

HIMALAYA COLLEGE OF ENGINEERING

Advanced C++ Programming Lab Report

Lab 2: Basics of C++ programming (II)

Prepared By: Aditya Man Shrestha

Subject: Object Oriented Programming (OOP)

Program: Bachelors of Electronics and Computer Engineering

Institution: Himalaya College of Engineering

Date: June 8, 2025

Objectives:

- To understand and implement function overloading in C++.
- To utilize inline functions for efficiency.
- To work with default arguments in functions.
- To implement call-by-reference.
- To manipulate arrays and pointers.
- To differentiate between structures and unions.
- To understand enumeration types in C++.
- To dynamically allocate and deallocate memory.

Tools and Libraries Used:

- Programming Language: C++
- IDE: Code::Blocks
- Libraries: include <iostream>, include <string>

Theory:

Function Overloading

Function overloading is a feature in C++ that allows multiple functions to share the same name as long as they differ in their **parameter list** (also known as the function **signature**). The compiler uses the number and types of arguments provided in a function call to determine which version of the overloaded function to invoke.

Example:

```
int add(int a, int b);  
float add(float x, float y);  
float add(int a, float b);
```

Inline Functions

An inline function is a function for which the compiler attempts to insert the function's body directly at the point of each function call, instead of performing a regular call (which involves pushing to the stack, jumping to the function location, etc.).

This can help reduce function call overhead, especially for small, frequently called functions.

Example:

```
inline int add(int a, int b) {  
    return a + b;  
}
```

Default Arguments

Default arguments allow a function to be called with fewer arguments than it is declared to take. If some arguments are not provided in the function call, the default values specified in the function declaration are used.

Example:

```
void func(int x, int y = 0, int z = 1);
```

Call-by-Reference

When a function parameter is declared as a reference (using the & symbol), it does not create a copy of the argument. Instead, it refers to the original variable, allowing the function to modify the actual value passed to it.

Example:

i) Call by reference

```
void swapNumbers(int &a, int &b);
```

ii) Return reference

```
int& getElement(int arr[], int index);
```

Pointers and Arrays

Pointers can access and manipulate array elements directly using pointer arithmetic.

Example:

```
int* ptr = arr;
```

Structures vs Unions

Structure: Allocates separate memory for each member.

Example:

```
struct StdStructure {  
    int roll;  
    string name;  
    float marks;  
};
```

Union: Allocates shared memory, and only one member can be used at a time.

Example:

```
union StdUnion {  
    int roll;  
    string name;  
    float marks;  
};
```

Enumerations

Enums allow defining a set of named integral constants to make code more readable and maintainable.

Example:

```
enum Day { Sunday, Monday, ... }
```

Dynamic Memory Allocation

Using new and delete operators in C++, memory can be allocated and deallocated at runtime.

Example:

```
int* arr = new int[n];  
delete[] arr;
```

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

```
#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 a + b; }  
  
int main() {  
    int a, b;  
    float x, y;  
  
    cout << "Enter 2 integer numbers: ";  
    cin >> a >> b;  
  
    cout << "Sum of integers is: " << add(a, b) << endl;  
  
    cout << "Enter 2 float numbers: ";  
    cin >> x >> y;  
  
    cout << "Sum of floats is: " << add(x, y) << endl;  
  
    cout << "Enter 1 integer and 1 float number: ";  
    cin >> a >> y;  
  
    cout << "Sum of integer and float is: " << add(a, y) << endl;  
  
    return 0;  
}
```

OUTPUT:

```
C:\work\OOP\lab2\two.exe X + v
Enter 2 integer numbers: 12
13
Sum of integers is: 25
Enter 2 float numbers: 13.5
23.34
Sum of floats is: 36.84
Enter 1 integer and 1 float number: 12
234.5344
Sum of integer and float is: 246.534

Process returned 0 (0x0)   execution time : 52.211 s
Press any key to continue.
```

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

```
#include<iostream>
using namespace std;
inline int square(int n){
return n*n;
}
int main(){
int num1,num2;
cout<<"Enter numbers you want sum of";
cin>>num1>>num2;
cout<<"The square of "<<num1<<" is"<<square(num1)<<endl;
cout<<"The square of "<<num2<<" is "<<square(num2)<<endl;
return 0;
}
```

OUTPUT

```
C:\work\OOP\lab2\two.exe X + v
Enter numbers you want sum of 12
564
The square of 12 is144
The square of 564 is 318096

Process returned 0 (0x0)   execution time : 7.864 s
Press any key to continue.
```

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.

```
#include<iostream>
using namespace std;
float calculateTotal(float price,int quantity = 1){
return price*quantity;
}
int main(){
float itemPrice;
cout<<"Enter the item price";
cin>>itemPrice;
cout<<"The price of 1 item is:" << calculateTotal(itemPrice)<<endl;
cout<<"The price of 5 item is:" << calculateTotal(itemPrice,5)<<endl;
return 0;
}
```

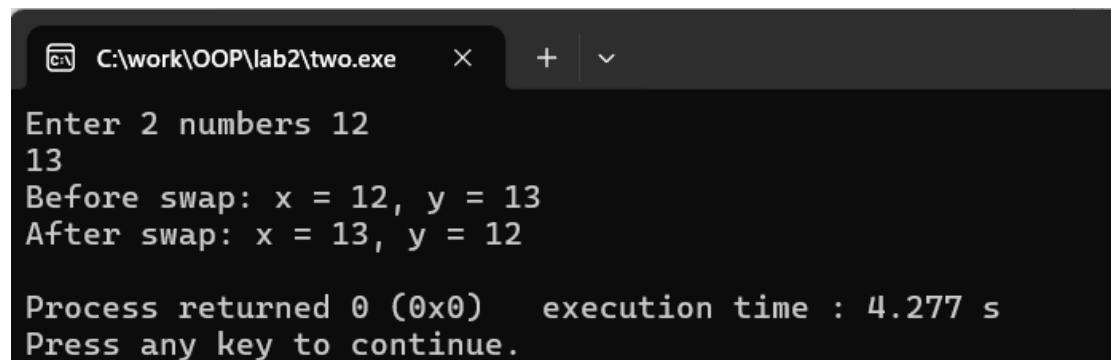
OUTPUT

A screenshot of a Windows command prompt window. The title bar shows the file path 'C:\work\OOP\lab2\two.exe' with standard window controls (minimize, maximize, close). The command prompt has a dark background with white text. The user has entered '100' in response to the prompt 'Enter the item price'. The program has then outputted two lines: 'The price of 1 item is:100' and 'The price of 5 item is:500'. At the bottom, it displays 'Process returned 0 (0x0) execution time : 5.991 s' and 'Press any key to continue.'

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

```
#include <iostream>
using namespace std;
void swapNumbers(int &a, int &b) {
    int temp = a;
    a = b;
    b = temp;
}
int main() {
    int x, y;
    cout<<"Enter 2 numbers";
    cin>>x>>y;
    cout << "Before swap: x = " << x << ", y = " << y << endl;
    swapNumbers(x, y);
    cout << "After swap: x = " << x << ", y = " << y << endl;
    return 0;
}
```

OUTPUT



```
C:\work\OOP\lab2\two.exe
Enter 2 numbers 12
13
Before swap: x = 12, y = 13
After swap: x = 13, y = 12

Process returned 0 (0x0)    execution time : 4.277 s
Press any key to continue.
```


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

```
#include <iostream>
using namespace std;
int& getElement(int arr[], int index) {
    return arr[index];
}
int main() {
    int n;
    cout<<"Enter the numbers of elements "<<endl;
    cin>>n;
    int numbers[n];
    cout<<"Enter "<<n<<" numbers"<<endl;
    for(int i=0;i<n;i++) {
        cout<<"Number "<<i+1 <<": ";
        cin>>numbers[i];
    }
    int a,b;
    cout<<" MODIFICATION"<<endl;
    cout<<"Enter the position [1- "<<n<<"] : ";
    cin>>a;
    cout<<"Value: ";
    cin>>b;
    cout << "Original value at index "<<a<<": " << numbers[a-1] << endl;
    getElement(numbers, a-1) = b;
    cout << "Modified value at index "<<a<<": " << numbers[a-1] << endl;
    return 0;
}
```

OUTPUT

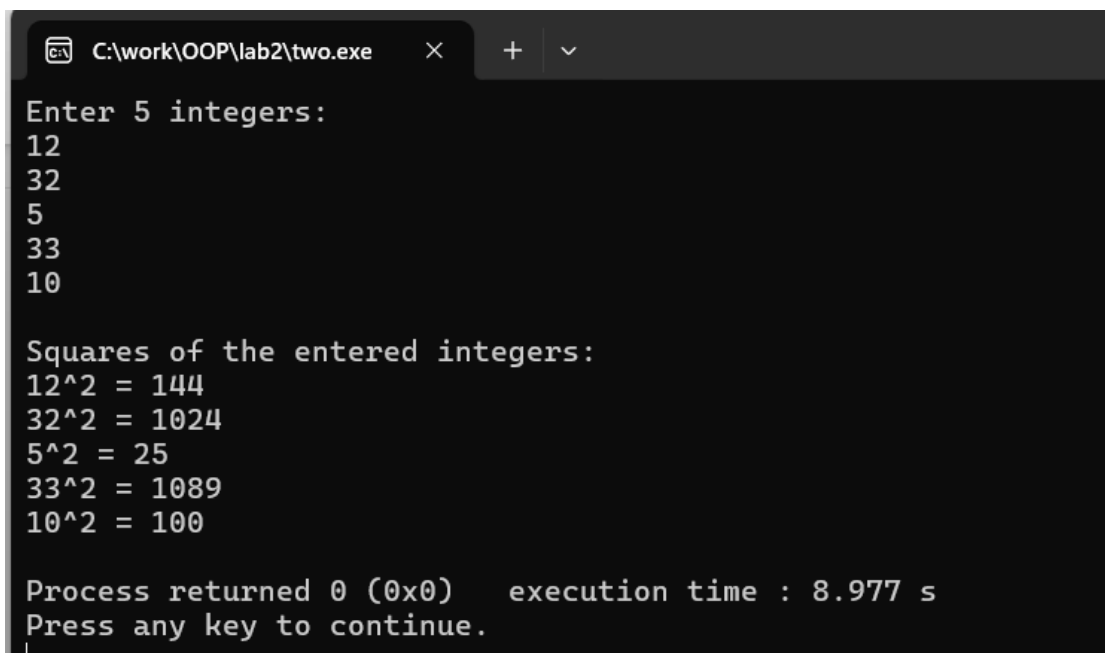
```
C:\work\OOP\lab2\two.exe
Enter the numbers of elements
5
Enter 5 numbers
Number 1: 23
Number 2: 342
Number 3: 353
Number 4: 55
Number 5: 234
MODIFICATION
Enter the position [1-5] : 3
Value: 23
Original value at index 3: 353
Modified value at index 3: 23

Process returned 0 (0x0)   execution time : 14.957 s
Press any key to continue.
```

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

```
#include <iostream>
using namespace std;
int main() {
    int arr[5];
    int* ptr = arr;
    cout << "Enter 5 integers:\n";
    for (int i = 0; i < 5; i++) {
        cin >> *(ptr + i);
    }
    cout << "\nSquares of the entered integers:\n";
    for (int i = 0; i < 5; i++) {
        cout << *(ptr + i) << "^2 = " << (*(ptr + i)) * (*(ptr + i)) << endl;
    }
    return 0;
}
```

OUTPUT

A screenshot of a Windows command prompt window with a dark background. The title bar shows the file path 'C:\work\OOP\lab2\two.exe'. The program prompts the user to 'Enter 5 integers:' and the user enters the values 12, 32, 5, 33, and 10 on separate lines. The program then displays 'Squares of the entered integers:' followed by the calculations: 12^2 = 144, 32^2 = 1024, 5^2 = 25, 33^2 = 1089, and 10^2 = 100. At the bottom, it shows 'Process returned 0 (0x0) execution time : 8.977 s' and 'Press any key to continue.'

```
C:\work\OOP\lab2\two.exe
Enter 5 integers:
12
32
5
33
10

Squares of the entered integers:
12^2 = 144
32^2 = 1024
5^2 = 25
33^2 = 1089
10^2 = 100

Process returned 0 (0x0)   execution time : 8.977 s
Press any key to continue.
```

CONCLUSION:

Through these programs, we have effectively implemented core C++ concepts. We practiced function overloading by using the same function name with different parameter types and applied inline functions to improve performance for simple operations. We used default arguments to make functions more flexible and concise. By implementing call-by-reference, we learned how to directly modify variables. We also manipulated arrays using references and pointers, allowing deeper understanding of memory access. Overall, these programs have strengthened our foundation in C++ programming and prepared us for more advanced applications.