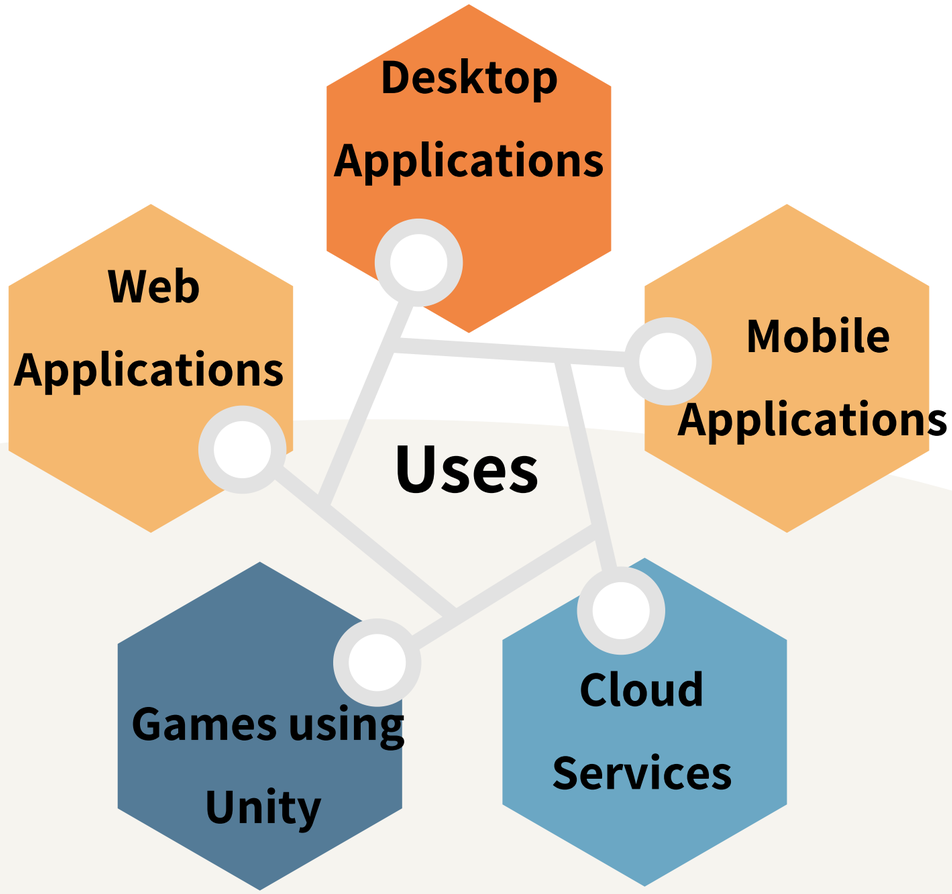
Introduction to C#

What is C#

C# is a high-level object-oriented programming language. It was developed by Microsoft and it is widely used for building various types of applications.

Features:

- **Object-Oriented:** Follows the principles of encapsulation, inheritance, and polymorphism.
- **Type-Safe:** Reduces runtime errors by checking data types during compilation.
- **Rich Library Support:** Comes with a vast .NET library for common functions.



What is .NET Framework:

Is an integrated development platform developed by Microsoft. It allows programmers to create applications that run on the Windows operating system and provides many libraries and functions (APIs) ready to speed up the development process. primarily used for creating desktop, web, and enterprise applications.

The .NET Framework includes two main components:

- **Common Language Runtime (CLR):** responsible for managing the execution of code written in any of the supported languages
- **NET command-line interface (CLI)** is a cross-platform toolchain for developing, building, running, and publishing. NET applications.

What is a Console Application?

A console application is a basic computer program that you use by typing text commands. There are no fancy graphics or buttons, it's all about the words you type. When you ask the program to do something by typing a command, it gives you back information in text, too. Think of it like texting with your computer to get tasks done.

The user types a command, the application processes it, and then it outputs the results in text form. This simplicity allows for efficient execution of tasks without the overhead of graphics.

What is a Console Application?

Advantages of Console Applications:

a. Resource Efficiency: Console applications are lightweight and consume minimal system resources

b. Simplicity: With a focus on text-based input and output, console applications are simpler to design and implement. This simplicity reduces development time and complexity.

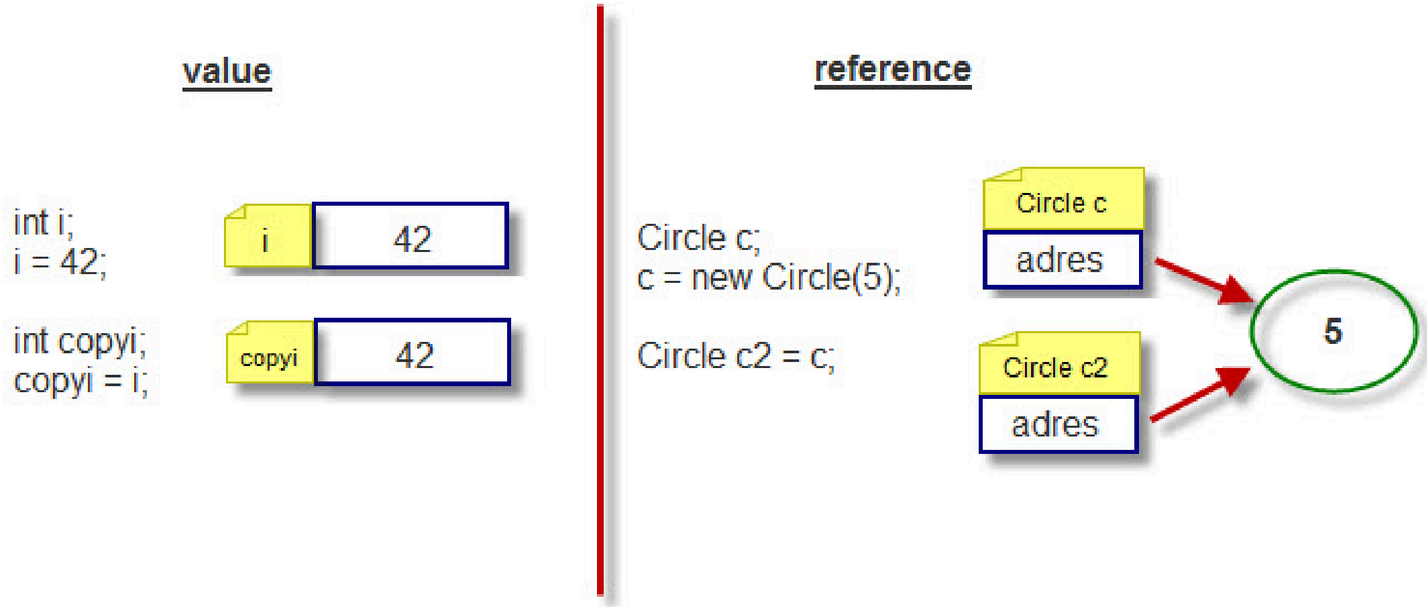
c. Portability: Console applications can be easily ported across different operating systems.

d. Speed: The absence of graphical elements allows console applications to run faster.

What are Variables?

Primitive Types:

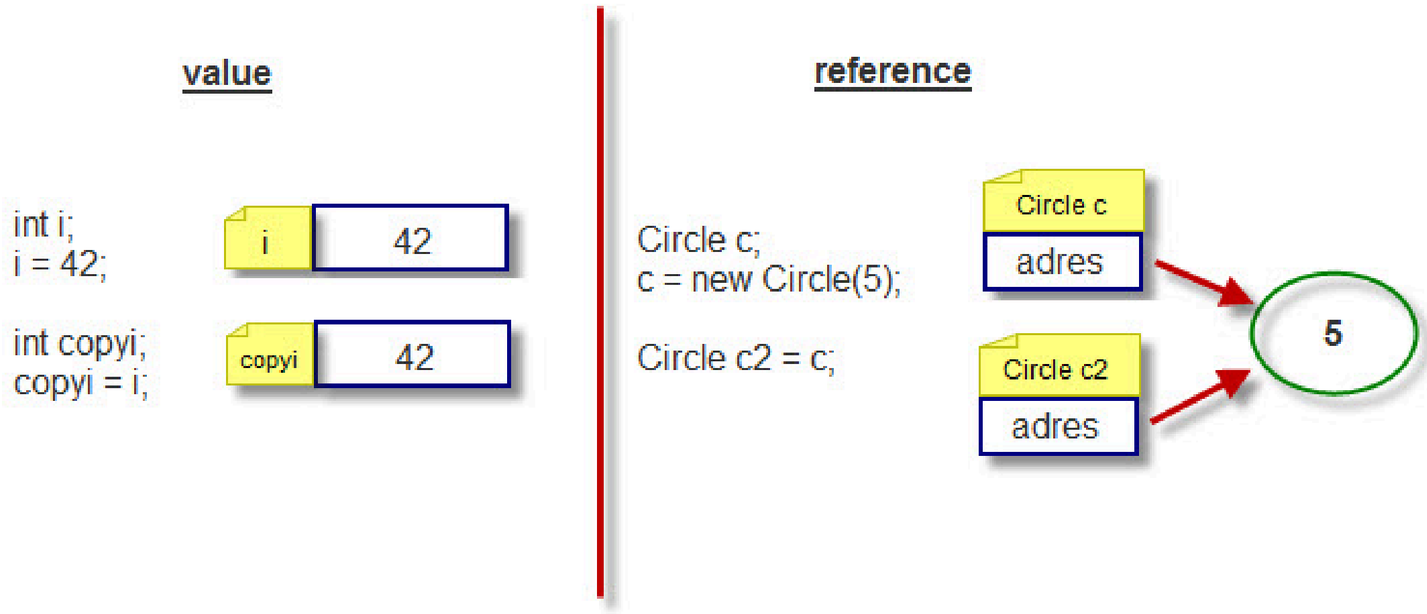
1. int: Used to store integer values.
2. double: Used for storing floating-point numbers (decimals).
3. char: Used to store a single character.
4. bool: Used to store Boolean values (true or false).



What are Variables?

Reference Types:

1. string: Used to store a sequence of characters.
2. arrays: Used to store multiple items of the same type.
3. classes: Used to define objects with properties and methods.



General Rules for Creating Variables

Naming Rules:

1. Must begin with a letter or an underscore (_).
2. Cannot use special characters (e.g., @, #, $).
3. Cannot use reserved keywords (e.g., int, class).
4. Must not have spaces.
5. Should use meaningful names for readability (age, totalPrice).

General Rules for Creating Variables

Case Sensitivity:

1. Variable names are case-sensitive (myVar is different from MyVar).

Type-Safety:

1. The value assigned must match the variable's data type.

Default Values:

1. Value types like int, double, etc., are initialized to 0 by default.
2. Reference types like string and objects are null by default.

Syntax

```
Datatype variable Name = value;
```

Example:

```
int age = 25;
double salary = 5000.50;
char grade = 'A';
bool isEmployed = true;
string name = "Ali";
```

Creating multiple variables:

```
int age = 25, score = 25;
```

Arrays in C#

Syntax

```
<type> [] <arrayName> = new <type>[<size>];
```

Example

```
int [] numbers = new int[5];
```

Initialize

```
int[] numbers = { 10, 20, 30, 40, 50 };
```

Accessing Elements in an Array

Array indexing starts from 0. Access elements using square brackets []:

```
numbers[0];        numbers[0] = 5;      { 5, 20, 30, 40, 50 };
```

User input (read / write)

WriteLine is used to print a message to the console. After printing, it moves the cursor to the next line.

```
Console.WriteLine("Hello, World!");
Console.WriteLine("This is C#.");
```

```
Hello, World!
This is C#.
```

User input (read / write)

ReadLine is used to read input from the user. It waits for the user to type something and press Enter.

The input is always read as a string.

```
Console.WriteLine("Enter your name:");
string name = Console.ReadLine();
Console.WriteLine("Hello, " + name + "!");
```