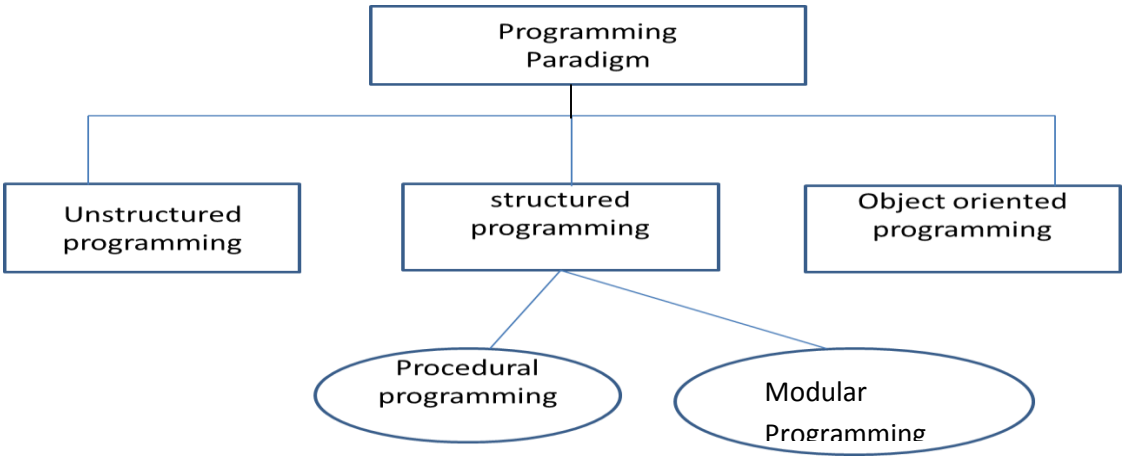
INTRODUCTION TO PROGRAMMING PARADIGMS

Program is a set of instructions required for processing. Paradigm means pattern ormodel. Programming paradigm refers to how program is written in order to solve the problem.



Programming paradigms

Unstructured programming

It means writing small simple program which consist of only one main program. Allactions (input, output, processing) are done within one program. So, this language is restricted.

Structured programming

It means writing large program which consist of small sub programs. When sub programsare completed, they are combined together to solve the problem.

It is classified into two types. They are,

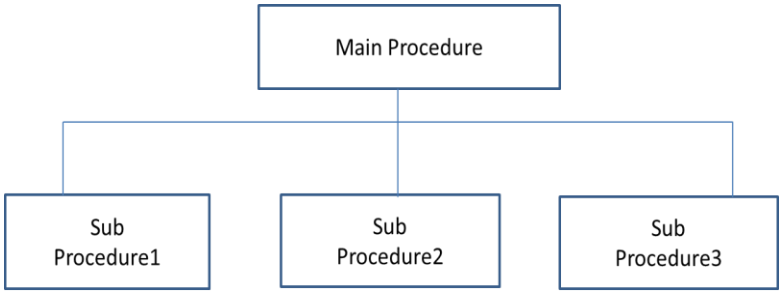
1. Procedural programming
2. Modular Programming

Procedural programming

Procedure is a group of statements. It means writing program into procedures. Whenprocedures are completed, they are combined together to solve the problem.

- Procedure call is used to invoke the procedure.
- Return statement is used to return from the procedure.

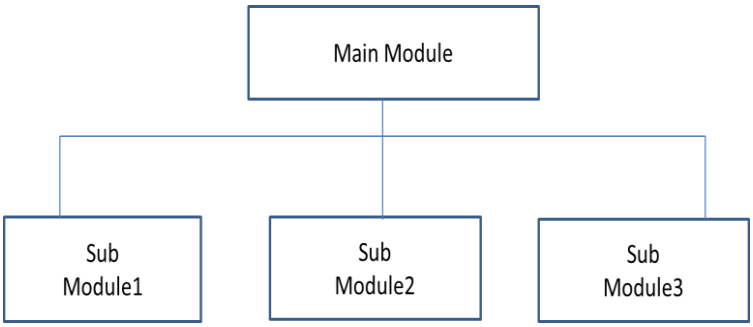
Example: CLIST, Hypertalk, MATLAB, PL/I



Procedural Programming

Modular Programming

It means writing program into modules. When modules are completed, they are combinedtogether to solve the problem.

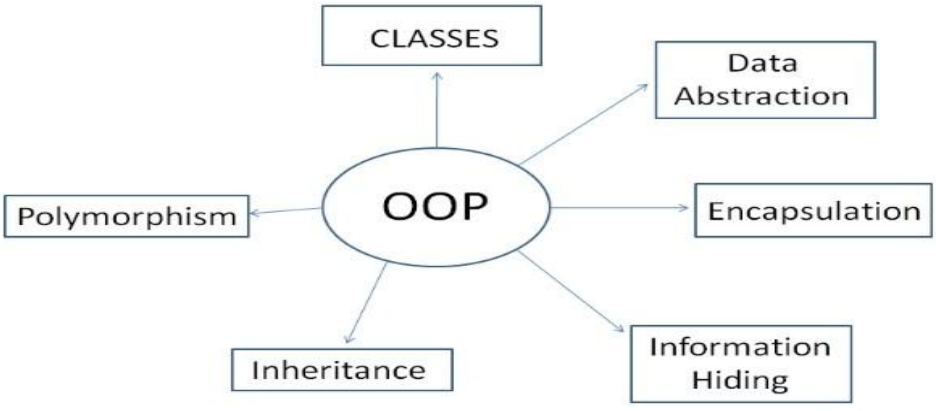


Modular Programming

Object oriented programming (OOPS)

Objects contain data and functions. It means writing program into collection of objects.

Example: Agora, Beta, Java, Moto.



OOPS concepts

Interpreted programming

An interpreted programming language is a programming language that executesinstructions directly, without compiling a program into machine-language instructions.

Example: BASIC (Beginners All Purpose Symbolic Instruction Code), LISP (ListProcessing Language), Python.

Functional programming

Functional Programming languages define every computation as a mathematicalevaluation. They are based on mathematical functions. They focus on application of functions.

Example: Clean, curry, F#, Haskell and Q

Compiled programming

Compiled Programming language is a programming language which uses compilers toconvert the source code into machine code.

Example: Ada, algol, C,C++,C#, COBOL, Java, Fortran, VB

Machine Programming:

In machine language, instructions are in the form of 0,,s and 1,,s. Instructions in machinelanguage consist of two parts.

- OPCODE tells the computer what functions are to be performed.
- OPERAND tells the computer where to store the data.

STRUCTURE OF C PROGRAM

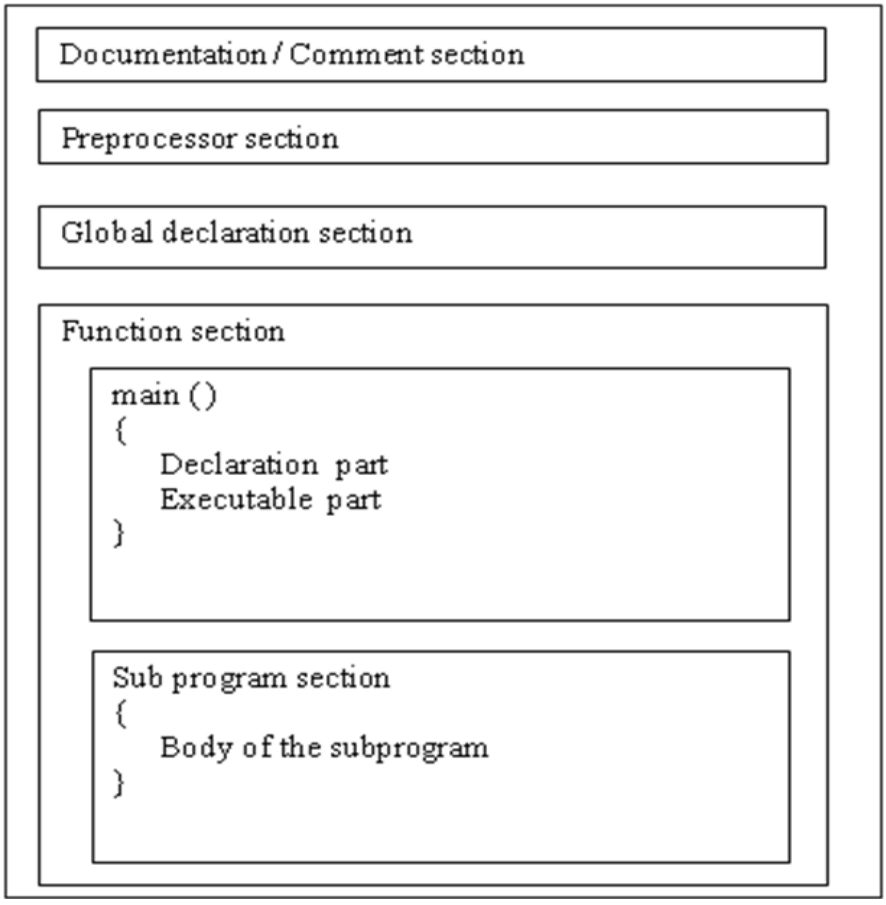
C is a middle level, structured, core programming language. Every C program contains a number of building blocks. These blocks should be written in a correct order and procedure. But some blocks may be optional.

In general, a C program is composed of the following sections:

Section 1: Documentation / Comments
Section 2: Pre-processor directives
Section 3: Global declarations
Section 4: Functions

- ***Section 1, 2 and 3 are optional***
- ***Section 3 is essential.***

The structure of the C program is given in the following figure.



structure of the C program

i) Documentation Section:

Comments are non executable statements. Comments are very helpful in identifying theprogram features and underlying logic of the program.

- ✓ The multi line comment begins with „/*‟ and end with „*/‟.
- ✓ Single line comment statement start with double slash sign „//‟.

Example:

- /* Matrix Addition Program */
- // Author : Joshna

ii) Preprocessor Section:

It is used to link system library files, for defining the macros and for defining theconditional inclusion.

- ✓ It starts with symbol (#).
- ✓ There is no termination symbol semicolon.

Example:

- #include <stdio.h>
- #define A 10
- # ifdef
- #endif

iii) Global Declaration Section:

Variables that are declared outside of all the function are called as global variables.

- ✓ It is accessed by all the functions
- ✓ It must be placed before the main function.

iv) Function Section:

Function section should contain only one main function and zero or more user definedfunction. ***Execution starts with the main function.***

Every function consists of 2 parts

a) Header of the function

b) Body of the function

a) Header of the function

The general form of the header of the function is

[return_type] function_name([argument_list])

b) Body of the function

The body of the function consists of a set of statements enclosed within curly bracketscommonly known as braces. The statements are of two types.

Local declaration:

Variables that are declared within function are called as Local variables.

Executable statements:

Other statements following the declaration statements are used to perform various

tasks.

Example

```
/*Addition of two numbers*/        ← Documentation Section
#include<stdio.h>                   Pre-processor directives
#include<conio.h>                   ←
```

```
#define A 10        ←   Definition Section
int a=10;           ←   Global declarations
void main()         ←    Main() function
{

 }
```