MANGALAYATAN UNIVERSITY
Learn Today to Lead Tomorrow
NAAC GRADE A+ Accredited University

BACHELOR OF COMPUTER APPLICATIONS (BCA)

CSB-1151 (Problem Solving Using C Lab)

Block 1: Programming and Algorithm Lab

Unit 1: Fundamentals of Programming and Problem Solving

Structure

1.0 INTRODUCTION

This lab course offers hands-on experience aimed at practical application. Participants have finished the CSB-1151 Problem solving through Lab support course, which covers C programming examples across Windows, UNIX, and DOS systems. Each session concludes with a series of programming problems for practice. It's essential to thoroughly review the program documentation regulations and adhere to the general guidelines provided.

Program development steps:

This program development involves a series of sequential steps essential for creating in a high-level language and converting it into machine-level language:

1. Drafting and refining the software design.
2. Linking the application with required library modules.
3. Compiling the software.

4. Executing the program for operation.

This lab course will be discussing the separate step compilation process of language C.

1.1 OBJECTIVES

After completing this lab course, you will be able to:

- The goal is to guide learners in comprehending the problem-solving rationale and algorithm formulation process.
- This includes crafting a corresponding flowchart and comprehending the syntax and framework of C programming.
- Proficiency in procedural language programming is emphasized, covering the methods for assembling, connecting, and resolving issues in C code.
- The primary aim is to impart a foundational comprehension of C language programming to students. This encompasses teaching them problem-solving strategies and proficient writing techniques in C programming.
- The curriculum covers fundamental concepts such as functions, pointers, file handling, structures, loops, and arrays, ensuring a comprehensive understanding of these key components in programming.

1.2 OVERALL DIRECTIONS

To try each of the tasks and challenges listed in the list, session by session.

You can ask the responsible lab teacher for help completing the lab exercises. The lab teacher is clearly not expected to provide you solutions to the assignments as they are credit-based, but you are welcome to ask questions about the C language or any technical issues.

You should put comments (text in between /*... */ delimiters) above every function in the code, including the main function, for every application. A description of the function that has been developed, its goal, the significance of the parameter it uses, and the meaning of the return result, if any, should also be included.

Prior to the primary function’s source code, comprehensive explanations outlining the purpose of the program will be incorporated within the comment block. These explanations will elucidate the program's objectives and functionalities. Throughout the code, relevant comments will be strategically placed to enhance readability and understanding.

The C program will strictly adhere to the ANSI standard for the language, ensuring compatibility and conformity. It will be developed as a generic and interactive application, meticulously documented with real input and output data to facilitate comprehension.

To maintain integrity, submissions that appear derivative, stemming from multiple sources but exhibiting remarkable similarities, will not be considered. It is strongly advised against replicating or mimicking someone else's work to ensure individuality and authenticity in submissions.

Your responsibility includes creating a separate directory inaccessible to others for storing all programs to ensure confidentiality. Keeping an Observation Book and Lab Record is mandatory. The lab manual provides a session-wise list of programs, and it's essential to prepare algorithms and record programs in the Observation Book prior to each session.

During lab hours, dedicate time to executing, testing, and enhancing programs for desired outputs. Upon completing a lab exercise, approach a lab instructor or in-charge for evaluation and signature in the Observation Book.

Lab assignments should be submitted in the form of a comprehensive Lab Record. This record should encompass algorithms, program codes with comments, and outputs for various inputs provided, ensuring a thorough documentation of the work completed.

1.3 STRUCTURE OF 'C' PROGRAM

A 'C' program is constructed from multiple instructions, each written as an individual statement. It commences with the main function enclosed in opening braces, signifying its initiation. This is succeeded by variable and constant declarations, succeeded by statements encompassing input and output operations.

The structure of a 'C' program typically involves several sections, as outlined below:

1. Initialization: Begins with the 'main' function, the entry point of execution.
2. Variable and Constant Declarations: Defines variables and constants necessary for the program.
3. Statements: Includes instructions for data processing, involving input/output operations and logic implementation.

These sections collectively form the structure of a 'C' program, organizing the flow of operations:

DOCUMENTATION SECTION

LINK SECTION

DEFINITION SECTION

GLOBAL DECLARATION SECTION

Main() Function Section

{

Declaration part

Executable part

}

SUBPROGRAM SECTION

User defined function

1.4 SALIENT FEATURES OF C

C language boasts several defining characteristics that have propelled its popularity within the programming landscape, many of which were thoroughly covered during the BCA-151 Problem Solving and Programming course:

1. **Small Size:** C's concise nature allows for efficient coding without unnecessary overheads.
2. **Extensive Use of Function Calls:** Its modular approach leverages functions for code organization and reusability.
3. **Structured Language:** Encourages organized and systematic programming practices, enhancing

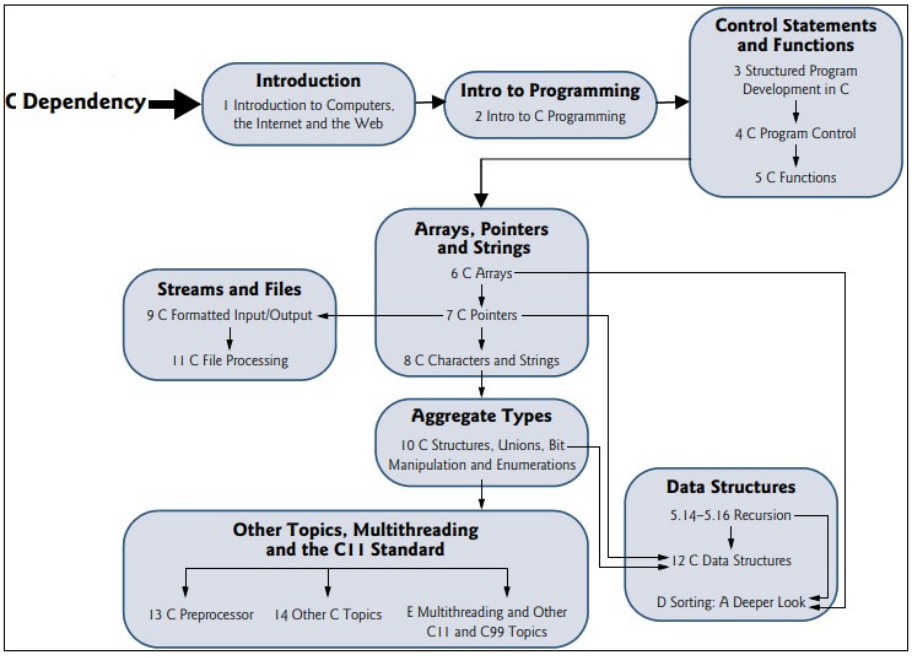
readability and maintenance.

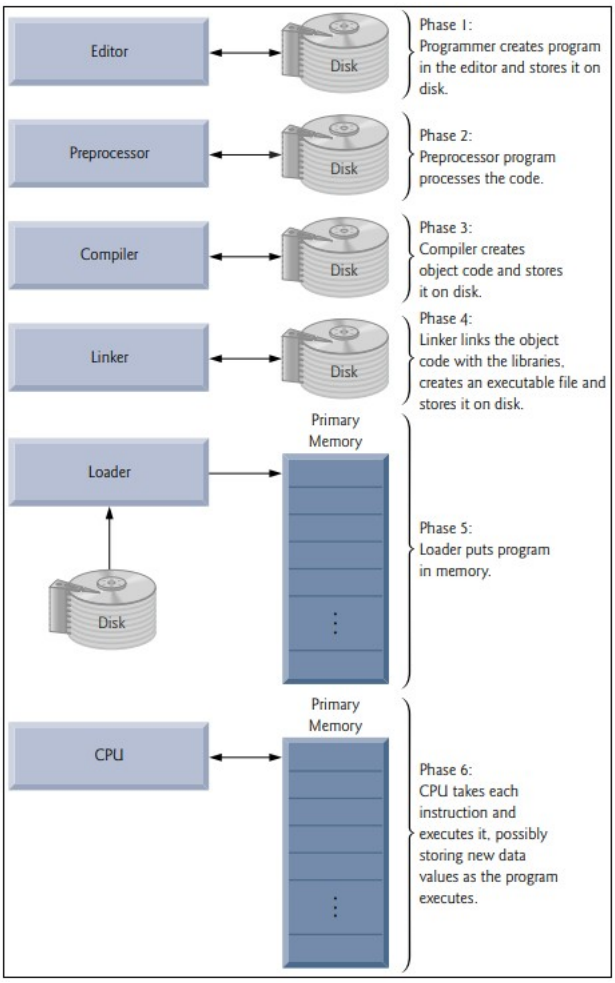
4. **Low-Level (Bitwise) Programming Availability:** Offers direct access to memory and bitwise operations for optimized code implementation.
5. **Pointer Implementation:** Extensively employs pointers for memory management, arrays, structures, and functions, enabling intricate data manipulation.
6. **High-Level Constructs:** Provides high-level constructs for abstraction, simplifying complex operations.
7. **Handling Low-Level Activities:** Allows direct manipulation of hardware, making it suitable for system programming and embedded systems.
8. **Efficient Program Output:** Produces highly efficient programs due to its close-to-hardware approach.
9. **Cross-Platform Compilation:** Offers portability, allowing compilation on various computer architectures, contributing to its versatility and widespread use."

These features collectively contribute to C's robustness, flexibility, and efficiency, making it a prominent choice for diverse programming needs.

1.5 'C' PROGRAM DEVELOPMENT ENVIRONMENT

C systems generally consist of several parts: a program-development environment, the language and the C Standard Library. ***Explain the following typical C development environment:***





C programs typically go through six phases to be executed. These are: edit, preprocess, compile, link, load and execute.

1.5.1 Phase I: Creating a Program

Visual Studio Code stands out as one of the most widely utilized code editors and integrated development environments (IDEs) developed by Microsoft. It serves as a versatile platform for coding in various programming languages, fostering the creation and optimization of codebases while facilitating efficient debugging. Notably, Visual Studio Code boasts cross-platform compatibility, running seamlessly on Windows, macOS, and Linux operating systems.

The editor's popularity extends globally, including in India, where it has gained widespread adoption. Its user-friendly interface and extensive language support, encompassing languages such as C, C++, Java, Python, JavaScript, React, and Node.js, contribute to its broad appeal. Visual Studio Code distinguishes itself by offering a rich ecosystem of in-app extensions tailored for diverse programming languages, enabling users to tailor their coding environment to specific needs.

One of the noteworthy features of Visual Studio Code is its visually appealing and dynamic user interface, complemented by a sophisticated night mode that enhances the coding experience. The editor facilitates a smooth coding process by providing users with auto-complete code suggestions, streamlining the writing of code and enhancing overall productivity.

In conclusion, Visual Studio Code has secured its position as a premier code editor and IDE due to its versatility, extensive language support, user-friendly interface, and a plethora of features, making it a top

choice for programmers across the globe, including in India.

1.5.2 Phase II & III: Preprocessing and Compiling a 'C' Program

The compiler translates the C program into machine-language code, also known as object code. Before the translation phase, a preprocessor program in the C system automatically executes. This preprocessor adheres to special commands known as preprocessor directives, instructing specific manipulations to be carried out on the program before compilation. These manipulations commonly involve including other files within the file being compiled and performing text replacements.

The compiler converts the C program into machine-language code. Syntax errors occur when the compiler encounters a statement that violates the language's rules and cannot be recognized. In response, the compiler issues an error message, aiding in locating and rectifying the erroneous statement. It's worth noting that the wording for error messages issued by the compiler isn't standardized by the C Standard, leading to potential variations in error messages across different systems. These errors are commonly referred to as compile errors or compile-time errors.

1.5.3 Install Visual Studio Code on Windows

To Install Visual Studio Code on a Windows System, follow these steps:

1. Download Visual Studio Code:

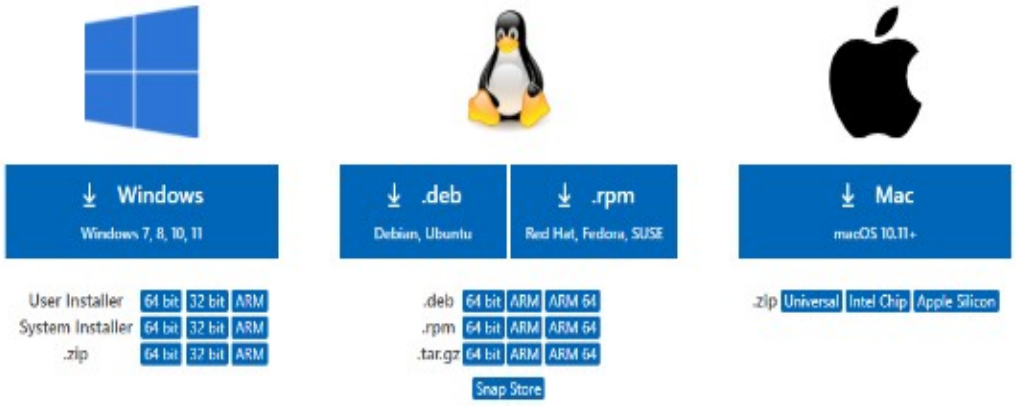
   - Open your web browser and navigate to the official Visual Studio Code website:

   https://code.visualstudio.com/.

   - Click on the "Download for Windows" button.

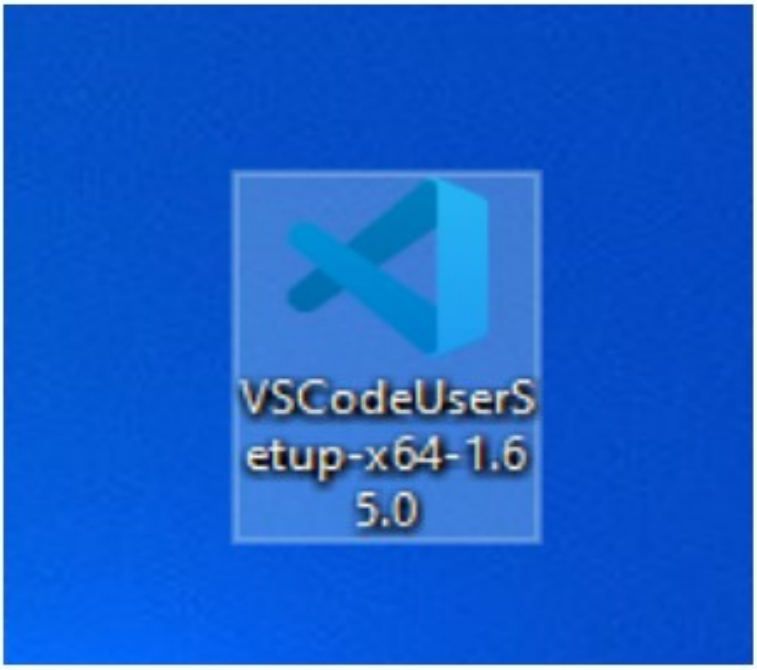
2. Run the Installer:

   - Once the installer executable (.exe) is downloaded, locate the file (usually in the Downloads folder) and double-click on it to run the installer.



3. Begin Installation:

   - The installer will prompt you to confirm that you want to install Visual Studio Code. Click "Yes" or "Run" to proceed.

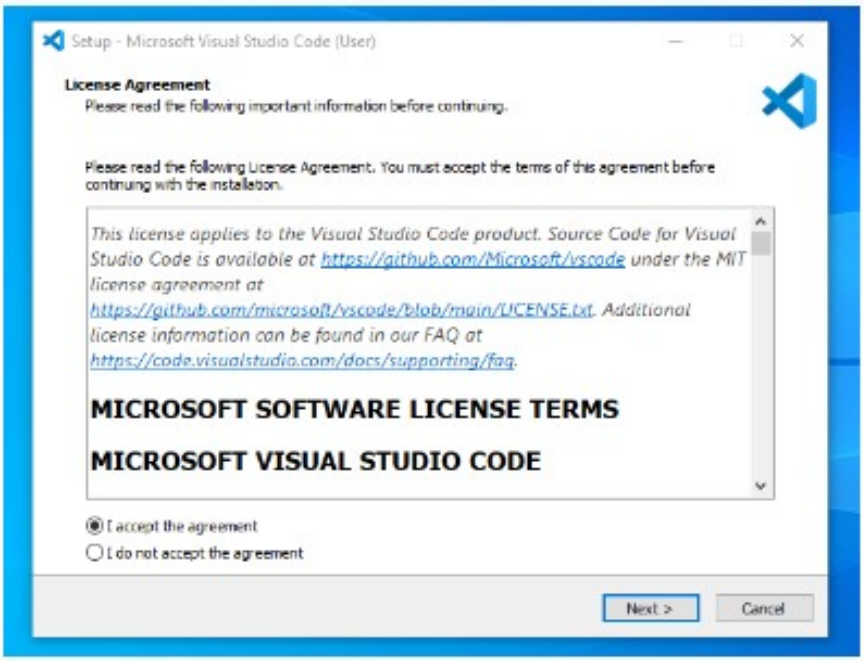


4. Choose Setup Options:

   - The installer will provide various setup options. You can choose the default settings or customize them according to your preferences. For most users, the default options are sufficient.

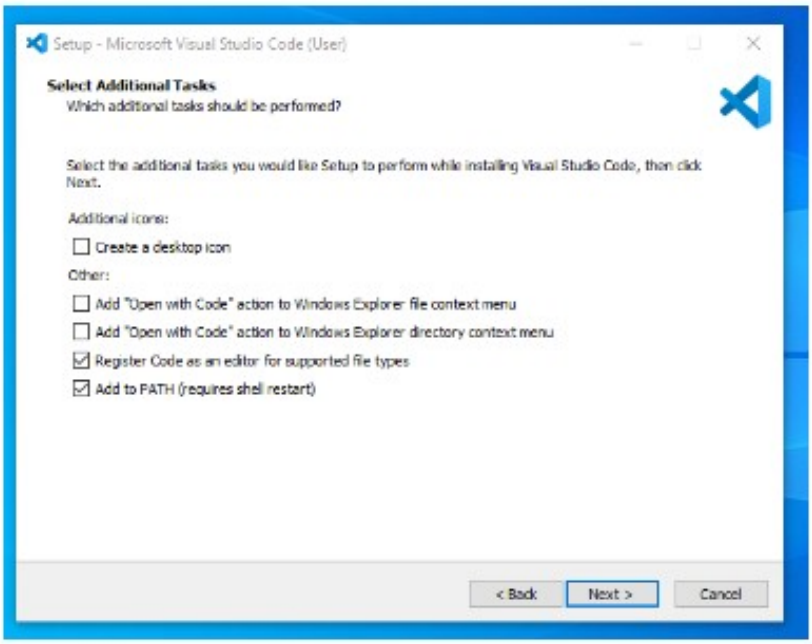
5. Select Additional Tasks:

   - The installer may offer additional tasks, such as creating desktop shortcuts or adding entries to the PATH environment variable. Choose the options that suit your preferences.



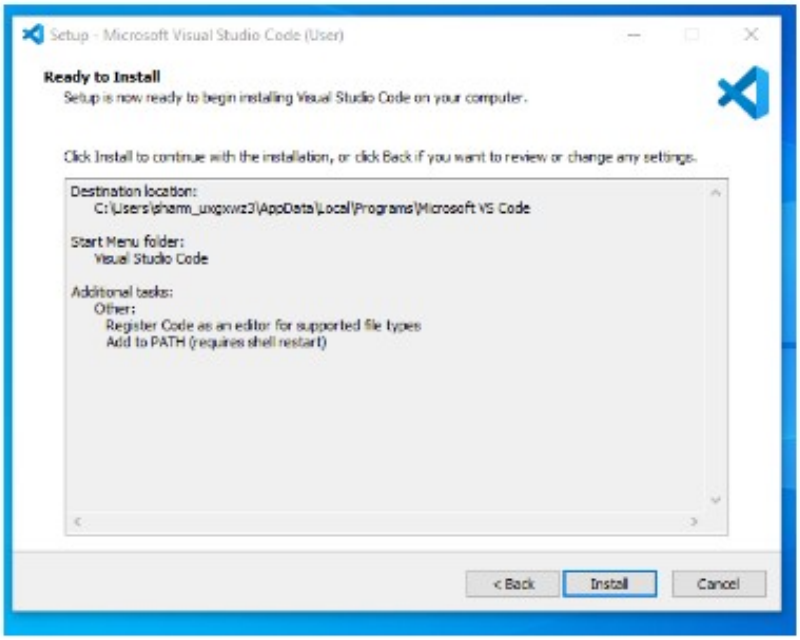
6. Install:

   - Click the "Install" button to initiate the installation process. The installer will copy the necessary files and set up Visual Studio Code on your system.



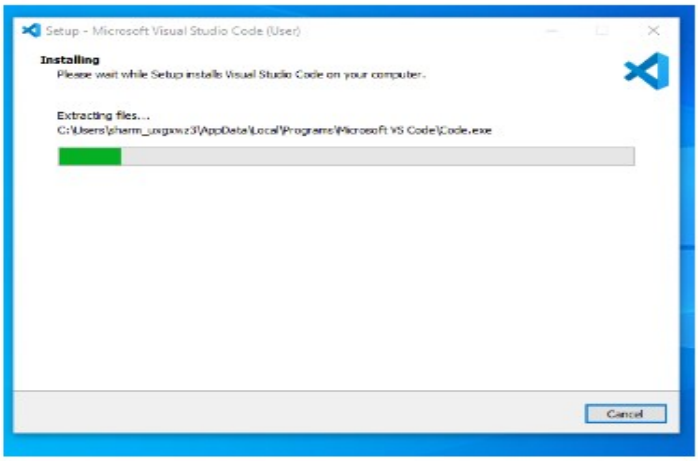
7. Complete Installation:

   - Once the installation is complete, you will see a confirmation message. You can choose to launch Visual Studio Code immediately by leaving the corresponding option checked.



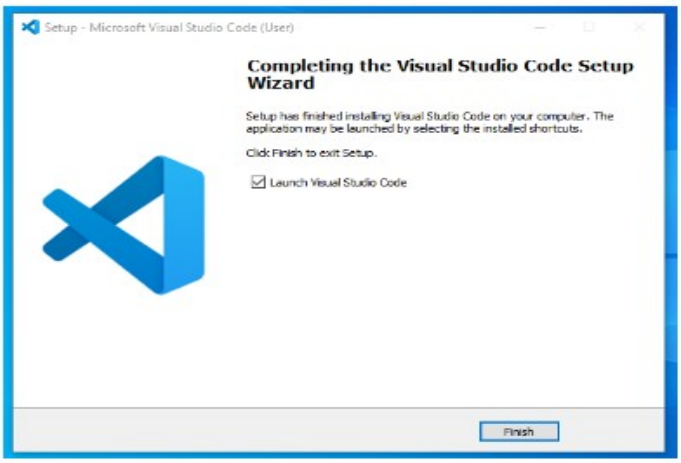
8. Launch Visual Studio Code:

   - If you didn't choose to launch it during the installation, you can find the Visual Studio Code shortcut on your desktop or in the Start menu. Double-click on the shortcut to open Visual Studio Code.

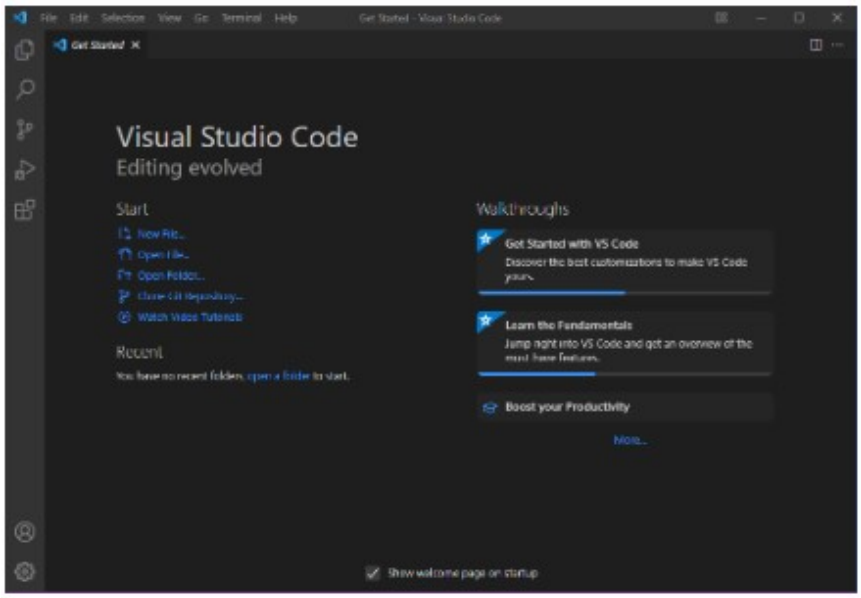


9. Update Extensions (Optional):

   - After launching Visual Studio Code, you may want to explore and install extensions based on your programming needs. You can access the Extensions view by clicking on the Extensions icon in the Activity Bar on the side.



That's it! You have successfully installed Visual Studio Code on your Windows system and ready to start coding in your preferred programming languages.



Successfully Installed Visual Studio Code **on our Windows system**

1.6 HOW TO DESIGN/DEVELOP PROGRAM

Steps involved in program development:

To develop the program in high level language and translate it into machine level language following steps have to be practised.

1. Writing and editing the program.
2. Linking the program with the required library modules.
3. Compiling the program.
4. Executing the program.

Algorithm:

It is a method of representing the step by step process for solving a problem. Each step is called an instruction.

Characteristics of algorithm are:

Finiteness: It terminates with finite number of steps.

Definiteness: Each step of algorithm is exactly defined.

Effectiveness: All the operations used in the algorithm can be performed exactly in a fixed duration of time.

Input: An algorithm must have an input before the execution of program begins.

Output: An algorithm has one or more outputs after the execution of the program.

Example of algorithm to find sum of two numbers:

Step1: BEGIN

Step2: READ a, b

Step3: ADD a and b and store in variable c

Step4: DISPLAY c

Step5: STOP

1.7 STRUCTURE OF 'C' PROGRAM

C program is a collection of several instructions where each instruction is written as a separate statement. The C program starts with a main function followed by the opening braces which indicates the start of the function. Then follows the variable and constant declarations which are followed by the statements that include input and output statements.

C program may contain one or more sections as:

DOCUMENTATION SECTION

LINK SECTION

DEFINITION SECTION

GLOBAL DECLARATION SECTION

Main() Function section

{

Declaration part

Executable part

}

SUBPROGRAM SECTION

User defined functions

Example:

Write a C program to find the sum and average of three numbers.

Algorithm:

Step 1: Start

Step 2: Declare variables num1, num2, num3 and sum, average.

Step 3: Read values num1, num2, num3.

Step 4: Add num1, num2, num3 and assign the result to sum.

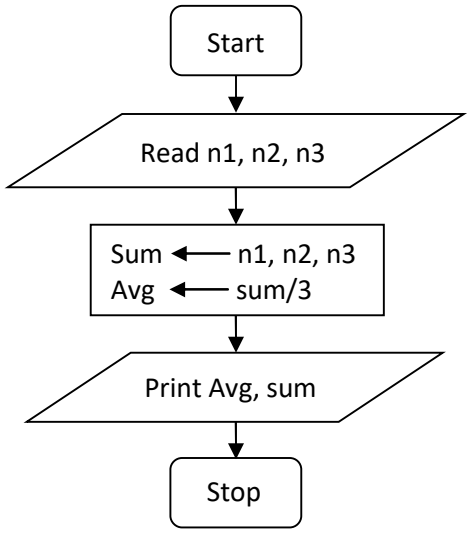
sum ⟵ num1 + num2 + num3

average ⟵ sum/3

Step 5: Display sum and average

Step 6: Stop

Flow Chart:



Program:

```
#include<stdio.h>
void main( )
```

```
{
int a,b,c;
int sum,average;
printf("Enter any three integers: ");
scanf("%d%d %d",&a,&b,&c);
sum = a+b+c;
average=sum/3;
printf("Sum and average of three integers: %d %d",sum,average);
return 0;
}
```

INPUT: Enter any three integers: 2 4 5

OUTPUT: Sum and average of three integers: 11 3

Example:

Write a C program to find the sum of individual digits of positive integer.

Algorithm:

Step 1: Start

Step 2: Read

Step 3: Initialize sum ⟵ 0

Step 4: while(n!=0)

Begin

Step 5: r ⟵ n%10

Step 6: Sum ⟵ Sum+r

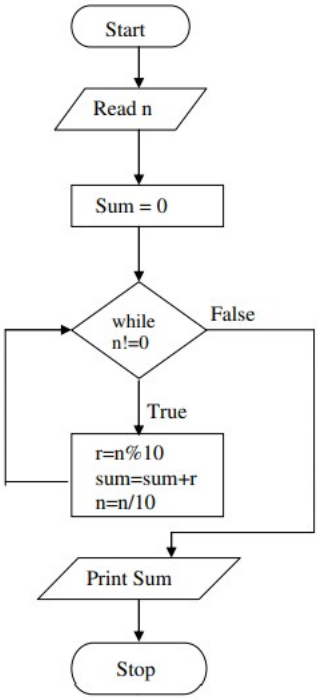
Step 7: n ⟵ n/10

End

Step 8: Print “sum”

Step 9: Stop

Flow Chart:



Program:

```
#include<stdio.h>
#include<conio.h>
void main( )
{
int n,r,sum=0;
clrscr();
printf("ENTER A POSITIVE INTEGER \n");
scanf("%d",&n);
while(n!=0)
{
r=n%10;
sum=sum+r;
n=n/10;
}
printf("THE SUMOF INDIVIDUAL DIGITS OF A POSITIVE INTEGER IS..%d",sum); getch();
}
```

INPUT: ENTER A POSITIVE INTEGER 5 3 2 1

OUTPUT: THE SUM OF INDIVIDUAL DIGITS OF A POSITIVE INTEGER IS..11

Example:

Write a C program to check whether given number is Armstrong Number or Not.

Algorithm:

Step 1: Start

Step 2: Read n

Step 3: assign sum

Step 4: if m>0 repeat

Step 4.1: m ⟵ m/10

Step 4.2: count++

Step 4.3: until the condition fail

Step5: if I>0 repeat step 4 until condition fail

Step 5.1: rem ⟵ I%10

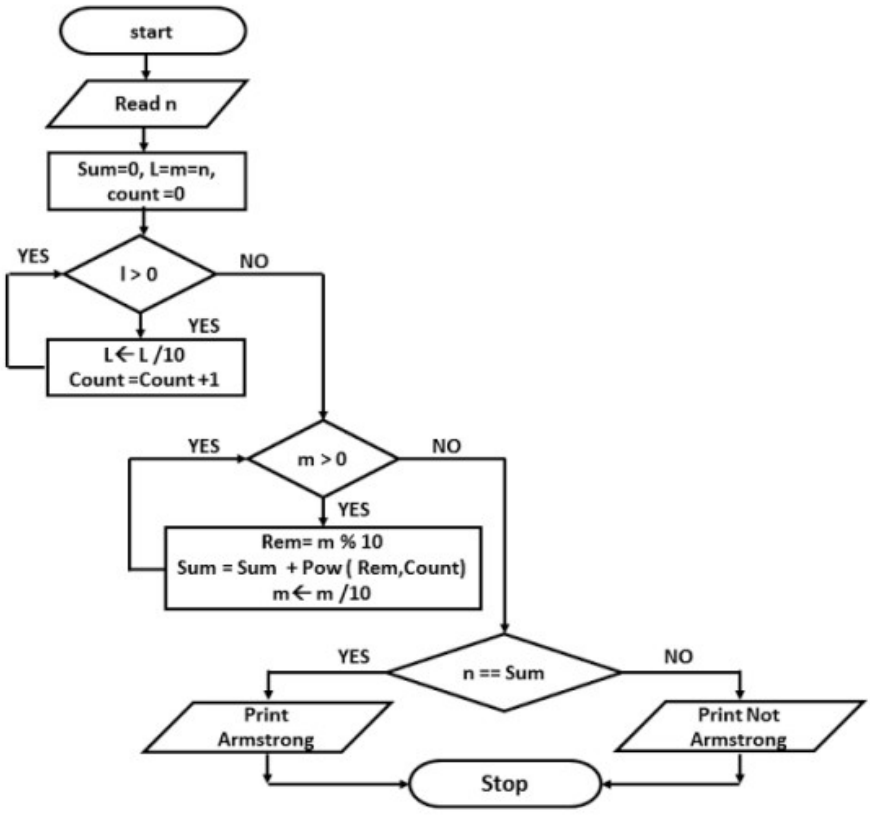
Step 5.2: sum ⟵ sum+pow(rem,count)

Step 5.3: I ⟵ I/10

Step 6: if n=sum print Armstrong otherwise print not Armstrong

Step 7: stop

Flow Chart:



Program:

```
#include<stdio.h>
void main()
{
int n, n1, rem, num=0;
printf("Enter a positive integer: ");
scanf("%d", &n); n1=n;
while(n1!=0)
{
rem=n1%10;
num+=rem*rem*rem;
n1/=10;
}
if(num==n)
printf("%d is an Armstrong number.",n);
else printf("%d is not an Armstrong number.",n);
}
```

Input: Enter a positive integer: 371

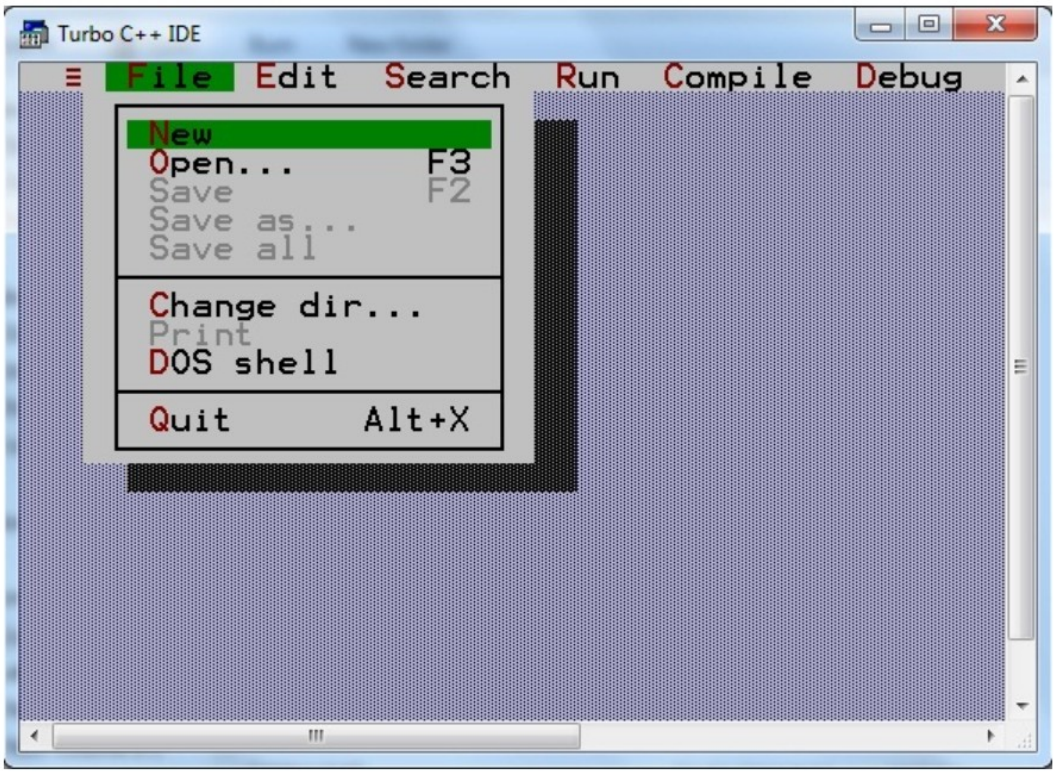
Output: 371 is an Armstrong number.

1.8 COMPILE AND RUN 'C' PROGRAM

To compile and run a C language program, you need a C compiler. A **compiler** is software that is used to compile and execute programs. To set up a C language compiler on your Computer/laptop, there are two ways:

1. Download a full-fledged IDE like Turbo C++ or Microsoft Visual C++ or DevC++, which comes along with a C language compiler.
2. Or, you can use any text editor to edit the program files and download the C compiler separately and then run the C program using the command line.

If you haven't already installed an IDE for the C language - Follow this step-by-step guide to **Install Turbo C++ for C Language**

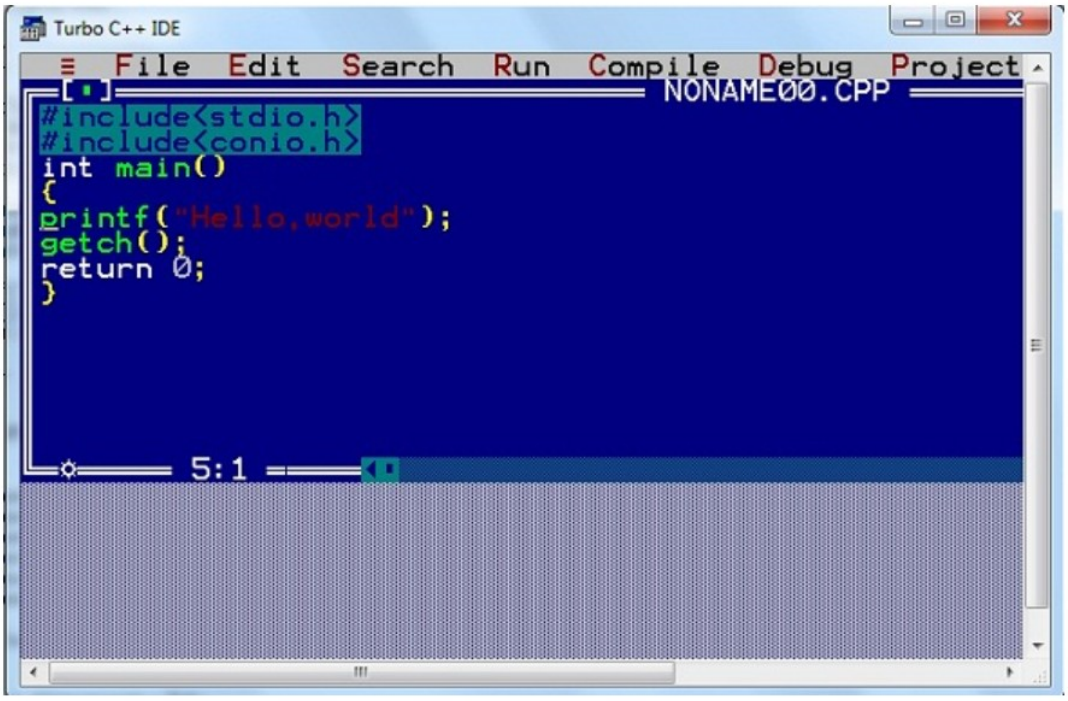


Using an IDE - Turbo C

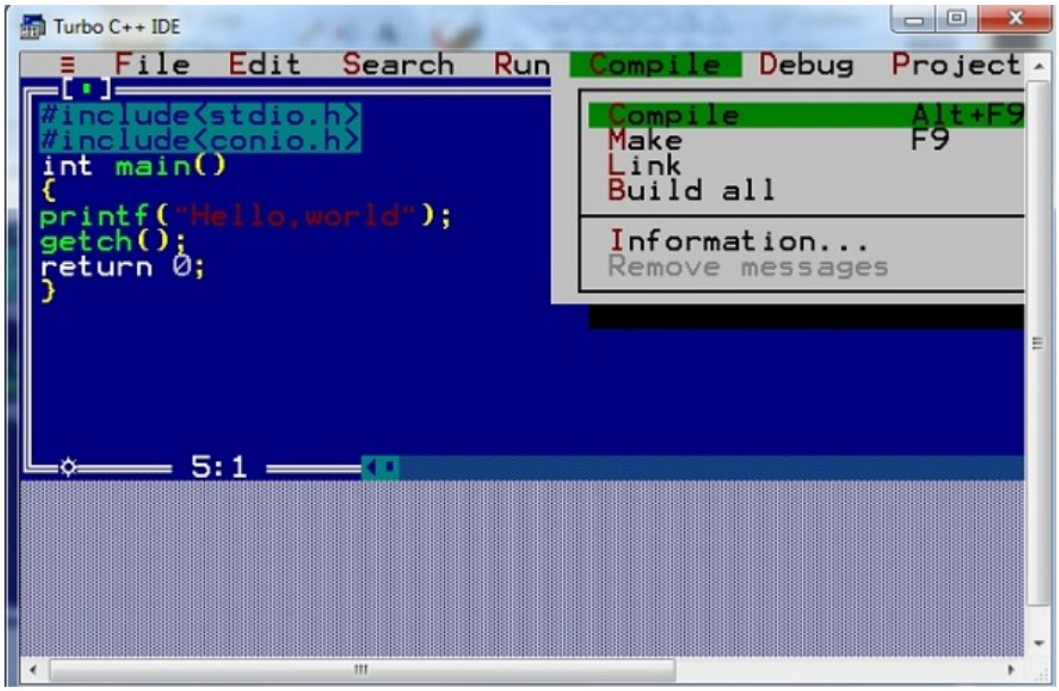
We will recommend you to use **Turbo C or Turbo C++** IDE, which is the oldest IDE for C programming. It is freely available over the Internet and is good for a beginner.

Step 1: Open turbo C IDE (Integrated Development Environment), click on **File,** and then click on **New**

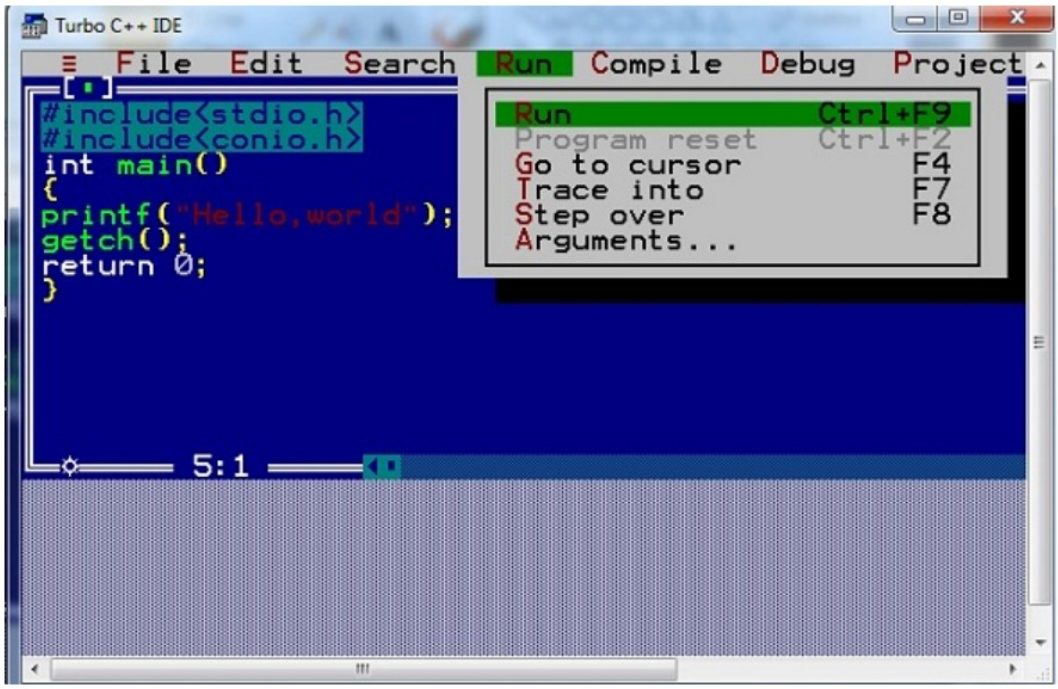
Step 2: Write a Hello World program that we created in the previous article - **C Hello World program**.



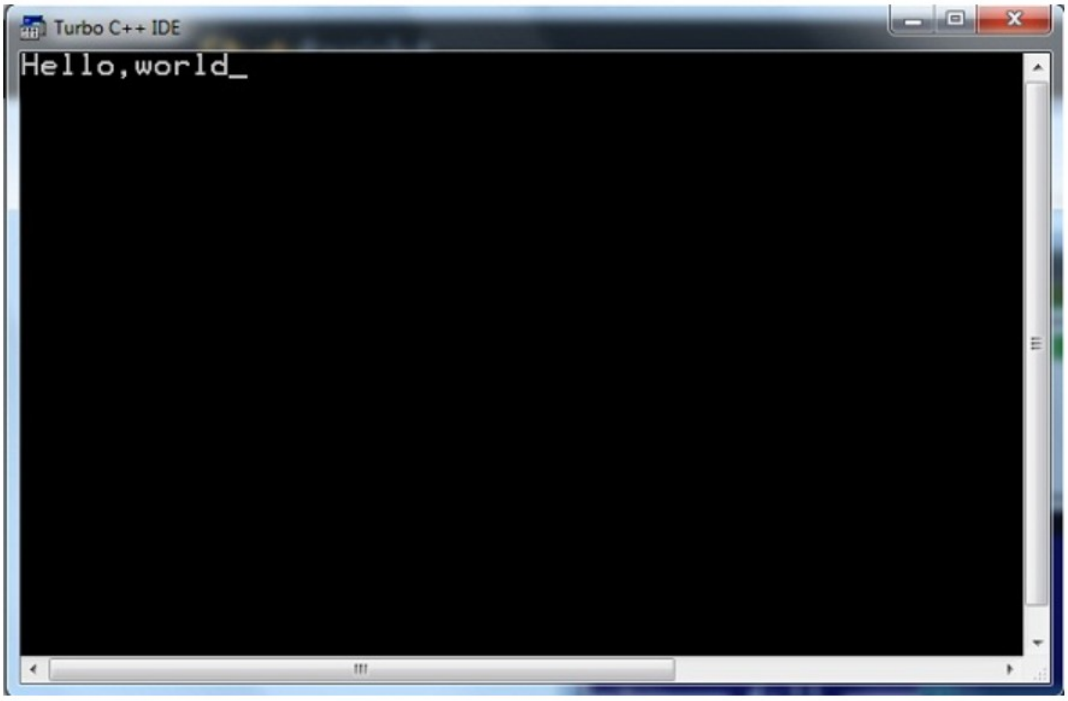
Step 3: Click on **Compile** menu and then on **Compile** option, or press the keys and press **Alt + F9** to compile the code.



Step 4: Click on **Run** or press **Ctrl + F9** to run the code. Yes, C programs are first compiled to generate the object code and then that object code is Run.



Step 5: Output is



Run C Program without using any IDE

- If you do not wish to set up an IDE and prefer the old-school way, then download the C compiler which is called gcc from the GCC website https://gcc.gnu.org/install/
- Once you have downloaded and installed the gcc compiler, all you have to do is, **open any text editor**, copy and paste the C program code for C Hello World Program, and save it with the name the **helloworld.c** like any other file you save with a name.
- Now, Open the **Command prompt or Terminal** (if you use Ubuntu or Mac OS), and go to the directory where you have saved the **helloworld.c** program file.
- Type the command gcc hello.c to compile the code. This will compile the code, and if there are no errors then it will produce an output file with name **a.out** (default name)
- Now, to run the program, type in ./a.out and you will see **Hello, World** displayed on your screen.

1.9 PROGRAMMING AND ALGORITHM LAB

Basic Programming Exercises

1.9.1 Simple Programs: Writing basic programs to understand syntax and structure

Introduction to Basic Programming:

Programming is a crucial skill in today's technology-driven world. At its core, programming involves writing instructions that a computer can execute to perform specific tasks. To begin programming, one must understand the syntax and structure of a programming language. This foundational knowledge enables the creation of simple programs, which serve as building blocks for more complex applications.

Programming Languages and Their Syntax:

Programming languages are formal languages comprising a set of instructions that produce various kinds of output. Examples include C, Java, and JavaScript. Each language has its syntax - the set of rules that defines the combinations of symbols considered to be correctly structured programs in that language.

Basic Syntax Elements

1. **Variables and Data Types:** Variables are used to store data. Each variable must be given a name and assigned a data type, which defines the kind of data it can hold, such as integers, floating-point numbers, characters, and strings.

   Example:

   age = 25 # integer

   name = "Kamal" # string

   height = 5.2 # floating-point

2. **Operators:** Operators are symbols that perform operations on variables and values. Common operators include arithmetic operators (`+`, `-`, `*`, `/`), relational operators (`==`, `!=`, `>`, `<`), and logical operators (`and`, `or`, `not`).

```c
#include <stdio.h>

int main() {
    int age = 25;            // integer
    char name[] = "Alice";   // string (array of characters)
    float height = 5.7;      // floating-point

    printf("Age: %d\n", age);
    printf("Name: %s\n", name);
    printf("Height: %.1f\n", height);

    return 0;
}
```

Explanation:

int, char, and float are data types.

printf is used for outputting data to the console.

%d, %s, and %f. If are format specifiers for integers, strings, and floating-point numbers, respectively.

```c
#include <stdio.h>

int main() {
    int a = 10, b = 5;
    int sum = a + b;
    int difference = a - b;
    int product = a * b;
    int quotient = a / b;

    printf("Sum: %d\n", sum);
    printf("Difference: %d\n", difference);
    printf("Product: %d\n", product);
    printf("Quotient: %d\n", quotient);

    return 0;
}
```

Explanation:

Basic arithmetic operations are demonstrated with '+',' –', '*', and '/'.

1.9.2 Control Structures: Exercises on if-else, switch-case, and loops

Control structures in C manage the flow of a program. The most common control structures are conditional statements and loops.

Conditional Statements

Conditional statements execute different blocks of code based on certain conditions.

```c
#include <stdio.h>

int main() {
    int age = 20;

    if (age > 18) {
        printf("Adult\n");
    } else {
        printf("Not an adult\n");
    }

    return 0;
}
```

Explanation:

if and else statements control the flow based on the condition age > 18.

Loops:

Loops are used to execute a block of code repeatedly.

```c
#include <stdio.h>

int main() {
    for (int i = 0; i < 5; i++) {
        printf("Iteration: %d\n", i);
    }

    return 0;
}
```

Explanation:

A for loop is used to print "Iteration" along with the current loop index.

Writing Simple C Programs

Example 1: Hello, World!

```c
#include <stdio.h>

int main() {
    printf("Hello, World!\n");
    return 0;
}
```

Example 2: Sum of Two Numbers

```c
#include <stdio.h>

int main() {
    float num1, num2, sum;

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

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

    sum = num1 + num2;

    printf("The sum of %.2f and %.2f is %.2f\n", num1, num2, sum);

    return 0;
}
```

Example 3: Even or Odd Number

```
#include <stdio.h>
int main() {
    int num;
    printf("Enter a number: ");
    scanf("%d", &num);
    if (num % 2 == 0) {
        printf("%d is even\n", num);
    } else {
        printf("%d is odd\n", num);
    }
    return 0;
}
```

***