**3**

**A(i)**

Arrays in C++ are a collection of elements, all of the same data type, stored in contiguous memory locations.

How to Deploy:

To declare an array, specify the data type and size of the array. Example: int arr[5]; creates an array of 5 integers.

Arrays can be used to store multiple values in a single variable (e.g., marks of 5 students, names, etc.).

Accessing Elements: Use an index to access array elements (starting from 0). Example: array[0] accesses the first element.

Arrays are useful when dealing with a fixed number of items and when you need to perform operations like sorting or searching.

Example**:** int array [5] = {10, 20, 30, 40, 50}; // Array with 5 integers

for (int i = 0; i < 5; i++) {

cout << array[i] << " "; // Access and print each element

}

**(ii)**

Multi-dimensional arrays are arrays of arrays, allowing for storage in tabular format (e.g., matrices).

How to Deploy:

A 2D array is declared as data\_type array name[rows][columns];. Example: int matrix[3][3];.

Use nested loops to traverse and manipulate multi-dimensional arrays.

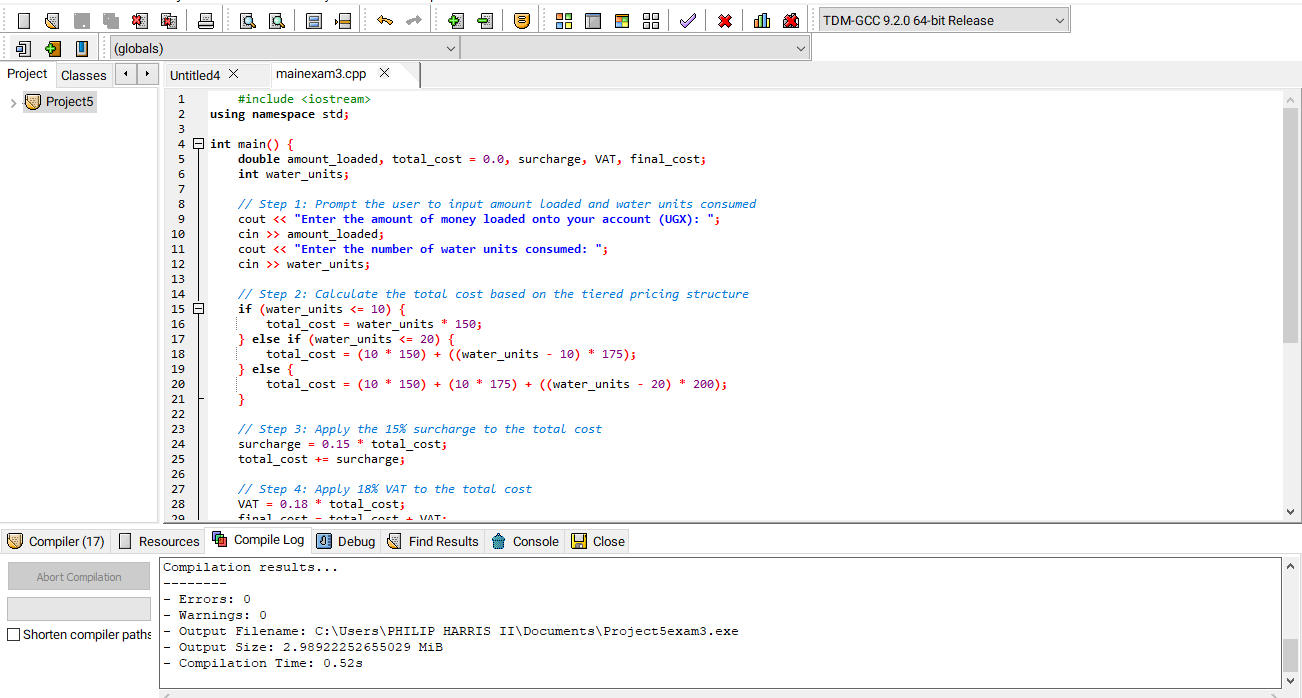
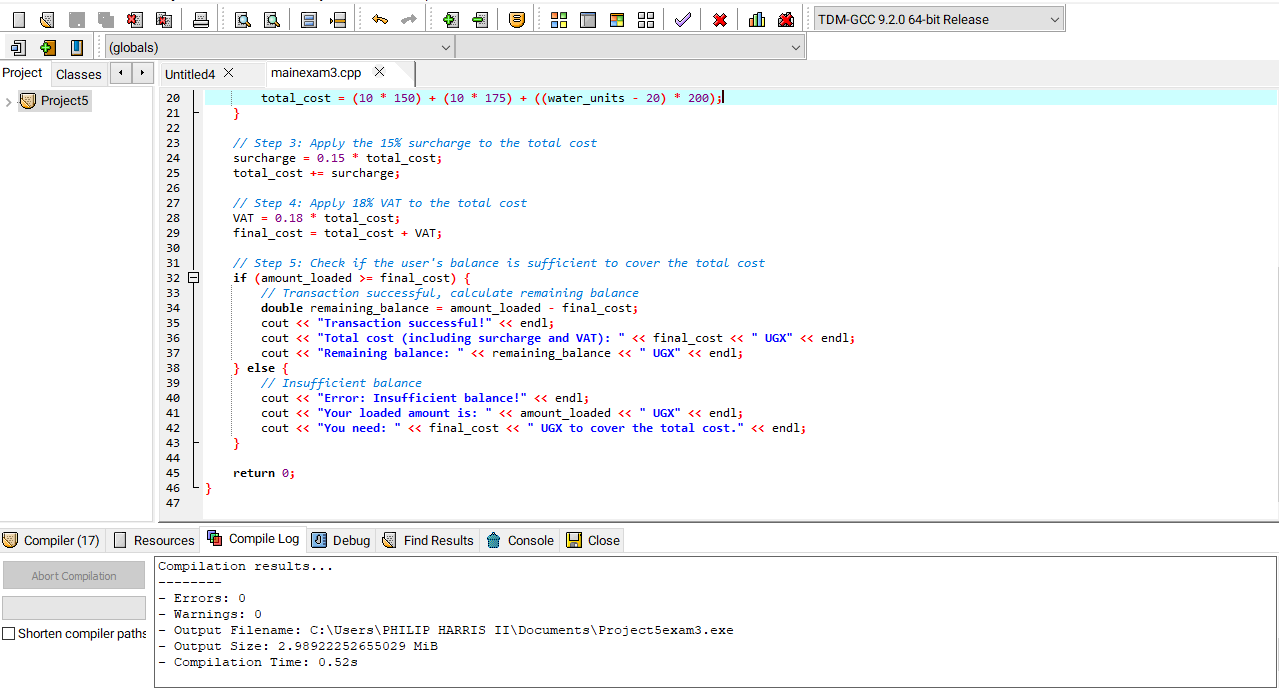
Multi-dimensional arrays are useful for complex data structures like matrices, grids, or tables.

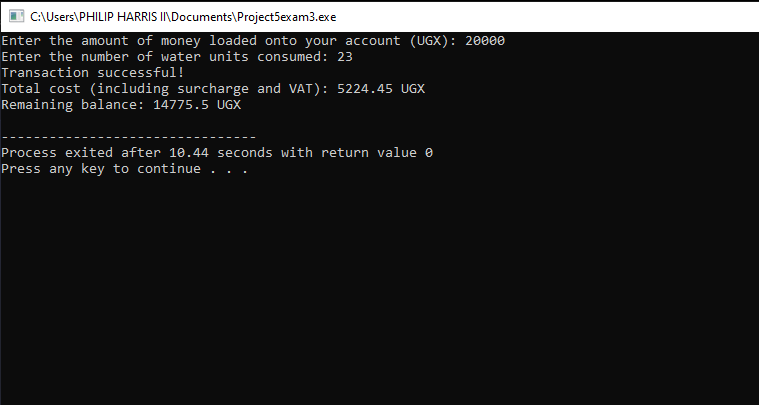
Accessing Elements: Use two indices for 2D arrays. Example: matrix [1][2] accesses the element in the second row and third column.

Example:

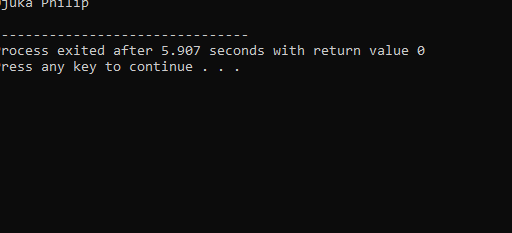
int matrix [2][3] = {{1, 2, 3}, {4, 5, 6}}; // 2x3 matrix

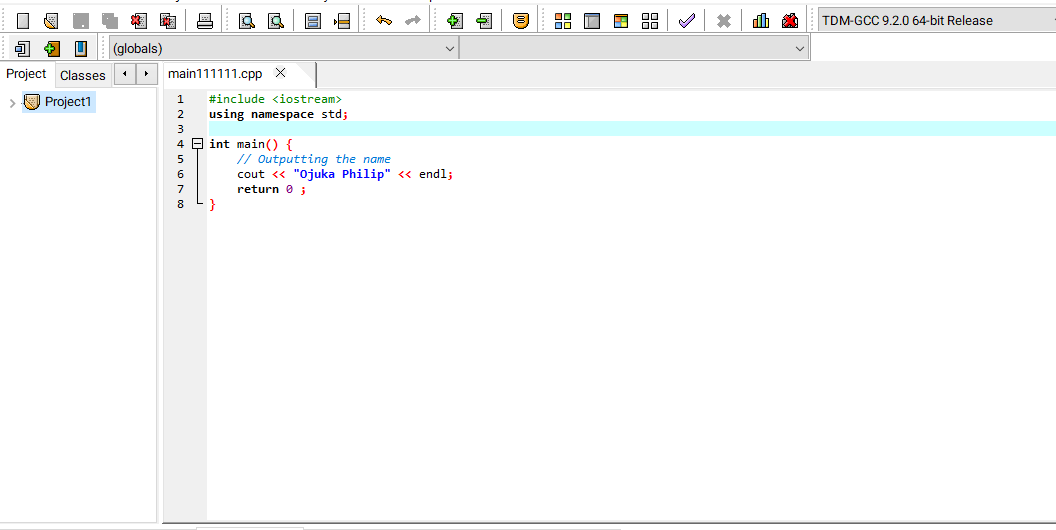
for (int i = 0; i < 2; i++) {



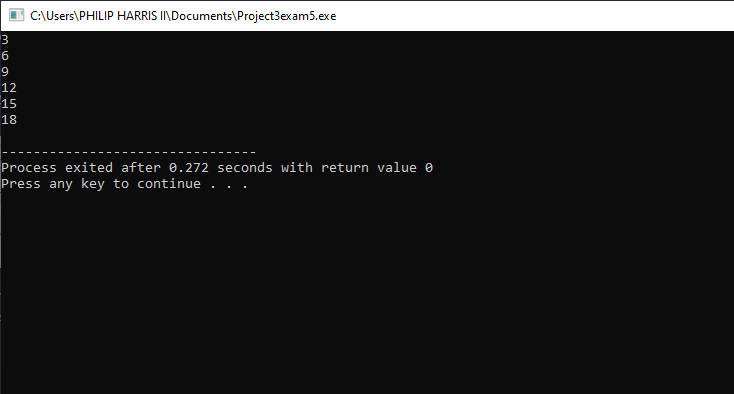
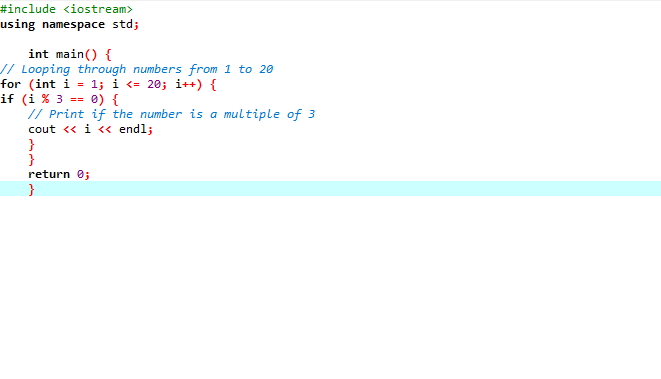


* **5(a)**
* **: Definitions in C++ Programming Language**
* **Data type:**
  + A data type in C++ defines the type of data a variable can hold. Examples include:
    - int for integers,
    - float for floating-point numbers,
    - char for characters, and
    - bool for boolean values.
  + It specifies the range of values that the variable can store and the operations that can be performed on it.
* **Variable:**
  + A variable in C++ is a named storage location that holds data that can be modified during program execution. It is declared with a specific data type (e.g., int x, float y).
  + Variables must be declared before they can be used.
* **Algorithm:**
  + An algorithm is a step-by-step procedure or set of rules used to perform a specific task or solve a particular problem. In C++ programming, algorithms are often implemented through sequences of code instructions.
* **Function:**
  + A function in C++ is a block of code that performs a specific task. Functions are reusable pieces of code that can be called by their name, passing data (arguments) to them, and can return a value. They allow modularity and code reusability.
  + Example: int add(int a, int b) { return a + b; } is a function that adds two numbers.
* **5b(i)**
  + #include <iostream>
* using namespace std;
* int main() {
* // Outputting the name
* cout << “Ojuka Philip" << endl;
* return 0;
* }





* **ii**
* #include <iostream>
* using namespace std;
* int main() {
* // Looping through numbers from 1 to 20
* for (int i = 1; i <= 20; i++) {
* if (i % 3 == 0) {
  + // Print if the number is a multiple of 3
  + cout << i << endl;
* }
* }
* return 0;
* }

****

* **5.C(i)**
* **Single-line comments**:
* **Why**: To explain or annotate specific parts of code in a concise manner.
* **How**: Use two forward slashes (//). Everything after // on the same line is treated as a comment and ignored by the compiler.
* **When**: Use when you want to explain a small part of your code (e.g., a single line or variable).
* For example
* // This is a single-line comment
* int x = 5; // Declaring a variable
* ii
* **Multi-line comments**:
* **Why**: To explain larger sections of code or provide more detailed descriptions.
* **How**: Use /\* to start and \*/ to end the comment block. Everything between is ignored by the compiler.
* **When**: Use for providing detailed explanations or commenting out multiple lines of code.

6

#include <iostream>

using namespace std;

double calculate weight (double mass) {

return mass \* 9.8;

}

int main () {

double mass;

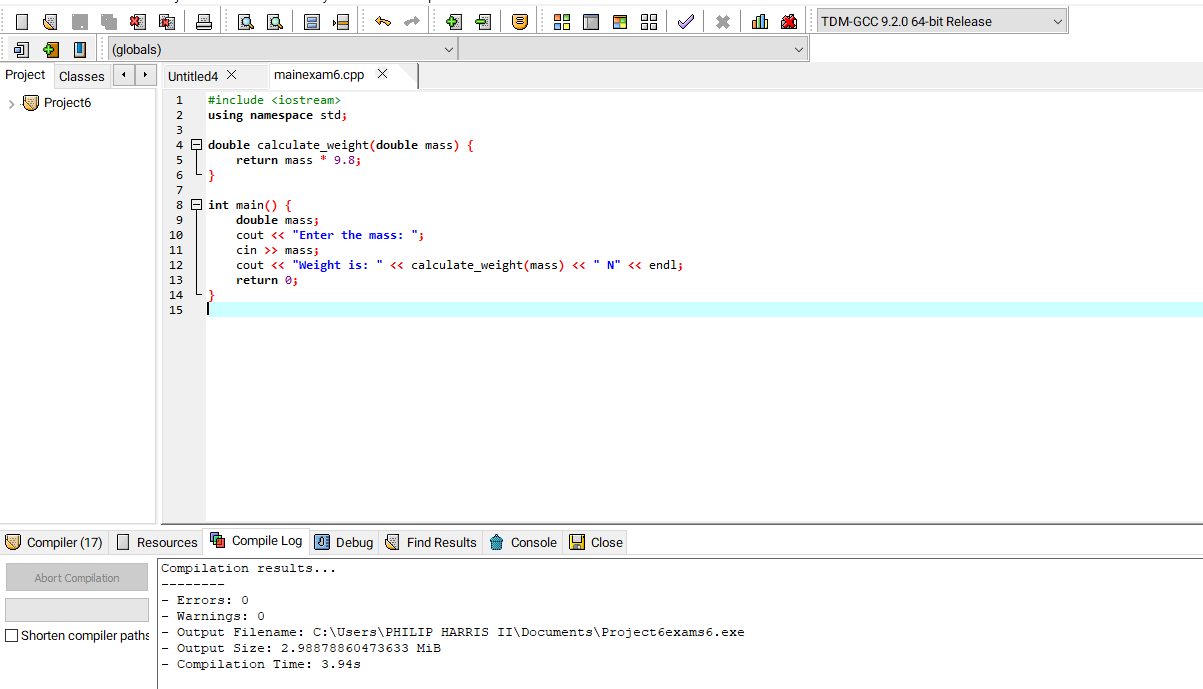
cout << "Enter the mass: ";

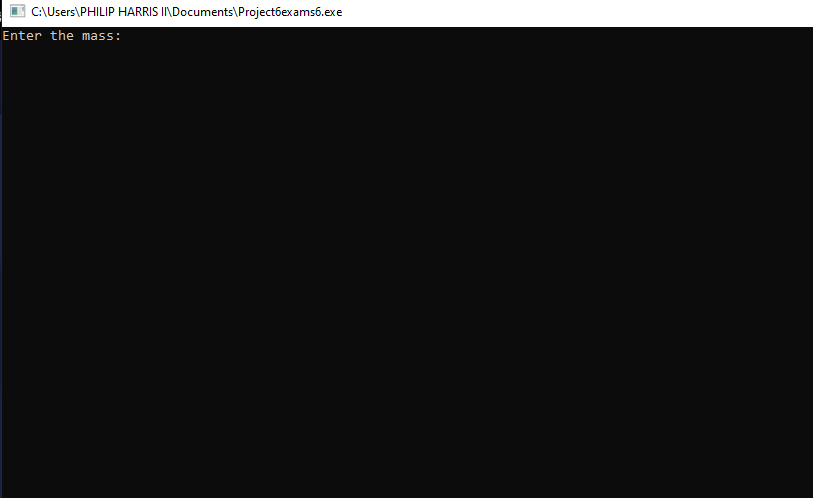
cin >> mass;

cout << "Weight is: " << calculate weight(mass) << " N" << endl;

return 0;

}





Difference between pointer and reference in C++ (5 Marks)

• Pointer: A variable that holds the memory address of another variable.

Example: int\* ptr = &x;.

• Reference: An alias for another variable. Example: int &ref = x;.

c) Evaluate the code (15 Marks)

The function prints out values of the array after multiplying each by 100.

Returns and prints the value of y (16) after passing through the function foo.

1

#include <iostream>

using namespace std;

bool is\_leap\_year(int year) {

if (year % 400 == 0)

return true;

if (year % 100 == 0)

return false;

if (year % 4 == 0)

return true;

return false;

}

int main() {

int year;

cout << "Enter a year: ";

cin >> year;

if (is\_leap\_year(year)) {

cout << year << " is a leap year." << endl;

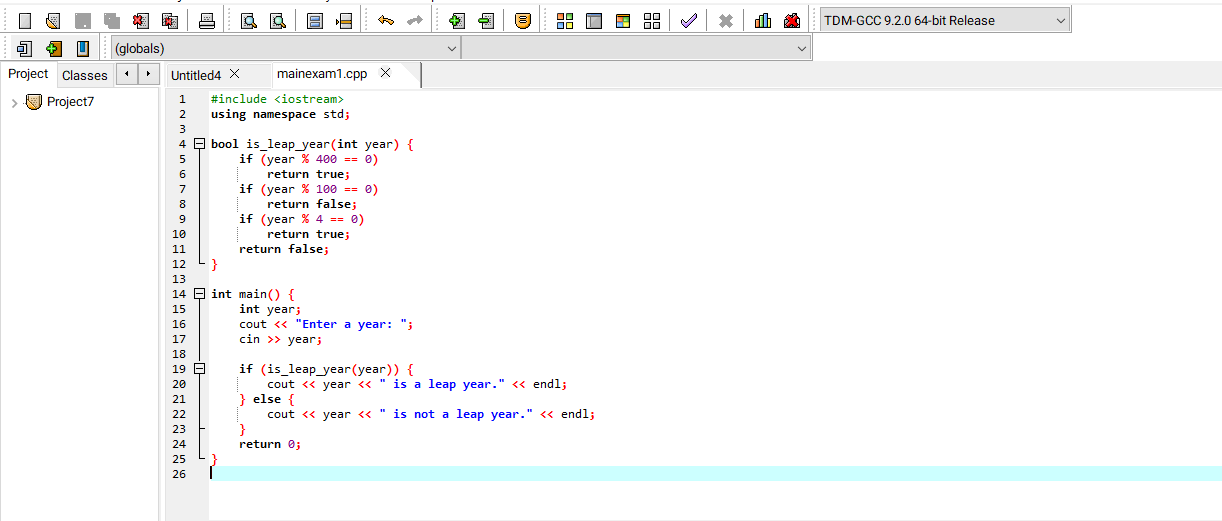
} else {

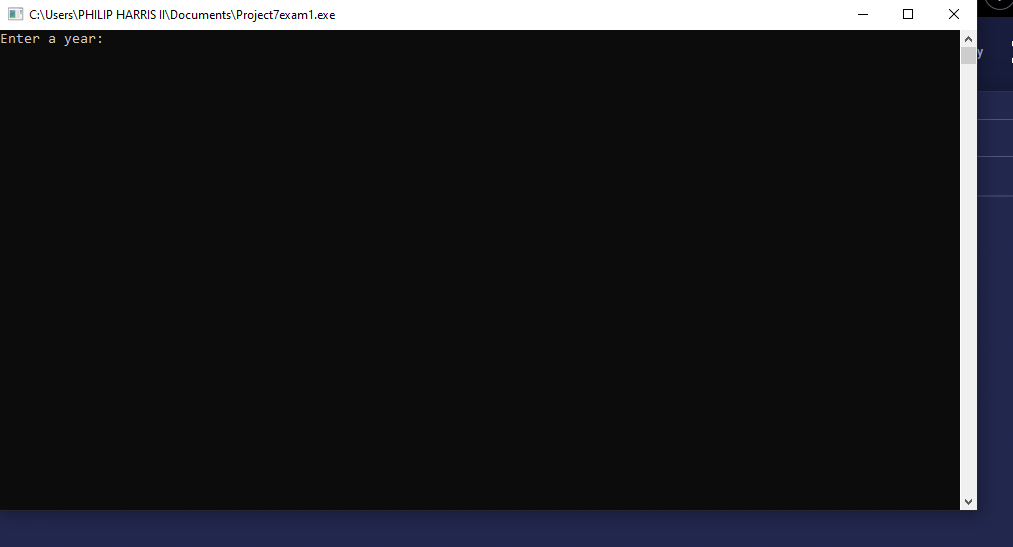
cout << year << " is not a leap year." << endl;

}

return 0;

}





B

#include <iostream>

using namespace std;

int main() {

double sum = 0;

for (int i = 1, j = 3; i <= 95 && j <= 97; i += 2, j += 2) {

sum += (double)i / j;

}

cout << "Sum of the series: " << sum << endl;

return 0;

}

