OBJECT ORINTED(OOP)

Organizes code into reusable objects that represent real world entities each with data "attributes" and behaver "methods" python" (2) lava

STRUCTURAL

Emphasizes logical flow using well defined control structures

C

Imperative

Focuses on step by step instructions for computer to follow. Like C, Java and Python.

Programming

Paradigms

Declarative

Focuses on describing what program should accomplish rather than detailing how it should do so. Like HTML web page design, SQl for database

LOGIC PROGRAMMING

Define rules and facts, and the program derives conclusions.

FUNCTIONAL PROGRAMMING

Reats computation as the evaluation of mathematical functions and avoids changing state or mutable data.

PROCEDURAL

emphasizing dividing tasks into procedures or routines.





An algorithm is a collection of precise guidelines for resolving an issue. In programming, an algorithm is constructed using a number of basic elements, such as variables, conditions, loops, functions, and more.

1.Variables

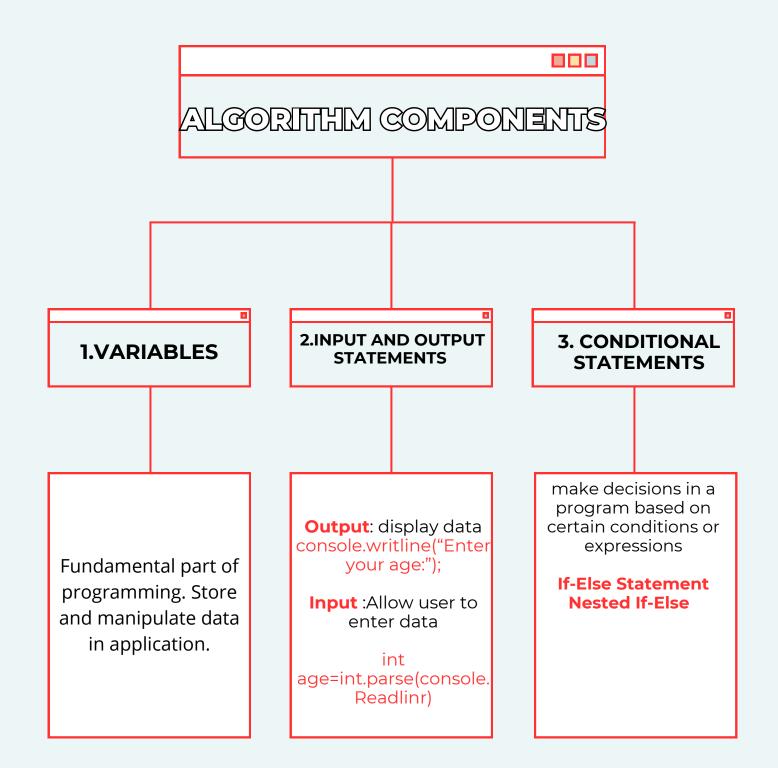
2.Input and
Output
statements

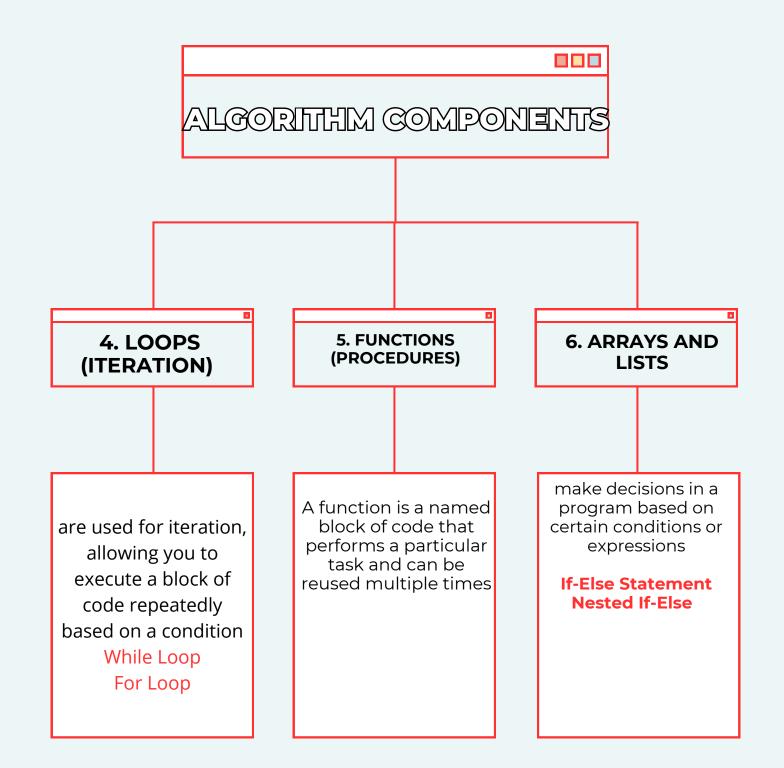
3. Conditional
Statements

4.Loops

5. Functions

6.Arrays and Lists





1.VARIABLES

1.Declaring Variables.

in C# we need a specific data type of variable when declaring it.

Syntax: dataType= varibleName=value;

Example:

int age=25; //Integer string name=Basma; //Text bool isActive=true; //Flag

2.Data Types

1.Primitive Type: int, double, bool , char

2.Reference Types: string, arrays.

3.Value Types: struct

3. Variable Scope

1.Local Variables: Declared inside a method and accessible only within that method.

2.Global Variables: Declared at the class level and accessible by all methods in that class.

4. Constant and Readonly Variables

1.Constant Variables: Their value cannot change after they are defined.

double PI=3.14159;

2.Global Variables: Can only be assigned during declaration or within the constructor.

int year=DateTime.Now.Year;

3. CONDITIONAL STATEMENTS

IF Statement

Syntax: if(condition){
code to execut if condation true }

Example: int age=25;

if (age>=25){

Console.WritLine("Age:); }

Nested If-Else

allows checking multitube
conditions
int mark=85;
if (marks >= 90)
{
Console.WriteLine("Grade: A");
}
else if (marks >= 75)
{
Console.WriteLine("Grade: B");
}
else{
Console.WriteLine("Grade: C");
}

IF-else Statements:

int age=25;
 if (age>=25){
 Console.WritLine("Age:); //If
 confation is true}
 eles {
 Console.WritLine("Age must be
 >=25.Try agin);//If confation is
 false}

4. LOOPS (ITERATION)

For Loop

used when the number iteration is know in advance

Syntax: for (initialization; condition; increment/decrement){

// Code to execute}

Example:

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

console.writLine(\${i});}

While Loop

Executes a block of code as long as the condition evaluates to true.

do-while Loop

Similar to the while loop but guarantees at least one execution of the block because the condition is checked after executing the loop body.

Types of programming languages

1.Based on Machine Interaction

1.Low-level Languages

- Machine code: directly understand by computer CPU.
 "written in binary or hexadecimal".
- Assembly language: Uses mnemonic codes instead of binary.

"requires an assembler to convert it into machine code."

2. High-level Languages

- Easier for humans to understand and write I EX: Python , Jave, C# etc.
- Requires a compiler or interpreter to translate into machine-readable code.

2.Based on Execution

1.Compiled Languages

- Code is transformed into machine code via a compiler before execution.
- C,C++,Rust.

2.Interpreted Languages

- Code is executed line-byline by an interpreter without prior compilation
- Python, JavaScript.

3. Hybrid Languages

- A mix of both compilation and interpretation.
- 4.Scripted LanguagesTypically interpreted and
- used for automating tasks, web development
- Php.

5.Just-in-Time (JIT)
Compiled Languages

- Combines aspects of compilation and interpretation; code is compiled during execution.
- .Net

| | compiler | interpreter |
|-----------------------|---|--|
| Mode of Operation: | Translates the entire program into machine code before execution. | Translates and executes code line by line during runtime. |
| Execution Speed: | Generally faster since the program is fully converted into machine code ahead of time. | Slower due to the line-by-line execution. |
| Errors | Detects all syntax errors during the compilation phase; execution doesn't proceed until errors are fixed. | Stops execution upon encountering errors, which allows debugging while running the program. |
| Output: | Produces an independent executable file; doesn't require the source code for subsequent execution. | Requires the source code every time the program is executed; no standalone output file is produced. |
| Examples | C,C++,Rust. | Python, JavaScript, Ruby. |

C# is primarily a compiled language, but it has characteristics of both compiled and interpreted languages due to its execution process.

Data Type

| Int | int Num=3; | |
|--------|--|--|
| double | double num=3.5; | |
| float | float length=3.14f; | |
| char | char grade='A'; | |
| bool | bool isRaining=true; | |
| string | string month=March; | |
| object | Can store any data type object myText="Hello world"; | |

| • | 12 | 12 | hl | |
|---|----|----|----|------------|
| V | a | а | v | IC3 |

Constants

| syntax | DataType VaribleName=Value; | const DataType ConstantName = Value; | |
|--------|--|---|--|
| EX | int num=3; Console.WritLine(\$"The Number is{num}"); | I (onsole Writtine(\$"The may Student is | |

| OutPut | Use Console.WriteLine()to print | Console.WriteLine("Enter your name") | |
|--------|--|---|--|
| Input | Use Console.ReadLine() to read input from the user | Console.WriteLine("Enter your name") string name=console.ReadLine(); Console.WritLine(\$"Hi {name}"); | |

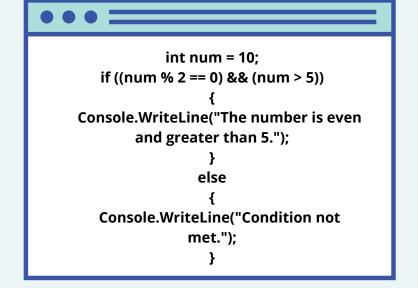
Reference Types

1. Arithmetic Operators

| Opera tor | Description | Ex |
|--------------|----------------|---|
| + | Addition | int a=2; int b=9; Console.WriteLine("Addition: " + (a + b)); |
| - | Subtraction | int a=2; int b=9; Console.WriteLine("Subtraction: " + (a - b)); |
| * | Multiplication | int a=2; int b=9; Console.WriteLine("Multiplication: " + (a * b)); |
| / | Division | int a=2; int b=9; Console.WriteLine("Division: " + (a / b)); |
| % | Modulus | int a=2; int b=9; Console.WriteLine("Modulus: " + (a % b)); |

2. Logical Operators

| Opera tor | Description | Ex |
|--------------|-------------|-----------------------|
| && | AND | (a<0&&b>0) |
| П | OR | (a<0 b>0) |
| ! | NOT | !(a <b)< th=""></b)<> |



1. If-Else Statement

```
Syntax:
if (condition)
{
    // Code to execute if condition is true}
else if (anotherCondition)
{
    // Code to execute if anotherCondition is true}
else{
    // Code to execute if none of the above conditions are true}
```

Example

```
static void Main()
{ int age = 20;

if (age < 18)
{ Console.WriteLine("You are a minor."); }
else if (age >= 18 && age <= 60) {
    Console.WriteLine("You are an adult."); }
else {
    Console.WriteLine("You are a senior citizen.");
}
}
```

2. Switch-Case Statement

Syntax:

```
switch (expression)
{
   case value1:
     // Code to execute for value1break;
   case value2:
     // Code to execute for value2break;
   default:
     // Code to execute if no case matchesbreak;
}
```

```
static void Main()
{
    char grade = 'B';

    switch (grade)
    {
        case 'A': Console.WriteLine("Excellent!"); break;
        case 'B': Console.WriteLine("Good job!"); break;
        case 'C': Console.WriteLine("Fair effort."); break;
        default: Console.WriteLine("Grade not recognized."); break;
    }
}
```

For Loop

Syntax:

```
for (initialization; condition; increment/decrement)
{
   // Code to execute
}
```

Used when the number of iterations is known beforehand.

Example

```
static void Main()
{
    for (int i = 1; i <= 5; i++)
    {
        Console.WriteLine("Iteration " + i);
    }
}
```

While Loop

Syntax:

```
while (condition)
{
  // Code to execute
}
```

Used when the condition is checked before entering the loop.

```
static void Main()
{
  int count = 1;
  while (count <= 5)
  {
    Console.WriteLine("Count: " + count);
    count++;
  }
```

3. Do-While Loop

Syntax:

```
do
{
   // Code to execute
} while (condition);
```

Used when the code block needs to be executed at least once, as the condition is checked after the execution

```
static void Main()
{
  int count = 1;
  do
  {
    Console.WriteLine("Count: " + count);
    count++;
  } while (count <= 5);
}</pre>
```

1. Nested Conditions

occurs when an if or else statement contains another if-else statement inside it.

Example

```
static void Main()
  int age = 25;
  bool hasLicense = true;
  if (age \geq 18)
    if (hasLicense)
       Console.WriteLine("You are allowed to drive.");
    else
       Console.WriteLine("You need a valid license to drive.");
  else
    Console.WriteLine("You are too young to drive.");
```

2. Nested Loops

Syntax:

involves placing one loop inside another.

```
static void Main()
{
    for (int i = 1; i <= 3; i++) // Outer loop
    {
        for (int j = 1; j <= 3; j++) // Inner loop
        {
            Console.WriteLine($"i: {i}, j: {j}");
        }
    }
}</pre>
```

Array

An array in C# is a fixed-size data structure designed to hold multiple elements of the same type in a sequential order.

1.Array Declaration and Initialization

```
// Declaring an array
```

int[] numbers;

// Initializing an array with size

numbers = new int[5]; // Array with 5 elements

// Declaring and initializing in one step

int[] numbers = { 85, 90, 78, 92, 88 };

Loop Through an array

```
for (int i = 0; i < scores.Length; i++)
{
    Console.WriteLine("Score " + (i + 1) +
": " + scores[i]);
}</pre>
```

Multidimensional Arrays

```
int[,] matrix = new int[2, 3]
{
     { 1, 2, 3 },
     { 4, 5, 6 }
};
Console.WriteLine(matrix[0, 1]);
```

Methods in array

```
.Length
Array.Sort()_
Array.Reverse
Array.IndexOf()
```

Jagged Arrays

```
int[,] matrix = new int[2, 3]
{
     { 1, 2, 3 },
     { 4, 5, 6 }
};
Console.WriteLine(matrix[0, 1]);
```

Functions in CSharp (built-in)

| Function | Description | EX |
|-------------------------------|---|---|
| Math.Abs() | Returns the absolute value of a number | Math.Abs(-5); |
| Math.Pow() | Raises a number to a power | Math.Pow(2,4); |
| Math.Sqrt() | Returns the square root of a number | Math.Sqrt(16); |
| Math.Max and Math.Min() | Return the larger or smaller of two numbers | int max = Math.Max(8,7); max = 10 int min = Math.Min(8,5); |
| Math.Round() | Round a number to the nearest integer. | Math.Round(3.7) |

| Function | Function Description | |
|------------------|--|---|
| string.Length | Returns the absolute value of a number | Math.Abs(-5); |
| string.Substring | Raises a number to a power | Math.Pow(2,4); |
| string.ToUpper | Returns the square root of a number | Math.Sqrt(16); |
| string.ToLower | Return the larger or smaller of two numbers | int max = Math.Max(8,7); max = 10 int min = Math.Min(8,5); |
| string.Trim | Round a number to the nearest integer. | Math.Round(3.7) |