Programming in C++

Tushar B. Kute,
http://tusharkute.com

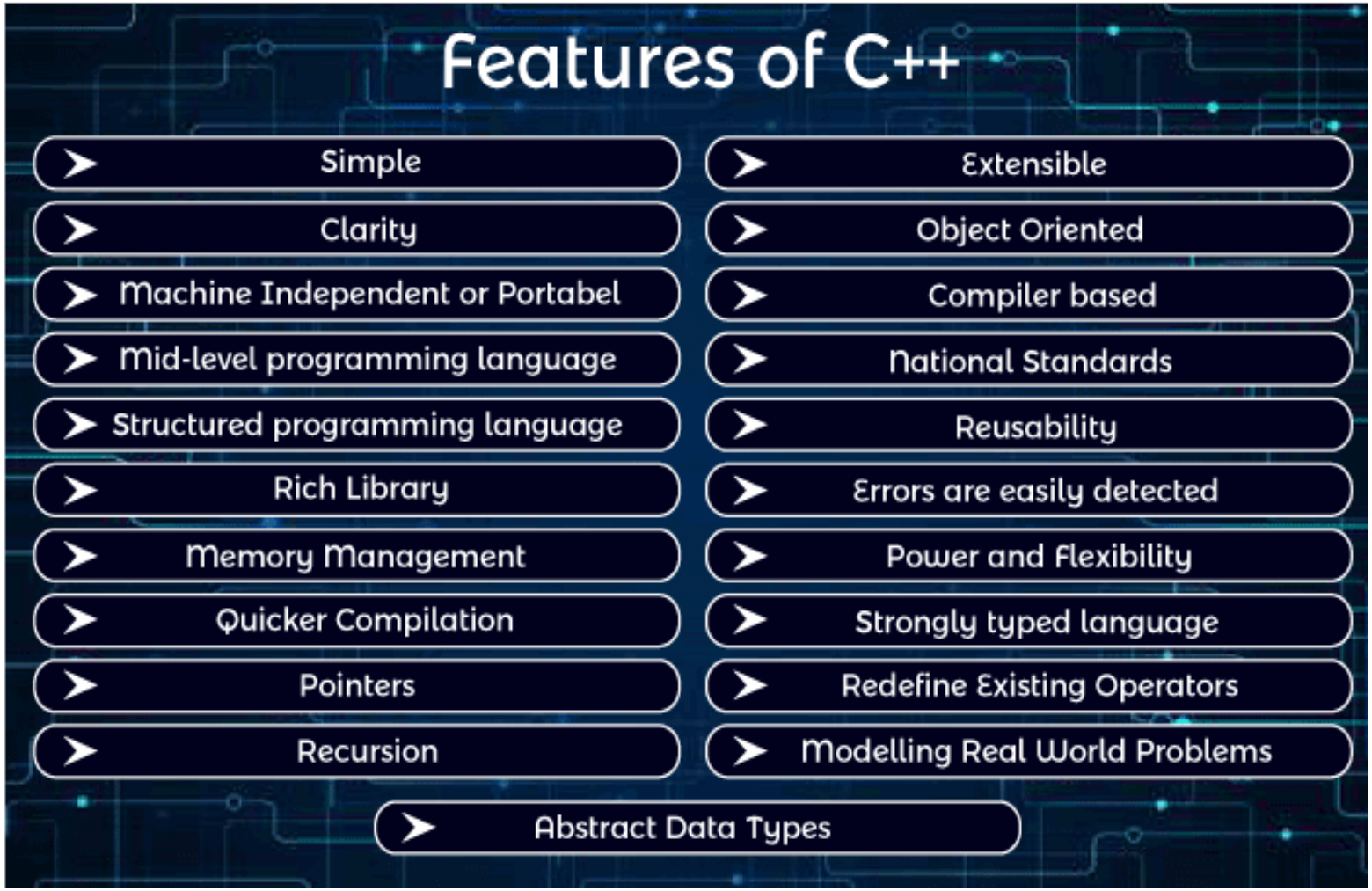
C++

- C++ is an object-oriented programming language. It is an extension to C programming.
- C++ contains, control statements, objects and classes, inheritance, constructor, destructor, this, static, polymorphism, abstraction, abstract class, interface, namespace, encapsulation, arrays, strings, exception handling, File IO, etc.
- C++ is a general purpose, case-sensitive, free-form programming language that supports object-oriented, procedural and generic programming.

C++

- C++ programming language was developed in 1980 by Bjarne Stroustrup at bell laboratories of AT&T (American Telephone & Telegraph), located in U.S.A.
- Bjarne Stroustrup is known as the founder of C++ language.
- It was develop for adding a feature of OOP (Object Oriented Programming) in C without significantly changing the C component

C++ Features


Features of C++
Simple
Clarity
Machine Independent or Portabel
Mid-level programming language
Structured programming language
Rich Library
Memory Management
Quicker Compilation
Pointers
Recursion
Extensible
Object Oriented
Compiler based
National Standards
Reusability
Errors are easily detected
Power and Flexibility
Strongly typed language
Redefine Existing Operators
Modelling Real World Problems
Abstract Data Types

C++ Program

```
#include <iostream>
using namespace std;

// main() is where program execution begins.
int main() {
   cout << "Hello World"; // prints Hello World
   return 0;
}
```

C++ Program

- The C++ language defines several headers, which contain information that is either necessary or useful to your program. For this program, the header <iostream> is needed.
- The line using namespace std; tells the compiler to use the std namespace. Namespaces are a relatively recent addition to C++.
- The next line '// main() is where program execution begins.' is a single-line comment available in C++. Single-line comments begin with // and stop at the end of the line.

C++ Program

- The line int main() is the main function where program execution begins.
- The next line cout << "Hello World"; causes the message "Hello World" to be displayed on the screen.
- The next line return 0; terminates main( )function and causes it to return the value 0 to the calling process.

C vs C++

C	C++
C is a procedural language and is based on functions.	C is an object-oriented programming language.
C doesn't support concepts like polymorphing, abstraction, encapsulation, and inheritance.	C++ is object-oriented and supports polymorphing, abstraction, encapsulation, and inheritance. On top of this, it is also procedural.
C has a small set of keywords.	C++ has a broader array of keywords.
C programming files are stored with .c extension	Files written in C++ are stored with .cpp extension.
Doesn't support user-defined data types.	It supports user-defined data types
Doesn't support user-defined data types.	Supports user-defined data types
Requires <stdio.h> header file to support I/O operations.	<iostream.h> header file is used for I/O operations.
No concept of access modifiers	Has access modifiers.
Doesn't support data encapsulation.	Encapsulation is supported.
Exception handling isn't supported.	Comes with exception handling.
Memory management is manual and complex. calloc() and malloc() functions are available for memory allocation.	Memory management is manual; however, it provides various memory management operators like new and free.
C, being procedural, focuses on the method more than the data.	C++ focuses on the data more than the method.
C uses scanf() and printf() functions for input/output.	It uses cin and cout for input/output.
Function and operator overloading isn't possible.	Supports function and operator overloading.
Calls to the main() function can be made through other functions.	The main() function cannot be invoked through other functions.

Variables

- A variable is a name of the memory location. It is used to store data. Its value can be changed, and it can be reused many times.
- It is a way to represent memory location through symbol so that it can be easily identified.
- Let's see the syntax to declare a variable:

```
type variable_list;
```

Variables

- The example of declaring the variable is given below:

```
int a;
float b;
char c;
```

- Here, a, b, c are variables. The int, float, char are the data types.

Variables

- We can also provide values while declaring the variables as given below:

```
int a=10,b=20;
//declaring 2 variable of integer type
float f=20.8;
char c='A';
```

Variables : Rules

- A variable can have alphabets, digits, and underscore.
- A variable name can start with the alphabet, and underscore only. It can't start with a digit.
- No whitespace is allowed within the variable name.
- A variable name must not be any reserved word or keyword, e.g. int, float, etc.

Variables : Rules

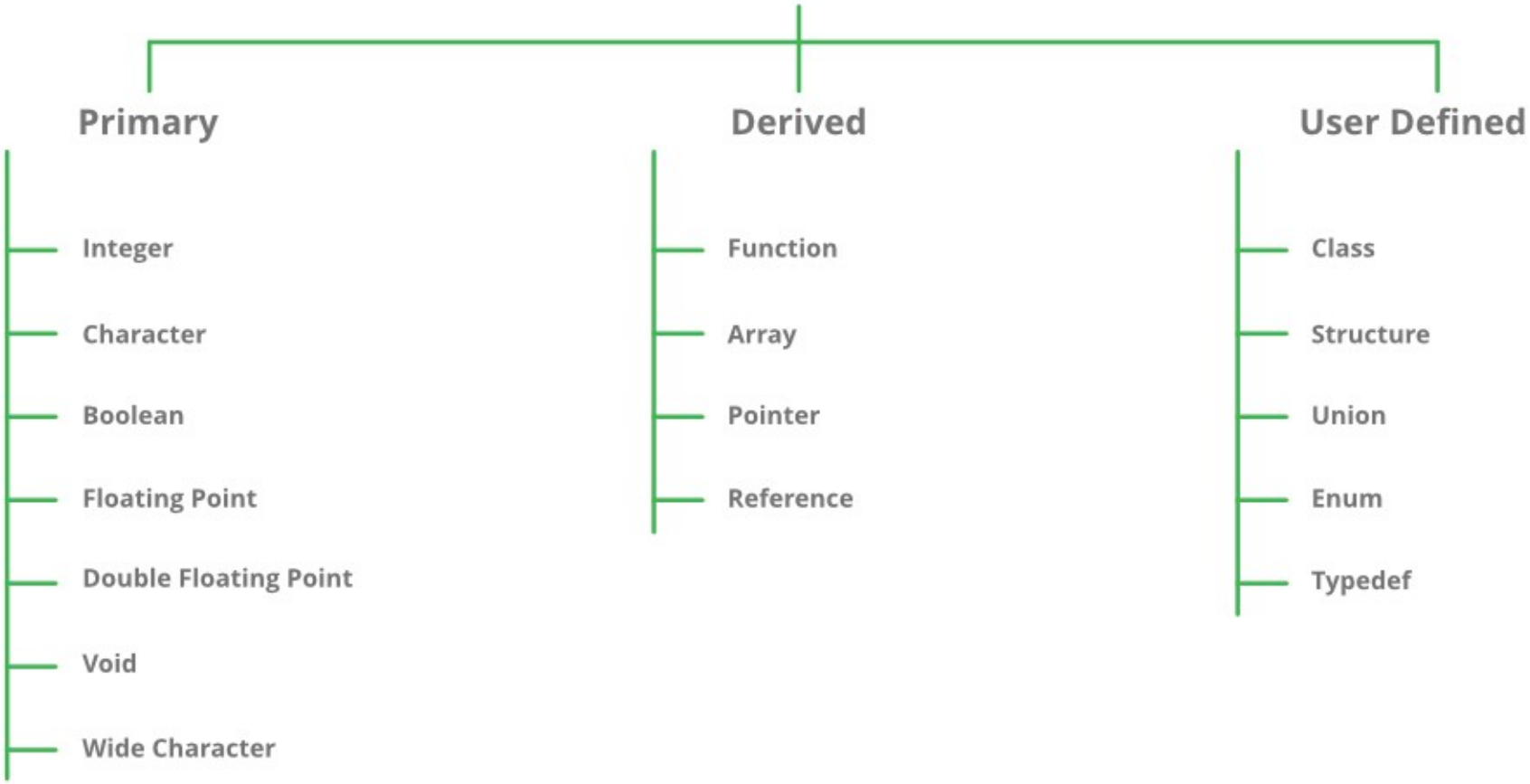
- Valid variable names:

```
int a;
int _ab;
int a30;
```

- Invalid variable names:

```
int 2;
int a b;
int long;
```

Data Types


Primary
Integer
Character
Boolean
Floating Point
Double Floating Point
Void
Wide Character
Derived
Function
Array
Pointer
Reference
User Defined
Class
Structure
Union
Enum
Typedef

Keywords

- A keyword is a reserved word. You cannot use it as a variable name, constant name, etc.
- A list of 30 Keywords in C++ Language which are not available in C language are given below.

asm	dynamic_cast	namespace	reinterpret_cast	bool
explicit	new	static_cast	false	catch
operator	template	friend	private	class
this	inline	public	throw	const_cast
delete	mutable	protected	true	try
typeid	typename	using	virtual	wchar_t

Operators

- An operator is simply a symbol that is used to perform operations. There can be many types of operations like arithmetic, logical, bitwise, etc.
- There are following types of operators to perform different types of operations in C language.

  Arithmetic Operators

  Relational Operators

  Shift Operators

  Logical Operators

  Bitwise Operators

  Ternary or Conditional Operators

  Assignment Operator

  Misc Operator

Operators

Category	Operator	Associativity
Postfix	() [] -> . ++ - -	Left to right
Unary	+ - ! ~ ++ - - (type)* & sizeof	Right to left
Multiplicative	* / %	Left to right
Additive	+ -	Left to right
Shift	<< >>	Left to right
Relational	< <= > >=	Left to right
Equality	== !=	Left to right
Bitwise AND	&	Left to right
Bitwise XOR	^	Left to right
Bitwise OR	|	Left to right
Logical AND	&&	Left to right
Logical OR	||	Left to right
Conditional	?:	Right to left
Assignment	= += -= *= /= %=>>= <<= &= ^= |=	Right to left
Comma	,	Left to right

Comments

- Comments in C++ language are used to provide information about lines of code.
- It is widely used for documenting code. There are 2 types of comments in the C language.

  Single Line Comments

  Multi-Line Comments

If statement

- The if-else statement in C++ is used to perform the operations based on some specific condition.
- The operations specified in if block are executed if and only if the given condition is true.
- There are the following variants of if statement in C++ language.

    If statement

    If-else statement

    If else-if ladder

    Nested if

If statement

- The if statement is used to check some given condition and perform some operations depending upon the correctness of that condition.
- It is mostly used in the scenario where we need to perform the different operations for the different conditions. The syntax of the if statement is given below.

```
if(expression){
    //code to be executed
}
```

if-else

```
if(expression){
    //code to be executed if condition is true
}else{
    //code to be executed if condition is false
}
```

Loops

- While loop
- For loop
- Do-while loop
- Break
- Continue
- goto

While Loops

- While loop is also known as a pre-tested loop. In general, a while loop allows a part of the code to be executed multiple times depending upon a given boolean condition.
- It can be viewed as a repeating if statement. The while loop is mostly used in the case where the number of iterations is not known in advance.
- Syntax of while loop:
- The syntax of while loop in c language is given below:

```
while(condition){
    //code to be executed
}
```

Do-While Loop

- The do while loop is a post tested loop. Using the do-while loop, we can repeat the execution of several parts of the statements.
- The do-while loop is mainly used in the case where we need to execute the loop at least once. The do-while loop is mostly used in menu-driven programs where the termination condition depends upon the end user.
- The syntax :

```
do{
    //code to be executed
}while(condition);
```

for Loop

- The for loop in C++ is used to iterate the statements or a part of the program several times.
- It is frequently used to traverse the data structures like the array and linked list.
- The syntax of for loop :

```
for(Expression 1; Expression 2; Expression 3)   {
    //code to be executed
}
```

break

- The break is a keyword in C++ which is used to bring the program control out of the loop.
- The break statement is used inside loops or switch statement.
- The break statement breaks the loop one by one, i.e., in the case of nested loops, it breaks the inner loop first and then proceeds to outer loops.

continue

- The continue statement in C++ is used to bring the program control to the beginning of the loop.
- The continue statement skips some lines of code inside the loop and continues with the next iteration.
- It is mainly used for a condition so that we can skip some code for a particular condition.
- Syntax:

```
//loop statements
continue;
//some lines of the code which is to be skipped
```

Array

- An array is defined as the collection of similar type of data items stored at contiguous memory locations.
- Arrays are the derived data type in C++ programming language which can store the primitive type of data such as int, char, double, float, etc. It also has the capability to store the collection of derived data types, such as pointers, structure, etc.
- The array is the simplest data structure where each data element can be randomly accessed by using its index number.

Array

- We can declare an array in the c language in the following way.

```
data_type array_name[array_size];
```

- Now, let us see the example to declare the array.

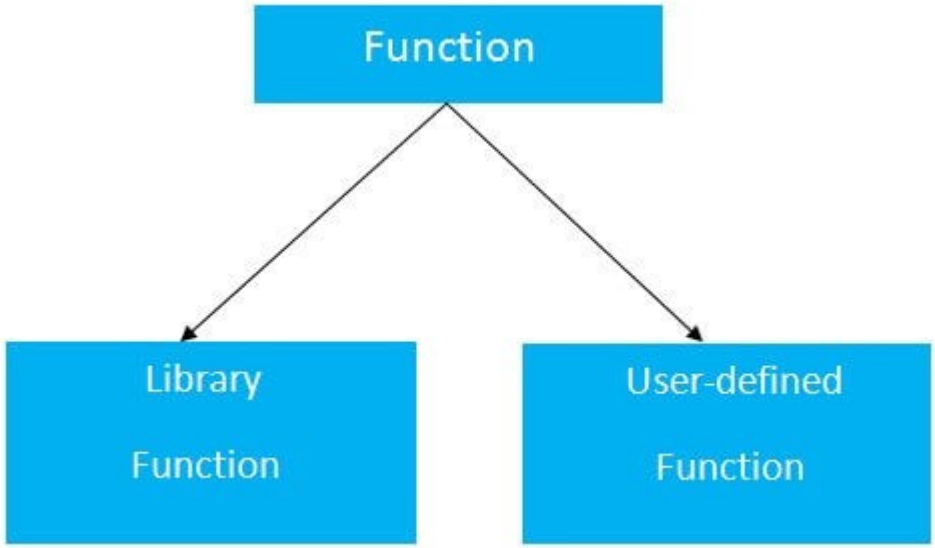
```
int marks[5];
```

- Here, int is the data_type, marks are the array_name, and 5 is the array_size.

Functions

- In c, we can divide a large program into the basic building blocks known as function.
- The function contains the set of programming statements enclosed by {}. A function can be called multiple times to provide reusability and modularity to the C++ program.
- In other words, we can say that the collection of functions creates a program.
- The function is also known as procedure or subroutine in other programming languages.

Functions



Functions

- Function declaration A function must be declared globally in a c program to tell the compiler about the function name, function parameters, and return type.
- Function call Function can be called from anywhere in the program. The parameter list must not differ in function calling and function declaration. We must pass the same number of functions as it is declared in the function declaration.
- Function definition It contains the actual statements which are to be executed. It is the most important aspect to which the control comes when the function is called. Here, we must notice that only one value can be returned from the function.

Functions

- Function declaration
  - return_type function_name (argument list);
- Function call
  - function_name (argument_list)
- Function definition
  - return_type function_name (argument list) {function body;}

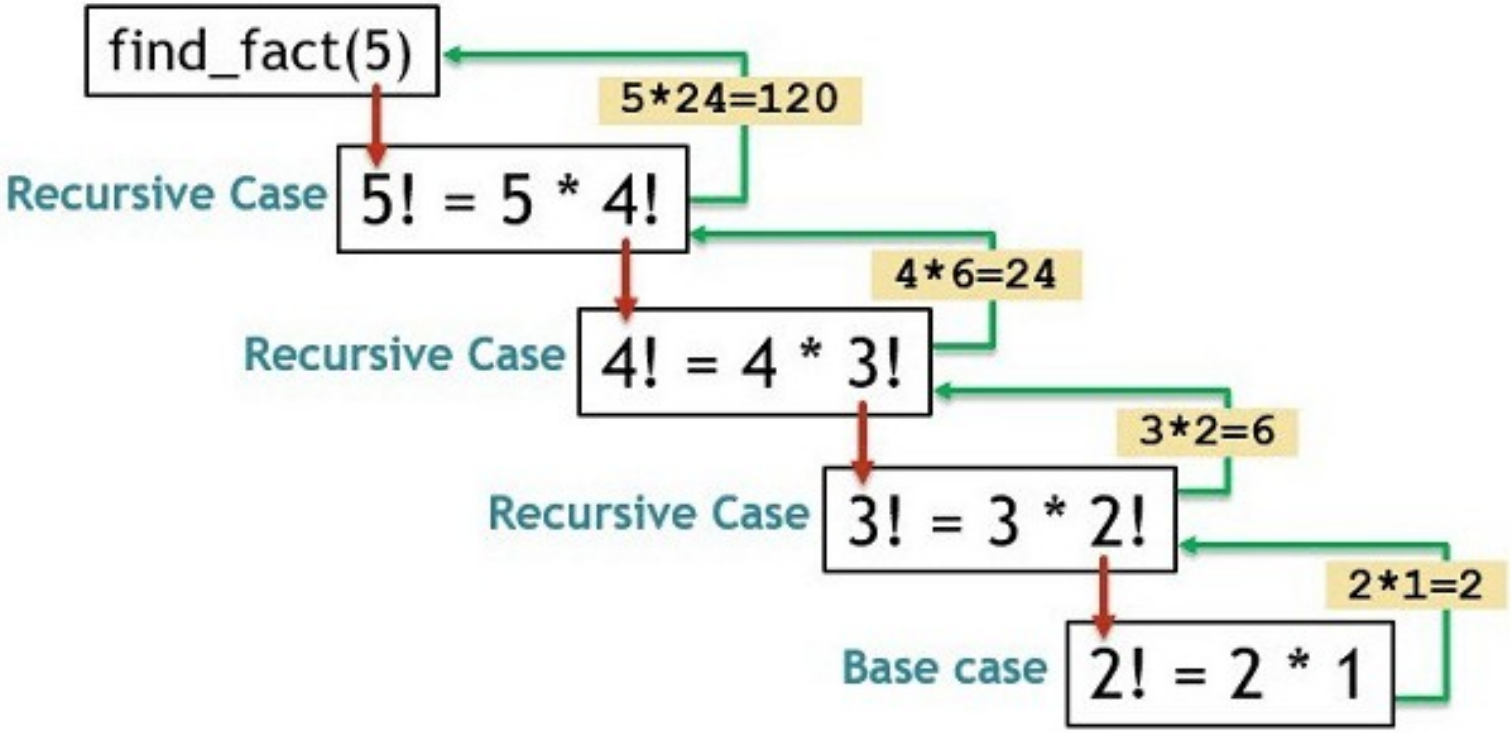
Recursion

- Recursion is the process which comes into existence when a function calls a copy of itself to work on a smaller problem.
- Any function which calls itself is called recursive function, and such function calls are called recursive calls.
- Recursion involves several numbers of recursive calls. However, it is important to impose a termination condition of recursion.
- Recursion code is shorter than iterative code however it is difficult to understand.

Recursion

- Recursion cannot be applied to all the problem, but it is more useful for the tasks that can be defined in terms of similar subtasks.
- For Example, recursion may be applied to sorting, searching, and traversal problems.
- Generally, iterative solutions are more efficient than recursion since function call is always overhead.
- Any problem that can be solved recursively, can also be solved iteratively. However, some problems are best suited to be solved by the recursion, for example, tower of Hanoi, Fibonacci series, factorial finding, etc.

Recursion


find_fact(5)
5*24=120
Recursive Case
5! = 5 * 4!
4*6=24
Recursive Case
4! = 4 * 3!
3*2=6
Recursive Case
3! = 3 * 2!
2*1=2
Base case
2! = 2 * 1

Thank you

@mitu_skillologies

@mITuSkillologies

@mitu_group

@mitu-skillologies

@MITUSkillologies

kaggle

@mituskillologies

Web Resources
https://mitu.co.in
http://tusharkute.com

@mituskillologies

contact@mitu.co.in

tushar@tusharkute.com