Here is a basic syntax for a C program:

```
#include <stdio.h> // header file
int main() {
   // main function
   printf("Hello, World!"); // output statement
   return 0; // exit status
}
```

Output:

```
Hello, World!
```

Here’s a breakdown of the syntax:

include – This is a preprocessor directive that includes the standard input/output library in the program.

int main() – This is the main function where the program execution starts.

{ and } – These curly braces enclose the statements that are executed as part of the main function.

printf(“Hello, World!”); – This is a function call that prints “Hello, World!” to the console.

return 0; – This statement exits the program with a status of 0, indicating successful execution.

---

# C syntax include:

## 1. Structure of a C Program:

- A C program consists of functions.
- The main function, main(), is the entry point of the program.

## 2. Comments:

- Comments are used to add explanatory text to the code.
- Single-line comments start with //.
- Multi-line comments start with /* and end with */.

## 3. Data Types:

- C has built-in data types, such as int, float, double, char, etc.
- You can define custom data types using struct, union, or enum.

## 4. Variables:

- Variables are used to store data.
- They need to be declared with a data type before use.
- Syntax: ;

## 5. Constants:

- Constants are fixed values that cannot be modified during program execution.
- They can be defined using #define or const.
- Syntax with #define: #define
- Syntax with const: const = ;

## 6. Input and Output:

- The printf() function is used for output.
- Syntax: printf(“”, )
- The scanf() function is used for input.
- Syntax: scanf(“”, &)

## 7. Control Flow:

- Conditional statements are used for decision making.

## The if statement:

```
if (<condition>) {
    // code to execute if condition is true
} else {
    // code to execute if condition is false
}
```

## The switch statement:

```
switch (<expression>) {
    case <value1>:
        // code to execute if expression matches value1
        break;
    case <value2>:
        // code to execute if expression matches value2
        break;
    default:
        // code to execute if expression does not match any value
}
```

## 8. Loops:

- Loops are used for repetitive execution.

## The while loop:

```
while (<condition>) {
    // code to execute repeatedly
}
```

## The for loop:

```
for (<initialization>; <condition>; <increment>) {
    // code to execute repeatedly
}
```

## The do-while loop:

```
do {
    // code to execute repeatedly
} while (<condition>);
```

## 9. Functions:

- Functions are reusable blocks of code.
- They have a return type, a name, optional parameters, and a body.
- Syntax:<return_type> <function_name>(<parameters>) { <function_body> }

## 10. Arrays:

- Arrays store multiple values of the same data type.
- Syntax: <data_type> <array_name>[<size>];

## 11. Pointers:

- Pointers store memory addresses.
- They are declared using the * symbol.
- Syntax: <data_type> *<pointer_name>;

## Practice Problems on Syntax:

## Problem 1:

Write a C program that calculates the factorial of a given number. The program should prompt the user to enter a positive integer and display its factorial.

```
#include <stdio.h>

int main() {
    int num, factorial = 1, i;
    printf("Enter a positive integer: ");
    scanf("%d", &num);
    for (i = 1; i <= num; i++) {
        factorial *= i;
    }
    printf("The factorial of %d is: %d\n", num, factorial);
    return 0;
}
```

Explanation: In this program, we declare three variables of type int: num, factorial, and i. The user is prompted to enter a positive integer using scanf(). A for loop is used to iterate from 1 to num and calculate the factorial by multiplying the current value of factorial with the loop variable i. Finally, the factorial is displayed using printf().

Output:

```
Enter a positive integer: 4
The factorial of 4 is: 24
```

## Problem 2:

Write a C program that reads a character from the user and determines whether it is an uppercase letter, lowercase letter, or a digit.

```
#include <stdio.h>

int main() {
    char ch;
    printf("Enter a character: ");
    scanf(" %c", &ch);
    if (ch >= 'A' && ch <= 'Z') {
        printf("Uppercase letter\n");
    } else if (ch >= 'a' && ch <= 'z') {
        printf("Lowercase letter\n");
    } else if (ch >= '0' && ch <= '9') {
        printf("Digit\n");
    } else {
        printf("Other character\n");
    }
    return 0;
}
```

Explanation: In this program, we declare a variable ch of type char. The user is prompted to enter a character using scanf(). The program then uses multiple if statements to check the range of the character and determines whether it is an uppercase letter, lowercase letter, digit, or any other character. The appropriate message is displayed using printf().

Output:

```
Enter a character: j
Lowercase letter
```

---

## Problem 3:

Write a C program that prints the Fibonacci series up to a given number. The program should prompt the user to enter a positive integer and display the Fibonacci series.

```
#include <stdio.h>
int main() {
    int num, prev = 0, current = 1, next;

    printf("Enter a positive integer: ");
    scanf("%d", &num);

    printf("Fibonacci series up to %d: %d, %d, ", num, prev,
current);

    while (prev + current <= num) {
        next = prev + current;
        printf("%d, ", next);
        prev = current;
        current = next;
    }
```

```
    printf("\n");

    return 0;
}
```

Explanation: In this program, we declare variables of type int: num, prev, current, and next. The user is prompted to enter a positive integer using scanf(). The program prints the initial two numbers of the Fibonacci series, which are 0 and 1. Then, using a while loop, it calculates the next Fibonacci number by adding the previous two numbers and updates the variables accordingly. The loop continues until the next Fibonacci number is less than or equal to the given number. Finally, the Fibonacci series is displayed using printf().

Output:

```
Enter a positive integer: 5
Fibonacci series up to 5: 0, 1, 1, 2, 3, 5,
```

## 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. Comments in C
32. Variables in C
33. Data types in C
34. Format specifiers in C
35. Type Conversion in C

36. Constants in C
37. Operators in C
38. Pre and Post Increament Practice Problems
39. Pre and Post Increament
40. Array in C
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