RegNo: 2021/ICT/113

Conditional Statements: Ternary Operator

The ternary operator (? :) is a shorthand for the if-else statement, often used for concise conditional expressions.

Syntax:

test_condition ? expression1 : expression2;

If test_condition is true, expression1 is executed; otherwise, expression2 is executed.

Example: Election Voting Eligibility (Ternary Operator)

```
// ternaryop.c
#include

int main() {
    int age;

    printf("Enter your age: ");
    scanf("%d", &age);
    (age >= 18) ? printf("You are eligible for election voting\n") : printf("You are not eligible for election voting\n");
    return 0;
}
```

Execution:

```
[2021ict113@fedora ~]$ touch ternaryop.c
[2021ict113@fedora ~]$ vi ternaryop.c
[2021ict113@fedora ~]$ gcc ternaryop.c -o ternaryop
[2021ict113@fedora ~]$ ./ternaryop
Enter your age: 23
You are eligible for election voting
```

Switch Statement in C

The switch statement specifies many alternative blocks of code to be executed based on the value of a variable or expression.

Syntax:

```
switch(variable/expression) {
    case 1: // body of case 1
        break;
    case 2: // body of case 2
        break;
    case n: // body of case n
        break;
    default: // body of default
}
```

Example 1: Weekdays Program

```
// weekdays.c
#include

int main() {
    int day;

    printf("Enter the number between 1 to 7: ");
    scanf("%d", &day);
    switch(day) {
        case 1: printf("Today is Sunday!\n"); break;
        case 2: printf("Today is Monday!\n"); break;
        case 3: printf("Today is Tuesday!\n"); break;
        case 4: printf("Today is Wednesday!\n"); break;
        case 5: printf("Today is Thursday!\n"); break;
        case 6: printf("Today is Friday!\n"); break;
        case 7: printf("Today is Saturday!\n"); break;
        default: printf("Invalid input! Please enter a number
between 1 and 7.\n");
    }
    return 0;
}
```

Example 2: Astrology Program

```
// astrology.c
#include

int main() {
    int date, a, b, lifePath;

    printf("Enter your birth date (DD): ");
    scanf("%d", &date);
    a = date % 10;
    b = date / 10;
    lifePath = a + b;
    // Ensure life path number is between 1-9
    if (lifePath > 9) {
        lifePath = (lifePath % 10) + (lifePath / 10);
    }
    printf("Your Life Path Number is: %d\n", lifePath);
    printf("Meaning: ");
    switch(lifePath) {
        case 1: printf("Lucky\n"); break;
        case 2: printf("Carefully do your work\n"); break;
        case 3: printf("Stronger\n"); break;
        case 4: printf("Happy\n"); break;
        case 5: printf("Can get help\n"); break;
        case 6: printf("Doubt\n"); break;
        case 7: printf("Sad\n"); break;
        case 8: printf("Like\n"); break;
        case 9: printf("Courage\n"); break;
        default: printf("Invalid life path number\n");
    }
    return 0;
}
```

Execution:

```
[2021ict113@fedora ~]$ touch astrology.c
[2021ict113@fedora ~]$ vi astrology.c
[2021ict113@fedora ~]$ gcc astrology.c -o astrology
[2021ict113@fedora ~]$ ./astrology
Enter your birth date (DD): 23
Your Life Path Number is: 5
Meaning: Can get help
```

Loop Statements: For Loop

Loop statements execute a block of code repeatedly as long as a specified condition is true.

For Loop

The for loop is typically used when the number of iterations is known.

Example: Sum and Multiplication Calculator

```
// sum_mul.c
#include

int main() {
    int n, i, sum = 0, mul = 1;

    printf("Enter the number of elements: ");
    scanf("%d", &n);
    int arr[n]; // Declaring a Variable Length Array (VLA)
    printf("Enter %d numbers: ", n);
    for(i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }
    for(i = 0; i < n; i++) {
        sum += arr[i];
        mul *= arr[i];
    }
    printf("Summation = %d\n", sum);
    printf("Multiplication = %d\n", mul);
    return 0;
}
```

Execution:

```
[2021ict113@fedora ~]$ touch sum_mul.c
[2021ict113@fedora ~]$ vi sum_mul.c
[2021ict113@fedora ~]$ gcc sum_mul.c -o sum_mul
[2021ict113@fedora ~]$ ./sum_mul
Enter the number of elements: 5
Enter 5 numbers: 1 2 3 4 5
Summation = 15
Multiplication = 120
```

While Loop in C Programming

The while loop is used when the number of iterations is not known beforehand and the loop continues as long as a condition remains true.

Example: Print Numbers in Range

```
// print_numbers.c
#include

int main() {
    int ao, i = 1;

    printf("Enter the upper limit (ao): ");
    scanf("%d", &ao);
    while (i <= ao) {
        printf("%d\n", i);
        i++;
    }
    return 0;
}
```

Execution:

```
[2021ict113@fedora ~]$ touch print_numbers.c
[2021ict113@fedora ~]$ vi print_numbers.c
[2021ict113@fedora ~]$ gcc print_numbers.c -o print_numbers
[2021ict113@fedora ~]$ ./print_numbers
Enter the upper limit (ao): 10
1
2
3
4
5
6
7
8
9
10
```

Advanced C Programming: Fibonacci Series

Write a c program to generate and print the fibonacci series up to a specifield number of terms. The program should take the number of terms as input from the user and then display the corresponding Fibonacci sequence.

```
// fibonacci.c
#include

int main() {
    int n, a = 0, b = 1, nextTerm;

    printf("Enter the number of terms: ");
    scanf("%d", &n);
    printf("Fibonacci Series: ");
    for (int i = 1; i <= n; i++) {
        printf("%d ", a);
        nextTerm = a + b;
        a = b;
        b = nextTerm;
    }
    printf("\n");
    return 0;
}
```

Execution:

```
[2021ict113@fedora ~]$ touch fibonacci.c
[2021ict113@fedora ~]$ vi fibonacci.c
[2021ict113@fedora ~]$ gcc fibonacci.c -o fibonacci
[2021ict113@fedora ~]$ ./fibonacci
Enter the number of terms: 10
Fibonacci Series: 0 1 1 2 3 5 8 13 21 34
```

Factorial Calculation in C

Write a C program to calculate the factorial of a given non-negative integer.

```
// factorial.c
#include

int main() {
    int n, i;
    unsigned long long fact = 1; // Use unsigned long long for larger factorials

    printf("Enter a non-negative integer: ");
    scanf("%d", &n);
    if (n < 0) {
        printf("Error! Factorial of a negative number doesn't exist.\n");
    } else {
        for (i = 1; i <= n; i++) {
            fact *= i;
         }
        printf("Factorial of %d = %llu\n", n, fact);
    }
    return 0;
}
```

Execution:

```
[2021ict113@fedora ~]$ touch factorial.c
[2021ict113@fedora ~]$ vi factorial.c
[2021ict113@fedora ~]$ gcc factorial.c -o factorial
[2021ict113@fedora ~]$ ./factorial
Enter a non-negative integer: 5
Factorial of 5 = 120
```

1 2 3 4

Input

Get non-negative integer

Validate

Ensure number is non-negative

Calculate

Multiply all integers from 1 to n

Display

Show the factorial result

String Concatenation in C

Write a C program that; Accepts two strings as input from the user. Concatenates the two strings Displays the concatenated result.

```
// concat.c
#include
#include  // Required for strcat and strcspn

int main() {
    char str1[100], str2[100]; // Declare character arrays to store strings

    printf("Enter first string: ");
    fgets(str1, sizeof(str1), stdin); // Use fgets for safer input
    printf("Enter second string: ");
    fgets(str2, sizeof(str2), stdin);
    // Remove newline characters read by fgets
    str1[strcspn(str1, "\n")] = 0;
    str2[strcspn(str2, "\n")] = 0;
    strcat(str1, str2); // Concatenate str2 to str1
    printf("Concatenated String: %s\n", str1);
    return 0;
}
```

Execution:

```
[2021ict113@fedora ~]$ touch concat.c
[2021ict113@fedora ~]$ vi concat.c
[2021ict113@fedora ~]$ gcc concat.c -o concat
[2021ict113@fedora ~]$ ./concat
Enter first string: Hello
Enter second string: Worls!
Concatenated String: HelloWorls!
```

Binary to Decimal Conversion in C

```
// binary_decimal.c
#include

int binaryToDecimal(int n) {
    int decimal = 0, base = 1, last_digit;

    while (n > 0) {
        last_digit = n % 10; // Get the last digit of the binary number
        n = n / 10; // Remove the last digit
        decimal += last_digit * base; // Add the product of the digit and its base value to decimal
        base *= 2; // Update the base for the next digit (powers of 2)
    }
    return decimal;
}

int main() {
    int binary;

    printf("Enter a binary number: ");
    scanf("%d", &binary);
    printf("Decimal equivalent: %d\n", binaryToDecimal(binary));
    return 0;
}
```

Execution:

```
[2021ict113@fedora ~]$ touch binary_decimal.c
[2021ict113@fedora ~]$ vi binary_decimal.c
[2021ict113@fedora ~]$ gcc binary_decimal.c -o binary_decimal
[2021ict113@fedora ~]$ ./binary_decimal
Enter a binary number: 1101
Decimal equivalent: 13
```