SAMPLE CONTENT

Comprehensive Text Book series

Std XII

BIFOCAL SCIENCE

COMPUTER SCIENCE – I

Std. XII (SYJC) Science

(Course Code : D-9)

Ms. Shraddha S. More
St. John COE and Management, Palghar (E)

Er. Manoj S. Kavedia
Thadomal Shahani Engineering College (TSEC), Bandra, Mumbai

Dr. Nilesh M. Patil
SVKM's D J Sanghvi COE, Mumbai

Prof. Santosh Kabir
(Visiting Faculty)
ARMIET Engg College, Asangaon.
Chetana College (BSc IT, CS)
Bandra

FEATURES

1. Fully Solved March 2023 Board Question Paper.
2. Includes solutions to previous 10 years Board Question Papers.
3. Programs are displayed with Output.
4. Board Questions are strategically placed in the flow of the chapter to benefit students from examination purview.

- A Joint Venture of -

Target Publications® Pvt. Ltd.
Transforming lives through learning
www.targetpublications.org mail@targetpublications.org

&

TECH-NEO PUBLICATIONS
Where Authors Inspire Innovation
A Sachin Shah Venture
www.techneobooks.in info@techneobooks.in

Computer Science - I

(Code : D-9)

Std. XII (S.Y.J.C.) Science

BIFOCAL SCIENCE

Strictly As per HSC Board Syllabus of Higher Secondary Education

Dr. Nilesh Madhukar Patil
Ph.D. (Computer Engineering)
Associate Professor, Computer Engg. Department,
SVKM's D J Sanghvi College of Engineering, Mumbai

Er. Manoj S. Kavedia
Assistant Professor
Department of Electronics and Telecommunication
Thadomal Shahani Engineering College (TSEC), Bandra, Mumbai

Ms. Shraddha Subhash More
M.E. (Information Technology)
Assistant Professor,
Department of Information Technology
St. John College of Engineering and Management, Palghar (E)

Prof. Santosh Kabir
B.E. Electronics
Visiting Faculty : ARMIET Engg college, Asangaon.
Chetana College (BSc IT, CS)
Bandra

- A Joint Venture of -

Target Publications® Pvt. Ltd.
Transforming lives through learning
www.targetpublications.org mail@targetpublications.org

&

TECH-NEO PUBLICATIONS
Where Authors Inspire Innovation
A Sachin Shah Venture
www.techneobooks.in info@techneobooks.in

Printed at: **Print to Print,** Mumbai

P.O. No. 9435

TEID: 3171

Computer Science - I (Code : D-9)

Std. XII : Science (S.Y.J.C.)

- **Authors** : Dr. Nilesh Madhukar Patil, Er. Manoj S. Kavedia, Ms. Shraddha Subhash More, Prof. Santosh Kabir
- **First Edition** : June 2023
- **ISBN** : 978-93-5583-404-1

Target Publications Pvt. Ltd	**Tech-Neo Publications, LLP**
B2, 9th Floor, Ashar IT Park, Road No. 16/Z, Wagle Industrial Estate, Thane (W) - 400604 **Phone : + 91 8879 9397 14 / 15** ***Email :*** mail@targetpublications.org **Website :** www.targetpublications.org	Dugane Ind. Area, Survey No. 28/25, Dhayari, Near Pari Company, Pune - 411041. Maharashtra State, India. **Phone : + 91 9850429188** **Email :** info@techneobooks.in **Website :** www.techneobooks.in

- For Writing/Editing Books, Suggestions/Critics – Please email : info@techneobooks.in
- For Sales Enquiries/Orders/Library Orders - Please email : mail@targetpublications.org

CAUTION : PHOTOCOPYING OF COPYRIGHTED BOOK IS ILLEGAL

SAVE YOURSELF, DON'T BUY PHOTOCOPIED BOOKS

Books Published are protected under **Copyright Act 1999** and sold subject to the condition that the book and any extract thereof **shall be not photocopied** and includes the said condition being imposed on any subsequent purchaser.

Any person found selling, stocking or carrying photocopied book may be arrested for indulging in criminal offence of copyright piracy and may be imprisoned for **3 years and also fined a sum of Rs. 2,00,000/- for first offence**.

Sharing of PDF's, any Drives, Links, Storing in Hard Disks, Pendrive and Circulating on Social Media like Instagram, Telegram, Facebook, Snapchat, Google Drive & Whatsapp etc also violates the Copyright Laws and will be reported to **Cyber Crime Division.**

Publisher has raided many such offenders. Their Machines were Seized. Criminal case has also been registered against them. Civil Suits are also filed for recovering damages. **Police investigations of Students who are indulged in this is also in process.**

Recently, the Supreme Court of India, in ***M/s Knit Pro International v. The State of NCT of Delhi*** on 20 May 2022, has observed and held that offences under **Section 63** of the Copyright Act, 1957 ("Copyright Act") are **cognizable and non–bailable**.

"Name of informer will be kept highly confidential. On successful raid he will be suitably rewarded"

Call / WhatsApp us on +91 98504 29188
Email : info@techneobooks.in

TECH-NEO PUBLICATIONS
Where Authors Inspire Innovation
A Sachin Shah Venture

Tech-Neo Publications LLP
Dugane Ind. Area, Survey No. 28/25, Dhayari, Near Pari Company, Pune, Maharashtra, Pune-411041.
Email : info@techneobooks.in ● **Website :** www.techneobooks.in

Preface

Dear Student,

We are extremely happy to present the book of "**Computer Science – I**" for you. The topics within the chapters have been arranged in a proper sequence to ensure smooth flow of the subject.

Salient Features of the Book are as follows :

1. Selective Board Examination Questions till March 2023 have been fully solved in this edition.

2. Theory is accompanied with neat & clean figures.

3. MCQ's with explanation are also included at the end of each chapter. The latest trend in education is the teaching through multiple choice questions. The MCQ's are intended to enable students to prioritise and plan their learning through regular practice. The book contains large number of multiple choice questions on the subject.

4. Each chapter is divided into various sections and sub-sections. Entire syllabus is divided into Chapters, sections and headings. Each paragraph has been given a unique section/subsection number which is used to explain that particular section for the students as a cross reference to enable them to refer to the related paragraph.

5. Through this book, the author has made an effort to provide rationale for the solutions. The book, therefore, meets the expectations of the students as it answers the demand and the quest in their mind. It would give rise to real learning which would stand in good stead for the student's career and his life.

6. The book is user-friendly and provides information in a well structured manner. It provides comprehensive and critical study of the various concepts of the subject matter. It is felt that the contents should be crystal clear.

7. Programs are made simple & Outputs are displayed.

A word or suggestion from your side may help us add another feather to the cap of the subject matter of the book. The author looks forward to the comments, suggestions and criticism from the readers. Constructive suggestions and feedback from users would be highly appreciated, acknowledged and suitably incorporated

We are thankful to team of Target Publications and Tech Neo Publications for the encouragement and support that they have extended to us.

- Authors

Syllabus

STD. XII - S.Y.J.C. (Science)

Computer Science - I

1. Operating Systems

What is operating system: Services in OS; Overview of OS: WINDOWS 98, WINDOWS NT, LINUX: Concepts related to Information management (Only definition), File System, Device Drivers, Terminal I/O; Concepts related to process management (Only definition), Process, Multiprogramming, Context switching. Process States, Priority, Multitasking. Timesharing: Concepts Related to Memory Management (Only definition), A typical map for single user computer. Partitioning, Fixed & Variable Partitioning. Paging, Segmentation, Virtual memory; GUI : Basic GUI features such as Windows, Task List, Drag, Resize, Close, Minimize, Maximize; Access and security aspect of OS, security Threats, Attacks on Security, Computer Worms, Computer Viruses. **(Refer Chapter 1)**

2. Data Structures

Introduction to data structures, Data structure operations, Algorithmic notation, Control structures; Arrays Representation in memory. Traversing, Inserting, Deleting. Sorting, Binary search in an array, Pointer arrays, Records in memory using arrays; Link list, Representation of link list in memory, Trees, Binary trees, Representing binary tree in memory.**(Refer Chapter 2)**

3. C++

Review of C++; Arrays, pointers, references, strings; principles of object oriented programming; classes and objects, Constructors and destructors; Operator overloading and type conversion, virtual functions and polymorphism; inheritance : working with files.

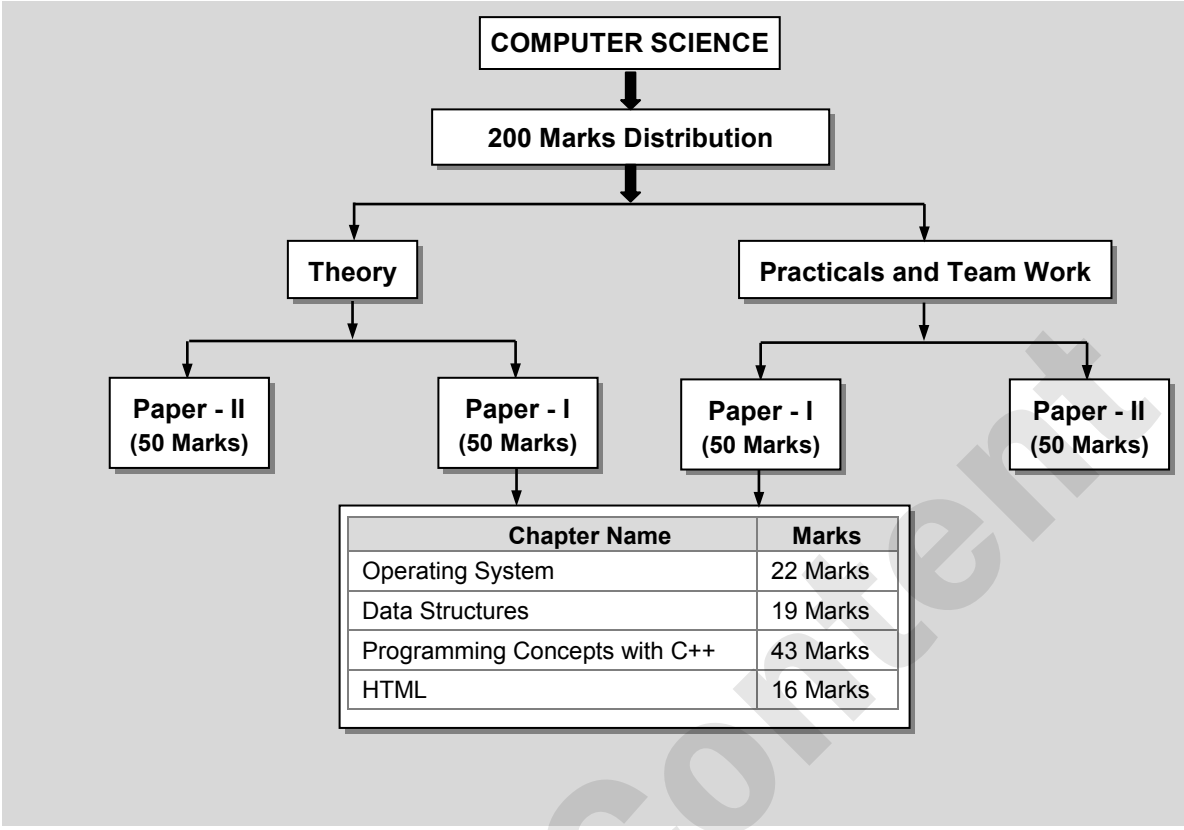
(Refer Chapter 3)

4. HTML

Introduction to HTML; Why HTML, Its advantages and Drawbacks, Study of Tags :<HTML>, <HEAD> <TITLE>, <BODY>, <P>,
, <UL>, <LI>, <PRE>, <MARQUEE> Font Styles :<B>, <I>, <U>, <BIG> <SMALL>, <SUB>, <SUP>, <FONT>; Images : <HREF>,<HR><IMG>, SRC, ALT, HEIGHT, WIDTH, ALIGN TABLES: <TABLE>, <CAPTION>,<TR>,<TH>,<TD>; Use of Scripting as a language support (NOTE : Only VB Script using FOR, NEXT, IF..THEN..ELSE, MsgBox, InBox, DIM, SET)

(Refer Chapter 4)

❑❑❑



Chapter Name	Marks
Operating System	22 Marks
Data Structures	19 Marks
Programming Concepts with C++	43 Marks
HTML	16 Marks

Question Paper Format

Each Question Paper will have Five Main Questions

Q. 1(A)	4 MCQs - 1 Mark each (Compulsory)	4 Marks
Q. 1(B)	**(a)** - 3 Marks **(b)** - 3 Marks **(c) -** 3 Marks *(Attempt any Two)*	6 Marks
Q. 2(A)	**(a)** - 3 Marks **(b)** - 3 Marks **(c)** - 3 Marks *(Attempt any Two)*	6 Marks
Q. 2(B)	**(a)** - 4 Marks **(b)** - 4 Marks *(Attempt any One)*	4 Marks
Q. 3(A)	**(a)** - 3 Marks **(b)** - 3 Marks **(c)** - 3 Marks *(Attempt any Two)*	6 Marks
Q. 3(B)	**(a)** - 4 Marks **(b)** - 4 Marks *(Attempt any One)*	4 Marks
Q. 4(A)	**(a)** - 3 Marks **(b)** - 3 Marks **(c)** - 3 Marks *(Attempt any Two)*	6 Marks
Q. 4(B)	**(a)** - 4 Marks **(b)** - 4 Marks *(Attempt any One)*	4 Marks
Q. 5	**(a)** - 5 Marks **(b)** - 5 Marks **(c)** - 5 Marks *(Attempt any Two)*	10 Marks
	OR	
Q. 5	**(a)** - 5 Marks **(b)** - 5 Marks **(c)** - 5 Marks *(Attempt any Two)*	10 Marks
	Total :	**50 Marks**

Table Of Contents

❑❑❑

CHAPTER

2 Data Structures

2.1 INTRODUCTION TO DATA STRUCTURES

Board Exam Question

Q. What is data structure ?

(Oct. 04, March 06, 18, 1 Mark)

i. Data structures are an essential part of computer science and programming. They provide a way to organize and store data in a computer's memory so that it can be efficiently accessed and manipulated.

ii. Understanding data structures is crucial for developing efficient algorithms and solving complex problems in programming.

iii. **Definition :** *A data structure is basically a group of data elements that are put together under one name, and which defines a particular way of storing and organizing data in a computer/memory so that it can be used efficiently.*

iv. Data structures can be thought as concept where an algorithm or set of algorithms are used to structure the ordered data/information while storing.

v. These algorithms can be implemented in any programming language. Widely used among them are C, C++, Java, .Net, Python etc.

vi. When we choose programming language like C or C++, we need to write code for these algorithms whereas languages such as Java and .Net have collection framework where predefined implementations for different data structures are provided.

vii. Data structures are used in almost every program or software system. They serve as building blocks of a program.

viii. A program built using improper data structures may not work as expected. So as a programmer it is mandatory to choose most appropriate data structures for a program.

ix. Every significant program will use one/some data structure(s) explicitly. For e.g., a stack is implicitly used in a program, whether or not you declared it.

x. Some common examples of data structures are arrays, linked lists, queues, stacks, binary trees, and hash tables.

xi. Data structures are widely applied in the following areas :

a. Compiler design
b. Operating system
c. Data analytics
d. Database Management Systems
e. Numerical analysis
f. Simulation
g. Artificial intelligence
h. Graphics
i. Social Networking
j. Data Science

2.1.1 Need of Data Structure

The need for data structures arises from the requirement to efficiently organize and manage data in computer programs. Here are some key reasons why data structures are essential:

i. Efficient Data Storage : Data structures allow for efficient storage and retrieval of data. By organizing data in a structured manner, data structures enable faster access and manipulation operations, reducing the time complexity of algorithms.

ii. Effective Data Organization : Data structures provide a systematic way to organize and represent data based on specific requirements. Different data structures are designed to handle different scenarios and operations, allowing for optimized data organization and management.

iii. Algorithm Design : Data structures play a crucial role in designing efficient algorithms. By choosing the appropriate data structure for a particular problem, you can significantly improve the efficiency and performance of algorithms. For example, using a hash table can provide constant-time access to elements, leading to faster search or retrieval operations.

iv. Memory Management : Data structures help in managing memory efficiently. They allow for dynamic allocation and deal location of memory, enabling the program to use memory optimally and avoid memory leaks.

v. Code Reusability : Data structures provide reusable components that can be utilized across different programs and applications. Once implemented, data structures can be used in various scenarios, saving time and effort in developing new solutions from scratch.

vi. **Problem Solving :** Data structures provide a framework for solving complex problems. They offer specific operations and algorithms tailored for tasks like searching, sorting, graph traversal, and more. By leveraging the appropriate data structure, you can simplify the problem-solving process and achieve optimal solutions.

vii. **Scalability :** As data sizes increase, the efficiency and scalability of data structures become crucial. Well-designed data structures can handle large volumes of data efficiently, allowing programs to scale and handle increasing demands without sacrificing performance.

2.1.2 Types of Data Structures

A. Primitive Data Structures

i. Primitive data structures, also known as basic or elementary data structures, are the fundamental building blocks provided by programming languages.

ii. These data structures are typically built-in and directly supported by the programming language itself. They are simple and represent the basic types of data that can be manipulated by programs. Here are some common examples of primitive data structures :

a. Integer : Also known as int, it represents whole numbers without any fractional parts. Integers are used to store values like 1, 2, -5, 100, etc.

b. Floating-point : Also known as float or double, it represents decimal numbers with a fractional part. Floating-point numbers can store values like 3.14, -0.5, 2.0, etc.

c. Character : Usually represented as char, it stores a single character, such as 'A', 'b', '@', '5', etc. Characters are often used to represent individual letters, digits, or symbols.

d. Boolean : Represented as bool, it has two possible values: true or false. Booleans are used to represent logical conditions or states, where true indicates a positive condition and false represents a negative condition.

iii. These primitive data structures are the foundation on which more complex data structures and algorithms are built. They have predefined operations and memory representations provided by the programming language.

iv. It's important to note that different programming languages may have slight variations in their primitive data types, but the fundamental concepts remain the same.

v. In addition to these basic data types, some programming languages may provide additional primitive data structures, such as :

a. **Floating-point with extended precision :** Some languages offer extended precision floating-point data types like long double, which provide more significant digits and increased precision compared to regular float or double types.

b. Strings : While strings are not considered primitive data structures in some languages, they are treated as a primitive type in others. A string represents a sequence of characters and is commonly used to store and manipulate text or textual data.

These primitive data structures provide the foundation for storing and manipulating data in programming languages. They are essential in performing basic calculations, comparisons, logical operations, and forming more complex data structures to solve various computational problems.

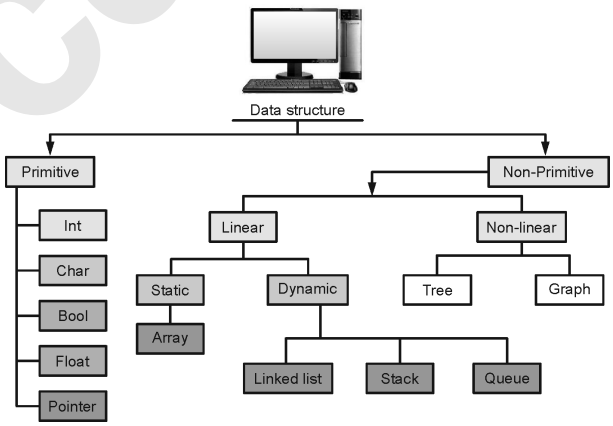
B. Non-Primitive Data Structures

Non-primitive data structures, also known as composite or derived data structures, are more complex data structures that are built using primitive data types or other non-primitive data structures. These data structures are designed to store and organize collections of data in a structured manner. They provide more flexibility and functionality compared to primitive data structures.

Here are some common examples of non-primitive data structures :

i. Arrays : Arrays are a collection of elements of the same type stored in contiguous memory locations. They provide efficient random access to elements using an index. Arrays are widely used due to their simplicity and efficiency for operations like searching and sorting.

ii. Linked Lists : Linked lists are a dynamic data structure where elements are stored in nodes that are linked together using pointers. Each node contains data and a reference to the next node. Linked lists are useful when you need efficient insertion and deletion operations, but random access is less important.



(1B1) **Fig. 2.1.1 : Types of Data Structures**

iii. Stacks : A stack is a Last-In-First-Out (LIFO) data structure where elements are added and removed from one end, called the top. It follows the principle of "last in, first out." Stacks are often used in programming for tasks like expression evaluation, function calls, and backtracking algorithms.

iv. Queues : A queue is a First-In-First-Out (FIFO) data structure where elements are added at one end, called the rear, and removed from the other end, called the front. It follows the principle of "first in, first out."

Queues are commonly used in scenarios like scheduling, buffering, and breadth-first search algorithms.

v. Trees : Trees are hierarchical data structures composed of nodes connected by edges. Each node can have zero or more child nodes. Trees are used in many applications, such as representing hierarchical relationships, organizing data efficiently, and implementing search algorithms like binary search trees.

vi. **Graphs :** Graphs are a collection of nodes connected by edges. They represent relationships between objects and are widely used in various fields like social networks, computer networks, and transportation networks. Graphs can be directed or undirected, and they can have weighted or unweighted edges.

vii. **Hash Tables :** Hash tables, also known as hash maps or dictionaries, are data structures that store data as key-value pairs. They use a hash function to map keys to specific locations in the underlying array, allowing for efficient retrieval and insertion operations.

Table 2.1.1 : Primitive Data Structure Vs Non-primitive Data Structure

Primitive data structure	Non-primitive data structure
Primitive data structure is a kind of data structure that stores the data of only one type and can hold only a single value in one specific location.	Non-primitive data structure is a type of data structure that can store the data of more than one type.
Examples of primitive data structure are integer, character, float. int a = 10; char b = 'A';	Examples of non-primitive data structure are Array, Linked list, stack. values= [10, 'A', "DSA"]
Primitive data structure will contain some value, i.e., it cannot be NULL.	Non-primitive data structure can consist of a NULL value.
The size depends on the type of the data structure.	In case of non-primitive data structure, size is not fixed.
It starts with a lowercase character.	It starts with an uppercase character.
Primitive data structure can be used to call the methods.	Non-primitive data structure cannot be used to call the methods.

C. Linear and Non-Linear Data Structures

Board Exam Question

Q. Explain Linear Date Structure and Non-linear Data structure. **(March 18, 3 Marks)**

i. Linear data structures organize and store data in a sequential manner, where each element has a predecessor and a successor, except for the first and last elements. Some commonly used linear data structures include Arrays, Linked Lists, Stacks, and Queues.

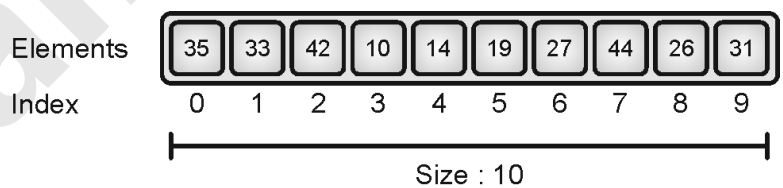
ii. Non-linear data structures organize and store data in a hierarchical or interconnected manner, allowing more complex relationships between elements. Some commonly used non-linear data structures include Trees, Graphs, Heaps, and Hash Tables.

iii. Non-linear data structures allow for more flexible relationships between data elements, while linear data structures are more straightforward and simpler to implement.

Table 2.1.2 : Linear Vs Non-linear Data Structure

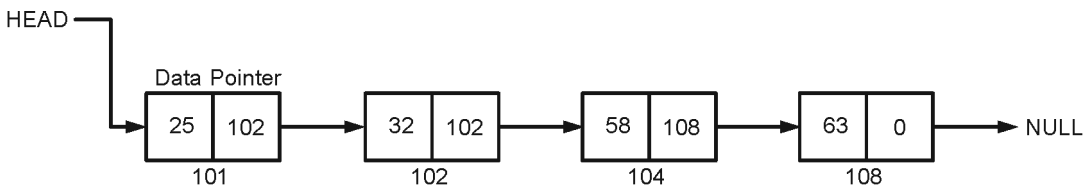
Linear Data Structures	Non-Linear Data Structures
Elements are ordered in a linear and sequential manner.	Elements are ordered in a hierarchy.
Only a single level is present.	Multiple level data structures are present.
Implementation is relatively easier.	Implementation is relatively complicated.
Traversed in a single run	Take multiple runs to traverse the data.
Memory utilization is not efficient compared to non-linear data structures	Memory utilization is efficient.
Examples: array, queue, linked list, etc.	Examples: trees, graphs, etc.

Representation of Linear Data Structure using Array



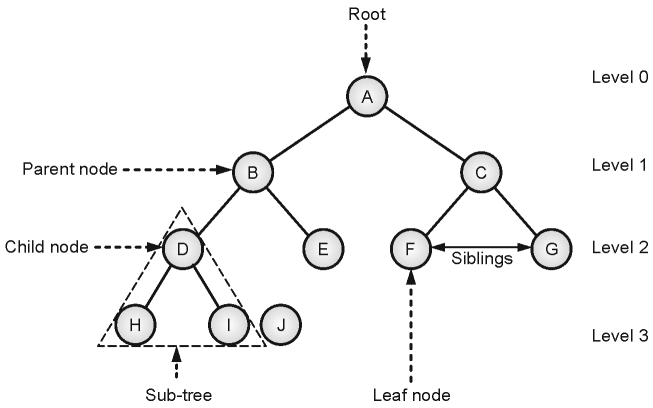
(1B2) **Fig. 2.1.2**

Representation of Linear Data Structure using Linked List



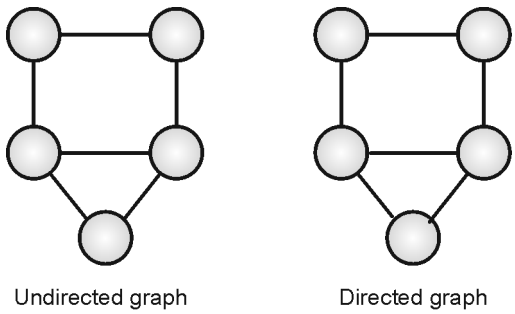
(1B3) **Fig. 2.1.3**

Representation of Non-linear Data Structure using Tree



(1B4) **Fig. 2.1.4**

Representation of Non-linear Data Structure using Graph



(1B5) **Fig. 2.1.5**

D. Record, File, Field and Key-Field

Board Exam Questions

Q. In case of data structure, explain the following terms
i. Field ii. Record iii. File
(March 05, 09, 16 Oct. 07, 14, 3 Marks)

Q. Define key-field. **(March 16, 1 Mark)**

Q. What is a Record ?
(Q. 2(A)(b), March 22, 3 Marks)

In data structures, the terms record, file, and field are used to describe different components of a data organization scheme. Let's define each term :

I. Record

i. A record is a collection of related data elements or fields that are grouped together as a single unit.
ii. It represents a complete set of information about a particular entity or object.
iii. Each field within a record holds a specific piece of data related to the entity.
iv. For example, if we have a record representing a student, the fields within that record could include the student's name, ID, age, and grade.
v. Records are commonly used in databases, where each record corresponds to a row in a table.
vi. They provide a structured way to organize and store data, allowing efficient retrieval and manipulation.

II. File

i. A file is a collection of records that are logically related to each other.
ii. It represents a cohesive unit of data stored on a storage medium, such as a hard disk or memory.
iii. Files are used to organize and manage large amounts of data.
iv. In the context of databases, a file can be thought of as a table that contains multiple records.
v. Each record in the file represents a row in the table, and the fields within the records correspond to the table columns.
vi. Files provide a way to group and organize related data records together.

III. Field

i. A field is the smallest unit of data within a record.
ii. It represents a single attribute or characteristic of the entity being described by the record.
iii. Each field holds a specific type of data, such as a number, text, date, or Boolean value.
iv. Going back to the student record example, the fields could include name, ID, age, and grade.
v. Fields are used to store and represent individual pieces of data within a record.
vi. They provide the granularity to access and manipulate specific data elements within a record.

IV. Key field

i. In data structures and databases, a key field, also known as a key or a key attribute, is a field within a record that uniquely identifies that record.
ii. It is used to establish a unique identifier for each record in a file or a database table.
iii. The key field is essential for efficient retrieval, modification, and organization of data.
iv. Key fields are crucial for maintaining data integrity, enabling efficient data retrieval, and establishing relationships between data elements in data structures and databases.

2.1.3 Data and Related Terms

In data structures and database systems, the following terms are used to describe different aspects of data organization:

i. Data : Data refers to the raw facts, figures, and symbols that represent information. It can be stored, processed, and manipulated to generate meaningful insights or facilitate decision-making. Data can exist in various forms, including numbers, text, images, audio, or video.

ii. Atomic Data : Atomic data, also known as elementary data, represents the smallest unit of data that cannot be further divided or broken down into simpler components. It refers to individual data items that cannot be decomposed into more basic data elements. For example, a single integer or a character is considered atomic data.

iii. **Composite Data :** Composite data, also known as structured data, refers to data that is composed of multiple atomic or composite data elements. It represents a combination or aggregation of atomic data items to form a more complex data structure. Examples of composite data include arrays, structures, records, and objects.

iv. **Entity :** In the context of database systems and entity-relationship modeling, an entity represents a distinct and identifiable object, concept, or thing in the real world that is of interest to the system. It could be a person, place, thing, event, or any other entity that has attributes and can be uniquely identified. For example, in a student database, a student would be considered an entity.

v. **Entity Set :** An entity set refers to a collection or group of similar entities. It represents a logical grouping of entities based on their shared characteristics or attributes. For example, the entity set "Students" would contain all the individual student entities in a student database.

2.2 DATA STRUCTURE OPERATIONS

Board Exam Questions

Q. What are the different operations that can be perform structures ? **(Oct. 04, March 06, 2 Marks)**

Q. Explain all six operations performed on a data structure. **(Oct. 06, 09, 12, 15, 17, March 10, 12, 3 Marks)**

Data structure operations refer to the actions or manipulations performed on data structures to insert, delete, retrieve, or modify data. The specific operations available depend on the type of data structure being used. Here are some common data structure operations:

i. **Insertion :** Insertion involves adding new data elements into a data structure. The insertion operation may require specifying the position or index where the new element should be inserted. The data structure may need to be resized or reorganized to accommodate the new element.

ii. **Deletion :** Deletion involves removing data elements from a data structure. The deletion operation may require specifying the position or key value of the element to be deleted. The data structure may need to be reorganized or adjusted after deletion to maintain its integrity.

iii. **Search :** Searching is the process of finding a specific data element within a data structure. The search operation may involve comparing the search key with the stored values to locate the desired element. Different searching algorithms, such as linear search or binary search, may be used depending on the data structure.

iv. **Access :** Accessing data involves retrieving or modifying the value of a specific data element in a data structure. Access operations typically require specifying the position or index of the element to be accessed. The access operation may involve reading or writing the value of the element.

v. **Update :** Updating data involves modifying the value of an existing data element in a data structure. The update operation may require specifying the position or key value of the element to be updated. The data structure may need to be adjusted or reorganized after the update to reflect the changes.

vi. **Traversal :** Traversal refers to the process of visiting and processing each data element in a data structure. Traversal operations are commonly used to iterate through all the elements in a data structure in a specific order. Different traversal techniques, such as in-order, pre-order, or post-order traversal, may be used depending on the data structure.

vii. **Sorting :** Sorting is the process of arranging the elements in a data structure in a specific order, such as ascending or descending. Sorting operations reorder the elements based on a defined comparison criterion, making it easier to search and retrieve data in a specific order. Various sorting algorithms, such as bubble sort, insertion sort, or quicksort, can be applied depending on the data structure.

2.3 ALGORITHMIC NOTATIONS

Board Exam Question

Q. What is an algorithm ? Write an algorithm to find the roots of a quadratic equation $ax^2 + bx + c = 0$, where $a \neq 0$ **(Oct. 17, 4 Marks)**

i. **Definition :** An algorithm is a step-by-step procedure or set of instructions used to solve a specific problem or perform a particular task.

ii. In the context of data structures, algorithms are designed to efficiently manipulate and operate on data stored within those structures.

iii. They determine how data is organized, accessed, inserted, deleted, searched, sorted, and modified.

iv. Here are some key properties and characteristics of algorithms :

a. **Correctness :** An algorithm should produce the correct and expected output for any valid input. It should solve the problem or accomplish the task as intended.

b. **Well-defined :** An algorithm should have precise and unambiguous instructions. Each step should be clearly defined and understandable, leaving no room for ambiguity or confusion.

c. **Finiteness :** An algorithm must terminate after a finite number of steps. It should not enter an infinite loop or continue indefinitely.

d. **Efficiency :** An algorithm should be efficient in terms of time and space complexity. It should perform the task or solve the problem in a reasonable amount of time and use a manageable amount of memory.

e. **Input :** An algorithm should take zero or more inputs, representing the initial problem or task. These inputs are used to guide the algorithm's execution and influence its behavior.

f. **Output :** An algorithm should produce an output or result that is meaningful and relevant to the problem being solved. The output could be a value, a data structure, or some other form of information.

g. **Determinism :** An algorithm should be deterministic, meaning that for a given input, it will always produce the same output. The steps and operations within the algorithm should be predictable and consistent.

h. Modularity : An algorithm should be organized into modular components or subroutines. This allows for code reuse, easier understanding, and maintenance of the algorithm.

i. Understandability : An algorithm should be designed and written in a way that is understandable to humans. It should be readable, well-documented, and follow standard coding conventions.

j. Scalability : An algorithm should be scalable, meaning it can handle larger input sizes or growing datasets without a significant loss in performance.

v. These properties ensure that algorithms are reliable, efficient, and capable of solving problems or performing tasks in a structured and predictable manner. They are essential for designing and analyzing algorithms and evaluating their suitability for specific use cases.

2.3.1 Flow Chart

i. Flowchart is a graphical representation of an algorithm.

ii. Programmers often use it as a program-planning tool to solve a problem.

iii. It makes use of symbols which are connected among them to indicate the flow of information and processing.

Rules for Creating Flowchart

A flowchart is a graphical representation of an algorithm. It should follow some rules while creating a flowchart.

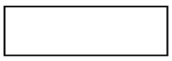
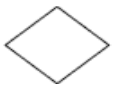
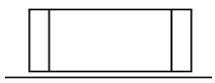
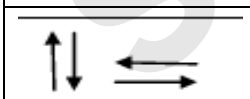
Rule 1 : Flowchart opening statement must be 'start' keyword.

Rule 2 : Flowchart ending statement must be 'end' keyword.

Rule 3 : All symbols in the flowchart must be connected with an arrow line.

Rule 4 : The decision symbol in the flowchart is associated with the arrow line.

Table 2.3.1 : Rules for creating flow chart

Symbol	Name	Function
	Process	Indicates any type of internal operation inside the processor or memory
	Input/Output	Used for any Input / Output (I/O) operation. Indicates that the computer is to obtain data or output results
	Decision	Used to ask a question that can be answered in a binary format (Yes/No, True/False)
	Connector	Allows the flowchart to be drawn without intersecting lines or without a reverse flow.
	Predefined process	Used to invoke a subroutine or an interrupt program
	Terminal	Indicates the starting or ending of the program, process, or interrupt program
	Flow lines	Shows direction flow

Example 1 : Algorithm to find the largest of three numbers.

i. Start

ii. Read the three numbers to be compared, as A, B and C.

iii. Check if A is greater than B.

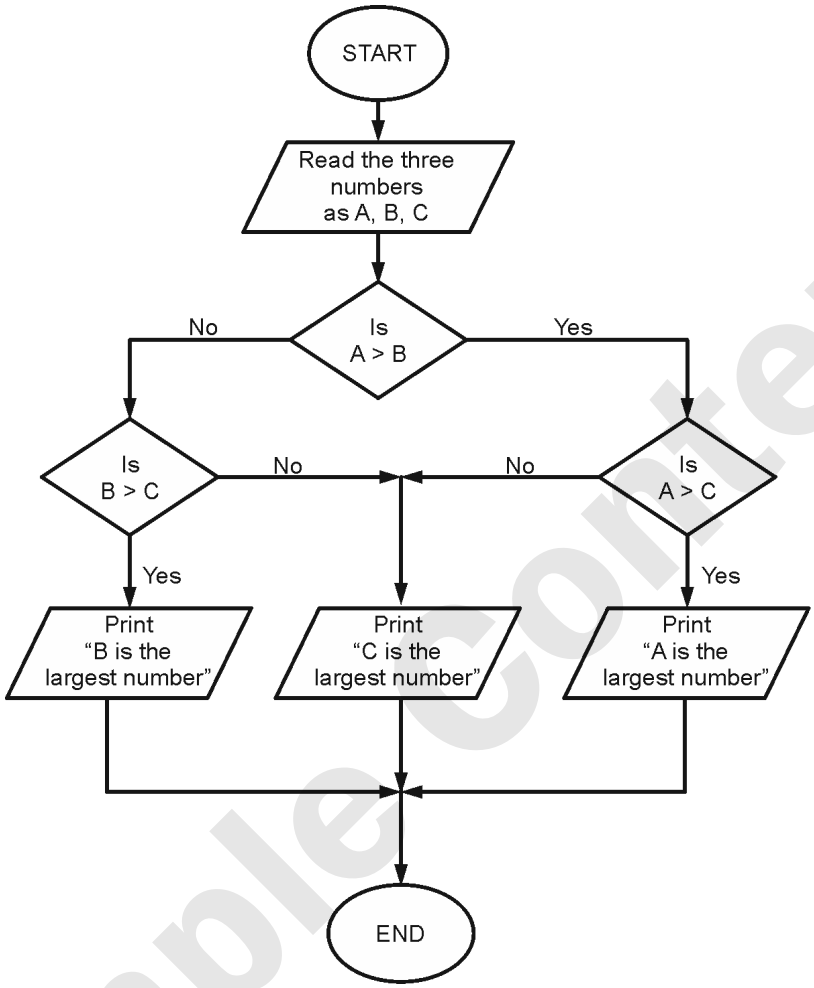
a. If true, then check if A is greater than C.

1. If true, print 'A' as the greatest number.
2. If false, print 'C' as the greatest number.

b. If false, then check if B is greater than C.
   1. If true, print 'B' as the greatest number.
   2. If false, print 'C' as the greatest number.

iv. End

Flowchart



(1B6) **Fig. 2.3.1**

Example 2 : Algorithm to find the roots of the quadratic equation.

i. Start

ii. Read the values of a, b, c

iii. Compute $d = b^2 - 4ac$

iv. If $d > 0$

a. calculate $root1 = \{-b + \sqrt{(b^2 - 4ac)}\}/2a$

b. calculate $root2 = \{-b - \sqrt{(b^2 - 4ac)}\}/2a$

else If $d = 0$

c. calculate $root1 = root2 = (-b/2a)$

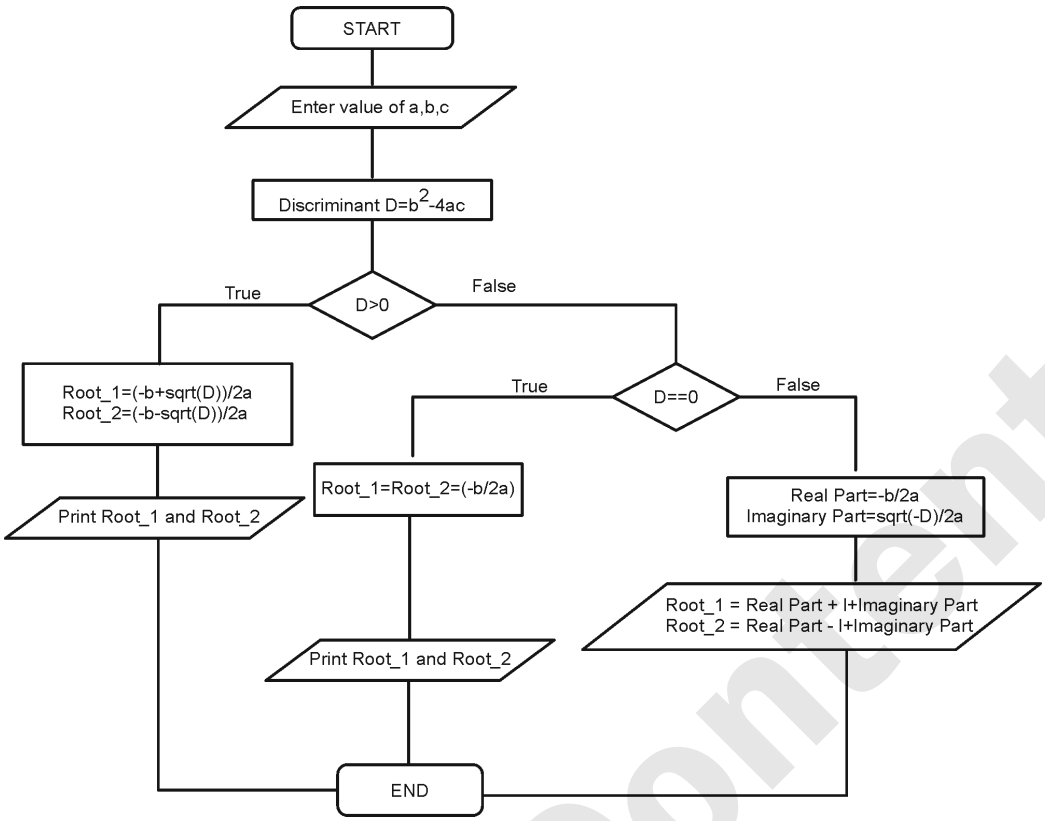
Else

d. calculate $root1 = \{-b + i\sqrt{-(b^2 - 4ac)}\}/2a$

e. calculate $root2 = \{-b + i\sqrt{-(b^2 - 4ac)}\}/2a$

v. print root1 and root2

vi. End



(1B7) **Fig. 2.3.2**

2.4 CONTROL STRUCTURES

Board Exam Questions

Q. Explain with flow charts the following control structures
i. Sequence logic ii. Selection logic
iii. Iteration logic.
(Oct. 03, March 09, 17, 3 Marks)

Q. Explain three control structures with flow chart in data structures **(Oct. 04, 05, 11, 3 Marks)**

Q. Explain the following control structure and their types used in data structure
i. Selection ii. loop
(Oct. 04, 05, 11, 3 Marks)

Q. Explain with flowchart the following control structure :
i. Sequence logic ii. Selection logic
iii. Iteration logic
(Q. 2A(a), April 23, 3 Marks)

i. **Control Structures** are just a way to specify flow of control in programs.

ii. Any algorithm or program can be more clear and understood if they use self-contained modules called as logic or control structures.

iii. It basically analyses and chooses in which direction a program flows based on certain parameters or conditions. There are three basic types of logic, or flow of control, known as :

1. Sequence logic, or sequential flow
2. Selection logic, or conditional flow
3. Iteration logic, or repetitive flow

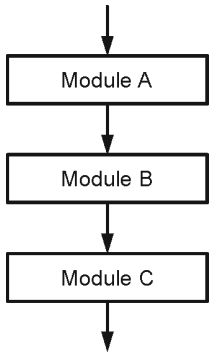
2.4.1 Sequence Logic (Sequential Flow)

i. Sequential logic as the name suggests follows a serial or sequential flow in which the flow depends on the series of instructions given to the computer.

ii. Unless new instructions are given, the modules are executed in the obvious sequence. The sequences may be given, by means of numbered steps explicitly.

iii. Also, implicitly follows the order in which modules are written.

iv. Most of the processing, even some complex problems, will generally follow this elementary flow pattern.



(1B8) **Fig. 2.4.1**

2.4.2 Selection Logic (Conditional Flow)

i. Selection Logic simply involves a number of conditions or parameters which decides one out of several written modules.

ii. The structures which use these type of logic are known as **Conditional Structures**.

iii. These structures can be of three types:

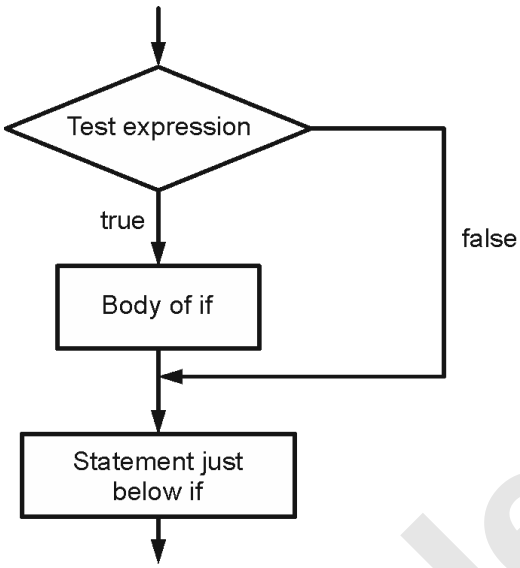
a. Single Alternative:

This structure has the form

If (condition) then :

[Module A]

[End of If structure]



(1B9) **Fig. 2.4.2**

Example

```
// C program to illustrate If statement
#include <stdio.h>
int main()
{
        int i = 10;
        if (i > 15) {
         printf("10 is greater than 15 \n");
        }
        printf("10 is smaller than 15 \n");
}
```

b. Double Alternative

This structure has the form

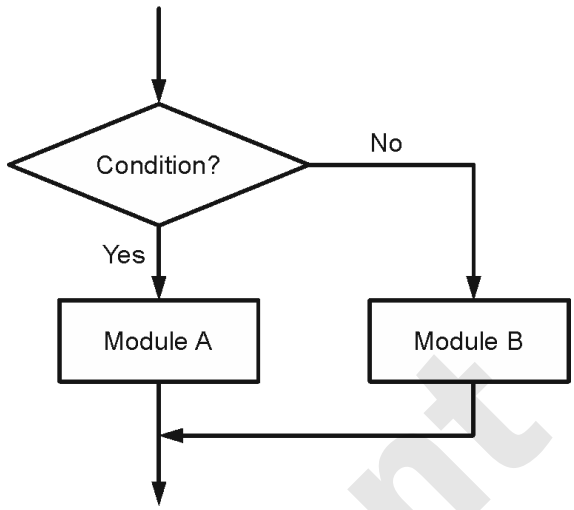
If (Condition), then :

[Module A]

Else :

[Module B]

[End if structure]



(1B10) **Fig. 2.4.3**

```
// C program to illustrate nested-if statement
#include <stdio.h>
int main()
{
        int i = 20;
        // Check if i is 10
        if (i == 10)
         printf("i is 10");
        // Since i is not 10
        // Check if i is 15
        else if (i == 15)
         printf("i is 15");
        // Since i is not 15
        // Check if i is 20
        else if (i == 20)
         printf("i is 20");
        // If none of the above conditions is true
        // Then execute the else statement
        else
         printf("i is not present");

        return 0;
}
```

c. Multiple Alternatives

This structure has the form:

If (condition A), then:

[Module A]

Else if (condition B), then:

[Module B]

..

..

Else if (condition N), then:

[Module N]

[End If structure]

AVAILABLE NOTES FOR STD. XI & XII:

SCIENCE

Perfect Series:

For students who want to excel in board exams and simultaneously study for entrance exams.

- Physics Vol. I
- Physics Vol. II
- Chemistry Vol. I
- Chemistry Vol. II
- Mathematics & Statistics Part - I
- Mathematics & Statistics Part - II
- Biology Vol. I
- Biology Vol. II

Precise Series:

For students who want to excel in board exams.

- Physics
- Chemistry
- Biology

Languages:

- English Yuvakbharati
- Hindi Yuvakbharati
- Marathi Yuvakbharati

COMMERCE (ENG. & MAR. MED.)

Smart Notes & Precise Notes:

- Book-Keeping and Accountancy
- Book Keeping and Accountancy (Practice)
- Book-Keeping & Accountancy Solutions To Textbook Problems
- Economics
- Organisation of Commerce and Management
- Secretarial Practice
- Mathematics and Statistics - I
- Mathematics and Statistics - II

ARTS (ENG. & MAR. MED.)

- History
- Geography
- Political Science
- Psychology
- Sociology

Target Publications Pvt. Ltd.
B2, 9th Floor, Ashar, Road No. 16/Z,
Wagle Industrial Estate,
Thane (W) - 400604,
Phone : 8879 9397 14 / 15
Email : mail@targetpublications.org

Tech-Neo Publications
Dugane Ind. Area Survey No. 28/25
Dhayari, Near Pari Company,
Pune, Maharashtra 411041
Email : info@techneobooks.in
Phone : +91 9850429188

Also available on - Flipkart amazon

ISBN : 978-93-5583-404-1 TEID : 3171

Visit Our Website