

c programming exercises with solutions

C programming exercises with solutions are essential for anyone looking to enhance their coding skills in C. C is a powerful general-purpose programming language that offers a rich set of features for system programming, game development, and application development. Engaging in practical exercises not only solidifies theoretical knowledge but also equips programmers with problem-solving skills and the ability to write efficient code. This article will explore a variety of C programming exercises, along with their solutions, to help you practice and improve your C programming capabilities.

Understanding Basic Concepts

Before diving into exercises, it's crucial to understand some fundamental concepts of C programming. This section will lay the groundwork for the exercises that follow.

1. Data Types

C supports several data types, including:

- `int` for integers
- `float` for floating-point numbers
- `double` for double-precision floating-point numbers
- `char` for characters

2. Control Structures

Control structures dictate the flow of the program:

- Conditional statements: `if`, `else if`, `else`, `switch`
- Loops: `for`, `while`, `do while`

3. Functions

Functions in C help modularize code. A function consists of:

- A return type
- A name
- Parameters (if any)
- A body containing code

4. Arrays and Strings

Arrays store multiple values of the same type, while strings are arrays of characters ending with a null character (`\0`).

Exercise 1: Calculate the Factorial of a Number

Problem Statement: Write a program that calculates the factorial of a number entered by the user.

Solution:

```
``c
include

int factorial(int n) {
    if (n == 0) {
        return 1; // Base case
    }
    return n * factorial(n - 1); // Recursive call
}

int main() {
    int number;
    printf("Enter a number: ");
    scanf("%d", &number);

    if (number < 0) {
        printf("Factorial is not defined for negative numbers.\n");
    } else {
        printf("Factorial of %d is %d\n", number, factorial(number));
    }
    return 0;
}
``
```

Exercise 2: Find the Largest Element in an Array

Problem Statement: Create a program to find the largest element in a given array of integers.

Solution:

```
``c
```

```
include
```

```
int main() {  
int n, i, largest;  
printf("Enter the number of elements: ");  
scanf("%d", &n);  
  
int arr[n];  
printf("Enter %d integers:\n", n);  
for (i = 0; i < n; i++) {  
scanf("%d", &arr[i]);  
}  
  
largest = arr[0]; // Assume first element is the largest  
for (i = 1; i < n; i++) {  
if (arr[i] > largest) {  
largest = arr[i];  
}  
}  
  
printf("Largest element is %d\n", largest);  
return 0;  
}  
``
```

Exercise 3: Reverse a String

Problem Statement: Write a program to reverse a string entered by the user.

Solution:

```
``c  
include  
include  
  
int main() {  
char str[100], reversed[100];  
int i, j = 0;  
  
printf("Enter a string: ");  
fgets(str, sizeof(str), stdin); // Get user input including spaces  
str[strcspn(str, "\n")] = 0; // Remove newline character
```

```

for (i = strlen(str) - 1; i >= 0; i--) {
    reversed[j++] = str[i];
}
reversed[j] = '\0'; // Null-terminate the reversed string

printf("Reversed string: %s\n", reversed);
return 0;
}
'''

```

Exercise 4: Check for Prime Number

Problem Statement: Create a program to check if a number is prime.

Solution:

```

'''c
include

int is_prime(int n) {
    if (n <= 1) return 0;
    for (int i = 2; i i <= n; i++) {
        if (n % i == 0) return 0; // Not prime
    }
    return 1; // Prime
}

int main() {
    int number;
    printf("Enter a number: ");
    scanf("%d", &number);

    if (is_prime(number)) {
        printf("%d is a prime number.\n", number);
    } else {
        printf("%d is not a prime number.\n", number);
    }
    return 0;
}
'''

```

Exercise 5: Fibonacci Series

Problem Statement: Write a program to generate Fibonacci series up to n terms.

Solution:

```
```c
include

int main() {
int n, first = 0, second = 1, next;
printf("Enter the number of terms: ");
scanf("%d", &n);

printf("Fibonacci Series: %d, %d", first, second);
for (int i = 3; i <= n; i++) {
next = first + second;
printf(", %d", next);
first = second;
second = next;
}
printf("\n");
return 0;
}
```
```

Exercise 6: Count Vowels and Consonants

Problem Statement: Create a program to count vowels and consonants in a string.

Solution:

```
```c
include
include
include

int main() {
char str[100];
int vowels = 0, consonants = 0;

printf("Enter a string: ");
```

```

fgets(str, sizeof(str), stdin);
str[strcspn(str, "\n")] = 0; // Remove newline character

for (int i = 0; i < strlen(str); i++) {
 char ch = tolower(str[i]);
 if (ch >= 'a' && ch <= 'z') {
 if (ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o' || ch == 'u') {
 vowels++;
 } else {
 consonants++;
 }
 }
}

printf("Vowels: %d, Consonants: %d\n", vowels, consonants);
return 0;
}
```

```

Exercise 7: Sort an Array

Problem Statement: Write a program to sort an array of integers in ascending order using the bubble sort algorithm.

Solution:

```

```c
include

int main() {
 int n, i, j, temp;
 printf("Enter the number of elements: ");
 scanf("%d", &n);

 int arr[n];
 printf("Enter %d integers:\n", n);
 for (i = 0; i < n; i++) {
 scanf("%d", &arr[i]);
 }

 for (i = 0; i < n - 1; i++) {
 for (j = 0; j < n - i - 1; j++) {

```

```

if (arr[j] > arr[j + 1]) {
 // Swap arr[j] and arr[j + 1]
 temp = arr[j];
 arr[j] = arr[j + 1];
 arr[j + 1] = temp;
}
}
}

```

```

printf("Sorted array: ");
for (i = 0; i < n; i++) {
 printf("%d ", arr[i]);
}
printf("\n");
return 0;
}
```

```

Exercise 8: Palindrome Checker

Problem Statement: Write a program to check if a string is a palindrome.

Solution:

```

```c
include
include

int main() {
 char str[100], reversed[100];
 printf("Enter a string: ");
 fgets(str, sizeof(str), stdin);
 str[strcspn(str, "\n")] = 0; // Remove newline character

 strcpy(reversed, str);
 strrev(reversed); // Reverse the string

 if (strcmp(str, reversed) == 0) {
 printf("%s is a palindrome.\n", str);
 } else {
 printf("%s is not a palindrome.\n", str);
 }
}

```

```
return 0;
}
'''
```

## Conclusion

Engaging in C programming exercises with solutions is a practical way to develop your programming skills. The exercises presented in this article cover a range of topics, from basic operations to more complex algorithms. Practicing these problems will not only enhance your coding proficiency but also deepen your understanding of the fundamental concepts of C programming. As you progress, consider challenging yourself with more complex problems, and explore data structures and algorithms to further your knowledge. Remember, consistent practice is key to becoming a proficient programmer!

## Frequently Asked Questions

### What are some beginner-friendly C programming exercises?

Beginner-friendly exercises include writing a program to calculate the factorial of a number, creating a simple calculator, or implementing a program to swap two numbers using a temporary variable.

### How can I find the largest number in an array using C?

You can iterate through the array using a loop, comparing each element to a variable that holds the largest value found so far, updating it as necessary.

### What is a common exercise for understanding pointers in C?

A common exercise is to create a function that swaps two integers using pointers, which helps reinforce the concept of memory addresses in C.

### Can you provide a solution for reversing a string in C?

To reverse a string, you can use two pointers: one at the start and one at the end of the string, swapping characters while moving the pointers towards the center.

### What is a good exercise for practicing file handling in C?

A good exercise is to write a program that reads data from a file, processes it (like counting the number of lines), and writes the results to another file.



## How do I implement a simple linked list in C?

You can create a struct for the linked list nodes, then implement functions for adding, deleting, and displaying nodes to practice linked list operations.

## What C program can help me understand recursion?

A classic exercise is to write a recursive function to compute the Fibonacci sequence, demonstrating how recursion can simplify certain problems.

## How can I create a multiplication table in C?

You can use nested loops: an outer loop for the rows (1 to 10) and an inner loop for the columns (1 to 10), multiplying the row and column indices to fill the table.

## What is an exercise for sorting algorithms in C?

Implementing bubble sort or quicksort on an array of integers is a great way to practice sorting algorithms, allowing you to compare efficiency and performance.

## How do I create a basic ATM program in C?

You can create an ATM program by using conditional statements to handle different operations like 'check balance', 'deposit', and 'withdraw', while maintaining user account information.

## [C Programming Exercises With Solutions](#)

C Programming Exercises With Solutions

## Related Articles

- [california business law and legal environment 5th edition](#)
- [caliphate definition world history](#)
- [c s lewis and the arts creativity in the shadowlands](#)

[Back to Home](#)