Name:	Mr. Nayan Prabhakar Kasturi
Class:	M. Sc. Bioinformatics (Part I)
Roll Number:	110
Course:	M. Sc. Bioinformatics
Department:	Department of Bioinformatics
Paper:	Mandatory Paper I
Paper Name and Code:	Introduction to Programming Languages and Database (GNKPSBI2P501)
Academic Year:	2023-24

SGCP's
Guru Nanak Khalsa College
of Arts, Science & Commerce (Autonomous)

DEPARTMENT OF BIOINFORMATICS

CERTIFICATE

This is to certify that Mr. Nayan Prabhakar Kasturi (Roll No: 110) of M. Sc. Bioinformatics (Part I) has satisfactorily completed the practical Semester I course prescribed by the University of Mumbai during the academic year 2023-2024.

Teacher-in-Charge (Signature)	Head of Department (Signature)	External Examiner (Signature)

INDEX

DATE: 26/08/2023

PRACTICAL: 1

SQL: Structured Query Language

AIM:

To collect information and store it as relational database using SQL commands.

INTRODUCTION:

SQL (Structured Query Language) is a programming language used to communicate with and manipulate databases. It was developed in the 1970s by IBM researchers Raymond Boyce and Donald Chamberlin. The programming language, known then as SEQUEL, was created following Edgar Frank Codd’s paper, “A Relational Model of Data for Large Shared Data Banks,” in 1970. SQL is an international standard (ISO) and is used with all kinds of relational databases like MySQL Database, Oracle, MS SQL Server, Sybase, etc.

SQL is a standard language for accessing and manipulating databases. It lets you access and manipulate databases. SQL became a standard of the American National Standards Institute (ANSI) in 1986, and of the International Organization for Standardization (ISO) in 1987. SQL can execute queries against a database, retrieve data from a database, and insert records in a database. However, to be compliant with the ANSI standard, they all support at least the major commands (such as SELECT, UPDATE, DELETE, INSERT, WHERE) in a similar manner. Most of the SQL database programs also have their own proprietary extensions in addition to the SQL standard.

Being able to wrangle and extract data from databases using SQL is an essential skill within the data industry and in increasing demand. Much of the world's raw data—from electronic medical records to customer transaction histories—lives in organized collections of tables called relational databases. SQL is extremely accessible across various platforms and generally user-friendly.

In conclusion, SQL is a standardized query language used to retrieve information from databases. It is a programming language used for managing and manipulating relational databases. SQL is an essential skill within the data industry and in increasing demand. It is user-friendly and accessible across various platforms.

Datatypes in SQL:

1. **String:**
    a. **CHAR**
    CHAR stands for Character. It stores values containing any characters in a character set. CHAR is defined to be of specific length.
    b. **VARCHAR**
    VARCHAR stands for varying character. It stores string values containing variable character in a set of character but of definable length. There are two

types of VARCHAR – VARCHAR (less character length) & VARCHAR2 (more character length)

   c. **BLOB**
      BLOB stands for Binary Large Object. It stores string values in hexadecimal format and is defined to be of a variable length.

2. **Numeric**:
   a. **NUMERIC/NUMBER**
      NUMERIC or NUMBER stores the exact number with a defined precision & scale. It stores the decimal values.
   b. **INT**
      INT stands for integer. It stores the exact number with a predefined precision and scaled to zero. That is, it stores the integer values.
3. **Temporal**:
   a. **TIMESTAMP**
      Stores the moment an event occurs, using definable fraction-of-second precision.
   b. **TIMESTAMP WITH LOCAL**
      Stores the moment an event occurs, using definable fraction-of-second precision, with reference to the local time zone.
   c. **TIMEZONE**
      Defines the time zone.
4. **BOOLEAN**
   BOOLEAN stores the True, False & Unknown values.

Advantages of SQL:

1. Efficient data retrieval: SQL allows for fast and efficient retrieval of specific data from large databases, making it easier for businesses to find the information they need quickly.
2. Data integrity: SQL ensures that data is accurate and consistent by using constraints, such as primary keys and foreign keys, to maintain data integrity.
3. Scalability: SQL databases can easily scale up or down to accommodate changing business needs, making it a flexible solution for businesses of all sizes.
4. Standardization: SQL is a standardized language for accessing and manipulating databases, making it easier for developers to work with different databases and for businesses to switch between different database systems.
5. Flexibility: SQL is a versatile programming language that can be used for a wide range of tasks, including data analysis, reporting, and data management.

Disadvantages of SQL:

1. Poor Interface: SQL can have a poor interface, making it difficult for users to interact with the database. This can lead to errors and mistakes in database management.
2. Cost Inefficient: SQL can be cost-inefficient, especially when dealing with large amounts of data. The cost of hardware, software, and maintenance can be high.

3. Partial Control: SQL databases provide partial control over the database, which can be a disadvantage for some users. Users may not have full control over the database, which can limit their ability to customize it to their needs.
4. Security Risks: SQL databases can be vulnerable to security risks, such as SQL injection attacks, which can compromise the integrity and confidentiality of data.
5. Complexity: SQL can be complex and difficult to learn for those who are new to programming, which can lead to errors and mistakes in database management.

SQL Commands

Following are the 4 types of SQL Commands –

1. **Data Definition Language (DDL)** – Adding or deleting the table
2. **Data Manipulation Language (DML)** – changing or inserting or replacing the data / table values
3. **Data Control Language (DCL)** – giving a set of controls / rights / access to the group of users for a particular table
4. **Data Query Language (DQL)** – stored data can be retrieved using a query and can be displayed / shown to the users.

COMMAND	DESCRIPTION	SYNTAX

CREATE	It creates a database and its tables.	CREATE DATABASE DATABASE_NAME;
		
		CREATE TABLE TABLE_NAME (COLUMN1 DATATYPE, COLUMN2 DATATYPE, …);

ALTER	It alters the structure of the existing database	ALTER TABLE TABLE_NAME ADD COLUMN1 DATATYPE;

DROP	It deletes tables and database	DROP DATABASE DATABASE_NAME;
		
		DROP TABLE TABLE_NAME;

TRUNCATE	It removes all records from a table	TRUNCATE TABLE TABLE_NAME;

SELECT	It is used to select & view data from a database	SELECT COLUMN1, COLUMN2, ...
		FROM TABLE_NAME;

SELECT DISTINCT	It is used to return only distinct (different) values	SELECT DISTINT
		COLUMN1, COLUMN2, ...
		FROM TABLE_NAME

WHERE	It is used to filter records.	SELECT COLUMN1, COLUMN2, ...
		FROM TABLE_NAME
		WHERE CONDITION;

ORDER BY	It is used to sort the result-set in ascending or descending order.	SELECT COLUMN1, COLUMN2, ...
		FROM TABLE_NAME
		WHERE CONDITION;

INSERT	It is used to insert new records in a table	INSERT TABLE_NAME (COLUMNS)
		VALUES (VALUE1, VALUE2, VALUE3);

UPDATE	It is used to modify the existing records in a table.	UPDATE TABLE_NAME
		SET COLUMN1=VALUE1,
		COLUMN2=VALUE2,
		WHERE CONDITION;

DELETE	It is used to delete existing records in a table.	DELETE FROM TABLE_NAME
		WHERE CONDITION;

COUNT	It returns the number of rows that matches a specified criterion	SELECT COUNT (COLUMN_NAME)
		FROM TABLE_NAME
		WHERE CONDITION;

SUM	It returns the total sum of a numeric column.	SELECT SUM (COLUMN_NAME)
		FROM TABLE_NAME
		WHERE CONDITION;

AVG	It returns the total sum of a numeric column.	SELECT AVG (COLUMN_NAME)
		FROM TABLE_NAME
		WHERE CONDITION;

GROUP BY	It is used with aggregate functions [COUNT(), MAX(), MIN(), SUM(), AVG()] to group the result-set by one or more columns.	SELECT COLUMN_NAME(S)
		FROM TABLE_NAME
		WHERE CONDITION
		GROUP BY COLUMN_NAME;

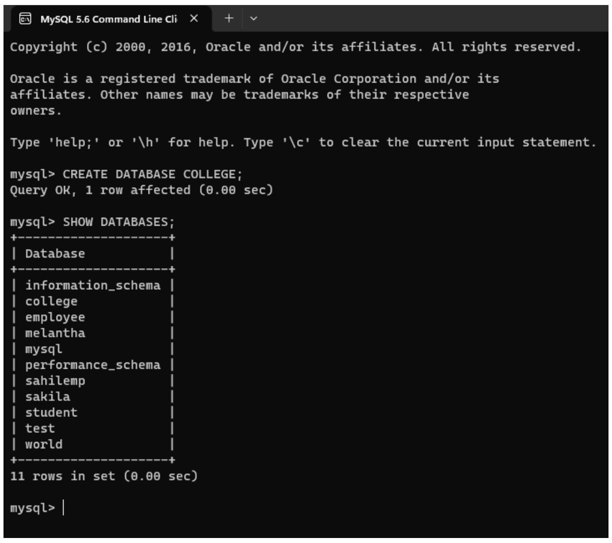
MIN and MAX	It returns the smallest value of the selected column.	SELECT MIN (COLUMN_NAME)
		FROM TABLE_NAME
		WHERE CONDITION;
		
		SELECT MAX (COLUMN_NAME)
		FROM TABLE_NAME
		WHERE CONDITION;

GRANT	It assigns the users' privileges to access the database	GRANT PRIVILEGE_NAME ON TABLE_NAME TO USER1, USER2, ...
REVOKE	It takes away the users' privileges to the database	REVOKE PRIVILEGE_NAME ON TABLE_NAME TO USER1, USER2, ...
COMMIT	It commits a transaction	COMMIT;

ROLLBACK	It rollbacks a transaction in case of any error occurs	ROLLBACK;
		ROLLBACK TO SAVEPOINT_NAME;

SAVEPOINT	It creates restore point to rollback the transaction to a particular point	SAVEPOINT SAVEPOINT_NAME;

QUESTIONS:



```
Copyright (c) 2000, 2016, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> CREATE DATABASE COLLEGE;
Query OK, 1 row affected (0.00 sec)

mysql> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| college            |
| employee           |
| melantha           |
| mysql              |
| performance_schema |
| sahilemp           |
| sakila             |
| student            |
| test               |
| world              |
+--------------------+
11 rows in set (0.00 sec)

mysql> |
```

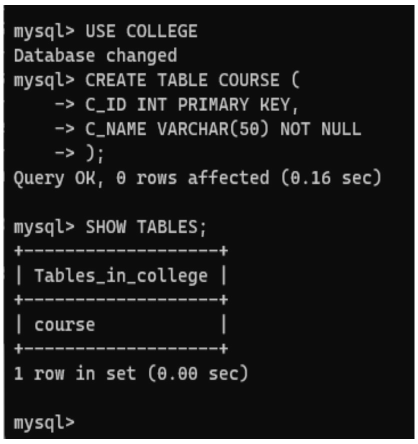
Figure 1: CREATE DATABASE COLLEGE AND SHOW THE LIST OF DATABASES

1. Create course table and attributes are C_ID and C_NAME.

Code:

```
CREATE TABLE COURSE(
C_ID INT PRIMARY KEY,
C_NAME VARCHAR(50) NOT NULL
);
```

Output:



```
mysql> USE COLLEGE
Database changed
mysql> CREATE TABLE COURSE (
    -> C_ID INT PRIMARY KEY,
    -> C_NAME VARCHAR(50) NOT NULL
    -> );
Query OK, 0 rows affected (0.16 sec)

mysql> SHOW TABLES;
+-------------------+
| Tables_in_college |
+-------------------+
| course            |
+-------------------+
1 row in set (0.00 sec)

mysql>
```

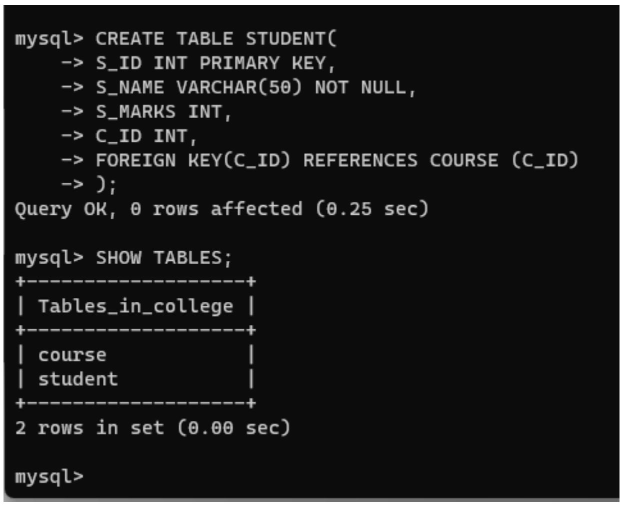
Figure 2: CREATE TABLE COURSE AND SHOW THE LIST OF TABLES IN DB COLLEGE

2. **Create student table and attributes are S_ID, S_NAME, S_MARKS AND C_ID.**

Code:

```
CREATE TABLE COURSE(
S_ID INT PRIMARY KEY,
S_NAME VARCHAR(50) NOT NULL
S_MARKS INT,
C_ID INT,
FOREIGN KEYREFERENCES COURSE (C_ID)
);
```

Output:



```
mysql> CREATE TABLE STUDENT(
    -> S_ID INT PRIMARY KEY,
    -> S_NAME VARCHAR(50) NOT NULL,
    -> S_MARKS INT,
    -> C_ID INT,
    -> FOREIGN KEY(C_ID) REFERENCES COURSE (C_ID)
    -> );
Query OK, 0 rows affected (0.25 sec)

mysql> SHOW TABLES;
+-------------------+
| Tables_in_college |
+-------------------+
| course            |
| student           |
+-------------------+
2 rows in set (0.00 sec)

mysql>
```

Figure 3: CREATE TABLE STUDENT AND SHOW THE LIST OF TABLES IN DB COLLEGE

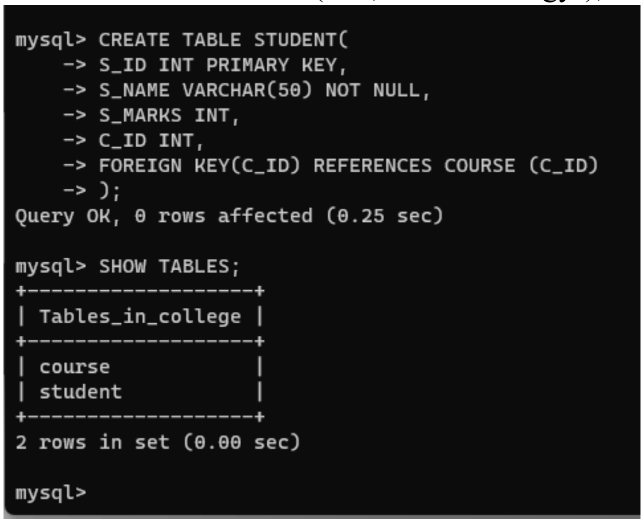
3. **Insert values into table student and course.**

Code:

Inserting Course Values:

```
INSERT INTO COURSE VALUES(101, 'Biology');
INSERT INTO COURSE VALUES(102, 'Bioinformatics');
INSERT INTO COURSE VALUES(103, 'Microbiology');
```

Output:



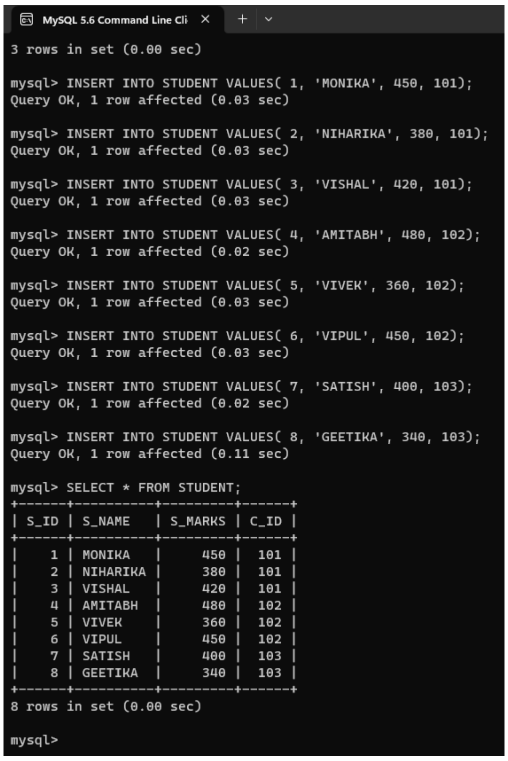
```
mysql> CREATE TABLE STUDENT(
    -> S_ID INT PRIMARY KEY,
    -> S_NAME VARCHAR(50) NOT NULL,
    -> S_MARKS INT,
    -> C_ID INT,
    -> FOREIGN KEY(C_ID) REFERENCES COURSE (C_ID)
    -> );
Query OK, 0 rows affected (0.25 sec)

mysql> SHOW TABLES;
+-------------------+
| Tables_in_college |
+-------------------+
| course            |
| student           |
+-------------------+
2 rows in set (0.00 sec)

mysql>
```

Figure 4: ENTERING & SHOWING DATA IN TABLE COURSE

Code:
Inserting
Student Values:

```
INSERT INTO STUDENT VALUES (1, ‘Monika’, 450, 101);
INSERT INTO STUDENT VALUES (2, ‘Niharika’, 380, 101);
INSERT INTO STUDENT VALUES (3, ‘Vishal’, 420, 101);
INSERT INTO STUDENT VALUES (4, ‘Amitabh’, 480, 102);
INSERT INTO STUDENT VALUES (5, ‘Vivek’, 360, 102);
INSERT INTO STUDENT VALUES (6, ‘Vipul’, 450, 102);
INSERT INTO STUDENT VALUES (7, ‘Satish’, 400, 103);
INSERT INTO STUDENT VALUES (8, ‘Geetika’, 340, 103);
```

Output:



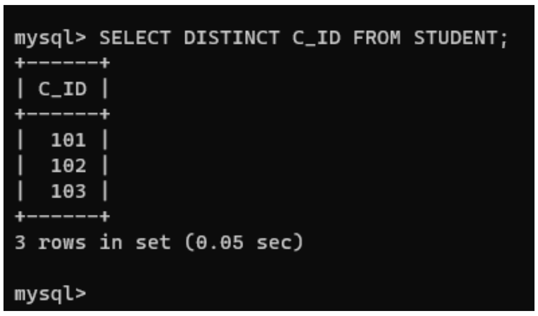
```
3 rows in set (0.00 sec)

mysql> INSERT INTO STUDENT VALUES( 1, 'MONIKA', 450, 101);
Query OK, 1 row affected (0.03 sec)

mysql> INSERT INTO STUDENT VALUES( 2, 'NIHARIKA', 380, 101);
Query OK, 1 row affected (0.03 sec)

mysql> INSERT INTO STUDENT VALUES( 3, 'VISHAL', 420, 101);
Query OK, 1 row affected (0.03 sec)

mysql> INSERT INTO STUDENT VALUES( 4, 'AMITABH', 480, 102);
Query OK, 1 row affected (0.02 sec)

mysql> INSERT INTO STUDENT VALUES( 5, 'VIVEK', 360, 102);
Query OK, 1 row affected (0.03 sec)

mysql> INSERT INTO STUDENT VALUES( 6, 'VIPUL', 450, 102);
Query OK, 1 row affected (0.03 sec)

mysql> INSERT INTO STUDENT VALUES( 7, 'SATISH', 400, 103);
Query OK, 1 row affected (0.02 sec)

mysql> INSERT INTO STUDENT VALUES( 8, 'GEETIKA', 340, 103);
Query OK, 1 row affected (0.11 sec)

mysql> SELECT * FROM STUDENT;
+------+----------+---------+------+
| S_ID | S_NAME   | S_MARKS | C_ID |
+------+----------+---------+------+
|    1 | MONIKA   |     450 |  101 |
|    2 | NIHARIKA |     380 |  101 |
|    3 | VISHAL   |     420 |  101 |
|    4 | AMITABH  |     480 |  102 |
|    5 | VIVEK    |     360 |  102 |
|    6 | VIPUL    |     450 |  102 |
|    7 | SATISH   |     400 |  103 |
|    8 | GEETIKA  |     340 |  103 |
+------+----------+---------+------+
8 rows in set (0.00 sec)

mysql>
```

Figure 5: ENTERING & SHOWING DATA IN TABLE STUDENT

4. Write an SQL query to fetch unique values of Course Id from the Student table.

Code: SELECT DISTINCT C_ID FROM STUDENT;

Output:



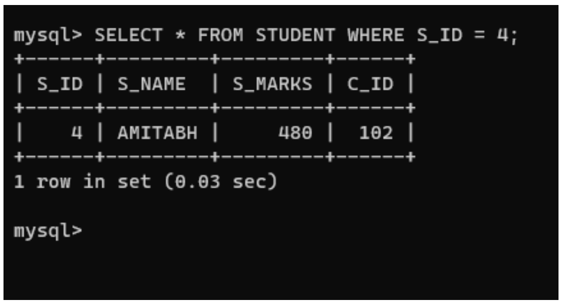
```
mysql> SELECT DISTINCT C_ID FROM STUDENT;
+------+
| C_ID |
+------+
|  101 |
|  102 |
|  103 |
+------+
3 rows in set (0.05 sec)

mysql>
```

Figure 6: FETCHING UNIQUE VALUES OF COURSE ID FROM THE STUDENT TABLE

5. Display the record of student whose id is 4.

Code: SELECT * FROM STUDENT WHERE S_ID = 4;

Output:



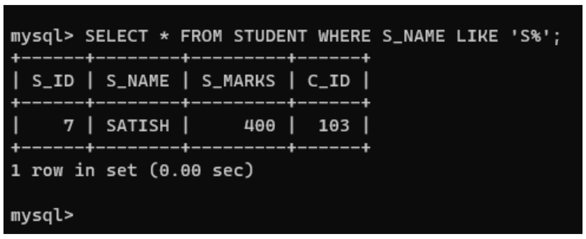
```
mysql> SELECT * FROM STUDENT WHERE S_ID = 4;
+------+---------+---------+------+
| S_ID | S_NAME  | S_MARKS | C_ID |
+------+---------+---------+------+
|    4 | AMITABH |     480 |  102 |
+------+---------+---------+------+
1 row in set (0.03 sec)

mysql>
```

Figure 7: DISPLAYING THE RECORD OF STUDENT WHOSE S_ID IS 4

6. Display the record of student whose name start with letter ‘S’.

Code: SELECT * FROM STUDENT WHERE S_NAME LIKE ‘S%’;

Output:



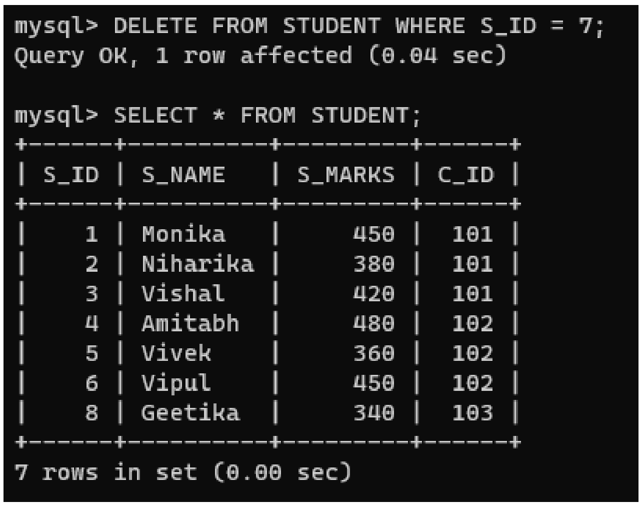
```
mysql> SELECT * FROM STUDENT WHERE S_NAME LIKE 'S%';
+------+--------+---------+------+
| S_ID | S_NAME | S_MARKS | C_ID |
+------+--------+---------+------+
|    7 | SATISH |     400 |  103 |
+------+--------+---------+------+
1 row in set (0.00 sec)

mysql>
```

Figure 8: DISPLAY THE RECORD OF STUDENT WHOSE NAME STARTS WITH ‘S’

7. Remove the record from student table whose student id is 7.

Code: DELETE * FROM STUDENT WHERE S_ID = 7;

Output:



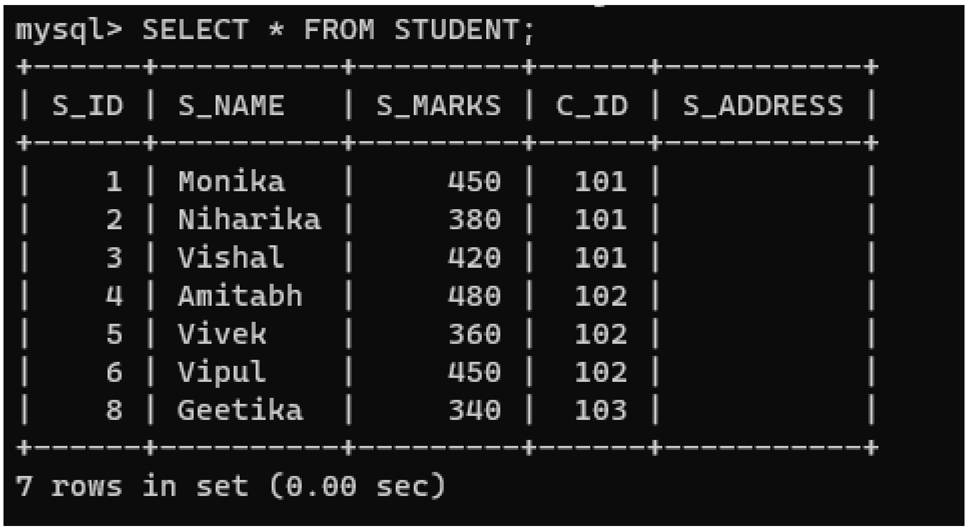
```
mysql> DELETE FROM STUDENT WHERE S_ID = 7;
Query OK, 1 row affected (0.04 sec)

mysql> SELECT * FROM STUDENT;
+------+----------+---------+------+
| S_ID | S_NAME   | S_MARKS | C_ID |
+------+----------+---------+------+
|    1 | Monika   |     450 |  101 |
|    2 | Niharika |     380 |  101 |
|    3 | Vishal   |     420 |  101 |
|    4 | Amitabh  |     480 |  102 |
|    5 | Vivek    |     360 |  102 |
|    6 | Vipul    |     450 |  102 |
|    8 | Geetika  |     340 |  103 |
+------+----------+---------+------+
7 rows in set (0.00 sec)
```

Figure 9: DELETING THE RECORD OF STUDENT WHERE S_ID = 7

8. **Add new column to STUDENT table as S_ADDRESS.**

Code:
```
ALTER TABLE STUDENT
ADD S_ADDRESS VARCHAR2(100) NOT NULL;
```

Output:



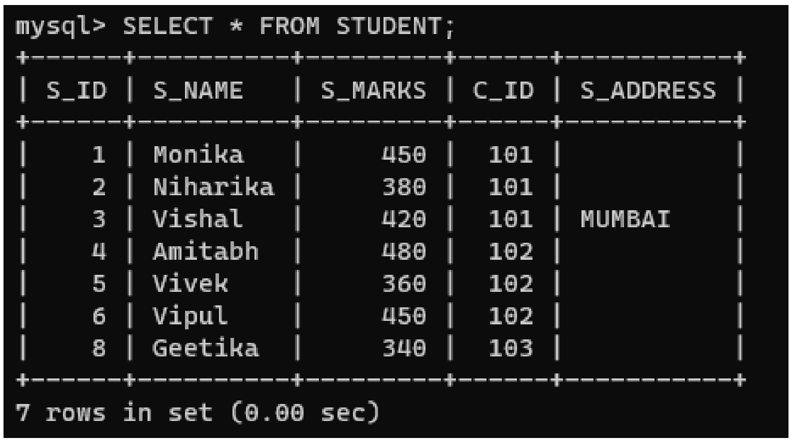
```
mysql> SELECT * FROM STUDENT;
+------+----------+---------+------+-----------+
| S_ID | S_NAME   | S_MARKS | C_ID | S_ADDRESS |
+------+----------+---------+------+-----------+
|    1 | Monika   |     450 |  101 |           |
|    2 | Niharika |     380 |  101 |           |
|    3 | Vishal   |     420 |  101 |           |
|    4 | Amitabh  |     480 |  102 |           |
|    5 | Vivek    |     360 |  102 |           |
|    6 | Vipul    |     450 |  102 |           |
|    8 | Geetika  |     340 |  103 |           |
+------+----------+---------+------+-----------+
7 rows in set (0.00 sec)
```

Figure 10: ADDING NEW COLUMN S_ADDRESS TO STUDENT TABLE

9. **Add values to S_ADDRESS column.**

Code:
```
UPDATE STUDENT
SET S_ADDRESS  = ‘MUMBAI’
WHERE S_ID = 3;
```

Output:



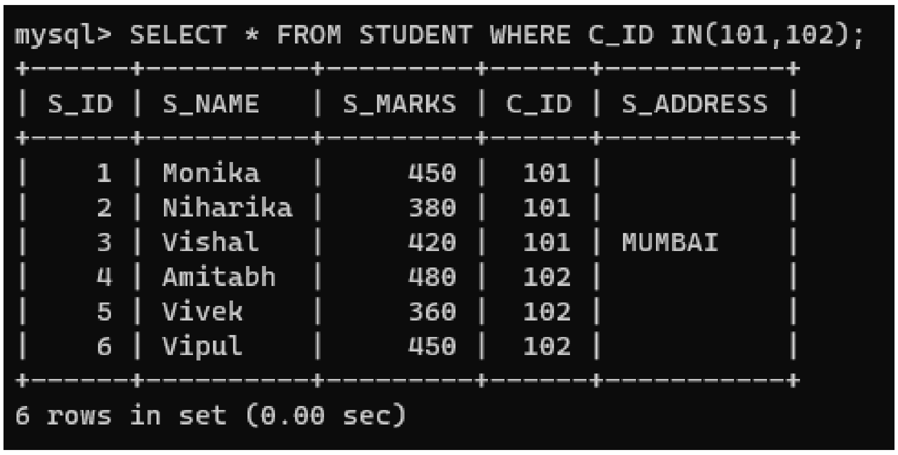
```
mysql> SELECT * FROM STUDENT;
+------+----------+---------+------+-----------+
| S_ID | S_NAME   | S_MARKS | C_ID | S_ADDRESS |
+------+----------+---------+------+-----------+
|    1 | Monika   |     450 |  101 |           |
|    2 | Niharika |     380 |  101 |           |
|    3 | Vishal   |     420 |  101 | MUMBAI    |
|    4 | Amitabh  |     480 |  102 |           |
|    5 | Vivek    |     360 |  102 |           |
|    6 | Vipul    |     450 |  102 |           |
|    8 | Geetika  |     340 |  103 |           |
+------+----------+---------+------+-----------+
7 rows in set (0.00 sec)
```

Figure 11: ADDING VALUES TO S_ADDRESS COLUMN

10. Display the record of student whose course id is 101 and 102 using IN clause.

Code: SELECT * FROM STUDENT WHERE C_ID IN(101, 102);

Output:



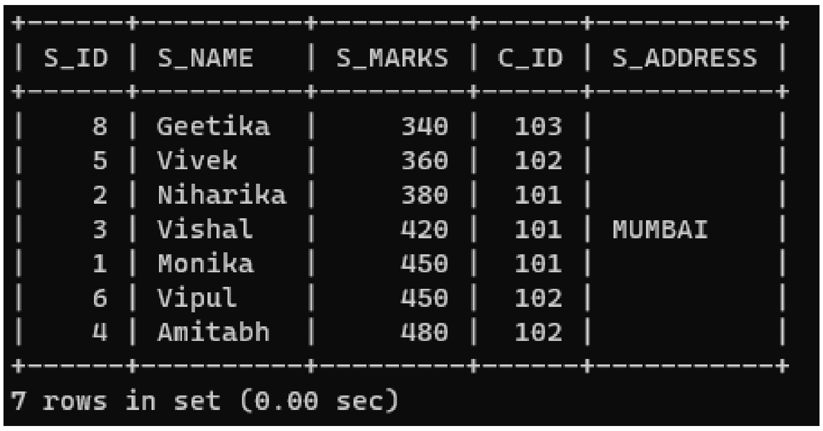
```
mysql> SELECT * FROM STUDENT WHERE C_ID IN(101,102);
+------+----------+---------+------+-----------+
| S_ID | S_NAME   | S_MARKS | C_ID | S_ADDRESS |
+------+----------+---------+------+-----------+
|    1 | Monika   |     450 |  101 |           |
|    2 | Niharika |     380 |  101 |           |
|    3 | Vishal   |     420 |  101 | MUMBAI    |
|    4 | Amitabh  |     480 |  102 |           |
|    5 | Vivek    |     360 |  102 |           |
|    6 | Vipul    |     450 |  102 |           |
+------+----------+---------+------+-----------+
6 rows in set (0.00 sec)
```

Figure 12: DISPLAYING RECORDS OF STUDENTS WITH COURSE ID 101 & 102

11. Display the record of student by sorting in ascending order of S_MARKS column.

Code: SELECT * FROM STUDENT ORDER BY S_MARKS ASC;

Output:



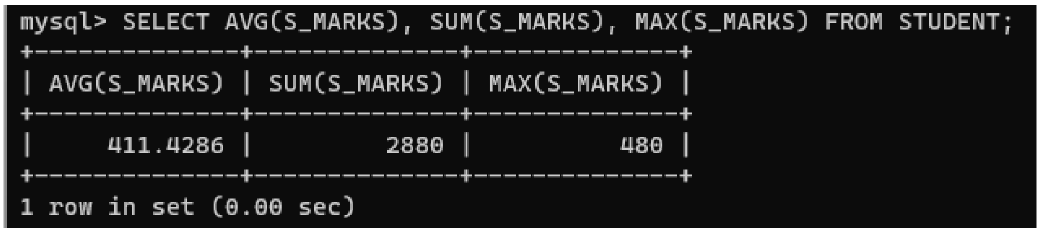
```
+------+----------+---------+------+-----------+
| S_ID | S_NAME   | S_MARKS | C_ID | S_ADDRESS |
+------+----------+---------+------+-----------+
|    8 | Geetika  |     340 |  103 |           |
|    5 | Vivek    |     360 |  102 |           |
|    2 | Niharika |     380 |  101 |           |
|    3 | Vishal   |     420 |  101 | MUMBAI    |
|    1 | Monika   |     450 |  101 |           |
|    6 | Vipul    |     450 |  102 |           |
|    4 | Amitabh  |     480 |  102 |           |
+------+----------+---------+------+-----------+
7 rows in set (0.00 sec)
```

Figure 13: ADDING NEW COLUMN S_ADDRESS TO STUDENT TABLE

12. Calculate Average, Sum and Max marks of student.

Code:
```
SELECT AVG(S_MARKS), SUM(S_MARKS), MAX(S_MARKS)
FROM STUDENT;
```

Output:



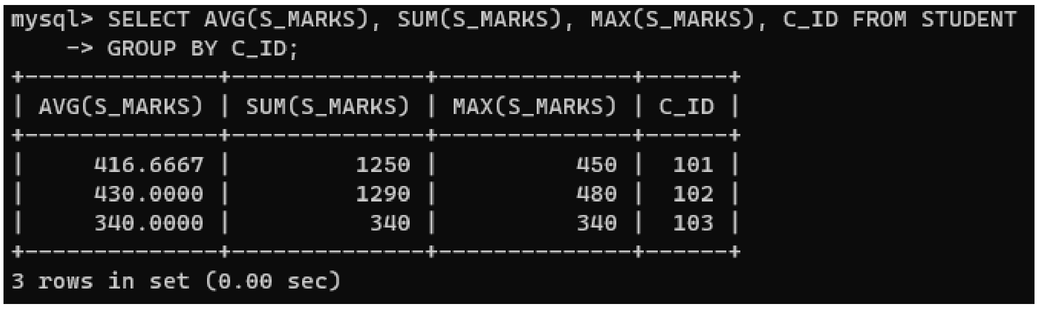
```
mysql> SELECT AVG(S_MARKS), SUM(S_MARKS), MAX(S_MARKS) FROM STUDENT;
+--------------+--------------+--------------+
| AVG(S_MARKS) | SUM(S_MARKS) | MAX(S_MARKS) |
+--------------+--------------+--------------+
|     411.4286 |         2880 |          480 |
+--------------+--------------+--------------+
1 row in set (0.00 sec)
```

Figure 14: DISPLAYING AVERAGE, SUMATION AND MAXIMUM MARKS OF THE STUDENTS

13. Calculate Average, Sum and Max marks of student by C_ID.

Code:
```
SELECT AVG(S_MARKS), SUM(S_MARKS), MAX(S_MARKS)
FROM STUDENT
GROUP BY C_ID;
```

Output:



```
mysql> SELECT AVG(S_MARKS), SUM(S_MARKS), MAX(S_MARKS), C_ID FROM STUDENT
    -> GROUP BY C_ID;
+--------------+--------------+--------------+------+
| AVG(S_MARKS) | SUM(S_MARKS) | MAX(S_MARKS) | C_ID |
+--------------+--------------+--------------+------+
|     416.6667 |         1250 |          450 |  101 |
|     430.0000 |         1290 |          480 |  102 |
|     340.0000 |          340 |          340 |  103 |
+--------------+--------------+--------------+------+
3 rows in set (0.00 sec)
```

Figure 15: DISPLAYING AVG, SUM & MAX MARKS OF STUDENTS BY GROUPING WITH RESPECT TO C_ID

RESULTS:

1. Using MySQL, database as well as the tables were created. Attributes were assigned to the tables and relevant data is inserted into the tables.
2. Various commands were used to view different data in different ways. Also, the commands were used to modify and delete data from the table.
3. Overall, the database has been studied in detailed using the MySQL software as the Database Management System software.

CONCLUSION:

SQL, a versatile programming language for interacting with databases, has been studied and explored using various commands.

<u>REFERENCES:</u>

1. *SQL Tutorial*. (n.d.). https://www.w3schools.com/sql/
2. Groff, J., Weinberg, P., & Oppel, A. (2009, August 12). *SQL The Complete Reference, 3rd Edition*. Mcgraw-hill.
3. *Learn MySQL: A Comprehensive Guide to Using and Managing Databases*. (n.d.). https://www.w3schools.in/mysql/
4. *SQL Tutorial, Tutorials SQL*. (n.d.). https://beginner-sql-tutorial.com/sql.htm

Date: 13/10/2023

PRACTICAL: 2
C++: Basic I/O Program

AIM:
To create basic input-output programs using C++ programming language.

INTRODUCTION:
C++ (an extension of C) was developed by Bjarne Stroustrup in early 1980s at Bell Laboratories, with the goal to make writing good programs easier and more pleasant for the individual programmer by adding OOP features to C without significantly changing the C component. C++ is an important language in computer science and software development, and it is used in a wide range of applications, including operating systems, video games, and financial trading systems. Its combination of low-level and high-level features make it a powerful language for programming complex systems.

Advantages of C++:
1. **Portability:** C++ offers the feature of portability or platform independence which allows the user to run the same program on different operating systems or interfaces at ease.
2. **Mid-level programming language:** C++ is a middle-level programming language that combines the features of high-level and low-level programming languages, making it a powerful language for programming complex systems.
3. **Object-Oriented:** C++ is an object-oriented programming language that includes concepts like classes, inheritance, polymorphism, data abstraction, and encapsulation that allow code reusability and makes a program even more reliable.
4. **Multi-paradigm programming language:** C++ supports multiple programming paradigms, including procedural, object-oriented, and generic programming.
5. **Memory management:** Pointers are used for dynamic memory allocation and deallocation, which allows for efficient use of memory and reduces the length of the program and its execution time.

Disadvantages of C++:
1. **Unsafe:** Pointers can be dangerous when not used properly, as they allow direct manipulation of memory addresses and can lead to memory corruption and other errors.
2. **No garbage collection:** C++ does not have automatic garbage collection, so it is the programmer's responsibility to manage memory allocation and deallocation using pointers.
3. **Complex:** Pointers can be complex to understand and use, especially for programmers who are new to the C++ language, which can increase the risk of errors or mistakes in the code.
4. **No custom operators:** C++ does not allow for custom operators, which can limit the flexibility and expressiveness of the language.
5. **No built-in threads:** C++ does not have built-in support for threads, which can make it more difficult to write concurrent programs.

6. **Lack of algebraic data types:** C++ does not have built-in support for algebraic data types, which can make it more difficult to write programs that require complex data structures.

Applications of C++:

1. **Operating systems and browsers:** C++ is used to develop operating systems like Windows, Linux, and macOS as well as web browsers like Google Chrome, Mozilla Firefox, and Microsoft Edge.
2. **Games and real-time simulation:** C++ is a popular language for game development and real-time simulation and interfacing with avionics in flight simulators, as it allows for efficient memory management and high performance.
3. **Financial trading systems:** C++ is used in finance for its low latency and high performance, which are critical for trading systems.
4. **Network and audio drivers:** C++ is used for network and sound drivers programming, as it allows for efficient memory management and high performance.

Header files:

A C++ header file contains declarations of functions, classes, and constructs, aiding code reuse and reducing complexity. Using '#include', these files separate declarations from implementations, promoting modularity. Standard library headers (e.g., <iostream>) and user-defined headers are essential for organized programming. For instance, #include <iostream>, allows access to input/output functions without writing their code directly, enhancing code manageability and fostering modularity in C++ development.

Functions:

Functions in C++ are modular units of code that perform specific tasks, promoting modularity and code reusability. They can be built-in, like main(), or user-defined. Functions encapsulate tasks, keeping the variable namespace clean, and can be repeatedly used.

Data types:

Data type	Meaning	Size (in bytes)
Primary Data types		
void	Valueless or Void	N/A
char	Stores a single character/letter/number, or ASCII values	1
bool	Stores true or false values	1
int	Stores whole numbers, without decimals	2 or 4
float	Stores fractional numbers, containing one or more decimals. Sufficient for storing 6-7 decimal digits	4
double	Stores fractional numbers, containing one or more decimals. Sufficient for storing 15 decimal digits	8
Derived Data types		
Function	A data type that defines a function	Varies

Array	A data type that represents a collection of items stored at continuous memory locations	Varies
Pointer	A data type that stores the memory address of another variable	2 or 4
Reference	A data type that provides an alternative name for an existing variable	Same as the referenced variable
User-defined Data types		
Class	A user-defined data type that defines a blueprint for creating objects	Varies
Structure	A user-defined data type that groups related data items of different data types	Varies
Union	A user-defined data type that allows storing different data types in the same memory location	Varies
Enumeration	A user-defined data type that assigns names to a set of related constants	4

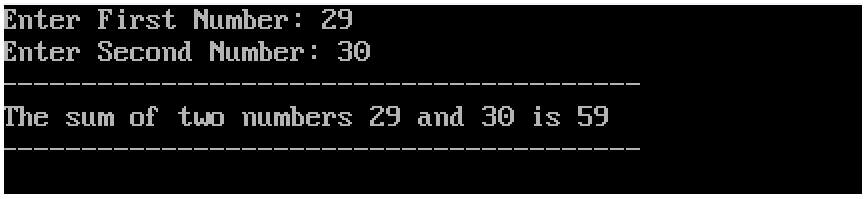
QUESTIONS:

1. **Write a program in C++ to print the sum of two numbers entered by user.**

Code:

```
#include <iostream.h>
#include <conio.h>
void main() {
clrscr();
int num1, num2, sum;
cout << "Enter First Number: ";
cin >> num1;
cout<< "Enter Second Number: ";
cin >> num2;
sum = num1 + num2;
cout << "---------------------------------------" << endl;
cout << "The sum of two numbers " << num1 << " and " << num2 << " is " <<
sum << endl;
cout << "---------------------------------------" << endl;
getch();
}
```

Output:



```
Enter First Number: 29
Enter Second Number: 30
---------------------------------------
The sum of two numbers 29 and 30 is 59
---------------------------------------
```

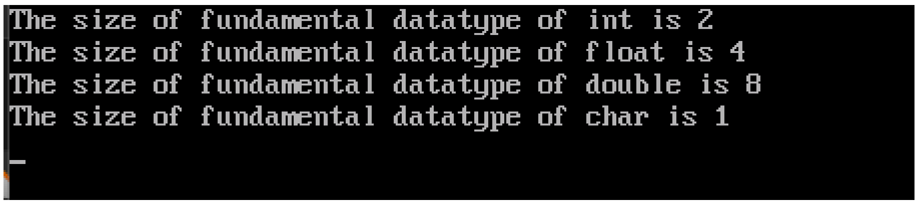
Figure 1: Output of addition of two numbers

2. Write a program in C++ to find Size of fundamental data types.

Code:

```
#include <iostream.h>
#include <conio.h>
void main(){
clrscr();
cout << "The size of fundamental datatype of int is " << sizeof(int) << endl;
cout << "The size of fundamental datatype of float is " << sizeof(float) << endl;
cout << "The size of fundamental datatype of double is " << sizeof(double) <<
endl;
cout << "The size of fundamental datatype of char is " << sizeof(char) << endl;
getch();
}
```

Output:



```
The size of fundamental datatype of int is 2
The size of fundamental datatype of float is 4
The size of fundamental datatype of double is 8
The size of fundamental datatype of char is 1
_
```

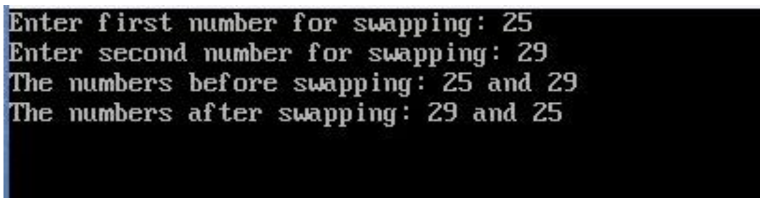
Figure 2: Output of the size of fundamental data types

3. Write a program in C++ to swap two numbers.

Code:

```
#include <iostream.h>
#include <conio.h>
void main(){
clrscr();
int num1, num2, num3;
cout << "Enter first number for swapping: ";
cin >> num1;
cout << "Enter second number for swapping: ";
cin >> num2;
cout << "The numbers before swapping: " << num1 << " and " << num2 << endl;
num3 = num1;
num1 = num2;
num2 = num3;
cout << "The numbers after swapping: " << num1 << " and " << num2 << endl;
getch();
}
```

Output:



```
Enter first number for swapping: 25
Enter second number for swapping: 29
The numbers before swapping: 25 and 29
The numbers after swapping: 29 and 25
```

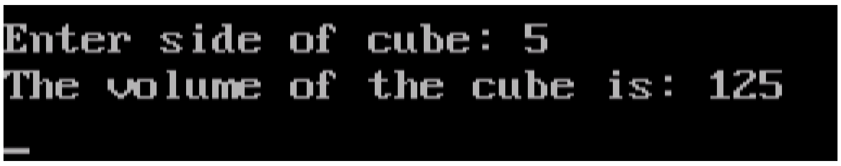
Figure 3: Output of the swapping of two numbers

4. **Write a program in C++ to calculate the volume of a cube.**

Code:

```
#include <iostream.h>
#include <conio.h>
void main() {
clrscr();
int side, vol;
cout << "Enter side of cube: ";
cin >> side;
vol = side*side*side;
cout << "The volume of the cube is: " << vol << endl;
getch();
}
```

Output:



```
Enter side of cube: 5
The volume of the cube is: 125
_
```

Figure 4: Output of the program calculating volume of cube

RESULTS:

The basic input-output and variable manipulation program in C++ have been written and executed successfully, for data type size computation and simple arithmetic operations, using Turbo C++ software.

CONCLUSIONS:

C++ fundamentals and basic programming concepts have been demonstrated through the creation of simple programs.

REFERENCES:

1. GeeksforGeeks. (2023, September 23). *C++ Data Types*. https://www.geeksforgeeks.org/cpp-data-types/
2. GeeksforGeeks. (2022, June 13). *History of C++*. https://www.geeksforgeeks.org/history-of-c/
3. DataFlair. (2020, February 3). *Advantages and Disadvantages of C++*. https://data-flair.training/blogs/advantages-and-disadvantages-of-cpp/
4. LearnCPP. (2023, September 26). 2.11 — Header files. https://www.learncpp.com/cpp-tutorial/header-files/

Date: 16/10/2023

PRACTICAL: 3
C++: Conditional Statements and Loops

AIM:
To create C++ programs using conditional statements and loops.

INTRODUCTION:
C++ is a versatile language that can be used for a wide range of applications, from operating systems and browsers to games and financial trading systems, due to its combination of low-level and high-level features which makes it a powerful language for programming complex systems. C++'s features like efficient memory management and high performance favored its extensive usage.
Conditional statements and loops are vital in C++ programming because they allow the programmer to regulate the flow of a program. They also help with decision-making and code block repetition. While loops let the programmer run a section of code repeatedly, conditional statements are used to compare variables and determine if a condition is true or false. They reduce code repetition, which makes programs easier to manage and maintain and acts as the essential elements for developing intricate algorithms and data structures.

Conditional statements:
Conditional statements are an important aspect of C++ programming that allow the programmer to control the flow of a program based on certain conditions. They are used to make decisions and execute specific blocks of code based on whether a condition is true or false. Conditional statements are essential for creating complex algorithms and data structures, as they help the programs to perform certain functions based on a condition.

Types of conditional statements:
1. 'If…' conditional statement:

The 'if…' conditional statement is used to implement single decision control instruction, based on whether the condition is true or not. If the condition is true, the if code block is executed or else, it is skipped.

Syntax:
```
if (condition) {
        //if_statements;}
```
Diagram:

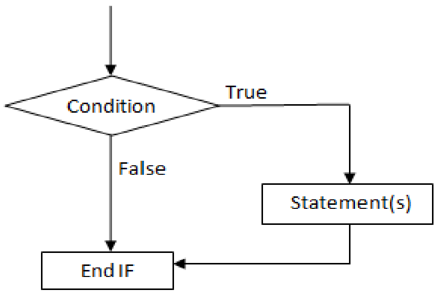


Figure 1: Flowchart for 'if…' conditional statement

2. 'If… else…' conditional statement:

The 'If… else…' conditional statement is used to implement dual decision control instruction, based on whether the condition is true or not. If the condition is true, the specified if code block is executed or else, the specified else code block is executed.

Syntax:

```
if (condition) {
        //if_statements;}
else {
        //else_statements;}
```

Diagram:

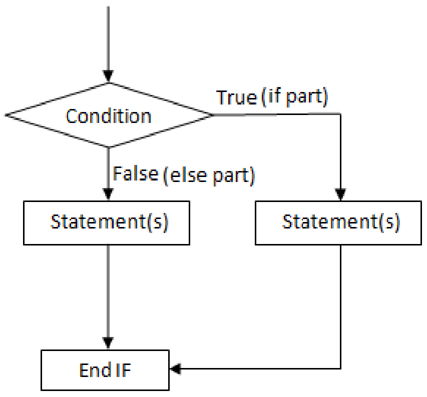


Figure 2: Flowchart for 'if… else…' conditional statement

3. 'If… else if…' conditional statement:

The 'If… else if…' conditional statement is used to implement multiple decision control instruction, based on whether the condition is true or not. If the condition is true, the specified if code block is executed or else, the specified else if code block is evaluated. If the else if code block condition is true, it is executed or else, skipped.

Syntax:

```
if (condition) {
        //statements;}
else if(condition) {
        //statements;}
        // as many else if as necessary
else {
        //statements;}
```

Diagram:

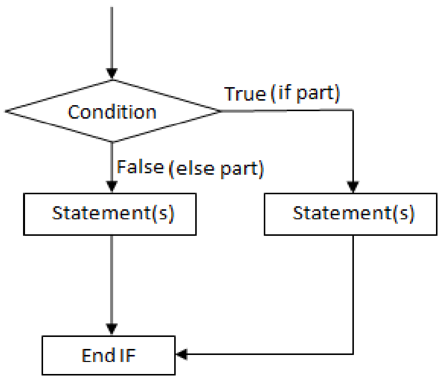


Figure 3: Flowchart for 'if… else if…' conditional statement

Loops:

Loops in C++ are used to execute a block of code repeatedly. There are three types of loops in C++: for, while, and do-while loops. Loops can be used to iterate the specific blocks of code. Loops are essential for creating complex algorithms and data structures, and they help to minimize the repetition of code, making programs more manageable and easier to maintain.

Types of loops:

1. **'For' loop:**

   The 'for' loop is used to repeat/iterate a single statement of code or a block of code for a limited/pre-defined number of times.

Syntax:

```
for(var_initilization; var_condition; var_updatation) {
        //statements;
        //place as many statements here as necessary}
```

Diagram:

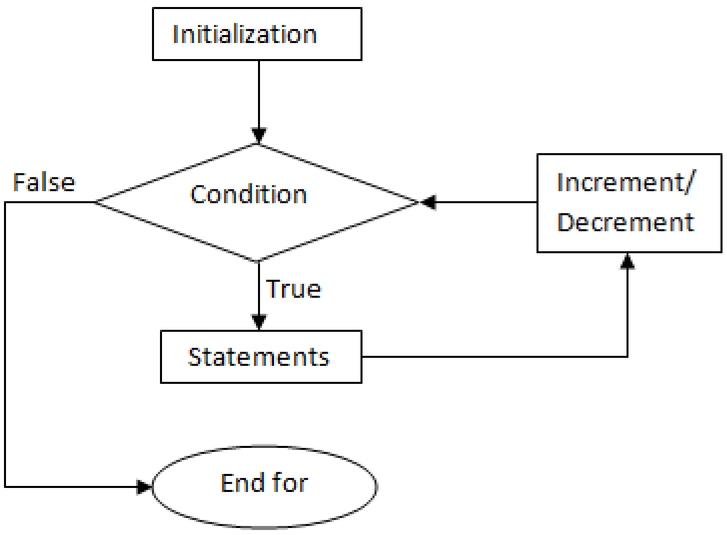


Figure 4: Flowchart for 'for' loop

2. **'While' loop:**

   The 'while' loop is used to repeat/iterate a single statement of code or a block of code until the provided condition is true or false.

Syntax:

```
while(condition) {
        //statements;
        //place as many statements here as necessary}
```

Diagram:

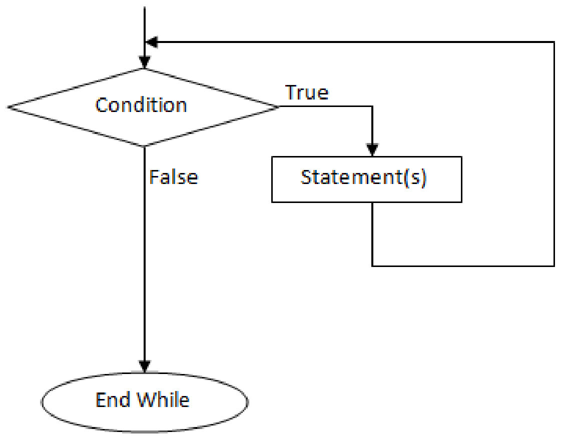


Figure 5: Flowchart for 'while' loop

3. **'Do… while…' loop:**

   The 'Do… while…' loop is used to repeat/iterate a single statement of code or a block of code for a limited/pre-defined number of times.

Syntax:

```
do {
        //statements;
        //place as many statements here as necessary}
while(condition);
```

Diagram:

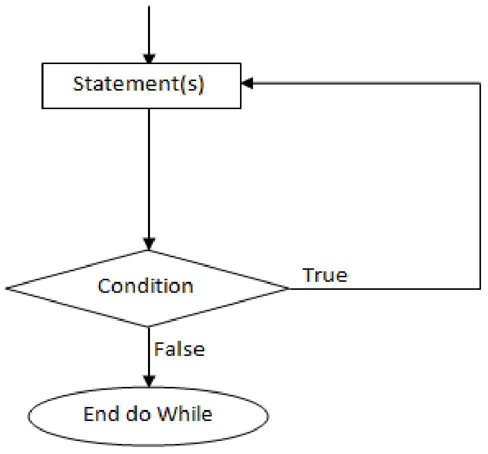


Figure 6: Flowchart for 'do… while…' loop

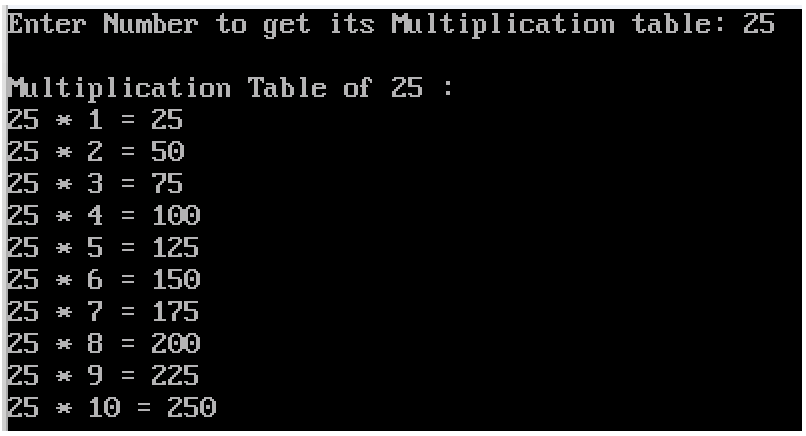
QUESTIONS:

1. **Write a program in C++ that takes a number as input and prints its multiplication table up to 10.**

Code:

```
#include <iostream.h>
#include <conio.h>
void main() {
clrscr();
cout << "Enter Number to get its Multiplication table: ";
int num, product;
cin >> num;
cout << endl <<"Multiplication Table of " << num << " :" << endl;
for (int i = 1; i <= 10; i++) {
product = num * i;
cout << num << " * " << i << " = " << product << endl;
}
getch();
}
```

Output:



```
Enter Number to get its Multiplication table: 25

Multiplication Table of 25 :
25 * 1 = 25
25 * 2 = 50
25 * 3 = 75
25 * 4 = 100
25 * 5 = 125
25 * 6 = 150
25 * 7 = 175
25 * 8 = 200
25 * 9 = 225
25 * 10 = 250
```

Figure 7: Output of the program that prints multiplication table

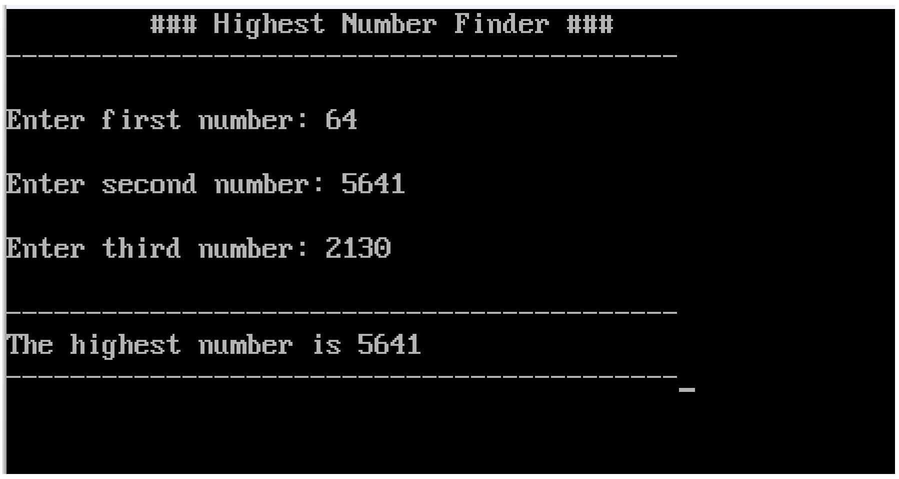
2. Write a C++ program which prints three highest numbers from entered 3 numbers.

Code:

```
#include <iostream.h>
#include <conio.h>
void main() {
clrscr();
cout << "\t ### Highest Number Finder ###";
int n1, n2, n3;
cout << "\n-----------------------------------------" << endl;
cout << "\nEnter first number: ";
cin >> n1;
cout << "\nEnter second number: ";
cin >> n2;
cout << "\nEnter third number: ";
cin >> n3;
cout << "\n-----------------------------------------" << endl;
if (n1>n2 && n1>n3) {
cout << "The highest number is " << n1;
}
else if (n2>n1 && n2>n3) {
cout << "The highest number is " << n2;
}
else {
```

```
cout << "The highest number is " << n3;
}
cout << "\n----------------------------------------";
getch();
}
```

Output:



```
          ### Highest Number Finder ###
------------------------------------------

Enter first number: 64

Enter second number: 5641

Enter third number: 2130

------------------------------------------
The highest number is 5641
------------------------------------------_
```

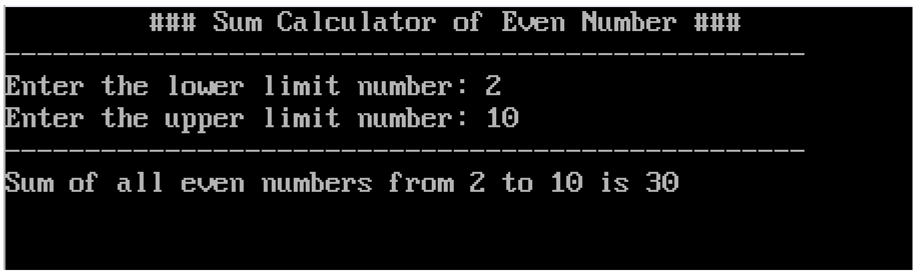
Figure 8: Output of the program that outputs the highest of inserted numbers

3. Write a C++ program to compute the sum of even numbers.

Code:

```
#include <iostream.h>
#include <conio.h>
void main(){
clrscr();
int lnum, unum, sum;
cout << "\t ### Sum Calculator of Even Number ###" << endl;
cout << "-----------------------------------------------";
cout << "\nEnter the lower limit number: ";
cin >> lnum;
cout << "Enter the upper limit number: ";
cin >> unum;
sum = 0;
for (int i=lnum; i<=unum; i+= 2) {
sum += i;
}
cout << "-----------------------------------------------";
cout<<"\nSum of all even numbers from " <<lnum << " to " <<unum << " is "
<<sum;
getch();
}
```

Output:



```
          ### Sum Calculator of Even Number ###
--------------------------------------------------
Enter the lower limit number: 2
Enter the upper limit number: 10
--------------------------------------------------
Sum of all even numbers from 2 to 10 is 30
```

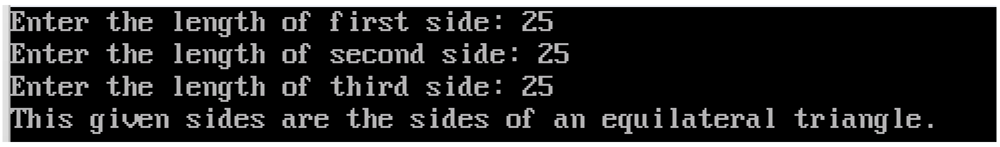
Figure 9: Output of the program that computes the sum of even numbers

4. **Write a c++ program to accept sides of a triangle and display equilateral, isosceles and scalene.**

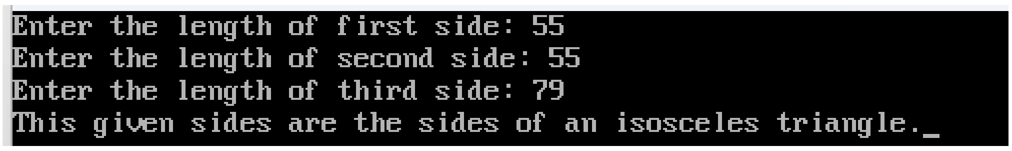
Code:

```
#include <iostream.h>
#include <conio.h>
void main() {
clrscr();
int side1, side2, side3;
cout << "Enter the length of first side: ";
cin >> side1;
cout << "Enter the length of second side: ";
cin >> side2;
cout << "Enter the length of third side: ";
cin >> side3;
if (side1==side2 && side2==side3 && side1==side3) {
cout << "This given sides are the sides of an equilateral triangle.";
}
else if (side1==side2 || side2==side3 || side1==side3) {
cout << "This given sides are the sides of an isosceles triangle.";
}
else {
cout << "This given sides are the sides of an scalene triangle.";
}
getch();
}
```

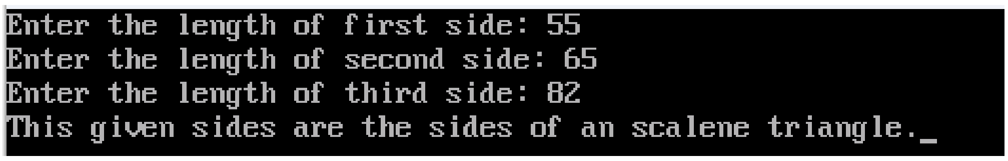
Output:



```
Enter the length of first side: 25
Enter the length of second side: 25
Enter the length of third side: 25
This given sides are the sides of an equilateral triangle.
```



```
Enter the length of first side: 55
Enter the length of second side: 55
Enter the length of third side: 79
This given sides are the sides of an isosceles triangle._
```



```
Enter the length of first side: 55
Enter the length of second side: 65
Enter the length of third side: 82
This given sides are the sides of an scalene triangle._
```

Figure 10: Output of the program that identifies the type of triangle based on the length of its sides

RESULTS:

Using C++ and its concepts of conditional statements and loops, programs were created to print multiplication table upto 10, identify the highest number, calculate the sum of all the even numbers between the inserted numbers as well as to identify the type of triangle based on the length of its sides.

CONCLUSION:

The concepts of C++ conditional statements and loops, which help in decision making and automations, have been demonstrated through the creation of the programs.

REFERENCES:

1. Yang. (2022). CSCI 2170: Computer Science II. Lectures - CSCI 2170: Computer Science II. https://www.cs.mtsu.edu/~xyang/2170/
2. W3Schools. (n.d.). C++ Conditions. https://w3schools.com/cpp/cpp_conditions.asp
3. GeeksforGeeks. (n.d.). C++ Loops. https://www.geeksforgeeks.org/cpp-loops/

Date: 02/11/2023

PRACTICAL: 4
C++: Arrays

AIM:
To explore and understand arrays by creating C++ programs.

INTRODUCTION:
C++ is a versatile and powerful programming language that combines low-level and high-level features, making it suitable for a wide range of applications. Some key features of C++ include efficiency and performance, cross-platform compatibility, effective memory management, and versatility. As a result of its noteworthy features, C++ has been extensively used in memory-intensive programs and other complex systems. Its ability to directly manipulate hardware resources and provide efficient memory management make it an attractive choice for developers working on resource-constrained applications or performance-critical tasks.

Arrays:
In C++, an array is a data structure used to store multiple values of the same data type in a contiguous memory location. It allows for efficient storage and access of elements, with indexing starting from 0. Arrays are a fundamental data structure in computer programming and are used to solve various problems, such as storing and manipulating large amounts of data, implementing algorithms and data structures, and representing graphs.
Arrays provide efficient access to elements, are easy to implement and understand, and are compatible with hardware. By mastering arrays, C++ programmers can enhance their overall programming skills and create more efficient and user-friendly software applications.

Syntax:
datatype array_name[size] = {“element1”, “element2”, …, “elementN”};
Eg. int arr[5] = {25, 3, 243, 2834, 82125} or string arr[2] = {“apple”, “ball”};

Types of Arrays:
1. **One-dimensional array:** One dimensional array is also known as a list or a linear array. It consists of only one column or one row.
   The declaration syntax for a one-dimensional array in C++ is given by the following: datatype array_name[x], where ‘x’ represents the number of elements to be stored in the array. For example, int arr[3].
2. **Two-dimensional array:** A two-dimensional array consists of columns and rows. Each element of the two-dimensional array is referenced by its index values or subscripts. The index value of a two-dimensional array consists of two subscripts. One subscript represents the row number and the second subscript represents the column number.
   The declaration syntax for a two-dimensional array in C++ is given by the following: data_type array_name[x][y], where ‘x’ represents the number of rows and ‘y’ represents the number of columns. For example, int arr[2][3].

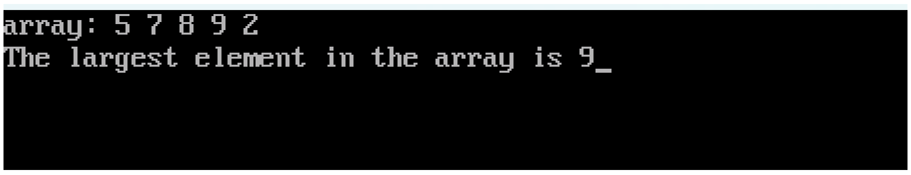
QUESTIONS:

1. **Write a C++ program to find the largest element of a given array of integers. 5,7,8,9,2.**

Code:

```
#include<iostream.h>
#include<conio.h>
void main() {
clrscr();
int arr[5] = {5, 7, 8, 9, 2};
int largest = arr[0];
for(int i=1; i<5; i++) {
if(arr[i] > largest) {
largest = arr[i];
}
}
cout << "array: ";
for(int j=0; j<5; j++) {
cout << arr[j] << " ";
}
cout << endl;
cout << "The largest element in the array is " << largest;
getch();
}
```

Output:



```
array: 5 7 8 9 2
The largest element in the array is 9_
```

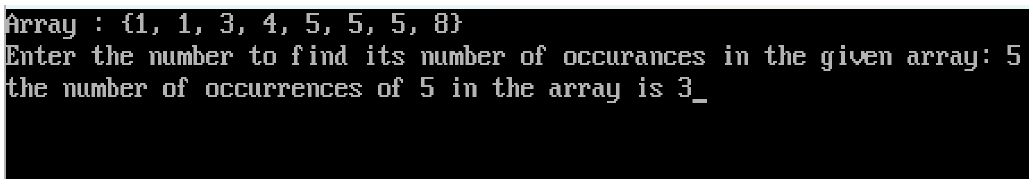
Figure 1: Output of the program that finds the largest element in the array

2. **Write a C++ program to count the number of occurrences of given number in an array of integers. 1, 1, 3, 4, 5, 5, 5, 8.**

Code:

```
#include<iostream.h>
#include<conio.h>
void main() {
clrscr();
int arr[8] = {1, 1, 3, 4, 5, 5, 5, 8};
int num;
int count = 0;
cout << "Array : {";
for (int j=0; j<7; j++) {
cout << arr[j] << ", ";
}
cout << arr[7] << "}" << endl;
cout << "Enter the number to find its number of occurances in the given array: ";
cin >> num;
for(int i=0; i<8; i++) {
if(arr[i] == num) {
count++;
}
}
cout << "the number of occurrences of "<<num<<" in the array is "<<count;
getch();
}
```

Output:



```
Array : {1, 1, 3, 4, 5, 5, 5, 8}
Enter the number to find its number of occurances in the given array: 5
the number of occurrences of 5 in the array is 3_
```

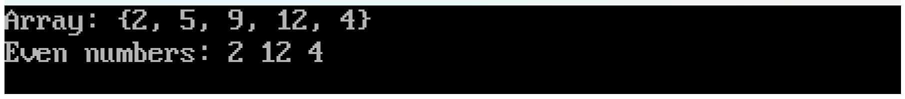
Figure 2: Output of the program that count the number of occurrences of given number in an array

3. Write a C++ program to display even number from an array. 2, 5, 9, 12, 4.

Code:

```
#include<iostream.h>
#include<conio.h>
void main() {
 clrscr();
 int arr[5] = {2, 5, 9, 12, 4};
 cout<<"Array: {";
 for(int j=0; j<4; j++) {
  cout<<arr[j]<<", ";
 }
 cout<<arr[4]<<"}"<<endl;
 cout<<"Even numbers: ";
 for(int i=0; i<5; i++) {
  if(arr[i] % 2 == 0) {
   cout<<arr[i]<<" ";
  }
 }
 getch();
}
```

Output:



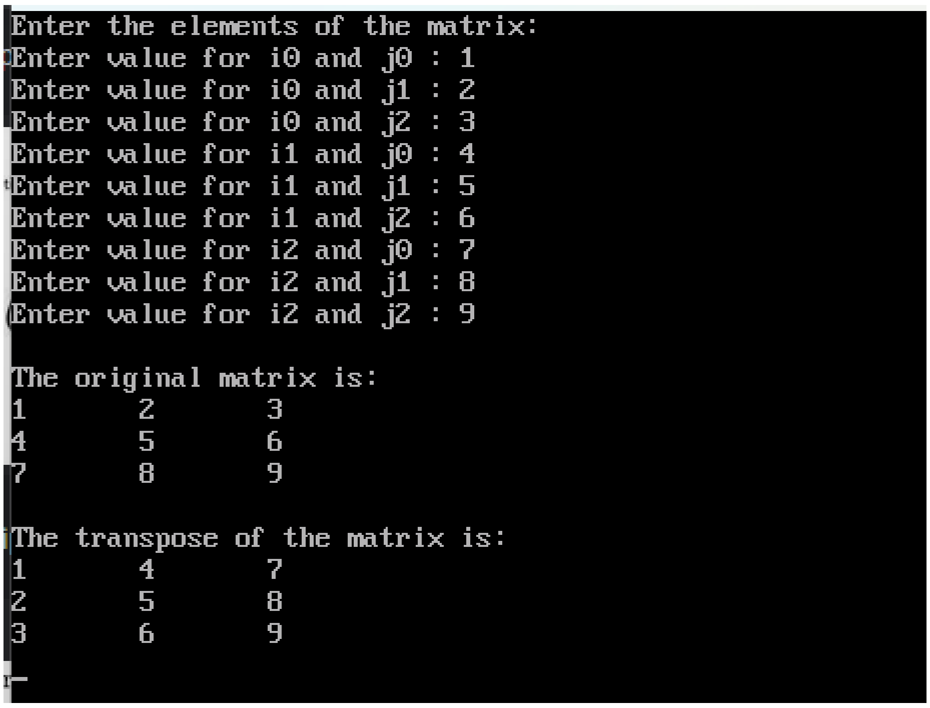
```
Array: {2, 5, 9, 12, 4}
Even numbers: 2 12 4
```

Figure 3: Output of the program that display even number from an array

4. **Write a C++ program to accept 3x3 matrix and display the transpose of a given matrix.**

Code:
```
#include<iostream.h>
#include<conio.h>
void main(){
clrscr();
int mat[3][3], trans_mat[3][3];
cout<<"Enter the elements of the matrix:"<<endl;
for(int i=0; i<3; i++){
for(int j=0; j<3; j++){
cout << "Enter value for i" << i << " and j" << j << " : ";
cin>>mat[i][j];
}
}
cout << endl;
cout<<"The original matrix is:"<<endl;
for(int m=0; m<3; m++){
for(int n=0; n<3; n++){
cout<<mat[m][n]<<"\t";
}
cout<<endl;
}
cout << endl;
for(int a=0; a<3; a++){
for(int b=0; b<3; b++){
trans_mat[b][a] = mat[a][b];
}
}
cout<<"The transpose of the matrix is:"<<endl;
for(int x=0; x<3; x++){
for(int y=0; y<3; y++){
cout<<trans_mat[x][y]<<"\t";
}
cout<<endl;
}
getch();
}
```

Output:



```
Enter the elements of the matrix:
Enter value for i0 and j0 : 1
Enter value for i0 and j1 : 2
Enter value for i0 and j2 : 3
Enter value for i1 and j0 : 4
Enter value for i1 and j1 : 5
Enter value for i1 and j2 : 6
Enter value for i2 and j0 : 7
Enter value for i2 and j1 : 8
Enter value for i2 and j2 : 9

The original matrix is:
1       2       3
4       5       6
7       8       9

The transpose of the matrix is:
1       4       7
2       5       8
3       6       9

_
```

Figure 4: Output of the program that accept 3x3 matrix and displays its transpose

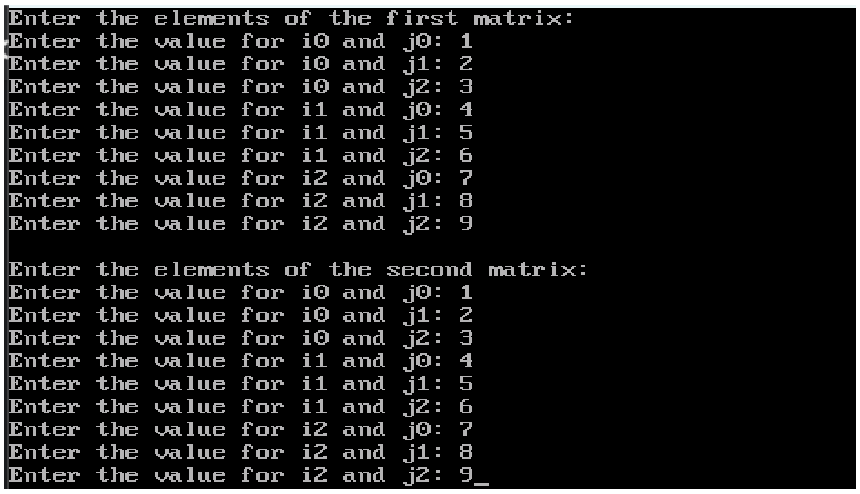
5. **Write a C++ program to accept two 3x3 matrix and display sum of two matrices.**

Code:

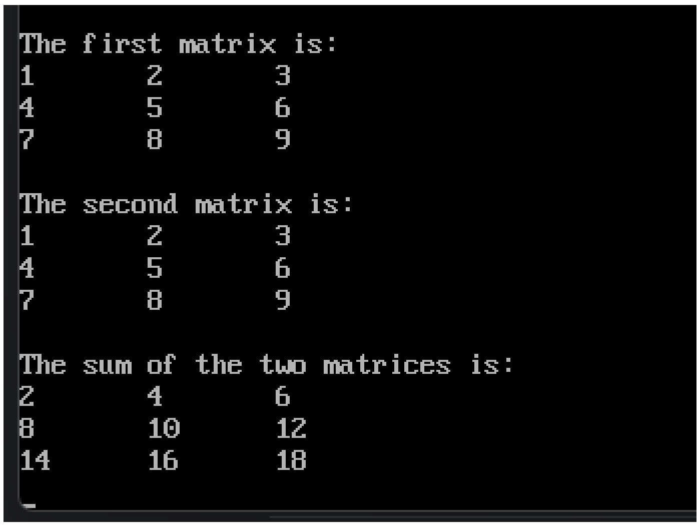
```
#include<iostream.h>
#include<conio.h>
void main() {
clrscr();
int mat1[3][3], mat2[3][3], sum[3][3];
cout<<"Enter the elements of the first matrix:"<<endl;
for(int i=0; i<3; i++) {
for(int j=0; j<3; j++) {
cout << "Enter the value for i" << i << " and j" << j << ": ";
cin >> mat1[i][j];
}
}
cout << endl;
cout<<"Enter the elements of the second matrix:"<<endl;
for(int m=0; m<3; m++) {
for(int n=0; n<3; n++) {
cout << "Enter the value for i" << m << " and j" << n << ": ";
cin>>mat2[m][n];
}
}
cout << endl;
cout << "The first matrix is:" << endl;
for(int a=0; a<3; a++) {
for(int b=0; b<3; b++) {
```

```
cout << mat1[a][b] << "\t";
}
cout << endl;
}
cout << endl;
cout << "The second matrix is:" << endl;
for(int x=0; x<3; x++) {
for(int y=0; y<3; y++) {
cout << mat2[x][y] << "\t";
}
cout<<endl;
}
for(int c=0; c<3; c++) {
for(int d=0; d<3; d++) {
sum[c][d] = mat1[c][d] + mat2[c][d];
}
}
cout << endl;
cout << "The sum of the two matrices is:" << endl;
for(int r=0; r<3; r++) {
for(int k=0; k<3; k++) {
cout<<sum[r][k]<<"\t";
}
cout<<endl;
}
getch();
}
```

Output:



```
Enter the elements of the first matrix:
Enter the value for i0 and j0: 1
Enter the value for i0 and j1: 2
Enter the value for i0 and j2: 3
Enter the value for i1 and j0: 4
Enter the value for i1 and j1: 5
Enter the value for i1 and j2: 6
Enter the value for i2 and j0: 7
Enter the value for i2 and j1: 8
Enter the value for i2 and j2: 9

Enter the elements of the second matrix:
Enter the value for i0 and j0: 1
Enter the value for i0 and j1: 2
Enter the value for i0 and j2: 3
Enter the value for i1 and j0: 4
Enter the value for i1 and j1: 5
Enter the value for i1 and j2: 6
Enter the value for i2 and j0: 7
Enter the value for i2 and j1: 8
Enter the value for i2 and j2: 9_
```



```
The first matrix is:
1       2       3
4       5       6
7       8       9

The second matrix is:
1       2       3
4       5       6
7       8       9

The sum of the two matrices is:
2       4       6
8       10      12
14      16      18
```

Figure 5: Output of the program that accept two 3x3 matrix and display sum of those two matrices.

RESULTS:

Using C++ and its concept of arrays, programs were created to take input and store the values or elements as one-dimensional as well as two-dimensional array such as to display the largest number, to find the number of occurrences, to display even numbers as well as to create the transpose matrix.

CONCLUSION:

The concept of arrays of C++, which can be used for storing and manipulating large amounts of data, have been explored and demonstrated through the creation of the programs.

REFERENCES:

1. Yang. (2022). *CSCI 2170: Computer Science II*. Lectures - CSCI 2170: Computer Science II. https://www.cs.mtsu.edu/~xyang/2170/
2. GeeksforGeeks. (n.d.). *Multidimensional Arrays in C*. https://www.geeksforgeeks.org/multidimensional-arrays-in-c/
3. W3Schools. (n.d.). *C++ Arrays*. https://www.w3schools.com/cpp/cpp_arrays.asp

Date: 02/11/2023

PRACTICAL: 5
C++: Functions and Structures

AIM:
To explore and understand functions and structures by creating C++ programs.

INTRODUCTION:
C++ is a versatile and powerful programming language that combines low-level and high-level features, making it suitable for a wide range of applications. Some key features of C++ include efficiency and performance, cross-platform compatibility, effective memory management, and versatility. As a result of its noteworthy features, C++ has been extensively used in memory-intensive programs and other complex systems. Its ability to directly manipulate hardware resources and provide efficient memory management make it an attractive choice for developers working on resource-constrained applications or performance-critical tasks.

Functions:
Functions in C++ are blocks of code that perform specific operations and can optionally define input parameters for passing arguments and return a value as output. They are essential for encapsulating common operations in a single reusable block, reducing code redundancy, and making code modular. By defining functions, programmers can avoid rewriting the same series of instructions over and over, making it easier to write programs and keep track of their functionality. Functions are used to perform logically isolated tasks, enhancing code reusability, maintainability, and readability. They are crucial for breaking down complex tasks into smaller, manageable parts, improving the overall structure and organization of programs, and promoting the concept of modular programming.
Additionally, functions are used extensively in the C++ standard library and can be defined at class scope or namespace scope, providing flexibility and code organization. Overall, functions play a vital role in simplifying program development, promoting code reuse, and enhancing the maintainability and scalability of C++ programs.

Types of Functions:
1. **Built-in Functions:**
   Built-in functions in C++, also called library functions, are pre-implemented and accessible without user implementation, residing in header files. They streamline common tasks, like math operations and string manipulations, contributing to code simplicity, development ease, and program scalability in C++.
2. **User-defined Functions:**
   User-defined functions in C++ are custom code blocks with a designated name, enabling modular programming by encapsulating specific tasks. They enhance code organization, readability, and reusability, promoting easier maintenance and scalability in C++ programs.

Types of User-defined Functions:

1. **Function with no argument and no return value:**
   Functions that have no arguments and no return values. Such functions can either be used to display information or to perform any task on global variables.
   Syntax:
   ```
   void function_name() {
   //statements
   }
   ```

2. **Function with no argument but a return value:**
   Functions that have no arguments but have some return values. Such functions are used to perform specific operations and return their value.
   Syntax:
   ```
   return_type function_name () {
   //statements
   return value;
   }
   ```

3. **Function with arguments but no return value:**
   Functions that have arguments but no return values. Such functions are used to display or perform some operations on given arguments.
   Syntax:
   ```
   void function_name (arg1, arg2, …, argN) {
   //statements
   }
   ```
   Where, arg = datatype variable and/or value

4. **Function with arguments and return value:**
   Functions that have arguments and some return value. These functions are used to perform specific operations on the given arguments and return their values to the user.
   Syntax:
   ```
   return_type function_name (arg1, arg2, …, argN) {
   //statements
   return value;
   }
   ```
   Where, arg = datatype variable and/or value

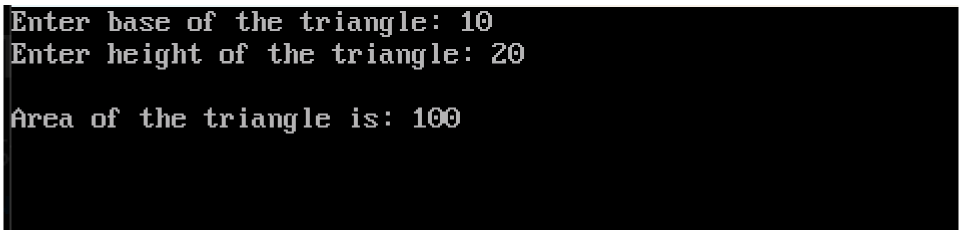
QUESTIONS:

1. **Write a C++ program to create a function named as area and find area of a triangle using no return and no parameter.**

Code:
```
#include <iostream.h>
#include <conio.h>
void area() {
float base, height, ans;
cout << "Enter base of the triangle: ";
cin >> base;
cout << "Enter height of the triangle: ";
```

```
cin >> height;
ans = 0.5 * base * height;
cout << endl << "Area of the triangle is: " << ans << endl;
}
void main() {
clrscr();
area();
getch();
}
```

Output:



```
Enter base of the triangle: 10
Enter height of the triangle: 20

Area of the triangle is: 100
```

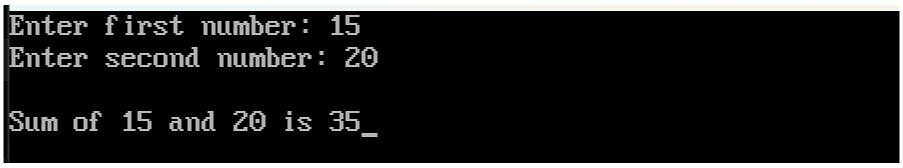
Figure 1: Output of the program that finds the area of a triangle

2. **Write a C++ program to create a function for adding two numbers using return and with parameters.**

Code:

```
#include <iostream.h>
#include <conio.h>
int add(int a, int b) {
int sum = a + b;
return sum;
}
void main() {
clrscr();
int num1, num2, ans;
cout << "Enter first number: ";
cin >> num1;
cout << "Enter second number: ";
cin >> num2;
ans = add(num1, num2);
cout << endl;
cout<<"Sum of "<<num1<<" and "<<num2<<" is "<<ans;
getch();
}
```

Output:



```
Enter first number: 15
Enter second number: 20

Sum of 15 and 20 is 35_
```

Figure 2: Output of the program that adds two numbers

RESULTS:

Using C++ and its concept of functions, programs were created to take input, process the input and give appropriate output by using user-defined functions, with and without function arguments (parameters) and return.

CONCLUSION:

The concept of functions in C++, which can be used for carrying out the same process for different parameters or values, have been explored and demonstrated through the creation of the programs.

REFERENCES:

1. Yang. (2022). *CSCI 2170: Computer Science II*. Lectures - CSCI 2170: Computer Science II. https://www.cs.mtsu.edu/~xyang/2170/
2. GeeksforGeeks. (2023, September 4). *Functions in C++*. https://www.geeksforgeeks.org/functions-in-cpp/

Date: 02/11/2023

PRACTICAL: 6
C++: String Manipulation

AIM:
To explore and understand string manipulation by creating C++ programs.

INTRODUCTION:
C++ is a versatile and powerful programming language that combines low-level and high-level features, making it suitable for a wide range of applications. Some key features of C++ include efficiency and performance, cross-platform compatibility, effective memory management, and versatility. As a result of its noteworthy features, C++ has been extensively used in memory-intensive programs and other complex systems. Its ability to directly manipulate hardware resources and provide efficient memory management make it an attractive choice for developers working on resource-constrained applications or performance-critical tasks.

String:
A string is an ordered array of characters that can consist of alphanumeric characters mixed with numbers and symbols. Strings are used to store data expressed as characters, transmit information between a software and its user, and store human-readable text. Programmers need to know how to manipulate strings to modify, validate, and extract text from strings. This includes operations like search-and-replace, string splitting and re-joining, and character addition and removal.

String functions:
String manipulation functions in C++ are essential for various applications, such as web apps and business ideas like Grammarly, which rely on string manipulation to check and correct grammar, spelling, and punctuation errors. These functions allow developers to modify, validate, and extract text from strings, involving tasks like adding and removing characters, splitting and re-joining strings, and performing search-and-replace operations. By mastering string manipulation functions, C++ programmers can enhance their overall programming skills and create more efficient and user-friendly software applications

Function Name	Function explanation	Function syntax
strcat()	Concatenates two strings	strcat(char *str1, char *str2);
strcpy()	Copies one string to another	strcpy(char *destination, char *source);
strcmp()	Compares two strings and returns a value indicating their relative order	strcmp(char *str1, char *str2);
strlen()	Returns the length of a string	int strlen(char *str);

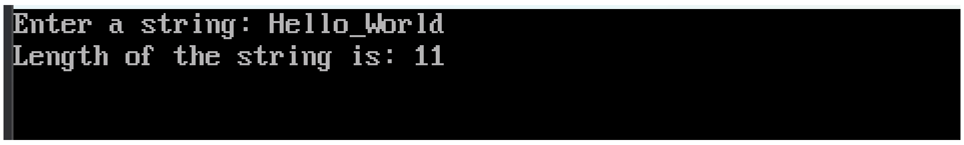
QUESTIONS:

1. **Write a C++ program to accept a string and find the length of a string.**

Code:

```
#include <iostream.h>
#include <conio.h>
#include <string.h>
void main() {
clrscr();
char str[100];
cout << "Enter a string: ";
cin >> str;
int len = strlen(str);
cout << "Length of the string is: " << len;
getch();
}
```

Output:



```
Enter a string: Hello_World
Length of the string is: 11
```

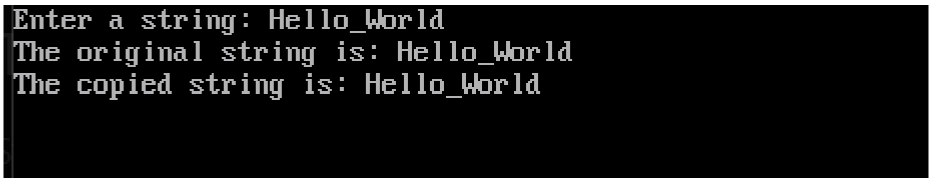
Figure 1: Output of the program that finds the length of the input string

2. **Write a C++ program to copy a string onto another string.**

Code:

```
#include <iostream.h>
#include <conio.h>
#include <string.h>
void main() {
clrscr();
char str1[100], str2[100];
cout << "Enter a string: ";
cin >> str1;
strcpy(str2, str1);
cout << "The original string is: " << str1 << endl;
cout << "The copied string is: " << str2 << endl;
getch();
}
```

Output:



```
Enter a string: Hello_World
The original string is: Hello_World
The copied string is: Hello_World
```

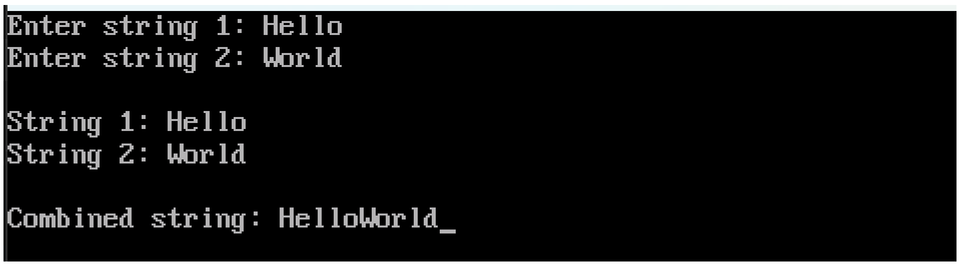
Figure 2: Output of the program that copies and outputs input string

3. Write a C++ program to combine two strings.

Code:

```
#include <iostream.h>
#include <conio.h>
#include <string.h>
void main() {
clrscr();
char str1[50], str2[50];
cout << "Enter string 1: ";
cin >> str1;
cout << "Enter string 2: ";
cin >> str2;
cout << endl;
cout << "String 1: " << str1 << endl;
cout << "String 2: " << str2 << endl;
strcat(str1, str2);
cout << endl;
cout << "Combined string: " << str1;
getch();
}
```

Output:



```
Enter string 1: Hello
Enter string 2: World

String 1: Hello
String 2: World

Combined string: HelloWorld_
```

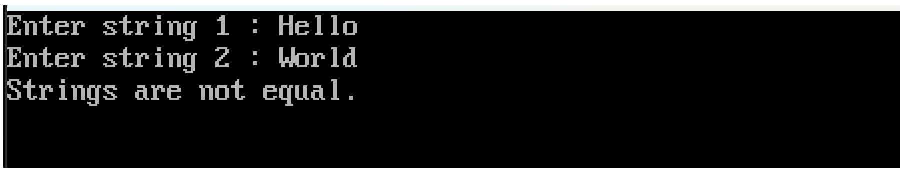
Figure 3: Output of the program that combines two strings

4. Write a C++ program to compare whether entered two strings are equal or not.

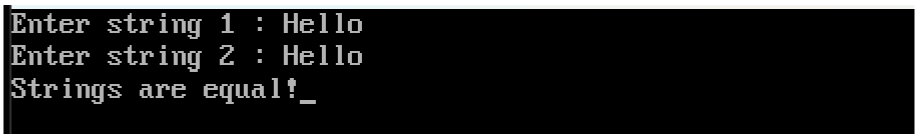
Code:

```
#include<iostream.h>
#include<string.h>
#include<conio.h>
int main () {
clrscr();
char str1[50], str2[50];
cout<<"Enter string 1 : ";
cin >> str1;
cout<<"Enter string 2 : ";
cin >> str2;
if(strcmp(str1, str2)==0)
cout << "Strings are equal!";
else
cout << "Strings are not equal.";
getch();
return 0;
}
```

Output:



```
Enter string 1 : Hello
Enter string 2 : World
Strings are not equal.
```



```
Enter string 1 : Hello
Enter string 2 : Hello
Strings are equal!_
```

Figure 4: Output of the program that compare whether entered two strings are equal or not

RESULTS:

Using C++ and its concept of string manipulation, programs were created to take input as a string and find its length along with other programs to copy a string onto another string as well as to combine and compare two strings.

CONCLUSION:

The concept of string manipulation of C++, which can be used as grammar and vocabulary checker to check and correct grammar, spelling, and punctuation errors, have been demonstrated through the creation of the programs.

REFERENCES:

1. Yang. (2022). *CSCI 2170: Computer Science II*. Lectures - CSCI 2170: Computer Science II. https://www.cs.mtsu.edu/~xyang/2170/
2. Great Learning Team. (2022, October 25). *Strings in C++ | What are Strings in C++?* My Great Learning. https://www.mygreatlearning.com/blog/strings-in-cpp/
3. Tutorials Point. (n.d.). *C++ Strings*. https://www.tutorialspoint.com/cplusplus/cpp_strings.htm

Date: 03/11/2023

PRACTICAL: 7
C++: Class, Object and Encapsulation

AIM:
To explore and understand concept of class, object and encapsulation by creating C++ programs.

INTRODUCTION:
C++ is a versatile and powerful programming language that combines low-level and high-level features, making it suitable for a wide range of applications. Some key features of C++ include efficiency and performance, cross-platform compatibility, effective memory management, and versatility. As a result of its noteworthy features, C++ has been extensively used in memory-intensive programs and other complex systems. Its ability to directly manipulate hardware resources and provide efficient memory management make it an attractive choice for developers working on resource-constrained applications or performance-critical tasks.

Class:
In C++, a class is a user-defined data type that encapsulates data members and member functions, which can be accessed and used by creating an instance of that class. Classes are used in object-oriented programming (OOP) to represent real-world entities and provide a blueprint for creating objects. They are essential for achieving encapsulation, inheritance, and polymorphism, allowing programmers to create complex data structures and algorithms. The class of a program can be executed by calling that class by creating an instance of that class, known as, object.
Classes in C++ can be used in creating user interfaces, implementing data structures, developing games, and building software applications. By defining classes, programmers can create reusable code, enhance code organization and readability, and promote the concept of modular programming.

Object:
an object is an instance of a class, representing a real-world entity with its own set of attributes and behaviors. Classes act as blueprints to create objects with similar properties, providing a fundamental idea around which the object-oriented approach revolves.

Access Specifier/Modifier:
The access specifiers determine which class members are accessible to other classes and functions and which are not. Access specifiers are used to implement an important aspect of Object-Oriented Programming known as Data Hiding, which is crucial for achieving encapsulation and ensuring the security and integrity of the data within the class. By using access specifiers, programmers can control the accessibility of class members, ensuring the security and integrity of the data within the class, and promoting the concept of modular programming. There are three access specifiers in C++: public, private, and protected.

Types of Access Specifiers/Modifiers:

1. **Public:**
   All the class members declared under the public specifier will be available to everyone. The data members and member functions declared as public can be accessed by other classes and functions too.
2. **Private:**
   The private members of a class can only be accessed by the member functions inside the class. They are not allowed to be accessed directly by any object or function outside the class.
3. **Protected:**
   The protected members are similar to private members, but they can be accessed in the derived classes. They are not accessible from outside the class.

Syntax:

```
class className {
        accessSpecifier: //can be public, private or protected
                void functionName() { cout << "Hello, World!" << endl; }
};
datatype main() {
        className obj1;
        obj1.functionName ();
        return type;}
```

Encapsulation:
Encapsulation in C++ involves bundling data members and member functions inside a single class, providing data hiding and abstraction. By combining similar data and functions into a single unit, encapsulation protects the data from unauthorized access and modification. This process uses classes to define the structure and behavior of objects, objects to represent real-world entities with their own attributes and behaviors, and access specifiers (public, private, and protected) to control the accessibility of class members. Encapsulation ensures better control of data, enhances security, and simplifies the usage of classes while allowing the internal implementation to be modified without impacting external code.

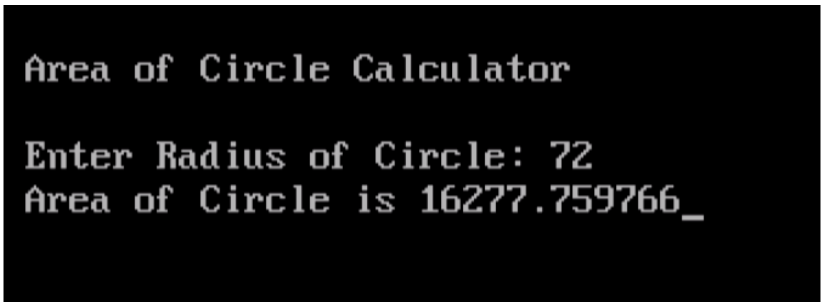
QUESTIONS:

1. **Write a program to create class Area to calculate area of circle using private variables and public functions.**

Code:

```
#include <iostream.h>
#include <conio.h>

class Area {
private:
float area;
float r;
public:
void carea(){
cout << "Area of Circle Calculator" << endl;
cout << endl << "Enter Radius of Circle: ";
cin >> r;
area = 3.14*r*r;
cout << "Area of Circle is " << area;
}
};

int main(){
clrscr();
Area obj;
obj.carea();
getch();
return 0;
}
```

Output:



```
Area of Circle Calculator

Enter Radius of Circle: 72
Area of Circle is 16277.759766_
```

Figure 1: Output of the program that calculates the area of a circle

RESULTS:

Using C++ and its concept of encapsulation, consisting of classes, objects & access specifiers, program was created to take input of the radius of the circle and calculate its area.

CONCLUSION:

The concept of encapsulation of C++, that ensures better control over the data, enhances security and integrity of data and also simplifies the usage of classes, have been demonstrated through the creation of the programs.

REFERENCES:

1. Yang. (2022). *CSCI 2170: Computer Science II*. Lectures - CSCI 2170: Computer Science II. https://www.cs.mtsu.edu/~xyang/2170/
2. Programiz. (n.d.). *C++ Encapsulation (With Examples)*. https://www.programiz.com/cpp-programming/encapsulation
3. GeeksforGeeks. (2022, June 22). *Access Modifiers in C++*. https://www.geeksforgeeks.org/access-modifiers-in-c/
4. W3Schools. (n.d.). *C++ Classes and Objects*. https://www.w3schools.com/cpp/cpp_classes.asp

Date: 03/11/2023

PRACTICAL: 8
C++: Inheritance

AIM:

To explore and understand concept of inheritance by creating C++ programs.

INTRODUCTION:

C++ is a versatile and powerful programming language that combines low-level and high-level features, making it suitable for a wide range of applications. Some key features of C++ include efficiency and performance, cross-platform compatibility, effective memory management, and versatility. As a result of its noteworthy features, C++ has been extensively used in memory-intensive programs and other complex systems. Its ability to directly manipulate hardware resources and provide efficient memory management make it an attractive choice for developers working on resource-constrained applications or performance-critical tasks.

Inheritance:

In C++, inheritance is a mechanism that allows new classes (derived/child classes) to reuse and extend the properties and behaviors of existing classes (base/parent classes) without modifying them, creating hierarchical relationships between classes. It promotes code reusability, enhances modularity, and simplifies the maintenance and extension of software systems. Inheritance is crucial for achieving polymorphism, enabling the creation of specialized classes from existing ones, and facilitating the implementation of the "is-a" relationship.

Its applications include code reuse, the creation of specialized classes, and the implementation of polymorphism, leading to efficient and scalable software development. Its applications range from creating class hierarchies in graphical user interfaces to modeling real-world entities in simulations and games, demonstrating its significance in various domains of software development.

Types of Inheritance:

1. **Single Inheritance:**
   A derived class inherits from only one base class, forming a simple parent-child relationship.
2. **Multiple Inheritance:**
   A derived class inherits from multiple base classes, allowing the derived class to access the features of all the base classes.
3. **Multi-level Inheritance:**
   A derived class serves as the base class for another class, creating a chain of inheritance.
4. **Hierarchical Inheritance:**
   Multiple derived classes inherit from a single base class, forming a hierarchical relationship.
5. **Hybrid Inheritance:**
   It is a combination of multiple and hierarchical inheritance, allowing for complex class hierarchies.

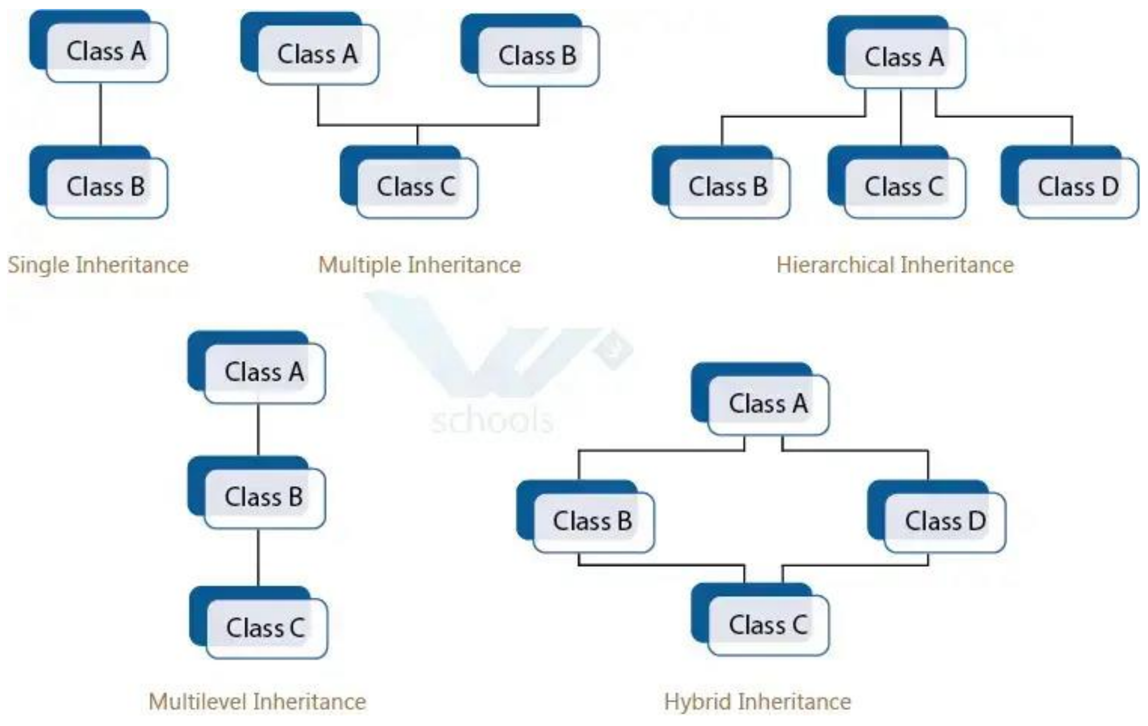


Figure 1: Visual representation of the Types of Inheritance

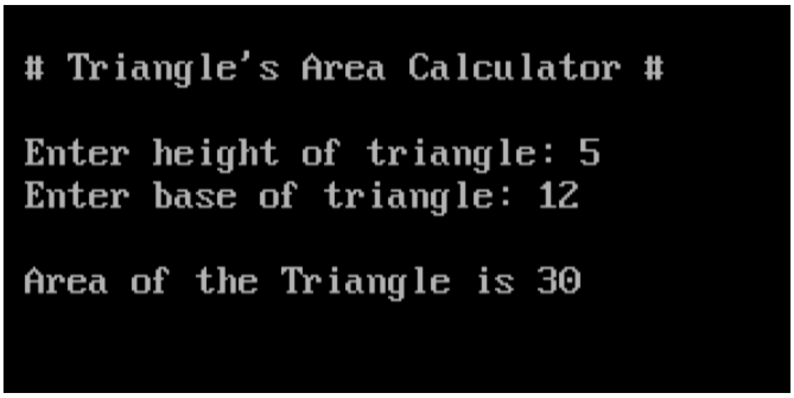
QUESTIONS:

1. Write a program to perform single inheritance to calculate the area of a triangle.

Code:

```
#include <iostream.h>
#include <conio.h>
class Dimensions {
protected:
float b, h;
public:
void Input() {
cout << endl << " Enter height of triangle: ";
cin >> h;
cout << " Enter base of triangle: ";
cin >> b;
}
};
class CalculateArea: public Dimensions {
private:
float tarea;
public:
float Area() {
tarea = 0.5 * h * b;
return tarea;
}
};
void main() {
clrscr();
cout << endl << " # Triangle's Area Calculator #" << endl;
CalculateArea a;
a.Input();
float tarea = a.Area();
cout << endl <<" Area of the Triangle is " << tarea;
getch();
}
```

Output:



```
# Triangle's Area Calculator #

Enter height of triangle: 5
Enter base of triangle: 12

Area of the Triangle is 30
```

Figure 2: Output of the program that calculates the area of a triangle

2. **Write a program to perform multilevel inheritance student details, course details, marks.**

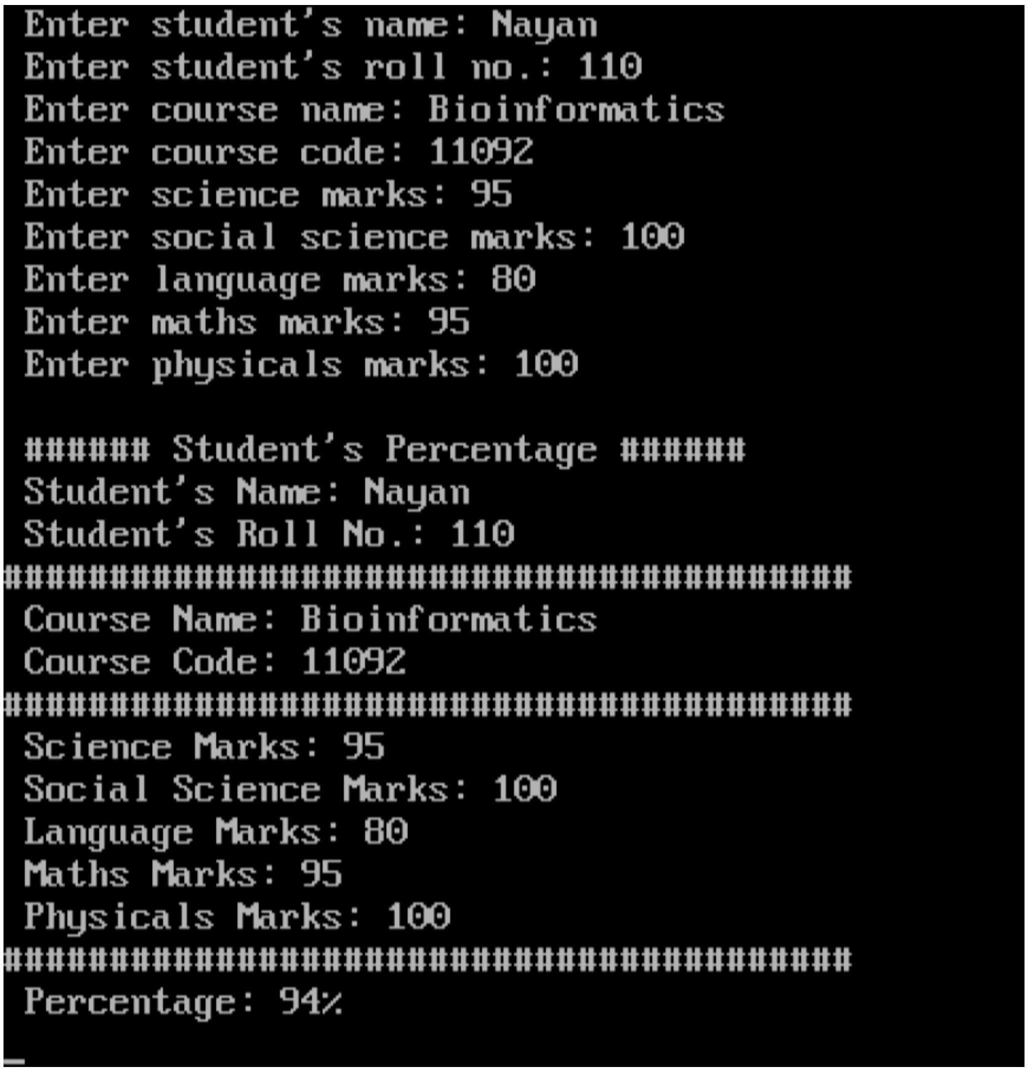
Code:

```
#include <iostream.h>
#include <conio.h>

class SDetails {
protected:
char sname[30];
int rollnum;
public:
void sinput(){
cout << endl << " Enter student's name: ";
cin >> sname;
cout << " Enter student's roll no.: ";
cin >> rollnum;
}
};

class CDetails: public SDetails {
protected:
char cname[15];
char ccode[15];
public:
void cinput(){
cout << " Enter course name: ";
cin >> cname;
cout << " Enter course code: ";
cin >> ccode;
}
```

```
};

class MDetails: public CDetails {
protected:
int scimks, sscimks, langmks, mathmks, phymks;
public:
void minput() {
cout << " Enter science marks: ";
cin >> scimks;
cout << " Enter social science marks: ";
cin >> sscimks;
cout << " Enter language marks: ";
cin >> langmks;
cout << " Enter maths marks: ";
cin >> mathmks;
cout << " Enter physicals marks: ";
cin >> phymks;
}
};

class CalcDetails: public MDetails {
public:
float percent, total;
public:
void Result (){
total = scimks + sscimks + langmks + mathmks + phymks;
percent = (total/500)*100;
}
};

class DDetails: public CalcDetails {
public:
void dispsdetails(){
cout << endl << " ###### Student's Percentage ######" <<endl;
cout << " Student's Name: " << sname << endl;
cout << " Student's Roll No.: " << rollnum << endl;
cout << "########################################" << endl;
cout << " Course Name: " << cname << endl;
cout << " Course Code: " << ccode << endl;
cout << "########################################" << endl;
cout << " Science Marks: " << scimks << endl;
cout << " Social Science Marks: " << sscimks << endl;
cout << " Language Marks: " << langmks << endl;
cout << " Maths Marks: " << mathmks << endl;
cout << " Physicals Marks: " << phymks << endl;
cout << "########################################" << endl;
cout << " Percentage: " << percent << "%" << endl;
}
};
void main () {
```

```
clrscr();
DDetails obj;
obj.sinput();
obj.cinput();
obj.minput();
obj.Result();
obj.dispsdetails();
getch();
}
```

Output:



```
 Enter student's name: Nayan
 Enter student's roll no.: 110
 Enter course name: Bioinformatics
 Enter course code: 11092
 Enter science marks: 95
 Enter social science marks: 100
 Enter language marks: 80
 Enter maths marks: 95
 Enter physicals marks: 100

 ###### Student's Percentage ######
 Student's Name: Nayan
 Student's Roll No.: 110
########################################
 Course Name: Bioinformatics
 Course Code: 11092
########################################
 Science Marks: 95
 Social Science Marks: 100
 Language Marks: 80
 Maths Marks: 95
 Physicals Marks: 100
########################################
 Percentage: 94%

_
```

Figure 3: Output of the program that calculates the percentage based on the input marks and displays the details.

RESULTS:

Using C++ and its concept of inheritance, a single inheritance program was created to calculate the area of triangle along with the multilevel inheritance program to take student's details as an input and calculate their percentage.

CONCLUSION:

The concept of inheritance of C++, that enables programmers to achieve polymorphism, enables them to create specialized classes from the pre-existing ones, have been demonstrated through the creation of the programs.

REFERENCES:

1. Yang. (2022). *CSCI 2170: Computer Science II*. Lectures - CSCI 2170: Computer Science II. https://www.cs.mtsu.edu/~xyang/2170/
2. W3Schools. (n.d.). *C++ Inheritance*. https://www.w3schools.in/cplusplus/inheritance
3. Udacity. (2021, August 16). *C++ Inheritance Explained*. https://www.udacity.com/blog/2021/06/cpp-inheritance-explained.html

Date: 06/11/2023

PRACTICAL: 9
C++: Polymorphism and Virtual and Friend Functions

AIM:
To explore and understand concepts of Polymorphism and Virtual and Friend Functions by creating C++ programs.

INTRODUCTION:
C++ is a versatile and powerful programming language that combines low-level and high-level features, making it suitable for a wide range of applications. Some key features of C++ include efficiency and performance, cross-platform compatibility, effective memory management, and versatility. As a result of its noteworthy features, C++ has been extensively used in memory-intensive programs and other complex systems. Its ability to directly manipulate hardware resources and provide efficient memory management make it an attractive choice for developers working on resource-constrained applications or performance-critical tasks.

Polymorphism:
Polymorphism in C++ refers to the ability of a function or object to take on different forms and behave differently in different scenarios. It allows a single function or object to exhibit different behaviors based on the context in which it is used. For example, a base class method can be overridden by a method in a derived class, allowing the same method name to perform different tasks based on the object that invokes it. Polymorphism is a key feature of object-oriented programming, promoting code reusability and flexibility by enabling different classes to be treated as instances of their common base class, and it is achieved through concepts such as inheritance, overriding, and overloading.
Polymorphism is implemented to reuse code functionality, enhance modularity, and simplify the maintenance and extension of software systems.

Types of Polymorphism:
1. **Compile-time Polymorphism:**
   Compile-time polymorphism, or static polymorphism, is achieved during the compilation of the program. This type of polymorphism is associated with early binding and is implemented through function overloading and operator overloading. It is beneficial for performance-critical applications as the function resolution is done at compile time, resulting in efficient code execution.

2. **Run-time Polymorphism:**
   Runtime polymorphism, or dynamic polymorphism, is achieved at runtime. This type of polymorphism is associated with late binding and is implemented through function overriding and virtual functions. It is essential for achieving flexibility and adaptability in object-oriented systems, as it allows the behavior of the same function or object to change dynamically based on the context in which it is used.

Virtual Function:

The virtual function is a member of the base class that can be redefined in derived classes, enabling runtime polymorphism. Declared with the 'virtual' keyword, these functions allow dynamic behavior based on the object calling them, resolved at runtime through pointers or references of the base class type.

Friend Function:

The friend function is not a class member but can access private and protected class members. Declared with the "friend" keyword inside the class, it enables external functions to manipulate private data. Friend functions offer flexibility for accessing and modifying class members across different classes, without compromising encapsulation, and can be either global or part of another class.

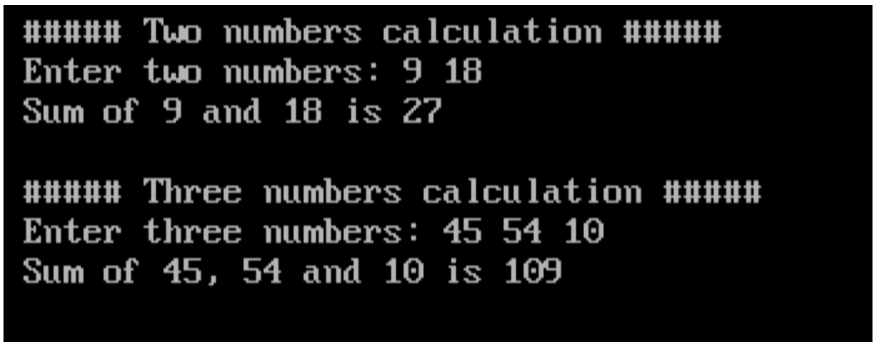
QUESTIONS:

1. **Write a program to perform addition of two and three numbers using method overloading.**

Code:

```
#include<iostream.h>
#include<conio.h>

class Summation {
public:
int sum;
void add(int a, int b) {
sum = a + b;
cout << " Sum of " << a << " and " << b << " is " << sum;
}
void add(int a, int b, int c) {
sum = a + b + c;
cout << " Sum of " <<a<< ", "<< b << " and " << c << " is " << sum;
}
};

void main() {
clrscr();
Summation obj;
cout << endl << " ##### Two numbers calculation #####" << endl;
int a, b, c, d, e;
cout << " Enter two numbers: ";
cin >> a >> b;
obj.add(a,b);
cout << endl << endl << " ##### Three numbers calculation #####" << endl;
cout << " Enter three numbers: ";
cin >> c >> d >> e;
obj.add(c,d,e);
getch();
}
```

Output:



```
##### Two numbers calculation #####
Enter two numbers: 9 18
Sum of 9 and 18 is 27

##### Three numbers calculation #####
Enter three numbers: 45 54 10
Sum of 45, 54 and 10 is 109
```

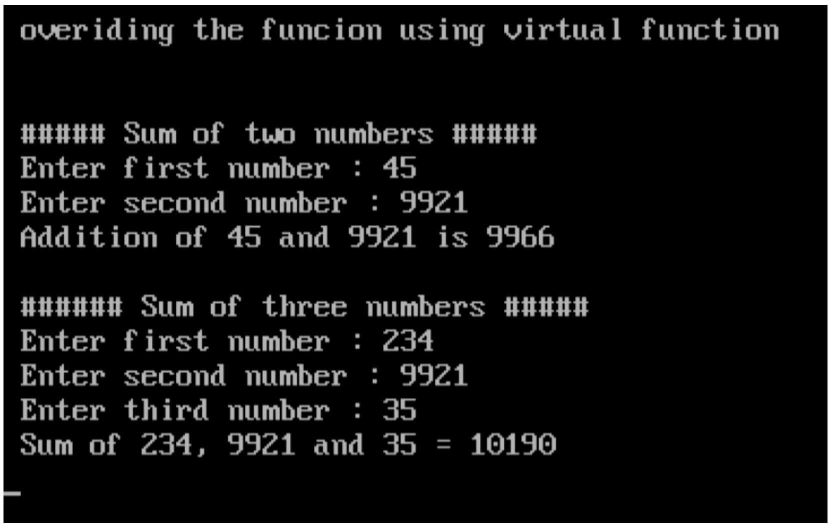
Figure 1: Output of a program that calculates the summation of two numbers as well as three numbers using function overloading

2. **Write a program to demonstrate method overriding using virtual function.**

Code:

```
#include<iostream.h>
#include<conio.h>

class Add1 {
public:
int a, b, sum;
public:
virtual void add() {
cout << endl << " ##### Sum of two numbers ##### " << endl;
cout << " Enter first number : ";
cin >> a;
cout << " Enter second number : ";
cin >> b;
sum = a + b;
cout << " Addition of " << a << " and " << b << " is " << sum << endl << endl;
}
};

class Add2 : public Add1 {
public:
int m, n, o, sum;
void add() {
cout << " ###### Sum of three numbers ##### " << endl;
cout << " Enter first number : ";
cin >> m;
cout << " Enter second number : ";
cin >> n;
cout << " Enter third number : ";
cin >> o;
sum = m + n + o;
cout <<  " Sum of " << m << ", " << n << " and " << o << " = " << sum << endl;
}
};

void main() {
clrscr();
cout << endl << " overiding the funcion using virtual function" << endl << endl;
Add1 *ptr;
Add1 obj1;
Add2 obj2;
ptr = &obj1;
ptr -> add();
ptr = &obj2;
ptr -> add();
getch();
}
```

Output:



```
overiding the funcion using virtual function


##### Sum of two numbers #####
Enter first number : 45
Enter second number : 9921
Addition of 45 and 9921 is 9966

###### Sum of three numbers #####
Enter first number : 234
Enter second number : 9921
Enter third number : 35
Sum of 234, 9921 and 35 = 10190

_
```

Figure 2: Output of a program that calculates the summation of two numbers as well as three numbers using function overriding and virtual function

RESULTS:

Using C++ and its concept of polymorphism, programs were created to calculate the summation of two and three numbers using function overloading as well as function overriding, which involves usage of virtual function.

CONCLUSION:

The concept of polymorphism of C++, that enables programmers to reuse the code functionality as well as simplifies the maintenance and extension of the program, have been demonstrated through the creation of the programs.

REFERENCES:

1. Yang. (2022). *CSCI 2170: Computer Science II*. Lectures - CSCI 2170: Computer Science II. https://www.cs.mtsu.edu/~xyang/2170/
2. GeeksforGeeks. (2023, May 6). *Virtual Function in C*. https://www.geeksforgeeks.org/virtual-function-cpp/
3. Udacity. (2021, August 16). *Beginner's Guide: Understanding Polymorphism in C++*. https://www.udacity.com/blog/2021/07/understanding-polymorphism-in-cpp.html

Date: 06/11/2023

PRACTICAL: 10
C++: Constructor and Destructor

AIM:

To explore and understand concepts of Constructor and Destructor by creating C++ programs.

INTRODUCTION:

C++ is a versatile and powerful programming language that combines low-level and high-level features, making it suitable for a wide range of applications. Some key features of C++ include efficiency and performance, cross-platform compatibility, effective memory management, and versatility. As a result of its noteworthy features, C++ has been extensively used in memory-intensive programs and other complex systems. Its ability to directly manipulate hardware resources and provide efficient memory management make it an attractive choice for developers working on resource-constrained applications or performance-critical tasks.

Constructor:

A constructor in C++ is a special method that is automatically called when an object of a class is created, used to initialize the data members of new objects. It has the same name as the class, does not have any return value, and can take parameters for setting initial values for attributes. Constructors can be defined inside or outside the class and are crucial for initializing objects and setting their initial state.
The constructor is a special member function that is automatically called when an object of a class is created. Constructors are fundamental to object-oriented programming in C++, ensuring that objects are properly initialized and ready for use, and they play a crucial role in maintaining the integrity and consistency of objects within a class.

Syntax:

```
class className {
        accessSpecifier:
                className(parameter1, parameter2, …, parameterN) {
                        //statements;
                }
};

void main() {
        className obj(value1, value2, …, valueN);
}
```

Types of Constructors:

1. **Default Constructor:**
   A constructor with no parameters, automatically called when an object is created without any arguments. It initializes the object's data members to default values.
2. **Parameterized Constructor:**
   A constructor with parameters, allowing the initialization of an object with specific values. It enables the customization of object initialization based on the provided arguments.
3. **Copy Constructor:**
   A constructor that creates a new object as a copy of an existing object. It is used to initialize an object using another object of the same class.

Destructor:

A destructor in C++ is a special member function that is automatically called when the lifetime of an object ends, used to release resources and perform cleanup activities, such as deallocating memory and releasing file handles, when an object is destroyed.
Destructors are necessary to release resources and perform cleanup activities when an object is destroyed, preventing memory leaks and other issues. Their applications include freeing dynamically allocated memory, closing files, releasing network connections, and ensuring proper cleanup in object-oriented systems for robust and efficient programming using C++.

Syntax:
```
class className {
        accessSpecifier:
                ~className(parameter1, parameter2, …, parameterN) {
                        //statements;
                }
};
```

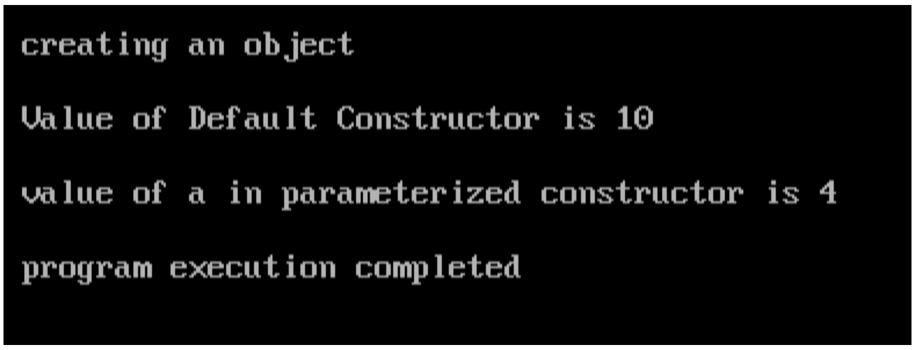
QUESTIONS:

1. **Write a program to demonstrate Constructor and Destructor.**

Code:
```
#include <iostream.h>
#include <conio.h>

class myclass {
int x;
public:
myclass() {
x=10;
cout<< endl <<" Value of Default Constructor is "<<x<<endl;
}
myclass(int a) {
x=a;
cout<< endl <<" value of a in parameterized constructor is "<<x <<endl;
}
~myclass() {
cout<< endl <<" Destructor called"<<endl;
}
};
```

```
void main() {
clrscr();
cout<<" creating an object"<<endl;
myclass obj1;
myclass obj2(4);
cout<< endl <<" program execution completed"<<endl;
getch();
}
```

Output:



```
creating an object

Value of Default Constructor is 10

value of a in parameterized constructor is 4

program execution completed
```

Figure 1: Output of the program that shows working of the constructor and the destructor

RESULTS:

Using C++ and its concept of constructor and destructor, programs were created to show the creation and execution of the constructor and its destruction to save the memory space using the destructor.

CONCLUSION:

The concept of constructor and destructor of C++, that enables programmers to construct and execute certain command using constructor and after execution release the memory space occupied by the constructor using a destructor, have been demonstrated through the creation of the programs.

REFERENCES:

1. Yang. (2022). *CSCI 2170: Computer Science II*. Lectures - CSCI 2170: Computer Science II. https://www.cs.mtsu.edu/~xyang/2170/
2. Great Learning. (2022, September 21). *Constructor in C++ and Types of Constructors*. https://www.mygreatlearning.com/blog/constructor-in-cpp
3. Rai, R. (2022, September 3). *Constructor and Destructor in C++*. Codementor. https://www.codementor.io/@supernerdd7/constructor-and-destructor-in-c-1r8kkogm6j
4. Whitney, T. (2022, February 8). *Constructors (C++)*. Microsoft Learn. Retrieved November 20, 2023, from https://learn.microsoft.com/en-us/cpp/cpp/constructors-cpp