. At the top, there is a toolbar with buttons for Run, Debug, Stop, Share, Save, Beautify, and a download icon. The language is set to C++. The main editor shows a C++ program in a file named main.cpp. The code includes a header comment, a preprocessor directive for iostream, and a main function that prints "Hello World" and returns 0. Below the editor, the output console shows the result of the program execution: "Hello World" followed by "...Program finished with exit code 0" and a prompt to press ENTER to exit the console.

```
1 | *****  
2 |  
3 | | | | | Online C++ Compiler.  
4 | | | | | Code, Compile, Run and Debug C++ program online.  
5 | Write your code in this editor and press "Run" button to compile and execute it.  
6 |  
7 | *****/  
8 |  
9 | #include <iostream>  
10 |  
11 | using namespace std;  
12 |  
13 | int main()  
14 | {  
15 |     cout<<"Hello World";  
16 |  
17 |     return 0;  
18 | }  
19 |
```

input  
Hello World  
...Program finished with exit code 0  
Press ENTER to exit console. □

# Create your first C++ program

- Try to create a simple adder by yourself
- Example output:

Please input two numbers: **2 3**

The solution of  $2 + 3$  is 5

SFU