Dynamic VS. Static Data Structures, Pointers, and Structures.

Pointers

A pointer is a variable that stores the memory address of another variable.

Syntax

```
int* ptr;
```

```
int var = 10;
ptr = &var;
```

& is the "address-of" operator.

* is the "dereference" operator.

Pointers

Dereferencing

```
cout << *ptr;
```

Assignment

```
int* ptr2 = ptr;
```

Pointers

Arithmetic

```
ptr++;
```

Pointers

Pointers and Arrays

```
int arr[5] = {1, 2, 3, 4, 5};
int* ptr = arr;
cout << *(ptr + 2);  // Outputs 3
```

Pointer to first element: arr == &arr[0]

Pointers

Pointers and Functions

```
void increment(int* p) { (*p)++; }
```

Pointers allow functions to modify variables in calling function.

Structure

A structure is a **user-defined** data type that allows the grouping of variables (of different types) under a single name.

```
struct StructureName {
    data_type member1;
    data_type member2;
    // more members...
};
```

```
struct Person {
    string name;
    int age;
    float height;
};
```

Structure

Accessing Structure Members

```
Person p1;  // Declare a structure variable
p1.name = "Alice";  // Access and assign structure members
p1.age = 25;
p1.height = 5.6;
```

```
Person p2 = {"John", 28, 6.1};
Person p3 = {"Emma", 22, 5.5};
```

Structure

Nested Structure

```
struct Address {
    string city;
    string state;
};

struct Employee {
    string name;
    int id;
    Address empAddress;  // Nested structure
};
```

```
Employee e1;
e1.empAddress.city = "New York";
e1.empAddress.state = "NY";
```

Structure

Array of Structure

```
Person people[3] = {
    {"Alice", 25, 5.6},
    {"Bob", 30, 5.9},
    {"Charlie", 22, 5.7}
};
```

```
cout << people[1].name;  // Outputs "Bob"
```

Structure

Pointers to Structure

```
Person p1 = {"Dave", 35, 6.0};
Person* ptr = &p1;
```

```
cout << ptr->name;  // Outputs "Dave"
```

Structure

Member gunctions

```
struct Person {
    string name;
    int age;

    // Member function to display person details
    void display() {
        cout << "Name: " << name << ", Age: " << age << endl;
    }

    // Member function to update age
    void updateAge(int newAge) {
        age = newAge;
    }
};
```

```
// Using the function to update age
p1.updateAge(30);
p1.display();  // Displays the updated age
```