C Program: Unit Converter

```c
#include <stdio.h>

void convertMetersToKilometers(float meters) {
    float kilometers = meters / 1000;
    printf("%.2f meters = %.4f kilometers\n", meters, kilometers);
}

void convertGramsToKilograms(float grams) {
    float kilograms = grams / 1000;
    printf("%.2f grams = %.4f kilograms\n", grams, kilograms);
}

void convertCentimetersToMeters(float centimeters) {
    float meters = centimeters / 100;
    printf("%.2f centimeters = %.4f meters\n", centimeters, meters);
}

void convertMillilitersToLiters(float milliliters) {
    float liters = milliliters / 1000;
    printf("%.2f milliliters = %.4f liters\n", milliliters, liters);
}

int main() {
    int choice;
    float value;

    printf("Unit Converter Program\n");
    printf("------------------------\n");
    printf("Select the conversion type:\n");
    printf("1. Meters to Kilometers\n");
    printf("2. Grams to Kilograms\n");
    printf("3. Centimeters to Meters\n");
    printf("4. Milliliters to Liters\n");
    printf("Enter your choice (1-4): ");
    scanf("%d", &choice);

    // Take the input value based on user's choice
    printf("Enter the value to convert: ");
    scanf("%f", &value);

    // Perform the conversion based on the user's choice
    switch (choice) {
        case 1:
            convertMetersToKilometers(value);
            break;
        case 2:
            convertGramsToKilograms(value);
            break;
        case 3:
            convertCentimetersToMeters(value);
            break;
        case 4:
            convertMillilitersToLiters(value);
            break;
        default:
            printf("Invalid choice! Please choose a valid option (1-4).\n");
    }

    return 0;
}
```

Explanation of the Program:

1. **Functions for Conversion**:
    - Each unit conversion (e.g., meters to kilometers, grams to kilograms) is handled by a separate function. These functions take a floating-point number as input and perform the necessary conversion, then print the result.
2. **User Input**:

    - The program first asks the user to choose the type of conversion they want to perform (1–4).
    - After the user makes a selection, it asks for the value to convert.
3. **Switch Case for Conversion**:
    - Based on the user's choice, the program calls the appropriate conversion function. If the user enters an invalid choice, the program will display an error message.
4. **Output**:
    - The program prints the result of the conversion in a user-friendly format with two decimal places for the input value and four decimal places for the result.

Example Output:

```
Unit Converter Program
------------------------
Select the conversion type:
1. Meters to Kilometers
2. Grams to Kilograms
3. Centimeters to Meters
4. Milliliters to Liters
Enter your choice (1-4): 1
Enter the value to convert: 1500
1500.00 meters = 1.5000 kilometers
```

To Compile and Run:

1. Save the program to a file, for example `unit_converter.c`.
2. Open your terminal or command prompt.
3. Compile the program with `gcc` (or any C compiler):
   bash
   Copy code

Adding More Conversions:

You can easily extend this program by adding more conversion functions. Just define a new function and add a corresponding case in the `switch` statement for each new conversion type you'd like to implement

Logic for Unit Conversion Program

The logic behind the **unit conversion program** revolves around:

1. **User Interaction**: The program interacts with the user to decide what type of conversion they want to perform.
2. **Input Handling**: After selecting the conversion type, the user provides a numerical input (the value to convert).
3. **Conversion Calculation**: The selected conversion type is mapped to a mathematical formula that computes the converted value.
4. **Output**: The program then prints the result of the conversion in a readable format.

Let's break down the logic step-by-step:

Step 1: Display the Menu

The program starts by displaying a list of conversion options, allowing the user to choose which type of unit conversion they want to perform.

- **Prompt the user** to select one of the available options:
  - Meters to Kilometers
  - Grams to Kilograms
  - Centimeters to Meters
  - Milliliters to Liters

Step 2: Input the Value to Convert

Once the user selects a conversion type, the program prompts the user to input the **value** they want to convert.

- **Store the input** value in a variable. This is typically a floating-point variable (`float`) because unit values can have decimal points (e.g., 1.5 meters, 100.25 grams).

Step 3: Perform the Conversion

The program performs the actual conversion using a **mathematical formula**. Here are the formulas for each type of conversion:

- **Meters to Kilometers**:
  kilometers=meters1000\text{kilometers} =
  \frac{\text{meters}}{1000}kilometers=1000meters
  (Since 1 kilometer = 1000 meters)
- **Grams to Kilograms**:
  kilograms=grams1000\text{kilograms} =
  \frac{\text{grams}}{1000}kilograms=1000grams
  (Since 1 kilogram = 1000 grams)
- **Centimeters to Meters**:
  meters=centimeters100\text{meters} =
  \frac{\text{centimeters}}{100}meters=100centimeters
  (Since 1 meter = 100 centimeters)
- **Milliliters to Liters**:
  liters=milliliters1000\text{liters} =
  \frac{\text{milliliters}}{1000}liters=1000milliliters
  (Since 1 liter = 1000 milliliters)

The conversion formulas are simple, just dividing by the appropriate factor (1000, 100, etc.) to convert from a smaller unit to a larger unit.

Step 4: Display the Result

After the calculation, the program prints the result of the conversion in a human-readable format.

- The original input value is printed along with the converted value.
- Use **two decimal points** to display the input value and **four decimal places** for the result, which ensures the output is precise and easy to read.

Step 5: Handle Invalid Input

The program should handle cases where the user enters an invalid conversion choice. If the user selects an option that's not available (e.g., entering a number outside the range of valid choices), the program will print an error message.

Overall Program Flow:

1. **Display menu options** for the user to choose the type of conversion.
2. **Accept the user's choice** of conversion type.
3. **Prompt the user for the input value** to convert.
4. **Perform the conversion** using a corresponding formula based on the selected type.
5. **Print the result** of the conversion.
6. **Handle invalid input** gracefully by using a `switch-case` structure.