Arrays in C Programming

Introduction to Arrays

An array is a fundamental data structure in C that allows you to store multiple values of the same data type under a single variable name. Each element in an array is identified by an index, starting from 0 for the first element. Arrays are useful for working with collections of data, such as lists of numbers, strings, or any other data type.

In this chapter, we will explore the concepts, declaration, initialization, and manipulation of arrays in C programming. Additionally, we will discuss some of the most common operations and provide ten important examples of array usage.

Declaration and Initialization

In C, you declare an array by specifying its data type, followed by the array name and the size of the array in square brackets. For example, to declare an integer array with five elements, you would write:

```
int myArray[5];
```

You can also initialize an array at the time of declaration, like this:

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

If you do not specify the size when initializing, the compiler will determine it based on the number of values provided in the initialization list. For example:

```
int myArray[] = {1, 2, 3, 4, 5}; // Size is automatically set to 5
```

Accessing Array Elements

Accessing elements of an array is done by using square brackets with the index of the element. Remember that array indices start from 0. For example, to access the third element of '`myArray`', you would use:

int thirdElement = myArray[2];

Important Array Operations and Examples

1. Sum of Array Elements

This program calculates the sum of all elements in an integer array.

```
#include <stdio.h>

int main() {

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

        int sum = 0;


        for (int i = 0; i < 5; i++) {

        sum += myArray[i];

        }


        printf("Sum: %d\n", sum);
```

```
    return 0;
}
```

2. Finding the Largest Element

This program finds the largest element in an array of integers.

```
#include <stdio.h>
int main() {
    int myArray[] = {5, 2, 9, 1, 6};
    int max = myArray[0];

    for (int i = 1; i < 5; i++) {
    if (myArray[i] > max) {
    max = myArray[i];
    }
    }

    printf("Largest element: %d\n", max);

    return 0;
```

```
}
```

3. Reversing an Array

This program reverses the order of elements in an array.

```
#include <stdio.h>

int main() {
        int myArray[] = {1, 2, 3, 4, 5};
        int temp;
        int size = 5;

        for (int i = 0; i < size / 2; i++) {
        temp = myArray[i];
        myArray[i] = myArray[size - 1 - i];
        myArray[size - 1 - i] = temp;
        }

        printf("Reversed array: ");
        for (int i = 0; i < size; i++) {
        printf("%d ", myArray[i]);
        }

        return 0;
}
```

4. Searching for an Element

This program searches for a specific element in an array.

```
#include <stdio.h>

int main() {
	int myArray[] = {10, 20, 30, 40, 50};
	int searchValue = 30;
	int found = 0;

	for (int i = 0; i < 5; i++) {
	if (myArray[i] == searchValue) {
	found = 1;
	break;
	}
	}

	if (found) {
	printf("Element %d found in the array.\n", searchValue);
	} else {
	printf("Element %d not found in the array.\n", searchValue);
	}

	return 0;
}
```

5. Copying Arrays

This program copies the elements of one array into another.

```
#include <stdio.h>

int main() {
	int sourceArray[] = {1, 2, 3, 4, 5};
	int targetArray[5];
```

```
    for (int i = 0; i < 5; i++) {
    targetArray[i] = sourceArray[i];
    }

    printf("Copied array: ");
    for (int i = 0; i < 5; i++) {
    printf("%d ", targetArray[i]);
    }

    return 0;
}
```

6. Sorting an Array

This program sorts an integer array in ascending order using the bubble sort algorithm.

```
#include <stdio.h>

int main() {
    int myArray[] = {64, 34, 25, 12, 22, 11, 90};
    int size = 7;

    for (int i = 0; i < size - 1; i++) {
    for (int j = 0; j < size - i - 1; j++) {
    if (myArray[j] > myArray[j + 1]) {
        int temp = myArray[j];
        myArray[j] = myArray[j + 1];
        myArray[j + 1] = temp;
    }
    }
    }

    printf("Sorted array: ");
    for (int i = 0; i < size; i++) {
    printf("%d ", myArray[i]);
```

```
    }

    return 0;
}
```

7. Deleting an Element

This program deletes an element from an array.

```
#include <stdio.h>

int main() {
    int myArray[] = {10, 20, 30, 40, 50};
    int size = 5;
    int deleteIndex = 2; // Index of element to be deleted

    for (int i = deleteIndex; i < size - 1; i++) {
    myArray[i] = myArray[i + 1];
    }

    size--;

    printf("Array after deleting element at index %d: ", deleteIndex);
    for (int i = 0; i < size; i++) {
    printf("%d ", myArray[i]);
    }

    return 0;
}
```

8. Inserting an Element

This program inserts an element into an array at a specified index.

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

```
int main() {
	int myArray[] = {10, 20, 30, 40, 50};
	int size = 5;
	int insertIndex = 2; // Index where the element will be inserted
	int insertValue = 25;

	for (int i = size; i > insertIndex; i--) {
	myArray[i] = myArray[i - 1];
	}
	myArray[insertIndex] = insertValue;
	size++;

	printf("Array after inserting element at index %d: ", insertIndex);
	for (int i = 0; i < size; i++) {
	printf("%d ", myArray[i]);
	}

	return 0;
}
```

9. Counting Even and Odd Numbers

This program counts the number of even and odd elements in an array of integers.

```
#include <stdio.h>

int main() {
	int myArray[] = {1, 2, 3, 4, 5, 6, 7, 8, 9};
	int evenCount = 0, oddCount = 0;
	int size = 9;

	for (int i = 0; i < size; i++) {
	if (myArray[i] % 2 == 0) {
	evenCount++;
	} else {
	oddCount++;
```

```
        }
        }

        printf("Even numbers: %d\n", evenCount);
        printf("Odd numbers: %d\n", oddCount);

        return 0;
}
```

10. Finding the Second Largest Element

This program finds the second largest element in an array of integers.

```
#include <stdio.h>

int main() {
        int myArray[] = {10, 30, 40, 50, 20};
        int size = 5;
        int max = myArray[0];
        int secondMax = myArray[0];

        for (int i = 1; i < size; i++) {
        if (myArray[i] > max) {
        secondMax = max;
        max = myArray[i];
        } else if (myArray[i] > secondMax && myArray[i] != max) {
        secondMax = myArray[i];
        }
        }

        printf("Second largest element: %d\n", secondMax);

        return 0;
}
```

11. Add Two Matrices Using Multi-dimensional Arrays

```
#include <stdio.h>
int main() {
  int r, c, a[100][100], b[100][100], sum[100][100], i, j;
  printf("Enter the number of rows (between 1 and 100): ");
  scanf("%d", &r);
  printf("Enter the number of columns (between 1 and 100): ");
  scanf("%d", &c);

  printf("\nEnter elements of 1st matrix:\n");
  for (i = 0; i < r; ++i)
        for (j = 0; j < c; ++j) {
        printf("Enter element a%d%d: ", i + 1, j + 1);
        scanf("%d", &a[i][j]);
        }

  printf("Enter elements of 2nd matrix:\n");
  for (i = 0; i < r; ++i)
        for (j = 0; j < c; ++j) {
        printf("Enter element b%d%d: ", i + 1, j + 1);
        scanf("%d", &b[i][j]);
        }

  // adding two matrices
  for (i = 0; i < r; ++i)
        for (j = 0; j < c; ++j) {
        sum[i][j] = a[i][j] + b[i][j];
        }

  // printing the result
  print("\nSum of two matrices: \n");
  for (i = 0; i < r; ++i)
        for (j = 0; j < c; ++j) {
        printf("%d   ", sum[i][j]);
        if (j == c - 1) {
        printf("\n\n");
```

```
        }
        }

  return 0;
}
```

Output

```
Enter the number of rows (between 1 and 100): 2
Enter the number of columns (between 1 and 100): 3

Enter elements of 1st matrix:
Enter element a11: 2
Enter element a12: 3
Enter element a13: 4
Enter element a21: 5
Enter element a22: 2
Enter element a23: 3
Enter elements of 2nd matrix:
Enter element b11: -4
Enter element b12: 5
Enter element b13: 3
Enter element b21: 5
Enter element b22: 6
Enter element b23: 3

Sum of two matrices:
-2   8   7

10   8   6
```

12. Multiply Two Matrices Using Multi-dimensional Arrays

A prerequisite for multiplying two matrices is that the number of columns in the first matrix must be equal to the number of rows in the second matrix.

```
#include <stdio.h>

// function to get matrix elements entered by the user
void getMatrixElements(int matrix[][10], int row, int column) {

   printf("\nEnter elements: \n");

   for (int i = 0; i < row; ++i) {
         for (int j = 0; j < column; ++j) {
         printf("Enter a%d%d: ", i + 1, j + 1);
         scanf("%d", &matrix[i][j]);
         }
   }
}

// function to multiply two matrices
void multiplyMatrices(int first[][10],
                      int second[][10],
                      int result[][10],
                      int r1, int c1, int r2, int c2) {

   // Initializing elements of matrix mult to 0.
   for (int i = 0; i < r1; ++i) {
         for (int j = 0; j < c2; ++j) {
         result[i][j] = 0;
         }
   }

   // Multiplying first and second matrices and storing it in result
   for (int i = 0; i < r1; ++i) {
         for (int j = 0; j < c2; ++j) {
         for (int k = 0; k < c1; ++k) {
         result[i][j] += first[i][k] * second[k][j];
```

```
            }
            }
    }
}

// function to display the matrix
void display(int result[][10], int row, int column) {

    printf("\nOutput Matrix:\n");
    for (int i = 0; i < row; ++i) {
            for (int j = 0; j < column; ++j) {
            printf("%d  ", result[i][j]);
            if (j == column - 1)
            printf("\n");
            }
    }
}

int main() {
    int first[10][10], second[10][10], result[10][10], r1, c1, r2, c2;
    printf("Enter rows and column for the first matrix: ");
    scanf("%d %d", &r1, &c1);
    printf("Enter rows and column for the second matrix: ");
    scanf("%d %d", &r2, &c2);

    // Taking input until
    // 1st matrix columns is not equal to 2nd matrix row
    while (c1 != r2) {
            printf("Error! Enter rows and columns again.\n");
            printf("Enter rows and columns for the first matrix: ");
            scanf("%d%d", &r1, &c1);
            printf("Enter rows and columns for the second matrix: ");
            scanf("%d%d", &r2, &c2);
    }

    // get elements of the first matrix
    getMatrixElements(first, r1, c1);
```

```
    // get elements of the second matrix
    getMatrixElements(second, r2, c2);

    // multiply two matrices.
    multiplyMatrices(first, second, result, r1, c1, r2, c2);

    // display the result
    display(result, r1, c2);

    return 0;
}
```

Output

```
Enter rows and column for the first matrix: 2
3
Enter rows and column for the second matrix: 3
2

Enter elements:
Enter a11: 2
Enter a12: -3
Enter a13: 4
Enter a21: 53
Enter a22: 3
Enter a23: 5

Enter elements:
Enter a11: 3
Enter a12: 3
Enter a21: 5
Enter a22: 0
Enter a31: -3
Enter a32: 4

Output Matrix:
```

```
-21  22
159  179
```

These examples cover a range of common operations and tasks you can perform with arrays in C. Understanding arrays and how to manipulate them is essential for effective programming in C, and these examples should provide a solid foundation for working with arrays in various contexts.