DATA STRUCTURES

MODULE 1

KtuQbank

Module 1

- Basic Concepts of Data Structures
- System Life Cycle, Algorithms, Performance Analysis, Space Complexity, Time Complexity, Asymptotic Notation, Complexity Calculation of Simple Algorithms

For More Visit : KtuQbank.com | Fair Use Policy

KtuQbank

Data Structure

- Data Structure is a way of collecting and organising data in such a way that we can perform operations on these data in an effective way.
- It represents the knowledge of data to be organized in memory.
- A **data structure** is a particular way of organizing data in a computer
- A data structure is a specialized format for organizing, processing, retrieving and storing data.

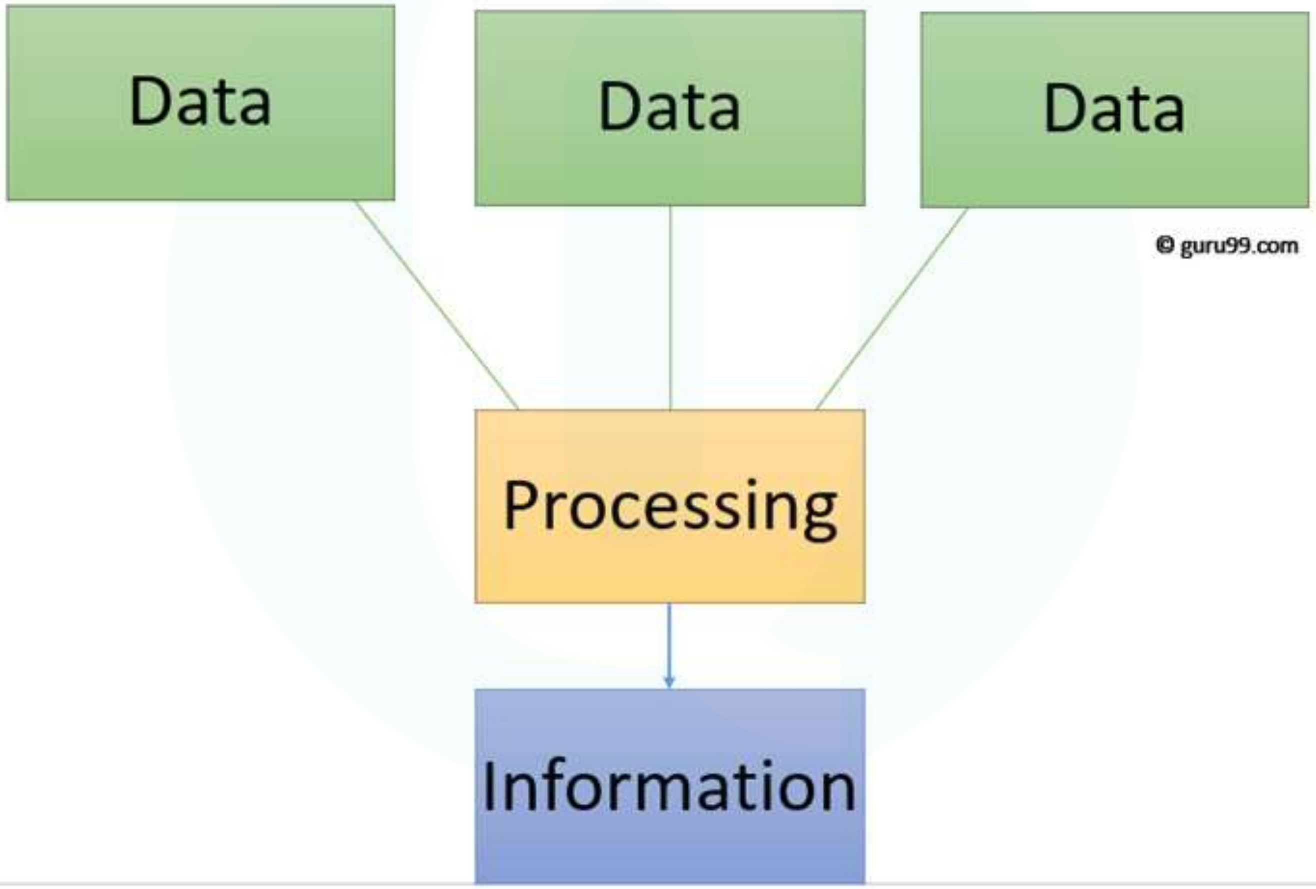
KtuQbank

Data Structure

- Data: Data is a raw and unorganized fact that is required to be processed to make it meaningful
- Information: Information is a set of data which is processed in a meaningful way according to the given requirement.

KtuQbank

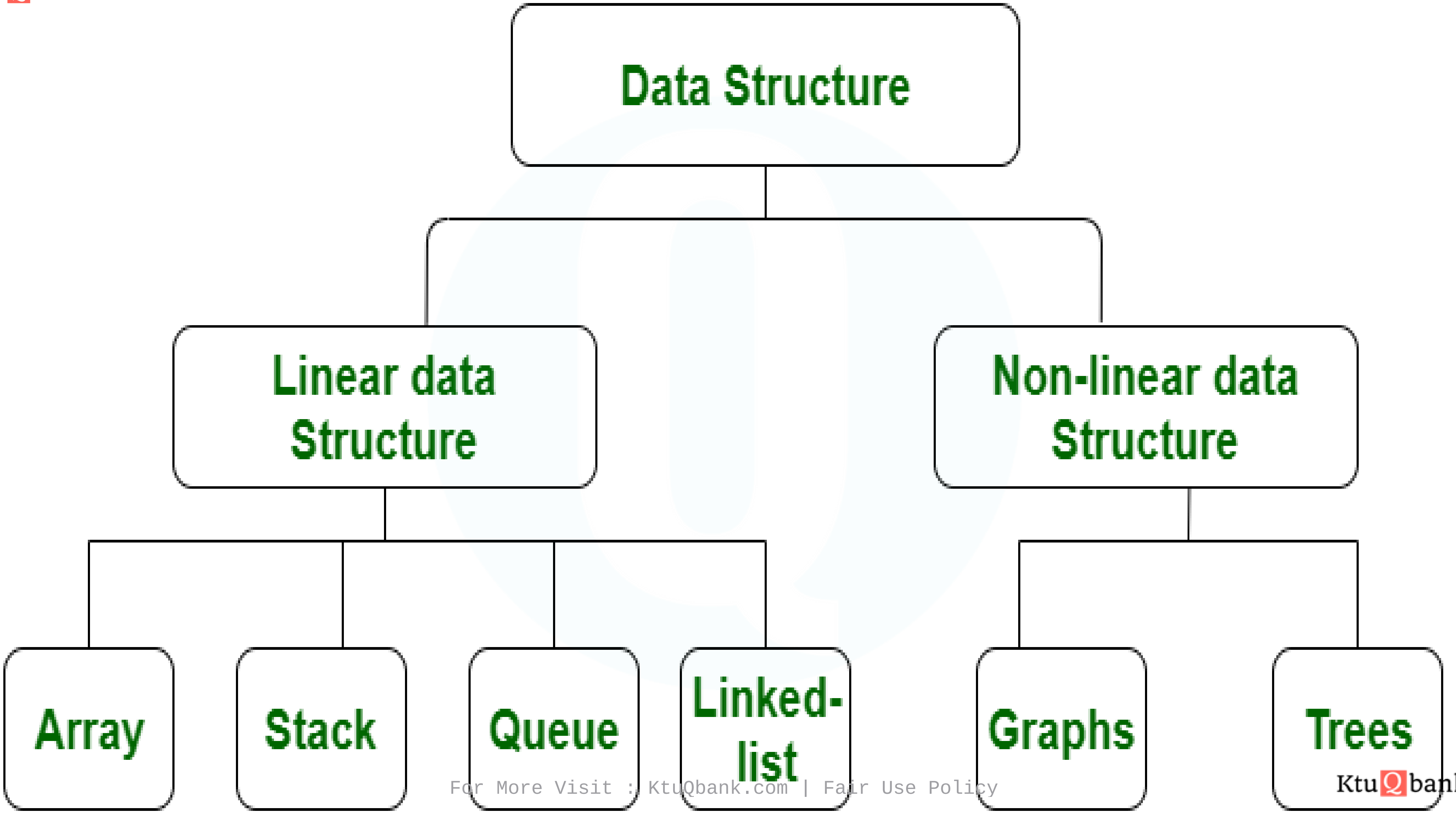
KtuQbank



For More Visit : KtuQbank.com | Fair Use Policy

KtuQbank

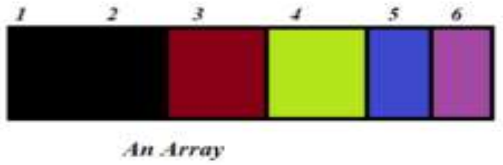
KtuQbank

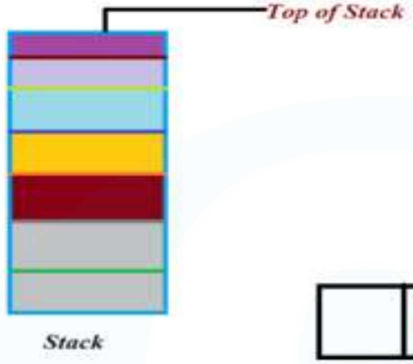


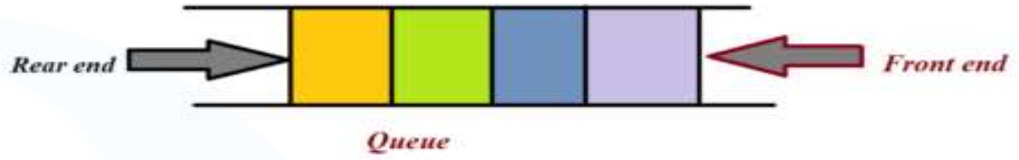
For More Visit : KtuQbank.com | Fair Use Policy

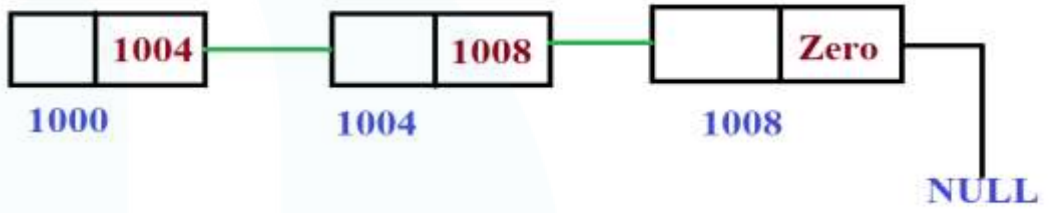
KtuQbank

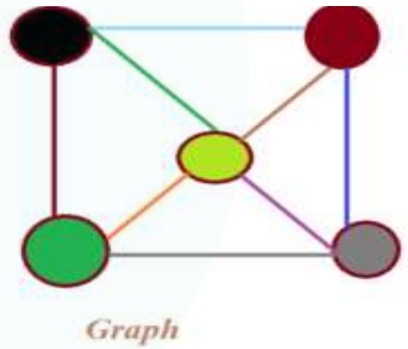
KtuQbank

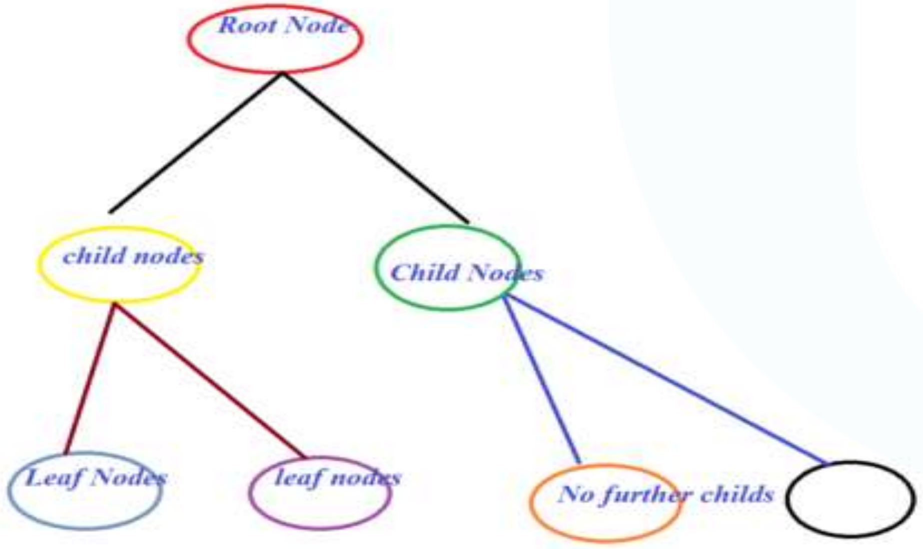












Tree Data Structure

For More Visit : KtuQbank.com | Fair Use Policy

KtuQbank

Operations on Data Structures

- The basic operations that are performed on data structures are as follows:
    1. Insertion
    2. Deletion
    3. Traversal
    4. Sorting
    5. Merging

For More Visit : KtuQbank.com | Fair Use Policy

KtuQbank

Operations on Data Structures

- Insertion means addition of a new data element in a data structure.
- Deletion means removal of a data element from a data structure if it is found.
- Searching involves searching for the specified data element in a data structure.
- Traversal of a data structure means processing all the data elements present in it.
- Arranging data elements of a data structure in a specified order is called sorting.
- Combining elements of two similar data structures to form a new data structure of the same type, is called merging.

For More Visit : KtuQbank.com | Fair Use Policy

KtuQbank

System Life Cycle

- The system life cycle is a series of stages that are worked through during the development of a new information system.
- A lot of time and money can be wasted if a system is developed that doesn’t work properly or do exactly what is required of it.

For More Visit : KtuQbank.com | Fair Use Policy

KtuQbank

System Life Cycle-5 Phases

- Requirements
- Design
- Analysis
- Coding
- Verification

KtuQbank

System Life Cycle

Requirements- Understanding the information you are given (the input) and what results you are to produce (the output).

- A rigorous description of the input and output which covers all cases.

For More Visit : KtuQbank.com | Fair Use Policy

KtuQbank

KtuQbank

System Life Cycle

Design- There are several data objects (such as a maze, a polynomial, or a list of names).

- For each object there will be some basic operations to perform on it (such as print the maze, add two polynomials, or find a name in the list).
- Assume that these operations already exist in the form of procedures.
- Write an algorithm which solves the problem according to the requirements.
- Use a notation which is natural to the way you wish to describe the order of processing.

For More Visit : KtuQbank.com | Fair Use Policy

KtuQbank

System Life Cycle

Analysis- If there is another algorithm which solves the same problem, write it down.

- Next, try to compare these two methods.
- It may already be possible to tell if one will be more desirable than the other.
- If you can't distinguish between the two, choose one to work on for now and we will return to the second version later

KtuQbank

System Life Cycle

Refinement and coding

- Choose representations for data objects (a maze as a two dimensional array of zeros and ones, a polynomial as a one dimensional array of degree and coefficients, a list of names possibly as an array)
- Write algorithms for each of the operations on these objects.
- The order in which you do this may be crucial, because once you choose a representation, the resulting algorithms may be inefficient.

KtuQbank

System Life Cycle

- Verification
    - Verification is concerned about the correctness of process

For More Visit : KtuQbank.com | Fair Use Policy

KtuQbank

Algorithm

- Algorithm is a step by step procedure, which defines a set of instructions to be executed in certain order to get the desired output.
- Algorithms are generally created independent of underlying languages, i.e. an algorithm can be implemented in more than one programming language.

KtuQbank

Characteristics of an Algorithm

- **Unambiguous** – Algorithm should be clear and unambiguous.
- Each of its steps or phases, and their input/outputs should be clear and must lead to only one meaning.
- **Input** – An algorithm should have 0 or more well defined inputs.
- **Output** – An algorithm should have 1 or more well defined outputs, and should match the desired output.
- **Finiteness** – Algorithms must terminate after a finite number of steps.
- **Feasibility** – Should be feasible with the available resources.
- **Independent** – An algorithm should have step-by-step directions which should be independent of any programming code.

KtuQbank

Algorithm

- Finite step- by-step sequence of instructions for solving a particular problem.
- Writing and executing programs and then optimizing them will be effective for smaller programs.
- Optimization of a program is directly related with the algorithm

  design.
- For a large program, each part of the program must be well organized before writing the program.
- There are few steps of refinement involved when a problem is converted to program- called **stepwise refinement method**.

KtuQbank

KtuQbank

Algorithm

- We can write an <u>informal algorithm- </u>If we have an appropriate mathematical model for a problem.

- The initial version of the algorithm will contain general statements, i.e., informal instructions.

- Then we convert this informal algorithm to <u>formal algorithm- </u>with more definite instructions by applying any programming language syntax and semantics.

- Finally a program can be developed by converting the formal algorithm by a programming language manual.

For More Visit : KtuQbank.com | Fair Use Policy

KtuQbank

KtuQbank

Algorithm

- So, there are several steps to reach a program from a mathematical model.
- In every step, there is a refinement(or conversion).
- That is to covert an informal algorithm to a program, we must go through several stages
- Three steps in refinement process:

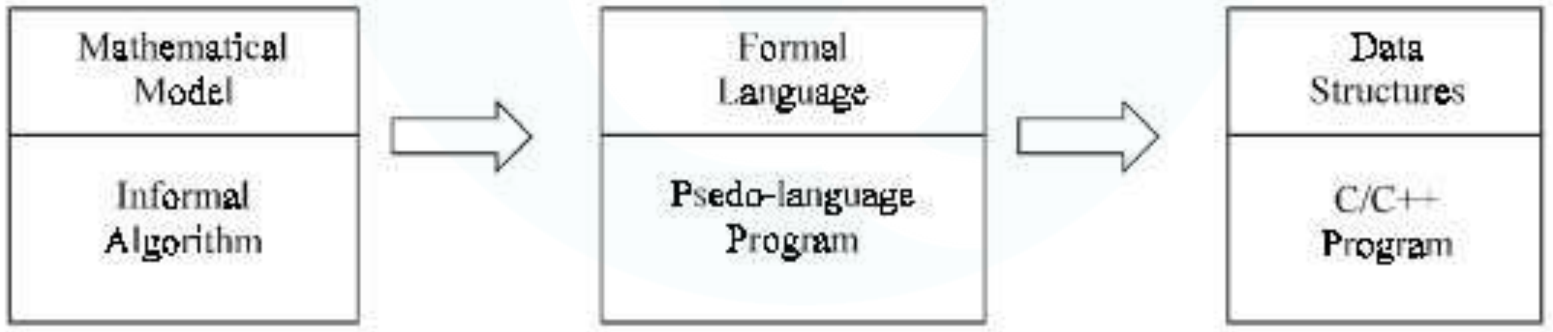


Fig. 1.1

KtuQbank

KtuQbank

Algorithm

- **In the first stage**, modeling, we try to represent the problem using an appropriate mathematical model such as a graph, tree etc.
- At this stage, the solution to the problem is an algorithm expressed very informally.

- **During next stage,** the algorithm is written in pseudo-language (or formal algorithm) which is a mixture of programming language constructs and less formal English statements.
- Here the operations to be performed on the various types of data become fixed.

- **In the final stage**, we choose an implementation and write the procedures for the various operations on that type. The remaining informal statements in the algorithm are replaced by C/C++ codes.

For More Visit : KtuQbank.com | Fair Use Policy

KtuQbank

KtuQbank

Analysis of algorithms

- After designing an algorithm, it has to be checked and its correctness needs to be predicted.
- This is done by analyzing the algorithm.
- An algorithm is analyzed with reference to the following:
    - Correctness
    - Execution time
    - Amount of memory used
    - Simplicity and clarity

For More Visit : KtuQbank.com | Fair Use Policy

KtuQbank

KtuQbank

Analysis of algorithms

- Algorithm Analysis Efficiency of an algorithm can be analyzed at two different stages, before implementation and after implementation, as mentioned below
- **A priori analysis**
- **A posterior analysis**

For More Visit : KtuQbank.com | Fair Use Policy

KtuQbank

KtuQbank

Algorithm Complexity

- Efficiency of algorithms
- **Algorithm complexity** is a **measure** which evaluates the order of the **count of operations**, performed by a given or algorithm as a function of the size of the input data.
- Complexity is a rough approximation of the number of steps necessary to execute an algorithm.

1) Space complexity
2) Time complexity

For More Visit : KtuQbank.com | Fair Use Policy

KtuQbank

- Complexity can be **constant**, **logarithmic**, **linear**, **n*log(n)**, **quadratic**, **cubic**, **exponential**, etc.

For More Visit : KtuQbank.com | Fair Use Policy

Analysis of algorithms

- **Best case, worst case and average case of an algorithm**
  - The best case implies minimum execution time that the algorithm would demand.
  - The worst case implies maximum execution time.
  - The average case indicates the behavior of the algorithm in an average situation.

For More Visit : KtuQbank.com | Fair Use Policy

KtuQbank

Asymptotic complexity

- Complexity of an algorithm is usually a function of n.
- Behavior of this function is usually expressed in terms of one or more standard functions.
- Expressing the complexity function with reference to other known functions is called asymptotic complexity.
- Three basic notations are used to express the asymptotic complexity- Big-oh(O), Big-omega(Ω), Theta(θ)

For More Visit : KtuQbank.com | Fair Use Policy

KtuQbank

- Asymptotic notations are the mathematical notations used to describe the running time of an algorithm when the input tends towards a particular value or a limiting value.
- For example: In bubble sort, when the input array is already sorted, the time taken by the algorithm is linear i.e. the best case.
- But, when the input array is in reverse condition, the algorithm takes the maximum time (quadratic) to sort the elements i.e. the worst case.
- When the input array is neither sorted nor in reverse order, then it takes average time. These durations are denoted using asymptotic notations.
- There are mainly three asymptotic notations: Theta notation, Omega notation and Big-O notation.

Big oh (O) Notations

- Formal method of expressing the upper bound of an algorithm's running time.
- i.e. it is a measure of longest amount of time it could possibly take for an algorithm to complete.
- It is used to represent the worst case complexity.

Definition:

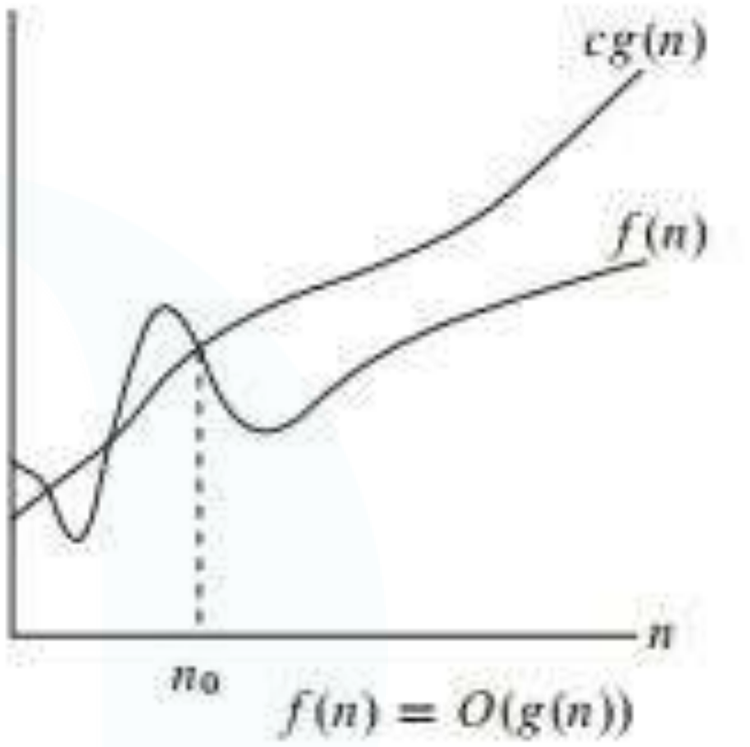
- For non negative functions f(n) and g(n), f(n)=O(g(n)) if there exist an integer n_0 and a constant C>0 such that for all integers $n>n_0$, $f(n) \leq C. g(n)$ then

- f(n)=O(g(n)) implies the behavior of f(n) with reference to another function g(n) and it signifies that g(n) is the upper bound of f(n)

KtuQbank

Big oh (O) Notations

Graphical representation



It is represented mathematically as $0 \leq f(n) \leq C.\ g(n))$

- For all values of n to the right of n_0, the value of f(n) is on or below C. g(n)

O(g(n)) = { f(n): there exist positive constants c and n_0 such that $0 \leq f(n) \leq cg(n)$ for all $n \geq n_0$ }

For More Visit : KtuQbank.com | Fair Use Policy

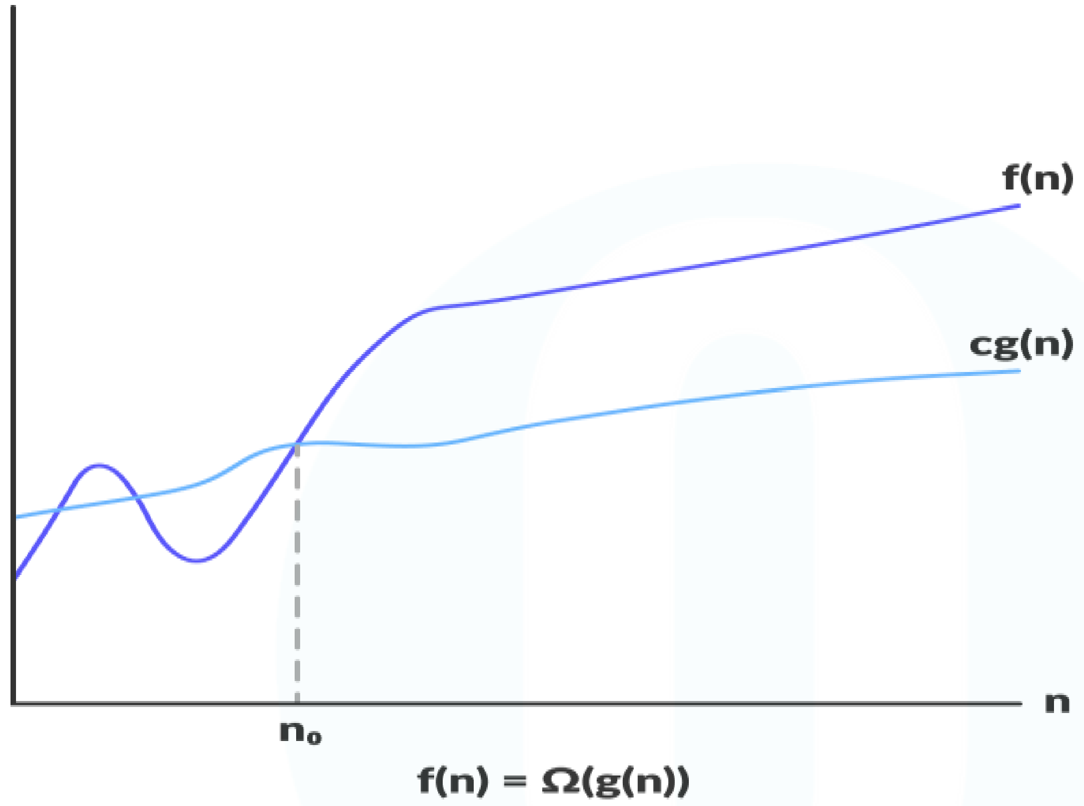
KtuQbank

Omega Notation (Ω-notation)

- Omega notation represents the lower bound of the running time of an algorithm.
- Thus, it provides best case complexity of an algorithm.

KtuQbank

KtuQbank



$\Omega(g(n))$ = { f(n): there exist positive constants c and n_0 such that $0 \le cg(n) \le f(n)$ for all $n \ge n_0$ }

The above expression can be described as a function f(n) belongs to the set Ω(g(n)) if there exists a positive constant c such that it lies above cg(n), for sufficiently large n.
For any value of n, the minimum time required by the algorithm is given by Omega Ω(g(n))

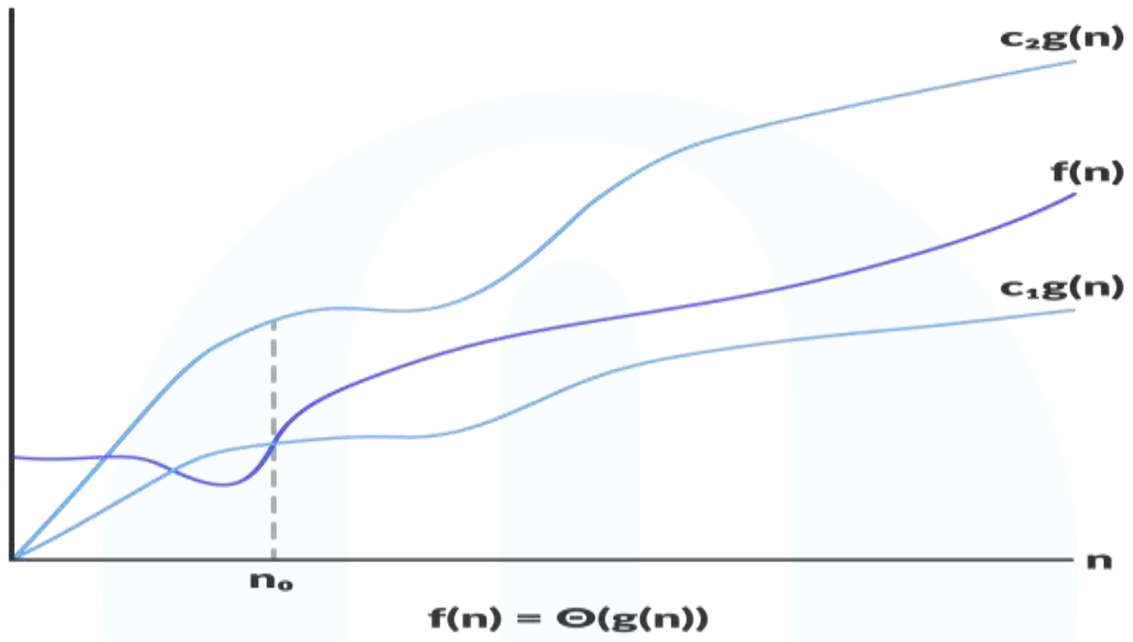
KtuQbank

Theta Notation (Θ-notation)

- Theta notation encloses the function from above and below. Since it represents the upper and the lower bound of the running time of an algorithm, it is used for analyzing the average case complexity of an algorithm.

For More Visit : KtuQbank.com | Fair Use Policy

KtuQbank



$\Theta(g(n))$ = { $f(n)$: there exist positive constants c_1, c_2 and n_0 such that $0 \le c_1g(n) \le f(n) \le c_2g(n)$ for all $n \ge n_0$ }

The above expression can be described as a function $f(n)$ belongs to the set $\Theta(g(n))$ if there exist positive constants c_1 and c_2 such that it can be sandwiched between $c_1g(n)$ and $c_2g(n)$, for sufficiently large n.

Complexity of an algorithm(evaluation of algorithm)

- The ***complexity*** of an algorithm is the *function*, which give the running time or storage space in terms of input size.
  - **Space Complexity:** How much space an algorithm needs to complete its task.
  - **Time Complexity:** Running time of a program as a function of input size.
- Analysis of algorithms is the determination of the amount of time and space resources required to execute it.

For More Visit : KtuQbank.com | Fair Use Policy

KtuQbank

Space Complexity

- ➢Space complexity = The amount of memory required by an algorithm to run to completion
- Two part :

1. Fixed part: independent of Input & output characteristics:
   - ✓Instruction space
   - ✓space for variables
   - ✓space for constants & so on
2. Variable part: Space needed by variables, whose size is dependent on the size of the problem:
   - ✓Space needed by referenced variables,
   - ✓recursion stack space,etc

KtuQbank

Calculation

$$S(p)=C+Sp$$

- Let p be the algorithm
- S(p) :space complexity
- C:fixed space
- Sp:variable space

For More Visit : KtuQbank.com | Fair Use Policy

KtuQbank

Example 1

```
void main()
{
int x,y,z,sum;
printf(“enter 3 numbers”);
scanf(“%d%d%d”,&x,&y,&z);
Sum=x+y+z;
Printf(“%d”,sum);
}
```

- Variables used are:
- Variable x: 1 word
- Variable y: 1 word
- Variable z: 1 word
- Variable sum: 1 word
- Total space:4 word

Example 2

```
Algoritm Sum(a,n)
{     S=0;
        for i=1 to n
           S=s+a[i];
              Return s;   }
```

- Size variable n:1 word
- Loop variable i:1 word
- Sum variable s:1 word
- Array a values: n words
- Total space:n+3 words

KtuQbank

Example 3

```
int main()
{
int a = 5,b = 5,c;
 c = a + b;
printf("%d", c);
}
```

KtuQbank

Example 4

```
Algoritm Sum(a,n)
{       S=0;
          for i=1 to n
          for j=1 to m
             S=s+a[i][j];
                }
```

- Size variable n:1 word
- Loop variable i:1 word
- Sum variable s:1 word
- Array a values: n*m words
- Total space:nm+3 words

KtuQbank

KtuQbank

Time complexity:

- How much time needed for the completion of a program

<u>2 components</u>:

1. Compile time :Does not depend on instance characteristics.
2. Run time:Dependent on particular problem instance

- **Time complexity :T(P)=C+Tp**
    - T(p):time complexity of algorithm P
    - C:Compile time
    - Tp :run time

For More Visit : KtuQbank.com | Fair Use Policy

KtuQbank

Time Complexity: Frequency count

➢ Time complexity can be calculated by counting the <u>number of steps</u> each statement in the program executes

Eg:

- Comments-0 steps
- Assignment statement-1 step
- Condition statement-1 step
- Loop condition for ‘n’ times-(n+1) steps
- Body of loop-N steps

For More Visit : KtuQbank.com | Fair Use Policy

KtuQbank

Frequency count

- Frequency count is, the number of times the statement is executed in the program.

- Let’s consider some of the program segments

- **Program Segment A:**

```
…
x = x+2
…
```

KtuQbank

KtuQbank

Frequency count

- **Program Segment B:**

```
...
for k =1 to n do
        x = x+2
end
...
```

- **Program Segment C:**

```
...
for j= 1 to n do
        for x = 1 to n do
                x = x+2
        end
end
```

For More Visit : KtuQbank.com | Fair Use Policy

KtuQbank

KtuQbank

Frequency count

- The frequency count of the statement in the program segment A is 1.(It gets executed once)

- In the program segment B, the frequency count of the statement is n, since the **for** loop in which the statement is embedded executes n(n>=1) times.

- In the program segment C, the statement is executed n^2(n>=1) times, since the statement is embedded in a nested **for** loop, executing n times each.

For More Visit : KtuQbank.com | Fair Use Policy

KtuQbank

Example 1

```
Void hello()
{
    for(i=1; i <=n; i++)          →n+1 times
      {
       print(”Hello”;  );         →n times
       }
}
```

- Add all statement:n+1+n=<u>2n+1 times</u>

KtuQbank

Example 2

```
float sum(float a[ ], int n)
{
        float temp = 0;                    →1 times
        int i;
        for(i=0; i <n; i++)                →n+1 times
            {
              temp=temp+a[i];              → n times
            }
        return (temp);                     →1 times
}
```

Total time: 2n+3 times

Example 3

```
Void add ()
{
   int i, j,m,n,a[];
   for (i = 0; i < n; i++)          →n+1 times
     for (j=0; j< m; j++)           →n(m+1) times
      s=s+a[i][j];                  →n*m times
}
```

Total:n+1+nm+n+nm=**2nm+2n+1**

KtuQbank

Different programming methodologies

- Deals with different methods of designing programs.
- Modular programming:
  - Procedural programming
  - The focus is entirely on writing code(functions)
  - Data is passive
  - Any code may access the contents of any data structure passed to it- i.e. no encapsulation.
  - Here programs are written as functions
  - Minimal interaction between functions.
  - Easy to detect error and to test it in isolation
  - It discourages the use of control variables and flags in parameters.
  - It encourages splitting the functionality into two types
    - Master and slave
- Two methods Modular programming:
  - Top-Down and Bottom-Up

KtuQbank

KtuQbank

Types of Programming Methodologies

1. Procedural Oriented Programming
2. Object-Oriented Programming
3. Structured Programming
4. Modular Programming

For More Visit : KtuQbank.com | Fair Use Policy

KtuQbank

1)PROCEDURAL ORIENTED PROGRAMMING	2)OBJECT ORIENTED PROGRAMMING
In procedural programming, program is divided into small parts called ***functions***.	In object oriented programming, program is divided into small parts called ***objects***.
Procedural programming follows ***top down approach***.	Object oriented programming follows ***bottom up approach***.
There is no access specifier in procedural programming.	Object oriented programming have access specifiers like private, public, protected etc.
Adding new data and function is not easy.	Adding new data and function is easy.
Procedural programming does not have any proper way for hiding data so it is ***less secure***.	Object oriented programming provides data hiding so it is ***more secure***.
In procedural programming, overloading is not possible.	Overloading is possible in object oriented programming.
In procedural programming, function is more important than data.	In object oriented programming, data is more important than function.
Procedural programming is based on ***unreal world***.	Object oriented programming is based on ***real world***.
Examples: C, FORTRAN, Pascal, Basic etc.	Examples: C++, Java, Python, C# etc.

KtuQbank

Different programming methodologies

3)Modular programming:

- Procedural programming
- The focus is entirely on writing code(functions)
- Data is passive
- No data encapsulation.
- Here programs are written as functions
- Minimal interaction between functions.
- Easy to detect error and to test it in isolation

- Two methods Modular programming:
  a) Top-Down approach
  b) Bottom-Up approach

For More Visit : KtuQbank.com | Fair Use Policy

KtuQbank

Modular programming:

a)Top-Down design:

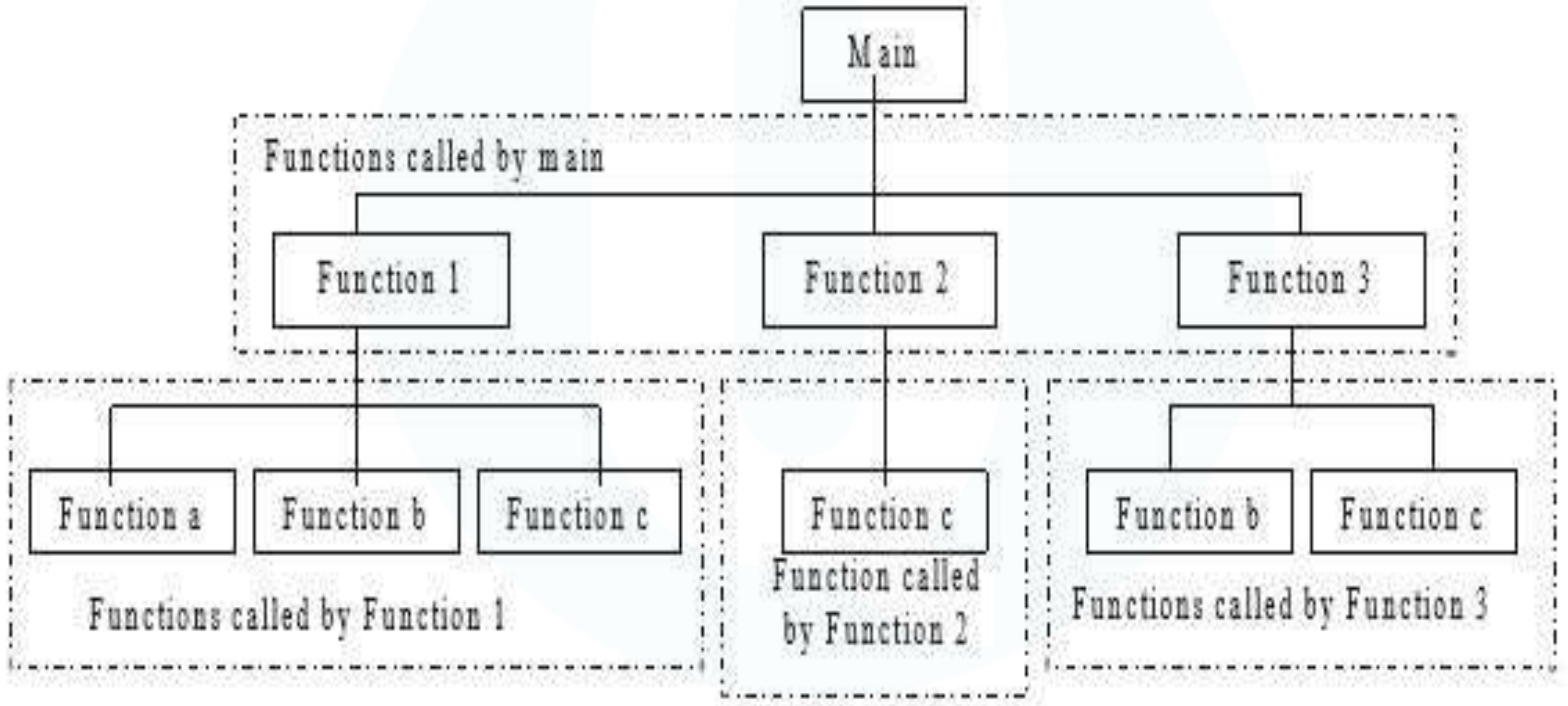
- Here a program should be divided into a main module and its related modules.
- Each module should also be divided into sub modules according to programming style.
- The division process continues, until the module consists only of elementary process that are understood and cannot be further subdivided.
- Here modules are arranged in a hierarchical manner.
- Here each module have single entry and single exit point.
- In C, It is done using function.

KtuQbank

KtuQbank

Modular programming:

- <u>Top-Down algorithm design:</u>



KtuQbank

KtuQbank

Modular programming:

- <u>Bottom-up algorithm design:</u>
  - Opposite of top-down design.
  - Starting the design with specific modules and build them into more complex structures, ending at the top.
  - Widely used in testing.
  - Here lowest level functions are written and tested first.
  - This process continues, moving up the level, until the main function is tested.

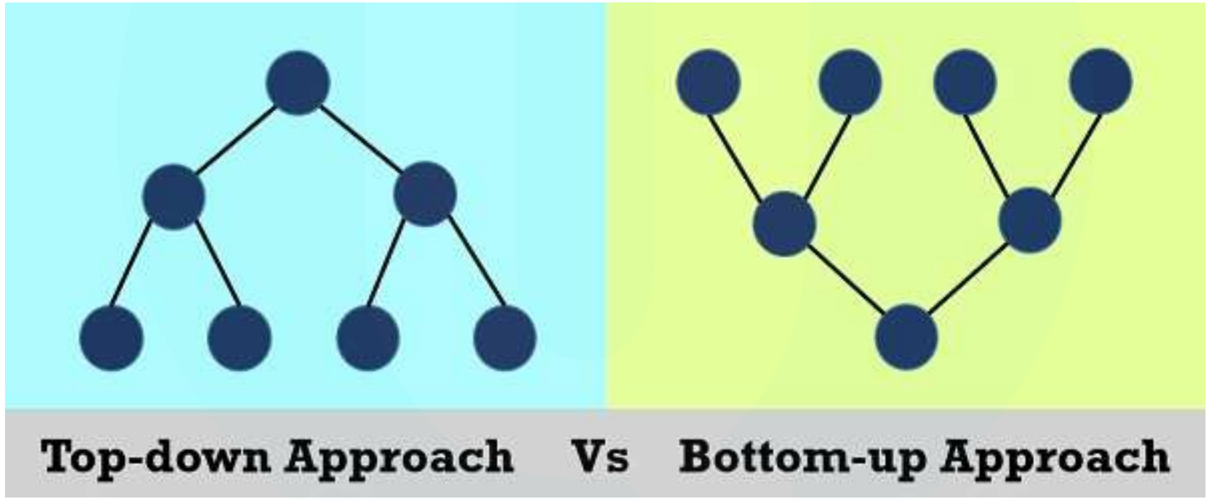
For More Visit : KtuQbank.com | Fair Use Policy

KtuQbank

S.NO.	TOP DOWN APPROACH	BOTTOM UP APPROACH
1.	In this approach We focus on breaking up the problem into smaller parts.	In bottom up approach, we solve smaller problems and integrate it as whole and complete the solution.
2.	Mainly used by structured programming language such as COBOL, Fortan, C etc.	Mainly used by object oriented programming language such as C++, C#, Python.
3.	Each part is programmed separately therefore contain redundancy.	Redundancy is minimized by using data encapsulation and data hiding.
4.	In this the communications is less among modules.	In this module must have communication.
5.	It is used in debugging, module documentation, etc.	It is basically used in testing.
6.	In top down approach, decomposition takes place.	In bottom up approach composition takes place.
7.	In this top function of system might be hard to identify.	In this sometimes we can not build a program from the piece we have started.
8.	In this implementation details may differ.	This is not natural for people to assemble.

KtuQbank

KtuQbank



KtuQbank

Programming methodologies

- <u>Structured programming:</u>
  - It is a Programming style.
  - <u>Not</u> programming with structures, but by using following types of codes structures to write programs:
    - 1. Sequence of sequentially executed statements.
    - 2. Conditional execution of statements(i.e. “if” statements)
    - 3. Looping or iteration(i.e. “for, do…while and while” stmts)
    - 4. Structured subroutine calls(i.e. functions)

- <u>Structured programming:</u>
  - Here, the following language usage is forbidden(not allowed)
    - 1. GoTo statements
    - “Break” or “Continue” out of the middle of loops
    - Multiple exit points or multiple “return” statements to a function/ procedure/subroutine
    - Multiple entry points to a function/ procedure/subroutine
  - In structured programming, there is a great risk that the implementation details of many data structures have to be shared between functions and thus globally exposed.
  - This tempts other functions to use there implementation details, there by creating unwanted dependencies between different parts of the program.

For More Visit : KtuQbank.com | Fair Use Policy

KtuQbank

- **Structured programming:**

  - The main disadvantage is that all decisions made from the start of the project depends directly or indirectly on the high-level specification of the application.

  - This specification tends to change over time.

  - So, there is a great risk that, large part of the application need to be rewritten.

For More Visit : KtuQbank.com | Fair Use Policy

KtuQbank

KtuQbank

Growth Rate

- Functions in order of increasing growth rate is as follows
  - 1
  - log N
  - N
  - N log N
  - N^2
  - N^3
  - 2^N

For More Visit : KtuQbank.com | Fair Use Policy

KtuQbank

	constant	*logarithmic*	*linear*	*N-log-N*	*quadratic*	*cubic*	*exponential*
n	**O(1)**	**O(log *n*)**	**O(*n*)**	**O(*n* log *n*)**	**O(n^2)**	**O(n^3)**	**O(2^n)**
1	1	1	1	1	1	1	2
2	1	1	2	2	4	8	4
4	1	2	4	8	16	64	16
8	1	3	8	24	64	512	256
16	1	4	16	64	256	4,096	65536
32	1	5	32	160	1,024	32,768	4,294,967,296
64	1	6	64	384	4,069	262,144	1.84 x 10^{19}

KtuQbank

THANK YOU

For More Visit : KtuQbank.com | Fair Use Policy

KtuQbank