

Lab1: A Simple Program

Fall 2022

Sept. 12, 2022

International Bachelor Program in Informatics

College of Informatics

Yuan Ze University

Rung-Bin Lin



Review of Chapter 2



What you Learn in Chapter 2

- A Program in C++
- Memory Concepts
- Arithmetic
- Decision Making
- Equality and Relational Operators
- Logical Operators
- A Simple Program to Kick Off



A Program you Learned

```
1 // Fig. 2.5: fig02_05.cpp
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 }
```

A program consists of a sequence of statements. A statement must be ended with a semicolon ; (or called statement terminator).

Fig. 2.5 | Addition program that displays the sum of two integers. (Part 1 of 2.)



2.5 Memory Concepts

- ▶ A variable is a location in the computer's memory where a value can be stored for use by a program
- ▶ 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, the value overwrites the previous value in that location; thus, placing a new value into a memory location is said to be **destructive**.
- ▶ When a value is read out of a memory location, the process is **nondestructive**.



Concept of Memory

number1

45

Fig. 2.6 | Memory location showing the name and value of variable number1.

number1

45

number2

72

Fig. 2.7 | Memory locations after storing values for number1 and number2.

number1

45

number2

72

sum

117

Fig. 2.8 | Memory locations after calculating and storing the sum of number1 and number2.



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>

Fig. 2.9 | Arithmetic operators.



Modulus (%) in C++

- $x \% y$ follows the following rule:
 - x and y are integers
 - Let $z = x / y$ and $t = x \% y$, then **$x = z * y + t$**
- Example,
 - Let $x = 11$, $y = 4$, then $z = x / y = 2$, and $t = x \% y = 3$, then check $11 = 2 * 4 + 3$
 - Let $x = -11$, $y = 4$, then $z = x / y = -2$, and $t = x \% y = -3$, then check $-11 = -2 * 4 + (-3)$. However, $11 \% -4 = 3$
 - Let $x = 1$, $y = -4$, then $z = x / y = 0$, and $t = x \% y = 1$, then check $1 = 0 * (-4) + 1$
 - Let $x = -1$, $y = 4$, then $z = x / y = 0$, and $t = x \% y = -1$, then check $-1 = 0 * 4 + -1$
- The above rule can be used to determine the results of $x \% y$.



Precedence of Arith. Operators

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.

Fig. 2.10 | Precedence of arithmetic operators.



2.7 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.
- ▶ If the condition is false, the body statement is not executed.
- ▶ Conditions in **if** statements can be formed using the **equality operators** and **relational operators** summarized in Fig. 2.12.
- ▶ The relational operators all have the same level of precedence and associate left to right.
- ▶ The equality operators both have the same level of precedence, which is lower than that of the relational operators, and associate left to right.



Relational and Equality Operators

Standard algebraic equality or relational operator	C++ equality or relational operator	Sample C++ condition	Meaning of C++ condition
<i>Relational operators</i>			
$>$	$>$	$x > y$	x is greater than y
$<$	$<$	$x < y$	x is less than y
\geq	$>=$	$x >= y$	x is greater than or equal to y
\leq	$<=$	$x <= y$	x is less than or equal to y
<i>Equality operators</i>			
$=$	$==$	$x == y$	x is equal to y
\neq	$!=$	$x != y$	x is not equal to y

No space character in between.

Fig. 2.12 | Equality and relational operators.



Logical Operators

- Used for testing and combining two or more conditions at a single time or complement the evaluation of the given condition.
 - AND (denoted by &&), ex., **x && y**
 - OR (denoted by ||), ex. **x || y**
 - NOT (denoted by !) ex. **!y**



Decision Statements

```
if (x > y) {  
    x=x+y;  
}
```

```
if (x<=y) {  
    y=x+y;  
}  
else {  
    z=x-y;  
}
```

```
if (x>0) {  
    t=y;  
    y=x;  
    x=y;  
}  
else if (t <0 && x<t) {  
    z=x;  
}
```

Here, x, y, z and t each denote a variable. Each section of code enclosed by a pair of { } is called a **block**. Here it is also the **body** of *if* or *else* statement. A block has its own scope.



LAB 1: A Simple Program



LAB 1: A simple program

• Write a C++ program to do the following computations :

- Get three integers from keyboard.
- Multiply the first two integers and store the result
- Divide the first integer by the second integer and store the result if the result is available
- Apply a **modulus operation** on the second and the third integers and store the result if the result is available.
- Add the first, second and third integers and store the result.
- Add the above four results and print the sum if all the above results are available. Otherwise, print “Some results are not available!”.



Input & Output Examples

Sample inputs

0 12 7
12 8 0
34 87 101
54 0 -23
-13 -3 7

Sample outputs

24
Some results are not available!
3267
Some results are not available!
31

1. Get three integers from keyboard = 0 12 7
2. Multiply the first two integers and store the result = 0
3. Divide the first integer by the second integer and store the result if the result is available = 0
4. Apply a modulus operation on the second and the third integers and store the result if the result is available. = 5
5. Add the first, second and third integers and store the result. = 19
6. Add the above four results and print the sum if all the above results are available. Otherwise, print "Some results are not available!". = 24



Code Submission

- Submit your program to the course portal according to the submission rules. For example, name the submitted file S1113234_LAB1.zip for this lab, where 1113234 should be replaced by your student ID number.

