CIS 129
Advanced Computer Programming

Chapter 1: An Overview of Computers and Programming Languages

Mr. Horence Chan

Object-oriented language

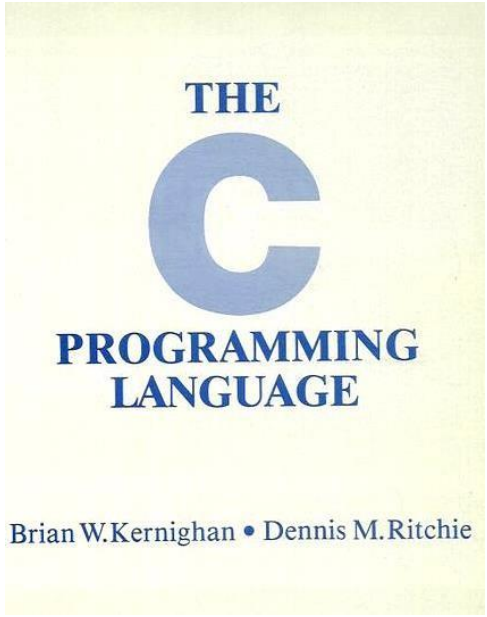
- Most popular programming language (Updated on Jan 2024)

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	v	Visual Basic	1.60%	-3.04%
9	8	v	SQL	1.46%	-1.04%

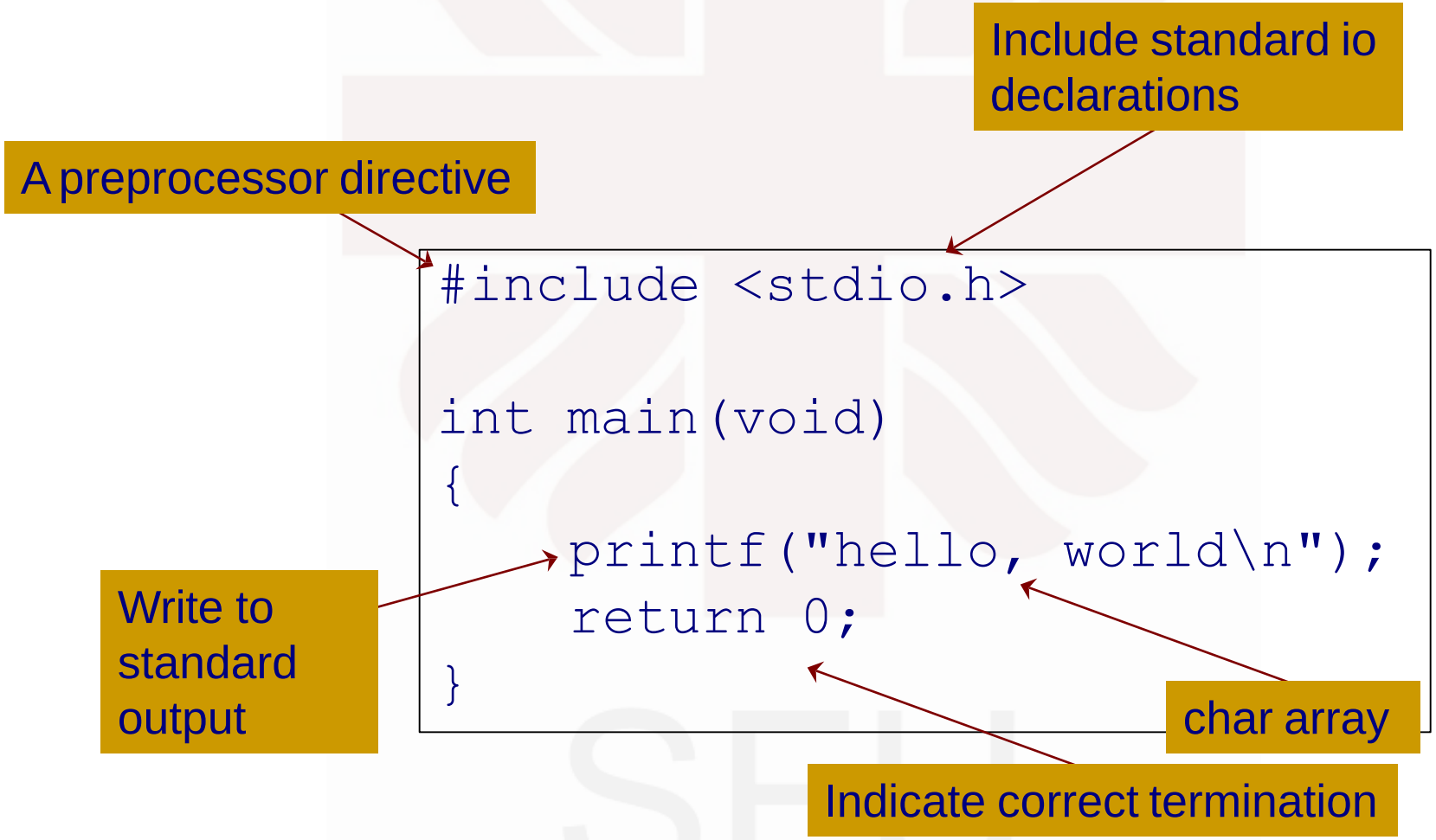
https://www.tiobe.com/tiobe-index/

What is C?

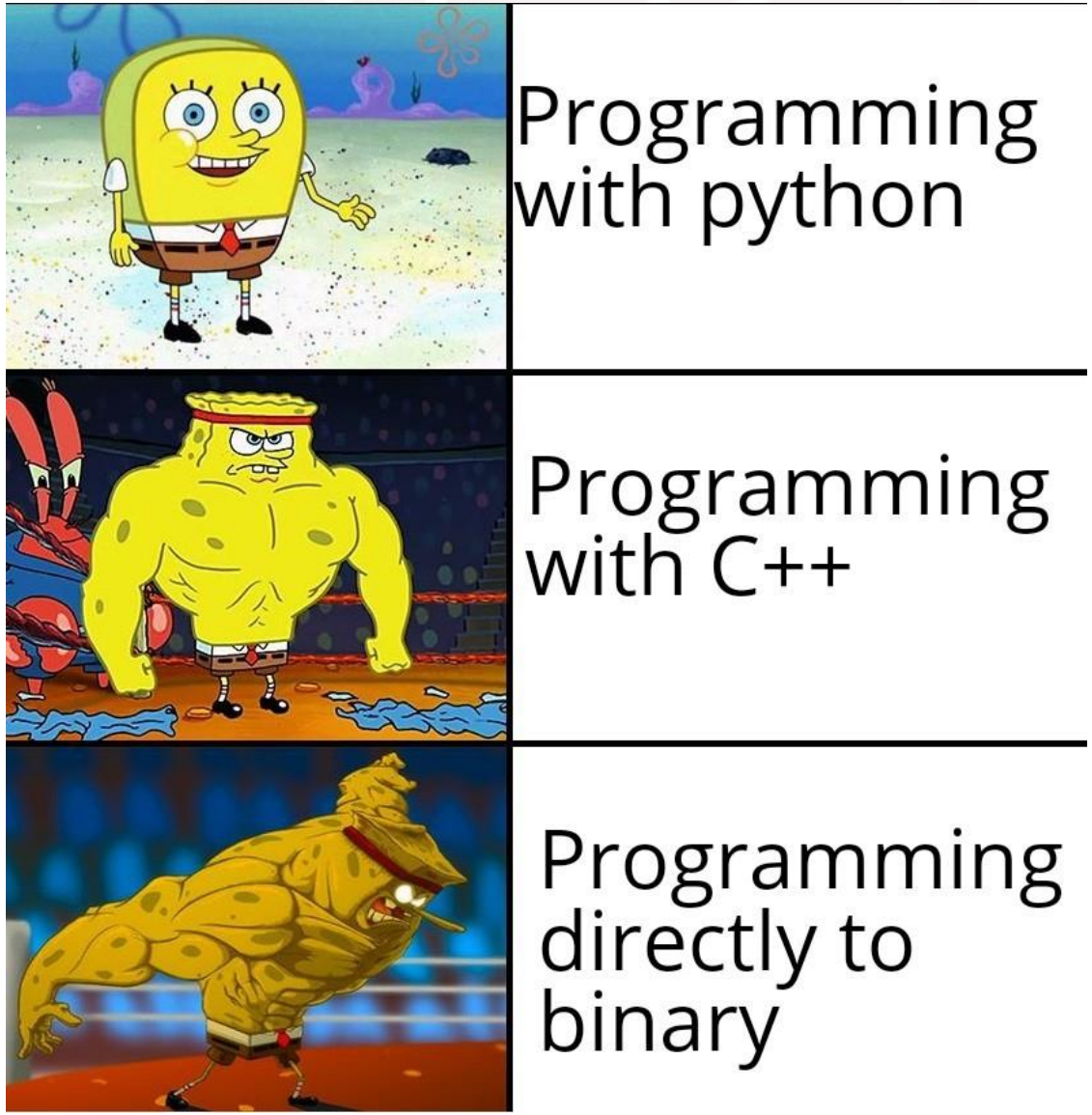
- C is a general purpose, procedural, imperative language developed in 1972 by Dennis Ritchie at Bell Labs for the Unix Operating System
  - Low-level access to memory
  - Language constructs close to machine instructions
  - Used as a “machine-independent assembler”



My first C Program

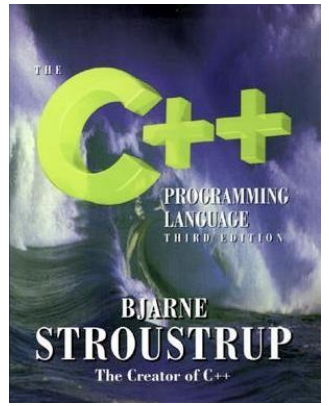


What is C++?


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

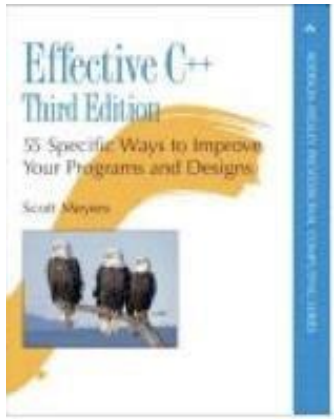
- Most C programs are also C++ programs.

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

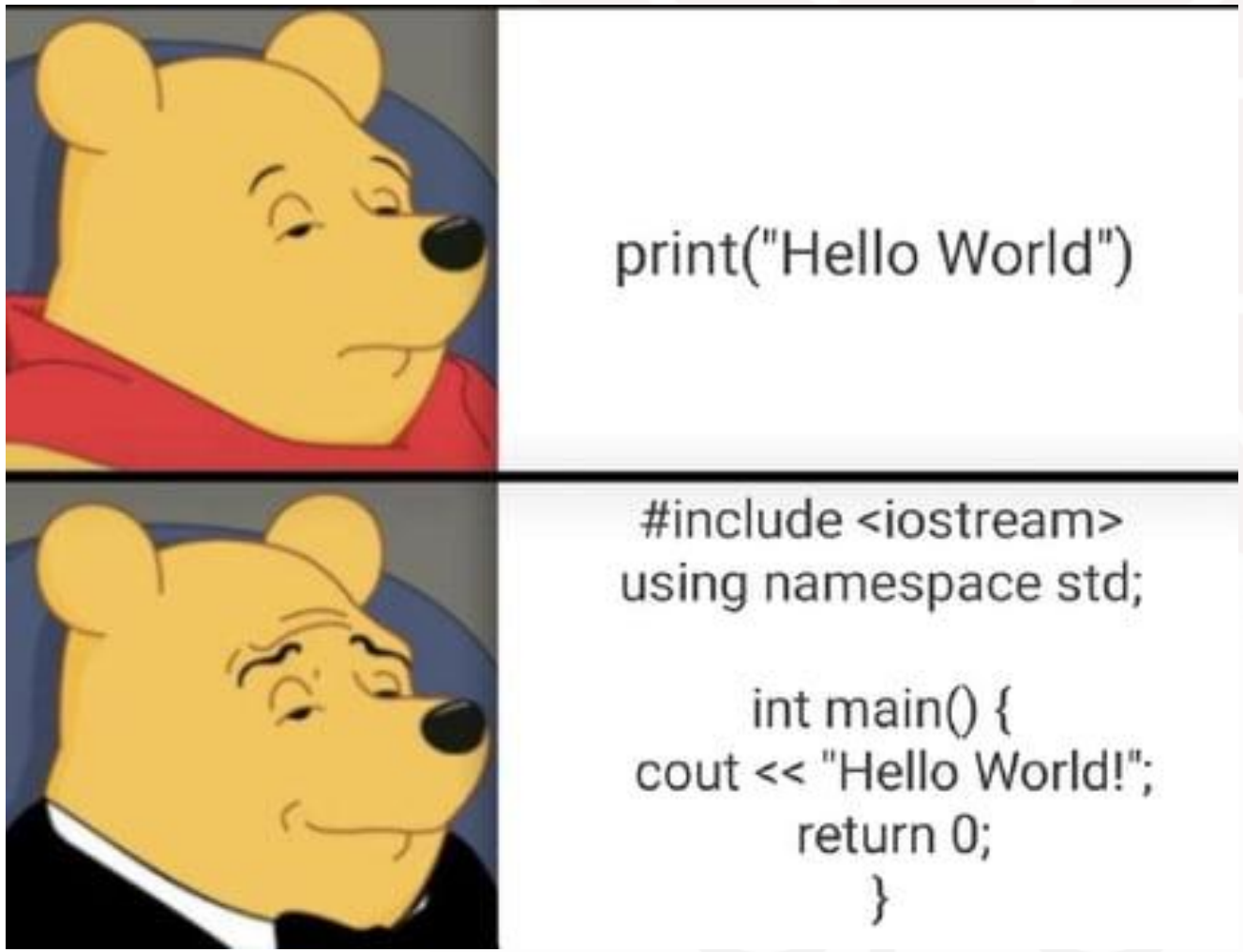
- avoid macros (use `inline`)
- avoid pointers (use references)
- avoid `malloc` and `free` (use `new` and `delete`)
- avoid arrays and `char*` (use `vectors` and `strings`) …
- avoid `structs` (use classes)

C++ encourages a different style of programming:

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



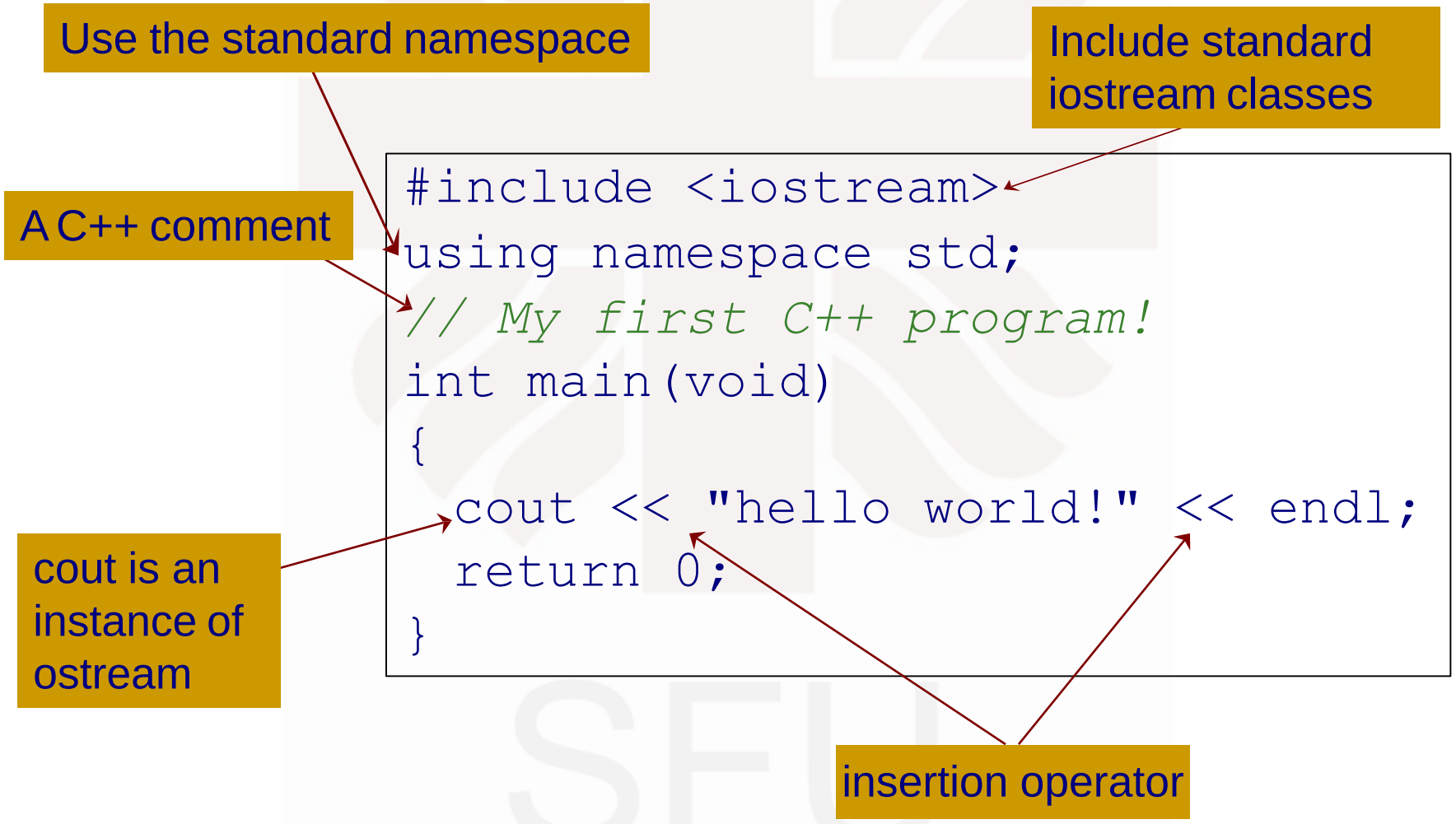
“Hello World” in C++



Python

C++

“Hello World” in C++



“Hello World” in C++

Makefiles / Managed Make in CDT

You could compile it all together by hand:

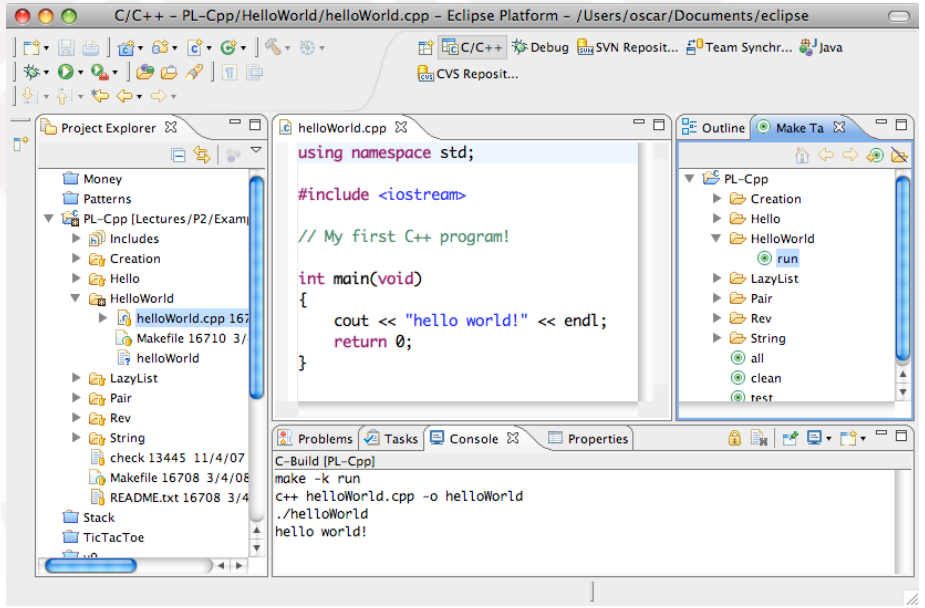
```
c++ helloWorld.cpp -o helloWorld
```

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

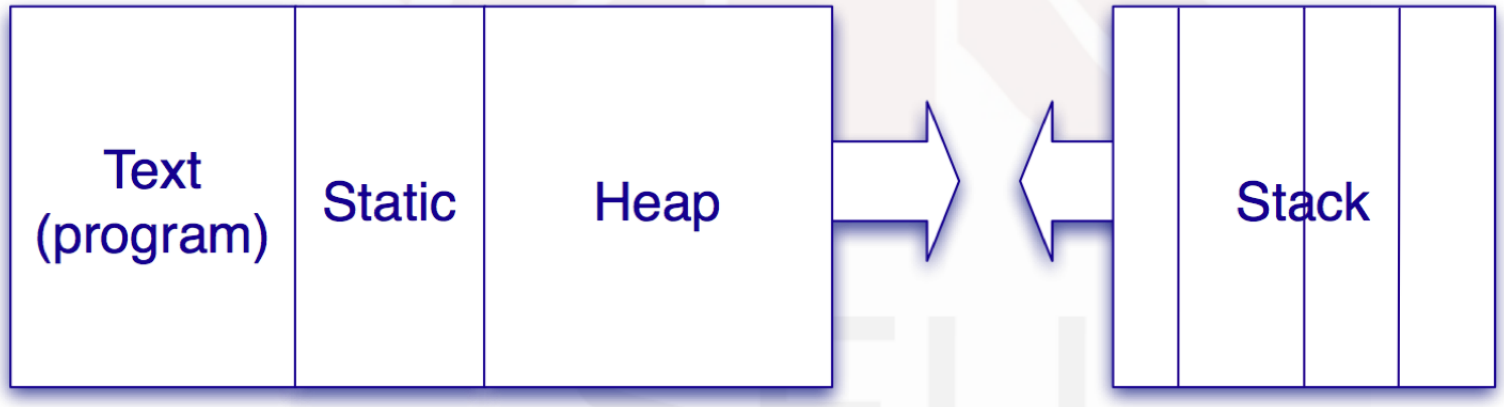
C++ Features

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

Memory Layout

The address space consists of (at least):

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



C++ Classes

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

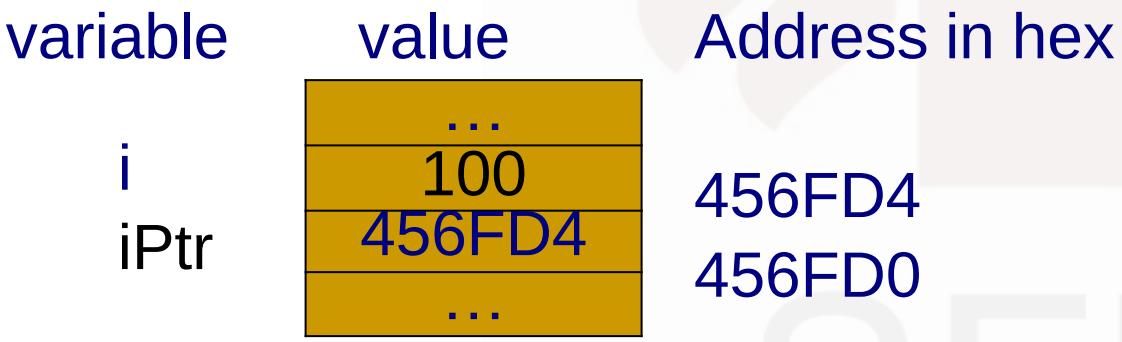
```
MyClass oVal;     // constructor called
                   // destroyed when scope ends
```

or *dynamically* (in the heap)

```
MyClass *oPtr;          // uninitialized pointer

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

Pointers in C++

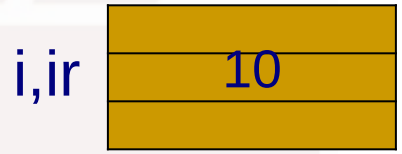
```
int i;
int *iPtr; // a pointer to an integer

iPtr = &i; // iPtr contains the address of I
*iPtr = 100;
```



References

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

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



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

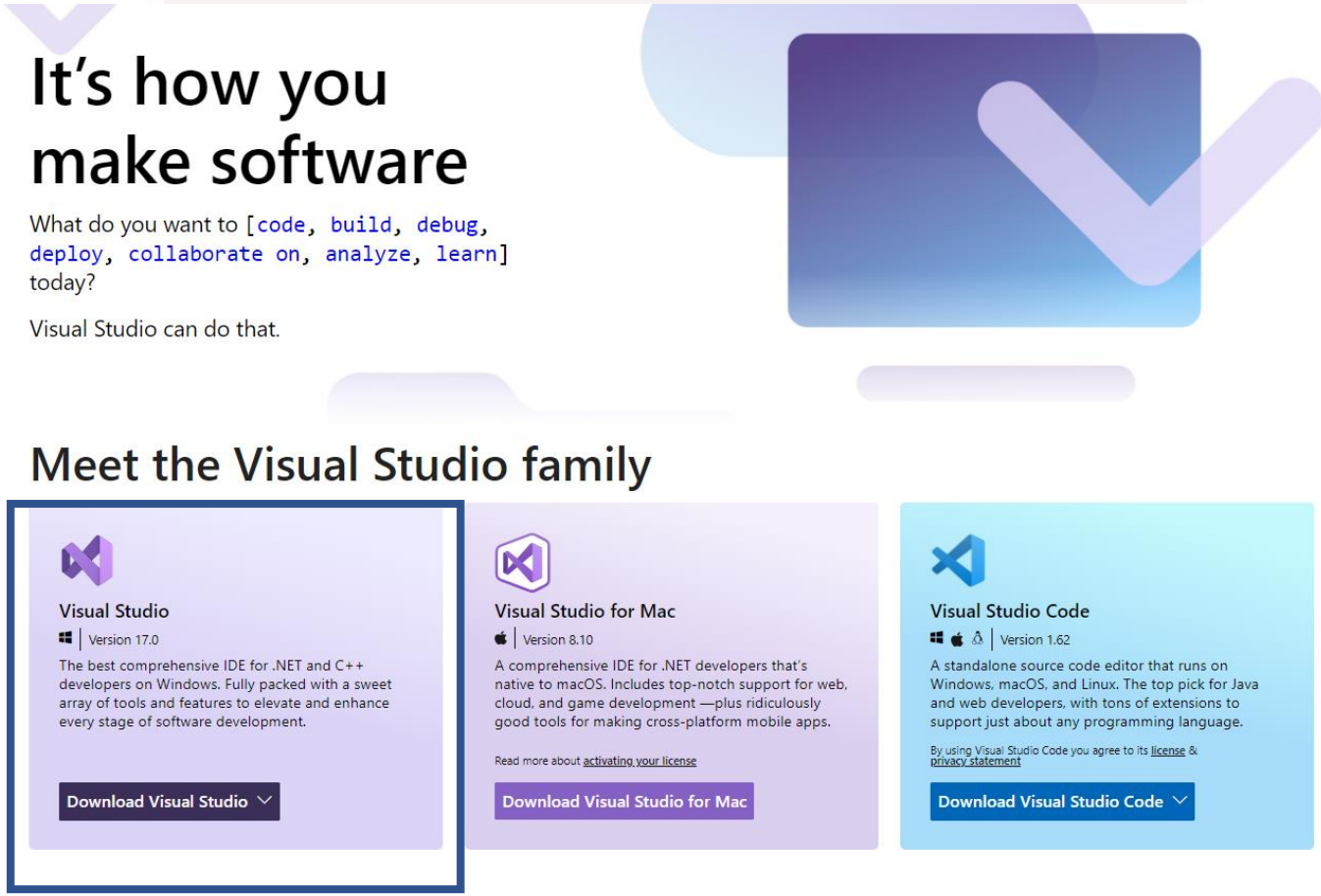
***except** when:*

- 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

Introduction of Visual Studio

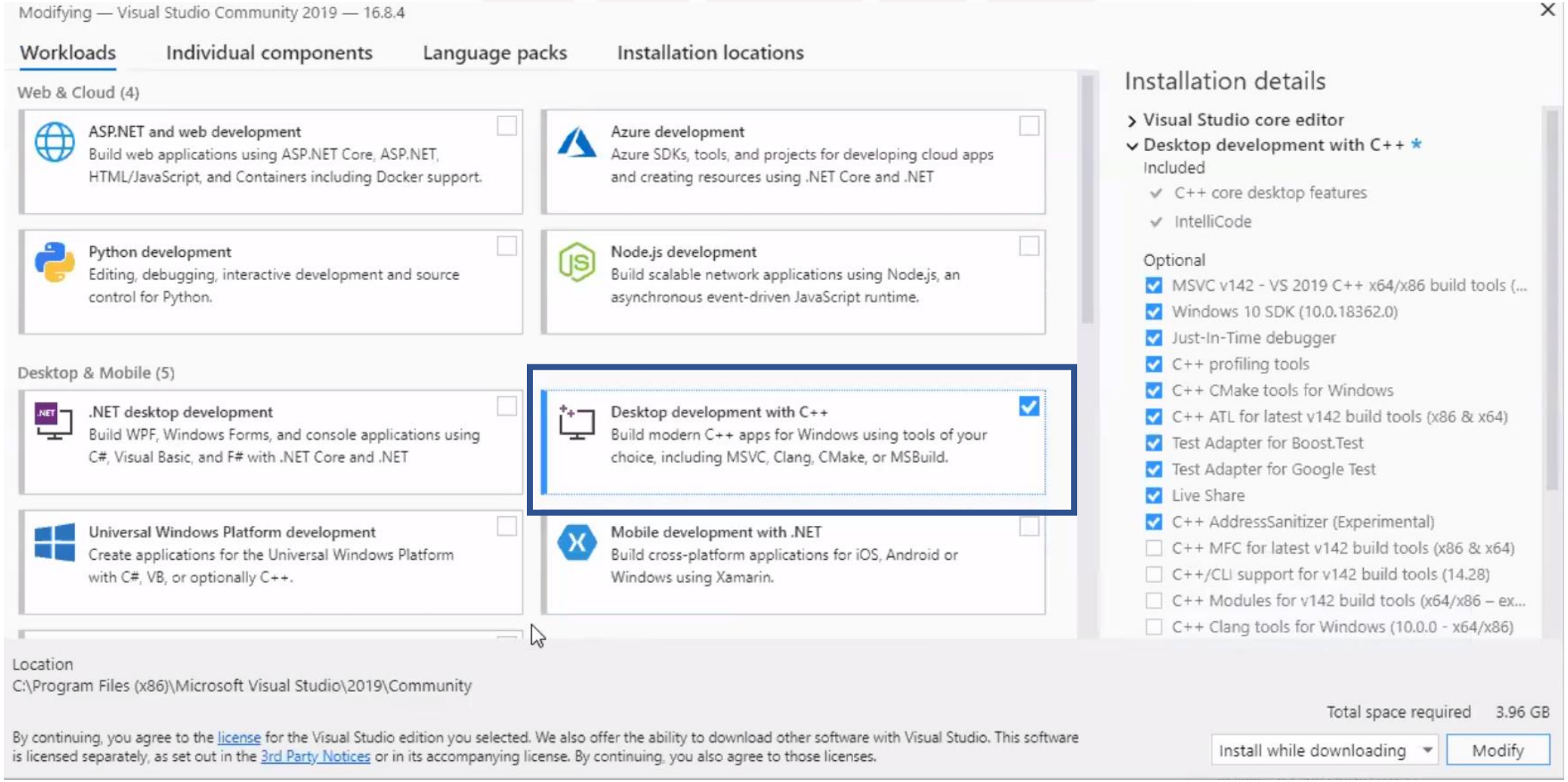
Visual Studio (For Windows user)

- https://visualstudio.microsoft.com/



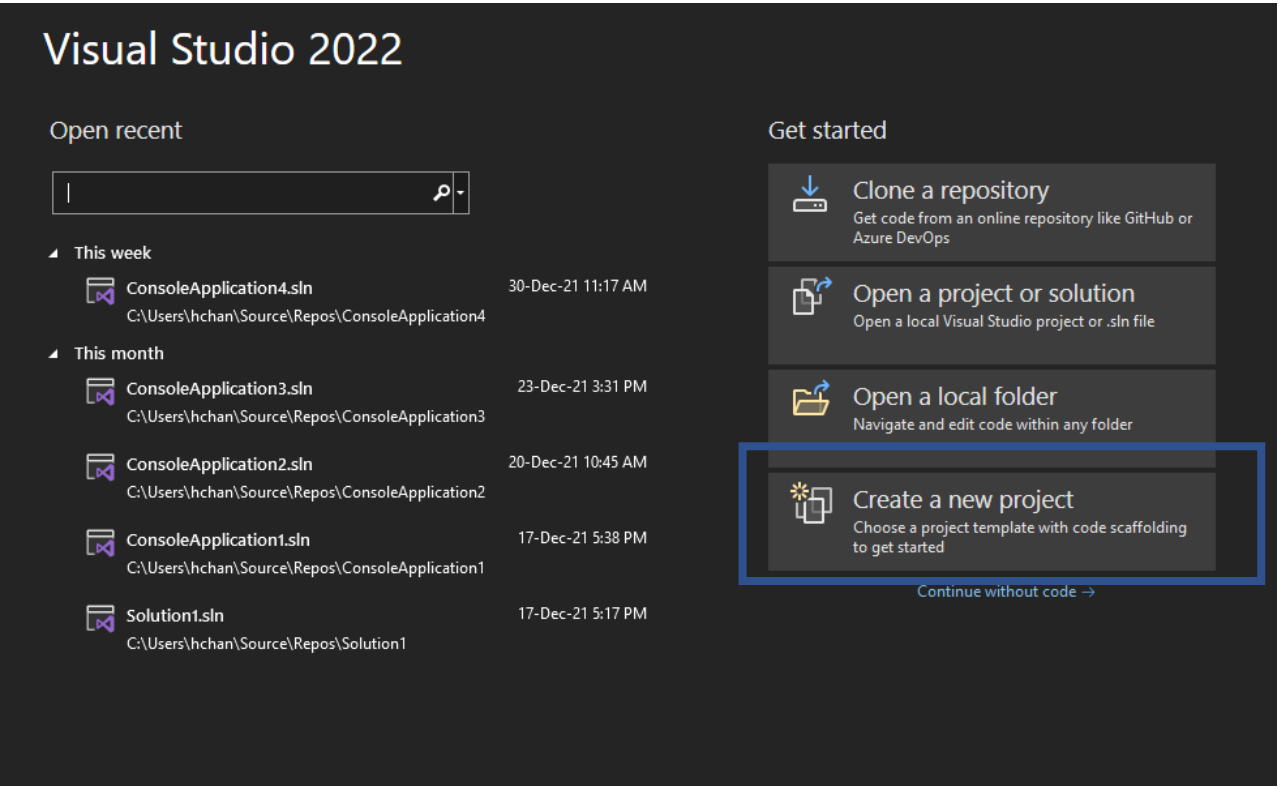
Visual Studio (For Windows user)

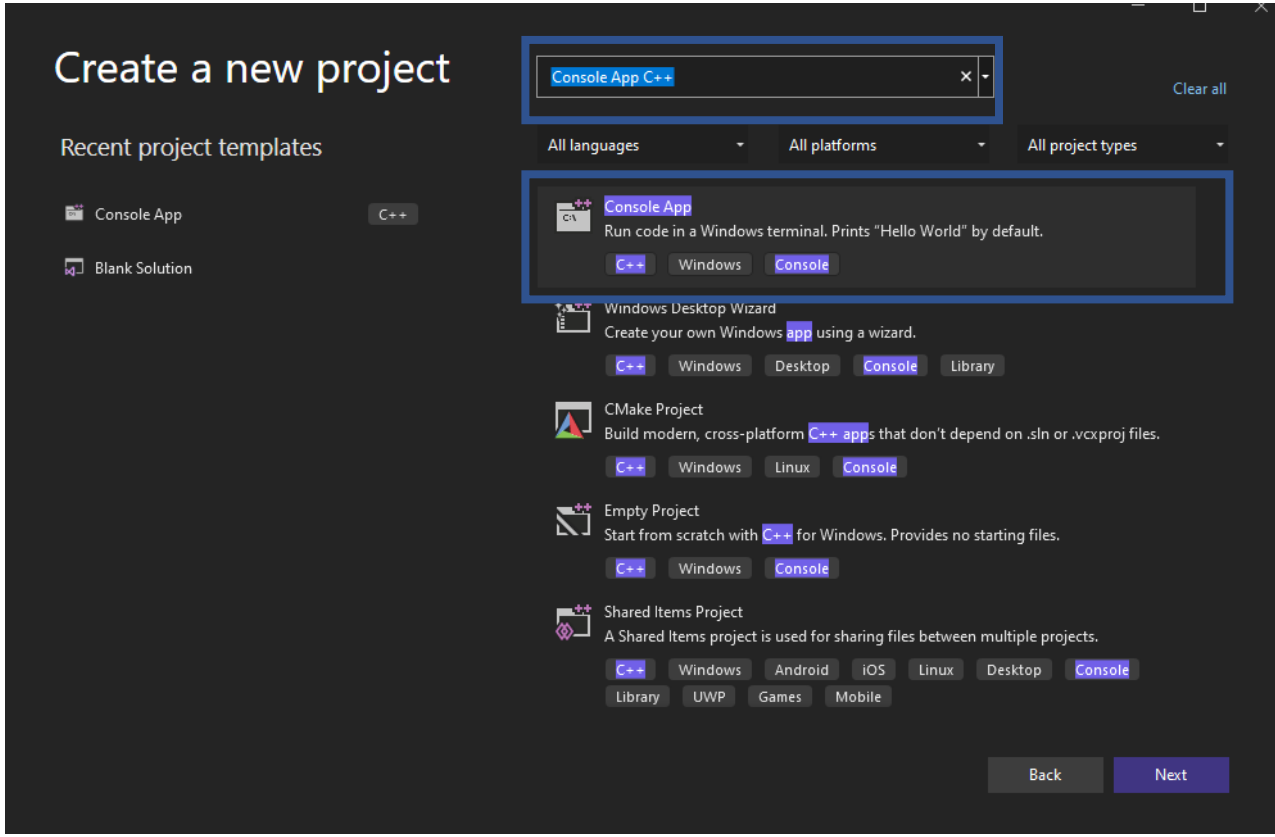
- During installation, please choose “Desktop development with C++”



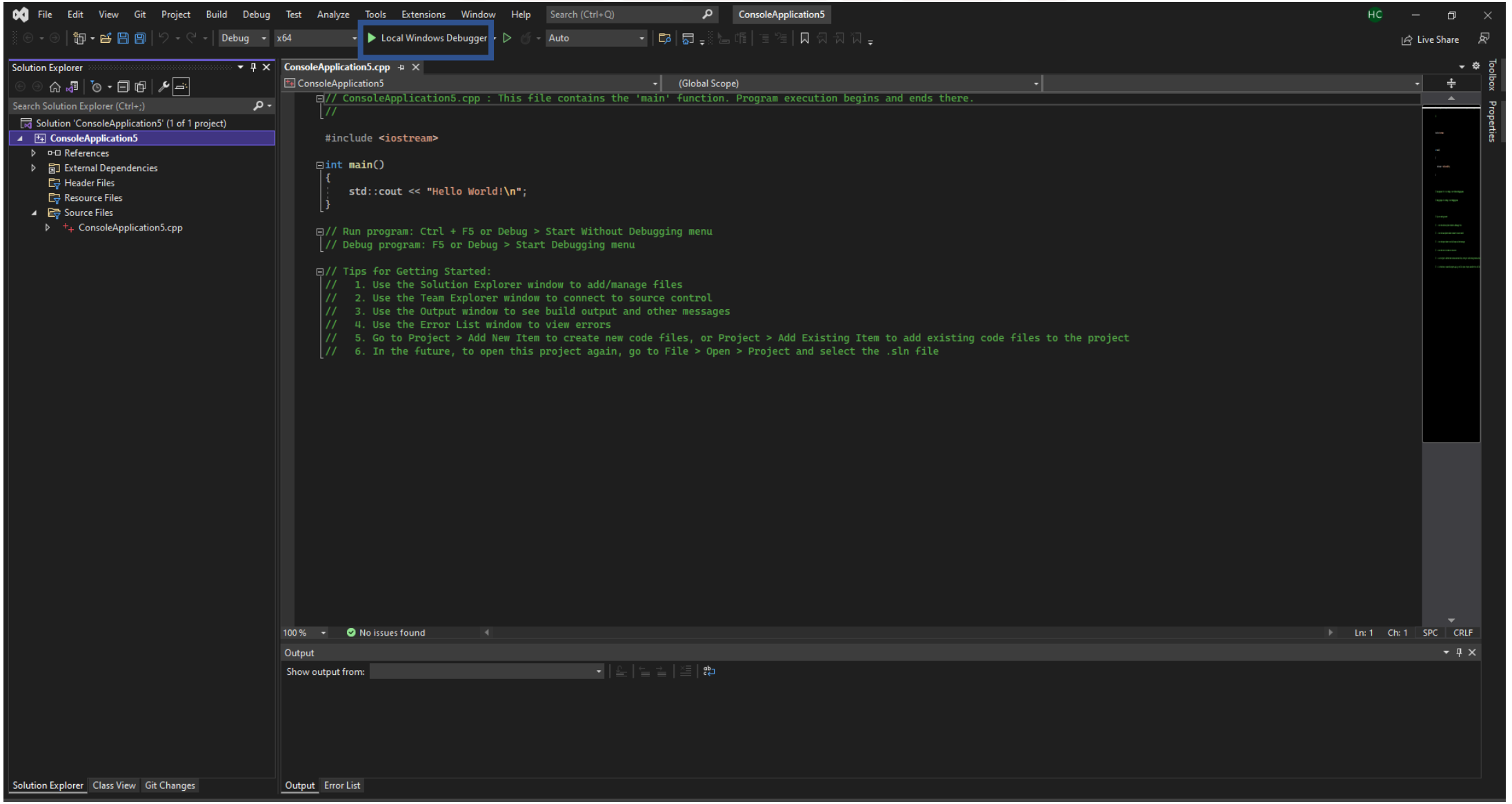
Create your first C++ program

- 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

Online GDB (For Windows / Mac user)

- https://www.onlinegdb.com/online_c++_compiler#

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