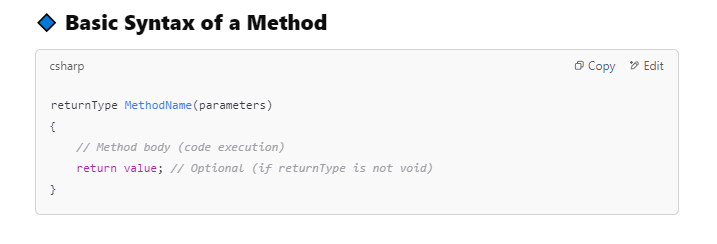
**Detailed Explanation of Methods in C#**

* **A method in C# is a block of code that performs a specific task.**
* **It helps in code organization, reusability, and reducing redundancy.**
* **Methods can take parameters, perform operations, and return values.**

***Defining a Method in C#***

* **A method consists of several key parts:**
* **Return Type: Specifies the type of data the method returns.**
* **If it does not return anything, we use void.**
* **Method Name: Should be unique and descriptive of its functionality.**
* **Parameters (Optional): Values that can be passed to the method for processing.**
* **Method Body: Contains the instructions that execute when the method is called.**
* **Return Statement (Optional): Used to return a value if the return type is not void.**

****

***Types of Methods in C#***

* **Methods with a Return Value**

**Returns a specific type like int, string, bool, etc.**

* **Void Methods (No Return Value)**

**Performs an action but does not return anything.**

* **Methods with Parameters**

**Accepts input values (parameters) for processing inside the method.**

* **Methods without Parameters**

**Does not require input values, just executes an operation.**

***Static Methods vs. Instance Methods***

**1️- Static Methods**

* **Can be called without creating an object (object) of the class.**
* **Belong to the class itself rather than objects.**
* **A screenshot of a computer program

  Description automatically generated**

**Instance Methods**

* **Require creating an object of the class before calling the method.**
* **Belong to objects rather than the class itself.**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer program

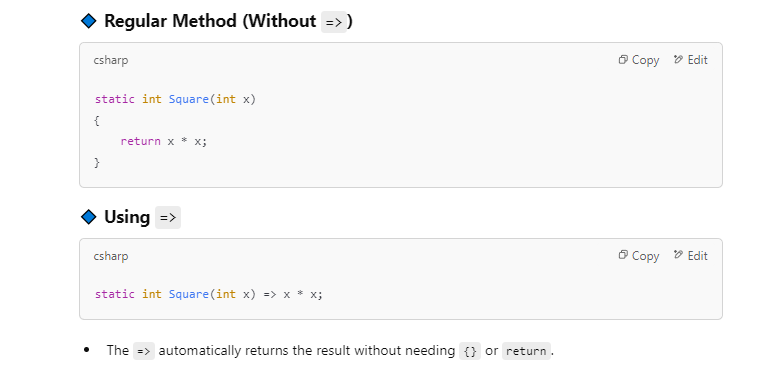
Description automatically generated**

***A screenshot of a computer program

Description automatically generated***

***Using => (Arrow Expression) in C#***

* **In C#, the => (Arrow Expression) is used to simplify method and property definitions.**
* **It is known as an "Expression-bodied Member" and helps make the code more concise.**
* **Using => in Methods**
* **You can use => instead of {} when the method contains only a single expression**.

******

***When to Use =>?***

* **When the method has only one expression.**
* **To make code shorter and more readable.**
* **When defining simple properties in objects.**

***Pass by Value vs. Pass by Reference in C#***

* **In C#, when passing arguments to methods, you can do it in two ways:**
* **Pass by Value (default behavior)**
* **Pass by Reference (ref, out, in)**

***1. Pass by Value (Default Behavior*)**

* **When passing a variable by value, a copy of the variable is sent to the method.**
* **Changes inside the method do not affect the original variable.**
* **Default behavior for value types (int, double, char, bool, etc.).**
* **A screenshot of a computer

  Description automatically generated**

***2. Pass by Reference (ref, out, in)***

* **When passing a variable by reference, the method receives a reference to the original variable, not a copy.**
* **Changes inside the method affect the original variable.**
* **Used with the ref, out, or in keywords.**

**A screenshot of a computer program

Description automatically generated**

**out (Pass by Reference, Must Assign New Value)**

* **The variable does not need to be initialized before passing it.**
* **The method must assign a value before it exits.**
* **Used when a method needs to return multiple values.**
* **A screenshot of a computer program

  Description automatically generated**

**in (Pass by Reference, Read-Only)**

* **The method cannot modify the variable (read-only).**
* **Used for performance optimization (especially with large objects).**

**A screenshot of a computer program

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**Summary**

1. **Pass by Value: The method gets a copy, and changes do not affect the original variable.**
2. **Pass by Reference (ref): The method modifies the original variable.**
3. **Pass by Reference (out): Used when the method must assign a new value.**
4. **Pass by Reference (in): The variable is read-only, preventing modification.**

**//This program is coded by mostafa ahmed in 22/2/2025**

**using System;**

**class User**

**{**

**public string Name { get; set; }**

**public int Age { get; set; }**

**public double ID { get; set; }**

**public string Adress { get; set; }**

**// Constructor to initialize user details**

**public User(string name, int age, double iD, string adress)**

**{**

**Name = name;**

**Age = age;**

**ID = iD;**

**Adress = adress;**

**}**

**}**

**class UserManager**

**{**

**// intialize an array that can contain 100 users**

**private User[] users = new User[100];**

**private int count = 0; // a count to indicates the number of places that have users**

**// Method to add a new user**

**public void AddUser(string name, int age,double iD,string adress)**

**{**

**//check if user list has an empty place or not**

**if (count < users.Length)**

**{**

**users[count] = new User(name, age,iD,adress);**

**count++; // Increment user count (add the desired user)**

**Console.WriteLine($"User {name} added successfully.");**

**}**

**// the list is full**

**else**

**{**

**Console.WriteLine("User list is full.");**

**}**

**}**

**// Method to remove a user by his name**

**public void RemoveUser(double id)**

**{**

**// check if list is emty or not**

**if (count == 0)**

**{**

**Console.WriteLine("There is no user to remove ! user list is empty!");**

**return;**

**}**

**else**

**{**

**for (int i = 0; i < count; i++)**

**{**

**if (users[i].ID == id)**

**{**

**users[i] = users[count - 1]; // Swap with last element**

**users[count - 1] = null; // Clear last element**

**count--; // number of user will be decresed**

**Console.WriteLine($"User {id} removed.");**

**return;**

**}**

**}**

**}**

**Console.WriteLine($"User {id} not found.");**

**}**

**// Method to dislay the users by name and age**

**public void DisplayUsers()**

**{**

**Console.WriteLine("User List: ");**

**if (count == 0) //Check if there are users**

**{**

**Console.WriteLine("No users available.");**

**}**

**else**

**{**

**for (int i = 0; i < count; i++) // Loop through avilable users**

**{**

**Console.WriteLine($"Name: {users[i].Name}, Adress: {users[i].Adress}, id: {users[i].ID}, Age: {users[i].Age}");**

**}**

**}**

**}**

**// Method to search for a user by name**

**public void SearchUser(string name)**

**{**

**for (int i = 0; i < count; i++)**

**{**

**if (users[i].Name==(name)) // Check if name matches**

**{**

**Console.WriteLine($"User Found! Name: {users[i].Name}, Adress: {users[i].Adress}, id: {users[i].ID}, Age: {users[i].Age}");**

**return;**

**}**

**}**

**Console.WriteLine($"User {name} not found."); // If user not found**

**}**

**}**

**class Program**

**{**

**static void Main()**

**{**

**UserManager userManager = new UserManager();**

**while (true)**

**{**

**Console.WriteLine("Welcome!!");**

**Console.WriteLine("1. Add User");**

**Console.WriteLine("2. Remove User");**

**Console.WriteLine("3. Display All Users");**

**Console.WriteLine("4. Search User");**

**Console.WriteLine("5. Exit");**

**Console.Write("Enter choice: ");**

**string choice = Console.ReadLine();**

**Console.Clear(); // Clear console for better readability**

**switch (choice)**

**{**

**case "1":**

**Console.Write("Enter The Name: ");**

**string Name = Console.ReadLine();**

**Console.Write("Enter The adress: ");**

**string adress = Console.ReadLine();**

**Console.Write("Enter id: ");**

**double id = double.Parse(Console.ReadLine());**

**Console.Write("Enter Age: ");**

**int age = int.Parse(Console.ReadLine());**

**userManager.AddUser(Name, age, id,adress);**

**break;**

**case "2":**

**Console.Write("Enter Id of user to Remove: ");**

**double removeId = double.Parse(Console.ReadLine());**

**userManager.RemoveUser(removeId);**

**break;**

**case "3":**

**userManager.DisplayUsers();**

**break;**

**case "4":**

**Console.Write("Enter Name to Search: ");**

**string searchName = Console.ReadLine();**

**userManager.SearchUser(searchName);**

**break;**

**case "5":**

**return;**

**default:**

**Console.WriteLine("Invalid choice. Try again.");**

**break;**

**}**

**Console.WriteLine("Press any key to continue...");**

**Console.ReadKey();**

**Console.Clear(); // Clear console for better readability**

**}**

**}**

**}**