Wednesday, 19 March 2025

Assignment

QUESTION 1:

**The History and Evolution of C Programming: Its Importance and Continued Use**

* The C programming language, developed in the early 1970s by Dennis Ritchie at Bell Laboratories, has had a profound impact on the field of computer science.
* Known for its efficiency, flexibility, and portability, C remains a foundational language in modern computing.

**Origins and Development**

* C emerged as an evolution of the B programming language, which itself was derived from BCPL (Basic Combined Programming Language). Dennis Ritchie and his colleague, Ken Thompson, aimed to create a more powerful and versatile language to facilitate the development of the Unix operating system. By 1972, C had been formally developed and quickly became the language of choice for system-level programming.
* The first edition of the C programming language was documented in the book "The C Programming Language" by Brian Kernighan and Dennis Ritchie, published in 1978. This book, often referred to as "K&R C," became the definitive guide for programmers and laid the foundation for the language's standardization.

**Standardization and Evolution**

* In 1989, the American National Standards Institute (ANSI) standardized C, resulting in ANSI C, also known as C89. This standard provided consistency and compatibility across different platforms and compilers. Subsequently, the International Organization for Standardization (ISO) adopted the ANSI standard, leading to the creation of ISO C, or C90.
* Further revisions and enhancements followed, including C99 (1999), which introduced features like variable-length arrays and improved support for mathematical functions. C11 (2011) added multithreading capabilities and enhanced security features, while C17 (2017) focused on minor improvements and bug fixes. The latest standard, C23 (2023), continues to refine the language with modern features and optimizations.

**Importance and Impact**

* C's significance lies in its low-level access to system memory and hardware, making it ideal for operating system development, embedded systems, and performance-critical applications. Unix, Linux, and Windows operating systems were all heavily influenced by or written in C.
* Moreover, C has served as the foundation for many other programming languages, including C++, C#, Java, and Python. Its syntax and concepts are widely adopted, allowing programmers to transition easily between languages.

**Continued Relevance**

Despite the emergence of higher-level programming languages, C remains relevant for several reasons:

* Efficiency and Performance: C allows for direct manipulation of hardware and system resources, making it ideal for applications that require high performance.
* Portability: Programs written in C can be easily adapted to different platforms with minimal modification.
* Embedded Systems: C is extensively used in embedded systems, such as microcontrollers and IoT devices.
* Educational Value: Learning C provides a solid foundation in programming concepts, such as memory management and algorithm design.

**Conclusion**

The C programming language has stood the test of time due to its efficiency, versatility, and influence on modern computing. From operating systems to embedded systems and beyond, C continues to play a vital role in the development of technology. Its enduring relevance is a testament to its robust design and the profound impact it has had on the programming world.

QUESTION 2:

**Describe the steps to install a C compiler (e.g., GCC) and set up an Integrated**

**Development Environment (IDE) like DevC++, VS Code, or CodeBlocks**.

**1. Install GCC Compiler**

**For Windows:**

1. Install it and choose the default options.

2. Add the MinGW bin folder (e.g., C:\MinGW\bin) to

your **Path** in system environment variables.

3. Open **Command Prompt** and type gcc --version to

check if it's installed correctly.

**For macOS:**

1. Open the **Terminal** and type xcode-select --install.
2. This installs GCC and other necessary tools.
3. Type gcc --version in the terminal to check if it’s installed.

**For Linux (Ubuntu):**

1. Open the **Terminal** and run sudo apt install build-essential.
2. Type gcc --version to verify the installation.

**2. Install and Set Up an IDE**

**For DevC++ (Windows)**:

1. Install DevC++ (it comes with MinGW).
2. Open DevC++, create a new **C Project**, and start coding.

**For Visual Studio Code (Windows, macOS, Linux)**:

1. Install the **C/C++ Extension** in VS Code from the Extensions Marketplace.

2. Ensure GCC is installed.

3. Write your code in a .c file, then press Ctrl+Shift+B to

compile and run.

**For CodeBlocks (Windows, macOS, Linux)**:

1.Choose the version with **MinGW** (for Windows).

2.Install it and open CodeBlocks.

3.Create a new **Console Application** in C, write your code, and click **Build and Run**.

QUESTION 3:

**Explain the basic structure of a C program, including headers, main function, comments, data types, and variables. Provide examples.**

Headers: Include necessary libraries.

Example: #include <stdio.h>

Main Function: Where the program starts.

Example: int main() {

// code here

return 0;

}

Comments: Single-line: // This is a comment

Multi-line: /\* This is a comment \*/

Data Types: Data types in C define what kind of data a variable can store.

Example: int for integers, float for decimal numbers, char for single characters.

Variables: Store data.

Example:  
int num = 5; // Integer

float pi = 3.14; // Floating point

char letter = 'A'; // Character

Example Program:

#include <stdio.h> // Include library for I/O

int main() {

int num = 10; // Integer variable

printf("Number: %d\n", num); // Print the number

return 0; // End the program

}

EXAMPLE

#include <stdio.h>

int main() {

int age = 20; // Integer

float price = 9.99; // Floating-point number

char grade = 'A'; // Character

printf("Age: %d\n", age);

printf("Price: %.2f\n", price);

printf("Grade: %c\n", grade);

return 0;

}

QUESTION 4:

**Write notes explaining each type of operator in C: arithmetic, relational, logical, assignment, increment/decrement, bitwise, and conditional operators.**

ARITHMETIC OPERATORS

USED FOR BASIC MATHEMATICAL OPERATIONS

**Operator** **Description Example**

+ Addition a + b

- Subtraction a - b

\* Multiplication a \* b

/ Division a / b

% Modulus (remainder) a % b

**Example:**

int a = 10, b = 5;

printf("Sum: %d\n", a + b); // Output: 15