I/O and Data Types

Ashish Dahal

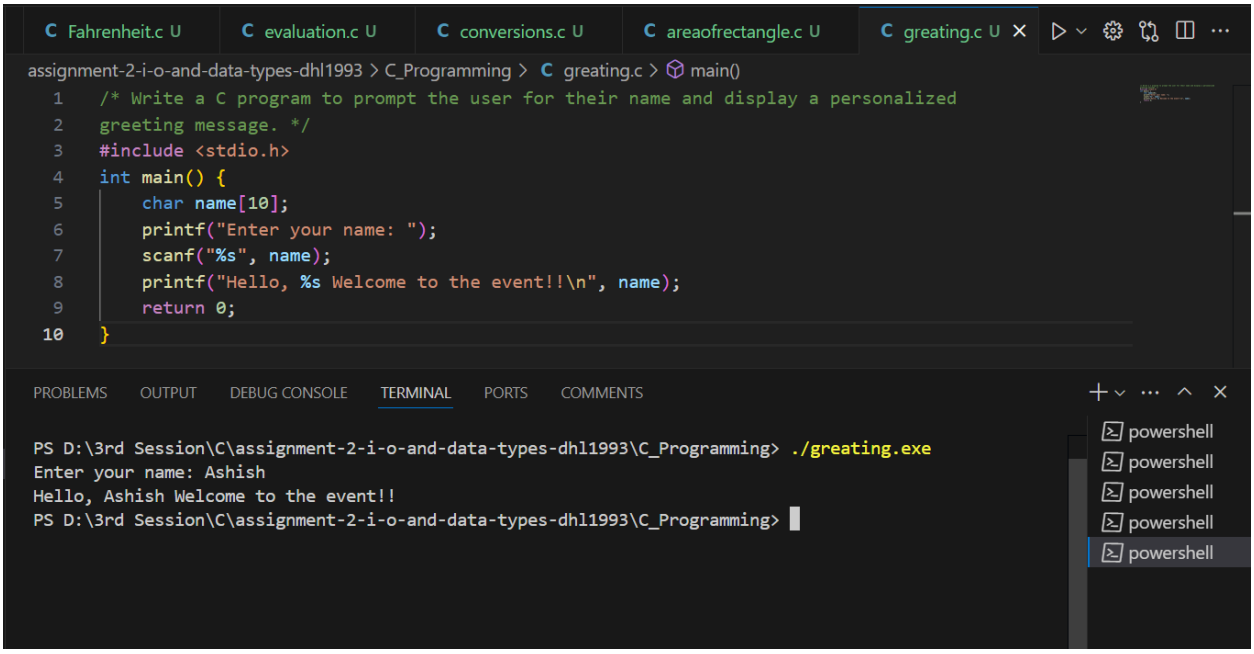
Westcliff University

C Programming

Professor Thapa

March 30, 2025

1. **Write a C program to prompt the user for their name and display a personalized greeting message.**



This code simply prompts the user to enter their name and greet them with the welcome message as shown in console window.

2. **Explain the concept of type conversions in C. Provide examples of implicit and explicit type conversions.**

Type conversion is the process of transforming a value from one data type to another in C. There are two sorts of conversions.

The compiler performs implicit conversions whenever different data types are involved in an expression. It follows a hierarchy to prevent data loss. Convert small types into larger types. The programmer performs explicit conversions manually using the cast operator

(type). This conversion is utilized when implicit conversion produces undesirable results.

```c
// Provide examples of implicit and explicit type conversions.
#include <stdio.h>
int main() {
    // Implicit type conversion
    int a = 5;
    double b = 2.5;
    double result1 = a + b;
    printf("Implicit conversion result: %f\n", result1);
    // Explicit type conversion
    double c = 7.8;
    int d = (int)c;
    printf("Explicit conversion result: %d\n", d);
    return 0;
}
```

```
[Running] cd "d:\3rd Session\C\assignment-2-i-o-and-data-types-dhl1993\C_Programming\" && gcc conversions.c -o
conversions && "d:\3rd Session\C\assignment-2-i-o-and-data-types-dhl1993\C_Programming\"conversions
Implicit conversion result: 7.500000
Explicit conversion result: 7

[Done] exited with code=0 in 2.002 seconds
```

In the above code we demonstrate implicit and explicit conversion and output displayed in the console shows the difference between them.

3. **Write a C program to calculate the area of a rectangle. Prompt the user to enter the length and width and display the result.**

```c
/* Write a C program to calculate the area of a rectangle. Prompt the user to enter the
length and width, and display the result.*/
#include <stdio.h>
int main() {
    float length, breadth, area;
    printf("Enter the length of the rectangle: ");
    scanf("%f", &length);
    printf("Enter the breadth of the rectangle: ");
    scanf("%f", &breadth);
    area = length * breadth;
    printf("The area of the rectangle is: %.2f\n", area);
    return 0;
}
```

```
PS D:\3rd Session\C\assignment-2-i-o-and-data-types-dhl1993\C_Programming> ./fahrenheit.exe
Enter temperature in Celsius: 100
Temperature in Fahrenheit is: 212.000000

PS D:\3rd Session\C\assignment-2-i-o-and-data-types-dhl1993\C_Programming> ./areaofrectangle.exe
Enter the length of the rectangle: 5
Enter the breadth of the rectangle: 5
The area of the rectangle is: 25.00
PS D:\3rd Session\C\assignment-2-i-o-and-data-types-dhl1993\C_Programming>
```

In above code user must just input length and breadth of rectangle and total area of rectangle will be displayed in console.

4. **Write a C program to convert temperature from Celsius to Fahrenheit. Prompt the user for a temperature in Celsius and display the equivalent temperature in Fahrenheit.**

```c
/* Write a C program to convert temperature from Celsius to Fahrenheit. Prompt the user
for a temperature in Celsius and display the equivalent temperature in Fahrenheit. */
#include <stdio.h>
int main() {
    float celsius, fahren;
    printf("Enter temperature in Celsius: ");
    scanf("%f", &celsius);
    fahren = (celsius * 9/5) + 32;
    printf("Temperature in Fahrenheit is: %f\n ", fahren);
    return 0;
}
```

```
included, verify that the path is correct and try again.
At line:1 char:1
+ ./faherenheit.exe
+ ~~~~~~~~~~~~~~~~~
    + CategoryInfo          : ObjectNotFound: (./faherenheit.exe:String) [], CommandNotFoundExceptio
   n
    + FullyQualifiedErrorId : CommandNotFoundException

PS D:\3rd Session\C\assignment-2-i-o-and-data-types-dhl1993\C_Programming> ./fahrenheit.exe
Enter temperature in Celsius: 100
Temperature in Fahrenheit is: 212.000000

PS D:\3rd Session\C\assignment-2-i-o-and-data-types-dhl1993\C_Programming>
```

In the above code user must enter the temperature in Celsius and it will be converted to Fahrenheit and the output will be displayed in console.

5. **Write a C program to swap the values of two variables using a temporary variable.**

```c
//  Write a C program to swap the values of two variables using a temporary variable.
#include <stdio.h>
int main() {
    int a, b, temp;
    printf("Enter two numbers: ");
    scanf("%d %d", &a, &b);
    printf("Before swapping: a = %d, b = %d\n", a, b);
    // swapping values using a temporary variable
    temp = a;
    a = b;
    b = temp;
    printf("After swapping: a = %d, b = %d\n", a, b);
    return 0;
}
```

```
PS D:\3rd Session\C\assignment-2-i-o-and-data-types-dhl1993\C_Programming> ./temporary_variable.exe
Enter two numbers: 2
5
Before swapping: a = 2, b = 5
After swapping: a = 5, b = 2
PS D:\3rd Session\C\assignment-2-i-o-and-data-types-dhl1993\C_Programming>
```

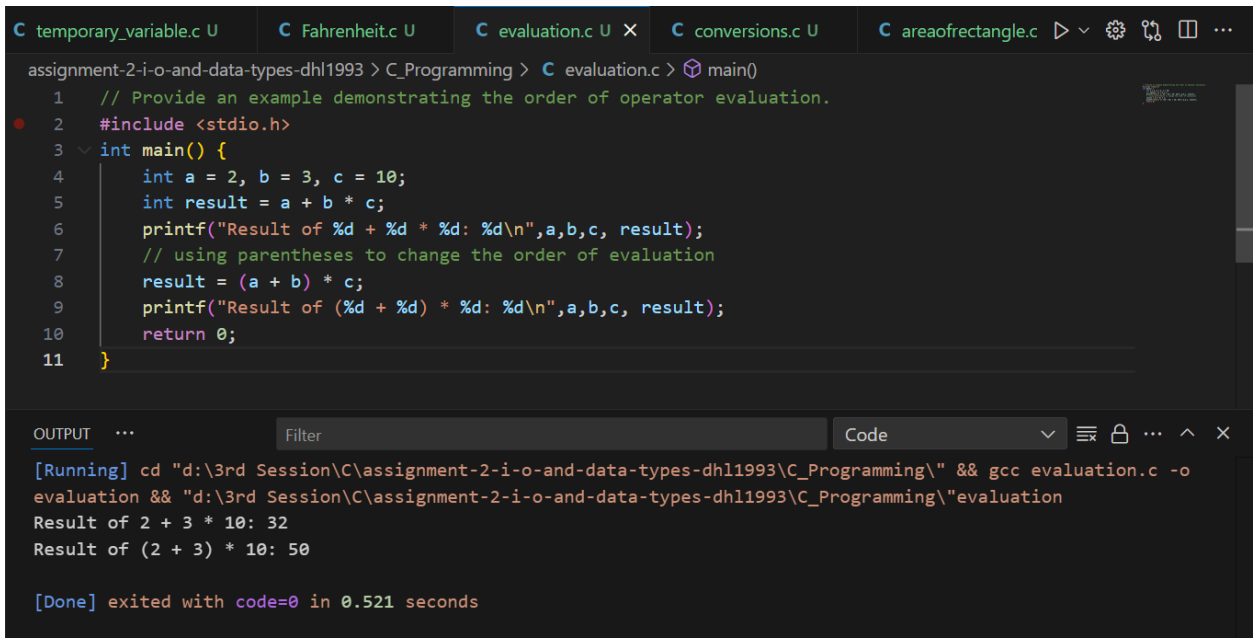
In the above code we are swapping the values of variable **a** and **b** with the help of third variable **temp**. As in console you can see the values before and after swapping.

6. **Explain the concept of operator precedence and associativity in C. Provide an example demonstrating the order of operator evaluation**.

In C, the operator prece

dence tells the computer which action to do first in an expression. For example, in Figure 6, the multiplication symbol (*) comes before the addition symbol (+). This means that when you type in **a + b * c**, it first figures out **b*c** and then adds the result to a to get the result. The parentheses () can be used to change precedence and make them do a certain evaluation first.

When two operators in C have the same amount of precedence, associativity helps decide the order in which they should be evaluated. Left-associative operators, which are ones like +, -, *, and /, are evaluated from left to right. For instance, **2/3/10** turns into **(2/3) /10**. However, assignment (=) and unary operators (++, --) work right associative, which means they work from left to right. Let's say **a = b = 3**. In this case, **3** is given to **b** first, then to **a**.



The above code simply demonstrates how we can simply override the operator precedence using parenthesis and operator evaluation. In first expression it first calculates product of **b** and **c** and add the result to **a** whereas in second expression it first adds **a** and **b** and multiply the result with **c** to give final result.