**Q1: Write a C++ program to overload a function add() to handle:**

* **Two integers**
* **Two floats**
* **One integer and one float**

**Answer:**#include <iostream>

using namespace std;

int add(int a, int b) {

return a + b;

}

float add(float a, float b) {

return a + b;

}

float add(int a, float b) {

return (float)a + b;

}

int main() {

int n1, n2;

cout << "Enter two numbers: ";

cin >> n1 >> n2;

cout << "Sum of two integers: " << add(n1, n2) << endl;

float f1, f2;

cout << "Enter two float numbers: ";

cin >> f1 >> f2;

cout << "Sum of two floats: " << add(f1, f2) << endl;

int N;

float F;

cout << "Enter an integer and a float number: ";

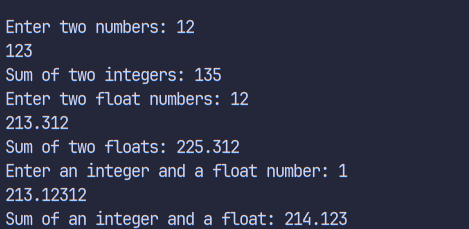
cin >> N >> F;

cout << "Sum of an integer and a float: " << add(N, F) << endl;

return 0;

}

**Output:**



**Q2: Write an inline function in C++ to calculate the square of a number and demonstrate it with at least two function calls.**

**Answer:**  
#include <iostream>

using namespace std;

inline int square(int num) {

return num \* num;

}

int main() {

int num1, num2;

cout << "Enter two numbers: ";

cin >> num1 >> num2;

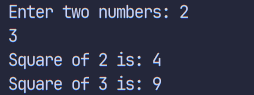
cout << "Square of " << num1 << " is: " << square(num1) << endl;

cout << "Square of " << num2 << " is: " << square(num2) << endl;

return 0;

}

**Output:**



**Q3: Write a program using a function with default arguments for calculating total price. The function should take the item price and quantity, with quantity defaulting to 1.**

**Answer:**  
#include <iostream>

using namespace std;

float calculateTotalPrice(float itemPrice, int quantity = 1) {

return itemPrice \* quantity;

}

int main() {

float itemPrice;

int quantity;

cout << "Enter the price of the item: ";

cin >> itemPrice;

cout << "Enter the quantity of the item: ";

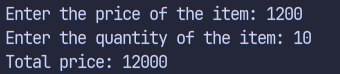
cin >> quantity;

cout << "Total price: " << calculateTotalPrice(itemPrice, quantity) << endl;

return 0;

}

**Output:**



**Q4: Write a C++ program to swap two numbers using pass-by-reference.**

**Answer:**#include <iostream>

using namespace std;

void swap(int &a, int &b) {

int temp = a;

a = b;

b = temp;

}

int main() {

int num1 = 10, num2 = 20;

cout << "Before swapping - num1: " << num1 << ", num2: " << num2 << endl;

swap(num1, num2);

cout << "After swapping - num1: " << num1 << ", num2: " << num2 << endl;

return 0;

}

**Output:**



**Q5: Create a function that returns a reference to an element in an array and modifies it.**

**Answer:**#include <iostream>

using namespace std;

int& mod\_array(int arr[], int index, int value) {

arr[index] = value;

return arr[index];

}

int main() {

int arr[5] = {1, 2, 3, 4, 5}, index, value;

cout << "Enter the index of the element to modify: ";

cin >> index;

cout << "Enter the value to modify the element to: ";

cin >> value;

cout << "\nBefore modification: ";

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

cout << arr[i] << " ";

}

int mod\_element = mod\_array(arr, index, value);

cout << "\nAfter modification: ";

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

cout << arr[i] << " ";

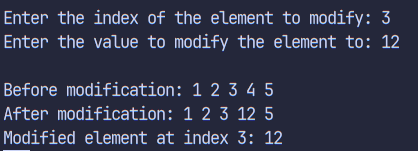
}

cout << "\nModified element at index " << index << ": " << mod\_element << endl;

return 0;

}

**Output:**



**Q6: Write a program to input 5 integers in an array and print their squares using a pointer.**

**Answer:**#include <iostream>

using namespace std;

int main() {

int arr[5];

int squares[5];

cout << "Enter 5 integers:" << endl;

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

cin >> arr[i];

}

int \*ptr = arr;

cout << "Squares of the integers:" << endl;

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

squares[i] = \*(ptr + i) \* \*(ptr + i);

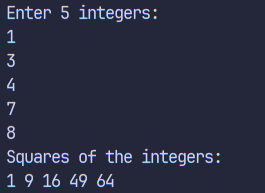
cout << squares[i] << " ";

}

return 0;

}

**Output:**



**Q7: Define a structure Student with data members roll, name, and marks. Input and display details of 3 students.**

**Answer:**#include <iostream>

using namespace std;

struct Student {

string name;

int rollNo;

int marks;

};

int main() {

Student student1;

cout << "Enter student's name: ";

cin >> student1.name;

cout << "Enter roll number: ";

cin >> student1.rollNo;

cout << "Enter marks: ";

cin >> student1.marks;

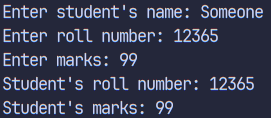
cout << "Student's roll number: " << student1.rollNo << endl;

cout << "Student's marks: " << student1.marks << endl;

return 0;

}

**Output:**



**Q8: Write a C++ program to demonstrate the difference between structure and union by declaring the same data members and showing memory usage.**

**Answer:**#include <iostream>

using namespace std;

struct Structure {

int num=10;

char a='A';

float num2=10.5;

};

union ExampleUni {

int num3;

char b;

float num4;

};

int main() {

Structure s;

ExampleUni u;

cout << "Structure data: " << sizeof(s);

cout << "Union data: " << sizeof(u.num3);

return 0;

}

**Output:**



**Q9: Create an enum for days of the week. Display a message depending on the selected day.**

**Answer:**#include <iostream>

using namespace std;

enum Day{Sunday, Monday, Tuesday, Wednesday, Thursday, Friday, Saturday};

int main(){

int day;

cout << "Enter a number corresponding to a day (1-7): ";

cin >> day;

day-=1;

switch(day) {

case Sunday:

cout << "Sunday";

break;

case Monday:

cout << "Monday";

break;

case Tuesday:

cout << "Tuesday";

break;

case Wednesday:

cout << "Wednesday";

break;

case Thursday:

cout << "Thursday";

break;

case Friday:

cout << "Friday";

break;

case Saturday:

cout << "Saturday";

break;

default:

cout << "Invalid day";

}

return 0;

}

**Output:**



**Q10: Write a C++ program to allocate memory for an array of integers using new, input values, calculate their sum, and free the memory using delete.**

**Answer:**#include <iostream>

using namespace std;

int main() {

int size;

int sum = 0;

cout << "Enter the size of the array: ";

cin >> size;

int \*arr = new int[size];

cout << "Enter " << size << " integers:" << endl;

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

cin >> arr[i];

sum += arr[i];

}

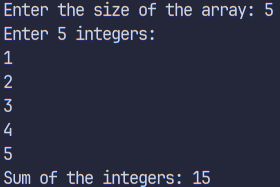
cout << "Sum of the integers: " << sum << endl;

delete[] arr;

return 0;

}

**Output:**



**Discussion**

This lab introduced various fundamental concepts in C++ through hands-on programs. Key topics covered included function overloading, inline functions, default arguments, pass-by-reference, structures, unions, enums, and dynamic memory allocation. Each program demonstrated how these features improve code modularity, efficiency, and maintainability. The use of pointers and references enhanced understanding of memory management and data manipulation.

### **Conclusion**

The lab successfully reinforced essential C++ programming concepts. Through practical implementation, students learned how to write more efficient and flexible code using advanced features like references, overloading, and dynamic memory. These skills are foundational for further learning in object-oriented and system-level programming.