**Paradigms in Programming Languages**

Frances Coronel

Department of Computer Science

Hampton University  
100 E. Queen Street

Hampton, Virginia 23668  
757-440-8668

frances.coronel@my.hamptonu.edu

**ABSTRACT**

This paper outlines how I approached implementing the classic bubble sort algorithm in 11 different programming languages throughout my CSC 308 – Organization of Programming Languages class.

**CCS Concepts**

CCS ⟶ Software and its engineering ⟶ Software notations and tools ⟶ General programming languages ⟶ Languages Types

**Keywords**

JavaScript, C, C++, Fortran, COBOL, LISP, MASM, Ruby, Python, PHP, JavaScript, Algorithms, Language Design

# INTRODUCTION

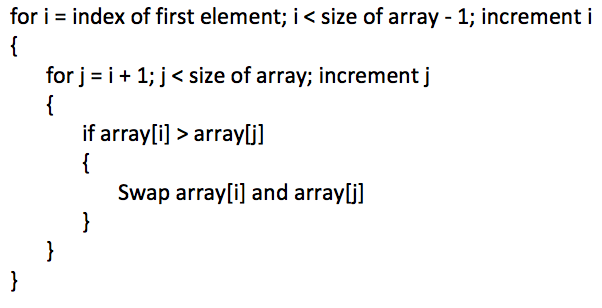
This Spring semester, I was given the task of implementing the bubble sort in a variety of different languages, starting with Java. Given that Hampton University’s curriculum primarily uses Java, I didn’t have too many problems with more OOP languages.

Having worked a lot with JavaScript outside of class, that was also extremely easy to implement.

However, I definitely struggled with COBOL, MASM, Fortran, and LISP because these languages were more mathematically oriented and worked on a lower-level than I was used to. All the other languages were easier to understand for me.

For all the implements, the same variable names were used as permitted by the language itself. The variable names are “array” for the array to be sorted, “i” and “j” for subscripts to index into the array, and “temp” to hold a variable’s contexts during a swap required by the algorithm. The size of the array should be determined by function calls such as “.length” or “sizeof( )”.

The basic structure for the pseudo code used to implement bubble sort in the majority of these languages is as follows can be found in Figure 1.



**Figure 1. Bubble Sort Pseudo Code**

# JAVA

Like any programming dialect, the Java dialect has its own structure, linguistic structure standards, and programming worldview. The Java dialect's modifying worldview depends on the idea of OOP, which the dialect's elements support [1].

The Java dialect is a C-dialect subsidiary, so its punctuation rules look much like C's. For instance, code squares are modularized into techniques and delimited by supports and variables are proclaimed before they are utilized [1].

Basically, the Java dialect begins with bundles. A bundle is the Java dialect's namespace system. Inside bundles are classes, and inside classes are strategies, variables, constants, and that's just the beginning. You find out about the parts of the Java dialect in this instructional exercise [1].

Programming in Java was straightforward to me since I have a bit of experience with Java already. All programming heavy courses at Hampton utilize Java as the main language. Even so, I’m not a big proponent of using the language myself since my interests primarily lie with web application development.

## Formatting

I always find it more convenient to use line breaks generously since it logically separates thoughts and doesn’t exceed the column limit.

Default indentation should be 2 columns and single line statements should be broken down so that they don’t surpass a general 100 column limit. There should also be a package installed in your development environment that there is no trailing whitespace.

// ideal formatting

if ( x < 0 ) {

negative(x);

} else {  
 nonnegative(x);

}

// not ideal

if ( x < 0 )

negative(x);

if ( x < 0 ) negative(x);

## Structs

Traditionally, Java is programmed in a very verbose style. However, I find it more ideal to to write classes that merely hold data like in C++ as so:

public class DataHolder {

public final String data;

public DataHolder(String data) {

this.data = data;

}

}

## Variables

It is good practice to use CamelCase for types, camelCase for variables, and UPPER\_SNAKE for constants. Extremely short variable names should be reserved for instances like loop indices only. Every other variable should be appropriately named.

// not ideal

private final int a;

private final String f;

// ideal

private final int ageInYears;

private final String firstName;

## Source Code

class BubbleSort {

public static void main(String[] args) {

// array of integers

int[] array = { 34, 56, 4, 10, 77, 51, 93, 30, 5, 52 };

// print unsorted array

System.out.println("Unsorted Array");

for (int i = 0; i < array.length; i++) {

System.out.print(array[i] + " ");

}

//bubble sort

for (int i = 0; i < array.length-1; i++) {

for (int j = i+1; j < array.length; j++) {

if (array[i] > array[j]) {

int temp = array[i];

array[i] = array[j];

array[j] = temp;

}

}

}

// print sorted array

System.out.println("\n\nSorted Array");

for (int i = 0; i < array.length; i++) {

System.out.print(array[i] + " ");

}

}

}

# C/C++

C++ is based on C and retains a great deal of the functionality and strcuture. However, C does not have source-level compatibility in absolutely every instance with C. When trying to write C code and compile with a C++ compiler, a few problems can arise since the syntax is different in some cases [2].

Preceding C++, C was a programming dialect created at Bell Labs around 1969-1973. The UNIX working framework was likewise being produced at Bell Labs in the meantime. C was initially produced for and actualized on the UNIX working framework. C is convenient, not fixing to a specific equipment or working framework. C consolidates the components of abnormal state dialects with the usefulness of low level computing construct and has once in a while been alluded to as a center level coding. C makes it simple to adjust programming for one kind of PC to another [3].

C++ was composed by Bjarne Stroustrup at Bell Labs amid 1983-1985. C++ is an augmentation of C. Preceding 1983, Bjarne Stroustrup added components to C. He had consolidated the use of classes and protested situated components with the force and proficiency of C. The term C++ was initially utilized as a part of 1983. [4].

My first formal programming class was in C++ when I took the “Intro to Computing” class at Tidewater Community College the summer of my senior year of high school. I’ve also taken a formal C programming language course at my past university, Jacobs University Bremen. More recently, I took a C++ class with Dr. Boonthum at Hampton University this past year. In other words, I didn’t find the program that difficult to implement.

## Source Code (C)

#include <stdio.h>

int main(void) {

// array of integers

int array[] = { 34, 56, 4, 10, 77, 51, 93, 30, 5, 52 };

// print unsorted array

printf("Unsorted Array\n");

for (int i = 0; i < sizeof(array)/sizeof(array[0]); i++) {

printf("%d ", array[i]);

}

// bubble sort

for (int i = 0; i < sizeof(array)/sizeof(array[0])-1; i++) {

for (int j = i+1; j < sizeof(array)/sizeof(array[0]); j++) {

if (array[i] > array[j]) {

int temp = array[i];

array[i] = array[j];

array[j] = temp;

}

}

}

// print sorted array

printf("\n\nSorted Array\n");

for (int i = 0; i < sizeof(array)/sizeof(array[0]); i++) {

printf("%d ", array[i]);

}

// return 0 necessary for C programs

return 0;

}

## Source Code (C++)

#include <stdio.h>

#include <iostream>

#include <iomanip>

using namespace std;

int main(int argc, char \*argv[]) {

// array of integers

int array[] = { 34, 56, 4, 10, 77, 51, 93, 30, 5, 52 };

// print unsorted array

cout << "Unsorted Array" << endl;

for (int i = 0; i < sizeof(array)/sizeof(array[0]); i++) {

cout << array[i] << " ";

}

// bubble sort

for (int i = 0; i < sizeof(array)/sizeof(array[0])-1; i++) {

for (int j = i+1; j < sizeof(array)/sizeof(array[0]); j++) {

if (array[i] > array[j]) {

int temp = array[i];

array[i] = array[j];

array[j] = temp;

}

}

}

// print sorted array

cout << "\n\nSorted Array" << endl;

for (int i = 0; i < sizeof(array)/sizeof(array[0]); i++) {

cout << array[i] << " ";

}

}

# FORTRAN

One of the most established programming dialects, the FORTRAN was created by a group of developers at IBM drove by John Backus, and was initially distributed in 1957. The name FORTRAN is an acronym for FORmula TRANslation, since it was intended to permit simple interpretation of math recipes into code [5].

Frequently alluded to as an investigative dialect, FORTRAN was the main abnormal state dialect, utilizing the principal compiler ever created. Before the improvement of FORTRAN PC software engineers were required to program in machine/get together code, which was a to a great degree troublesome and tedious errand, also the repulsive task of investigating the code. The target amid it's configuration was to make a programming dialect that would be: easy to learn, reasonable for a wide assortment of utilizations, machine free, and would permit complex scientific expressions to be expressed likewise to standard logarithmic documentation [5].

Fortran was difficult for me to wrap my head around because it was very mathematically based and the printing structure was very different than what I was used to. Even so, the pseudo code could still be applied with relative ease here.

## Source Code

PROGRAM BubbleSort

IMPLICIT NONE

!Declaring Variables

INTEGER :: temp, i, j

!An array of integers

INTEGER, DIMENSION(10) :: array = (/ 34, 56, 4, 10, 77, 51, 93, 30, 5, 52 /)

!Print Unsorted Array

WRITE(\*,\*) "Unsorted Array"

WRITE(\*, 100) array

100 FORMAT (10I3)

!Bubble Sort

DO i = 1, SIZE(array)

DO j = i+1, SIZE(array-1)

IF (array(i) > array(j)) then

temp = array(i)

array(i) = array(j)

array(j) = temp

END IF

END DO

END DO

!Print Sorted Array

WRITE(\*,\*) "Sorted Array"

WRITE(\*, 200) array

200 FORMAT (10I3)

END PROGRAM BubbleSort

# COBOL

COBOL remains for Common Business Oriented Language. The US Department of Defense, in a gathering, framed CODASYL (Conference on Data Systems Language) to build up a dialect for business information preparing needs which is currently known as COBOL [6].

COBOL is utilized for composing application projects and we can't utilize it to compose framework programming. The applications like those in barrier space, protection area, and so on which require immense information handling make broad utilization of COBOL [6].

COBOL was definitely not a very enjoyable language experience for me. It’s a very wordy language and declaring variables is a pain that requires a lot of documentation referencing. Tedious to setup and tedious to compile since it’s such an old language.

## Source Code

identification division.

program-id. BubbleSort

environment division.

configuration section.

data division.

working-storage section.

01 arraydata value ' 34 56 4 10 77 51 93 30 5 52'.

 05 array occurs 10 times pic 999.

01 displayarray pic zz9.

01 spacing pic 99 value 1.

01 spacing2 pic 99 value 1.

01 i pic 99.

01 j pic 99.

01 temp pic 99.

01 sizeof pic 999.

procedure division.

compute sizeof = length of arraydata / length of array.

display "Unsorted Array".

perform varying i from 1 by 1 until i is greater than sizeof

move array(i) to displayarray

display displayarray at line number 2 column spacing

add 3 to spacing

end-perform.

display " ".

move 1 to i.

perform until i > sizeof

move i to j

perform until j > sizeof

if (array(i) > array(j))

move array(i) to temp

move array(j) to array(i)

move temp to array(j)

end-if

add 1 to j giving j

end-perform

add 1 to i giving i

end-perform

move 0 to i

display "Sorted Array" at line 4 column 1.

move 1 to spacing2

perform varying i from 1 by 1 until i is greater than sizeof

move array(i) to displayarray

display displayarray at line number 5 column spacing2

add 3 to spacing2

end-perform.

display " ".

end program BubbleSort.

# LISP

LISP is a computer programming language developed around 1960 by John McCarthy at the Massachusetts Institute of Technology (MIT). LISP was established on the numerical hypothesis of recursive capacities (in which a capacity shows up in its own definition). A LISP system is a capacity connected to information, instead of being an arrangement of procedural strides as in FORTRAN and ALGOL [7].

LISP turned into a typical dialect for artificial intelligence (AI) programming, incompletely attributable to the conjunction of LISP and AI work at MIT and mostly on the grounds that AI programs able to do "learning" could be composed in LISP as self-changing projects. LISP has developed through various tongues, for example, Scheme and Common LISP [7].

LISP’s syntax was interesting to me since it uses parentheses to define functions which is just how it usually is with other languages. However, there was a lot of built in functionality that I found I could use to sort arrays.

There is even a built in sort function that works really well. So technically, the program could have been a one-liner if the given variables weren’t required for use:

(write (sort '(34 56 4 10 77 51 93 30 5 52) #'<))

## Source Code

; defining array list with predefined values

(defparameter \*list\* (list 34 56 4 10 77 51 93 30 5 52))

(defun swap(array i j)

(rotatef (nth i array) (nth j array))

 ; rotatef modifies the values of each place by rotating values from one place into another

 ; rotatef is equivalent to the following in Java:

 ; int temp = array[i];

 ; array[i] = array[j];

 ; array[j] = temp;

)

; elt - accesses the element of sequence identified by index

(defun sort(array)

 ; for (int i = 0; i < array.length-1; i++)

(dotimes (i (list-length array))

   ; for (int j = i+1; j < array.length; j++)

(dotimes (j (list-length array))

     ; if (array[i] > array[j]))

(if (< (elt array i) (elt array j))

       ; perform swap on each element

(swap \*list\* i j)

)

)

)

)

; print unsorted array

(print "Unsorted array")

(print \*list\*)

; (sort '(34 56 4 10 77 51 93 30 5 52))

(sort \*list\*)

; print sorted array

(print "Sorted array")

(print \*list\*)

# MASM

The Microsoft Macro Assembler is an x86 assembler for MS-DOS and Microsoft Windows. While the name MASM has prior usage it is regularly comprehended in later years to allude to the Microsoft constructing agent. It bolsters a wide assortment of macro facilities and structured idioms, including high-level functions for looping and procedures [8].

The Microsoft Macro Assembler (MASM) gives a few points of interest over inline get together. MASM contains a large scale dialect that has components, for example, circling, number-crunching, and content string preparing. MASM likewise gives you more noteworthy control over the equipment since it underpins the direction sets of the 386, 486, and Pentium processors. By utilizing MASM, you likewise can lessen time and memory overhead [8].

I probably had the worst experience working with MASM since I’m a passionate OS X user and MASM programs can only run on Windows machines. It took me longer to figure this program out than any other one.

## Source Code

;

; Bubble Sort Program

;

.386

model flat, stdcall

.STACK

include kernel32.inc

include masm32.inc

includelib kernel32.lib

includelib masm32.lib

.DATA

ExitMsg byte "Press any key to continue . . . ", 00dh, 00ah, 0

array dword 34, 56, 4, 10, 77, 51, 93, 30, 5, 52

endarray dword 0

arraysize equ endarray-array

.CODE

main proc

;

; Print Unsorted Array Message

;

invoke StdOut, addr UnsortedArray ; print "Unsorted Array: "

;

; for (i = 0; i < arraysize; i++)

;

mov ebx, 0 ; start loop at 0 (i = 0)

top:

cmp ebx, arraysize ; compare # in ebx variable to array size (i < arraysize)

jge exit ; variable to

;

; Print Sorted Array Message

;

invoke StdOut, addr SortedArray ; print "Sorted Array: "

;

; invoke CRLF

;

exit:

invoke StdOut, addr CRLF ; print CR LF

;

; Print exit message

;

invoke StdOut, addr ExitMsg ; print exit message

invoke StdIn, addr continue, 1 ; read to keep console window open after exit message

invoke ExitProcess, 0 ; exit

main endp

end main

# RUBY

Ruby is a pure object-oriented programming language. It was created in 1993 by Yukihiro Matsumoto of Japan [9].

Ruby has features that are similar to those of Smalltalk, Perl, and Python. Ruby is an open-source and is freely available on the Web, but it is subject to a license. It can be used to write Common Gateway Interface (CGI) scripts and can be embedded into Hypertext Markup Language (HTML) [9].

Ruby has a clean and easy syntax that allows a new developer to learn Ruby very quickly and easily. It is similar to to that of many programming languages such as C++ and Perl. It’s also very much scalable with big programs written in Ruby being easily maintainable [9].

I had dabbled with Ruby once or twice before through online programming courses but I had never seriously attempted to learn it. Either way, I found it that it reminded me of Python because of its simplified English format so I didn’t find it that bad.

## Source Code

#!/usr/bin/ruby

# define array

array = [ 34, 56, 4, 10, 77, 51, 93, 30, 5, 52 ]

# print out unsorted array

print "Unsorted Array \n"

for i in 0..(array.length - 2)

 print array[i], " "

end

# bubble sort algo

for i in 0..(array.length - 2)

 for j in i + 1..(array.length - 1)

   # swap method described in class

   if array[i] > array[j]

     array[i], array[j] = array[j], array[i]

   end

 end

end

# print sorted array

print "\n\nSorted Array\n"

for i in 0..array.length-2

 print array[i], " "

end

# press any key to continue

print "\n\nPress any key to continue . . ."

gets

# PYTHON

Python is an unmistakable and intense object oriented programming dialect, similar to Perl, Ruby, Scheme, or Java.

Python was produced by Guido van Rossum in the late eighties and mid nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands [10].

Python is gotten from numerous different dialects, including ABC, Modula-3, C, C++, Algol-68, SmallTalk, and Unix shell and other scripting dialects [10].

Python is copyrighted. Like Perl, Python source code is presently accessible under the GNU General Public License (GPL) [10].

I enjoy Python. It’s simple, fast, and easier to read than most languages. Having taken Python at Hampton University and at my past university, I was also comfortable with the syntax and minimalism.

## Source Code

# defining array of integers

array = [34, 56, 4, 10, 77, 51, 93, 30, 5, 52 ]

# print unsorted array

print("Unsorted Array")

buffer = ""

for i in range(len(array)):

buffer = buffer + " " + str(array[i])

print(buffer)

# bubble sort

for j in range(len(array)-1,0,-1):

for i in range(j):

if array[i] > array[j]:

# swap elements

temp = array[i]

array[i] = array[j]

array[j] = temp

# print sorted array

print("\nSorted Array")

buffer = ""

for i in range(len(array)):

buffer = buffer + " " + str(array[i])

print(buffer)

input('\nPress any key to continue . . .')

# PHP

PHP (recursive acronym for PHP: Hypertext Preprocessor) is a broadly utilized open source universally useful scripting dialect that is particularly suited for web improvement and can be implanted into HTML. Rather than loads of charges to yield HTML (as found in C or Perl), PHP pages contain HTML with implanted code that does "something" [11].

What differs PHP from something like client side JavaScript is that the code is executed on the server, producing HTML which is then sent to the client. The client would get the aftereffects of running that script, yet would not comprehend what the fundamental code was. You can even design your web server to process all your HTML records with PHP, and after that there's truly no chance that clients can tell what you have up your sleeve [11].

The best things in utilizing PHP are that it is to a great degree basic for a newcomer, however offers numerous propelled highlights for an expert software engineer. So in spite of the fact that PHP's advancement is centered around server-side scripting, you can do a great deal more with it [11].

Having a background in web development with full stack JavaScript, running the PHP server was a piece of cake for me but I still enjoy working with JavaScript more. The process was very easy with XAMP installed on my computer.

## Source Code

<html>

<body>

<?php

 // create short variable name from array

 $array = $\_POST['array'];

 // print unsorted array

 echo "Unsorted Array<br>";

 for ($i = 0; $i < count($array); $i++)

   echo $array[$i]." ";

 echo "<br>";

 // bubble sort

 $array\_length = count($array);

 for ($i = 0; $i < $array\_length - 1; $i++) {

   for ($j = $i + 1; $j < $array\_length; $j++) {

     if ($array[$i] > $array[$j]) {

       $temp = $array[$i];

       $array[$i] = $array[$j];

       $array[$j] = $temp;

     }

   }

 }

 echo "<br>";

 // print sorted array

 echo "Sorted Array<br>";

 for ($i = 0; $i < count($array); $i++)

   echo $array[$i]." ";

?>

</body>

</html>

# JAVASCRIPT

JavaScript (frequently abbreviated to JS) is a lightweight, translated, object-arranged dialect with top of the line capacities, and is best known as the scripting dialect for Web pages, yet it's utilized as a part of numerous non-program situations too. It is a model based, multi-worldview scripting dialect that is dynamic, and backings object-arranged, basic, and useful programming styles [12].

JavaScript keeps running on the customer side of the web, which can be utilized to plan/program how the site pages act on the event of an occasion. JavaScript is a simple to learn furthermore capable scripting dialect, generally utilized for controlling page content 12].

As opposed to famous misguided judgment, JavaScript is not " Java". More or less, JavaScript is an element scripting dialect supporting model based article development. The essential grammar is deliberately like both Java and C++ to lessen the quantity of new ideas required to take in the dialect [12].

JavaScript can work as both a procedural and an article situated dialect. Items are made automatically in JavaScript, by connecting strategies and properties to generally exhaust objects at run time, instead of the syntactic class definitions basic in gathered dialects like C++ and Java. Once an article has been developed it can be utilized as an outline (or model) for making comparable items [12].

JavaScript's dynamic abilities incorporate runtime object development, variable parameter records, capacity variables, dynamic script creation (through eval), object reflection (by means of for ... in), and source code recuperation (JavaScript projects can decompile capacity bodies once again into their source content) [12].

JavaScript is my absolute favorite language. I’ve been working with JS since high school in its various forms because of how useful it is. Being able to use it across a wide set of platforms and areas, I’ve found it be to not just immensely useful to my career, but also to my satisfaction as a programmer. No wonder it’s the most popular programming language in the world right now. It’s easy to understand, flexible, and powerful.

## Source Code

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en">

<head>

   <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />

   <script type="text/javascript">

   // print out unsorted array

   document.write("Unsorted Array<br>");

   var array = new Array(34, 56, 4, 10, 77, 51, 93, 30, 5, 52);

   for (var i = 0; i < array.length; i++)

       document.write(array[i] + " ");

   // bubble sort

   for (var i = 0; i < array.length; i++) {

       for (var j = i + 1; j < array.length; j++) {

           if (array[i] > array[j]) {

               var temp = array[i];

               array[i] = array[j];

               array[j] = temp;

           }

       }

   }

   // print out sorted array

   document.write("<br><br>Sorted Array<br>");

   for (var i = 0; i < array.length; i++)

       document.write(array[i] + " ");

   </script>

</head>

<body>

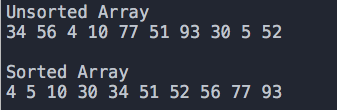
</body>

</html>

# CONCLUSION

In the end, I learned a lot about how various languages piece together with their own unique syntax and grammar. In many ways, this semester was similar to someone practicing writing a paragraph or two in 11 different languages (e.g. Spanish, French, etc.) since each programming language requires a lot more practice to master but not too much effort to dabble in.

Ultimately, each program resulted in the following output highlighted below in Figure 10. Each program outputted the given array of integers provided and then also outputted that same array but now in a sorted order, from least to greatest.



**Figure 10. Output of Bubble Sort Programs**

Overall, it was definitely an enjoyable experience being able to solve a classical algorithm in almost a dozen different languages.

# REFERENCES

1. IBM DeveloperWorks, Introduction to Java programming, Part 1: Java language basics. Website is accessible at <http://www.ibm.com/developerworks/java/tutorials/j-introtojava1/>.
2. Differences between C and C++. Website is accessible at <http://www.cprogramming.com/tutorial/c-vs-c++.html>.
3. C Language – Overview. Website is accessible at <http://www.tutorialspoint.com/cprogramming/c_overview.htm>.
4. C++ - Overview. Website is accessible at <http://www.tutorialspoint.com/cplusplus/cpp_overview.htm>.
5. The FORTRAN Programming Language. Website is accessible at <http://groups.engin.umd.umich.edu/CIS/course.des/cis400/fortran/fortran.html>.
6. COBOL Tutorial. Website is accessible at <http://www.tutorialspoint.com/cobol/>.
7. What Made LISP Different. Website is accessible at <http://www.paulgraham.com/diff.html>.
8. Microsoft Macro Assembler Reference. Website is accessible at <https://msdn.microsoft.com/en-us/library/afzk3475.aspx>.
9. Programming Ruby, The Pragmatic Programmer's Guide. Website is accessible at <http://ruby-doc.com/docs/ProgrammingRuby/>.
10. About Python, A Byte of Python. Website is accessible at <http://python.swaroopch.com/about_python.html>.
11. What is PHP? Website is accessible at <http://php.net/manual/en/intro-whatis.php>.
12. About JavaScript, Mozilla Developer Network. Website is accessible at <https://developer.mozilla.org/en-US/docs/Web/JavaScript/About_JavaScript>.