

Lecture 2: Basic C++ I/O and Expression

Class page: <https://github.com/tsung-wei-huang/cs1410-40>

Dr. Tsung-Wei Huang
Department of Electrical and Computer Engineering
University of Utah, Salt Lake City, UT



Announcement

Lab – virtual or physical?

Our World is Driven by Computing

Transmission and Interface

Connecting



Screen Phone



Mobile Phone

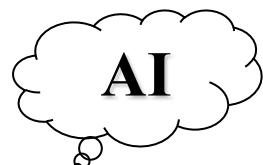
Smart Mobile Phone



PDA



Palm PC



AI



Set Top Box



Auto PC



Handheld PC

Computing

Audio and Video

Consumer



Game Console



DSC

Data Processing



MP3



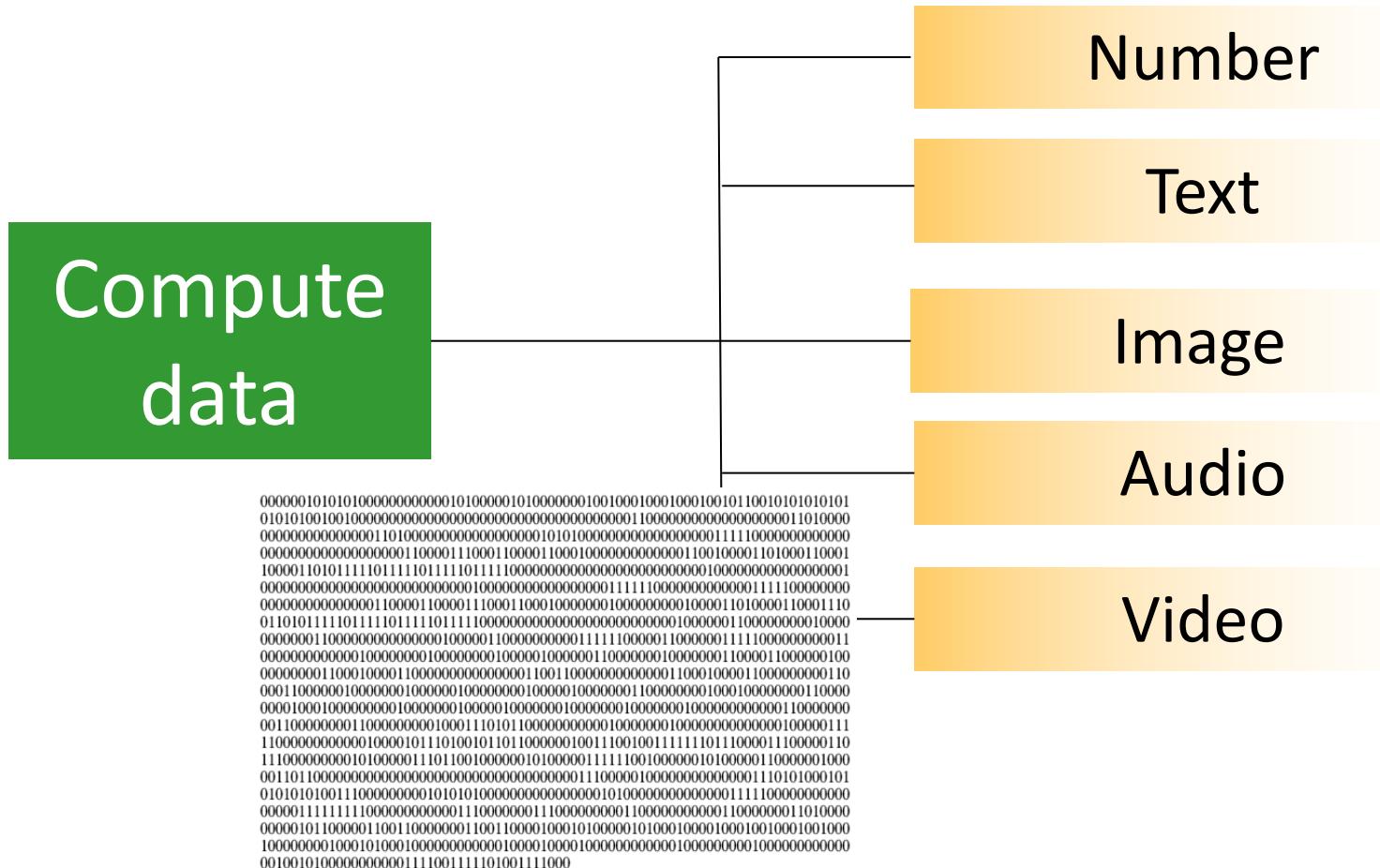
Net TV



Thin Client

Computing

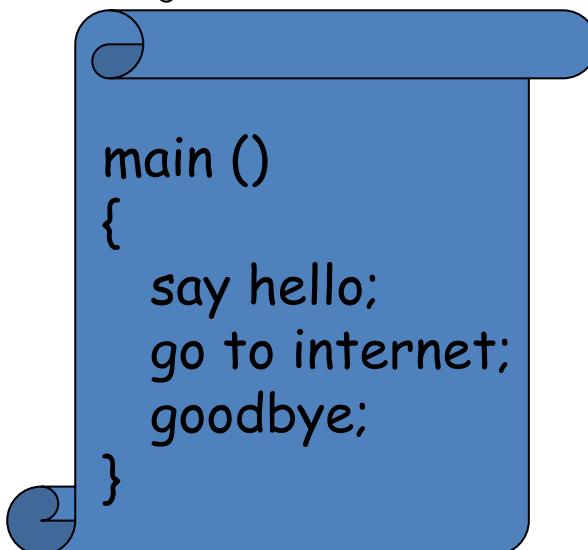
Computing Plays a Key Role



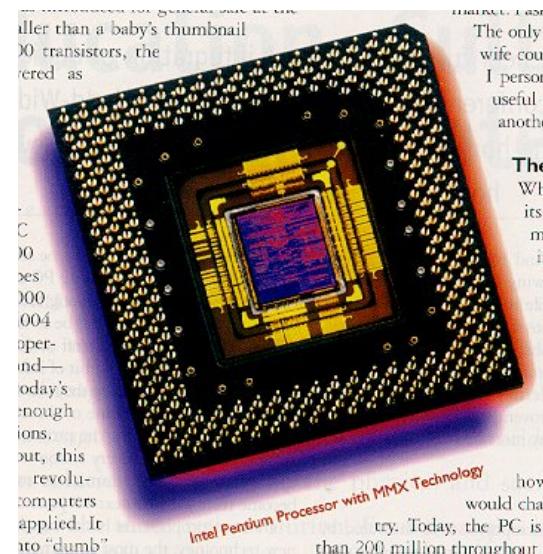
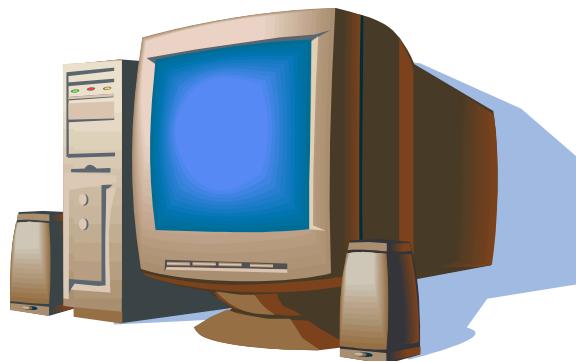
Computer see everything in binary (0 and 1)

Hardware vs Software

The focus
of this
course

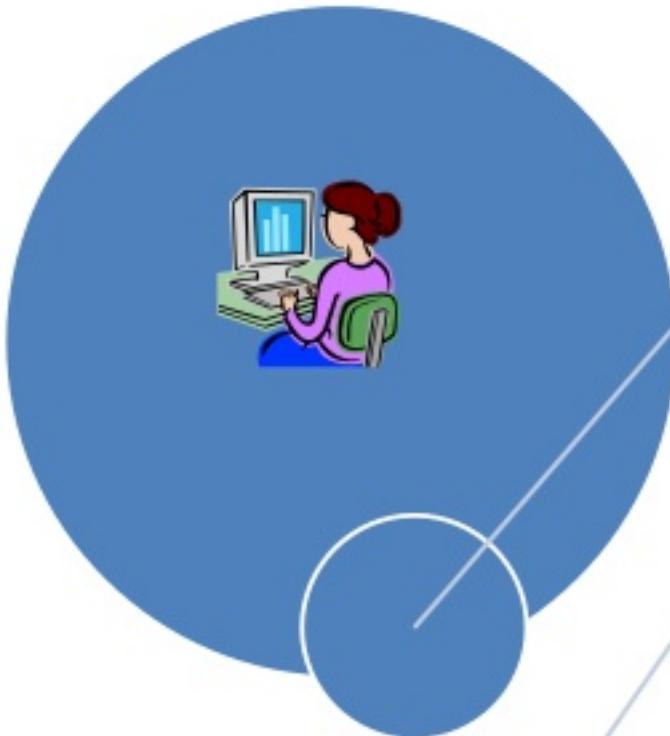


Software



Hardware

What is Software?



Software is a general term for the various kinds of programs used to operate computers and related devices. (The term hardware describes the physical aspects of computers and related devices.)

How do We Talk to Computers?

- Like we talk to human through “language”
 - English, Mandarin, Hindi, ... => depending on countries
 - C++, PHP, Python, JS, ... => depending on applications



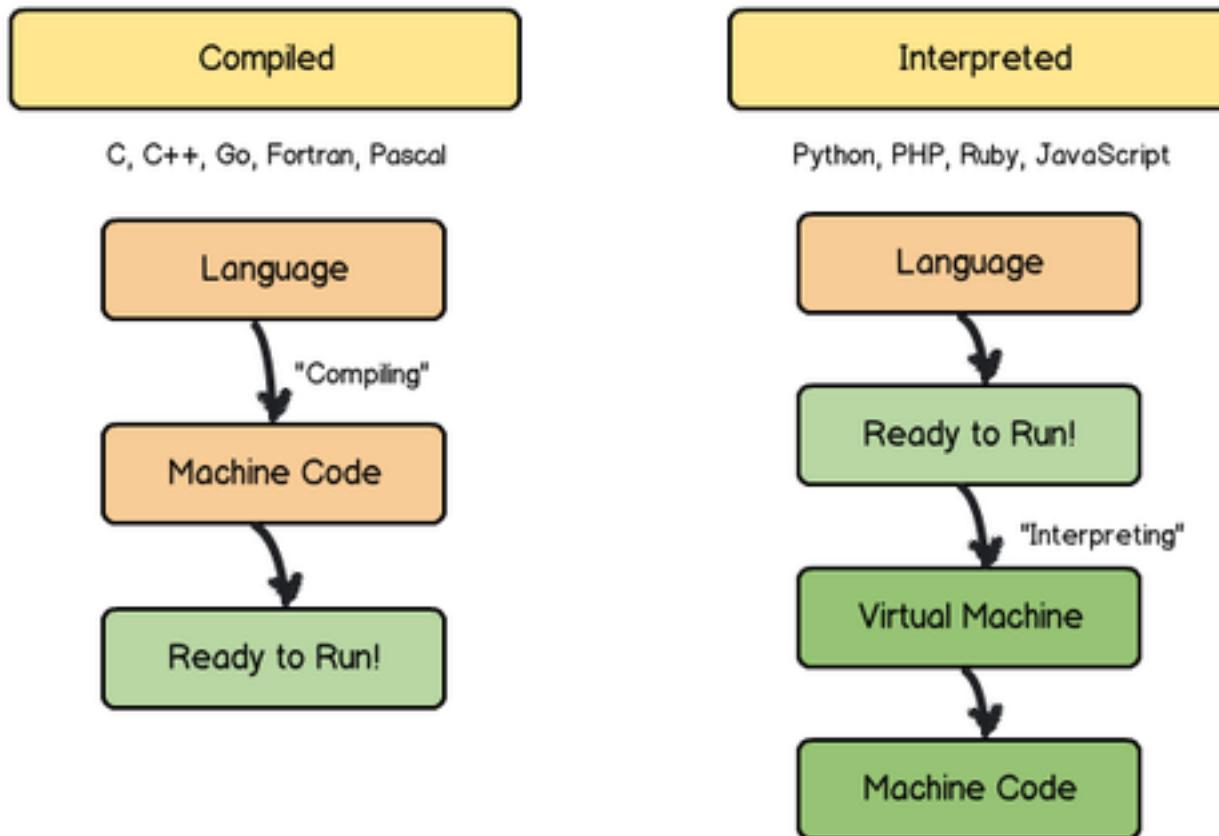
Top 10 Programming Languages

Rank	Language	Type	Score
1	Python	🌐💻⚙️	100.0
2	Java	🌐📱💻	96.3
3	C	📱💻⚙️	94.4
4	C++	📱💻⚙️	87.5
5	R	💻	81.5
6	JavaScript	🌐	79.4
7	C#	🌐📱💻⚙️	74.5
8	Matlab	💻	70.6
9	Swift	📱💻	69.1
10	Go	🌐💻	68.0

1	Java	
2	C	
3	Python	
4	C++	
5	Visual Basic .NET	
6	Javascript	
7	C#	
8	PHP	
9	SQL	
10	Objective-C	
11	MATLAB	
12	R	
13	Perl	
14	Assembly Language	
15	Swift	
16	Go	
17	Delphi/Object Pascal	
18	Ruby	
19	PL/SQL	
20	Visual Basic	

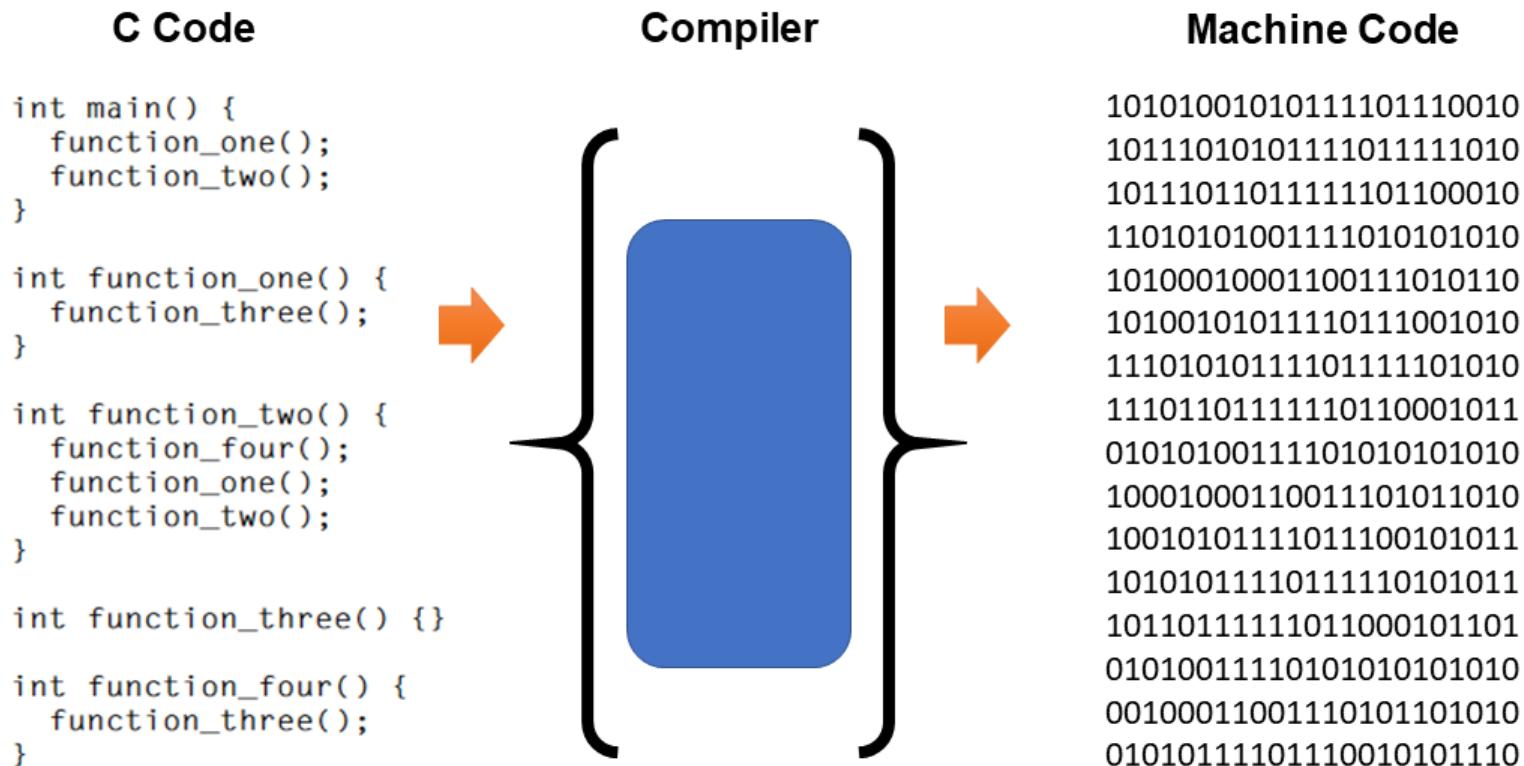
Two Types of Programming Languages

□ Compiled language vs Interpreted language



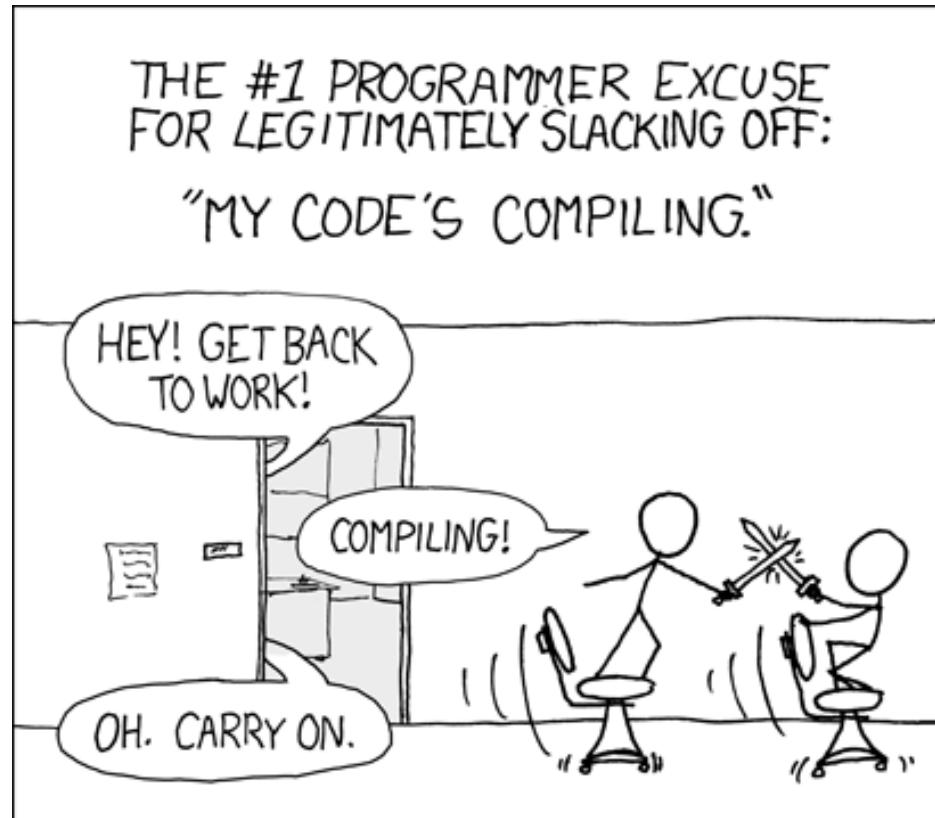
Compilation (e.g., C/C++)

- Compile your “entire” code to an executable



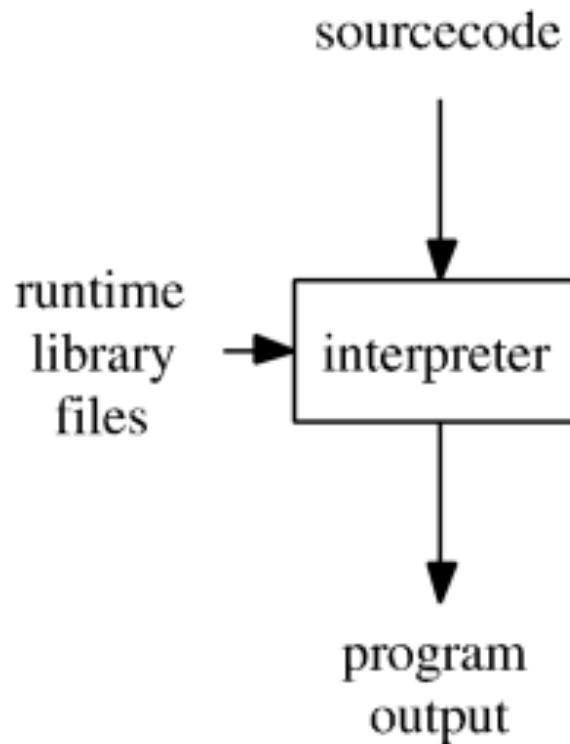
Compilation can be Very Long

- May take **HOURS** to compile a large codebase ...
 - Do you know how long it takes to compile Google code?



Interpretation

- Run your code line by line



Interpretation can be Very Slow

Slowest things on earth:



Python



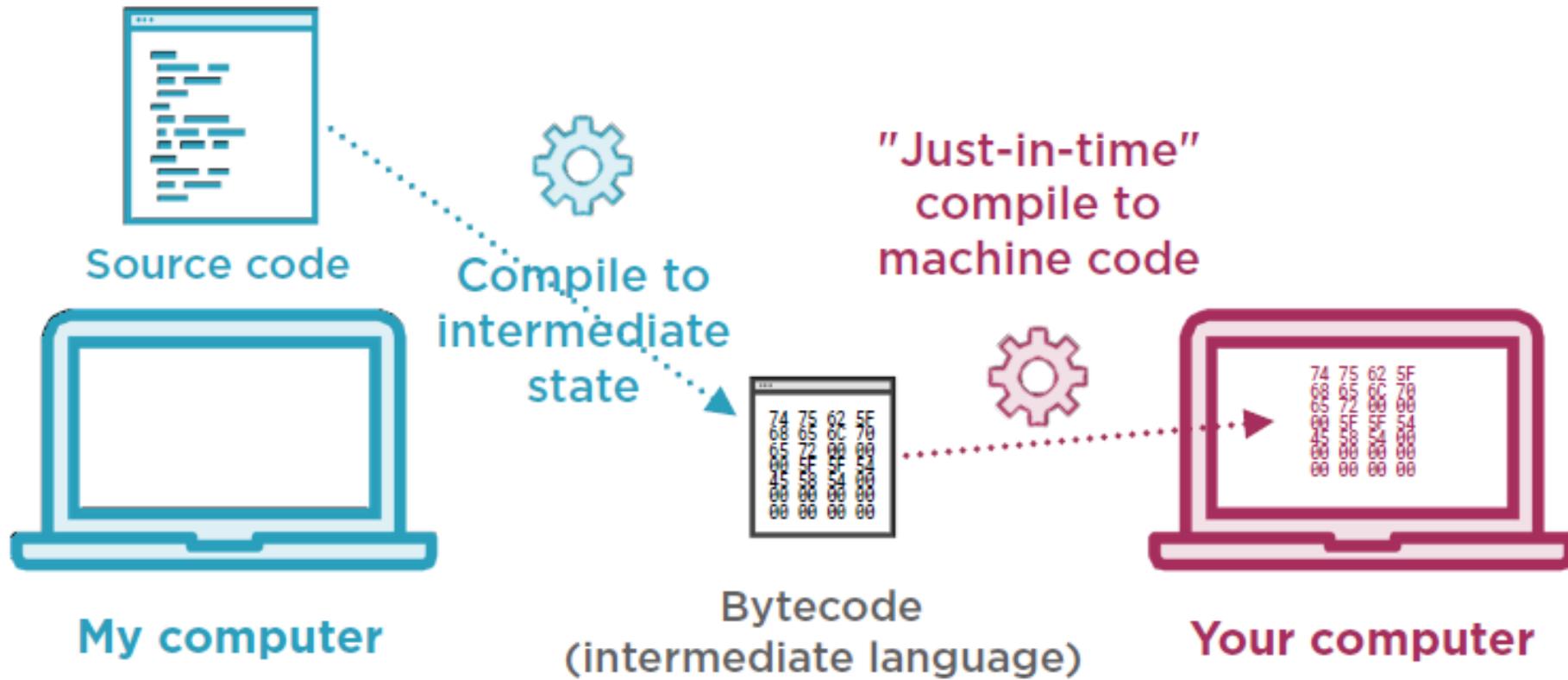
Pros and Cons

- ❑ Again, real cost depends on applications

Compiled		Interpreted	
PROS	CONS	PROS	CONS
ready to run	not cross platform	cross-platform	interpreter required
often faster	inflexible	simpler to test	often slower
source code is private	extra step	easier to debug	source code is public

Just-in-time Compilation

- ❑ A mix of both (e.g., JavaScript)



What Types of Language you will use?

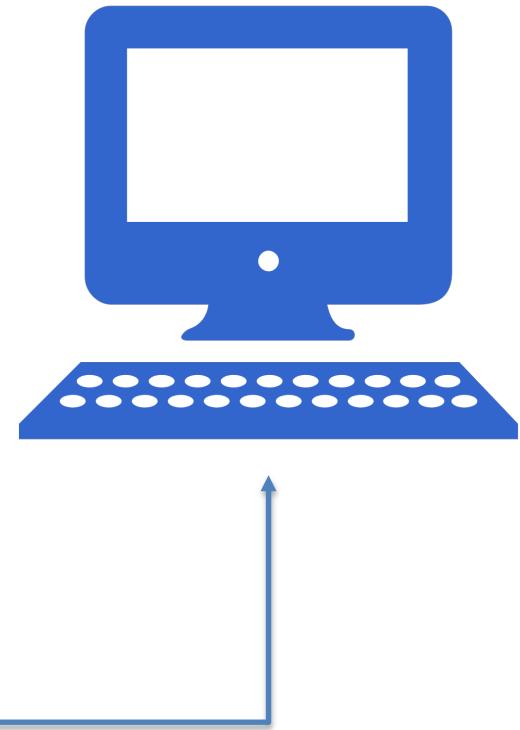
- I am doing embedded software for autonomous cars
- I am working on performance-critical applications
- I am building a high-frequency trading platform
- I am prototyping my software products
- I am building circuit design software
- I am creating my own website

Programming Environment

☐ Terminal

- ☐ A computer terminal is an electronic or electromechanical hardware device that can be used for *entering data into*, and displaying or printing data from, a computer or a computing system.

```
Last login: Wed Mar 2 13:15:38 on ttys000
Artoo:~ paperweight$ defaults write com.apple.finder AppleShowAllFiles -bool TRUE
Artoo:~ paperweight$ killall Finder
```



Editor (We will use VIM)

```
61 +-- 3 lines: Is a socket 'connection oriented' ?-----|211 EXPORT_SYMBOL(netpoll_poll_dev);  
64 static inline int connection_based(struct sock *sk)|212  
65 +-- 3 lines: {-----|213 void netpoll_poll(struct netpoll *np)  
68 |214 +-- 3 lines: {-----|217 EXPORT_SYMBOL(netpoll_poll);  
69 static int receiver_wake_function(wait_queue_t *wait, unsigned|218  
70 |219 static void refill_skbs(void)  
71 +-- 10 lines: {-----|220 +-- 14 lines: {-----|224  
81 +-- 3 lines: Wait for a packet...|225 static void zap_completion_queue(void)  
84 static int wait_for_packet(struct sock *sk, int *err, long *tim|226 +-- 26 lines: {-----|234  
85 +-- 45 lines: {-----|227  
130 |228  
131 /**|229  
132 * _skb_recv_datagram - Receive a datagram skbuff|230 static struct sk_buff *find_skb(struct netpoll *np, int len, int  
133 * @sk: socket|231 |232  
134 * @flags: MSG_ flags|233 +-- 24 lines: {-----|234  
135 * @peeked: returns non-zero if this packet has been seen befo|235 static int netpoll_owner_active(struct net_device *dev)  
136 * @err: error code returned|236 +-- 9 lines: {-----|237  
137 *|238  
datagram.c 132,1 8%|239  
1 +-- 6 lines: net/core/dst.c Protocol independent destination c|240 void netpoll_send_skb_on_dev(struct netpoll *np, struct sk_buff *  
7 |241 |242  
8 #include <linux/bitops.h>|243 |244  
9 #include <linux/errno.h>|245 +-- 54 lines: {-----|245  
10 #include <linux/init.h>|246 EXPORT_SYMBOL(netpoll_send_skb_on_dev);  
11 #include <linux/kernel.h>|247  
12 #include <linux/workqueue.h>|248 void netpoll_send_udp(struct netpoll *np, const char *msg, int le  
13 #include <linux/mm.h>|249 +-- 59 lines: {-----|249  
14 #include <linux/module.h>|250 EXPORT_SYMBOL(netpoll_send_udp);  
15 #include <linux/slab.h>|251  
16 #include <linux/netdevice.h>|252 static void arp_reply(struct sk_buff *skb)  
17 #include <linux/skbuff.h>|253 +-- 116 lines: {-----|253  
18 #include <linux/string.h>|254  
19 #include <linux/types.h>|255 int __netpoll_rx(struct sk_buff *skb)  
20 #include <net/net_namespace.h>|256 +-- 90 lines: {-----|256  
21 #include <linux/sched.h>|257  
22 |258 void netpoll_print_options(struct netpoll *np)  
dst.c 1,1 Top netpoll.c 645,30 39%|259 +-- 14 lines: {-----|259  
|260 EXPORT_SYMBOL(netpoll_print_options);
```

There are also online C++ Editors

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



The screenshot shows the interface of the Online C++ Compiler. At the top, there's a tab labeled "main.cpp". Below it is a code editor window containing the following C++ code:

```
1 //*****  
2  
3 //*****  
4 //*****  
5 //*****  
6 //*****  
7 //*****  
8 //*****  
9 #include <iostream>  
10  
11 int main()  
12 {  
13     std::cout<<"Hello World";  
14  
15     return 0;  
16 }  
17
```

Below the code editor is a terminal window showing the output of the program. The output is:

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

A small circular icon with a speech bubble and a green dot is located in the bottom right corner of the terminal window.

Output - I

```
1 // Text-printing program.
2 #include <iostream> // allows program to output data to the screen
3
4 // function main begins program execution
5 int main()
6 {
7     std::cout << "Welcome to C++!\n"; // display message
8
9
10    return 0; // indicate that program ended successfully
11 } // end function main
```

Welcome to C++!

Output - II

- A single statement can print multiple lines by using newline characters.
 - Each time the \n (newline) is encountered, the screen cursor is positioned to the beginning of the next line.
-

```
1 // Printing multiple lines of text with a single statement.
2 #include <iostream> // allows program to output data to the screen
3
4 // function main begins program execution
5 int main()
6 {
7     std::cout << "Welcome\n to\n\nC++!\n";
8 }
9 // end function main
```

```
Welcome
to

C++!
```

Header files #include <xxx>

Header files contain definitions of **Functions and Variables**, which is imported or used into any C++ program by using the pre-processor #include statement.

Header file have an extension ".h" which contains C++ function declaration and macro definition.

Why need of header files

When we want to use any function in our C++ program then first we need to import their definition from C++ library, for importing their declaration and definition we need to include header file in program by using #include.

Input - I

- The input stream object **std::cin** and the **stream extraction operator, >>**, can be used obtain data from the user at the keyboard.

```
1 // Addition program that displays the sum of two integers.
2 #include <iostream> // allows program to perform input and output
3
4 // function main begins program execution
5 int main()
6 {
7     // variable declarations
8     int number1; // first integer to add
9     int number2; // second integer to add
10    int sum; // sum of number1 and number2
11
12    std::cout << "Enter first integer: "; // prompt user for data
13    std::cin >> number1; // read first integer from user into number1
14
15    std::cout << "Enter second integer: "; // prompt user for data
16    std::cin >> number2; // read second integer from user into number2
17
18    sum = number1 + number2; // add the numbers; store result in sum
19
20    std::cout << "Sum is " << sum << std::endl; // display sum; end line
21
22 } // end function main
```

Input - II

```
9     int number1; // first integer to add
10    int number2; // second integer to add
11    int sum; // sum of number1 and number2
```

- **Declarations** introduce the identifiers **number1**, **number2** and **sum** into programs; they are the names of **variables**.
- A variable is a location in the computer's memory where a value can be stored for use by a program.
- Variables **number1**, **number2** and **sum** are data of type **int**, meaning that these variables will hold **integer** values.
- All variables must be declared with a name and a data type before they can be used in a program.
- If more than one name is declared in a declaration, the names are separated by commas (,) → called **comma-separated list**.
 - **int number1, number2;**

Input – III (other data types)

- Data type **double** is for specifying real numbers, and data type **char** for specifying character data.
 - Real numbers are numbers with decimal points, such as 3.4, 0.0 and –11.19.
- A **char** variable may hold only a single lowercase letter, a single uppercase letter, a single digit or a single special character (e.g., \$ or *).
- Types such as **int**, **double** and **char** are called **fundamental types**.
- Fundamental-type names are **keywords** and therefore must appear in all lowercase letters.

Fundamental Data Types

❑ <https://en.cppreference.com/w/cpp/language/types>

Type specifier	Equivalent type	Width in bits by data model				
		C++ standard	LP32	ILP32	LLP64	LP64
short	short int	at least 16	16	16	16	16
short int						
signed short						
signed short int						
unsigned short						
unsigned short int						
int	int	at least 16	16	32	32	32
signed						
signed int						
unsigned						
unsigned int						
long	long int	at least 32	32	32	32	64
long int						
signed long						
signed long int						
unsigned long						
unsigned long int	unsigned long int	at least 64	64	64	64	64
long long						
long long int						
signed long long						
signed long long int						
unsigned long long	unsigned long long int	(C++11)	64	64	64	64
unsigned long long int						

Range of Data Types

Type	Size in bits	Format	Value range	
			Approximate	Exact
character	8	signed		-128 to 127
		unsigned		0 to 255
	16	unsigned		0 to 65535
	32	unsigned		0 to 1114111 (0x10ffff)
integer	16	signed	$\pm 3.27 \cdot 10^4$	-32768 to 32767
		unsigned	$0 \text{ to } 6.55 \cdot 10^4$	0 to 65535
	32	signed	$\pm 2.14 \cdot 10^9$	-2,147,483,648 to 2,147,483,647
		unsigned	$0 \text{ to } 4.29 \cdot 10^9$	0 to 4,294,967,295
	64	signed	$\pm 9.22 \cdot 10^{18}$	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
		unsigned	$0 \text{ to } 1.84 \cdot 10^{19}$	0 to 18,446,744,073,709,551,615
floating point	32	IEEE-754 	<ul style="list-style-type: none"> min subnormal: $\pm 1.401,298,4 \cdot 10^{-45}$ min normal: $\pm 1.175,494,3 \cdot 10^{-38}$ max: $\pm 3.402,823,4 \cdot 10^{38}$ 	<ul style="list-style-type: none"> min subnormal: $\pm 0x1p-149$ min normal: $\pm 0x1p-126$ max: $\pm 0x1.fffffep+127$
	64	IEEE-754 	<ul style="list-style-type: none"> min subnormal: $\pm 4.940,656,458,412 \cdot 10^{-324}$ min normal: $\pm 2.225,073,858,507,201,4 \cdot 10^{-308}$ max: $\pm 1.797,693,134,862,315,7 \cdot 10^{308}$ 	<ul style="list-style-type: none"> min subnormal: $\pm 0x1p-1074$ min normal: $\pm 0x1p-1022$ max: $\pm 0x1.fffffffffffffp+1023$

C++ is a Strongly Typed System

Declare Variables

- Using the fundamental data types
 - e.g. int my_variable
- A variable name is any valid **identifier** that is not a keyword.
- An identifier is a series of characters consisting of letters, digits and underscores (_) that does **not begin with a digit**.
- C++ is **case sensitive**—uppercase and lowercase letters are different, so **a1** and **A1** are different identifiers.
- Declarations of variables can be placed almost anywhere in a program, but they must appear **before** their corresponding variables are used in the program.

Variable Naming is Important ...

- ❑ Tip 1: C++ allows identifiers of any length, but your C++ implementation may restrict identifier lengths.
Use identifiers of 31 characters or fewer to ensure portability
- ❑ Tip 2: choose meaningful identifiers makes a program self-explanatory—a person can understand the program simply by reading it rather than having to refer to the manuals or comments
 - ❑ int parent, PARENT;
 - ❑ int mom, dad;
 - ❑ Good naming is always fundamental to effective code reviews and collaboration

Adding Integers - I

```
13     std::cout << "Enter first integer: " // prompt user for data  
14     std::cin >> number1; // read first integer from user into number1
```

- A **prompt** directs the user to take a specific action.
- A **cin** statement uses the **input stream object cin** (of namespace **std**) and the **stream extraction operator, >>**, to obtain a value from the keyboard.
- Using the stream extraction operator with **std::cin** takes character input from the standard input stream, which is usually the keyboard.

Adding Integers - II

14

```
std::cin >> number1; // read first integer from user into number1
```

- When the computer executes an input statement, it waits for the user to enter a value for variable **number1**.
- The user types the number (as characters) then **press the *Enter* key** (or the Return key) to send the characters to the computer.
- The computer converts the character representation of the number to an integer and put the **value** to the variable **number1**.
- Any subsequent references to **number1** in this program will use this same value.

Adding Integers - III

19

```
sum = number1 + number2; // add the numbers; store result in sum
```

20

- In this program, an assignment statement adds the values of variables **number1** and **number2** and assigns the result to variable **sum** using the **assignment operator** **=**.
 - Most calculations are performed in assignment statements.
- The **= operator** and the **+ operator** are called **binary operators** because each has two operands.

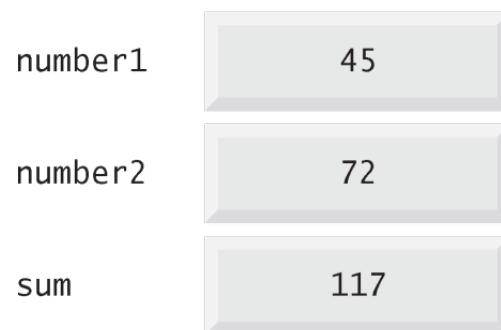
Adding Integers - IV

```
21     std::cout << "Sum is " << sum << std::endl; // display sum; end line
```

- **std::endl is a so-called stream manipulator.**
 - The name endl is an abbreviation for “end line”
- **std::endl outputs a newline and flushes the output buffer.**
 - Some systems may accumulate outputs in the machine (output buffer) and display them simultaneously on the screen.
 - std::endl forces any accumulated outputs to be displayed.
- **Using multiple stream insertion operators (<<) in a single statement is referred to as concatenating, chaining or cascading stream insertion operations.**
- **Calculations can also be performed in output statements.**

Memory Concepts

- Variable names such as **number1**, **number2** and **sum** actually correspond to **locations** in the computer's memory.
- Every variable has a **name**, a **type**, a **size** and a **value**.
- When a value is placed in a memory location, it **overwrites** the previous value in that location → **destructive**
- When a value is read out of a memory location, the process is **nondestructive**.



Yes, Memory is Limited

- ❑ You may not create too many variables
 - ❑ e.g. one trillion integers or more



Surface Laptop 3 - 13.5", Black (metal), Intel Core i5, 8GB, 256GB

Wish list

Slim and stylish, available in 13.5" and 15" touchscreens, rich color options,¹ and two durable finishes. Make a powerful statement and get improved speed, performance, and all-day battery life.²

Open gallery

The Microsoft Store Promise for Surface

Shop with confidence at Microsoft Store. We're offering 60-day returns on Surface products, plus free digital workshops, remote learning opportunities, and more to help you get the most from your new device.*

[Learn more >](#)

Bundle and save with the Surface Laptop 3 Essentials Bundle

Includes your choice of Surface Laptop 3, Microsoft 365 and Microsoft Complete Protection Plan.

[Build your bundle >](#)

This literally means your program can run up to **8 GB** memory

Illustration

```
std::cout << "Enter first integer: ";
std::cin >> number1;
std::cout << "Enter second integer: ";
std::cin >> number2;
sum = number1 + number2;
std::cout << "Sum is " << sum << std::endl;
```

Output

Memory

sum	0012FF74
	0012FF75
number2	0012FF76
	0012FF77
	0012FF78
	0012FF79
number1	0012FF7A
	0012FF7B
	0012FF7C
	0012FF7D
	0012FF7E
	0012FF7F

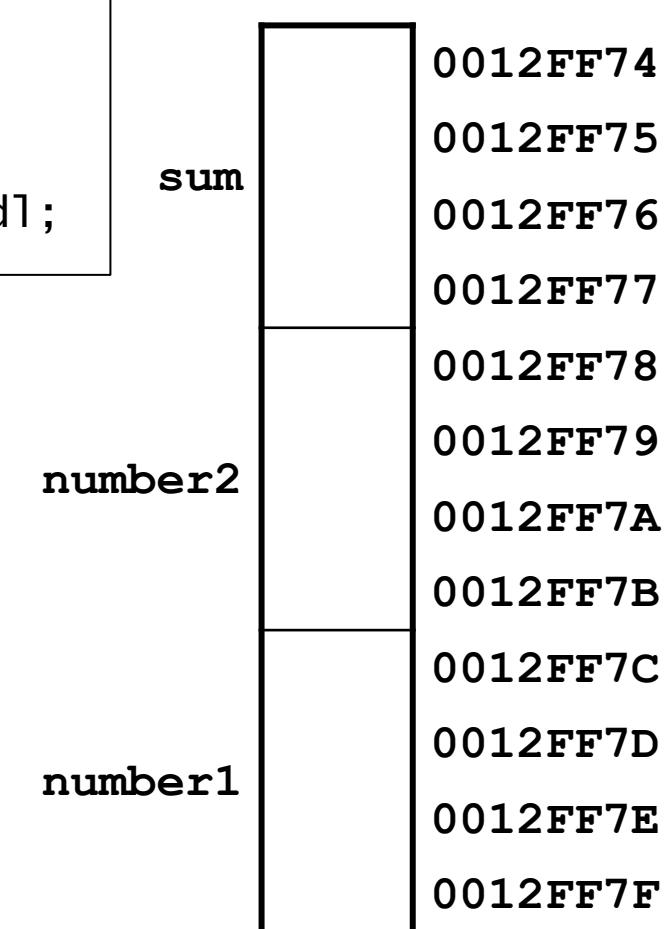
Illustration

```
std::cout << "Enter first integer: ";
std::cin >> number1;
std::cout << "Enter second integer: ";
std::cin >> number2;
sum = number1 + number2;
std::cout << "Sum is " << sum << std::endl;
```

Output

```
Enter first integer: _
```

Memory



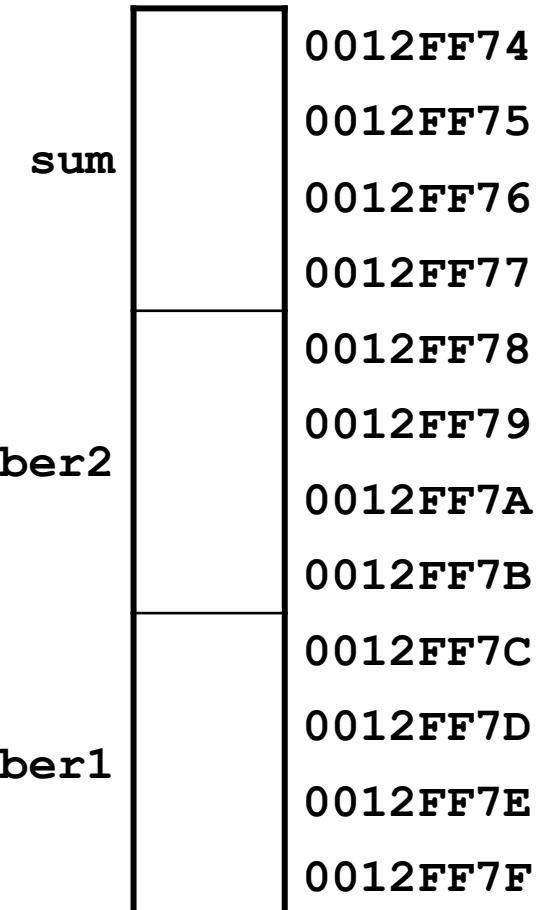
Illustration

```
std::cout << "Enter first integer: ";
std::cin >> number1;
std::cout << "Enter second integer: ";
std::cin >> number2;
sum = number1 + number2;
std::cout << "Sum is " << sum << std::endl;
```

Output

```
Enter first integer: 45_
```

Memory



Illustration

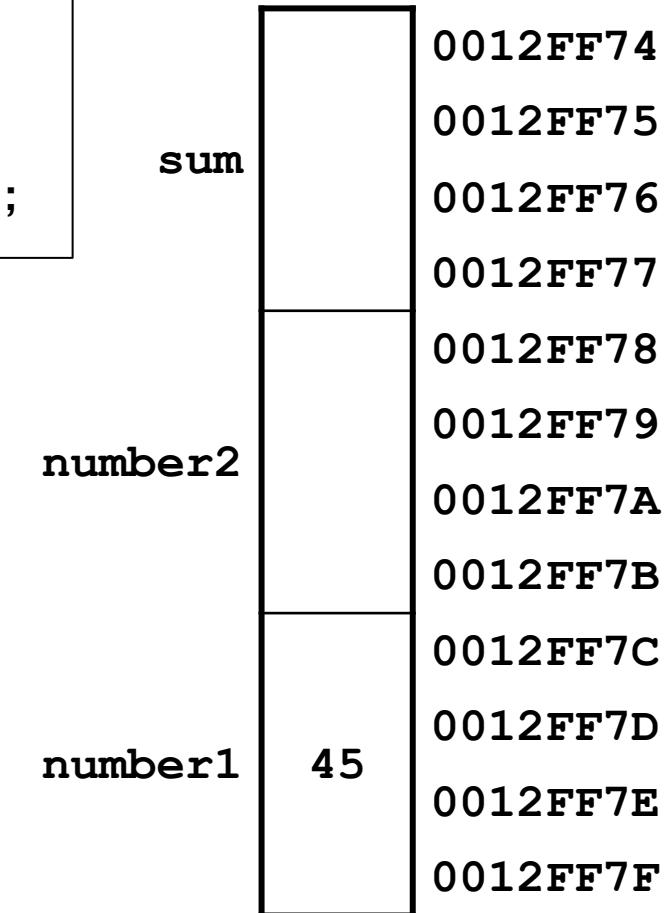
```
std::cout << "Enter first integer: ";
std::cin >> number1;
std::cout << "Enter second integer: ";
std::cin >> number2;
sum = number1 + number2;
std::cout << "Sum is " << sum << std::endl;
```

Output

```
Enter first integer: 45
```

```
-
```

Memory



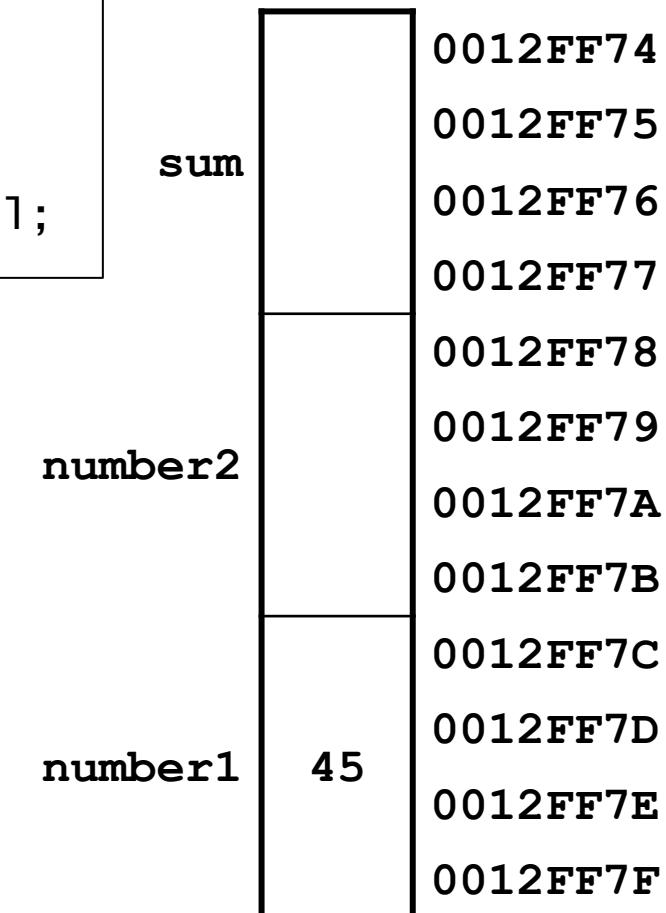
Illustration

```
std::cout << "Enter first integer: ";
std::cin >> number1;
std::cout << "Enter second integer: ";
std::cin >> number2;
sum = number1 + number2;
std::cout << "Sum is " << sum << std::endl;
```

Output

```
Enter first integer: 45
Enter second integer: _
```

Memory



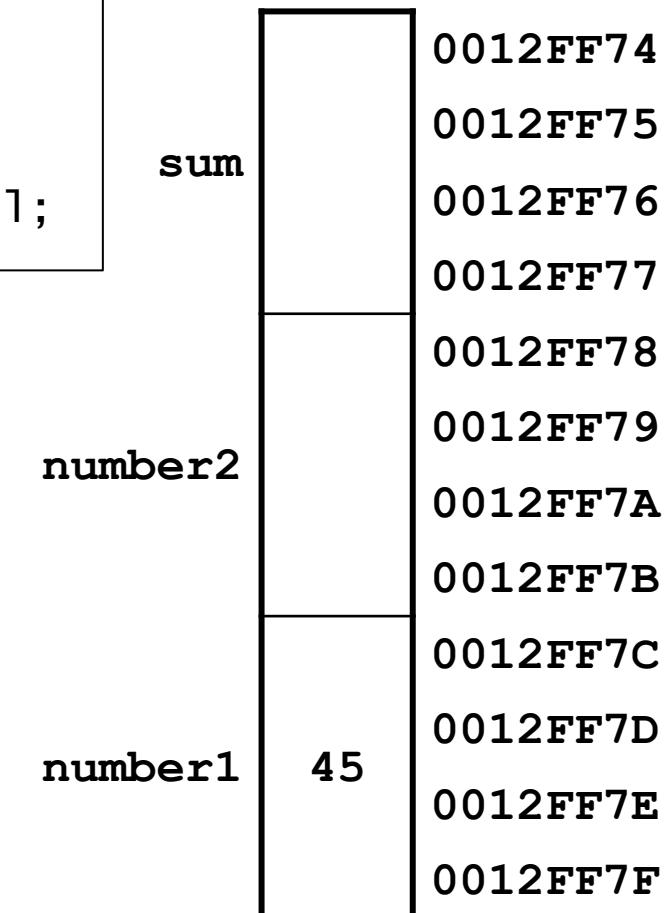
Illustration

```
std::cout << "Enter first integer: ";
std::cin >> number1;
std::cout << "Enter second integer: ";
std::cin >> number2;
sum = number1 + number2;
std::cout << "Sum is " << sum << std::endl;
```

Output

```
Enter first integer: 45
Enter second integer: 72_
```

Memory



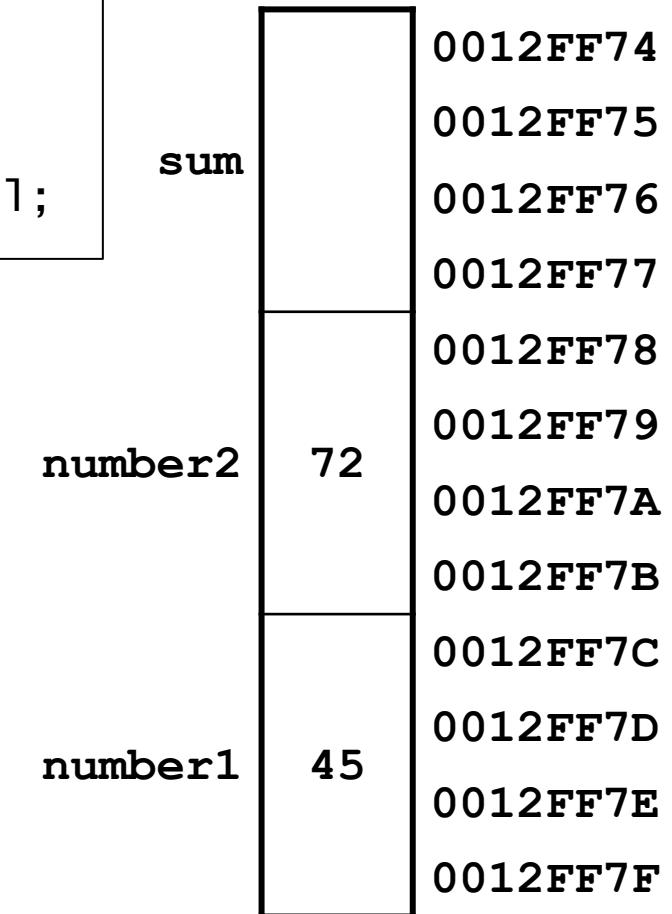
Illustration

```
std::cout << "Enter first integer: ";
std::cin >> number1;
std::cout << "Enter second integer: ";
std::cin >> number2;
sum = number1 + number2;
std::cout << "Sum is " << sum << std::endl;
```

Output

```
Enter first integer: 45
Enter second integer: 72
-
```

Memory



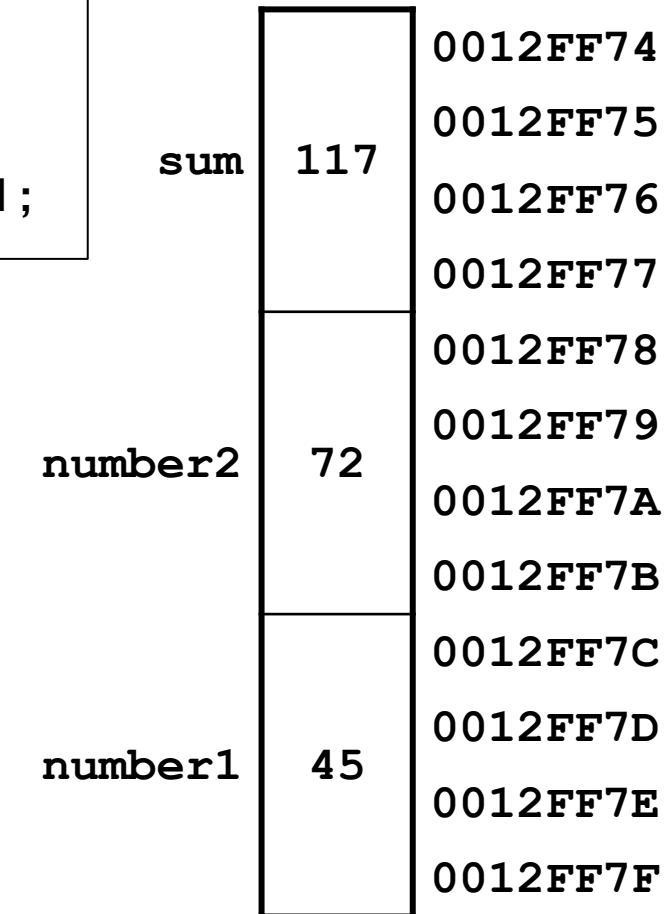
Illustration

```
std::cout << "Enter first integer: ";
std::cin >> number1;
std::cout << "Enter second integer: ";
std::cin >> number2;
sum = number1 + number2;
std::cout << "Sum is " << sum << std::endl;
```

Output

```
Enter first integer: 45
Enter second integer: 72
-
```

Memory



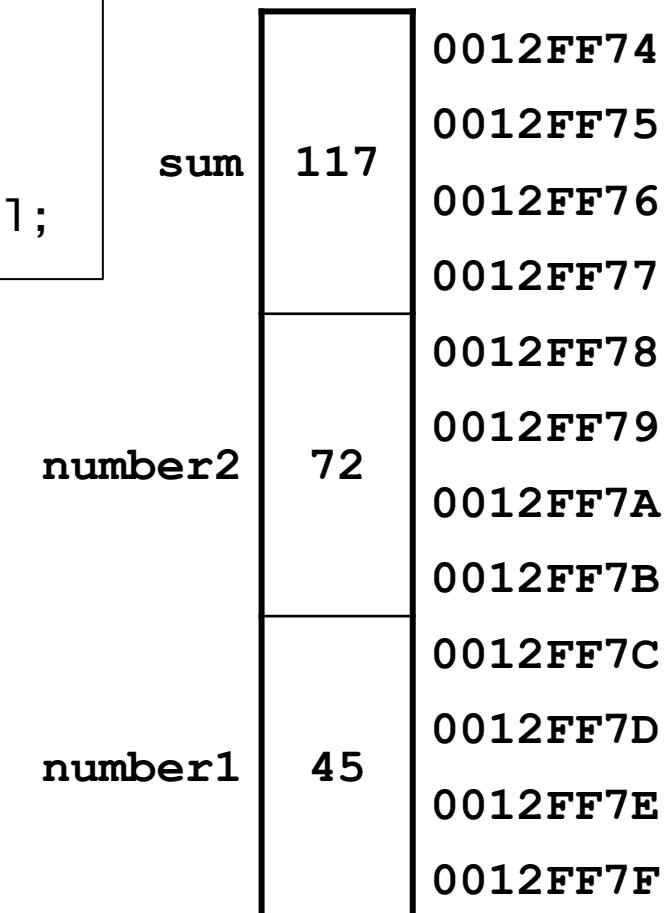
Illustration

```
std::cout << "Enter first integer: ";
std::cin >> number1;
std::cout << "Enter second integer: ";
std::cin >> number2;
sum = number1 + number2;
std::cout << "Sum is " << sum << std::endl;
```

Output

```
Enter first integer: 45
Enter second integer: 72
Sum is 117_
```

Memory



Illustration

```
std::cout << "Enter first integer: ";
std::cin >> number1;
std::cout << "Enter second integer: ";
std::cin >> number2;
sum = number1 + number2;
std::cout << "Sum is " << sum << std::endl;
```

Output

```
Enter first integer: 45
Enter second integer: 72
Sum is 117
Press any key to continue_
```

Memory

sum	117	0012FF74
		0012FF75
		0012FF76
		0012FF77
		0012FF78
number2	72	0012FF79
		0012FF7A
		0012FF7B
		0012FF7C
number1	45	0012FF7D
		0012FF7E
		0012FF7F

binary representation	hexadecimal	decimal
00000000 00000000 00000000 00000000	00000000	0
00000000 00000000 00000000 00000001	00000001	1
00000000 00000000 00000000 00000010	00000002	2
00000000 00000000 00000000 00000011	00000003	3
00000000 00000000 00000000 00000100	00000004	4
00000000 00000000 00000000 00000101	00000005	5
00000000 00000000 00000000 00000110	00000006	6
00000000 00000000 00000000 00000111	00000007	7
00000000 00000000 00000000 00001000	00000008	8
00000000 00000000 00000000 00001001	00000009	9
00000000 00000000 00000000 00001010	0000000A	10
00000000 00000000 00000000 00001011	0000000B	11
00000000 00000000 00000000 00001100	0000000C	12
00000000 00000000 00000000 00001101	0000000D	13
00000000 00000000 00000000 00001110	0000000E	14
00000000 00000000 00000000 00001111	0000000F	15

Value Representation

□ Binary, Octal, Decimal, Hex

- The six letters (in addition to the 10 integers) in **hexadecimal** represent: 10 (A), 11 (B), 12 (C), 13 (D), 14 (E), and 15 (F), respectively.

Name	Radix	Digits
Binary	2	0,1
Octal	8	0,1,2,3,4,5,6,7
Decimal	10	0,1,2,3,4,5,6,7,8,9
Hexadecimal	16	0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F

Common Arithmetic Operators

C++ operation	C++ arithmetic operator	Algebraic expression	C++ expression
Addition	+	$f + 7$	<code>f + 7</code>
Subtraction	-	$p - c$	<code>p - c</code>
Multiplication	*	bm or $b \cdot m$	<code>b * m</code>
Division	/	x / y or $\frac{x}{y}$ or $x \div y$	<code>x / y</code>
Modulus	%	$r \bmod s$	<code>r % s</code>

Summary

- ❑ C++ input and output
- ❑ Variables, data types
 - ❑ C++ is a strongly-typed system
- ❑ Adding two integers
- ❑ Value representation (binary, hex, decimal)
- ❑ Arithmetic operators (+, -, *, /)
- ❑ Programming Assignment 1 is out!
 - ❑ Email your solution to your section TA (see class page)
 - ❑ Due by class time on 9/2