*Object Oriented Programming Language:*

*A comprehensive analysis of C++, java, and C#*

*Zachary Lessard*

*Table of contents*

1. Introduction …………………………………………………………………………
   1. What is object oriented programming? …………………………………………
   2. History of OOPL? ………………………………………………………………

1.2.2 History of C++

1.2.3 History of Java

1.2.4 History of C#

1. Fundamental Features

2.1 Fundamental Features ………………………………………………………...

2.2 Types of Programming Domains ……………………………………………..

2.2.1 Scientific Application

2.2.2 Business Application

2.2.3 Artificial Intelligence

2.2.4 Systems programming

2.2.5 Web Software

2.3 C, C#, and java programming domain …………………………………… 2.3.1 Java Domain 2.3.2 C++ Domain 2.3.3 C# Domain

2.4 Programming Paradigm …………………………………………………….... 2.4.1 Java Paradigm 2.4.2 C++ Paradigm 2.4.3 C# Paradigm

2.5 Readability …..……………………………………………………................... 2.4.1 Java Readability 2.4.2 C++ Readability 2.4.3 C# Readability

2.6 Simplicity …..……………………………………………………................... 2.4.1 Java Simplicity 2.4.2 C++ Simplicity 2.4.3 C# Simplicity

2.7 Orthogonally …..……………………………………………………................... 2.4.1 Java Orthogonally 2.4.2 C++ Orthogonally 2.4.3 C# Orthogonally

2.8 Portability …..……………………………………………………........................ 2.4.1 Java Portability 2.4.2 C++ Portability 2.4.3 C# Portability

2.9 Programming Environment …..…………………………………………………. 2.4.1 Java Programming environment 2.4.2 C++ Programming environment 2.4.3 C# Programming environment

3. Advance Features

3.1 Translation process ……………………………………………….................... 3.1.1 Java Translation Process 3.1.2 C++ Translation Process 3.1.3 C# Translation Process

3.2 Data types, variables, and support for abstraction………………………………...................

3.2.1 Java Data types, variables, and support for abstraction 3.2.2 C++ Data types, variables, and support for abstraction 3.2.3 C# Data types, variables, and support for abstraction

3.3 Expression and alignment statements………………………………................... 3.3.1 Java Expression and alignment statements 3.3.2 C++ Expression and alignment statements 3.3.3 C# Expression and alignment statements

3.4 Control Structures ……………………………………………………...................

3.5 Subprograms …..……………………………………………………...................

3.6 Exception handling…………………………………………………................... 3.6.1 Java Exception handling 3.6.2 C++ Exception handling 3.6.3 C# Exception handling

4. Summary and Concluding Remark

4.1 Conclusion …..……………………………………………………...................

5. References

**1.0 Introduction**

* 1. **What is Object oriented programming Language?**

Object oriented programming language (OOPL) refers to a type of computer programming where programmers define the data type of a data structure, and the types of operations that can be applied to the data structure. Once this methodology is applied that particular data structure becomes an object that has both data and functions. Relationships can be made among these objects, thus inheriting characteristics of one another; this methodology in OOP is known as **inheritance**.

**Classes** and **object** are the two main aspects of object oriented programming. A class creates a new type, where as an object is an instance of that new type. Objects can store data using variables that belong to the object. Variables that are associated with an object or a class are referred to as **fields**. Objects can have functionality by using other functions that belong to a class; these functions are called **methods** of a class. Together fields and methods are referred to as **attributes** of a class.

* 1. **Brief history of object oriented programming**

The first object oriented programming language dates back to the early 1960s, where the first programming language, “Simula 67” was invented. “Simula” was the first object oriented programming language to use classes and some form of inheritance, that’s why it is considered to be the father of all object oriented programming languages. Off of this language many other object oriented languages would spawn; the larger object oriented programming languages include, Java, C#, Python, C, Ruby, PHP, Objective-C. The languages being analyzed are Java, C++, and VB.NET. Below is a short history of each language:

**1.2.1 History of C++**

C++ origins date back to 1979, when Bjarne Stroustrup was doing work with the “Simula” language which was slow for any practical use. “C with class” was shortly developed, which combined “C” and “Simula” and brought with it strong type checking, basic inheritance, and classes. The first “C with classes” compiler was called Cfront. Cfront would have a huge impact on the later implementations of future “C++” compilers and on the Unix operating system.

Around 1983, the name changed from “C with classes” to “C++”, and with the name change came many new features, these new features included virtual functions, function overloading, the const keyword, and single-line comments. Within the next few years of these feature releases, in 1985 *“the C++ programming language”* was published and in that same year C++ officially a commercialized product. From then to the present, there have been improvements to both the language and its compiler.

**1.2.2 History of Java**

Java was first developed in 1991, and owes most of its origins to C++. Java was developed by a group of sun engineers known as the “green team” and was originally designed for small, embedded systems in electronic appliance. Platform independence is what separates java from its counterpart C++. This means that java can run the same code on JVM (Java Virtual Machine) and get the same results every time.

In 1995 Java development teams started to look towards the internet with the release of the net scape browser which incorporated java, and in that same year java was named one of the top 10 business programming languages available, and remains to be to this day. Java is implemented over numerous places in the modern world, these include standalone applications, web applications, enterprise application and mobile application and is the go to language for business applications.

**1.2.3 History of C#**

The History of Visual Basics dates back to 1991 when the first version (VB 1.0) of visual basics was released. Visual Basic was build off of the BASIC language, which gained a lot of popularity through the 1980s. Alan Cooper, the developer of BASIC was asked by Microsoft to develop his original project into a form building application. The “Ruby” programming language was developed from this and didn’t include any sort of programming language at all. Microsoft decided to bundle the form building application, with BASIC, and from this Visual Basic was born.

Ruby also provided the ability to load dynamic link libraries containing additional controls, which would later become the VBX interface. Since its original release in 1991 there have been 16 additional versions of the programming language released since.

**2.0 Fundamental Features**

**2.1 Programming Domains**

Computers are used in a number of different areas, from aiding flights, to helping in the medical field. Because of the great diversity in the field naturally the programming languages that control them are going to have very different goals. In this section we will identify where each of the programming languages being evaluated fall. To do this we must first identify each of the different types of programming domains.

**2.2 Types of Programming Domains**

**2.2.1 Scientific Applications**

A Scientific Application simulates real world activities using mathematics. Real-world objects are turned into mathematical models and they are manipulated using formulas. Scientific applications can be dated back to the first computers in the late 1940s and early 1950s.

**2.2.2 Business Applications**

Business software or a business application is any software or set of computer programs used by business users to perform a number of business functions. Common functions for business applications include increasing productivity, measuring productivity and to perform other business functions accurately. Business programming languages date back to the 1950s, but the first commercially successful business programming language, COBOL, was released in 1960. To this day COBOL, is the widest used business application.

**2.2.3 Artificial intelligence**

Artificial intelligence (AI) is a large area of computer applications characterized by symbolic rather than numeric computations. The working definition for symbolic in this context means that symbols, rather than numbers are manipulated. AI dates back to 1959, when programming language called “LISP” was made by McCarthy. A well-known application that falls under this domain is “Watson” which was developed by IBM to assist in many areas of study.

**2.2.4 Systems Programming**

The operating system (OS) and the programming support tools of a computer system are also known as its system software’s. Unlike application programming which provides services to the user directly (e.g. word processing), where-as with system programming aims to produce software/software platforms which provide services to other software’s. System programming dates back to the 1960s and 1970s when computer manufacturers such as IBM, Digital, and Burroughs, started development on machine-oriented high level programming languages.

**2.2.5 Web Software**

The World Wide Web has a wide collection of programming languages ranging from markup languages, such as HTML, to general purpose languages such as C++ and java. There are also scripting languages such as JavaScript and PHP. Web software’s date back to the beginning of the internet in the 1990s.

**2.3 C, C#, and java programming domains**

All of the programming languages being discussed are general purpose languages. Meaning that they could fall under any of these programming domains, but in this section we will explore what domain fits best for C, C#.net, and Java. To determine what domain each language falls into, we must first examine the strengths and weaknesses of each language.

**2.3.1 Java Domain**

Because java is really good at storing decimal numbers, and character data, it makes it a perfect business programming language, and Javas automatic memory allocation and deallocation is ideal for large amounts of data. Along with its great data storage Java is multi-platform, making it another great choice for business.

Java is one of the best and undeniably the most popular languages used today, and there are lots of advantages over C++, .net for business applications. Although java was originally designed for large enterprise solutions, it is flexible enough for the needs of smaller projects, which allows you to start small and then evolve in accordance with business needs without the necessity to rewrite the whole code. It’s fast, reliable, and easy to develop and work with.

**2.3.2 C++ Domain**

Like Java, C++ is one of the most widely used languages in the world, from areas such as business applications, to video games. C++ along with C, is widely used in the video game industry, because of all the tools that have been developed for the language. Video games fall under the domain of AI, thus C++ is most widely used for Artificial intelligence. As stated before this is not the only area where C++ is used it is just widely used in this particular field.

**2.3.3 C# Domain**

Unlike Java and C++, C# is not multiplatform. C#.net is used within the Microsoft Windows environment, this includes Windows Client, Windows Servers, Active Directory, IIS and SQL server. Because of this C# is limited to just windows applications it makes it very limited if you are trying for a cross platform system, but because of windows server making web applications through C# is made easy.

**2.4 Programming Paradigm**

A programming paradigm is a style or way of programming. The four main paradigms include imperative paradigm, declarative paradigm, object oriented paradigm, and the flow-driven paradigm, procedural programming, generic programming. The imperative paradigm the control flow is an explicit sequence of commands. The declarative paradigm states the results that the programmer wants, not how to get it. The OOP is effected by sending data to objects, these objects have behaviors and states that include class based, and prototype based. The flow driven paradigm is specified by multiple processes communication over predefined channels. Procedural Programming is achieved through passing arguments to functions and returning values from functions. Generic programming is a style of computer programming in which algorithms are written in terms of types that are specified later in the program, in which are instantiated when needed for specific type parameters.

Each of the programming languages being discussed are known as “multiparadigm programming” (Concepts of Programming Languages, 536). Like the name implies, this means that the language follows multiple paradigms. A language is said to support a paradigm if it provides facilities that make it convenient to use that style. A language does not support that particular paradigm if it takes a lot of effort to write such a program, for example, you can write object oriented programs in C, but it is difficult to do so because C does not directly support that type of paradigm. In this section we will discuss what paradigms each language follows.

**Section 2.4.1 Java Paradigm**

Java follows the following paradigms: imperative and object oriented programming. A good example of imperative paradigm used in java is the counter program:

Figure 2.0



This program is considered imperative because it changes the state of the program.

Java uses objects, classes, abstraction, encapsulation, inheritance, and polymorphism to implement object – oriented programming. An example of object oriented programming is as followed:

Figure 2.1



In this particular example there methods that are called in main in the form of objects.

**Section 2.4.2 C++ Paradigm**

C++ follows the following paradigms: data abstraction, generic programming and object-oriented programming. A good example of procedural programming in C++ is the square root program. Within this program a double-precision floating-point argument is in place to produce this common mathematical computation.

Figure 2.2



The Curly brackets ({ }), in this particular example represent the start and the ending of the function. The keyword “void” indicates that the function is going to return null.

Object – oriented paradigm is easily applied in the C++ language, by using objects, classes, abstraction, encapsulation, inheritance, polymorphism and overloading. A quick example of object oriented programming is as followed:

Figure 2.3



The above example shows how a class is defined, its object is created and the member functions are used. Variables can be declared anywhere in the entire program, but it has to be declared, before they are used, C++ gets this procedural format from its father programming language C.

The simplest form for generic programming are container classes, like arrays, and lists that contain a collection of other objects. A basic example of generic programming can be shown through templates of containers. C++ is a statically typed language meaning that separate containers must be defined that hold floats, integers, and other types, but templates deals with this constraint by letting the programmer define classes where one or more parameters are unspecified at the time you define the class. When the programmer instantiates the template later in the program, the programmer will define what type the template should use to create the class out of the template. An example of this methodology in use is as followed:

Figure 2.4



In this particular example, the template is generic because the compiler translates the template into actual code. If the template was instantiated, there would be no generated code.

**Section 2.4.3 C# Paradigm:**

C#.net is a multiparadigm programming language that is comprised by the imperative paradigm, and the object oriented paradigm. The imperative paradigm in C#.net is illustrated by the following example:

Figure 2.5



**Section 2.5 Readability**

One of the most important criteria for judging a programming language is the ease at which programs can be read and understood. We as humans write code in higher languages so people can understand what we are writing. Readability must be considered for a problem domain among programming languages. If a program that describes a computation is written in a language that is not designed for such a computation, the program becomes convoluted, making it very difficult to read. As languages update, readability becomes harder to maintain.

**Section 2.5.1 Java Readability**

The readability of java and C# are very similar because of their similarities in syntax. Like C#, java is a general purpose programming language and gets a lot of its readability aid through the use of object oriented programming.

**Section 2.5.2 C++ Readability**

C++ has a pretty complex readability. C++ is a large language thus readability is greatly affected. C++ has what is called operator overloading, this concept alone effects readability greatly. Operator overloading can make a symbol mean two or more things causing the program to become convoluted. Hackers like C++ for this reason because it can make the code cryptic if reversed engineered.

**Section 2.5.3 C# Readability**

C# is a general purpose programming language; unlike C++, C# calls for high abstraction; unlike java it is more aggressively driven. C# 6 has taken steps in the last few updates to introduce a new syntax for primary constructors, which will simplify classes. This is just one of the many updates that Microsoft has put out to make the readability of C# less convoluted.

**Section 2.6 Simplicity**

The overall simplicity of a language is strongly characterized by the languages readability. For example a language with a large amount of basic constructs is more difficult to learn than one with a smaller number. Simplicity of a programming language is affected if it supports multiplicity. Multiplicity is a practice in programming where there is more than one way to write an operation. Like readability, operation overloading, can affect a programming languages simplicity.

**Section 2.6.1 Java Simplicity**

Java owes a lot of its success because of the simplicity of the language. There is a reason that java is used by an estimate 9 million developers, and is the driving force behind more than 7 billion devices. When Java was being developed simplicity and removal of many features that made its father languages C and C++ so complex. Java doesn’t use pointers, which causes a lot of bugs if not used properly, this alone makes java simpler.

**Section 2.6.2 C++ Simplicity**

A large language takes more time to learn, and C++ is a very large language, because of this programmers might learn only a subset of the language. C++ supports operator overloading, which as mentioned in the last section greatly affects readability and the simplicity of a language, because if used improperly can cause many bugs. Along with this C++ supports multiplicity which makes the language more complex.

**Section 2.6.3 C# Simplicity**

The main design goal of C# was simplicity, rather than having a powerful language such as C++. C# simplifies C++ by eliminating “irksome operators, such as “::”, and pointers. C# treats Boolean and integers as two different types. This means that the use of “=” instead of “= =” the complier will catch it.

**Section 2.7 Orthogonally**

The word orthogonal comes from the mathematical concept of orthogonal vectors, which are independent of each other. Orthogonality in a programming language means that a relatively small set of primitive constructs can be constructed in a relatively small number of ways to create the data structures and the control of the languages. Furthermore orthogonality means that features can be used in any combination.

**Section 2.8 Portability**

Portability in high-level computer programming is the usability of the software in different environments. The pre-requirements for portability is the generalized abstraction between the application logic and system interfaces.

**Section 2.7 Java Portability**

Java is the most portable of the languages that are being evaluated. Java has three different “packages”: ME for mobile, SE for desktops, and EE for enterprise. Java SE runs bytecode on any hardware that has the JVM (Java virtual machine). ME and SE are less portable for obvious reasons, because they are specific to certain devices.

**Section 2.7 C++ Portability**

Because of the sheer size of C++ it is hard to find a C++ standard. There are multiple versions and revisions of C++, the newer versions use the least compliant compilers. Compilers all add their own extensions and it’s all too easy to unknowingly use them.

**Section 2.7 C# Portability**

C# technically is technically portable using the “mono” IDE. Originally C# was just for Microsoft platforms, but in recent years it has moved towards other platforms. Mono IDE is not as good as the .NET IDE, but it has the advantage of being cross platform, which only make C# a better language.

**Section 2.8 Programming environment**

A programming environment is the collection of tools used in the development software. A collection could be as small as a file system, a text editor, a compiler, and a linker. A collection can also consist of a large collection of integrated tools, each accessed through a uniform user interface.

**Section 2.8.1 Java Programming environment**

NetBeans is a development environment that is primarily used for Java. NetBeans is not only an interpreter, it is also a framework, which means it provides common parts of the code of the application. Along with this NetBeans offers an easy to use UI builder; which is very useful when developing GUI interfaces.

**Section 2.8.2 C++ Programming environment**

There are many different programming environments made to program in C++. Some of these environments include Eclipse, IntelliJ Idea, and NetBeans. Most of these environments are free, but IntellJ idea is used for the professional world, meaning it costs a lot to use. The open source programs such as Eclipse still have great environments and IDEs to work with.

**Section 2.8.3 C# Programming environment**

Microsoft visual studio (.NET) is a development environment that includes a large and elaborate collection of software development tools, all of which are used through a windowed interface. Along with C#, Microsoft visual studio supports four other languages which include: Visual Basic .NET, Jscript, and C++.

**Section 2.9 Usage Cost**

The total cost of a programming is a function of many its characteristics. There are 5 features to consider when determining the cost of a programming language. First the cost of training programmers the language, secondly there is a cost of writing the language, third is the cost of compiling the language and the fifth and final factor is the cost of the language implementation system. All of these factors are the determining force behind a program’s cost.

**Section 2.1.9 Java Usage Cost**

Because Java is so extensive, there are a lot of features that need to be taught to programmers. There is no cost for compiling and writing because there are tons of open source compilers / interpreters that can be used for java, but in the case of business compilers that contain certain IDEs have a cost. Java programs, dependent upon how large they are and what they are written for, vary in cost for implementation.

**Section 2.2.9 C++ Usage Cost**

C++ is a more complex language to learn when being compared to C# and Java, thus training programmers to become fluent in C++ takes more effort than with the other programming languages. Like Java, C++ has compilers and interpreters that can be easily obtained for free, but there are environments for C++ that do have a cost such as IntelliJ, which is used primarily in business. Like in Java implementation cost all depends on how large the language is, and what it is written for.

**Section 2.3.9 C# Usage Cost**

C#’s syntax is very similar to that of Java, thus its learning curve is very similar to that of Javas. The compiler on the other hand is not. Visual Basic and the .NET bundle are charged for by Microsoft. The mono bundle (more on mono in the previous section), is a free compiler that has support for cross-platform. Like the other two languages implementation cost depends on what the usage is.

**3.0 Advanced Features**

**Section 3.1 Translation Process**

The translation problem involves conversions of instructions written in a high-level language program to machine readable instructions for implementation of the program. For this to happen the programming language must go through what is called a pass. There are two types of passes that will be discussed and these include two pass and three pass.

**Section 3.1.1 C++ and C# compiler**

Both C++ and C# use what is known as a two pass compiler. A two pass compiler checks for syntax errors, if there are none, than the compiler completes its final pass by generating the code. During the syntax analysis the compiler looks at the syntax rules of the language using Abstract syntax tree or a directed graph. Once this occurs the code is then generated into machine language.

**Section 3.2.1 Java Compiler**

Java uses a three pass compiler. A three pass compiler goes through three major states, these include lexical analysis, syntax analysis, and finally the code generation. Three pass is very similar to two pass, but the add step, lexical analysis, removes all of the irrelevant information from the source program, that will not be examined during the syntax analysis are removed, these include whitespace, and comments.

**Section 3.2 Data types, Variables, and support for abstraction**

A data type is a set of value(s) and operation(s) that are predefined within the programming language. The most commonly used data types include integers, whole numbers, floating points, Booleans, characters, and finally strings. Integers are positive and negative whole numbers, real numbers are those that include a floating point, Boolean allows for true and false outcomes, characters refer to alphanumeric data, and finally strings which are a sequence of characters.

Variables are names that are used by the programmer to label a data item. There are certain words in which a programmers cannot name their variables; these words are known as “reserved words”. These “reserved words” include primitive data types, and qualifiers. Abstract data types consist of enumerated types, arrays, record, pointers, and sub-range.

**Section 3.2.2 Java: Data types, Variables, and support for abstraction**

The java programming language is statically-typed which means that all variables must be declared before they are called. Declarations of a variable in java involve stating the variables type and the name. An example showing a declaration of a variable and how java is statically typed is as followed:

Figure 3.0

Java has eight primitive data types which include byte, short, int, long, float, double, Boolean, and char.

The byte data type is an 8-bit signed two’s complement integer, it has a minimum value of -128 and a maximum value of 127. The byte data type can be useful for saving memory to a large array. The short data type is a 16-bit signed twos compliment integer. It has a minimum value of -32,768 and a maximum value of 32,767. The int data type is 32-bits, by default, and has a maximum value of -232 and a maximum value of -232-1. The long data is 64-bit data type that has a maximum value of -263 and a maximum value of 263-1. The float is a 32-bit data type and the double is a 64 bit data type, both of these data have no precise maximum and minimum value. The char data type is a single 16-bit that has a minimum value of 0 and a maximum value of 65,535. Finally the Boolean data type has two possible outcomes, true or false. In addition to these primitive data types, java supports a String class. The following example shows how each of the primitive data types are declared:

Figure 3.1



this example shows how to declare a String:

figure 3.2

 Arrays, are one of the more commonly used abstract data types in java. An array is a list of data of a certain type. The below example shows how to declare an array in Java:

Figure 3.4



**Section 3.2.3 C++: Data types, Variables, and support for abstraction**

C++, like java, uses the same data types as java, but has unsigned integers. Unsigned integers are able to hold large positive values, but they are unable to negative numbers. Variable declaration in C++ is identical to Java, because Java is a language that was adapted from C++. C++ is different from Java in that it supports pointers, which gives the programmer more power. C++ has very similar abstract data types to java, such as pointers, arrays, and records.

The example below shows how a list is declared in C++:

Figure 3.6



**Section 3.2.4 C#: Data types, Variables, and support for abstraction**

C# may have the similar syntax to java, the way of declaring data types and variables is very different. Like C++, C# supports unsigned data types, along with the support of pointers. Because C# represents all primitive data types as objects, it is possible to call an object method on a primitive data type, for example:

Figure 3.7



Some Data types are called differently than in C++ and Java, which are similar in that sense, for instance in C#, in order to call a Boolean, the “keyword” used is “Bool”.

**Section 3.3 Expressions and Assignment Statements**

In all programming languages expressions and assignment statements are very important. Expressions are the fundaments means of specifying computations in a programming language. An expression typically is made up of a combination of variables, optional subprogram calls, and operators of the language. There are two types of expressions, there are Boolean operators, and there are arithmetic expressions.

Most of the characteristics of the arithmetic expressions in programming languages were inherited from conventions that had evolved in mathematics. Arithmetic expressions are used for evaluating values that will be used in assignment statements or other Boolean expression(s).

**Section 3.3.2 Java: Expressions and Assignment Statements**

Java programs store data values in variables. To make a variable in java the programmer must declare the type and name the variable, and then assign a value to it. Assignment statements in java use the assignment operator (=) to assign the result of an expression to a variable. An example of this is as followed:

Figure 3.8

this expression is an example of what is called a “compound assignment operator”. An assignment statement performs a calculation and an assignment simultaneously. With compound statements comes compound assignment operators. The following statements are used for compound statements:

Figure 3.9

|  |  |
| --- | --- |
| += | Addition and Assignment |
| -= | Subtraction and Assignment |
| \*= | Multiplication and Assignment |
| %= | Division and Assignment |
| /= | Reminder and Assignment |

In java an assignment is an expression and not technically a statement. Therefore abc = 5 , is technically an assignment expression and not an assignment statement, only when a semi-colon is added it becomes a statement.

**Section 3.3.3 C++: Expressions and Assignment Statements**

The main statement in C++ for carrying out computation and assigning values to variables is the assignment statement. The following example statement assigns a value to the variable “abc”:

Figure 3.10

as you can see assignment statements for all two languages are very similar. In C++ the type of operands of an arithmetic operator is very important, the list below provides the rules that C++ follows:

* If the expression is of type float and the result variable is an int than the float will be converted to an int.
* If the expression is of type int and the result is of type float than the int will be converted to a float.
* If both operands are ints then the result of the type is an int.
* If both or either of the operands are of the type float than the result is of type float.

The division of integers will result in an integer. If the correct float result is needed, then the compiler will generate code that looks at the expression as a float. If either of the operands is a const than it can be expressed as a floating point const, by adding a .0. to perform this; if “A” is an int, 1/n doesn’t give the reciprocal of n, to force a conversion, the programmer would have to use 1.0/n.

**Section 3.3.4 C#: Expressions and Assignment Statements**

Like C++ and Java, statements carry out assignment operations, which consist of taking the value on the right side of the assignment operator (=) and storing it in the variable on the left. example:

Figure 3.11

This expression assigns the value 1234 to the variable “abc”. The variable on the left must be able to accept and store values, meaning it must be a variable or property that is not “ReadOnly”, or it must be an array or list element. These variables are known in C# as lvalue, which means left value. The value on the right side of the assignment operator is constructed by the expression. The following example makes this concept clearer:

Figure 3.12



The above example add the variable b with the variable a and then adds the value returned by the call to function findResult, this is the stored in the variable x.

C#, on top of the addition of numeric values the assignment operator can also be assigned to strings values, and Boolean values. The following example shows how this is done in C#:

Figure 3.13

The above statement stores “Hello world!” to the values A and B. Below is an example of how C# handles storing Boolean values:

Figure 3.14

The above statement assigns the value false to “a” and “c”, and true to b. Similar to the above examples you can assign values to lvariable to “char”, “object” and “date” data types.

Like in Java and C++, C# also supports compound assignment statements. The += operator increments the right side. The following example shows the how this operator is used:

Figure 3.15

The above statement increments n, 23 times, and then stores the new value to “n”.

**Section 3.4 Control Structures**

Modern day programming languages are not restricted to a linear sequence of instructions. Control structures have a single entry and an exit point. According to structure theorem, any computer programming language can be written using the basic control structures, which include selection, and iteration. The iteration structure executes a sequence of statements repetitively until the given expression returns false. Finally, the selection structure dependent upon if the condition is true or false. Because all the languages being discussed are c-based the way that they declare control structures are nearly identical.

Iterations structures are widely used in C#, C++, and java. The most basic iteration statement is the while statement. The while statement can be implemented in all of the three languages that are being discussed. The syntax for the while statement is as followed:

Figure 3.16



Before the while loop is executed, the condition will be tested. If the condition is false, the while statement will not execute, and the program will skip to the line after the while statement. There is another loop, where you can count the iterations that a specific statement does through, this is known as a “for” loop. The for loops syntax is as followed:

Figure 3.17



All variables in the “for” loop must be integers. If the end integer > start integer then the loop is going to execute end – start times. If the end < start than the loop will never start.

The Selection statement causes the program control to be transferred to specifically flow based upon whether a certain condition is true or not. The types of switch statements that each of the languages employees is as followed: the “if” statement, “else” statement, “switch” statements and “case” statements. The following are examples of the “if” statement and the “switch” statement. The switch statement is as followed:

Figure 3.18



the body of a switch statement is known as a switch block. A statement in the switch block can be labeled with one or more case or default labels. The switch statement evaluates its expression, then executes all statements that follow the match case label. The “if” statement is as followed:

figure 3.19

if(condition)

{

Statements

}

An “if” statement identifies which statement to run based on the value of a Boolean expression.

**Section 3.5 Subprograms**

Sub-programming is a very important concept when it comes to programming design. There are two fundamental abstractions facilities can be included in programming language: process abstraction and data abstraction. There are three fundamentals to sub programming, these include: each subprogram has a single entry point, there are only one sub program executing at a time, and control always returns to the caller when the subprogram executes. There are two ways for a subprogram to gain access to the data that has to be processed, this is through direct access to nonlocal variables or through the passing of parameters. There are three modes to parameter passing these include in mode, out mode, and inout mode (both). The implementations to these include pass by value (in), pass by result (out), pass by value-result (inout), pass by reference (inout) and Pass by Name (this varies).

Programming languages, in particular C++, not only provide a set of basic operations and statements, but C++ also provides a way to define operations and statements. These are called functions and procedures or subprograms. Within these functions and procedures there might be parameters. These parameters represent the objects from our program that are used in the subprogram. An example of a function call in C++ goes as followed:

Figure 3.20



breaking down this function goes as follow: the type of the result comes first, the int. Second it is time to name the function, in this case we named it “hello”. Next inside the parenthesis are the parameters.

Next we will examine how C++ handles subprograms:

Figure 3.21



C# and Java handle sub programming very similarly to that of C++.

**Section 3.6 Exception Handling**

An exception is a problem that arises during the execution of a program. An exception can occur for many different reasons. The following are examples on what could cause an exception within a program:

* A user has entered an invalid data
* A file that needs to be opened cannot be found
* A network connection has been lost in the middle of communications or the JVM has run out of memory.

Some exceptions can be caused by the user of the program, the programmer, or the physical resources that may have failed.

**Section 3.6.2 Java Exception Handling**

All exception classes are subtypes of the java.lang.Excetion class. The exception class is a sub class of the throwable class. Other than the exception classthere is another subclass called Error which is derived from the throwable class. Below are a list of methods that are available through the throwable class:

Figure 3.22

This method returns a detailed message about the exception that has occurred. This message is initialized in the throwable constructor.

Figure 3.23

 This method returns the cause of the exception as represented by a throwable object.

Figure 3.24

This method of the throwable class returns the name of the class concatenated with the result of the getMessage().

Figure 3.25

This method fills in the stack trace of this throwable object with the current stack trace, adding to any previous information in the stack trace.

Figure 3.26

This method returns an array containing each element on the stack trace. The element at the index 0, represents the top of the call stack, and the last element in the array represents the method at the bottom of the call stack.

Java uses the same methodology for catch exceptions are C++, and that is the try / catch statement. The syntax is the same as C++ for the try / catch statement, this is illustrated in the next section.

**Section 3.6.3 C++ Exception Handling**

A C++ exception is a response to an exceptional circumstance that arises while a program is running. Exceptions provide a way to transfer control from one part of a program to another. C++ handling is built upon three keywords: try, catch, and throw statements.

* A “catch” statement catches an exception with an exception handler at the place in a program where the programmer wants to handle the problem. The catch keyword indicated the catching of an exception
* The “try” block identifies a block of code for which a particular, exceptions will be activated, it is followed by one or more catch blocks. Below is an example of a “try statement”:

Figure 3.27



* The “throw” statement throws an exception when a problem shows up. This is done using a throw keyword. Below is an example of a “throw” statement:

Figure 3.28



**Section 3.6.4 C# Exception Handling**

Like java C# has exception classes. The exception classes in C# are mainly directly or indirectly derived from the System.Exception class. Some of the classes that come from the System.Exceotion class are the System.ApplicationException and the System.SystemException classes. The following are examples of each of the exceptions that come from the System.Exception class:

Figure 3.29

This class handles input and output errors.

Figure 3.30

 This class handles errors generated when type is mismatched with the array type.

Figure 3.31

 This class handles errors generated when a method refers to an array index out of range.

Figure 3.32

 This class handles the errors that happen from referencing a null object

Figure 3.33

 This class handles the errors that happen from dividing a dividend with zero.

Figure 3.34

 This class handles the errors that happen during typecasting.

Throwing objects in C# follows the same methodology as C++ and Java.

**4.0 Summary and Concluding Remarks**

All of the languages that have been evaluated serve a purpose. Java, a language used all over the world, is a very versatile language and is the go to language in terms of enterprise computing. Anecdotal evidence shows that about 90 percent of fortune 500 companies use java. C#, like java and C++, is a general purpose language that is used from business applications to scientific applications, and finally C++ is a language that is competing with Java as the two forerunners in business applications, and ruled the domain for a while until Javas multiplatform abilities were truly taken advantage of. All of these programming languages will be widely used in the future, because of all of the legacy systems that they are currently run. All languages are headed towards dealing with big data, and as the internet of things is becoming larger and larger a support for multiplatform is going to be needed for all languages, and that why I think Java will prevail among C++ and C#, and that is because it has the cross-plateform abilities

References

[Foster 2017a] Foster, Elvis C. 2017. “Introduction to Programming Languages.” In Lecture Notes in Principles of Programming Languages. Accessed April 24, 2017. <https://www.elcfos.com/lecture-series/index/entry/id/10/title/Principles-of-Programming-Languages>

[Foster 2017b] Foster, Elvis C. 2017. “Chronology of Programming Languages.” In Lecture Notes in Principles of Programming Languages. Accessed April 24, 2017. <https://www.elcfos.com/lecture-series/index/entry/id/10/title/Principles-of-Programming-Languages>

[Foster 2017c] Foster, Elvis C. 2017. “Overview of a compiler.” In Lecture Notes in Principles of Programming Languages. Accessed April 16, 2017. <https://www.elcfos.com/lecture-series/index/entry/id/10/title/Principles-of-Programming-Languages>

[Foster 2017d] Foster, Elvis C. 2017. “Translation issues.” In Lecture Notes in Principles of Programming Languages. Accessed April 20, 2017. <https://www.elcfos.com/lecture-series/index/entry/id/10/title/Principles-of-Programming-Languages>

[Foster 2017e] Foster, Elvis C. 2017. “Data types and variables.” In Lecture Notes in Principles of Programming Languages. Accessed April 25, 2017. <https://www.elcfos.com/lecture-series/index/entry/id/10/title/Principles-of-Programming-Languages>

[Foster 2017F] Foster, Elvis C. 2017. “Expressions.” In Lecture Notes in Principles of Programming Languages. Accessed April 24, 2017. <https://www.elcfos.com/lecture-series/index/entry/id/10/title/Principles-of-Programming-Languages>

[Foster 2017G] Foster, Elvis C. 2017. “Control Structures.” In Lecture Notes in Principles of Programming Languages. Accessed April 24, 2017. <https://www.elcfos.com/lecture-series/index/entry/id/10/title/Principles-of-Programming-Languages>

[Foster 2017H] Foster, Elvis C. 2017. “Introduction to Programming Languages.” In Lecture Notes in Principles of Programming Languages. Accessed April 24, 2017. <https://www.elcfos.com/lecture-series/index/entry/id/10/title/Principles-of-Programming-Languages>

[Foster 2017I] Foster, Elvis C. 2017. “Support for Abstract data types.” In Lecture Notes in Principles of Programming Languages. Accessed April 24, 2017. <https://www.elcfos.com/lecture-series/index/entry/id/10/title/Principles-of-Programming-Languages>

[Foster 2017J] Foster, Elvis C. 2017. “Support for object-oriented programming.” In Lecture Notes in Principles of Programming Languages. Accessed April 8, 2017. <https://www.elcfos.com/lecture-series/index/entry/id/10/title/Principles-of-Programming-Languages>

[Foster 2017K] Foster, Elvis C. 2017. “Concurrency Control.” In Lecture Notes in Principles of Programming Languages. Accessed April 24, 2017. <https://www.elcfos.com/lecture-series/index/entry/id/10/title/Principles-of-Programming-Languages>

[Foster 2017a] Foster, Elvis C. 2017. “Introduction to Programming Languages.” In Lecture Notes in Principles of Programming Languages. Accessed April 8, 2017. <https://www.elcfos.com/lecture-series/index/entry/id/10/title/Principles-of-Programming-Languages>

[Sebesta 2017a] Sebesta, Robert C. 2017. “Concepts of programming Languages.” Tenth addition. Accessed April 20, 2017.

[Mensah 2017a] Mensah, Kuassi C. 2006. “Oracle Database Programming using Java and Web Services - 2006” Accessed April 20, 2017. <https://docs.oracle.com/cd/A87860_01/doc/appdev.817/a77069/03_struc.htm>

[Masashi 2017a] Masashi, Ohba-itaru sellowiine C. 2004. “[Sellowiine 2017a] sellowine C. 2004. “deethylsuaveoline, the Struc- ture Proposed for Sellowiine” Accessed April 20, 2017. <https://docs.oracle.com/cd/A87860_01/doc/appdev.817/a77069/03_struc.htm>

[Krill 2017a] Krill, Paul C. 2010. “Four reasons to stick with Java, and four reasons to dump it” Accessed April 21, 2017. http://www.javaworld.com/article/2689406/java-platform/four-reasons-to-stick-with-java-and-four-reasons-to-dump-it.html

[unknown 2017a] unknown C. 2009. “[unknown 2017a] unknown C. 2009. “Java Exceptions” Accessed April 20, 2017. <https://www.tutorialspoint.com/java/java_exceptions.htm>

[Wagner 2017a] Wagner, Bill C. 2010. “Exception Handling (C# Programming Guide)” Accessed April 22, 2017. <https://docs.microsoft.com/en-us/dotnet/articles/csharp/programming-guide/exceptions/exception-handling>

[unknown 2017a] unknown C. 2011. “[unknown 2017a] unknown C. 2011. “The Java Language Environment” Accessed April 20, 2017. <http://www.oracle.com/technetwork/java/simple-142339.html>

[unknown 2017a] unknown C. 2005. “[unknown 2017a] unknown C. 2005. “Control Flow Statement” Accessed April 20, 2017. <https://docs.oracle.com/javase/tutorial/java/nutsandbolts/flow.html>