Pointers

- What are pointers?
    - Variables that store memory addresses

- Syntax

```
int a = 10;
int *p = &a;  // p stores the address of a
```

- Importance in DSA:
    - Used in dynamic memory allocation, linked lists, trees, etc.

Arrays

- Definition: Contiguous memory locations holding elements of the same type.

- Example: `int arr[5] = {1, 2, 3, 4, 5};`

- Arrays and pointers relationship

```
int arr[5] = {1, 2, 3, 4, 5};
int *ptr = arr; // ptr points to arr[0]
// ptr++ points to arr[1]
int *ptr2 = ptr + 2; // ptr2 points to arr[2]
```

Structs and Memory Layout

- What are structs?
  - User-defined data types to group variables.

- Example

```
struct Point {
    int x, y;
};
struct Point p1 = {10, 20};
```

- Use in implementing data structures in DSA (e.g., Nodes in Linked Lists).

Dynamic Memory Allocation

- Functions:
    - malloc, calloc, realloc, free

- Example:

```c
int *arr = (int *)malloc(5 * sizeof(int));
if (arr == NULL) {
    printf("Memory not allocated\n");
} else {
    for (int i = 0; i < 5; i++) {
        arr[i] = i + 1;
    }
}
free(arr);
```

Debugging Tips

- Use of debugging tools (e.g., gdb, Visual Studio Code debugger).
- Common issues: Segmentation faults, uninitialized variables.

```
$ gcc -g program.c -o program
$ gdb ./program
(gdb) break main        # Set breakpoint at main
(gdb) run               # Run program
(gdb) print var         # Print variable value
(gdb) next              # Move to next line
(gdb) backtrace         # Display call stack
(gdb) quit              # Exit GDB
```

Putting It All Together

- Simple program to showcase:
  - Array and pointer traversal
  - Struct usage
  - Dynamic memory allocation

```c
#include <stdio.h>
#include <stdlib.h>

// Define a struct to represent a student
typedef struct {
    int id;
    char name[50];
    float grade;
} Student;


...
```

Putting It All Together

```
// Function to display student details
void displayStudent(const Student *student) {
    printf("ID: %d, Name: %s, Grade: %.2f\n", student->id,
student->name, student->grade);
}

int main() {
    int n, i;

    // Get the number of students
    printf("Enter the number of students: ");
    scanf("%d", &n);

    // Dynamically allocate memory for an array of students
    Student *students = (Student *)malloc(n * sizeof(Student));
    if (students == NULL) {
        printf("Memory allocation failed!\n");
        return 1;
    }
...
```

Putting It All Together

```
    // Input student details
    for (i = 0; i < n; i++) {
        printf("\nEnter details for student %d:\n", i + 1);
        printf("ID: ");
        scanf("%d", &students[i].id);
        printf("Name: ");
        scanf("%s", students[i].name);
        printf("Grade: ");
        scanf("%f", &students[i].grade);
    }

    // Display student details using pointer traversal
    printf("\nStudent Details:\n");
    for (i = 0; i < n; i++) {
        displayStudent(&students[i]);
    }

    free(students); // Free the allocated memory

    return 0;}
```

Resources

- '**The C Programming Language**' by Kernighan and Ritchie
- Online platforms like LeetCode, HackerRank