Functions and Arrays in C++

Function Basics

Function Structure

```
returnType functionName(parameter1, parameter2) {
    // Function body
    return value; // if returnType is not void
}
```

Simple Function Examples

```
// Function with no parameters
void sayHello() {
    cout << "Hello!" << endl;
}

// Function with parameters
void greetPerson(string name) {
    cout << "Hello, " << name << "!" << endl;
}

// Function with return value
int addNumbers(int a, int b) {
    return a + b;
}

// Function with multiple parameters
double calculateArea(double length, double width) {
    return length * width;
}
```

Function Prototypes

```
#include <iostream>
using namespace std;

// Function prototypes (declarations)
void displayMenu();
int multiply(int x, int y);
double findAverage(int arr[], int size);

int main() {
    displayMenu();
    int result = multiply(5, 3);
    cout << "Result: " << result << endl;
    return 0;
}

// Function definitions
void displayMenu() {
```

```
    cout << "=== Main Menu ===" << endl;
}

int multiply(int x, int y) {
    return x * y;
}
```

Array Basics

Creating Arrays

```
// Declare and initialize
int numbers[5] = {10, 20, 30, 40, 50};
string names[3] = {"Alice", "Bob", "Charlie"};

// Declare then assign
int scores[4];
scores[0] = 85;
scores[1] = 92;
scores[2] = 78;
scores[3] = 96;

// Partial initialization
int values[5] = {1, 2}; // Rest will be 0
```

Accessing Arrays

```
int numbers[5] = {10, 20, 30, 40, 50};

// Access elements
int first = numbers[0];    // 10
int third = numbers[2];    // 30
int last = numbers[4];     // 50

// Modify elements
numbers[1] = 25;          // Array becomes {10, 25, 30, 40, 50}

// Array size
int size = sizeof(numbers) / sizeof(numbers[0]); // 5
```

Array Loops

```
int scores[5] = {85, 92, 78, 96, 88};
int size = 5;

// Traditional for loop
for (int i = 0; i < size; i++) {
    cout << "Score " << i << ": " << scores[i] << endl;
}

// Range-based for loop (C++11)
for (int score : scores) {
    cout << "Score: " << score << endl;
```

```
}
```

Functions with Arrays

Passing Arrays to Functions

```
// Array parameter (size needed separately)
void printArray(int arr[], int size) {
    for (int i = 0; i < size; i++) {
        cout << arr[i] << " ";
    }
    cout << endl;
}

// Finding maximum in array
int findMax(int arr[], int size) {
    int max = arr[0];
    for (int i = 1; i < size; i++) {
        if (arr[i] > max) {
            max = arr[i];
        }
    }
    return max;
}

// Calculating sum
int calculateSum(int arr[], int size) {
    int sum = 0;
    for (int i = 0; i < size; i++) {
        sum += arr[i];
    }
    return sum;
}
```

Example Usage

```
int main() {
    int numbers[6] = {15, 8, 23, 42, 7, 19};
    int size = 6;

    printArray(numbers, size);

    int maximum = findMax(numbers, size);
    int total = calculateSum(numbers, size);
    double average = (double)total / size;

    cout << "Maximum: " << maximum << endl;
    cout << "Sum: " << total << endl;
    cout << "Average: " << average << endl;

    return 0;
}
```

String Arrays

```
// Array of strings
string cities[4] = {"New York", "London", "Tokyo", "Paris"};

// Function to print string array
void printCities(string arr[], int size) {
    for (int i = 0; i < size; i++) {
        cout << arr[i] << endl;
    }
}

// Function to find longest string
string findLongest(string arr[], int size) {
    string longest = arr[0];
    for (int i = 1; i < size; i++) {
        if (arr[i].length() > longest.length()) {
            longest = arr[i];
        }
    }
    return longest;
}
```

Common Patterns

Pattern 1: Search Array

```
bool findElement(int arr[], int size, int target) {
    for (int i = 0; i < size; i++) {
        if (arr[i] == target) {
            return true;
        }
    }
    return false;
}
```

Pattern 2: Count Elements

```
int countEven(int arr[], int size) {
    int count = 0;
    for (int i = 0; i < size; i++) {
        if (arr[i] % 2 == 0) {
            count++;
        }
    }
    return count;
}
```

Pattern 3: Reverse Array

```
void reverseArray(int arr[], int size) {
    for (int i = 0; i < size / 2; i++) {
```

```
        int temp = arr[i];
        arr[i] = arr[size - 1 - i];
        arr[size - 1 - i] = temp;
    }
}
```