# Lecture Notes: C++ Programming (Beginner Level)

## 1. Arrays Declaration, Initialization, and Processing

An array is a collection of elements of the same data type stored in contiguous memory locations.

### Declaring an Array in C++

int numbers[5]; // Declaring an array of size 5

### Initializing an Array

int numbers[5] = {10, 20, 30, 40, 50}; // Initializing an array with values

### Accessing Array Elements

cout << numbers[0]; // Output: 10

### Processing an Array (Example: Printing all elements)

#include <iostream>  
using namespace std;  
  
int main() {  
 int numbers[5] = {10, 20, 30, 40, 50};  
 for(int i = 0; i < 5; i++) {  
 cout << "Element at index " << i << ": " << numbers[i] << endl;  
 }  
 return 0;  
}

## 2. Examples of Arrays

A stack is a data structure that follows LIFO (Last In, First Out).

### C++ Stack Implementation using Arrays

#include <iostream>  
#define SIZE 5  
  
using namespace std;  
  
class Stack {  
 int arr[SIZE], top;  
  
public:  
 Stack() { top = -1; }  
  
 void push(int val) {  
 if (top >= SIZE - 1)  
 cout << "Stack Overflow" << endl;  
 else  
 arr[++top] = val;  
 }  
  
 void pop() {  
 if (top < 0)  
 cout << "Stack Underflow" << endl;  
 else  
 top--;  
 }  
  
 void display() {  
 if (top < 0)  
 cout << "Stack is empty" << endl;  
 else {  
 for (int i = top; i >= 0; i--)  
 cout << arr[i] << " ";  
 cout << endl;  
 }  
 }  
};  
  
int main() {  
 Stack s;  
 s.push(10);  
 s.push(20);  
 s.display();  
 s.pop();  
 s.display();  
 return 0;  
}

## 3. Two-Dimensional Arrays and Strings

2D Arrays store data in rows and columns.

### 2D Arrays Declaration and Initialization

int matrix[2][3] = {  
 {1, 2, 3},  
 {4, 5, 6}  
};

### String Handling in C++

#include <iostream>  
#include <string>  
using namespace std;  
  
int main() {  
 string name = "John Doe";  
 cout << "Name: " << name << endl;  
 return 0;  
}

## 4. Structures in C++

A structure is a user-defined data type that groups related variables.

### Declaring a Structure

struct Student {  
 string name;  
 int age;  
 float marks;  
};

### Using Structures

#include <iostream>  
using namespace std;  
  
struct Student {  
 string name;  
 int age;  
 float marks;  
};  
  
int main() {  
 Student s1 = {"Ali", 20, 85.5};  
 cout << "Name: " << s1.name << ", Age: " << s1.age << ", Marks: " << s1.marks << endl;  
 return 0;  
}

## 5. Pointers in C++

Pointers store memory addresses.

### Declaring a Pointer

int x = 10;  
int\* ptr = &x;

### Pointer Arithmetic

cout << ptr << endl; // Address of x  
cout << \*ptr << endl; // Value of x

### Pointers and Arrays

int arr[] = {1, 2, 3};  
int\* ptr = arr; // Points to the first element  
  
cout << \*ptr; // Output: 1  
cout << \*(ptr + 1); // Output: 2

### Pointers and Functions

void square(int\* num) {  
 \*num = (\*num) \* (\*num);  
}  
  
int main() {  
 int x = 5;  
 square(&x);  
 cout << x; // Output: 25  
 return 0;  
}