National University of Sciences and Technology (NUST)
School of Electrical Engineering and Computer Science

Department Of Electrical Engineering and Computer Sciences

Instructor: Mehreen Tahir **Date:** December 4, 2023

Lab Engineer: Mehwish Kiran **Time:** 10:00am – 12:50pm

CS 212: Object Oriented Programming

Lab 11: Templates

Information	Description
Name:	Irfa Farooq
CMS ID:	412564
Class:	BEE-14
Section:	D
Tenure:	Fall 2023

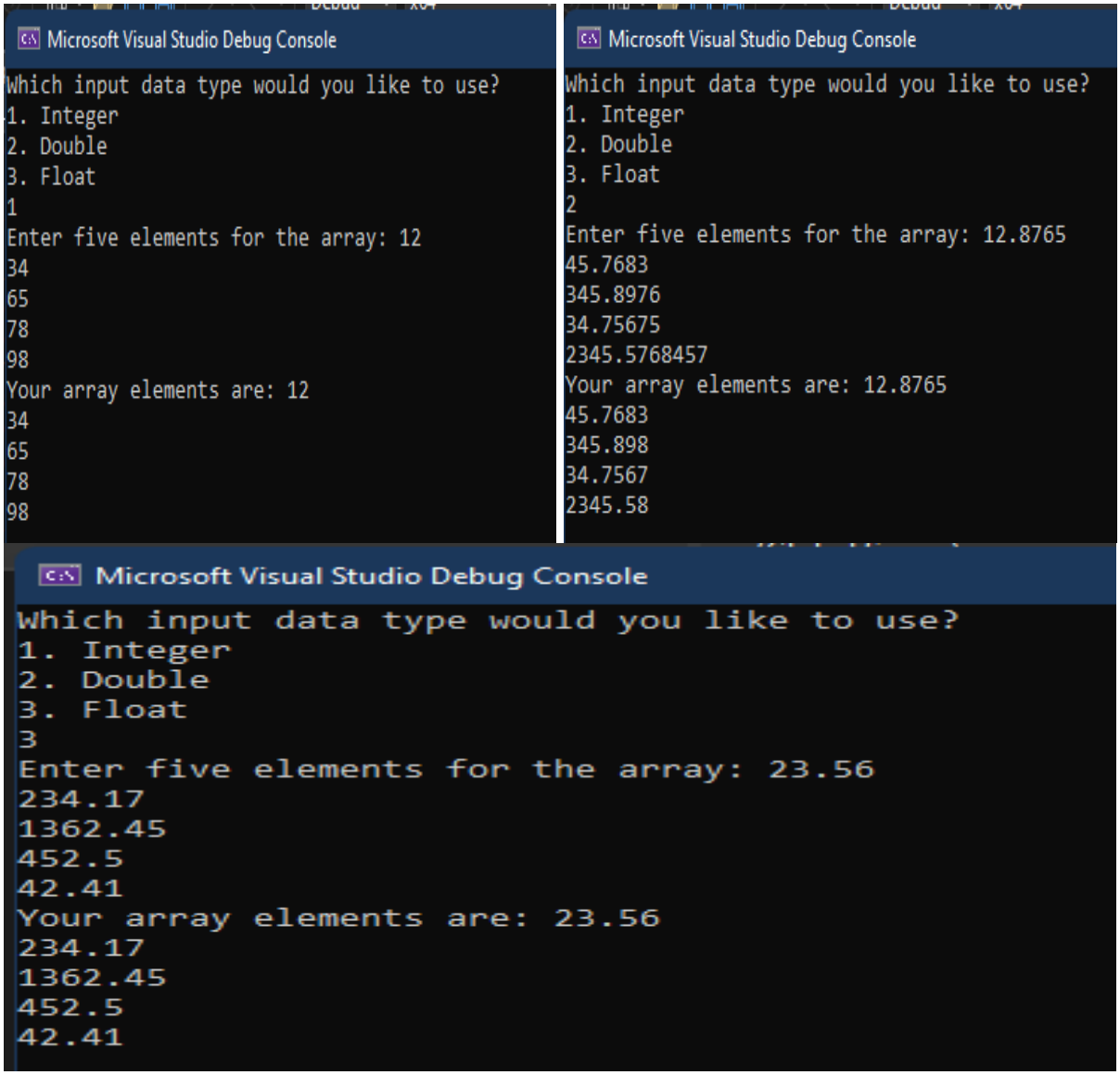
Task 1: Create a C++ Template Function named store so that it accepts an array of 5 elements. A Template Function created will store an array of five elements of any given primitive data type. Define a function to print the contents of the array.

Code:

```
#include <iostream>

template <class Y>
void print(Y* a) {
	std::cout << "Your array elements are: ";
	for (int i = 0; i < 5; i++) {
		std::cout << a[i] << std::endl;
	}
}

template <class T>
void store(T* array1, int size) {
	T array2[5];
	for (int i = 0; i < size; i++) {
		array2[i] = array1[i];
	}
	print(array2);
}

int main() {

	int choice;
	std::cout << "Which input data type would you like to use?" << std::endl;
	std::cout << "1. Integer" << std::endl;
	std::cout << "2. Double" << std::endl;
	std::cout << "3. Float" << std::endl;
	std::cin >> choice;

	switch (choice) {

	case 1: {
		std::cout << "Enter five elements for the array: ";
		int array[5];
		for (int i = 0; i < 5; i++) {
			std::cin >> array[i];
		}
		store(array, 5);
		return 0;
	}

	case 2: {
	std::cout << "Enter five elements for the array: ";
		double array[5];
		for (int i = 0; i < 5; i++) {
			std::cin >> array[i];
		}
		store(array, 5);
		return 0;
	}
```

```
        case 3: {
                std::cout << "Enter five elements for the array: ";
                float array[5];
                for (int i = 0; i < 5; i++) {
                        std::cin >> array[i];
                }
                store(array, 5);
                return 0;
        }
        default: {
                std::cout << "Invalid Input!" << std::endl;
                std::cout << "Input again: ";
        }
        }
        std::cout << "Thank You!" << std::endl;
}
```

Output Screenshots



Task 2: Write and implement a template class of Matrix.
Each Matrix is a two-dimensional array with a number of columns and rows.

Code:

```
#include <iostream>

template <class T>
class Matrix {
public:
	T arr[50][50];
	int rows;
	int columns;

public:
	Matrix() {
		rows = 50;
		columns = 50;
		for (int i = 0; i < rows; i++) {
			for (int j = 0; j < columns; j++) {
				arr[i][j] = 0;
			}
		}
	}
	void setElements(int r, int c, T value) {
		rows = r;
		columns = c;
		arr[r][c] = value;
	}
	void printMatrix(int r, int c) {
		std::cout << "Your matrix is : " << std::endl;
		for (int i = 0; i < r; i++) {
			for (int j = 0; j < c; j++) {
				std::cout << arr[i][j] << " ";
			}
			std::cout << std::endl;
		}
	}
	void addMatrix(int r, int c) {
		std::cout << "Adding your matrix to itself. New matrix is: " <<
std::endl;
		T temp[r][c];
		for (int i = 0; i < r; i++) {
			for (int j = 0; j < c; j++) {
				temp[i][j] = arr[i][j] + arr[i][j];
				std::cout << temp[i][j] << "  ";
			}
			std::cout << std::endl;
		}
	}
	void subtractMatrix(int r, int c) {
		std::cout << "Subtracting your matrix from itself. New matrix is: "
<< std::endl;
		T temp[r][c];
		for (int i = 0; i < r; i++) {
			for (int j = 0; j < c; j++) {
				temp[i][j] = arr[i][j] - arr[i][j];
```

```
                        std::cout << temp[i][j] << "  ";
                    }
                    std::cout << std::endl;
                }
        }

        void MultiplyMatrix(int r, int c) {
                if (r == c) {
                    std::cout << "Multiplying your matrix with itself gives: " <<
std::endl;
                    T temp[r][c];
                    for (int i = 0; i < r; i++) {
                        for (int j = 0; j < c; j++) {
                            for (int k = 0; k < r; k++) {
                                temp[i][j] += arr[i][k] * arr[k][j];
                            }
                            std::cout << temp[i][j] << "  ";
                        }
                        std::cout << std::endl;
                    }
                }
                else {
                    std::cout << "Multiplication is not possible." << std::endl;
                }
        }
};


int main() {
        int row = 0, coln = 0;
        std::cout << "Enter number of rows: ";
        std::cin >> row;
        std::cout << "Enter number of columns: ";
        std::cin >> coln;
        while (1) {
                std::cout << "Which type of matrix do you want to make: " <<
std::endl;
                std::cout << "1. Double" << std::endl;
                std::cout << "2. Float" << std::endl;
                std::cout << "3. Int" << std::endl;
                std::cout << "4. Exit" << std::endl;
                int choice;
                std::cout << "Your choice: ";
                std::cin >> choice;
                switch (choice) {
                case 1: {
                    Matrix <double> M;
                    double value;
                    std::cout << "Enter elements of the matrix: " << std::endl;
                    for (int i = 0; i < row; i++) {
                        for (int j = 0; j < coln; j++) {
                            std::cin >> value;
                            M.setElements(i, j, value);
                        }
                    }
                    M.printMatrix(row, coln);
                    M.addMatrix(row, coln);
                    M.subtractMatrix(row, coln);
                    M.MultiplyMatrix(row, coln);
                    return 0;
                }
```

```
		case 2: {
			Matrix <float> M;
			float value;
			std::cout << "Enter elements of the matrix: " << std::endl;
			for (int i = 0; i < row; i++) {
				for (int j = 0; j < coln; j++) {
					std::cin >> value;
					M.setElements(i, j, value);
				}
			}
			M.printMatrix(row, coln);
			M.addMatrix(row, coln);
			M.subtractMatrix(row, coln);
			M.MultiplyMatrix(row, coln);
			return 0;
		}
		case 3: {
			Matrix <int> M;
			int value;
			std::cout << "Enter elements of the matrix: " << std::endl;
			for (int i = 0; i < row; i++) {
				for (int j = 0; j < coln; j++) {
					std::cin >> value;
					M.setElements(i, j, value);
				}
			}
			M.printMatrix(row, coln);
			M.addMatrix(row, coln);
			M.subtractMatrix(row, coln);
			M.MultiplyMatrix(row, coln);
			return 0;
		}
		case 4: {
			return 0;
		}
		default: {
			std::cout << "Invalid Input!" << std::endl;
			std::cout << "Input again: ";
		}
		}
	}
	std::cout << "Thank You!" << std::endl;
	return 0;
}
```

Output Screenshots

```
Enter number of rows: 2
Enter number of columns: 2
Which type of matrix do you want to make:
1. Double
2. Float
3. Int
4. Exit
Your choice: 1
Enter elements of the matrix: 243.5245
524.254
7826.7542
86.62
Your matrix is :
243.524 524.254
7826.75 86.62
Adding your matrix to itself. New matrix is:
487.049  1048.51
15653.5  173.24
Subtracting your matrix from itself. New matrix is:
0  0
0  0
Multiplying your matrix with itself gives:
4.16251e+06  173080
2.58396e+06  4.11071e+06
```

```
Enter number of rows: 2
Enter number of columns: 3
Which type of matrix do you want to make:
1. Double
2. Float
3. Int
4. Exit
Your choice: 2
Enter elements of the matrix:
5.24
2867.5
524.25
52.26
52.52
52.32
Your matrix is :
5.24 2867.5 524.25
52.26 52.52 52.32
Adding your matrix to itself. New matrix is:
10.48  5735  1048.5
104.52  105.04  104.64
Subtracting your matrix from itself. New matrix is:
0  0  0
0  0  0
```

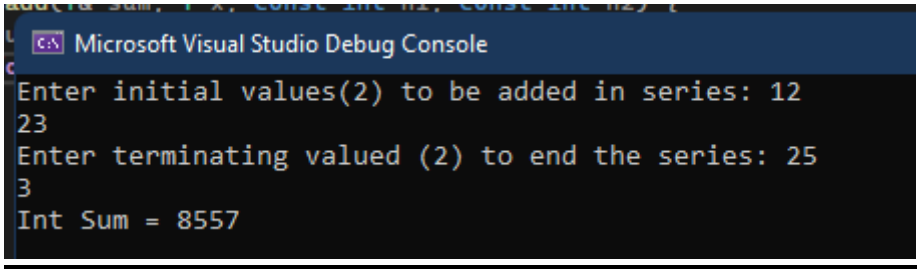
Task 3:Create the C++ Template Function named add so that it has four parameters sum, x, and n1 and n2. The first two parameters will have the type represented by the function template type parameter T. n1 and n2 will always be int. The return type is void. All parameters are passed by value except for sum which is passed by reference.

A Template Function created from Add will compute...
sum = 1 + x + 2x + 3x + ... + n1x + n2x

Code:

```
#include <iostream>
template <class T>
void add(T& sum, T x, const int n1, const int n2) {
	sum = 1;
	for (int i = 1; i < n1; i++) {
		sum += x * i;
		sum += n2 * x;
	}
}


int main() {
	int n1 = 0;
	int n2 = 0;
	int sum1, x1;
	std::cout << "Enter initial values(2) to be added in series: ";
	std::cin >> sum1 >> x1;
	std::cout << "Enter terminating valued (2) to end the series: ";
	std::cin >> n1 >> n2;
	add(sum1, x1, n1, n2);
	std::cout << "Int Sum = " << sum1 << std::endl;
}
```

Output Screenshots



Conclusion:

In this lab, we were able to understand the use of templates and how to make template functions and template classes. We also got to verify that templates make our codes easier and prevents overloading functions. In addition, it also reduces the risk of errors to a maximum extent.