**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, efficient use of modular coding techniques are essential for developing robust & scalable applications. This lab focuses on several key programming concepts that form the please programming foundation for such practices.

**Arrays:**

Arrays are fixed-size data structures that store elements of the same type in contigous memory locations. They allow efficient access and manipulation of data using indexing. Arrays are particularly useful when working with collection of similar data types.

It is called as a homogeneous collection of similar data elements.

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 defined as a self-contained block of Statements that performs a particular task. This is a logical unit composed of a number of statements grouped into a single unit. Every C program is a collection of these functions.

Two types of functions:

1. Library functions: Pre-defined function.
2. User defined function! Defined by user at a time of programing.

**Structure:**

It is a collection of heterogeneous data types. Data member of is known as the items inside Structure. In some languages structure is known as record. The main difference between array & Structure is the element of array has same type while the element of structure can be of different type.

Syntax:

struct name { data-type 1 member 1; data-type 2 member2; datatype n member n; };

**Inline functions:**

Those defined with the inline keyword suggesting the compiler to insert the complete function code wherever the function i's called. Thi's technique is particularly usefull for functions that are short and used repeatedly in a Program. It can be defined inside or outside the class.

Syntax: inline return\_type function\_name (parameters) {

//function body

}

Example:

inline int square (int x){ return x\*x;

}

**Default arguments:**

In c++, a default argument is a value to a function parameter during its declaration. When the function is called without providing a value for that parameter, the compiler automatically substitutes the default value. If a value is provided in function call, it overrides the default value.Syntax:

return\_type function\_name (parameter 1 = value 1,----);

Example:

void display (int x=110) cout << x <<endl; main() { display(); //output: 110 display (220); //output: 220

}

**Pass by reference:**

In c++, functions can receive arguments by value, reference or pointer. Among these, pass by reference allows the function to directly access and manipulate the original variable passed by the caller. Implemented using the reference operator &. It is a parameter-passing method in which the function receives the memory address of the arguments, allowing it to operate directly on the original data rather than copy.

Syntax:

return\_type function\_name (datatype &parameter);

Example: void update (int &x){ x=x+5;

} main(){ int a = 10;

update (a); //a now becomes 15

}

**Return by reference:**

It is a feature in C++ where a function returns a reference variable, instead of returning a copy of its value. This allows the function to directly modify the (alias) to a orginal variable from the caller's scope.

Syntax:

data Type & function\_Name (parameters);

Example:

int &returnReference (int & x)

{

return x; // returns a reference to x

} main(){ int a = 20; cout <<”Value of a= "<<a<<endl; returnReference(a) = 50; // modifiles the value of x cout <<"value of x = " << a<<endl;

}

**DMA:**

Dynamic Memory Allocation (DMA) means assigning memory at the time of program execution (runtime). It is useful when we don't know the exact amount of memory needed before running program. In C++, DMA is allocated from the heap using the new operator & released 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(10); 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) {

cout<<"Sum of first two numbers:"<<a+b<<endl; return 0;

} int add(int a, int b, int c) { cout<<"Sum of all three numbers:"<<a+b+c;

return 0;

} int main()

{

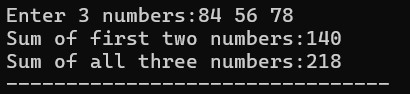
int a, b, c;

cout<<"Enter 3 numbers:"; cin>>a>>b>>c;

add(a,b); add (a,b,c); return 0;

}

**Output:**



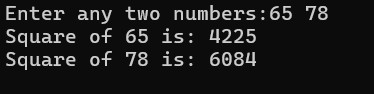
**Q.No.2. Write an inline function in C++ to calculate the square of a number and demonstrate it with at least two function calls.**

#include <iostream> using namespace std; inline int square(int n) { return n \* n;

} int main() { int num1,num2; cout<<"Enter any two numbers:"; cin>>num1>>num2; cout << "Square of " << num1 << " is: " << square(num1) << endl; cout << "Square of " << num2 << " is: " << square(num2) << endl; return 0;

}

**Output:**



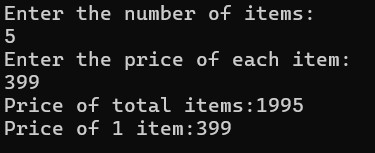
**Q.No.3. Write a program using a function with default arguments for calculating total price. The function should take the item price and quantity, with quantity defaulting to 1.**

#include<iostream> using namespace std; int total (int ab, int ba=1) { return ab\*ba;

} int main() { int a,b; cout<<"Enter the number of items: "<<endl; cin>>a; cout<<"Enter the price of each item:"<<endl; cin>>b; int totall=total(b,a); int total2=total (b); cout<<"Price of total items:"<<totall<<endl; cout<<"Price of 1 item:"<<total2<<endl; return 0;

}

**Output:**



**Q.No.4. Write a C++ program to swap two numbers using pass-by reference.**

#include<iostream> using namespace std; void swap(int &a, int &b)

{ int temp=a; a=b;

b=temp;

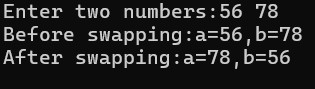
} int main() { int a,b;

cout<<"enter two numbers:"; cin>>a>>b; cout<<"Before swapping:a="<<a<<",b="<<b<<endl; swap(a,b); cout<<"After swapping:a="<<a<<",b="<<b<<endl;

return 0;

}

**Output:**



**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 &array (int arr[], int b)

{

return arr[b];

} int main(){ int num, ab, newnum; cout<<"Enter the number of elements:"; cin>>num;

int arr[num]; for(int i=0;i<num;i++){ cout<<"Enter element"<<i+1<<endl; cin>>arr[i];

} cout<<"Enter the number of element you want to edit:"<<endl; cin>>ab; cout<<"what do you want to replace with it?"; cin>>newnum; array (arr,ab-1)=newnum; **Output:**

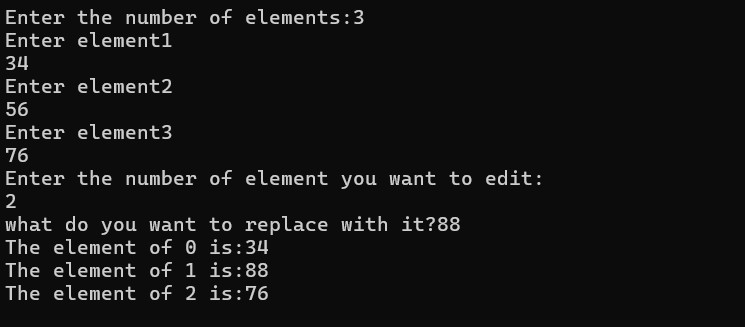
for(int i=0;i<num;i++) {

cout<<"The element of"<<" "<<i<<"

is:"<<array(arr, i) <<endl;

} return 0;

}

 **Output:**

**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: "; for(int i = 0; i < 5; i++)

{ cin >> \*(ptr + i);

}

cout << "Squares of the integers:"<<endl; for(int i = 0; i < 5; i++)

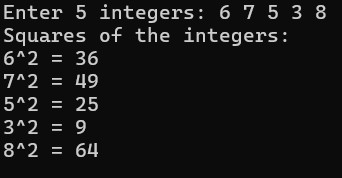
{

cout << \*(ptr + i) << "^2 = " << \*(ptr + i) \*(\*(ptr + i))<< endl;

} return 0;

}

**Output:**



**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: "; **Output:**

cin >> s[i].roll; cout << "Name: "; cin>>s[i].name; cout << "Marks: "; cin >> s[i].marks;

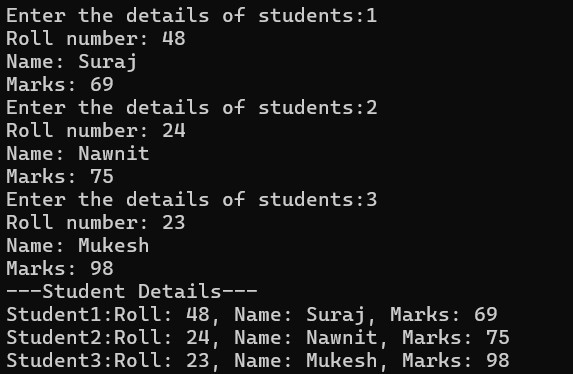
}

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:**



**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; char grade;

float marks; };

union StudentUnion {

int roll; char grade;

float marks; };

int main() {

StudentStruct s; StudentUnion u; cout << "Size of Structure: " << sizeof(s) << " bytes" << endl; cout << "Size of Union : " << sizeof(u) << " bytes" << endl; s.roll = 79;

s.grade = 'A';

s.marks = 99.5; cout << "\nStructure Data:\n"; cout << "Roll: " << s.roll << ", Grade: " << s.grade << ",Marks: " << s.marks << endl; u.roll = 79;

u.grade = 'A';

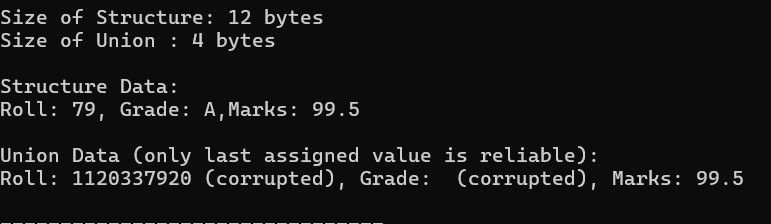
u.marks = 99.5; cout << "\nUnion Data (only last assigned value is reliable):\n"; cout << "Roll: " << u.roll << " (corrupted), Grade: " <<

u.grade << " (corrupted), Marks: " << u.marks << endl;

return 0;

}

**Output:**



**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 << "Hooray!! It's Monday";

break; case Tuesday:

cout << “Oh It's Tuesday"; break; case Wednesday:

cout << "Enjoy It's Wednesday";

break; case Thursday:

cout << "YOOO It's Thursday";

break;

case Friday: cout << "YOOO It's Friday";

break; case Saturday:

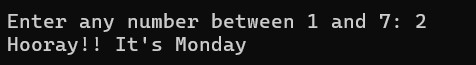
cout << "YOOO It's Saturday";

break;

} return 0;

}

**Output:**



**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.**

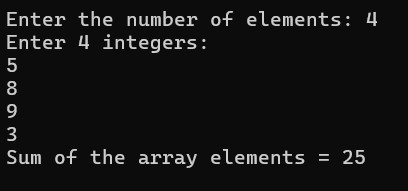
#include <iostream> using namespace std; int main() { int n; cout << "Enter the number of elements: "; cin >> n; int\* arr = new int[n]; cout << "Enter " << n << " integers:" << endl; for (int i = 0; i < n; i++) { cin >> arr[i];

} 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:**



**DISCUSSION**

In this lab, we explored some essential C++ concepts that help make programs more efficient and organized. We worked with arrays, functions, inline functions, and default arguments, and got hands-on experience with structures and dynamic memory allocation (DMA). A big focus was on how to pass and return values by reference, which helps boost performance by avoiding unnecessary copying. Using structures and dynamic memory allowed us to handle more complex data and memory management in a flexible way. Overall, these tools and techniques helped us write cleaner, more efficient, and modular code.

**CONCLUSION**

This lab helped deepen our understanding of fundamental C++ programming techniques that are essential for writing efficient and effective code. We explored key concepts like passing and returning values by reference, as well as dynamic memory allocation (DMA), both of which play a vital role in managing memory and building scalable programs. These skills lay a strong foundation for developing more advanced applications and applying object-oriented programming principles in future projects.