Lecture 3: Arithmetic and Conditional Expressions

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

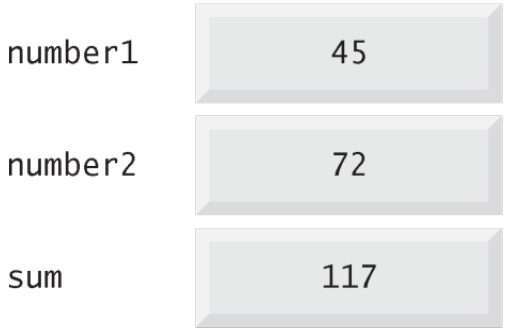
- **Programming Assignment 1 is due this Wed (9/2)**
  - Email your solution to your LAB session TA
  - Yasin: yasin.zamani@gmail.com
    - Section 43: 11:50-12:40 PM
    - Section 44: 12:55-01:45 PM
  - Dian-Lun: Dian-Lun.Lin@utah.edu
    - Section 41: 9:40-10:30 AM
    - Section 42: 10:45-11:35 AM
    - Section 45: 8:35-9:25 AM

Adding Two Integers

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

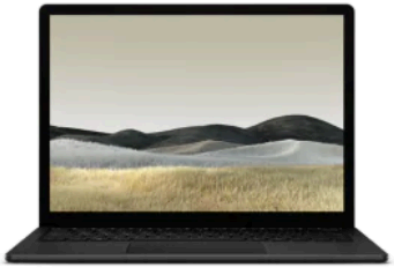
Memory Concepts

- 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



Open gallery

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]

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 >

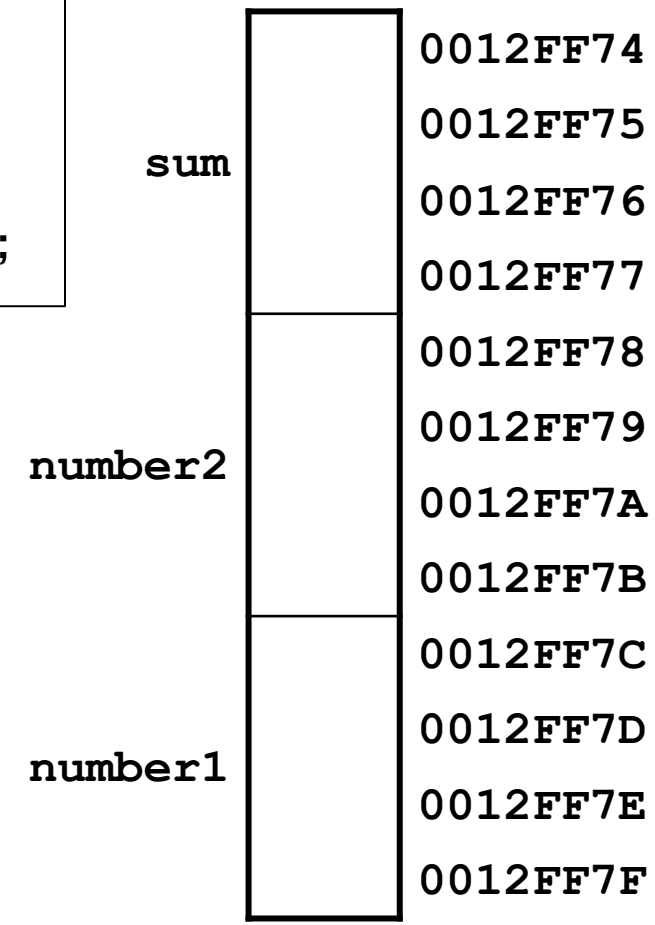
This literally means your program can run up to **8 GB** memory

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



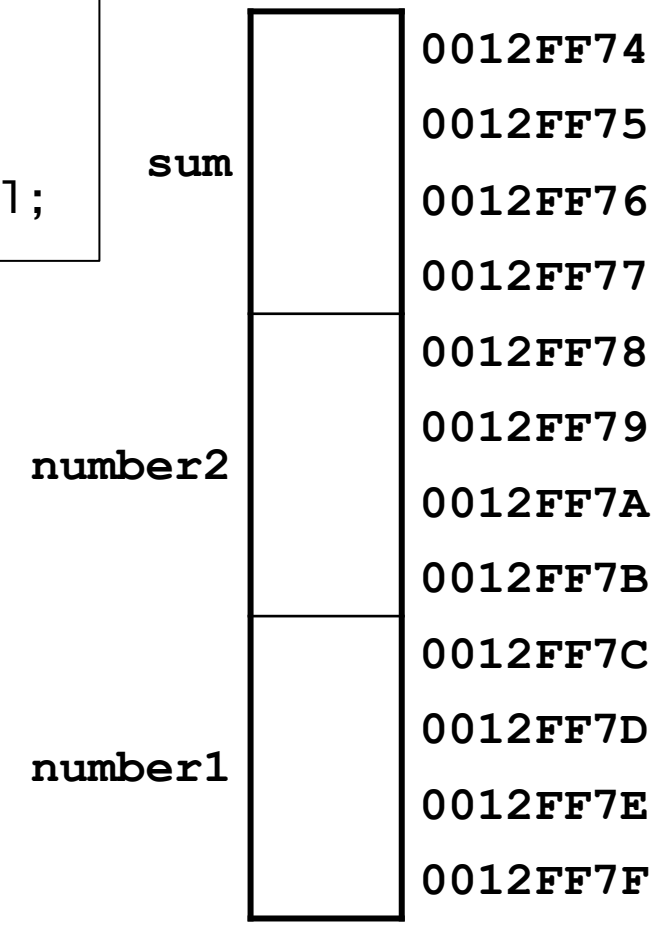
Illustration

```
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



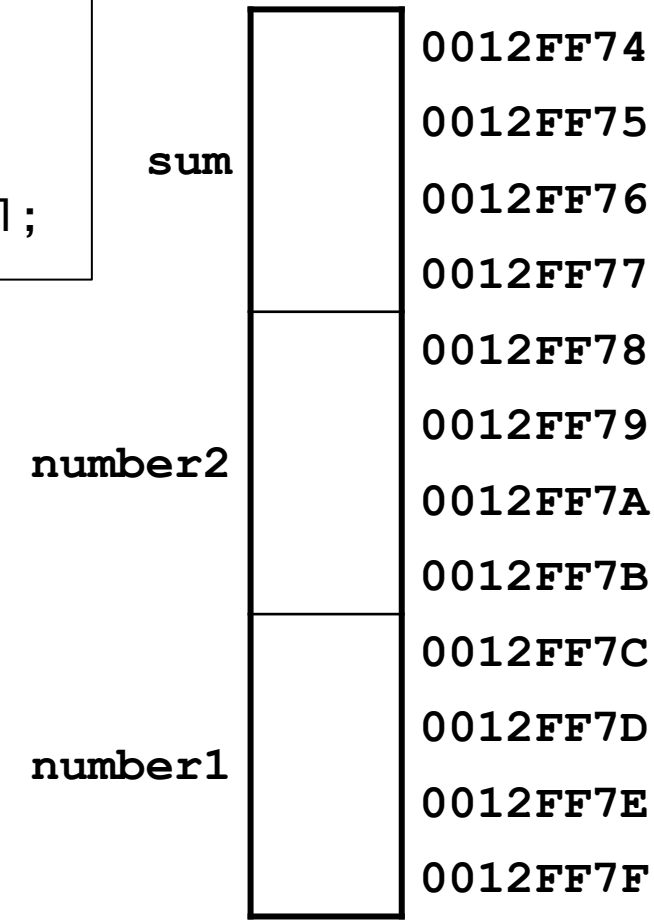
Illustration

```
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



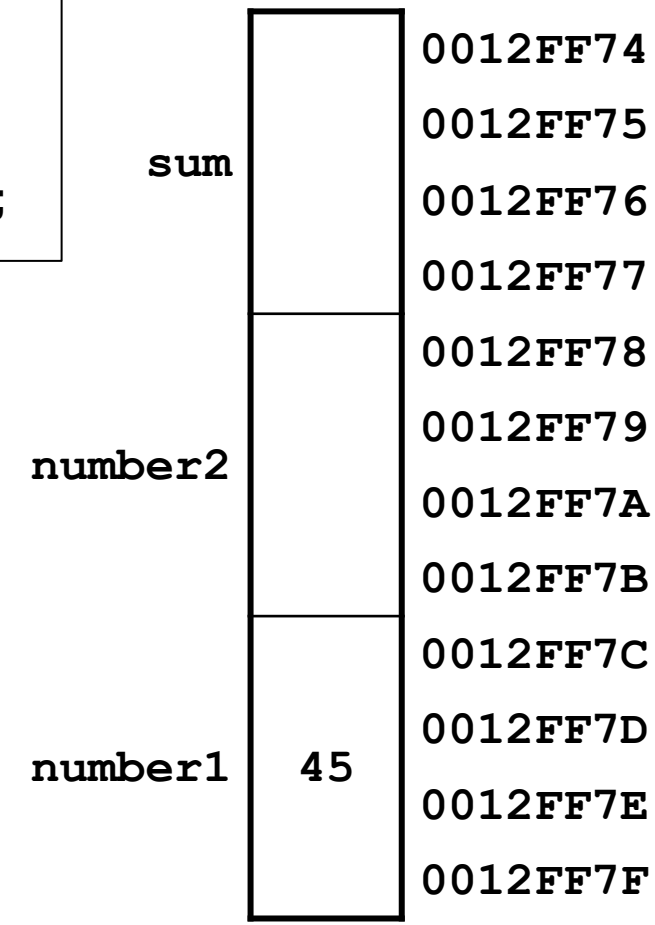
Illustration

```
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



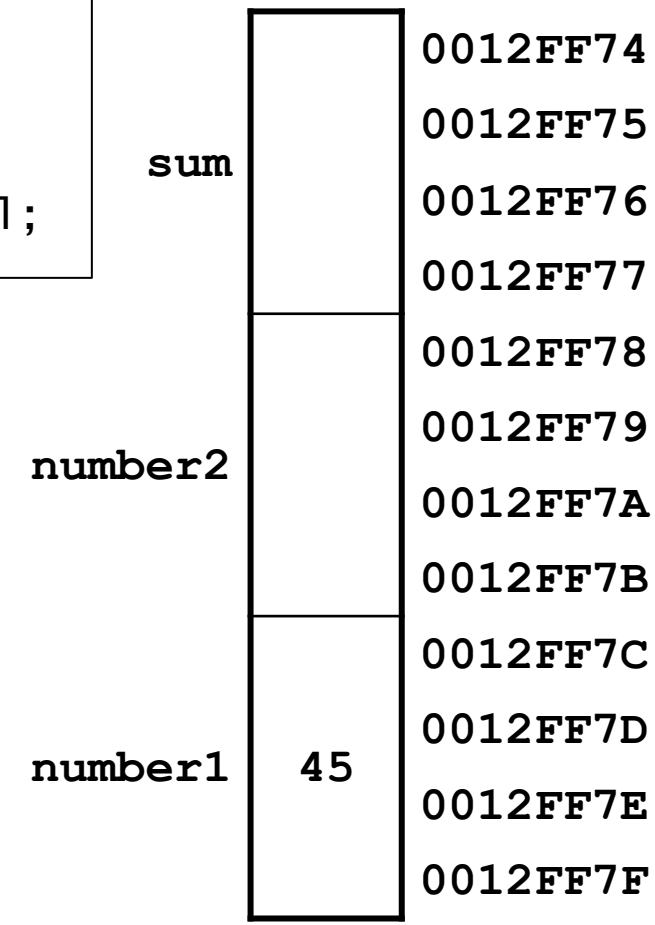
Illustration

```
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



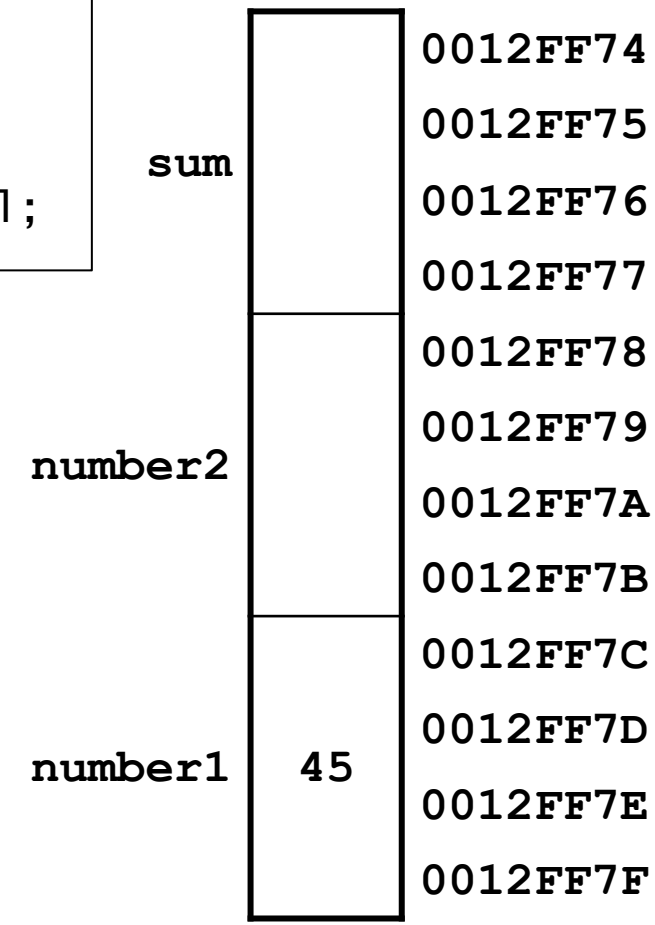
Illustration

```
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



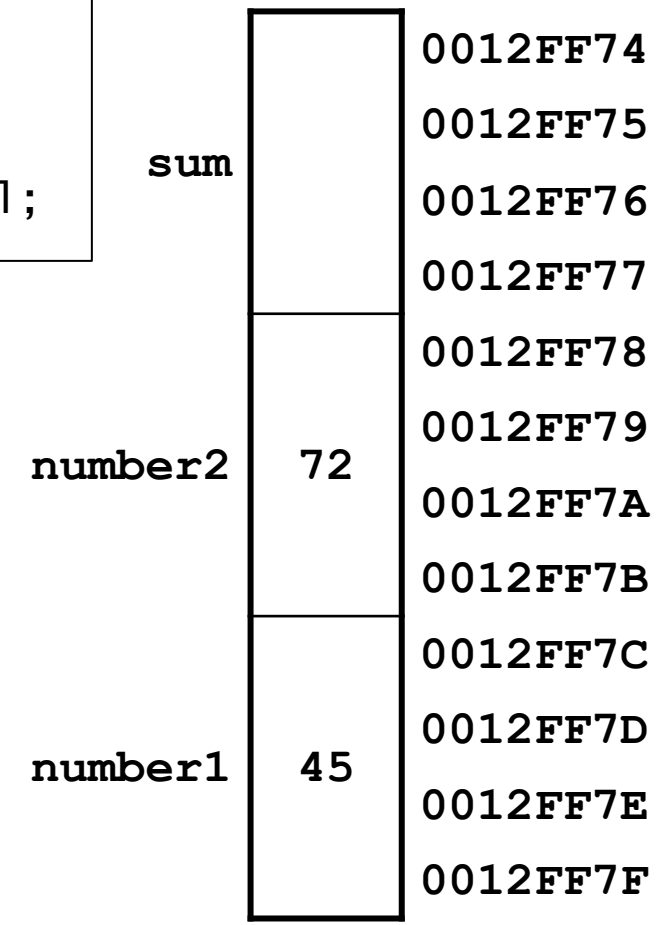
Illustration

```
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



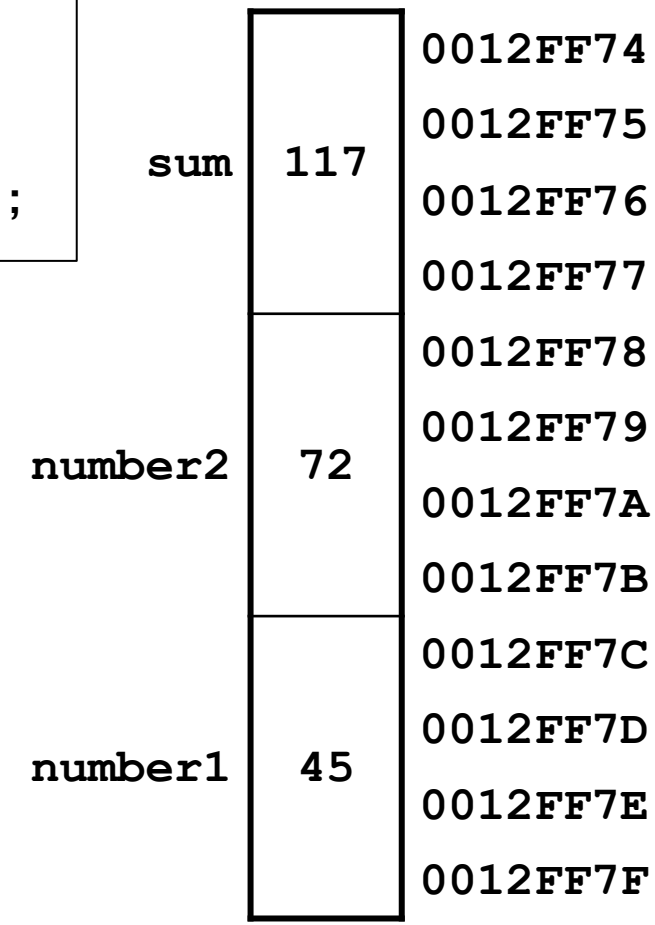
Illustration

```
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



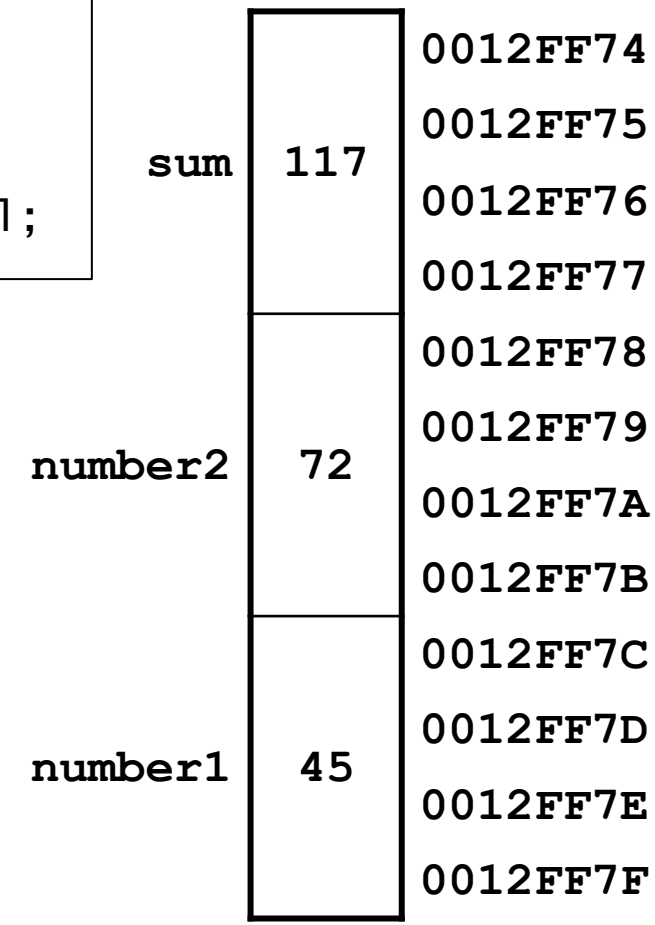
Illustration

```
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



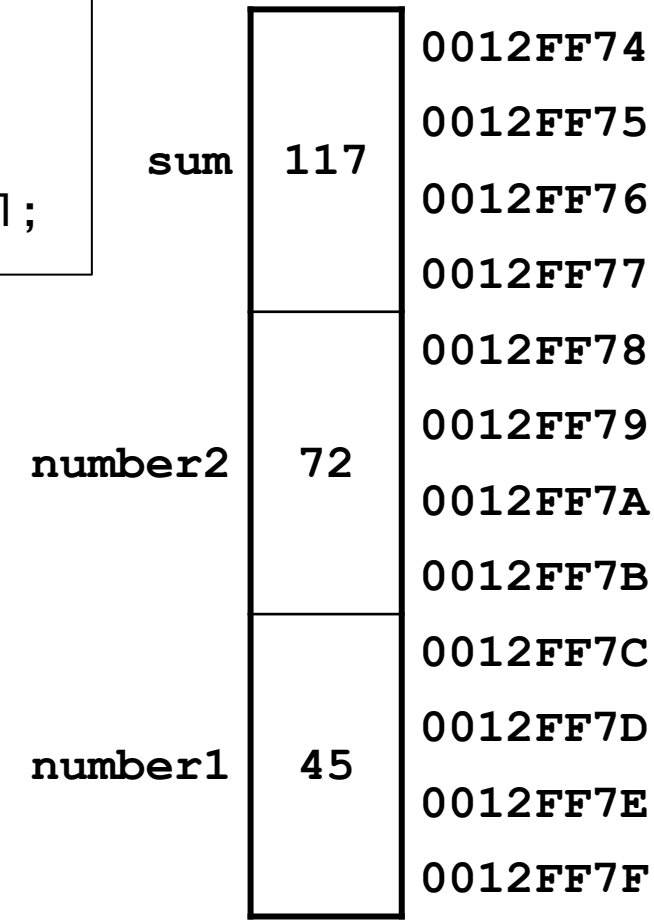
Illustration

```
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



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

- **The arithmetic operators below are all binary operators.**
- **Integer division yields an integer quotient.**
  - Any fractional part in integer division is discarded (i.e., truncated) without rounding.
- **The percent sign (%) is the modulus operator.**
  - The modulus operator provides the remainder after integer division.
  - The modulus operator can be used only with integer operands.

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

Writing Arithmetic Statement

- **Arithmetic expressions in C++ must be entered into the computer in straight-line form.**
- **Expressions such as "a divided by b" must be written as `a / b`, so that all constants, variables and operators appear in a straight line.**
  - $\frac{a}{b}$ is not acceptable
- **Parentheses are used in C++ expressions in the same manner as in algebraic expressions.**
- **For example, to multiply `a` times the quantity `b + c`, we write `a * ( b + c )`.**

Operator Precedence

- **C++ applies the operators in arithmetic expressions in a precise sequence determined by the following rules of operator precedence, which are generally the same as those followed in algebra.**

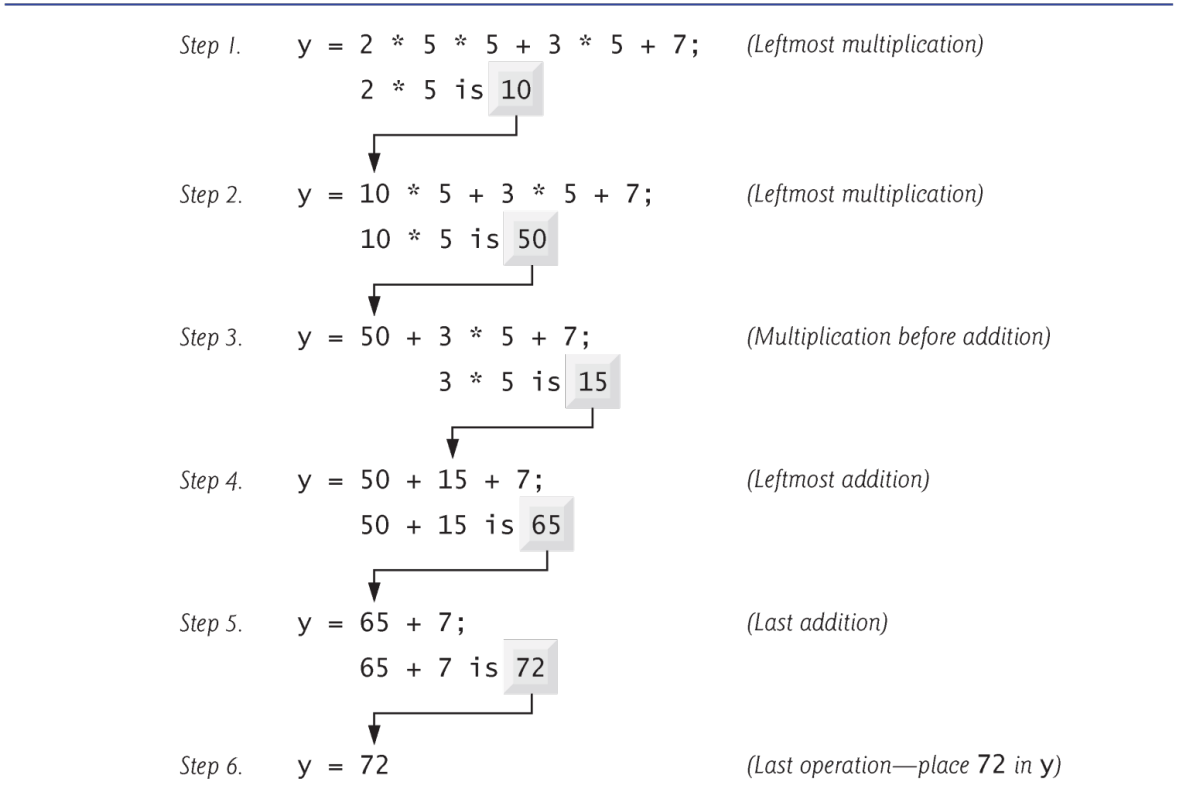
Operator(s)	Operation(s)	Order of evaluation (precedence)
( )	Parentheses	Evaluated first. If the parentheses are nested, the expression in the innermost pair is evaluated first. If there are several pairs of parentheses "on the same level" (i.e., not nested), they're evaluated left to right.
*, /, %	Multiplication, Division, Modulus	Evaluated second. If there are several, they're evaluated left to right.
+ -	Addition Subtraction	Evaluated last. If there are several, they're evaluated left to right.

All Statements are Character by Character

- **There is no arithmetic operator for exponentiation in C++, so x^2 is represented as x * x.**
- **As in algebra, it's acceptable to add unnecessary parentheses to make the expression clearer.**
  - x = (a / b) * c instead of x = a / b * c
  - y = a + (b/c) instead of x = a + b / c
- **These are called redundant parentheses.**
  - Using redundant parentheses in complex arithmetic expressions can make the expressions clearer

Example

- $y = ax^2 + bx + c$
  - y=(a*x*x)+(b*x)+c;
  - a=2, b=3, c=7, x=5



Practice

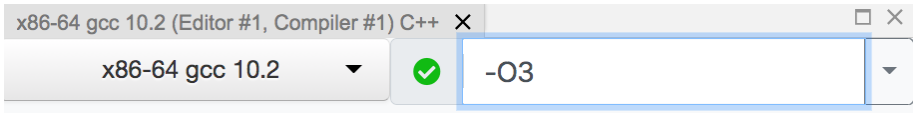
- Evaluate the following expression and print the value of y, assuming all variables are integers

a = 2, b = 3, c = -5, d = 5, e = -12;

$y = (a+b)(c-d) + e^3 + 20$

How many lines of assembly you see from compiler explorer? https://godbolt.org/

What about enabling optimization –O3?



Decision Making: Equality and Relational Operators

- **The if statement allows a program to take alternative action based on whether a condition is true or false.**
- **If the condition is true, the statement in the body of the `if` statement is executed.**
  - Otherwise, the body statement is not executed.
- **Conditions can be formed by using the equality operators and relational operators**
- **The relational operators all have the same precedence level and associate left to right.**
- **The precedence level of equality operators is lower than that of the relational operators. (association is still left to right)**

Decision Making: Equality and Relational Operators

Standard algebraic equality or relational operator	C++ equality or relational operator	Sample C++ condition	Meaning of C++ condition
Relational operators			
>	>	x > y	x is greater than y
<	<	x < y	x is less than y
≥	>=	x >= y	x is greater than or equal to y
≤	<=	x <= y	x is less than or equal to y
Equality operators			
=	==	x == y	x is equal to y
≠	!=	x != y	x is not equal to y

Equality operator “==“ vs Assignment “=“

Confusing the equality operator == with the assignment operator = results in logic errors!

The equality operator should be read “is equal to,” and the assignment operator should be read “gets” or “gets the value of” or “is assigning the value of .”

Some people prefer to read the equality operator as “double equals.”

Example

```
1
2  // Comparing integers using if statements, relational operators
3  // and equality operators.
4  #include <iostream> // allows program to perform input and output
5
6  using std::cout; // program uses cout
7  using std::cin; // program uses cin
8  using std::endl; // program uses endl
9
10 // function main begins program execution
11 int main()
12 {
13    int number1; // first integer to compare
14    int number2; // second integer to compare
15
16    cout << "Enter two integers to compare: "; // prompt user for data
17    cin >> number1 >> number2; // read two integers from user
18
19    if ( number1 == number2 )
20       cout << number1 << " == " << number2 << endl;
21
```

Example (cont'd)

```
22      if ( number1 != number2 )
23         cout << number1 << " != " << number2 << endl;
24
25      if ( number1 < number2 )
26         cout << number1 << " < " << number2 << endl;
27
28      if ( number1 > number2 )
29         cout << number1 << " > " << number2 << endl;
30
31      if ( number1 <= number2 )
32         cout << number1 << " <= " << number2 << endl;
33
34      if ( number1 >= number2 )
35         cout << number1 << " >= " << number2 << endl;
36   } // end function main
```

```
Enter two integers to compare: 3 7
3 != 7
3 < 7
3 <= 7
```

Example (cont'd)

```
Enter two integers to compare: 22 12
22 != 12
22 > 12
22 >= 12
```

```
Enter two integers to compare: 7 7
7 == 7
7 <= 7
7 >= 7
```

Line-by-Line Debrief

```
1
2  // Comparing integers using if statements, relational operators
3  // and equality operators.
4  #include <iostream> // allows program to perform input and output
5
6  using std::cout; // program uses cout
7  using std::cin; // program uses cin
8  using std::endl; // program uses endl
```

- After inserting the `using` declarations, we can write `cout` instead of `std::cout` in the remainder of the program.
  - `std::cin` → `cin`, `std::endl` → `endl`
- Many programmers prefer to use the declaration

  `using namespace std;`

  which enables a program to use all the names in any standard C++ header file (such as `<iostream>`).

=> NEVER DO THIS (BAD BEHAVIOR)

Line-by-Line Debrief

```
13      int number1; // first integer to compare
14      int number2; // second integer to compare
15
16      cout << "Enter two integers to compare: "; // prompt user for data
17      cin >> number1 >> number2; // read two integers from user
```

- This program uses cascaded stream extraction operations to input two integers.

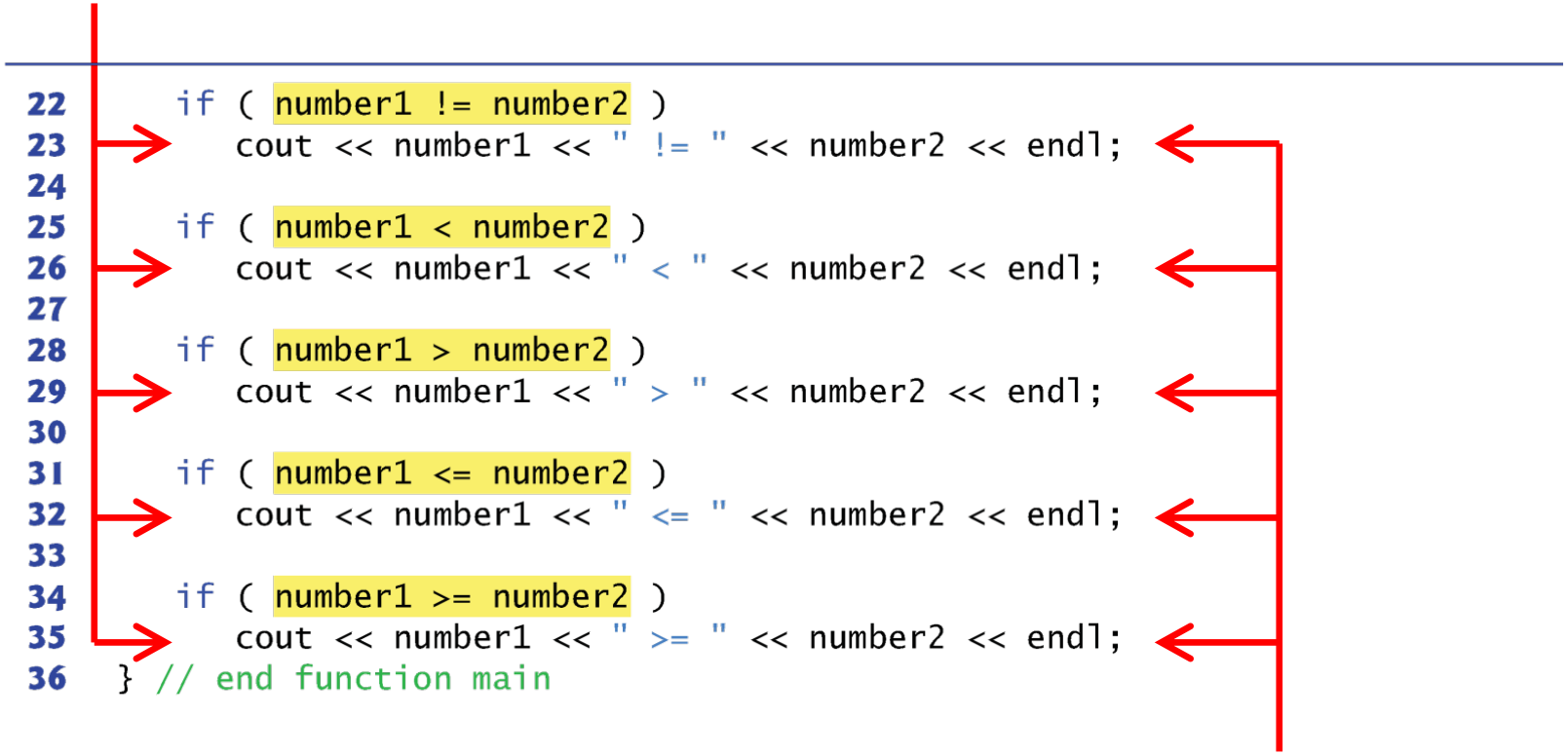
Line-by-Line Debrief

```
22      if ( number1 != number2 )
23         cout << number1 << " != " << number2 << endl;
24
25      if ( number1 < number2 )
26         cout << number1 << " < " << number2 << endl;
27
28      if ( number1 > number2 )
29         cout << number1 << " > " << number2 << endl;
30
31      if ( number1 <= number2 )
32         cout << number1 << " <= " << number2 << endl;
33
34      if ( number1 >= number2 )
```

- Each `if` statement in our example has a single statement in its body and each body statement is indented.
- For `if` statements with multiple-statement bodies, you need to enclose the body statements in a pair of braces, `{ }`, creating what's called a compound statement or a block).

Line-by-Line Debrief

Indent the statements in the body of an if statement to enhance readability



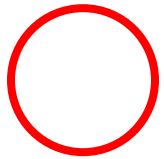
```
22     if ( number1 != number2 )
23        cout << number1 << " != " << number2 << endl;
24
25     if ( number1 < number2 )
26        cout << number1 << " < " << number2 << endl;
27
28     if ( number1 > number2 )
29        cout << number1 << " > " << number2 << endl;
30
31     if ( number1 <= number2 )
32        cout << number1 << " <= " << number2 << endl;
33
34     if ( number1 >= number2 )
35        cout << number1 << " >= " << number2 << endl;
36  } // end function main
```

For readability, there should be no more than one statement per line in a program

Line-by-Line Debrief

Placing a semicolon immediately after the right parenthesis after the condition in an if statement is often a logic error (although not a syntax error). The semicolon causes the body of the if statement to be empty, so the if statement performs no action, regardless of whether or not its condition is true

```
if (number1 < number 2)
   cout << number1 << " < " << number2;
```

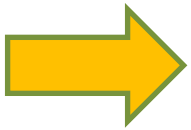


```
if (number1 < number 2) ;
   cout << number1 << " < " << number2;
```

Line-by-line Debrief

- **Statements may be split over several lines and may be spaced according to your preferences.**

```
cout << number1 << " < " << number2;
```



```
cout << number1
     << " < " << number2;
```

- **It's a syntax error to split identifiers, strings (such as `"hello"`) and constants (such as the number `1000`) over several lines.**

Decision-Making Operator Precedence

- The operators are shown top to bottom in decreasing order of precedence.
- All these operators, with the exception of the assignment operator =, associate from left to right.

Operators				Associativity	Type
()				left to right	parentheses
*	/	%		left to right	multiplicative
+	-			left to right	additive
<<	>>			left to right	stream insertion/extraction
<	<=	>	>=	left to right	relational
==	!=			left to right	equality
=				right to left	assignment

Summary

- **C++ input and output**
- **Arithmetic expression**
  - Binary operators, +, -, *, /
  - Operator precedence (e.g., * precede +)
- **Conditionals**
  - Equality operator ==, !=
  - Relational operator <=, <, >, >=