UNIT-1: INTRODUCTION TO DIFFERENT PROGRAMMING PARADIGMS

Faculty In-charge

Ms. Aaysha Shaikh
Assistant Professor (IT Dept.)
Room No. 322
email: aayshashaikh@sfit.ac.in

Academic Year: 2022-23

OUTLINE OF SYLLABUS

Introduction to programming paradigms and core language design issues [1]

Imperative Paradigm: Data abstraction in object orientation [2]

Declarative programming paradigm: Functional 3 programming

Declarative programming paradigm: Logic 4

programming

5 Alternative paradigm: Concurrency 6 Alternative paradigm: Scripting Languages

OUTLINE OF UNIT-1

1.1 Introduction to different programming paradigms

Names, Scopes, Bindings, Scope Rules, Storage

Management 1.2

Type Systems, Type checking, Equality testing, and

assignment 1.3

Subroutine and control abstraction, Stack layout, 1.4

calling sequence, parameter passing

Generic subroutines and modules, Exception

handling, 1.5

co-routines and events

1.1-INTRODUCTION TO DIFFERENT PROGRAMMING PARADIGMS

Any fool can write code that a computer can understand.
Good programmers write code that human's can understand
-Martin Fowler

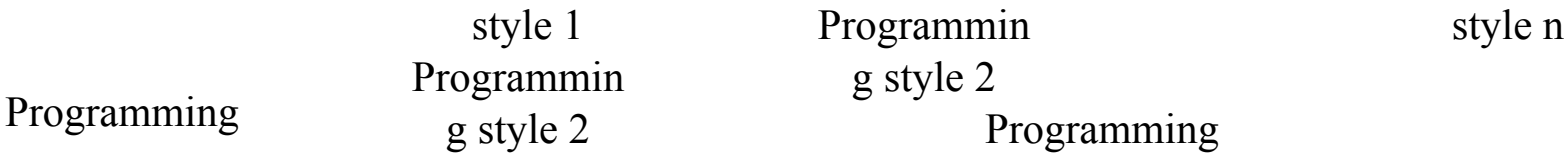
(Martin Fowler is a British software developer, author and international public speaker on software development, specializing in object-oriented analysis and design, UML, patterns)

1. When programming , complexity is always the enemy

2. Managing complexity is a programmer’s main concern

What is Programming Paradigm?

Definition: Programming paradigm is a style or a "way" of programming. Some languages make it easy to write programs in some paradigms but others do not.



You have different "styles"/ "ways"/ **"Paradigms"** of programming to tackle

the issues of handling complexity

Are Concrete **Are way of doing** Programming Programming

Languages

Paradigms

IMPERATIVE PROGRAMMING

#(first do this then do that…….)

. The imperative paradigm is the oldest and the most popular programming paradigm

. Based on the von Neumann architecture of computers (https://www.javatpoint.com/von neumann-model)

. Imperative programs define sequences of commands/statements for the computer that change a program state (i.e., set of variables)

- Commands are stored in memory and executed in the order found
- Commands retrieve data, perform a computation, and assign the result to a memory location

. The hardware implementation of almost all computers is imperative

. Machine code which is naïve to the computer hardware is written in imperative style

IMPERATIVE PROGRAMMING

Program: Sum of first 5 natural numbers in C #include<stdio.h>

```
int main( )
{
int sum=0;
sum+=1;
sum+=2;
sum+=3;
sum+=4;
sum+=5;
printf("The sum is: %d/n", sum);
return 0;
}
```

Data

Most closely resembles the actual machine itself

The order of steps is very important

Given step will have different consequences depending on the current values of the variables when the step is executed

MEMORY

(Data and Program) **CPU** **Address**

Are there any CENTRAL elements of imperative paradigms

YES!!! YES!!!!

Assignment Variables Statement

Flow control

Step by step execution

Examples of imperative programming language (languages that follow imperative programming paradigm)

C: Developed by Dennis Ritchie and ken Thompson

FORTAN: Developed by John Backus from IBM

Basic: Developed by John Kemeny and Thomas E Kurtz

PROCEDURALPROGRAMMING

(for repetitive job)

. Procedural programming is a **refinement** of the imperative paradigm adding

subroutines (or procedures)

. Procedures can be used the same way that built-in commands are used (allows re usability)

?? What is a procedure

. Set of subroutines

. May or may not return a value

In a program for drawing shapes, the program could ask the user what shape to draw. The instructions for drawing a square could be captured in a **procedure**.

The algorithm for this action could be a set of tasks, such as these:

Repeat the next two steps four times:

Draw a line of length n.

Turn right by 90 degrees.

If this were a computer program, this set of instructions could be given the name '*square*' and this sequence would be executed by **running** (calling) that **procedure**.

Example of computing the factorial of a number:

IMPERATIVE

```
int n = 5;
int result = 1; while(n > 1)
{ result *= n; n--;
}
```

return type
PROCEDURAL

Forming a procedure

```
int factorial(int n)
{
int result = 1;
while(n > 1) {
result *= n;
n--;
}
return result;
}
```

Returning a value

Introduce procedure, have

Procedure will be called from main/home

Example of computing the addition of n numbers : (n=5)

PROCEDURAL IMPERATIVE

```
#include<stdio.h> int main( )
{
int sum=0;
sum+=1;
sum+=2;
sum+=3;
sum+=4;
sum+=5;

printf("The sum is: %d/n", Sum); }
return 0;
}
```

Introduce procedure, have return type

Main can be a procedure

```
#include <stdio.h>
int main()
{
int sum=0;
int i=0;
for (i=1;i<=5;i++) {
sum+=i;

printf("The sum is:%d\n", sum) return 0;
}
```

Returning a value

Examples of procedural programming language (languages that follow procedural programming paradigm)

C: Developed by Dennis Ritchie and ken Thompson

C++: Developed by Bjarne Stroustrup at Bell Labs

JAVA: Developed by James Gosling at Sun Microsystems

ColdFusion: Adobe, Joseph J.Allaire, Macromedia

OBJECT ORIENTED PROGRAMMING PARADIGM

. An approach to the solution of problems in which all computations are performed in the context of objects

. The program is written as a collection of classes and objects.

. The smallest and the basic entity is object

. Emphasis is on data rather than procedure

. Methods that operate on the data of an object is tied together in the data structure

. Data is hidden and cannot be accessed by external function

. Objects may communicate with each other through methods

. Follows bottom up approach in program design

. **Ruby, Java, C++, Python, Simula**

Example of a class:

```
Class Account
{
 int account_number;
 int account_balance;
 public void showdata()
 {
 system.out.println("Account Number"+account_number)
 system.outprintln("Account Balance"+ account_balance)
 }
}
```

Here you have a Class named as: Account

Attributes are : account_number, account_balance

Method/Action is : showdata()

```
Public static void main(String args[]) { Account obj= new Account()
   obj.account_number=A00000010;  obj.account_balacnce=5000;
            obj.showdata();
}
```

OBJECT

Data Members: id, name

Member function: getdata(), display()

- **Aclass acts as a blueprint for objects and basically defines a type**
- **Parts declared private are not accessible from the outside**
- **Parts declared public are visible to all**

So classes accomplish encapsulation and information hiding

OBJECT ORIENTED PROGRAMMING PARADIGM

Advantages:

- Data security
- Inheritance
- Code reusability
- Flexible and abstraction is also present

DECLARATIVE POGRAMMING PARADIGM

. The style expresses the logic of a computation without talking about its control flow

. It defines what needs to be accomplished by the program without defining how it needs to be implemented

Mam……pls tell

IMPERATIVE WHAT to do and HOW to do

Mam……Only tell

DECLARATIVE

FORGET how to do

St. Francis Institute of Technology
Department of Information Technology

WHAT to do and

IMPERATIVE: Provides instructions for assembly

DECLARATIVE: Provides a

picture of finished piece as a template

The more sophisticated the application, the greater the danger that the code becomes so convoluted that it can only be read by the developer who originally wrote it

Examples of declarative programming paradigm

Prolog : Developed by by Alain Colmerauer at the University ofAix-Marseille, France

Haskell : First proposed by Philip Wadler and Stephen Blott

Miranda : Developed by David Turner Research Software Ltd.

Advantages:-

. Short, efficient code

. Easy optimization as implementation is controlled by an algorithm

FUNCTIONALPROGRAMMING

In functional programming we write the function exactly as mathematical function **f(x)=x+1**

Immutable means non modifiable, mutable means modifiable

FUNCTIONAL PROGRAMMING

Functional programming is a programming paradigm where you have a style of building the structure and elements of computer programs. Here you treat computation as an

evaluation of mathematical functions and you avoid changing-state and mutable data. Functional programming consists only of PURE functions. So, what do you understand by Pure functions?

Features of Functional Paradigm

- **Pure functions** – As seen above, if the input is an array, the output will be a new array and the input array will not be modified. So in case of pure functions, the output depends only on the input.

Here's a function in the language Scala that takes values and returns their sum.

```
scala> def add(a:Int,b:Int) = a + b
add: (a: Int, b: Int)Int
```

The add function caused no side-effects. It did not alter the input values provided, it used another pure function, the + operator, and returned the sum of the values as the result of the call. The add function is a pure function.

FUNCTIONAL PROGRAMMING

•***Advantages***

```
int add(int a, int b)
 {
 return a + b
 }
```

• The high level of

abstraction, especially when functions are used, supresses many of the details of programming and thus removes the possibility of commiting many classes of errors;

- **The lack of dependence on assignment operations, allowing programs to be evaluated in many different orders. This evaluation order independence makes function-oriented languages good candidates for programming massively parallel computers;**
- **The absence of assignment operations makes the function-oriented programs much more amenable to mathematical proof and analysis than are imperative programs, because functional programs possess referential transparency.**

- ***Disadvantages***

- **Perhaps less efficiencey**
- **Problems involving many variables or a lot of sequential activity are sometimes easier to handle imperatively or with object-oriented programming.**

LOGIC POGRAMMING PARADIGM

Logic programming refers loosely to

- The use of facts and rules to represent information
- The use of deduction to answer queries

Kowalski illustrates the division of labor in logic programming by writing the informal equation

ALGORITHM LOGIC CONTROL

Here ***logic refers*** to the facts and rules specifying what the algorithm does, and ***control refers*** to how the algorithm can be implemented by applying the rules in a particular order.

We (The programmers) supply the logic part and the programming language supplies the control

LOGIC POGRAMMING PARADIGM

- **Any deducible solution to a query is returned. The definitions and declarations are constructed entirely from relations. i.e. X is a member of Y or X is in the internal between a and b etc.**

•***Advantages:***

The advantages of logic-oriented programming are bifold:

- **The system solves the problem, so the programming steps themselves are kept to a minimum;**
- **Proving the validity of a given program is simple.**

. Programs are written in language of some LOGIC

. Execution of a logic program is a theorem proving process; that is computation is done by logic inferences

. Prolog (PROgramming in LOGic) is a representative programming language

Here logic refers to the facts and rules specifying what the algorithm does

. A**logic** is a language. It has syntax and semantics. More than a language, it has inference rules .

. **Syntax:** The rules about how to form formulas, this is usually the easy part of a logic

. **Semantics:** About meaning carried by the formulas, mainly in terms of a logical consequences

. **Inference rules:** describes the correct way to derive conclusions

REVISION

1. Imperative: what to do and how to do

2. Procedural: Refinement of imperative, introduces the concept of procedures

3. Object Oriented Programming: Solution of problems in which all computations are performed in the context of objects

4. Declarative: Only tell what to do, forget how to do

5. Logic: Use facts and rules to represent information and deduce answer

References

1. Michael L Scott, " Programming Language Pragmatics", Third edition, Elsevier publication (Chapter-1)
2. Ravi Sethi, " Programming Languages-concepts and constructs", Pearson Education (Chapter-1)

Web References

1. NPTEL Online Video resources- Lecture-01
http://www.nptelvideos.in/2012/11/principles-of-programming-languages.html

2. Stanford University Online lectures- Lecture-01 and Lecture-02

https://www.youtube.com/watch?v=Ps8jOj7diA0&list=PL9D558D49CA734A02