COSC 122
Computer Fluency

Algorithmic Thinking

Dr. Firas Moosvi

Key Points

1) There are five essential properties for algorithms.

2) The five basic steps of development are a general approach for solving problems using a computer.

Algorithm

An ***algorithm*** is a precise, systematic method for producing a specified result.

We use algorithms all the time to complete tasks.

A common example is following assembly directions (as with IKEA assembly) or using a **recipe**. Simpler examples include how to perform arithmetic or look up a person's name in a list.

Some algorithms are so simple or ingrained that we do not consciously remember the steps. However, precision is required when communicating the algorithm to others.

Five Essential Properties of Algorithms

1) *Inputs specified* – must specify the **type, amount,** and **form** of **data** to be used during the algorithm

2) *Outputs specified* – must describe the result of the algorithm (it is possible to have no output).

3) *Precision* – specify precisely the **sequence of steps** to be performed including how to handle errors.

4) *Reasonable Operations* - The operations are doable.

5) *Finite* – The algorithm must eventually stop (terminate).

CQ 7.1- Five Essential Properties of Algorithms

Question: The algorithm on the shampoo bottle says: "Apply shampoo. Lather. Rinse. **Repeat**." Which one of the five essential properties does this algorithm not meet?

A) inputs specified

B) outputs specified

C) precision

D) reasonable operations

E) finiteness

Group Discussions

Provide an algorithm for brushing your teeth.

Specifying Algorithms using Language

An algorithm must be written using a **language understood** by both the *writer* of the algorithm and the *reader* who will use it.

For computer algorithms, the writer is a human programmer, and the reader is the computer. Natural languages like English are easy for humans, but are ambiguous and often require domain knowledge and context. Instead, we **use precise programming languages** (e.g. HTML/JavaScript).

A common barrier for students with programming is that the language is unfamiliar and that the computer requires precision. Remember, **have patience!**

◆ Learning a computer language is similar to learning a foreign language like Spanish.

★ *The 5 Basic Steps of Software Development*

1) Specification

- Determine the scope of your problem and **what** you want your program to do.

2) Design

- Determine the structures and algorithms necessary (**how**) to solve your problem at a high-level of abstraction.

3) Implementation

- Start implementing your algorithms/structures on the computer.

4) Testing, Execution, and Debugging

- Test your program on various data sets and fix any problems.

5) Maintenance

- Over time, modify your program as necessary to handle new data or more complicated problems.

CQ 7.2- Software Development Steps

Question: Which of the 5 steps is most often the cause of projects being unsuccessful?

A) Specification

B) Design

C) Implementation

D) Testing

E) Maintenance

Programming - Art or Science?

There is a debate whether programming is an art or a science.

- It is similar to a **science** because **algorithms** and data structures can **be analyzed for performance** and chosen with respect to their relevance to a particular problem.

- It is like an **art** or craft because **skills of programmers vary** widely, even with similar training, and the **"best" solution to the problem is often open to debate**.

In computer science, we teach you the "science" component.

- We want you to understand the **choices you make** and the reasons for them.

- However, students will all have different **natural abilities and talents** with respect to programming.

  If it is easy or natural for you, great! If not, then fall back on the science and the techniques we teach to help you. And PRACTICE as much as you can.

CQ 7.3- Programming: Art or Science?

Question: What do YOU think programming is most like?

A) Art (creativity)

B) Science (experimentation)

C) Engineering (construction)

D) All of the above

E) Other or none of the above

CQ 7.4- Programming: Experience

Question: What is your programming experience?

A) I have never programmed before.

B) I have wrote instructions, recipes, manuals, or other precise information before (maybe not electronic).

C) I have wrote HTML or created web sites before this class.

D) I have experimented on my own with programming.

E) I have taken a programming class in high school or university.

Conclusion

An ***algorithm*** is a precise sequence of steps to produce a result that is encoded in a language to produce a ***program***.

The five essential properties of an algorithm are:

- ◆ Inputs specified
- ◆ Output specified
- ◆ Precision
- ◆ Reasonable operations
- ◆ Finite

Following the five basic steps for developing solutions to problems on a computer will make you more successful and efficient while programming.

Objectives

- Define: algorithm, program
- List and explain the five essential properties of an algorithm.
- Explain why special programming languages are used to communicate algorithms to the computer instead of English.
- List and explain the five basic steps of software development.

COSC 122
Computer Fluency

Programming Basics

Dr. Firas Moosvi

Key Points

1) We will learn JavaScript to write instructions for the computer.
   - The fundamental programming concepts apply to all languages.

2) The key programming concepts covered:
   - variables, values, and locations
   - initialization and assignment
   - expressions

History: The First Programmers

Did you know that the first programmers were almost all women?

◆ Women worked on the first computer - the ENIAC (Electronic Numerical Integrator and Calculator) developed for the US Army in 1946 by J. Eckert and John Mauchley.

◆ These women were recruited from the ranks of "computers", humans that used mechanical calculators to solve complex math problems before the invention of computers.

◆ These pioneer programmers laid the foundation of many of the original ideas including compilers and programming languages.

Introduction to Programming

Remember that an *algorithm* is a precise sequence of steps to produce a result. A *program* is an encoding of an algorithm in a *language* to solve a particular problem.

There are numerous languages that programmers can use to specify instructions. Each language has its different features, benefits, and usefulness.

The language we will use is called JavaScript. However, our focus will be understanding the primary programming concepts that apply to all languages.

Introduction to JavaScript

JavaScript is a *scripting* language used primarily for web pages.

- JavaScript was developed in 1995 and released in the Netscape web browser (since renamed to Mozilla Firefox).

- JavaScript is standardized and supported by most browsers.

Despite the name, JavaScript is not related to Java, although its syntax is similar to other languages like C, C++, and Java.

- There are some major differences between JavaScript and Java that will not concern us here.

- Aside: The term *scripting* means the language is interpreted (processed when needed) instead of compiled (converted to machine language directly). The difference is irrelevant to us.

Some Quotes

If you can't write it down in English, you can't code it.
-- Peter Halpern

If you lie to the computer, it will get you.
-- Peter Farrar

Demonstration
Do you want fries with that?

This example program is for demonstrating a JavaScript program that calculates the total cost of a fast food order.

Don't worry now about understanding the code!

Inputs:

- `burger` – may be "none", "hamburger", or "cheeseburger"
- `fries` – may be "none", "small", or "large"
- `drink` – may be "none", "small", or "large"

Output:

- the `total` in dollars of the order including tax (7%)

Fast Food Example Code

```
var total;
var taxRate = 0.07;

total = 0;

if (burger == "hamburger" || burger == "cheeseburger")
    total = 0.99;

if (fries == "small")
    total = total + 1.19;
if (fries == "large")
    total = total + 1.79;

if (drink == "small")
    total = total + 1.49;
else if (drink == "large")
    total = total + 1.89;

total = total + total * taxRate;
```

Declare and initialize variables

← Assignment

Decision using IF

Flow of Execution
-Start at first statement at top and proceed down executing each statement.
-For `if` statement only execute one of the possibilities if condition is true otherwise go to next statement after `if`.

Expression

Creating and Running a JavaScript Program

We run JavaScript programs within a web browser.

This means several things:

- ◆ 1) The file that stores the program will be an HTML document. It should have a name like `myProgram.html`.

- ◆ 2) The JavaScript program is part of the HTML file.

- ◆ 3) Edit the document using a text editor. Test the document by opening it in Internet Explorer, FireFox, Chrome, Safari, …

Hello World!

```
<html>
<head>
<title>HelloWorld using JavaScript</title>
</head>
<body>
<h1>
```

<script> tag
indicating code

```
   <script type="text/javascript">
      document.write("Hello, world!");
   </script>
```

} JavaScript code

```
</h1>
</body>
</html>
```

document is HTML document
document.write() puts that text into the document
at this location

The + Operator

```
<html>
<body>
<h1>
    <script type="text/javascript">
        document.write(3 + 5);
        document.write("A" + "B");
        document.write("A" + 3);
        document.write("A" + 3 + 2);
        document.write("A" + (3+2) );
    </script>
</h1>
</body>
</html>
```

Output is **8**

Output is **AB**

Output is **A3**

Output is **A32**

Output is **A5**

General Syntax Rules

A program is a list of statements (instructions).

PRIMARY RULE: Every statement must be terminated by a semi-colon ";".

- Note the statement terminator character varies by language.

Other rules:

- You may have multiple statements on a line as long as each ends with a semi-colon.

- A statement may cross multiple lines.

JavaScript: Basic Rules

To program in JavaScript you must follow a set of rules for specifying your commands. This set of rules is called a *syntax*.

◆ Just like any other language, there are rules that you must follow if you are to communicate correctly and precisely.

Important general rules of JavaScript syntax:

◆ JavaScript is *case-sensitive*.

　　Main() is not the same as main() or MAIN()

◆ JavaScript accepts *free-form layout*.

　　Spaces and line breaks are not important except to separate words.

　　You can have as many words as you want on each line or spread them across multiple lines.

　　However, you should be consistent and make your code easy to read.

Comments

Comments are used by the programmer to document and explain the code. Comments are ignored by the computer.

There are two choices for commenting:

- 1) One line comment: put "`//`" before the comment and any characters to the end of line are ignored by the computer.

- 2) Multiple line comment: put "`/*`" at the start of the comment and "`*/`" at the end of the comment. The computer ignores everything between the start and end comment indicators.

Example:

```
/* This is a multiple line
      comment.
With many lines. */

// Single line comment
// Single line comment again
d = 5.0; // Comment after code
```

Hello World! Program with Comments

```
<html>
<head>
<title>HelloWorld using JavaScript</title>
</head>

<body>

<h1>

    <script type="text/javascript">
      //Greet the world!          ⟵——————  This is a comment
      document.write("Hello, world!");
    </script>

</h1>

</body>
</html>
```

JavaScript Basics

Question: what is the output of this JavaScript statement?
(assume all HTML tags are properly used)

```
document.write("Hi" + " There" + 6); //Bye
```

A) Hi There

B) Hi There 6

C) Hi There 6 Bye

D) Error

JavaScript Basics (2)

Question: what is the output of this JavaScript statement?
(assume all HTML tags are properly used)

```
document.write(6 + 7 + " km"); //6+7=13
```

A) 67 km
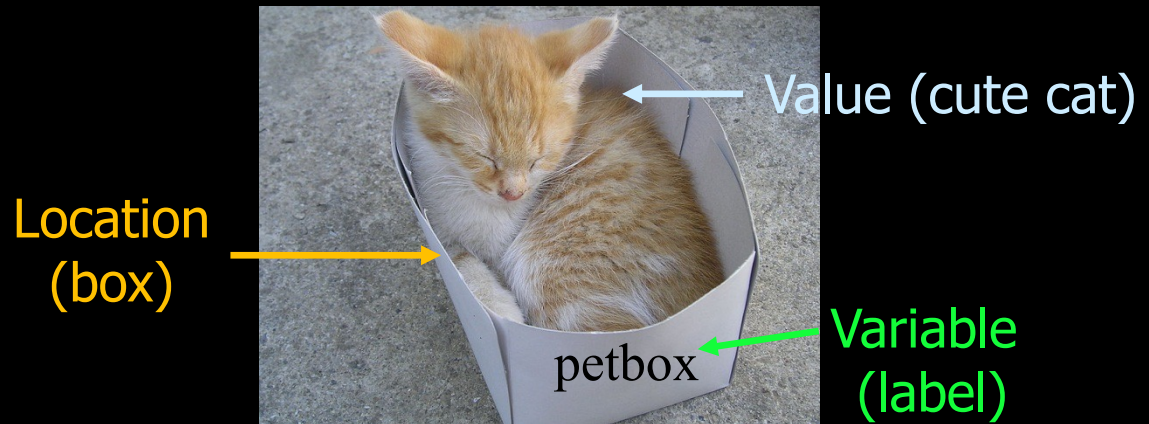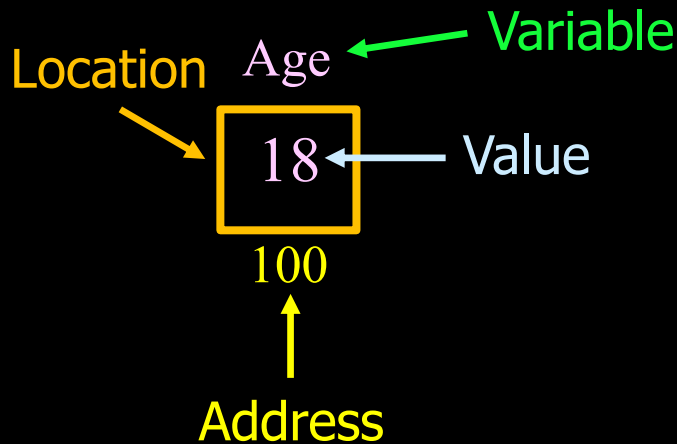
B) 13 km

C) 67 km 6+7=13

D) 13 km 6+7=13

Values, Variables, and Locations

A *value* is a data item that is manipulated by the computer.

A *variable* is the name that the programmer uses to refer to a location in memory.

A *location* has an address in memory and stores a value.

Variable

Age

Location

18

Value

100

Address

Value (cute cat)

Location
(box)

Variable
(label)

petbox

IMPORTANT: The *value* at a given location in memory (named using a variable name) can change using initialization or assignment.

Values, Variables, and Locations Example

We want to store a number that represents the total order value.
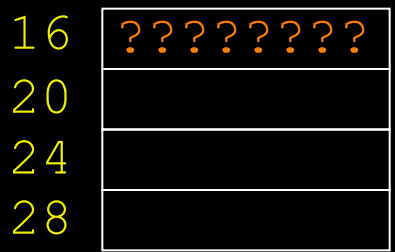
Step #1: Declare the variable by giving it a name

```
var total;
```

◆ The computer allocates space for the variable in memory (at some memory address). Every time we give the name `total`, the computer knows what data item we mean.

Variable Name Lookup Table

Memory

Name	Location	Type
total	16	number

16	????????
20	
24	
28	

Values, Variables, and Locations Example (2)

Step #2: Initialize the variable to have a starting value

- ◆ If you do not initialize your variable to a starting value when you first declare it, the value of the variable is *undefined*.

Example:

```
total = 1;
```

Variable Name Lookup Table

Name	Location	Type
total	16	number

Memory

16	1
20	
24	
28	

Values, Variables, and Locations Example (3)

Step #3: Value stored in location can be changed throughout the program to whatever we want using *assignment* ("**=**" symbol).

```
total = total * 5 + 20;
```

Variable Name Lookup Table   Memory

Name	Location	Type
total	16	number

16	25
20	
24	
28	

Variable Rules

Variables are also called identifiers. An *identifier* is a name that *must contains only letters*, numbers, underscore ('_') and $.

The keyword `var` is used to declare to the computer that you want a variable created. This declaration is a type of *statement*.

Rules:

- Every variable used in a program must be declared.

- Variables can be declared anywhere in the program, but usually should be declared right at the start.

- Variable names **ARE** case-sensitive. Numbers are allowed (but not at the start). Only other symbol allowed is underscore ('_');

- Beware of declaring two variables with the same name.

- The syntax of the language allows you to declare and initialize multiple variables in the same statement:

```
var total = 0, taxRate = 0.07;
```

Aside: Good Variable Names

As a programmer you have flexibility on the names that you assign to your variables.

◆ However, names should be meaningful and explain how the variable is actually used in your program.

Example:
```
var t = 0;
var total = 0;
```

Avoid naming variables as reserved words. A *reserved word* is a string that has special meaning in the language.

◆ e.g. `if, var, else`

Variables – Basic Terminology

Question: Of the following three terms, what is most like a *box*?

A) value

B) variable

C) location

Variables - Definitions

Question: Which of the following statements is correct?

A) The location of a variable may change during the program.

B) The name of a variable may change during the program.

C) The value of a variable may change during the program.

Variables – Correct Variable Name

Question: Which of the following is a valid JavaScript variable?

A) `aBCde123`

B) `123test`

C) `t_e_s_t!`

Variable Types

A variable has a *name* for a data item and a *type*.

- ◆ JavaScript is different than most languages because you do not have to tell the computer what type the variable is when you declare it. The variable can store any type (although it is not recommended to change types).

The data types that we will use are:

- ◆ numbers – both integers and float/doubles
- ◆ strings – sequences of characters
- ◆ Boolean – true or false

Strings

Strings are sequences of characters that are surrounded by either single or double quotes.

Example:

```
var personName = "Ramon Lawrence";
personName = "Joe Smith";
```

Question: What is the difference between these two statements?

Rules for Strings in JavaScript

String rules:

◆ Must be surrounded by single or double quotes.

◆ Can contain most characters except enter, backspace, tab, and backslash.

These special characters must be escaped by using an initial "\".

e.g. \n – new line, \' – single quote, \\ - backslash, \" – double quote

◆ Double quoted strings can contain single quoted strings and vice versa.

◆ Any number of characters is allowed.

◆ The minimum number of characters is zero " ", which is called the *empty string*.

◆ String *literals* (values) have the quotation marks removed when displayed.

Practice Questions

1) Write the statements to create two variables: one called `hourlyRate` and the other called `hoursWorked`.

     Aside: Are the following variable names valid or invalid:

```
var A;
var A123;
var 123A;
var aReallyLongName;
```

2) Using your statements from question #1, write the code to calculate and store a person's salary by multiplying their `hoursWorked` times their `hourlyRate`.

3) Create a string variable that has an initial value of `'Joe'`.

4) Print the string variable from question #3 followed by the salary from question #2. e.g. `Joe's salary is $54000`.

The Assignment Statement

An *assignment statement* changes the value of a variable.

The variable on the left-hand side of the **=** is assigned the value from the right-hand side.

The value may be changed to a constant, to the result of an expression, or to be the same as another variable.

The values of any variables used in the expression are always their values before the start of the execution of the assignment.

Examples:

```
var A, B;

A = 5;
B = 10;
A = 10 + 6 / 2;
B = A;
A = 2*B + A - 5;
```

Question: What are the values of A and B?

Expressions

An *expression* is a sequence of operands and operators that yield a result. An expression contains:

- ◆ *operands* - the data items being manipulated in the calculation

  e.g. 5, "Hello, World", myDouble

- ◆ *operators* - the operations performed on the operands

  e.g. +, -, /, *, % (modulus - remainder after integer division)

An operator can be:

- ◆ *unary* - applies to only one operand

  e.g. d = - 3.5;   // "-" is a unary operator, 3.5 is the operand

- ◆ *binary* - applies to two operands

  e.g d = e * 5.0;   // "*" is binary operator, e and 5.0 are operands

The Remainder Operator

Remainder operator **(%)** yields the remainder after division (e.g. 5 % 2 yields 1)

Exercise: Show the result of the following remainders.
- **14 % 6**   **// 2**
- **3 % 0**   **// Runtime error. Can't divide by zero**
- **34 % -5**   **// 4**
- **-34 % 5**   **// -4**
- **-34 % -5**   **// -4**
- **5 % 1**   **// 0**
- **1 % 5**   **// 1**

Expressions - Operator Precedence

Each operator has its own priority similar to their priority in regular math expressions:

- ◆ 1) Any expression in parentheses is evaluated first starting with the inner most nesting of parentheses.
- ◆ 2) Unary + and unary - have the next highest priorities.
- ◆ 3) Multiplication and division (*, /, %) are next.
- ◆ 4) Addition and subtraction (+,-) are then evaluated.

What is the result of:

$$20 – (\ (4 + 5) – (\ 3 * (\ 6 – 2\ )\ )\ ) * 4$$

String Operators: Concatenation

The ***concatenation operator*** is used to combine two strings into a single string. The notation is a plus sign '**+**'.

Example:

```
var string1 = "Hello";
var string2 = " World!";
var result = string1 + string2; //result = "Hello World!"
```

The plus sign is used for addition, but it makes sense as the symbol for string concatenation as well.

Using the same symbol as a operator in multiple different ways is called ***operator overloading***.

Assignment

Question: What are the values of `A` and `B` after this code?

```
var A, B;

A = 2;
B = 4;
A = B + B / A;
B = A * 5 + 3 * 2;
```

A) `A = 6, B = 36`

B) `A = 4, B = 26`

C) `A = 6, B = 66`

String Concatentation

Question: What is the value of result after this code?

```
var st1="Joe", st2="Smith";

var result = st1 + st2;
```

A) `"Joe Smith"`

B) `"JoeSmith"`

String Concatentation (2)

Question: What is the result after this code?

```
var st1="123", st2="456";

var result = st1 + st2;
```

A) `579`

B) `"579"`

C) `"123456"`

Getting Input into a JavaScript Program

There are two ways to get input from the user into your program:

- ◆ 1) Make the user fill in form fields and get the value of those fields when the user clicks a button.

    We will see how to do this later.

- ◆ 2) Prompt the user with a separate window asking them for a value.

Getting Input Using JavaScript Code

```
<html>
<head>
<title>Prompt for a Value using JavaScript</title>
</head>

<body>
<h1>
    <script type="text/javascript">
        var val = window.prompt("Enter a value: ");
        document.write(val);
    </script>
</h1>
</body>
</html>
```

Prompt for value from user

write out value retrieved

Outputting from a JavaScript Program

There are three ways to output information to the user:

- ◆ 1) Have your code set the value of a form field.

- ◆ 2) Have your code write out text directly into the HTML document.

- ◆ 3) Open an alert output window to the user with a message.

Outputting Data from JavaScript Code

```
<html>
<head>
<title>Display a Value using an Alert Window</title>
</head>

<body>
    <script type="text/javascript">
        var val = window.prompt("Enter a value: ");
        window.alert("You said: "+val);
    </script>
</body>
</html>
```
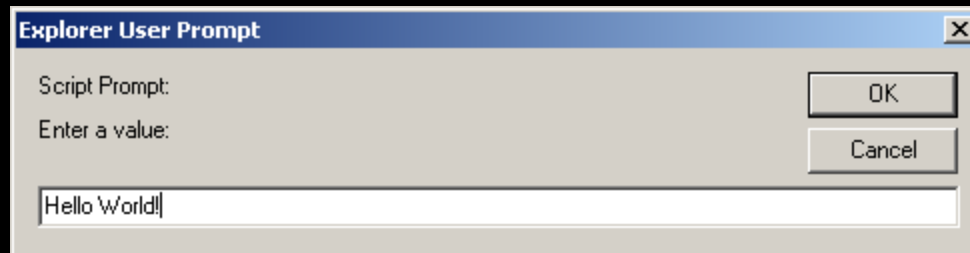
Prompt for value from user

Open up new window with message and value that the user just entered.

Prompt and Output Example

Prompt window:

Explorer User Prompt

Script Prompt:
Enter a value:

OK

Cancel

| Hello World!|

Alert (output) window:

Microsoft Internet Explorer

⚠️ You said: Hello World!

OK

Input/Output Question

Question: Assume the user typed in **10** when prompted. What is shown in the HTML document after this code?

```
var val = window.prompt("Enter a value: ");
window.alert("You said: "+val);
```

A) Nothing

B) You said: 10

C) Error

Input/Output Question (2)

Question: Assume the user typed in **10** when prompted. What is shown in the HTML document after this code?

```
var val;
window.prompt("Enter a value: ");
document.write("You said: "+val);
```

A) Nothing

B) `You said: 10`

C) `You said: undefined`

D) Error

Practice Questions

For these questions, use slide #33 as an example. Do not copy the HTML code, just write the JavaScript statements.

1) Write the JavaScript code to print:

Hello, World!

Goodbye, World!

2) Write the JavaScript code to print:

Testing…

1..2..3..

1+2+3 = 6

1*2*3 = 6

Note: You must calculate 6 in both cases not just print it!

3) Write a program to calculate and print: (a=5, b=10)

c = 25*a+b-32

Review: Key Programming Concepts

Some key concepts in programming:

◆ *variables* – names for data items to be manipulated

◆ *locations* – addresses of data items in memory

◆ *values* – the value stored at a particular location and referenced using a given variable name

◆ *initialization* – setting beginning values for variables

◆ *assignment* – general form of initialization where the value of a variable is set to another value

◆ *expressions* – consist of operands and operators and yield a result

Conclusion

We learned the basics of the JavaScript language to communicate instructions to the computer including:

- ◆ declaring and using variables
- ◆ initialization and assignment of values to variables
- ◆ reading input and displaying output
- ◆ expressions

Objectives

- Compare and contrast: algorithm and program
- List and define the key programming concepts covered.
- Explain the difference between variables, values, and locations.
- Remember the rules for variables, comments, and statements.
- Remember the rules for declaring and using strings.
- Understand and explain assignment operator.
- Define: operator, operand, unary, binary
- Remember operator precedence for expressions.
- Recall the string concatenation operator.
- Be able to write and execute JavaScript code in HTML files.
- Define: operator overloading
- Know how to get input and send output to and from the user.