Understanding the For Loop in C

Yasin Musfic Hussain

1st Semester MCA
Roll No.: 03

November 8, 2024

Contents

Introduction to "For Loop" in C

- The **for** loop is a control flow statement in C.
- Used for repeating a block of code a specific number of times.
- Commonly used for iterating over arrays, performing calculations, and controlling program flow.

For Loop Syntax

Syntax

```
for (initialization; condition; update)
{
Code to execute
}
```

- **initialization**: Sets the starting value of the loop variable.
- **condition**: Determines if the loop should continue.
- **update**: Modifies the loop variable after each iteration.

How the For Loop Works

1. **Initialization:** Executed once, before the loop begins.
2. **Condition Check:** Evaluated before each iteration.
3. **Execution:** If the condition is true, the loop executes its statements.
4. **Update:** The loop variable is updated, then the condition is checked again.

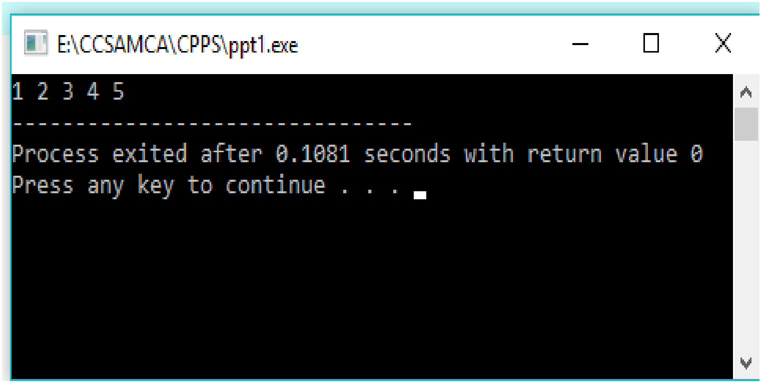
Example Code 1

Example: Print numbers 1 to 5

```
#include <stdio.h>
int main()
{
    int i;
    for (i = 1; i <= 5; i++)
    {
        printf("%d ", i);
    }
    return 0;
}
```

- This loop prints numbers from 1 to 5.

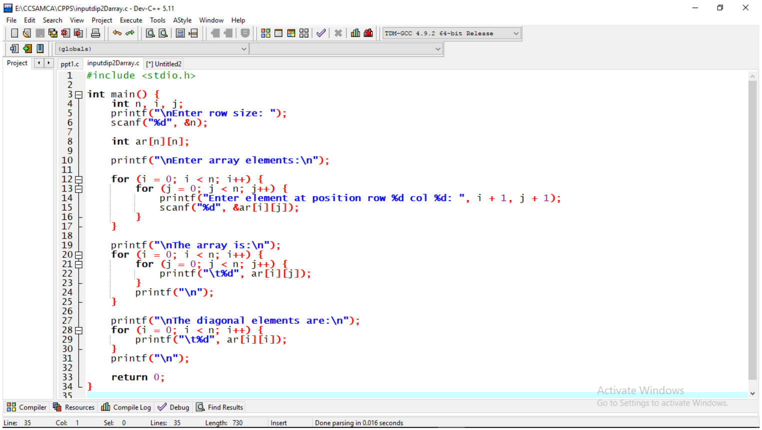
Output



```
1 2 3 4 5
--------------------------------
Process exited after 0.1081 seconds with return value 0
Press any key to continue . . .
```

Figure: This is the output for the above code

Example Code 2

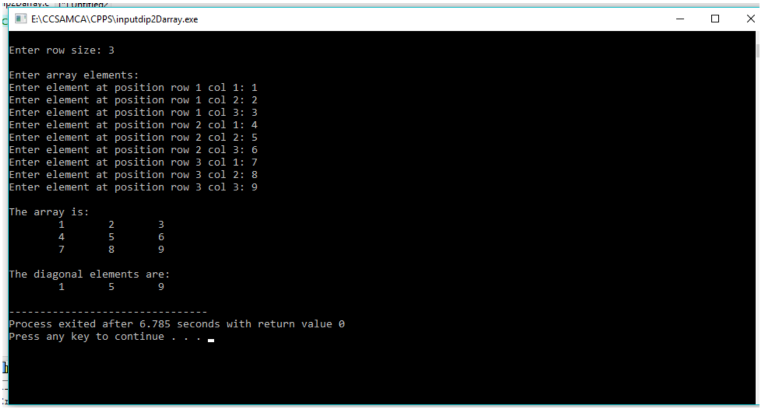


E:\CCSAMCA\CPPS\inputdip2Darray.c - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
TDM-GCC 4.9.2 64-bit Release
(globals)
Project
ppt1.c
inputdip2Darray.c
[*] Untitled2
Compiler
Resources
Compile Log
Debug
Find Results
Activate Windows
Go to Settings to activate Windows.

```c
#include <stdio.h>

int main() {
    int n, i, j;
    printf("\nEnter row size: ");
    scanf("%d", &n);

    int ar[n][n];

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

    for (i = 0; i < n; i++) {
        for (j = 0; j < n; j++) {
            printf("Enter element at position row %d col %d: ", i + 1, j + 1);
            scanf("%d", &ar[i][j]);
        }
    }

    printf("\nThe array is:\n");
    for (i = 0; i < n; i++) {
        for (j = 0; j < n; j++) {
            printf("\t%d", ar[i][j]);
        }
        printf("\n");
    }

    printf("\nThe diagonal elements are:\n");
    for (i = 0; i < n; i++) {
        printf("\t%d", ar[i][i]);
    }
    printf("\n");

    return 0;
}
```

```
Line: 35    Col: 1    Sel: 0    Lines: 35    Length: 730    Insert    Done parsing in 0.016 seconds
```

Figure: Code for taking input and displaying a 2D array and its diagonal elements



```
Enter row size: 3

Enter array elements:
Enter element at position row 1 col 1: 1
Enter element at position row 1 col 2: 2
Enter element at position row 1 col 3: 3
Enter element at position row 2 col 1: 4
Enter element at position row 2 col 2: 5
Enter element at position row 2 col 3: 6
Enter element at position row 3 col 1: 7
Enter element at position row 3 col 2: 8
Enter element at position row 3 col 3: 9

The array is:
        1       2       3
        4       5       6
        7       8       9

The diagonal elements are:
        1       5       9

--------------------------------
Process exited after 6.785 seconds with return value 0
Press any key to continue . . .
```

Figure: Output displaying the 2D array and its diagonal elements

Uses of For Loop

- **Iterating over arrays:** Access each element of an array.
- **Repeating calculations:** Perform repetitive calculations.
- **Control structures:** Create complex program flows.

Real-life Implementations

- **Data Processing:** Process each item in a dataset.
- **Game Development:** Update positions, detect collisions.
- **Web Applications:** Render UI elements dynamically.

Advantages and Limitations

- **Advantages:**
  - Easy to understand and implement.
  - Efficiently manages repetitive tasks.
- **Limitations:**
  - Can lead to infinite loops if the condition is not managed properly.
  - Limited by data types and range.

Summary

- The `for` loop is a fundamental concept in C programming.
- It is widely used for repetitive tasks.
- Important for iterating through data and managing program flow.