Arrays and structures in C

Course: Introduction to Programming and Data Structures

Laltu Sardar

Institute for Advancing Intelligence (IAI),
TCG Centres for Research and Education in Science and Technology (TCG Crest)

Arrays

Arrays

- An array is a collection of elements of the same type placed in contiguous memory locations.
- The elements are accessed using an index.
- Example:
  - `int arr[5];`
    This creates an array of 5 integers.
  - `arr[0] = 10;`
    Accessing the first element.

Array of Integers

- An array of integers is a simple array where each element is an integer.
- Example:
  - `int numbers[3] = {1, 2, 3};`
  - Here, `numbers[0] = 1`, `numbers[1] = 2`, `numbers[2] = 3`

Array of Floats

- Similar to an array of integers, but each element is a float.
- Example:
  - `float decimals[3] = {1.1, 2.2, 3.3};`
  - Here, `decimals[0] = 1.1`, `decimals[1] = 2.2`, `decimals[2] = 3.3`

Structures

- A structure is a user-defined data type in C that groups different types of variables.
- Example:

  -
    ```
    struct Student {
      char name[50];
      int roll;
      float marks;
    };
    ```
  - `struct Student s1 = {"John", 101, 92.5};`

Static Memory Allocation

- Static memory allocation is done at compile-time.
- Example:
  - `int arr[10];`
    The size of the array is fixed.
  - Memory is allocated when the program starts and deallocated when the program ends.

Dynamic Memory Allocation

- Dynamic memory allocation is done at run-time using functions like `malloc()`, `calloc()`, `realloc()`, and `free()`.
- Example:
  - `int *ptr = (int*)malloc(5 * sizeof(int));`
  - Memory is allocated during the execution of the program and can be resized or freed.

Memory Allocation in Structures

- Structures can be dynamically allocated memory using pointers.
- Example:
  - `struct Student *sPtr = (struct Student*)malloc(sizeof(struct Student));`
  - Accessing members:

```
sPtr->roll = 101;
```

or

```
(*sPtr).roll =101
```

Array of Structures

- An array of structures is an array where each element is a structure.
- Useful for storing data related to multiple entities that share a common structure.
- Example:
  - `struct Student students[3];`
    Creates an array of 3 `Student` structures.
  - Each element in the array is a structure with its own set of member variables.

tcg crest
Inventing Harmonious Future

Initializing an Array of Structures

- An array of structures can be initialized at the time of declaration.
- Example:
  - `struct Student students[3] = {{"John", 101, 92.5}, {"Alice", 102, 85.0}, {"Bob", 103, 88.7}};`
  - Access individual elements as `students[0].name`, `students[1].roll`, etc.

Accessing Members in an Array of Structures

- Members of the structures within the array are accessed using the array index and the member operator (`.`).
- Example:
  - `printf("Name: %s, Roll: %d, Marks: %f n", students[0].name, students[0].roll, students[0].marks);`
  - Iterating through the array:
    - `for(int i = 0; i < 3; i++) { printf("%s %d %f n", students[i].name, students[i].roll, students[i].marks); }`

Array of Structures and Memory Allocation

- Memory for the array of structures is allocated statically if the array size is fixed.
- Dynamic allocation can be used if the array size needs to be determined at runtime.
- Example of dynamic allocation:
  - `struct Student *students = (struct Student*)malloc(3 * sizeof(struct Student));`
  - Access and assignment is similar: `students[0].roll = 101;`

Common Operations on Array of Structures

- **Initialization**: Initialize each structure in the array individually or at the time of declaration.
- **Traversal**: Loop through the array to access or modify the structure members.
- **Sorting:** Sort the array of structures based on a particular member, like sorting students by marks.
- **Searching:** Search for a structure with a particular member value, like finding a student by roll number.

Sorting an Array of Structures

- Sorting can be performed on an array of structures based on any member, such as sorting by `roll` or `marks`.
- Example:
  - Bubble Sort based on `marks`:

```c
for(int i = 0; i < 3; i++) {
  for(int j = 0; j < 2; j++) {
    if(students[j].marks > students[j+1].marks) {
      struct Student temp = students[j];
      students[j] = students[j+1];
      students[j+1] = temp;
    }
  }
}
```

Thank You
for your attention.

Questions?

tcg crest
Inventing Harmonious Future

Laltu Sardar
laltu {dot} sardar [at]tcgcrest(.)org
https://laltu-sardar.github.io