COMPARISON OF OBJECT-ORIENTED AND PROCEDURE-BASED COMPUTER LANGUAGES: CASE STUDY OF C++ PROGRAMMING

Kuan C. Chen, Ph.D.
Assistant Professor
Management Information Systems
School of Management
Purdue University Calumet
E-mail: kchen@calumet.purdue.edu

ABSTRACT

C++ is a hybrid computer programming language, which includes the characteristics of procedure-based and object-oriented. Most studies in the comparison of procedure-based programming and object-oriented languages are focused on the algorithm or scientific applications. In this paper we strive to compare the characteristics of the object-oriented and procedure-based language with implications for business applications. Data anchor and structures applications in the business information system design are demonstrated and also are used to compare the performance between procedure-based and object-oriented languages. In this experiment we will test only basic operations and not the performance of complex systems such as the speed of reading and modifying data. The outcome of this study will serve as a future technical analysis reference for software business application in designing information systems.

Keywords: C++, procedure-based, object-oriented, data anchor, structures, business application

INTRODUCTION

Although object-oriented languages have been around since the 1960s, the past 10 years have seen unparalleled growth in the use and acceptance of object technologies throughout the software industry (7). Many programmers and businesses have now embraced the concept of object-oriented programming (OOP) for their tasks. Some have gone as far to make this their only method of developing information systems. However, there are several advantages and disadvantages to using OOP over traditional procedure-based programming techniques that cannot be overlooked. In general, OOP leads to faster programming development (programmer efficiency), whereas procedure-based programming will lead to faster programs (runtime efficiency).

In this paper we will compare the characteristics of the object-oriented and procedure-based language using data anchor as the business application case. Specifically we will review these two kinds of languages from performance and applications aspects. Then, we will try to quantify the runtime performance differences of OOP and procedure-based operations using the C++ programming language in data anchor. C++ is very useful in that it allows for the development of both varieties, whereas the code of other languages such as Java are technically always in object oriented format. In this experiment we will test only basic operations and not the performance of complex systems such as the speed of reading and modifying data. The outcome of this study
OVERVIEW OF PROGRAMMING LANGUAGES

Procedure-based Programming

All of the first programming languages were procedure-based, such as FORTRAN, COBOL, APL, PL/1, Ada, and Basic. Procedure-based programming focuses on how to accomplish a task. The programmer must develop the step-by-step to instruct the computer to follow. The steps are from the start of the task to its completion. The programmer determines and controls the order in which the computer processes the instructions. Thus, the order of the instructions is extremely important. The general process of developing a system using procedure-based and Object-oriented approach is shown in Figure 1.

Figure 1: Design comparison of procedure-based and Object-oriented

Procedure-based language tends to produce programs that run quickly and use system resources efficiently. It is a conventional approach that is understood by many programmers, software engineers, and system analyst. However, the drawback of the procedure-based language is that does not fir gracefully with certain types of problems, such as it forces the programmers to view problems as a series of steps; however, some problems might better be visualized as interacting objects or as interrelated words, concepts, and ideas.

Object-oriented Programming

Object-oriented Programming (abbreviated in what follows to OOP) is an approach to formulate program as a series of objects and methods that interact to perform a specific task. It can be used in most languages, but works best in one that was meant to be OOP, like Java, C++, or C#. It is an entirely different concept from 'procedure-based' programming, which languages like C and BASIC are. A lot of the interesting research work on OOP is hard to apply to business needs, while business badly needs technologies which can ease the burden of developing and maintaining application code (4).
The object-oriented program is cognitively similar to the way human beings perceive the real world (5). Using the object-oriented approach, programmers may be able to visualize the solutions to problems more easily. Facets of the object-oriented program can also increase programmer’s efficiency because encapsulation allows objects to be adopted and reused in a variety of different programs.

A potential disadvantage of object-oriented program is runtime efficiency. Object-oriented programs tend to require more memory and processing resources than procedure-based programs. Programmers, software engineers, and system analysts can work together to weigh the tradeoffs between the OOP and runtime efficiency. This is premise of this study to observe the performance between OOP and procedure-based languages.

**Summary of Comparisons**

First, in procedure-based programming the programmers code program based on what the program does and when. Most of the time, we have variables and functions that are global (this is in C); the functions of course have their own variables that are localized, but besides that, everything is system wide. This is quite different from OOP. In OOP we are programming based on 'Objects', which I will explain in detail later. Each object has its own methods (functions) and variables. Most variables are not global, though in some languages, method names are (it is important to realize however that in OOP we can be sure what object is being told to perform that method).

**CASE STUDY**

**Characteristics of C++ Language**

The C++ programming languages is an extension of C that was developed by Bjarne Stroustrup in the early 1980s at Bell Laboratories. C++ provides a number of features that "spruce up" the C language, but more importantly, it provides capabilities for object-oriented programming. C++ is a hybrid language—it is possible to program in a C-like style, an object-oriented style, or both. C++ programs consist of pieces called classes and functions. You can program each piece you may need to form a C++ program. The advantage of creating your own functions and classes is that you will know exactly how they work. You will be able to examine the C++ code.

There are four major elements in C++ object-oriented language, which are data abstraction (reuse code), encapsulation, inheritance (extend reused code), polymorphism (dynamically modify reused code).

Data abstract provide an important function in computer programming. They are a hedge against complexity, allowing the programmer to concentrate on the essential attributes of a category and ignore all subordinate attributes.

Encapsulation refers to the process of hiding the internal details of objects and their methods. Once an object is coded, it becomes a “black box,” which essentially hides its data from other objects and allows the data to be accessed via methods. Encapsulation objects can be easily reused, modified, and repurposed.
Polymorphism is the ability to redefine a method in a subclass. It allows programmer to create a single, generic name for a procedure that behaves in unique ways for different classes. Inheritance refers to passing certain characteristics from one class to other classes.

**Benchmark**

In this study, we implemented the test in Microsoft Visual Studio 6.0 environment. Although the Visual Studio has Dot Net today, the old environment has the capability to create the object-oriented and procedure-based programming language test environment. Also, we used the Personal Computer Pentium 4-3GB with 1 Gaga Byte Memory. We tested the following scenarios:

**Anchor**

In the business application we used the data anchor to test the performance between procedure-based and object-oriented C++ programming. Data anchor is derived from the Anchor, and whose nodes are data tables. The Anchor is a handle to generic Link List class. It is doubly linked, having both forward (next) and reverse (previous) links. “Head” and “Tail” nodes are maintained within Anchor structure. Each node in the Anchor must be the same size, but the actual data inside a node is up to the application. The Anchor has a “Current Node” concept; i.e. functions that operate at the node level work with the node which has been designed as the “Current Node.” There is only one current node, and only few ways to change which node is the current node: Rewind, FastForward, Advance, and Backup.

![Figure 2. A visual representation of an Anchor](image-url)
**Structures**

In either procedure-based or object-oriented C++ the programmer can create some aggregate data-types called structures, and this is the most interesting data-type in C++. Structures are the things that most resemble an object in object-oriented program, (i.e. JAVA). A structure is a variable containing several named member variables, like this:

```cpp
struct foo {
    int bar;
    char baz[4];
};
```

This creates a structure with the name struct foo (the struct keyword becomes part of the name) with the members "bar" and "baz". To define a variable of this type, it can be written as:

```cpp
struct foo MyFoo;
```

To access the members of this structure, you write:

```cpp
MyFoo.bar = 6;
```

Now you see why structures superficially look like objects. Structures are just what they say; they are structures lumping several other named variables together in a unit.

**Test Procedure**

The first step in testing OOP is to think in terms of objects instead of procedures. To begin with, each object should be a noun. This is about the most important thing in the OOP approach. Each object then contains actions it can perform, and these should all be verbs, as would be in normal English. Then we have what are called instance variables, these are 'object globals' which are a part of every instance (occurance) of that object. Think of these as an integral part of a thing, like your hand is a 'part' of your body.

In procedure-based C++, you would have functions that are global, that deal with char pointers indifferent ways. You could put these in a separate file, but if later you forgot and called some other function the same name; then things could explode. In OOP, and object-oriented C++, that is not a concern.

The next step is creating a char pointer. It has characters and a length, so the instance variables will have to be just that.

```cpp
{
    char *string; /* All the syntax in this intro is done
    int length;  * in Objective C, but the concepts apply
    * to all OOP languages. */
}
```

In the process we need to be able to define this object. What string is, and how long it is. So we will want to be able to tell it to put data there. Here are a couple of methods this object could perform:
CONCLUSION

In this study we provide a milestone process for the programmer or system analyst to select the language structure. Since the benchmark is hard to standardize, the end result of the test becomes doubtable. The fundamental goal for this research is based on the literature reviews to build the business application case for comparisons of procedure-based and object-oriented programming language using C++. Due the space limits, we only can briefly describe the concept and general processes. For the test result, we summarize three key points:

1. Although OOP is the trend to programming language, the data structure in business application is an important factor to be considered. In general most business applications are the emphasis on the database management to information system development. Unlike the science, it focuses on the algorithm. So prior to using the language, data structure in application should be the first element for choosing either the procedure-base or object-oriented.

2. Benchmark is another key factor to testing and selecting the language programming structure. For the purpose of future research, we recommend to use different platforms and code structure. In this study we could not compare the giant system. Based on the previous research literatures, OOP has the strong case in effectively and efficient to a large system. However, there is also some augments about the performance (run time) issue.
3. In this study we used procedure-based and object-oriented C++ for testing. The reason for using the same language because C++ can implement two structures. In the future it needs to identify the test with other OOP language, such as C# and JAVA.

REFERENCES