Programming with C and C++

CSC-101 (Lecture 5)

Dr. R. Balasubramanian

Professor
Department of Computer Science and Engineering
Mehta Family School of Data Science and Artificial Intelligence
Indian Institute of Technology Roorkee
Roorkee 247 667

bala@cs.iitr.ac.in
https://faculty.iitr.ac.in/cs/bala/

Unfolding some more layers

Another Task: Circle's Area and Circumference

- Given the radius, calculate the area and circumference.

```
#include <stdio.h>
#include <math.h>

int main() {
    float radius, area, circumference;
    printf("Enter the radius of the circle:
        ");
    scanf("%f", &radius);

    area = M_PI * radius * radius;
    circumference = 2 * M_PI * radius;

    printf("Area = %f, Circumference = %f\n",
        area, circumference);
    return 0;
}
```

Dissecting the Program: Including the Math Library

New Directive:

```
#include <math.h>
```

- Introduces the mathematics library in our program.
- Offers predefined constants and functions, like *M_PI* for the value of π.
- Think of it as adding an advanced mathematical toolkit to our collection.

Dissecting the Program: Taking User Input

Interactivity:

```
printf("Enter the radius of the circle: ");
scanf("%f", &radius);
```

- ‘printf’ prompts the user to enter a value.
- ‘scanf’ captures the user input.
- `%f` tells ‘scanf’ we’re expecting a floating-point number.
- The & before ‘radius’ is a pointer, directing where to store the input.
- It’s like asking for an unknown in a math problem and solving for it.

Dissecting the Program: Calculations

Formulas in Action:

```
area = M_PI * radius * radius;
circumference = 2 * M_PI * radius;
```

- Uses the formula of a circle's area and circumference.
- *M_PI* is a constant for the value of π, provided by '<math.h>'.
- It's like applying mathematical formulas to given values.

A New Task: Adding Two Numbers

Task:

- Calculate the sum of two given numbers.

C Program:

```
#include<stdio.h>

int main() {
    int num1, num2, sum;
    printf("Enter first number: ");
    scanf("%d", &num1);
    printf("Enter second number: ");
    scanf("%d", &num2);
    sum = num1 + num2;
    printf("Sum = %d\n", sum);
    return 0;
}
```

Taking User Input: First Number

Input for First Number:

```
printf("Enter first number: ");
scanf("%d", &num1);
```

- ‘printf’ prompts the user to enter the first number.
- ‘scanf’ reads the user’s input for the first number.
- ‘%d’ is a placeholder indicating we expect an integer input.
- ‘&num1’ tells the program where to store the input number.

Taking User Input: Second Number

Input for Second Number:

```
printf("Enter second number: ");
scanf("%d", &num2);
```

- Similarly, ‘printf’ prompts the user to enter the second number.
- ‘scanf’ reads the user’s input for the second number and stores it in ‘num2’.

Performing the Addition

Addition:

```
sum = num1 + num2;
```

- Here, the values stored in ‘num1’ and ‘num2’ are added.
- The result is then stored in the variable ‘sum’.
- It’s a straightforward representation of the arithmetic addition operation.

Displaying the Result

Output:

```
printf("Sum = %d\n", sum);
```

- The ‘printf’ function displays the sum of the two numbers.
- ‘%d’ will be replaced by the value of ‘sum’.
- The user gets immediate feedback on the sum of their provided numbers.

Introduction: Basic Data Types in C

- In programming, data is categorized into types.
- Types help the compiler understand how to interpret and manipulate data.
- Common basic types in C: integers (‘int‘), floating-point numbers (‘float‘), characters (‘char‘), etc.
- Let’s explore these fundamental building blocks!

Integer Type ('int')

Definition: Represents whole numbers, both positive and negative.

```
int age = 25;
int negative_number = -100;
```

- Typical size: 4 bytes (varies by system).
- Range: -2^{31} to $2^{31} - 1$ (based on 4 bytes).
- Used for counting, ranking, etc.

Floating-Point Type (‘float’)

Definition: Represents real numbers, containing both integer and fractional parts.

```
float pi = 3.14;
float negative_float = -0.45;
```

- Typical size: 4 bytes.
- Can represent numbers with decimals, like measurements or scientific data.
- Note: Precision is limited; not suitable for financial calculations.

Character Type (‘char’)

Definition: Represents individual characters such as letters, numbers, or symbols.

```
char letter = ‘A’;
char digit = ‘5’;
```

- Size: 1 byte.
- Range: -128 to 127 or 0 to 255.
- Used to store text, symbols, etc.
- Characters are enclosed in single quotes (‘ ’).

Double Precision Floating-Point Type (double)

Definition: Represents real numbers with higher precision than float.

```
double gravity = 9.81;
```

- Typical size: 8 bytes.
- Offers more significant digits and a wider range than float.
- Suitable for scientific calculations requiring high precision.

Introduction to Derived Data Types

- **Derived Data Types:** Built from basic data types and provide more complexity.
- **Arrays:** Collection of elements (e.g., integers or characters) of the same type.
- **Strings:** Sequence of characters, often used to represent words or text.
- **Structures:** Collection of variables under a single name, allowing different data types.
- **Unions:** Similar to structures, but variables share the same memory location.
- **Pointers:** Stores the memory address of another variable.
- These will be introduced one by one as we progress, enhancing our programming capabilities!

Summary: Basic Data Types in C

- **Integers (int):** Whole numbers, e.g., -3, 0, 42.
- **Floating-Point (float):** Decimal numbers with single precision, e.g., 3.14.
- **Double Precision (double):** Decimal numbers with higher precision, e.g., 9.81234567.
- **Characters (char):** Individual symbols or letters, e.g., ‘A’, ‘9’.
- Understanding these types is foundational for writing clear and efficient code.
- Challenge: How would you represent a book’s title, price, number of pages, and average rating in C?

Think!

- What will be the output of the following code?

```
#include <stdio.h>

int main() {
    int integer = 10;
    float floating = 3.14;
    integer = floating;
    printf("%d\n", integer);
    return 0;
}
```

- Why does this happen, and how might you prevent it?

Basic Conversions

- In C, characters can be implicitly converted to integers using their ASCII values.
- Example:

```
#include <stdio.h>

int main() {
    int integer;
    char character = 'A';
    integer = character;
    printf("%d\n", integer); // prints 65
    return 0;
}
```

- The character 'A' has an ASCII value of 65, so when it is assigned to the integer variable, that value is stored.
- This concept can be useful, but also can lead to unexpected behaviour if not handled with care.

Character Arithmetic

- What will be the output of the following code?

```
#include <stdio.h>

int main() {
    char character = 'A';
    character = character + 3;
    printf("%c\n", character);
    return 0;
}
```

- Why does this work, and what underlying principles does it illustrate?

Recap

- We dived deep into simple C programs, dissecting each line and understanding its purpose.
- Just like in mathematics, every component in a program has a role. Recognizing these components is key to mastering coding.
- Remember, programming is as much about logic and structure as math is about patterns and rules.
- Always be curious! Ask "why" and "how" to deepen your understanding.

Boosting Your Learning

- **Practice!** The best way to learn programming is to code regularly.
- Relate coding concepts to real-world scenarios or other subjects you know. Analogies can make complex ideas simpler.
- Join coding communities or groups. Discussing and collaborating can offer fresh perspectives.
- Challenge yourself. Once you grasp a concept, push your boundaries. Explore what else you can do with what you've learned.

Questions to Ponder

1. Why do we use the ‘return 0;’ statement in our ‘main’ function? What would happen if we didn’t?
2. Why is understanding the memory address concept (‘&’ in ‘scanf’) essential in C programming?

Thank You and Keep Coding!

"Don’t be pushed around by the fears in your mind. Be led by the dreams in your heart."

- Roy T. Bennett