Programming Paradigms In C++

Subhash khalkho & Sunny kumar
Department of IT, 3rdsem, DCE, Gurgaon , Haryana
subhash.16938@ggnindia.dronachary.info ; sunny.16941@ggnindia.dronacharya.info

ABSTRACT
This paper discuss about the generation of programming types evolved in C++.The major reason behind the popularity and success of C++ is that it supports the object oriented technology which is among best in software development and it seems real to the existing world, C++ supports various types of programming paradigms i.e, procedural programming style, object based programming styles, object oriented programming styles. This paper discusses all the key featuresof all given programming styles and their implementations along with their merits and disadvantages.

INTRODUCTION

A programming paradigm defines the method of designing and implementing programs using a key feature of a programming language. A programming paradigm provides an idea that how problems are generally understood and then solved using a particular programming language**.**

PROCEDURAL PROGRAMMING

Procedural programming gives stress on procedure rather than on data. Procedural programming is derived from structured programming, which is based upon the concept of theprocedure call. Procedures, also known as routines, methods, or functions simply contain a series of computational steps to be carried out. Any given procedure might be called at any instance during a program's execution, included by other procedures. Procedural programming is a list of instructions telling a computer what to do step by step and how to perform from the one code to the other code.

Procedural programming languages include:

C, Go, Fortran, Pascal, and BASIC.

As we know C++ is subset of C, which uses procedural programming style, therefore C++ can also be used as procedural programming language but with some inherited features like default arguments, type checking, inline functions etc.

Procedural programming paradigm completely separates the functions in program from the data manipulated by those programs, and this where it gets complicated and give rise to various issues in order to extend the features of the software and to maintain it.

In procedural program if type of data is changed then the function using this data must also be changed for proper flow of program.

For Example:

```
Struct student
{       introllno;
        Char name[30];
        intclas;
};
Void readstudent(student s1)
{       cout<<”Enter rollno of student”;
        cin>>s1.rollno;
        cout<<”Enter name of student”;
        gets(name);
        cout<<”Enter class of student”;
        cin>>clas;
}
```

Now suppose user want to upgrade this program by making this program to hold marks and grade of student by modifying the structure student. So now,

Struct student

```
{       introllno;
        char name[25];
        intclas;
        float marks;
        char grade;
};
```

Now the functions using structure student must also be updated to keep up with the changes made in structure student, thus the function readstudent() needs to be rewritten again. Similarly if we had ten more functions that were using 'student' then they must also be updated or modified.

DISADVANTAGES

i. Procedural programming is prone to design changes and modifications.
ii. Procedural programming leads to increased time and high cost during design change and modification.

OBJECT BASED PROGRAMMING

Object based programming is a new programming paradigm that implements few features of object oriented programming but not all of it. In object based programming data and its functions are enclosed within one single packet called class. A class provides data hiding and data abstraction and thus separate implementation details (how process actually happening) and only concern with the interface (how the user uses it).For example imagine a car, you know how to increase speed of car i.e just putting pressure on transmission pedal but here you are not concerned with the actual process taking place inside the engine to result for transmission, that means the transmission pedal is an interface between the driver an the transmission in engine. In other words the user is allowed to access the interface (transmission pedal) but he cannot access the actual implementation details inside the engine for transmission.

Example of object-oriented languages include:

Simula, Smalltalk, C++(whose object model was based on Simula's), Objective-C (whose object model was based on Smalltalk's), Eiffel, Python, Ruby, Java, C#.

In object based programming paradigm whenever there is a change in type of the definition users interface remains unaffected.

For Example:

```
Class student
{       introllno;
        char name[25];
        intcls;
        float marks;
                public:
                        voidreadstudent();
                        voiddispstudent();
};
```

Now suppose you want to add some data members to class like grade and fathers name then also the users interface would remain same as earlier.

```
Class student
{       introllno;
        char name[25];
        intcls;
        float marks;
        float grade;
        charfname[25];
        public:
                voidreadstudent();
                voiddispstudent();
};
```

Hence there would be no change in users interface and user would never know that a new data member has been added as the interface remains same for him, but there would be certain changes in the member functions by programmer in readstudent() and dispstudent(). But here is one difference, the user cannot have a direct access to the data members of class student which was possible in procedural programming paradigm.

ADVANTAGES:

Object based programming is subset of object oriented programming and hence it has some similar properties like it:

i. It overcomes most of the shortcomings that were present in procedural programming.
ii. It modifies and hides process implementation details from user.
iii. It supports user defined types of program.
iv. It provides data abstraction.

Having all these advantages object based programming suffers from one major drawback and that is the inability to relate real world relationship that exists among objects around us. For example chair and sofa both are furniture but this relationship cannot be represented in object based programming as it does not support inheritance**.**

OBJECT ORIENTED PROGRAMMING (OOPS)

Object oriented programming paradigm combines both the features of procedural and object based programming styles. It uses all features of object based programming and overcomes its limitations by applying inheritance so it can easily relate to the real world and among its objects. Object Oriented Programming paradigm mainly uses class and objects. In which object may be represented as a run time entity that may be any name, person, place or anything and class represents a collection of data or objects that share common property.

Examples of object oriented programming:

C++, Objective-C, Smalltalk, Delphi, Java, Javascript, C#, Perl, Python, Ruby and PHP.

For example we have a car then its characteristic is wheel, brake, motor, seats etc, and their behaviour is mobility and comfort. Therefore a class is a derived class of base class automobiles which is further a derived class from base class vehicles.

Basic concepts of OOPS:

i. DATA ABSTRACTION: It is an act of displaying essential features without including background details.

ii. DATA ENCAPSULATION: It can be explained as an act of wrapping up of data within a single unit.

iii. MODULARITY: It is an act of dividing a program into individual components.

iv. INHERITANCE: It is the process by which one class acquires the property of another class.

v. POLYMORPHISM: It means one name and many forms, it is the ability to take more than one form and operation may exhibit different behaviour in different instances.

ADVANTAGES

i. RE-USABILITY OF CODE: Linking of code to object and relation between objects allows related objects to share code and Encapsulation allows class definition be re-used in other application.

ii. EASE IN COMPREHENSIN: OOPs code is more close to existing real world than other programming paradigms code.

iii. EASY TO DESIGN: The concepts like data abstraction and encapsulation provides a way for clean designing of program.

iv. EASY TO EXTEND: Facilities like modularity, inheritance, polymorphism, data encapsulation and abstraction provides an easy way for program extension and modification.

CONCLUSION

All four of the main programming paradigms are useful in their own way, but pure programming languages of only one paradigm are known to be slightly more limiting but Object-oriented design is currently the most versatile and widely used programming paradigm.

REFFERENCE

1. Stroustrup, B. (1995). *The C++ programming language*. Pearson Education India.

2. Chandy, K. M., & Kesselman, C. (1993). *Compositional C++: Compositional parallel programming* (pp. 124-144). Springer Berlin Heidelberg.

3. Coplien, J. (1997, November). Advanced C++ programming tylesand idioms. In *Technology of Object-Oriented Languages, International Conference on* (pp. 352-352). IEEE Computer Society.

4. Watson, M. (1994). *C++ Power Paradigms*. McGraw-Hill, Inc..

5. Burrus, N., Duret-Lutz, A., Géraud, T., Lesage, D., & Poss, R. (2003, October). A static C++ object-oriented programming (SCOOP) paradigm mixing benefits of traditional OOP and generic programming. In *Proceedings of the Workshop on Multiple Paradigm with OO Languages (MPOOL), Anaheim, CA, USA*.

6. Coplien, J. O. (1999). *Multi-paradigm Design for C+*. Addison-Wesley.

7. Gregor, D., Järvi, J., Siek, J., Stroustrup, B., Dos Reis, G., & Lumsdaine, A. (2006, October). Concepts: linguistic support for generic programming in C++. In *ACM SIGPLAN Notices* (Vol. 41, No. 10, pp. 291-310). ACM.