



Himalaya College of Engineering

Advanced C++ Programming Lab Report

Lab 2: Array, Function, Inline function, Default argument, Structure, Pass by reference, Return by reference and DMA

Prepared By : Ankit Belbase
ROLL : HCE081BEI007
Subject : Object-Oriented Programming (OOP)
Program : Bachelor of Electronics, Communication and Information Engineering
Institution : Himalaya College of Engineering
Date : July 8, 2025

OBJECTIVE:

- To understand the use of array, functions, default arguments and also about DMA.
- To use the function, inline functions for basic programs.
- Demonstrate the concept of pass by reference, return by reference & DMA.

BACKGROUND THEORY:

In C++ programming, using modular coding techniques effectively is important for building strong and flexible applications. This lab covers some basic programming concepts that help build a solid foundation for these practices.

Arrays:

Arrays are fixed-size data structures that store elements of the same type in contiguous memory locations. They provide efficient access and manipulation of data using indexing. Arrays are especially useful when dealing with collections of similar data types and are known as homogeneous collections.

Syntax:

data type array name [size];

Two types of array:

- a) single dimensional array: data_type array_name [size] = {value1,value2.....,value n;
- 6) Multi-dimensional array: data_type array_name [size1][size 2][size n].

Function:

A function is a self-contained block of statements designed to perform a specific task. It is a logical unit where multiple statements are grouped together to act as one. In C programming, every program is made up of one or more of these functions.

Two types of functions:

- a. Library functions: pre-defined function.
- b. User defined function: Defined by user at a time of programming.

Structure:

A structure is a collection of heterogeneous data types, meaning it can store elements of different types. The individual elements inside a structure are called data members. In some programming languages, a structure is also referred to as a record. The main difference between an array and a structure is that all elements in an array must be of the same type, while a structure can contain elements of different types.

Syntax:

struct name {

```
data-type 1 member 1;  
data-type 2 member 2;  
data-type 3 member 3;  
data-type n member n;}
```

Inline functions:

Functions defined with the inline keyword suggest to the compiler to insert the complete function code at every place the function is called. This technique is especially useful for short functions that are used frequently in a program. Inline functions can be defined either inside or outside a class.

Syntax:

```
inline return_type function_name (parameters) {  
    //function body  
}
```

Example:

```
inline int cube (int a){  
    return a*a*a;  
}
```

Default arguments:

In C++, a default argument is a value assigned to a function parameter during its declaration. If the function is called without providing a value for that parameter, the compiler automatically uses the default value. However, if a value is given during the function call, it overrides the default value.

Syntax:

```
return_type function_name (parameter 1 = value 1,    );
```

Example:

```
#include <iostream>  
  
using namespace std;  
  
void greet(string name = "Guest") {  
    cout << "Hello, " << name << "!" << endl;  
}  
  
int main() {
```

```
greet();  
greet("Pradip");  
return 0;  
}
```

Pass by reference:

In C++, functions can receive arguments by value, reference, or pointer. Among these, pass by reference allows a function to directly access and modify the original variable passed by the caller. It is implemented using the reference operator &. In this parameter-passing method, the function receives a reference to the original data, not a copy, enabling direct manipulation of the actual values.

Syntax:

```
return_type function_name (datatype &parameter);
```

Example:

```
#include <iostream>  
using namespace std;  
void increase(int &num) {  
    num = num + 1;  
}  
  
int main() {  
    int value = 5;  
    increase(value);  
    cout << "Value after increase: " << value << endl;  
    return 0;  
}
```

Return by reference:

In C++, a function can return a reference variable instead of a copy of a value. This means the function gives back a direct link (alias) to the original variable from the caller, allowing it to be changed directly.

Syntax:

```
data Type & function_Name (parameters);
```

Example:

```

int &returnReference (int & x)
{
return x;
}

int main(){
int a = 20;
cout <<" Value of a= "<<a<<endl;
returnReference(a) = 33;
cout <<"value of x = " << a<<endl;
}

```

DMA:

Dynamic Memory Allocation (DMA) is the process of allocating memory while the program is running (at runtime). It is helpful when the exact amount of memory needed is not known beforehand. In C++, memory for DMA is taken from the heap using the `new` operator and freed using the `delete` operator.

new operator:

Allocates memory from heap & returns address of memory block.

Syntax: `pointer_variable = new data_type;`

Example:

```

int *ptr = new int;
*ptr = 10;

```

delete operator:

Used to free memory that was previously allocated using `new`.

Syntax:

`delete pointer_variable;`

Example:

```

int *ptr = new int(50);
delete ptr;

```

Q.No.1. Write a C++ program to overload a function add():

```
#include <iostream>

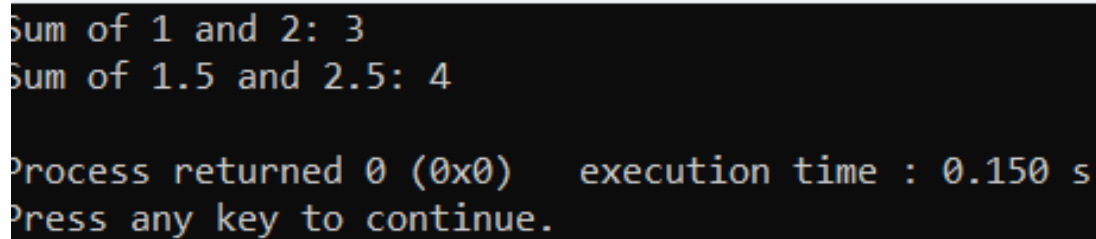
using namespace std;

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

double add(double a, double b) { return a + b;
}

int main() {
    cout << "Sum of 1 and 2: " << add(1, 2) << endl;
    cout << "Sum of 1.5 and 2.5: " << add(1.5, 2.5) << endl;
    return 0;
}
```

Output:

A screenshot of a terminal window with a black background and light blue text. It shows the output of the C++ program: 'Sum of 1 and 2: 3' followed by 'Sum of 1.5 and 2.5: 4'. Below this, it says 'Process returned 0 (0x0) execution time : 0.150 s' and 'Press any key to continue.' with a cursor on the next line.

```
Sum of 1 and 2: 3
Sum of 1.5 and 2.5: 4

Process returned 0 (0x0)   execution time : 0.150 s
Press any key to continue.

```

Q.No.2. Write an inline function in C++ to calculate the square of a number and demonstrate it with at least two function calls.

```
#include <iostream>

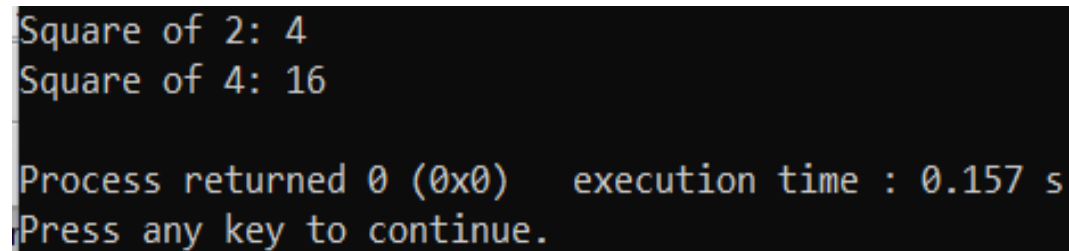
using namespace std;

inline int square (int x) {
    return x * x;
}
```

```
}
```

```
int main() {  
    cout << "Square of 2: " << square (2) << endl;  
    cout << "Square of 4: " << square (4) << endl;  
    return 0;  
}
```

Output:



```
Square of 2: 4  
Square of 4: 16  
  
Process returned 0 (0x0) execution time : 0.157 s  
Press any key to continue.
```

Q.No.3. 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;  
  
double totalPrice(double price, int quantity = 1) {  
    return price * quantity;  
}  
  
int main() {  
    cout << "Total price (1 item at $10.5): $" << totalPrice(10.5) << endl;  
    cout << "Total price (3 items at $10.5 each): $" << totalPrice(10.5, 3) << endl;  
    return 0;  
}
```

Output:

```
Total price (1 item at $10.5): $10.5
Total price (3 items at $10.5 each): $31.5

Process returned 0 (0x0)   execution time : 0.126 s
Press any key to continue.
```

Q.No.4. 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 = 1, y = 2;
    cout << "Before swap: x = " << x << ", y = " << y << endl;
    swapNumbers(x, y);
    cout << "After swap: x = " << x << ", y = " << y << endl;
    return 0;
}
```

Output:

```
Before swap: x = 1, y = 2
After swap: x = 2, y = 1

Process returned 0 (0x0)   execution time : 0.132 s
Press any key to continue.
```


Q.No.5. Create a function that returns a reference to an element in an array and modifies it.

```
#include <iostream>

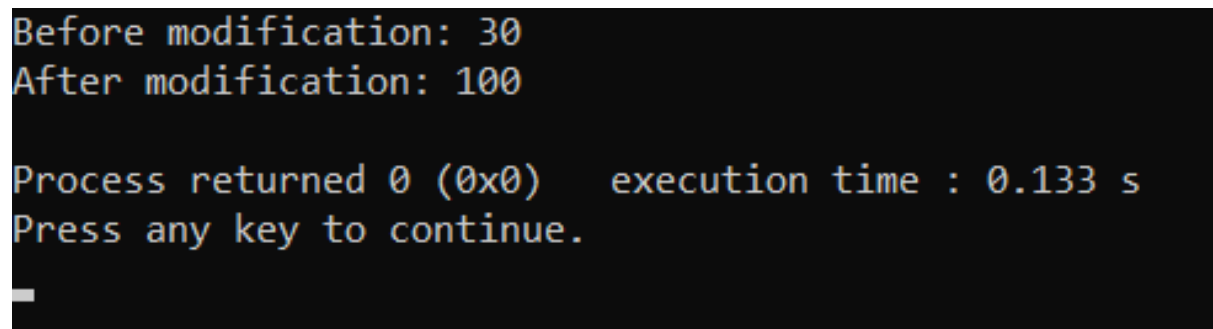
using namespace std;

int& getElement(int arr[], int index) {
    return arr[index];
}

int main() {
    int numbers[5] = {10, 20, 30, 40, 50};

    cout << "Before modification: " << numbers[2] << endl;
    getElement(numbers, 2) = 100;
    cout << "After modification: " << numbers[2] << endl;

    return 0;
}
```



```
Before modification: 30
After modification: 100

Process returned 0 (0x0)   execution time : 0.133 s
Press any key to continue.
```

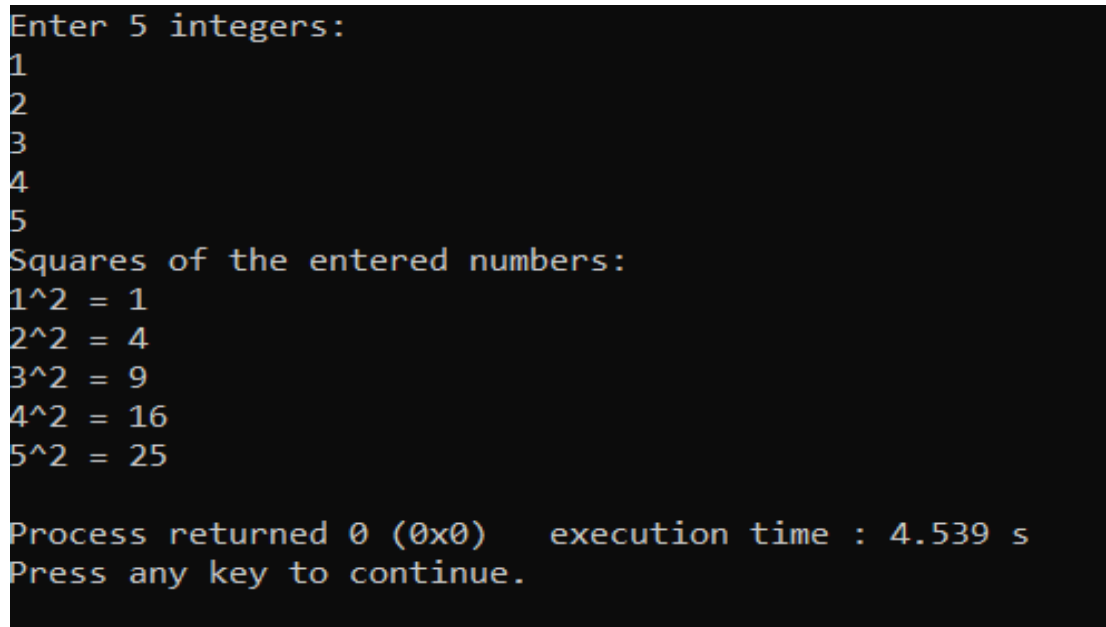
Q.No.6. Write a program to input 5 integers in an array and print their squares using a pointer.

```
#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 << "Squares of the entered numbers:\n";  
    for (int i = 0; i < 5; i++) {  
        int val = *(ptr + i);  
        cout << val << "^2 = " << val * val << endl;  
    }  
    return 0;  
}
```

Output:



```
Enter 5 integers:  
1  
2  
3  
4  
5  
Squares of the entered numbers:  
1^2 = 1  
2^2 = 4  
3^2 = 9  
4^2 = 16  
5^2 = 25  
  
Process returned 0 (0x0)    execution time : 4.539 s  
Press any key to continue.
```

Q.No.7. Define a structure Student with data members roll, name, and marks. Input and display details of 3 students.

```
#include<iostream>
#include<string>
using namespace std;
struct student {
    int roll;
    char name[25];
    float marks;
};
int main()
{
    student s[3];
    for(int i = 0; i < 3; i++) {
        cout << "Enter the details of students:" << (i + 1) << endl;

        cout << "Roll number: ";
        cin >> s[i].roll;
        cout << "Name: ";
        cin>>s[i].name;
        cout << "Marks: ";
        cin >> s[i].marks;
    }
    cout << "---Student Details---"<<endl;
    for(int i = 0; i < 3; i++) {
        cout<<"Student"<<i+1<<":";

        cout<<"Roll: " << s[i].roll << ", Name: " << s[i].name << ", Marks: " << s[i].marks << endl;
    }
    return 0;
}
```

Output:

```
Enter the details of students:1
Roll number: 1
Name: ankit
Marks: 100
Enter the details of students:2
Roll number: 2
Name: hari
Marks: 200
Enter the details of students:3
Roll number: shyam
Name: Marks: ---Student Details---
Student1:Roll: 1, Name: ankit, Marks: 100
Student2:Roll: 2, Name: hari, Marks: 200
Student3:Roll: 0, Name: α$, Marks: 6.39117e-035
Process returned 0 (0x0)   execution time : 22.662 s
Press any key to continue.
```

Q.No.8. Write a C++ program to demonstrate the difference between structure and union by declaring the same data members and showing memory usage.

```
#include <iostream>
using namespace std;
struct StudentStruct {
    int roll;
    float marks;
    char grade;
};
union StudentUnion {
    int roll;
    float marks;
    char grade;
};
int main() {
    StudentStruct s1;
    StudentUnion u1;
    cout << "Size of structure: " << sizeof(s1) << " bytes" << endl;
    cout << "Size of union: " << sizeof(u1) << " bytes" << endl;
    s1.roll = 101;
    s1.marks = 85.5;
    s1.grade = 'A';
    cout << "\nStructure values:" << endl;
    cout << "Roll: " << s1.roll << endl;
    cout << "Marks: " << s1.marks << endl;
    cout << "Grade: " << s1.grade << endl;
    u1.roll = 101;
    cout << "\nUnion value after setting roll: " << u1.roll << endl;
```

```

u1.marks = 85.5;
cout << "Union value after setting marks: " << u1.marks << endl;
u1.grade = 'A';
cout << "Union value after setting grade: " << u1.grade << endl;
cout << "\nAccessing all union members (may show unexpected values):" << endl;
cout << "Roll: " << u1.roll << endl;
cout << "Marks: " << u1.marks << endl;
cout << "Grade: " << u1.grade << endl;
return 0;
}

```

```

Size of structure: 12 bytes
Size of union: 4 bytes

Structure values:
Roll: 101
Marks: 85.5
Grade: A

Union value after setting roll: 101
Union value after setting marks: 85.5
Union value after setting grade: A

Accessing all union members (may show unexpected values):
Roll: 1118502977
Marks: 85.5005
Grade: A

Process returned 0 (0x0)   execution time : 0.187 s
Press any key to continue.

```

Q.No.9. Create an enum for days of the week. Display a message depending on the selected day.

```

#include <iostream>
using namespace std;
enum Day { Sunday, Monday, Tuesday, Wednesday, Thursday, Friday, Saturday };
int main() {

```

```
int a;

Day Today;

cout << "Enter any number between 1 and 7: ";

cin >> a;

if (a < 1 || a > 7) {
    cout << "Invalid input. Please enter a number between 1 and 7.";
    return 1;
}

Today = static_cast<Day>(a - 1);

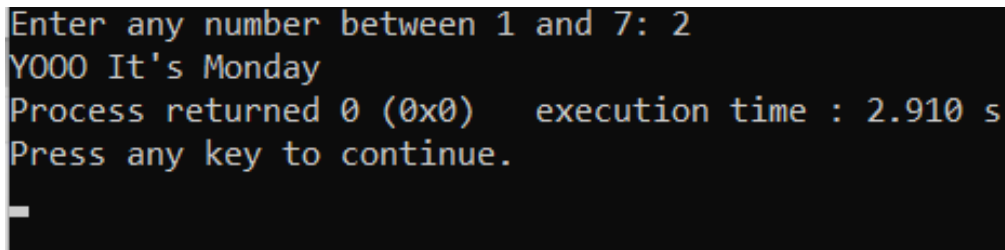
switch (Today) {
    case Sunday:
        cout << "YOOO It's Sunday";
        break;
    case Monday:
        cout << "YOOO It's Monday";
        break;
    case Tuesday:
        cout << "YOOO It's Tuesday";
        break;
    case Wednesday:
        cout << "YODO It's Wednesday";
        break;
    case Thursday:
        cout << "YOOO It's Thursday";
        break;
    case Friday:
        cout << "YOOO It's Friday";
        break;
    case Saturday:
        cout << "YOOO It's Saturday";
```

```

        break;
    }
    return 0;
}

```

Output:



```

Enter any number between 1 and 7: 2
Y000 It's Monday
Process returned 0 (0x0)   execution time : 2.910 s
Press any key to continue.
_

```

Q.No.10. 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.

```

#include <iostream>

using namespace std;

int main() {
    int n;

    cout << "Enter the number of elements: ";
    cin >> n;

    int* arr = new int[n];

    cout << "Enter " << n << " integers:" << endl;
    for (int i = 0; i < n; i++) {
        cin >> arr[i];
    }

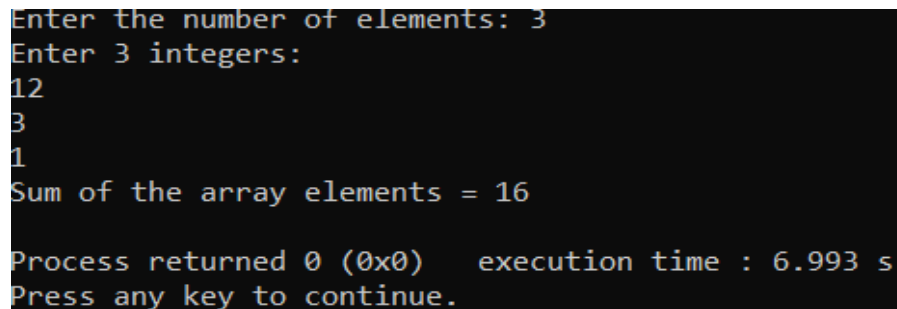
    int sum = 0;
    for (int i = 0; i < n; i++) {
        sum += arr[i];
    }

    cout << "Sum of the array elements = " << sum << endl;
}

```

```
delete[] arr;  
  
return 0;  
  
}
```

Output:

A terminal window with a black background and white text. The text shows the program's execution flow: a prompt to enter the number of elements, followed by three integers, the calculation of their sum, and a final status message with execution time.

```
Enter the number of elements: 3  
Enter 3 integers:  
12  
3  
1  
Sum of the array elements = 16  
  
Process returned 0 (0x0)   execution time : 6.993 s  
Press any key to continue.  
_
```

DISCUSSION

In this lab, we covered fundamental C++ concepts such as arrays, functions, inline functions, default arguments, structures, and dynamic memory allocation. We also worked with passing and returning values by reference to enhance performance. These techniques help make programs more modular, efficient, and adaptable. By using structures and dynamic memory allocation, we learned how to handle complex data types and manage memory at runtime effectively.

CONCLUSION

This lab deepened our grasp of essential C++ programming techniques that support efficient and effective coding. Features such as passing and returning by reference, along with dynamic memory allocation, improve memory management and scalability. These skills form a solid base for building more advanced applications and implementing object-oriented programming in future projects.