What is Computer programing

Computer programming also known as coding is the process of creating instructions that tell a computer how to perform specific tasks. It involves writing a ser of command called source code. Using a programing language that the computer can understand.

**Importance**

1. Automation and Efficiency: programing allows for the automation of repetitive tasks and the creation of efficient algorithms.
2. Problem Solving: programing encourages logical thinking and problems solving skills. It enables individuals to break down complex problem into smaller.
3. Innovation and Creativity: programing provides a platform for creativity and innovation. It allows to individuals to bring their ideas to life by designing and building software and websites
4. Control and customization: programing empowers individuals to have control over technology. By writing code, we can customize software and system and application according to specific need.
5. Critical thinking and Analytical skills: programing fosters critical thinking and analytical skills. It requires individuals to analyze problems and identify patterns and devise logical solutions.

**Advantages**

1. Automation: computer programing allows for the automation of tasks
2. Efficiency: programing enables the development of efficient algorithms and optimized solutions.
3. Problems solving: programing fosters problem solving skills
4. Collaboration and open source: the programing community is known for collaboration and sharing knowledge.
5. Handling Data: programming skills are valuable for data analysis and manipulation.

**Why we use computer programing**

We use programing for

* Automating Tasks
* Developing software
* Web developing
* Data Analysis
* Artificial Intelligence (AI)
* Machine learning
* Robotic and Automation
* Game Development
* Engineering Simulation
* Internet of Things

In summary, computer programming is used to automate tasks, develop software applications, perform data analysis, drive AI and machine learning, enable robotics and automation, create video games, manage systems, conduct simulations, enable IoT, and much more. It is a foundational skill in the digital age, facilitating innovation and enabling us to leverage the power of computers to solve complex problems and improve various aspects of our lives.

* Top of Form

**DEV C++**

DEV C++ is an integrated Development Environment (IDE) specifically designed for programing in C and C++. It provides a user-friendly interface and set of tools to facilitate software development in these languages. DEV C+++ is free and open source software widely used by beginners and experienced programmers alike.

1. Code Editor
2. Compiler
3. Debugging
4. Project Management
5. Integrated Tools
6. Code Templates
7. Libraries
8. Multiple Language support
9. Customization
10. Extensibility

Overall DEV C++ is a popular and accessible IDE for C and C++ programing. It provides a range of feature and tools to support the development process

**Why we DEV C++ software**

DEV C++ is used in programing to benefit form its simplicity, user-friendly interface, IDE features Integrated compile, project management capabilities, support form the community, open source nature, compatibility with windows, lightweight performance, extension support and educational values, it is a versatile tool that caters to the needs of both novice and experience programmer working with C and C++.

**The working of compiler in our computer**

Compiler is a software tool that translates source code written in a high-level programing language into a form that can be executed directly by a computer or other programs. It takes the human-readable code and converts it into machine-readable instructions that the computer can understand and execute.

The specific working of a compiler can vary based on the programing language the compiler implementation and the target platform; however, the general process involves analyzing the source code, generating intermediate code, optimizing it, translating it into machine code or language and finally executing the compiled program on the computer.

**Format Specifier**

In the C programing language format specifiers are used with input and output functions to define the expected format of the data being read or written. Format specifiers specify the type and formatting of the data during input/output operations. They are placeholders that help in correctly interpreting and displaying data

1. %d integers
2. %of Floating
3. %c character
4. %s strings
5. %o octal number
6. %x hexadecimal
7. %u unsigned integers
8. %e floating point number
9. %g floating point number either or scientific
10. %p pointers
11. %% percentage symbol itself

**Variable and data type**

In the C programming language variable are used to store and manipulates data. A variable is a named memory location that holds a value of a particular data type. Data types specify the type of data that a variable can hold, such as integers, floating-point number, characters and more. Here are some commonly used data types and variable declarations in C:

1. **Integers**
   1. Int: used to store whole number
   2. Short: used for short integers with a smaller range
   3. Long: used for long integers with a larger range
   4. Unsigned int: used to store positive integers or zero

int age = 25;

short quantity = 100;

long population = 789456123;

unsigned int count = 500;

1. **Floating-point number**
   1. Float: used to store single-precision floating-point number.
   2. Double: used to store double-precision floating numbers.

float pi = 3.14;

double salary = 2500.75;

1. **Characters:**
   1. Char: used to store individual character
2. **Strings:**
   1. Char [] or char used to store sequences of characters also known as strings

char name [] = "John";

char\* message = "Hello, world!";

1. **BOOLEAN and Void**

**Scape Sequence**

In c programing language escape sequences are special character sequences that are used to represent certain characters that are difficult to type directly or have special meanings. Escape sequences start with a backslash **\** followed by one or more characters.

1. \n new line
2. \t Horizontal line
3. \x Carriage return
4. \” Double quote
5. \’ Single quote
6. \\ Backslash
7. \an Alert or bell
8. \b Backspace
9. \f Form feed
10. \v Vertical tab
11. \0 Null character

**Which operator use the calculation of Mathematics**

In mathematics, various operators are used to perform calculations. Similarly, programming languages like C also provide operators that allow you to perform mathematical calculations on numeric data. Here are some commonly used mathematical operators in C:

* Addition: `+`
  1. Used to add two values together.
* Subtraction: `-`
  1. Used to subtract one value from another.
* Multiplication: `\*`
  1. Used to multiply two values.
* Division: `/`
  1. Used to divide one value by another.
* Modulo: `%`
  1. Returns the remainder after dividing one value by another.
* Increment: `++`
  1. Increases the value of a variable by 1.
* Decrement: `--`
  1. Decreases the value of a variable by 1.
* Assignment: `=`
  1. Assigns a value to a variable.

**Compare the two loops / In which situation we use for loop and do-while loop**

|  |  |
| --- | --- |
| **For Loop** | **While Loop** |
| It is used when the number of iterations is known. | It is used when the number of iterations is not known. |
| In case of no condition, the loop is repeated infinite times. | In case of no condition, an error will be shown. |
| Initialization is not repeated. | Initialization is repeated if carried out during the stage of checking. |
| Statement of Iteration is written after running. | It can be written at any place. |
| Initialization can be in or out of the loop | Initialization is always out of the loop. |
| The nature of the increment is simple. | The nature of the increment is complex. |
| Used when initialization is simple. | Used when initialization is complex. |

The for loop and do-while loop are both loop constructs in the c language but they have different use cases depending on the situation and the desired loop behavior

* For Loop

The for loop is typically used when you know the exact number of iterations in advances or when you need to literate over a sequence of values. It consists of three parts: Initialization and condition and increment or decrement.

Syntax

**for (initialization; condition; increment/decrement) {**

**// Code to be executed in each iteration**

**}**

Example

**for (int i = 0; i < 5; i++) {**

**printf ("%d ", i);**

**}**

n the above example, the for loop iterates from i = 0 to i < 5, incrementing i by 1 in each iteration. It is useful when you have a fixed number of iterations and want to perform a specific task repeatedly.

* Do-while loop

The do-while loop is used when you want to execute a block of code at least once, regardless of the loop condition. It checks the loop condition after executing the loop body

Syntax

**do {**

**// Code to be executed in each iteration**

**} while (condition);**

Example

**int num = 0;**

**do {**

**printf ("%d ", num);**

**num++;**

**} while (num < 5);**

In this above example the do-while loop executes the loop body at least once because the condition is checked at the end of each iteration. It is used when you need to ensure that a certain task is performed at least once before checking the condition.

Summary you would typically use a for loop when you know the number of iterations in advance or need to iterate over a sequence of values. On the other hand, you would use a do-while loop when you want to execute a block of code at least once regardless of the loop condition.

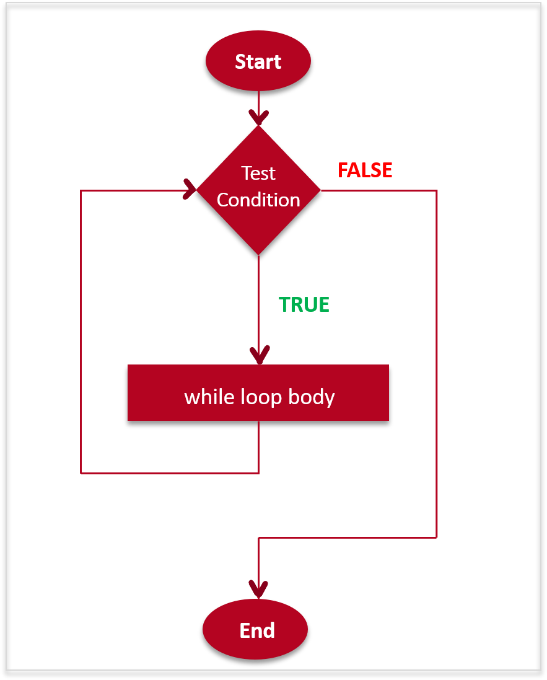
**Advantages of Switch case over if, if else**

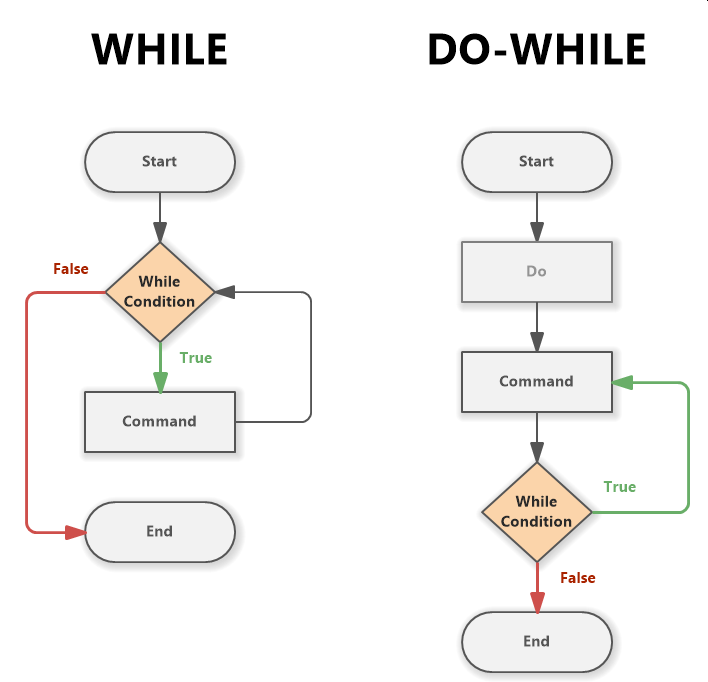
The switch case statement in C provides an alternative to using multiple if and if else statement for making decisions based on multiple possible values of a single variable.

1. Simplicity and readability: the switch case can make the code more concise and readable
2. Efficiency: in some cases, the switch case can more efficient than a chain of if else statement
3. Clarity of Intent: the switch case conveys the intention of checking a single variable against multiple values.
4. Enhancing code structure: using switch case can help improve the modularity and maintainability
5. Default case: switch case can provide a built-in default case that can be used to handle situations where none of the specified case match.

It's important to choose between if-else statements and switch cases based on the specific requirements of your code. The decision depends on factors such as the number of cases, the complexity of conditions, and the readability you want to achieve.

**Pretest Loop**

1. The process starts with an initialization step, where any necessary variables or counters are set to their initial values.
2. The loop condition is evaluated. If the condition is false, the loop terminates, and the control flow moves to the end of the loop.
3. If the loop condition is true, the loop body is executed.
4. After executing the loop body, the control flow returns to the loop condition evaluation.
5. This process continues until the loop condition becomes false, at which point the loop terminates, and the control flow moves to the end of the loop



1. The loop body is executed first.
2. After executing the loop body, the loop condition is evaluated.
3. If the loop condition is true, the control flow loops back to the beginning of the loop body and repeats the process.
4. If the loop condition is false, the loop terminates, and the control flow moves to the end of the loop.

It's important to note that in a post-test loop, the loop body always executes at least once, regardless of the initial state of the loop condition.

This flowchart represents the basic structure of a post-test loop, specifically the do-while loop in C. It can be adapted based on specific loop conditions and loop body operations.

**Guess number is even or odd**

#include<stdio.h>

#include<conio.h>

int main()

{

int a;

int i=0;

printf("Enter any Number\n");

scanf("%d",&a);

if(a%2==0){

printf("your Number is even %d",a);

}

else if(a%2==1){

printf("your Number is odd %d",a);

}

Getch ();

Return 0;

}

**Given number is prime or not**

#include <stdio.h>

#include <conio.h>

int prime(int a)

{

int i;

if (a<=1)

{

return 0;

}

for (i=2; i\*i<=a;i++)

{

if (a%i==0)

return 0;

}

return 1;

}

int main ()

{

int a;

printf("Enter a nummber \n");

scanf("%d",&a);

if(prime(a))

{

printf("Numbe is Prime %d",a);

}

else{

printf("Number is not prime");

}}

**Calculator in Switch case**

#include<stdio.h>

#include<conio.h>

int main()

{

float b,c,d;

char a;

printf("Enter any Symb [+,-,/,\*] \n");

scanf("%c",&a);

printf("Enter any Number \n");

scanf("%f",&b);

printf("Enter any number \n");

scanf("%f",&c);

switch (a)

{

case'+':

{

d=b+c;

printf("Answer is %f",d);

break;

}

case'-':

{

d=b-c;

printf("Answer is %f",d);

break;

}

case'\*':

{

d=b\*c;

printf("Answer is %f",d);

break;

}

case'/':

{ d=b/c;

printf("Answer is %f",d);

break; }

default:

printf("Error");

break; } }

**Difference b/w Post and per test loop**

pretest loops check the loop condition before executing the loop body, and if the condition is false, the loop body is skipped entirely.

Post test loops execute the loop body at least once before checking the loop condition, ensuring that the loop body is always executed before evaluating the condition for the first time.

The choice between a pertest loop and a post-test loop depends on the specific requirements of your program. If you need to guarantee that a certain block of code executes at least once, regardless of the initial condition, a post-test loop like do-while is suitable. On the other hand, if the loop should execute only when the condition is initially true, a pertest loop like while or for may be more appropriate

**What is Array**

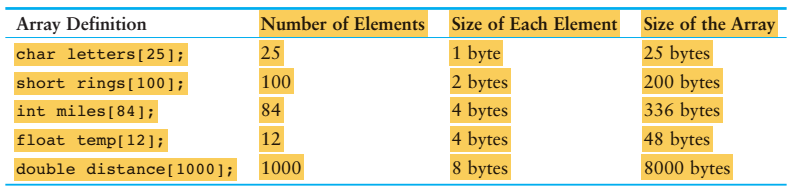
An Array is a collection of homogeneous elements. An Array allows you to store and work with multiple values of the same data type. The variable you have worked with so far are designed to hold only one at a time

Like other definitions an array definition specifies a variable type and a name. but it includes another feature that is size. The size specifies how many data items the array will contain.

An array work like a variable that can store a group of values, all of the same type. Yhe values are stored together in consecutive memory location.

Int a [5];

Element 0 to element 4



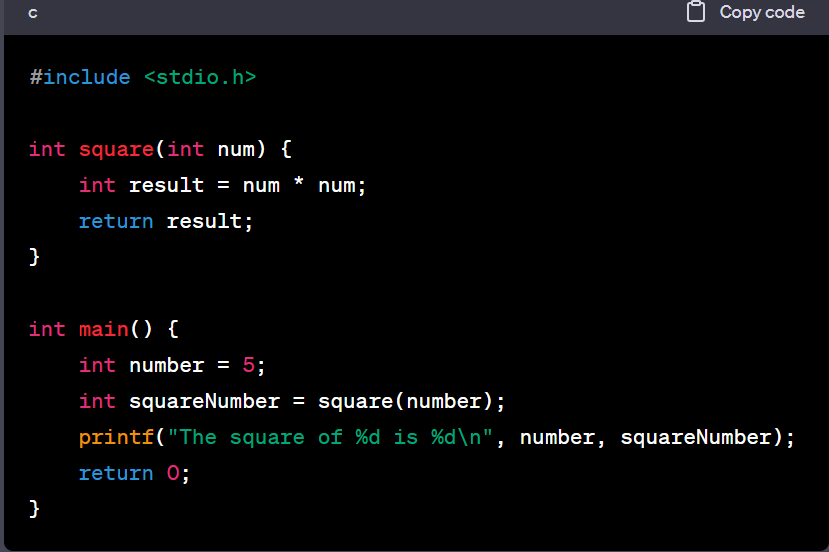
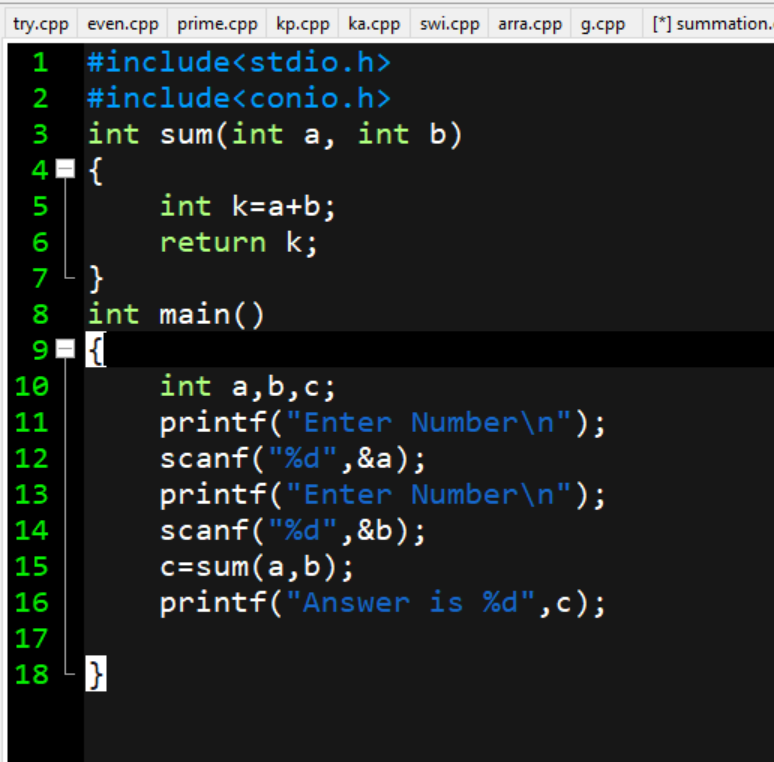
**User Define Function**

A user defined function in c programing language is a function that is created by the programmer to perform a specific task or set of tasks. It allows you to encapsulate a block of code into a separate function that can be called from different parts of the program.

**Benefits**

1. **Modularity**
2. **Abstraction**
3. **Code reusability**
4. **Structured programming**

To define a user-define function in C. you typically provide a function prototype or declaration at the beginning of your code, followed by the function definition. The function prototype specifies the return type of the function the function name, and the types of parameters. The function definition includes the actual code block that in executed when the function is called.



**What is string**

In C language programing language, a string is a sequence of the characters stored in an array. It represents textual data and is terminated by a null character. C does not have a built-in string data type like some other high-level programming language. Instead strings are typically represented as arrays of characters.

**Important**

1. Textual Data manipulation; string allow the manipulation of textual data in c. they enable you to store and process words, sentences and other textual information efficiently.
2. Input/Output; strings are used extensively for input and output operation in C.
3. String Handling Function; C provides a set of library function specifically designed for string manipulation such as strlen, strcpy, strcat, strcmp.
4. User interaction; string facilitate interaction with the user. For instance, you can prompt the user to input their name, address and other information using string.
5. Text processing; C programing that involves text processing task,
6. Data structures; strings are often used as fundamental components in various data structures.
7. File operations; strings are essential when dealing with file operations in C. they are used for reading data form text files.

