INTRODUCTION TO PROGRAMMING PARADIGMS

- Paradigm can also be termed as method to solve some problem or do some task.
- Programming paradigm is an approach to solve problem using some programming language or also we can say it is a method to solve a problem using tools and techniques that are available to us following some approach.
- There are lots for programming language that are known but all of them need to follow some strategy when they are implemented and this methodology/strategy is paradigms. Apart from varieties of programming language there are lots of paradigms to fulfil each and every demand. The programming paradigm is divided into two broad categories.
  - Imperative programming paradigm
  - Declarative programming paradigm

Imperative programming paradigm

- It is one of the oldest programming paradigm. It features close relation to machine architecture. It is based on Von Neumann architecture. It works by changing the program state through assignment statements. It performs step by step task by changing state. The main focus is on how to achieve the goal. The paradigm consist of several statements and after execution of all the result is stored.

Advantage

1. Very simple to implement
2. It contains loops, variables etc.

Disadvantage

1. Complex problem cannot be solved
2. Less efficient and less productive
3. Parallel programming is not possible

Examples of **Imperative** programming paradigm: C, FORTAN, Basic

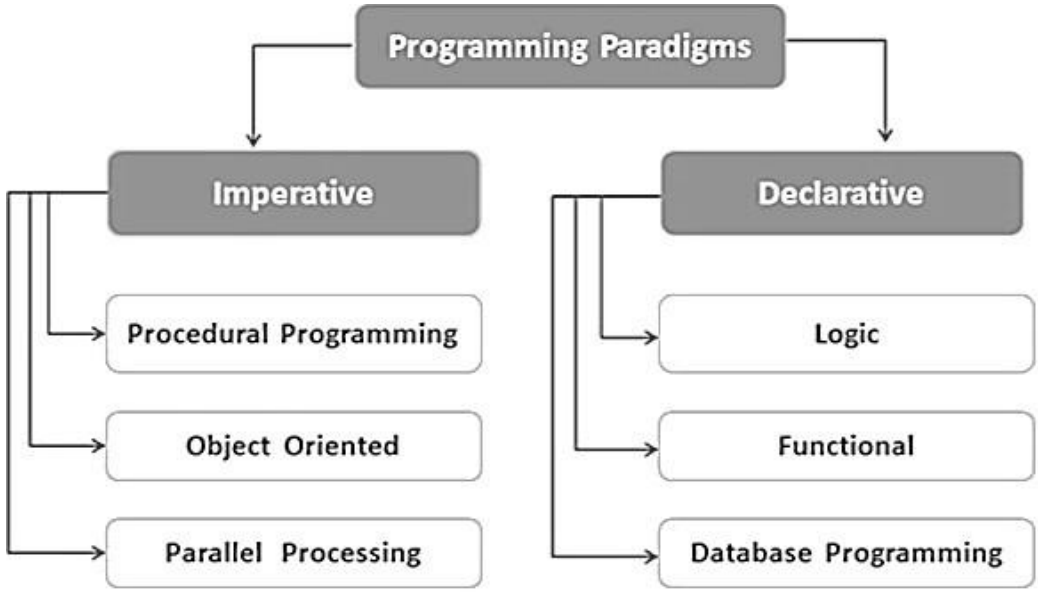


Figure 1.1 Types Programming Paradigms

- Imperative programming is divided into three broad categories: Procedural, OOP and parallel processing. These paradigms are as follows:

Procedural programming paradigm

- This programming has a single program that is divided into small piece called procedure (also known as functions, routines, subroutines). These procedures are combined into one single location with the help of return statements.
- From the main controlling procedure, a procedure call is used to invoke the required procedure. After the sequence is processed, the flow of control continues from where the call was made.
- The main program coordinates calls to procedures and hands over appropriate data as parameters. The data is processed by the procedures and once the program has finished, the resulting data is displayed.
- Example: C, Pascal.

Object Oriented Programming

- It is a type of programming in which programmers define not only the data type of a data structure, but also the types of operations (functions) that can be applied to the data structure.

- In this way, the data structure becomes an object that includes both data and functions. In addition, programmers can create relationships between one object and another. For example, objects can inherit characteristics from other objects.
- One of the principal advantages of object-oriented programming techniques over procedural programming techniques is that they enable programmers to create modules that do not need to be changed when a new type of object is added. A programmer can simply create a new object that inherits many of its features from existing objects. This makes object-oriented programs easier to modify. The basic concepts of OOP are as follows:

1. Objects
2. Classes
3. Data abstraction and encapsulation
4. Inheritance
5. Polymorphism
6. Dynamic binding
7. Message passing.

Objects:

Objects are the basic run-time entities in an object oriented system. They may represent a person, place or a bank a/c or a table of data that the program has to handle. When a program is executed the objects interact by sending messages to one another. They can interact without knowing the details of each other's data or code. Thus an object is considered to be a partitioned area of computer memory that stores data and set of functions that can access the data.

Classes:

The entire set of data and code of an object can be made a user-defined data type with the help of a class. Objects are variables of the type class. Each object is associated with the data of type class with which they are created. A class is thus a collection of objects of similar type.

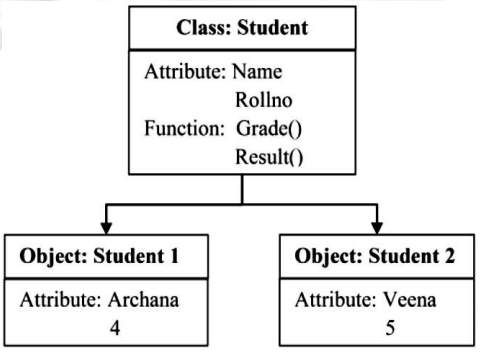


Fig. 1.2 Class and Objects

Data abstraction and encapsulation

Encapsulation:

The wrapping up of data and functions in to a single unit (that unit is called a class) is known as encapsulation. The data is not accessible to the outside world (other functions which are not the members of that class) and only those functions which are wrapped in the class can access it. This insulation of data from the direct access by the program is called data hiding or information hiding.

Abstraction:

Abstraction refers to the act of representing essential features without including the background details or explanations. Classes use the concept of abstraction and are defined as a list of abstract attributes such as size, weight and cost and functions to operate on these attributes. The attributes are sometimes called data members because they hold information. The functions that operate on these data are called member functions.

Inheritance:

Inheritance is the process by which objects of one class acquire the properties of objects of another class. It supports the concept of hierarchical classification. In OOP, the concept of inheritance provides the idea of reusability. This means that we can include additional features to an existing class without modifying it. It is important because it supports the concept of classification. C++ supports different types of inheritance such as single inheritance, multiple inheritance, multilevel inheritance and hierarchical inheritance.

To understand the concept of inheritance, let us consider an example of vehicles as shown in Fig.1.3. Here the class car is a subclass of automatic vehicle, which is again a subclass of the class vehicle. This implies that the class car has all the characteristics of automatic vehicles which in turn has all the properties of vehicles. However, car has some unique features which differentiate it from other subclasses. For example, it has four wheels and five gears, while scooter has two wheels and four gears.

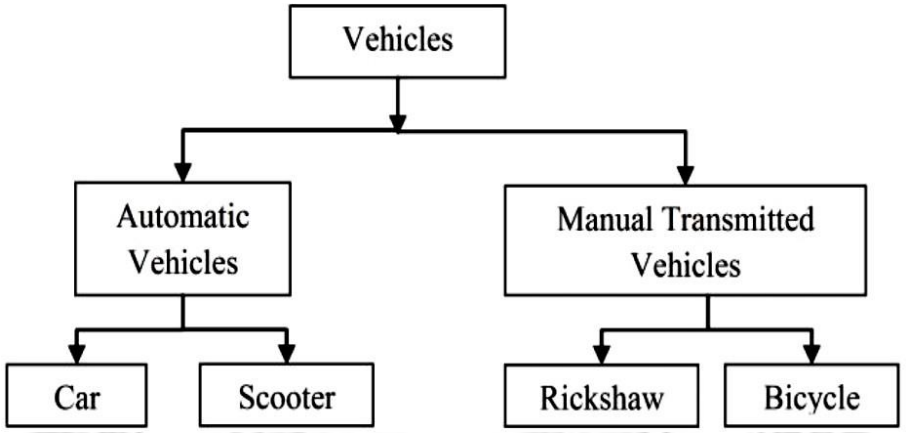


Fig. 1.3 Inheritance

Polymorphism:

Polymorphism, a Greek term means the ability to take more than one form. An operation may exhibit different behaviour in different instances. The behaviour depends up on the types of data used in the operation. The concepts of polymorphism are Operator overloading and Function overloading. For two numbers, the operator + will give the sum. If the operands are strings, then the operation would produce a third string by concatenation. Thus the process of making an operator to exhibit different behaviours in different instances is known as operator overloading. Similarly, we can use a single function to perform different tasks which is known as function overloading. A single function can be used to handle different number and types of arguments.

Binding:

Linking of procedure call to the corresponding code in response to the call.

Dynamic Binding:

Dynamic binding means that the code associated with a given procedure call is not known until the time of the call at run-time. It is associated with polymorphism concept.

Message Passing:

OOP consists of a set of objects that communicate with each other by sending and receiving information. A message for an object is a request for execution of a procedure (function) and therefore will invoke a function in the receiving object that generates the desired result.

Benefits of OOP

- Through inheritance we can eliminate redundant code and extend the use of existing classes.
- We can build secure programs by the principle of data hiding.
- It is easy to partition the work in a project based on objects.
- Object oriented systems can be easily upgraded from small to large systems.
- Communication with external systems are much simpler by means of message passing techniques.
- Software complexity can be easily managed.

Applications of OOP

OOP can be applied in the following areas:

- Real time systems
- Simulation and modeling
- Object oriented data bases
- Hypertext, hypermedia and expertext
- Artificial Intelligence and expert systems.
- Neural networks and parallel programming
- Office automation systems
- CIM / CAM / CAD systems

Example: Simula, JAVA, Python, VB.NET, Ruby.

Parallel processing approach

- Parallel processing is the processing of program instructions by dividing them among multiple processors. A parallel processing system possess many numbers of processor with the objective of running a program in less time by dividing them. This approach seems to be like divide and conquer.
- Examples are NESL (one of the oldest one) and C/C++ also supports because of some library function.

Declarative programming paradigm

- It is divided as Logic, Functional, and Database. In computer science the declarative programming is a style of building programs that expresses logic of computation without talking about its control flow.

- It often considers programs as theories of some logic. It may simplify writing parallel programs. The focus is on what needs to be done rather how it should be done basically emphasize on what code is actually doing.
- It just declare the result we want rather how it has be produced. This is the only difference between imperative (how to do) and declarative (what to do) programming paradigms.

Logic programming paradigms

- It can be termed as abstract model of computation. It would solve logical problems like puzzles, series etc.
- In logic programming we have a knowledge base which we know before and along with the question and knowledge base which is given to machine, it produces result. In normal programming languages, such concept of knowledge base is not available but while using the concept of artificial intelligence, machine learning.
- In logical programming the main emphasize is on knowledge base and the problem. The execution of the program is very much like proof of mathematical statement, e.g., Prolog

Functional programming paradigms

- The functional programming paradigms has its roots in mathematics and it is language independent. The key principal of this paradigms is the execution of series of mathematical functions.
- The central model for the abstraction is the function which are meant for some specific computation and not the data structure. Data are loosely coupled to functions.
- The function hide their implementation. Function can be replaced with their values without changing the meaning of the program. Some of the languages like Perl, java script mostly uses this paradigm.

Database/Data driven programming approach

- This programming methodology is based on data and its movement. Program statements are defined by data rather than hard-coding a series of steps.
- A database program is the heart of a business information system and provides file creation, data entry, update, query and reporting functions. There are several

programming languages that are developed mostly for database application.

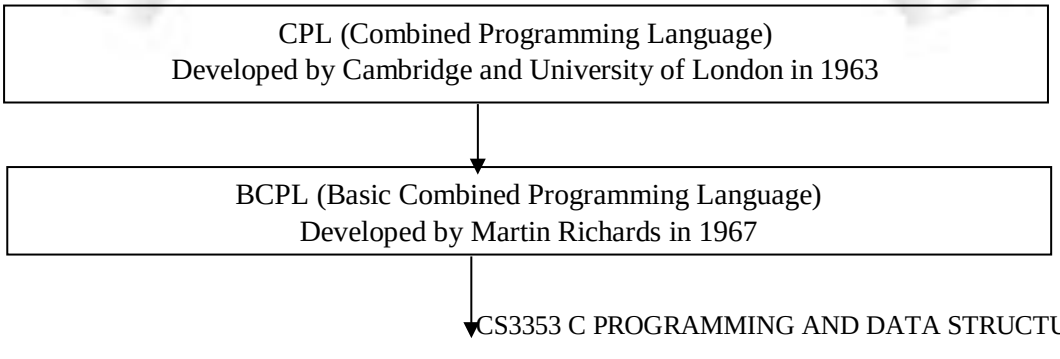
- For example SQL. It is applied to streams of structured data, for filtering, transforming, aggregating (such as computing statistics), or calling other programs. So it has its own wide application.

Characteristics of a Good Programming Language

- A programming language must be simple, easy to learn and use, have good readability and human recognizable.
- Abstraction is a must-have Characteristics for a programming language in which ability to define the complex structure and then its degree of usability comes.
- A portable programming language is always preferred.
- Programming language's efficiency must be high so that it can be easily converted into a machine code and executed consumes little space in memory.
- A programming language should be well structured and documented so that it is suitable for application development.
- Necessary tools for development, debugging, testing, and maintenance of a program must be provided by a programming language.
- A programming language should provide single environment known as Integrated Development Environment (IDE).
- A programming language must be consistent in terms of syntax and semantics.

HISTORY OF C PROGRAMMING

- C is a general purpose structured programming language. C was developed by Dennis Ritchie at AT & T Bell laboratories in 1972. It is an outgrowth of an earlier language called BCPL & B. It was named as C to present it as the successor of B language which was developed earlier by Ken Thompson in 1970 at AT & T Bell laboratories. The various stages in evolution of C language is given in Fig 1.4.



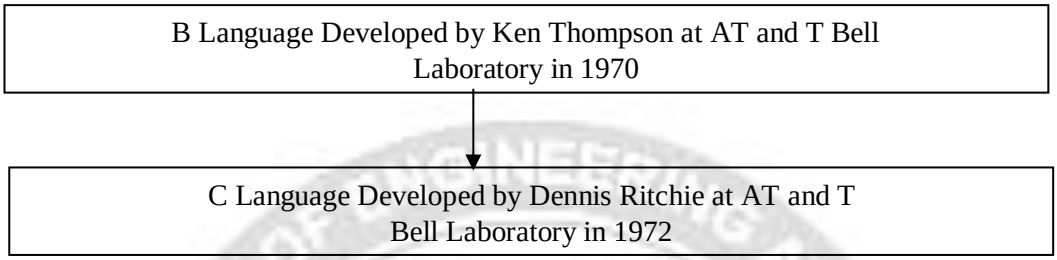


Fig. 1.4 Various Stages in Evaluation of C Languages

- C is a highly portable, which means that C program written for one computer can be run on another with little or no modification. C is well suited for structured programming; thus user has to think of a problem in terms of function modules and blocks.
- The proper collection of these modules would makes a complete program. This modular structures makes program debugging, testing and maintenance easier.
- Another important feature of C is its ability to extend itself. A C program is basically a collection of functions that are supported by the C library. We can addour own functions to the C library. With the availability of a large number of functions, the programming task becomes simple.
- Its flexibility allows C to be used for system programming (For example: the UNIX operating system, the C compiler and all UNIX application software are written in C) as well as for application programming.

Features of C Programming

- C is the widely used language. It provides many features that are given below.

1. Simple
2. Machine Independent or Portable
3. Mid-level programming language
4. Structured programming language
5. Rich Library
6. Memory Management
7. Fast Speed
8. Pointers
9. Recursion

10. Extensible

APPLICATIONS OF C LANGUAGE

- The applications of C are not only limited to the development of operating systems, like Windows or Linux, but also in the development of GUIs (Graphical User Interfaces) and, IDEs (Integrated Development Environments).
- Here are some striking applications offered by the C programming language:

1. Operating Systems

The first operating system to be developed using a high-level programming language was UNIX, which was designed in the C programming language. Later on, Microsoft Windows and various Android applications were scripted in C.

2. Embedded Systems

The C programming language is considered an optimum choice when it comes to scripting applications and drivers of embedded systems, as it is closely related to machine hardware.

3. GUI

GUI stands for Graphical User Interface. Adobe Photoshop, one of the most popularly used photo editors since olden times, was created with the help of C. Later on, Adobe Premiere and Illustrator were also created using C.

4. New Programming Platforms

Not only has C given birth to C++, a programming language including all the features of C in addition to the concept of object-oriented programming but, various other programming languages that are extensively used in today's world like MATLAB and Mathematica. It facilitates the faster computation of programs.

5. Google

Google file system and Google chromium browser were developed using C/C++. Not only this, the Google Open Source community has a large number of projects being handled using C/C++.

6. Mozilla Firefox and Thunderbird

Since Mozilla Firefox and Thunderbird were open-source email client projects, they were written in C/C++.

7. MySQL

MySQL, again being an open-source project, used in Database Management

Systems was written in C/C++.

8. Compiler Design

One of the most popular uses of the C language was the creation of compilers. Compilers for several other programming languages were designed keeping in mind the association of C with low-level languages, making it easier to be comprehensible by the machine.

Several popular compilers were designed using C such as Bloodshed Dev-C, Clang C, MINGW, and Apple C.

9. Gaming and Animation

Since the C programming language is relatively faster than Java or Python, as it is compiler-based, it finds several applications in the gaming sector. Some of the simplest games are coded in C such as Tic-Tac-Toe, The Dino game, The Snake game and many more. Increasing advanced versions of graphics and functions, Doom3 a first-person horror shooter game was designed by id Software for Microsoft Windows using C in 2004.

STRUCTURE OF C PROGRAM

- As C is a programming language, let us go into the concepts of programming in C and it is a structured programming language. Every C program contains a number of building blocks.
- These building blocks should be written in the correct order and procedure, for

```
documentation section
preprocessor section
definition section
global declaration section
main( )
{
declaration part;
executable part;
}
sub program
{
body of the subprogram;
}
```

the C program to execute without any errors. The structure of C is given below.

Program 1.1

```
/* Program to find the area of a circle */          /* Documentation Section */
# include<stdio.h>                                  /* Preprocessor Section */
# include<conio.h>
# define PI 3.14                                   /* Definition Section */
void main()                                        /* main( ) function */
{
float area,r;                                      /* Local variable declaration */
clrscr();                                          // Executable part of the program
printf("\n Enter the radius:\n");
scanf("%f",&r);
area= PI*(r*r);
printf("\n Area of the Circle = %8.2f", area);
getch();
}
```

Output

Enter the radius: 4.5

Area of the Circle = 63.58

Program 1.2

```
/* Program to find the sum of two numbers using function */
                                                   /*Documentation Section */
# include<stdio.h>                                 /* Preprocessor Section */
# include<conio.h>
int a,b;                                           /* Global Variable declaration */
int add(int,int);                                  /* Function declaration */
void main()                                        /* main( ) function */
{
```

```
int c;                          /* Local Variable declaration */
clrscr();                       // Executable part of the main() program
printf("\n Enter the values for a and b:\n");
scanf("%d %d",&a,&b);
c = add(a,b);
printf("\n Sum of %d + %d = %d", a,b,c);
getch();
}
int add(int a,int b)            /* Subprogram of add() function definition */
{
int c;                          /* Local Variable declaration */
c=a+b;                          // Executable part of the function
return(c);
}
```

Output

Enter the values for a and b: 5 3

Sum of 5 + 3 = 8

Documentation section

The documentation section is included in the comments, which contains the author name, the date of development and the program details.

Preprocessor section

The preprocessor section provides preprocessor statements which direct the compiler to link functions from the system library.

Definition section

The definition section defines all symbolic constants refer to assigning a macro of a name to a constant. The general syntax of a symbolic constant is

#define constant_name constant_value

Global declaration section

The global declaration section contains variable declarations which can be accessed anywhere within the program.

Main section

Main section is divided into two portions, the declaration part and the executable part. The declaration part used to declare any variables in the main block of the program. The executable part contains set of statements within the open and close braces. Execution of the program begins at the opening braces and ends at the closing braces.

User defined function section

The user defined function section (or) the Sub program section contains user defined functions which are called by the main function. Each user defined function contains the function name, the argument and the return value.

LEXICAL ELEMENTS OF C

- Data's can be of any kind, it may be numbers, characters and strings. These data should be processed in order to produce the information output. Programming languages are used for this processing of data into information.
- Every program consists of a sequence of steps or instruction for processing data. Each instruction must abide to certain syntax rules of grammar of the particular language. Likewise C has its own grammar. Data may be either a constant or a variable.
- In 'C' language each and every individual unit is called as Token or Lexical element. The various 'C' Tokens are
  i) Character set
  ii) Delimiters
  iii) Keywords
  iv) Identifiers
  v) Data types
  vi) Constants
  vii) Variables

Character Set

- The set of characters used to write the program words, expressions and statements. It is the basic building block to form program elements. The set of characters used in a language is known as its *character set*. These characters can be represented in the computer.

- The C character set consists of upper and lower case alphabets, digits, special characters and white spaces. The alphabets and digits are together called the alphanumeric characters.

i) Alphabets

A, B, C,…..Z

a, b, c,….z

ii) Digits

0 1 2 3 4 5 6 7 8 9

iii) Special Characters

.	Comma	-	Underscore
;	Semicolon	~	Tilde
:	Colon	$	Dollar sign
#	Number sign	%	Percent sign
‘	Apostrophe	&	Ampersand
“	Quotation mark	?	Question mark
!	Exclamation mark	*	Asterisk
|	Vertical bar	-	Minus sign
+	Plus sign	.	Period
[	Left bracket	]	Right bracket
{	Left brace	}	Right brace
/	Slash	\	Backslash
^	Caret	( )	Parenthesis left/right
<	Less than	=	Equal to
>	Greater than	@	At the rate

Table 1.1 List of Special Characters

iv) White space characters

Blank space, newline, form feed, horizontal tab, vertical tab

v) Trigraph characters

The *trigraph characters* are used to type certain characters that are not available on some keyboards. It consists of three characters. Two question marks followed by character.

Trigraph Characters	Translation
??=	# (pound sign)
?? (	[(left bracket)
??)	] (right bracket)
?? <	{ (left brace)
?? >	} (right brace)
?? !	/ backslash
?? /	| vertical bar
??'	^ caret

Table 1.2 Trigraph Characters

Delimiters

- The language pattern of C uses a special kind of symbols, which are called *delimiters*. They are given in Table. 1.3.

Delimiters	Use
: Colon	Useful for label.
; Semicolon	Terminates the statement.
( ) Parenthesis	Used in expression and function.
[ ] Square Bracket	Used for array declaration.
{ } Curly Brace	Scope of the statement.
# Hash	Preprocessor directive.
, Comma	Variable separator.

Table 1.3 Delimiters

Keywords

- These are certain reserved words called *keywords*, which are standard, predefined meanings in C. They must be written in lower case. There are 32 keywords available in C. The standard keywords are shown in Table 1.4.

auto	double	int	struct
break	else	long	switch
case	enum	register	typedef
char	extern	return	union
const	float	short	unsigned
continue	for	signed	void
default	goto	sizeof	volatile
do	if	static	while

Table 1.4 Keywords in C

Identifiers

- *Identifiers* are the names defined by the programmer for various program elementssuch as variables, constant and functions. An identifier consists of a sequence of letters and digits.
- The following rules are to be followed to declare an identifier.
  i) The first character must be a letter followed by a letter or a digit.
  ii) Special characters are not allowed except underscore.
  iii) The length of the variable varies from one compiler to another. Generally, most of the compilers support eight characters excluding extension. However, the ANSI standard recognizes the maximum length of a variable up to 31 characters.
  iv) The variable should not be a C keyword.
  v) The variable names may be a combination of upper and lower characters. For example, suM and sum are not the same variable.
  vi) The variable name should not start with a digit.

Examples

- The following names are valid identifier