ST0523 Fundamentals of Programming

Topic 2
Operators & Selections (I)

Operations & Selections

- To identify a constant variable
- To identify the various operators for the different data types
- To execute a program using the readline-sync library
- To convert variable to different data types using existing function
- To differentiate the different types of errors - syntax, runtime, and logic errors.
- To implement selection control using one-way if statements.

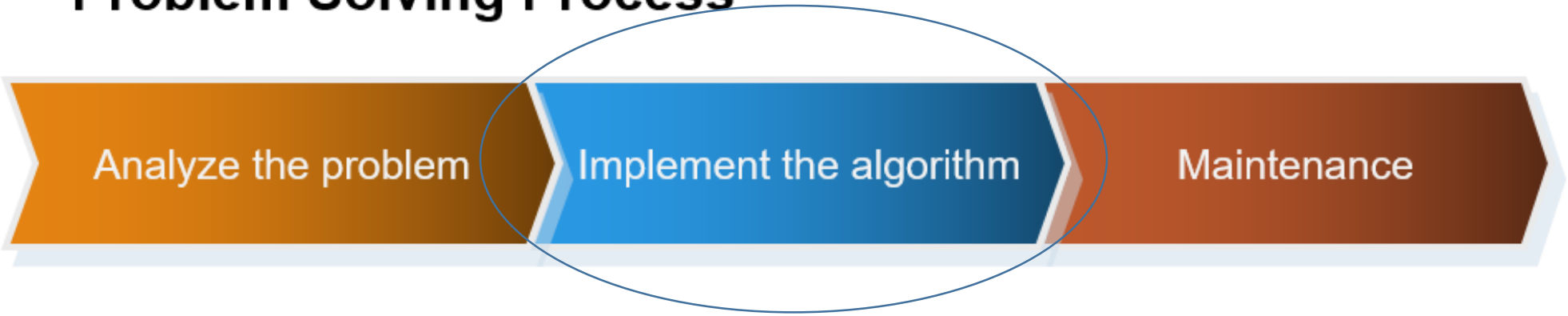
Recap last session……

Recap!

What are the 3 stages of Problem Solving?

Problem Solving Process

- **Problem Solving Process**



- Outline the problem and its requirements
- Design steps (algorithm) to solve the problem

- Implement the algorithm
- Verify that the algorithm works

- Use and modify the program if the problem domain changes

Constants

- Represents permanent data that never changes
  (eg ∏ or pi = 3.14159)

- 3 benefits:
  - Don't have to repeatedly type same value
  - Makes program easier to maintain (Value assigned in a single location)
  - Makes program readable

- Syntax:

```
const variableName = value;
```

  - E.g: **const PI = 3.14159;**

- The **const** keyword ensures that the value cannot be altered/changed.

Class Exercise:

- What is wrong with the program?

Try now and see what error message you will get.

```
const a = 3.1;
a = 6;
```

const → Cannot change value

Numeric Operators +, -, *, /, %

Operator	Meaning	Example	Result
+	Addition	34 + 1	35
-	Subtraction	34.0 – 0.1	33.9
*	Multiplication	300 * 30	9000
/	Division	1.0 / 2.0	0.5
%	Remainder (Modulus)	20 % 3	2
**	Exponentiation	4 ** 3	64

Coding examples:

```
var a = 34;
var b = 1;
var c = a +b;
```

or:

```
var a = 34;
var c = a + 1;
```

or:

```
var c = 34 + 1;
```

Numeric Operators

Operator	Meaning	Exercises	Result
+	Addition	46+6	
-	Subtraction	20.0 – 0.4	
*	Multiplication	(a) 2 * 5 (b)1.5 * 3	
/	Division	(a) 5 / 2 (b) 5.0/2.0	
%	Remainder (Modulus)	(a) 5 % 2 (b) 15%5 (c) 1%6	
**	Exponentiation	(a) 4**2 (b) 3.0 ** 3	

Numeric Operators (Answers)

Operator	Meaning	Exercises	Result
+	Addition	46+6	52
-	Subtraction	20.0 – 0.4	19.6
*	Multiplication	(a) 2 * 5 (b)1.5 * 3	(a) 10 (b) 4.5
/	Division	(a) 5 / 2 (b) 5.0/2.0	(a) 2.5 (b) 2.5
%	Remainder (Modulus)	(a) 5 % 2 (b) 15%5 (c) 1%6	(a) 1 (b) 0 (c) 1
**	Exponentiation	(a) 4**2 (b) 3.0 ** 3	(a) 16 (b)27

Use of Remainder (Modulus) Operator %

>To determine whether a number is even or odd:

- An even number % 2 is always 0
- An odd number % 2 is always 1.

> To determine if a number is divisible by 5:

e.g Result of 67 % 5 is 2, but 65%5 is 0.

So if a number % 5 is always 0 ➔ that number is divisible by 5

Evaluating Expressions

$$\frac{3+4x}{5}-\frac{10(y-5)(a+b+c)}{x}+9\left(\frac{4}{x}+\frac{9+x}{y}\right)$$

is translated to ?

(3 + 4 * x) / 5 – 10 * (y - 5) * (a+b+c) / x + 9 * (4 / x + (9+x) / y)

Shorthand Operators

Operator	Example	Equivalent
+=	i+=8	i = i+8
-=	f-=8.0	f = f-8.0
*=	i*=8	i = i*8
/=	i/=8	i = i/8
%=	i%=8	i = i%8
**=	i**=8	i = i**8

Example:

```
var a = 10;
```

`a -= 3;` ➔ `a = a - 3;` ➔ `a = 10 - 3;` ➔ `a = 7;`

Examples

```
x  +=  x  + a + b;
x = x + x + a+ b;

z -= x*y;
z = z – (x*y);
```

Increment and Decrement Operators

Operator	Name	Description
++var	preincrement by 1 and the increment.	The expression (++var) increments var evaluates to the *new* value in var *after*
var ++	postincrement *original* value	The expression (var++) evaluates to the in var and increments var by 1.
--var	predecrement by 1 and the decrement.	The expression (--var) decrements var evaluates to the *new* value in var *after*
var--	postdecrement	The expression (var--) evaluates to the *original* value in var and decrements var by 1.

- If **++k or --k occurs in an expression**, k is incremented or decremented by 1 **before its** *new* value is used in the expression.
- If **k++ or k-- occurs in an expression**, k is incremented or decremented after its *original* value is used in the expression.

Examples

```
x = 5;
y=4
z=3

z = x          //z become
z = ++x;       //z =6; x also becomes 6
z= x++;        // z =5, x then increase by 1 , becomes 6
```

Example

- To solve a=++b + c++. Let's work through it, given :

a = 1; b = 2; c = 3;

a = ++b + c++;

b is incremented first (b = 3)

Then new value of b is used

Original value of c is used

Then c is incremented. (c = 4)

a = 3 + 3;

```
console.log(a);   //6 is printed
console.log(b);   //3 is printed
console.log(c);   //4 is printed
```

Example

```
i=10;
newNum = 10* i++;
```

Equivalent to →

```
newNum = 10*i;
i = i + 1;
```

```
i=10;
newNum = 10 *(++i);
```

Equivalent to →

```
i = i + 1;
newNum = 10*i;
```

Increment and Decrement Operators

- Using increment and decrement operators makes expressions short, but it also makes them complex and difficult to read.

- Avoid using these operators in expressions that modify multiple variables, or the same variable for multiple times such as this: k = ++i + i

What is the output for each of the followings?

No.	JavaScript codes	What is the output?

1.	a = 2;
	a= a++ + a ;
	console.log(a);	

2.	b = 2;
	b= b + b++ ;
	console.log(b);	

3.	c = 2;
	c= c + ++c ;
	console.log(c);	

4.	d = 2;
	d= ++d + d;
	console.log(d);	

Solutions

```
a = 2;
a = a++ + a; // 2 + 3
console.log(a) // 5

b = 2;
b = b + b++; // 2 + 2
console.log(b) // 4

c = 2;
c = c + ++c; // 2 + 3
console.log(c) // 5

d = 2;
d = ++d + d; // 3 + 3
console.log(d) // 6
```

Comparison Operators

Operator	Name	Example	Boolean result
<	Less than	1 < 2	true
<=	Less than or equal to	1 <= 2	true
>	Greater than	1 > 2	false
>=	Greater than or equal to	1 >= 2	false
==	Equal to	1 == 2	false
!=	Not equal to	1 != 2	true

Boolean Operators

Operator	Name
!	not
&&	and
||	or

Boolean Operators

Truth table definitions of the boolean operators: AND (&&),OR (||), and NOT (!).

a	!a	a	b	a && b	a || b
true	false	false	false	false	false
false	true	false	true	false	true
		true	false	false	true
		true	true	true	true

Boolean data have only two possible values, true and false

!a is true when a is false.

a && b is true only if both a and b are true.

a || b is true if either a or b is true.

Example

Let’s work out the following text & translate to JS:

How to check if 8 can be divided by 2?

Solution: ??

Next how to check if 8 can also be divided by 3?

Solution: ??

Next, can 8 be divided by BOTH 2 & 3?

Next, can 8 be divided by EITHER 2 OR 3?

Possible Solutions?

```
var num = 8;
console.log("Is " + num + " divisible by 2 and 3? " +
  ((num % 2 == 0) && (num % 3 == 0)));

console.log("Is " + num + " divisible by 2 or 3? " +
  ((num % 2 == 0) || (num % 3 == 0)));
```

Operator Precedence

Highest order

var++, var-- (Not applicable)

++var, --var

! (Not)

** (Exponentiation)

*, /, % (Multiplication, Division and Modulus)

+, - (Binary addition and subtraction)

<, <=, >, >= (Comparison)

==, !=; (Equality)

&& (Conditional AND)

|| (Conditional OR)

=, +=, -=, *=, /=, %= (Assignment operator)

Lowest order

IMPORTANT

Operator Precedence and Evaluation Order

- The expression in the parentheses is evaluated first.

- When evaluating an expression without parentheses, the operators are applied according to the precedence rule.

- When 2 operators have the same precedence, the operands are then evaluated from left to right (of the expression), except for ** which is right to left
  - a – b + c – d is equivalent to ((a – b) + c) – d
  - 3%2*4 is equivalent to (3%2) * 4 , it is NOT equivalent to 3%(2*4)
  - 2**3**2 results in 512, NOT 64

Operator Precedence and Evaluation Order

- The expression 3 + 4 * 4 > 5 * (4 + 3) - 1 is evaluated as follows:

```
3 + 4 * 4 > 5 * (4 + 3) - 1
3 + 4 * 4 > 5 * 7 - 1
3 + 16 > 5 * 7 - 1
3 + 16 > 35 - 1
19 > 35 - 1
19 > 34
false
```

(1) inside parentheses first

(2) multiplication

(3) multiplication

(4) addition

(5) subtraction

(6) greater than

String Data - Type

- To represent a string of characters, use the data type called String.

- Any value that is enclosed with either double quotation marks (" ") or single quotation marks (') is treated as a String.

- Declaring and assigning Strings

```
var message = "Hello class!";
message = 'Bye class!';
```

String Concatenation

```
// Three strings are concatenated
var message = "Welcome " + "to " + "FOP! ";
```

message = “Welcome to FOP!”

```
// String Chapter is concatenated with number 2
var  s = "Chapter" + 2;
```

s = “Chapter2”

```
// String hi is concatenated with Boolean, true
var s1 = 'hi ' + !false;
```

s1 = “hi true”

Data Types

Primitive Types	Reference Types
1. String	1. Array
2. Number (integer & floating point)	2. Object
3. Boolean	3. Function
4. Null	
5. Undefined	

Later chapters

Obtaining Input

- Any information or data that is sent to a computer for processing is considered input. There are many types of input in programming such as keyboard input, or data from external files/database, data from barcode reader, data from RFID, etc.

- For now, we will only be focusing on how to obtain keyboard input (capture/process what is keyed into our program) from the user.

- To allow our JavaScript program to read user's keyboard input, you will need to install an additional module, called **readline-sync**.

Obtaining Input

- To install this module, we need to run the following command in the integrated terminal:

<u>npm install readline-sync</u>

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Users\CK\Documents\tmp> npm install readline-sync
```

Getting User's Keyboard Input

- 1. Create a readline-sync object

```
var input = require('readline-sync');
```

Declares that input is an object needed to capture keyboard input

- 2. Use the methods question() to obtain user's input in a form of a String

Getting User's Keyboard Input

```
var input = require('readline-sync');

var userName = input.question('May I have your name? ');
console.log('Hi ' + userName + '!');

var favFood = input.question('What is your favourite food? ');
console.log('Oh, ' + userName + ' loves ' + favFood + '!');
```

Program is paused here for user's input

```
May I have your name? C.K.
Hi C.K!
What is your favourite Food? Satay
Oh, C.K. loves Satay!
```

User's Keyboard Input

Exercise.....start Visual Studio

- Run the following command to install this module:
- Make sure you are in the correct folder

npm install readline-sync

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Users\CK\Documents\tmp> npm install readline-sync
```

Converting Strings to Numbers

- The input returned from the keyboard is a string.
- To obtain the input as a number, you have to convert a string into a number.
- To convert a string into an integer (whole number), you can use the built-in **parseInt** function as follows:

var intValue = **parseInt** (intString);

where intString is a numeric string such as “123”.

- Similarly, To convert a string into an float, you can use the built-in **parseFloat** function.

Example

- Parsing String values into numerical format

```
var adminNum = parseInt("123456");

var salary = parseFloat("1234.5");
```

Example

```
var input = require('readline-sync');

var salary, userInput;

userInput = input.question("Please enter your salary: ");
```

Sample output:

```
Please enter your salary:
```

User enters 1234.50 and press enter

```
Please enter your salary: 1234.50
```

userInput = “1234.50”

This is a string value but we need a floating point number !!

Example

```
var input = require('readline-sync');

var salary, userInput;

userInput = input.question("Please enter your salary: ");
salary = parseFloat(userInput);
```

Converts a string value into a floating-point number

Therefore, assuming user enters 1234.50,
userInput = “1234.50”
salary = 1234.5

Capturing numeric value directly

- Use .questionInt() or .questionFloat() instead of .question() when reading input.

- This will convert to numeric and if a non-numeric is entered, it will reprompt.

Let’s try on Visual Studio now!!

Exercises:

Assuming you have a variable year as an integer value :

a) Write JS code using a % operator to check if year is divisible by 4.

b) Write JS code using a % operator to check if year is not by divisible by 100.

c) Write JS code using a % operator again to check if year is divisible by 400.

Exercises:

Assuming you have a variable year as an integer value:

a) Write JS code using a % operator to check if year is divisible by 4.

Ans : (year % 4 == 0) must be true

b) Write JS code using a % operator to check if year is not by divisible by 100

Ans : (year % 100 != 0) must be true

c) Write JS code using a % operator again to check if year is divisible by 400.

Ans : (year % 400 == 0) must be true

Example:
Let's combine them to test for leap Year

- Assuming a program first prompts the user to enter a year as an integer value and checks if it is a leap year.

- Given a year is a leap year if it is divisible by 4 but not by 100, or it is divisible by 400.

```
((year % 4 == 0) && (year % 100 != 0)) || (year % 400 == 0))
```

Example: Leap Year

```
var input = require('readline-sync');

var userInput, intYear, isLeapYear;

//prompt the user to enter a year
userInput = input.question("Enter a year: ");

//Convert the string entered into an integer value
intYear = parseInt(userInput)

//Check if the year is a leap year
isLeapYear = ((intYear % 4 == 0) && (intYear % 100 != 0)) ||
             (intYear % 400 == 0);

//Display the result
console.log(intYear + " is a leap year? " + isLeapYear);
```

Converting Strings to Numbers

- However, in some circumstances these conversions might not be necessary in JavaScript.
- JavaScript is capable of doing certain calculations with string i.e. multiplication, subtraction and division.
- The result of the calculation is of type “number”.
- Unfortunately, this auto-conversion **does NOT** work with ‘addition’. The + symbol serves 2 purposes i.e. concatenation and addition. Whenever one of the operands is a string, the result of the operation will be a string too.
- Therefore, it is advisable to always use conversion function whenever you accept the keyboard input.

Converting Strings to Numbers

```
var input = require('readline-sync');

x = "5";
num1 = x * 2; //10
num2 = x / 2; //2.5
num3 = x - 2; //3
num4 = x + 2; //"52"

console.log(typeof num1); //number
console.log(typeof num2); //number
console.log(typeof num3); //number
console.log(typeof num4); //string
```

Error ?

Programming Errors

- Syntax Errors
  - Detected by the compiler
- Runtime Errors
  - Causes the program to abort
- Logic Errors
  - Produces incorrect result

Syntax Errors

```
var x = 20;
var y = x * 2;

Console.log(x + y);
```

Programming Errors

- Syntax Errors
  - Detected by the compiler
- Runtime Errors
  - Causes the program to abort
- Logic Errors
  - Produces incorrect result

Runtime Errors

```
var i = 10/0;
```

Programming Errors

- Syntax Errors
  - Detected by the compiler
- Runtime Errors
  - Causes the program to abort
- Logic Errors
  - Produces incorrect result

Logic Errors

```
// Add num1 to num2
var num1 = 3,
var num2 = 2;
num2 += num1 + num2;

console.log("Answer" + num2);
```

Selection Statement (I)

What if …

Selection Statements

- Most of the time when we write code, we would want the code to perform different actions for different decisions.

- We can use selection statements in our code to do this.
- {… } is used to form a block that groups components of a program

- Syntax:

```
if ( condition ) {
// code to be executed if the condition is true
}
```

Example

- Since mark is more than 80, the condition is **true**, so “Your grade is A” will be displayed in the output.

```
var mark = 90;
if ( mark >= 80 ) {
      grade = ‘A’;
}
console.log( ‘Your grade is ’ + grade );
```

Example

- Now, mark is less than 80 resulting the condition is **false**, hence does not execute line grade= ‘A’;
- There is no output displayed but resulted in ReferenceError: grade is not defined

```
var mark = 79;
if ( mark >= 80 ) {
    grade = ‘A’;
    console.log(“congratulations”);
}
console.log( ‘Your grade is ’ + grade );
```

Take note!

- Adding a semicolon at the end of an if clause is a common mistake.

```
if ( mark >= 80 );
{
      grade = 'A';
}
console.log( 'Your grade is ' + grade );
```

WRONG !!

- This mistake is hard to find, because it is not a compilation error or a runtime error, it is a logic error.
- This error often occurs when you use the next-line block style.

What have we learnt?

- To identify a constant variable
- To identify the various operators for the different data types
- To execute a program using the readline-sync library
- To differentiate the different types of errors - syntax, runtime, and logic errors.
- To implement selection control using one-way if statements.

Great job everyone !!

SP SCHOOL OF Computing