Introduction to Arrays:

An array in C is a collection of elements of the same data type, stored in contiguous memory locations. It allows you to store and manipulate a fixed-size sequence of values.

Declaring an Array:

To declare an array, you need to specify the data type of its elements and its size. The general syntax for declaring an array is:

```
data_type array_name[size];
```

For example, to declare an integer array with 5 elements, you can use:

```
int numbers[5];
```

Initializing an Array:

You can initialize an array at the time of declaration using an initializer list. The number of values in the initializer list must match the size of the array.

Here’s an example:

```
int numbers[] = {1, 2, 3, 4, 5};
```

```
```

This initializes an integer array with 5 elements and assigns the values 1, 2, 3, 4, 5 to the respective elements.

Accessing Array Elements:

You can access individual elements of an array using their index. Array indices start from 0 and go up to size-1.

For example, to access the second element of the array “numbers”, you can use:

```
int secondElement = numbers[1];
```

Modifying Array Elements:

You can modify the value of an array element by assigning a new value to it. For example, to change the third element of the array “numbers” to 10, you can use:

```
numbers[2] = 10;
```

Iterating over Array Elements:

To access all elements of an array, you can use loops. The most common loop used for iterating over an array is the “for” loop.

Here’s an example that prints all the elements of the array “numbers”:

```
for (int i = 0; i < 5; i++) {
    printf("%d ", numbers[i]);
}
```

Array Size:

To determine the size of an array, you can use the “sizeof” operator. For example, to get the size of the array “numbers”, you can use:

```
int size = sizeof(numbers) / sizeof(numbers[0]);
```

This calculates the total size of the array in bytes and divides it by the size of a single element to get the number of elements.

Arrays as Function Parameters:

You can pass arrays to functions in C. When passing an array to a function, you typically need to pass its size as well.

Here’s an example function that calculates the sum of elements in an integer array:

```
int calculate_sum(int arr[], int size) {
    int sum = 0;
    for (int i = 0; i < size; i++) {
        sum += arr[i];
```

```
    }
    return sum;
}
```

To call this function with the “numbers” array, you can use:

```
int sum = calculate_sum(numbers, size);
```

These are the basics of working with arrays in C programming.

Practice Problems on Array:

Problem 1: Guess the Output

```
#include <stdio.h>

int main() {
    int arr[] = {1, 2, 3, 4, 5};
    printf("%d\n", arr[2]);
    return 0;
}
```

Explanation:

This program declares an integer array arr with 5 elements: {1, 2, 3, 4, 5}. Each element of the array is initialized with a specific value.

The printf function is then used to print the value of arr[2]. Array indexing in C starts from 0, so arr[2] refers to the element at index 2, which is the third element of the array (with a value of 3).

Problem 2: Guess the Output

```
#include <stdio.h>

int main() {
    int arr[] = {1, 2, 3, 4, 5};
    printf("%d\n", arr[5]);
    return 0;
}
```

Problem 3: Guess the Output

```
#include <stdio.h>

int main() {
    int arr[] = {1, 2, 3, 4, 5};
    arr[2] = 7;
    printf("%d\n", arr[2]);
    return 0;
}
```

Problem 4: Guess the Output

```
#include <stdio.h>

int main() {
    int arr[] = {1, 2, 3, 4, 5};
    for (int i = 0; i < 5; i++) {
        printf("%d ", arr[i]);
    }
    return 0;
}
```

Problem 5:

Calculate the sum of all elements in an array.

```
#include <stdio.h>

int calculate_sum(int arr[], int size) {
    int sum = 0;
    for (int i = 0; i < size; i++) {
        sum += arr[i];
    }
    return sum;
}

int main() {
    int arr[] = {5, 9, 3, 1, 8};
    int size = sizeof(arr) / sizeof(arr[0]);
    int sum = calculate_sum(arr, size);
    printf("Sum of all elements in the array: %d\n", sum);
    return 0;
```

```
}
```

Output:

```
Sum of all elements in the array: 26
```

Problem 6:

Find the maximum element in an array.

```
#include <stdio.h>

int find_max_element(int arr[], int size) {
    int max_element = arr[0];
    for (int i = 1; i < size; i++) {
        if (arr[i] > max_element) {
            max_element = arr[i];
        }
    }
    return max_element;
}

int main() {
    int arr[] = {5, 9, 3, 1, 8};
    int size = sizeof(arr) / sizeof(arr[0]);
    int max = find_max_element(arr, size);
    printf("Maximum element in the array: %d\n", max);
    return 0;
}
```

Output:

```
Maximum element in the array: 9
```

Problem 7:

Find the second largest element in an array.

```
#include <stdio.h>

int findSecondLargest(int arr[], int size) {
    int largest = arr[0];
    int secondLargest = arr[0];

    for (int i = 1; i < size; i++) {
        if (arr[i] > largest) {
            secondLargest = largest;
            largest = arr[i];
        } else if (arr[i] > secondLargest && arr[i] < largest) {
            secondLargest = arr[i];
        }
    }

    return secondLargest;
}

int main() {
    int arr[] = {5, 9, 3, 1, 8};
    int size = sizeof(arr) / sizeof(arr[0]);
    int secondLargest = findSecondLargest(arr, size);
```

```
    printf("Second largest element in the array: %d\n",
secondLargest);
    return 0;
}
```

Output:

```
Second largest element in the array: 8
```

Problem 8:

Calculate the average of all elements in an array.

```
#include <stdio.h>

float calculateAverage(int arr[], int size) {
    int sum = 0;

    for (int i = 0; i < size; i++) {
        sum += arr[i];
    }

    return (float)sum / size;
}

int main() {
    int arr[] = {5, 9, 3, 1, 8};
    int size = sizeof(arr) / sizeof(arr[0]);
    float average = calculateAverage(arr, size);
    printf("Average of all elements in the array: %.2f\n", average);
    return 0;
```

```
}
```

Note: In C, the sizeof() operator is used to determine the size, in bytes, of a type or an object. When used with an array, sizeof() returns the total size of the array in bytes.

Output:

```
Average of all elements in the array: 5.20
```

Problem 9:

Given an array arr of size n, write a C program to find the sum of all elements present at even index positions.

```
#include <stdio.h>

int main() {
    int arr[] = {2, 7, 3, 9, 4, 8, 6};
    int n = sizeof(arr) / sizeof(arr[0]);
    int sum = 0;

    for (int i = 0; i < n; i += 2) {
        sum += arr[i];
    }

    printf("Sum of elements at even index positions: %d\n", sum);

    return 0;
}
```

Explanation:
In this program, we initialize an array arr with the given values. We calculate the size of the array n by dividing the total size of the array by the size of a single element. We then initialize a variable sum to store the sum of the elements at even index positions. We iterate through the array using a for loop, starting from index 0 and incrementing i by 2 in each iteration to access only the even index positions. We add the element at the current index to the sum variable. Finally, we print the sum of the elements at even index positions.

Output:

```
Sum of elements at even index positions: 15
```

Problem 10:

Calculate the percentage of marks.

```
#include <stdio.h>

int main() {
    int marks[] = {85, 92, 78, 90, 88};
    int totalSubjects = sizeof(marks) / sizeof(marks[0]);
    int totalMarks = 0;

    // Calculate total marks
    for (int i = 0; i < totalSubjects; i++) {
        totalMarks += marks[i];
    }
```

```
    // Calculate percentage
    float percentage = (float)totalMarks / totalSubjects;

    printf("Total marks: %d\n", totalMarks);
    printf("Percentage: %.2f%%\n", percentage);

    return 0;
}
```

Explanation:

In this program, we have an array marks that stores the marks obtained in different subjects. The variable totalSubjects is used to calculate the total number of subjects by dividing the total size of the array by the size of a single element.

We initialize totalMarks to 0 and then use a loop to iterate through the array and add up all the marks to calculate the total marks.

Next, we calculate the percentage by dividing the total marks by the total number of subjects. The result is stored in the variable percentage.

Finally, we use printf to display the total marks and the calculated percentage. The .2f in the format specifier ensures that the percentage is displayed with two decimal places, and the % is escaped with %% to display it as a literal percentage symbol.

When you run the program, it will output the total marks and the percentage of marks obtained based on the array values.

Output:

```
Total marks: 433
Percentage: 86.60%
```

Related posts:

1. C prgoram to convert inch to feet
2. C program to convert KM to CM
3. C program to convert meter to centimeter
4. C program to calculate remainder, difference, division, product
5. C program to use printf() without semicolon ” ; “
6. C program to swap two numbers using 2 variables
7. C program to find nth term using Arithmetic progrssion
8. C program to find sum of first n even positive numbers
9. C program to calculate sum of first n even numbers
10. C program to find nth odd number
11. C program to find sum of first n odd positive numbers
12. C program to calculate perimeter and area of a rectangle
13. C program to calculate perimeter and area of a square
14. C program to calculate Perimeter and Area of Circle
15. Function in C Programming
16. C Programming Q & A
17. Main function in C Programming Q and A
18. Void main in C Programming
19. Variables Q and A in C Programming
20. Write a C Program to find the percentage of marks ?
21. Write a c program to find age of a person ?

22. Write a c program to get table of a number
23. What is Break statement in C Programming ?
24. Write a c program to generate all combinations of 1, 2 and 3 using for loop.
25. Write a C program to print all the prime numbers between 1 to 50.
26. Write a C program to get factorial of a number ?
27. What is user defined function in C programming ?
28. Difference between C and C++ Programming ?
29. Difference between C, C++ and Java Programming
30. C program addition of numbers using pointer
31. C Syntax
32. Comments in C
33. Variables in C
34. Data types in C
35. Format specifiers in C
36. Type Conversion in C
37. Constants in C
38. Operators in C
39. Pre and Post Increament Practice Problems
40. Pre and Post Increament
41. C Introduction
42. C Get Started
43. C Pointers
44. C History
45. C Program Compiling and running
46. C While loop
47. C Do While Loop
48. C For loop

49. break and continue statement
50. Control Statements in C
51. C if-else ladder
52. C if statements
53. C 2-Dimensional array
54. C String library functions
55. C Functions
56. C Functions Categories
57. C Actual Arguments
58. Write a program that prints the message “Hello, World!”
59. Write a program that asks the user to enter two numbers, and then prints the sum of those two numbers.
60. Write a program that asks the user to enter a number and then determines whether the number is even or odd.
61. Write a program that swaps the values of two variables.
62. Write a program that asks the user to enter a number and then calculates and prints its factorial.
63. Write a program that asks the user to enter a number N and then prints the first N numbers in the Fibonacci sequence
64. Write a program that swaps the values of two variables without using a temporary variable
65. Converts a number into integer, float, and string
66. Program to find the length of the string
67. Program to convert string to uppercase or lowercase
68. Program to prints the numbers from 1 to 10.
69. What is identifier expected error
70. Difference between static and non static methods in Java

71. C String Input
72. C Character input
73. C Programming Variables MCQ
74. Object & Classes
75. C Programming find the output MCQs