Removing Elements from List in C# - Without Lambda & References

Complete Code Examples

```
using System;
using System.Collections;
using System.Collections.Generic;

namespace ListRemovalExamples
{
    class Student
    {
        public int Id { get; set; }
        public string Name { get; set; }
        public int Age { get; set; }

        public Student(int id = 0, string name = " ", int age = 6)
        {
            this.Id = id;
            this.Name = name;
            this.Age = age;
        }

        public override string ToString()
        {
            return $"{Id}-{Name}-{Age} years old";
        }

        public override int GetHashCode()
        {
            return HashCode.Combine(Id, Name, Age);
        }

        public override bool Equals(object? obj)
        {
            Student s = obj as Student;
            return (Id == s.Id && Name == s.Name && Age == s.Age);
        }
    }

    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("=== REMOVING ELEMENTS FROM LIST ===\n");

            // Method 1: Remove by Value (Value Types)
            RemoveByValue_ValueType();

            // Method 2: Remove by Index
            RemoveByIndex();

            // Method 3: Remove Multiple Elements by Value
            RemoveMultipleByValue();

            // Method 4: Remove by Condition (Manual Loop)
            RemoveByCondition();

            // Method 5: Remove All Matching Elements
            RemoveAllMatching();

            // Method 6: Remove from Student List (Reference Type)
            RemoveFromStudentList();

            // Method 7: Remove Range of Elements
            RemoveRange();

            // Method 8: Clear Entire List
            ClearList();

            // Method 9: Remove First/Last Element
            RemoveFirstLast();

            // Method 10: Remove Duplicates
            RemoveDuplicates();
        }

        #region Method 1: Remove by Value (Value Types)
        static void RemoveByValue_ValueType()
        {
            Console.WriteLine("Method 1: Remove by Value (Value Types)");
            Console.WriteLine("=========================================");

            List<int> numbers = new List<int>() { 10, 20, 30, 20, 40, 50, 20
};

            Console.WriteLine("Original List:");
            PrintList(numbers);
            // Remove first occurrence of 20
            numbers.Remove(20);

            Console.WriteLine("\nAfter Remove(20) - Removes FIRST occurrence 
only:");
            PrintList(numbers);

            Console.WriteLine("\n");
        }
        #endregion

        #region Method 2: Remove by Index
        static void RemoveByIndex()
        {
            Console.WriteLine("Method 2: Remove by Index");
            Console.WriteLine("=========================");

            List<string> names = new List<string>()
            {
                "Ali", "Sara", "Omar", "Fatma", "Ahmed"
            };

            Console.WriteLine("Original List:");
            PrintList(names);

            // Remove element at index 2 (Omar)
            names.RemoveAt(2);

            Console.WriteLine("\nAfter RemoveAt(2):");
            PrintList(names);

            Console.WriteLine("\n");
        }
        #endregion

        #region Method 3: Remove Multiple Elements by Value
        static void RemoveMultipleByValue()
        {
            Console.WriteLine("Method 3: Remove Multiple Elements by Value");
            Console.WriteLine("============================================");

            List<int> numbers = new List<int>() { 10, 20, 30, 20, 40, 50, 20 
};

            Console.WriteLine("Original List:");
            PrintList(numbers);

            // Remove all occurrences of 20 using loop
            Console.WriteLine("\nRemoving all occurrences of 20...");

            // Method A: While loop (backwards safe)
            int i = numbers.Count - 1;
            while (i >= 0)
            {
                if (numbers[i] == 20)
                {
                    numbers.RemoveAt(i);
                }
                i--;
            }

            Console.WriteLine("After removing all 20s:");
            PrintList(numbers);

            Console.WriteLine("\n");
        }
        #endregion

        #region Method 4: Remove by Condition (Manual Loop)
        static void RemoveByCondition()
        {
            Console.WriteLine("Method 4: Remove by Condition (Manual Loop)");
            Console.WriteLine("============================================");

            List<int> numbers = new List<int>()
            {
                5, 12, 8, 20, 15, 3, 18, 25, 7
            };

            Console.WriteLine("Original List:");
            PrintList(numbers);

            // Remove all numbers greater than 15
            Console.WriteLine("\nRemoving all numbers > 15...");

            // Loop backwards to avoid index issues
            for (int i = numbers.Count - 1; i >= 0; i--)
            {
                if (numbers[i] > 15)
                {
                    numbers.RemoveAt(i);
                }
            }

            Console.WriteLine("After removal:");
            PrintList(numbers);

            Console.WriteLine("\n");
        }
        #endregion

        #region Method 5: Remove All Matching Elements
        static void RemoveAllMatching()
        {
            Console.WriteLine("Method 5: Remove All Matching Elements");
            Console.WriteLine("=======================================");

            List<int> numbers = new List<int>()
            {
                1, 2, 3, 4, 5, 6, 7, 8, 9, 10
            };

            Console.WriteLine("Original List:");
            PrintList(numbers);

            // Remove all even numbers (Manual way without lambda)
            Console.WriteLine("\nRemoving all even numbers...");

            for (int i = numbers.Count - 1; i >= 0; i--)
            {
                if (numbers[i] % 2 == 0)  // Even number
                {
                    numbers.RemoveAt(i);
                }
            }

            Console.WriteLine("After removing even numbers:");
            PrintList(numbers);

            Console.WriteLine("\n");
        }
        #endregion

        #region Method 6: Remove from Student List
        static void RemoveFromStudentList()
        {
            Console.WriteLine("Method 6: Remove from Student List");
            Console.WriteLine("===================================");

            List<Student> students = new List<Student>()
            {
                new Student(1, "Ali", 22),
                new Student(2, "Sara", 20),
                new Student(3, "Omar", 25),
                new Student(4, "Fatma", 19),
                new Student(5, "Ahmed", 23)
            };

            Console.WriteLine("Original List:");
            foreach (Student s in students)
            {
                Console.WriteLine(s);
            }

            // Remove student with Id = 3 (without reference)
            Console.WriteLine("\nRemoving student with Id = 3...");

            // Method A: Find index first, then remove
            int indexToRemove = -1;
            for (int i = 0; i < students.Count; i++)
            {
                if (students[i].Id == 3)
                {
                    indexToRemove = i;
                    break;
                }
            }

            if (indexToRemove >= 0)
            {
                students.RemoveAt(indexToRemove);
                Console.WriteLine("Student removed successfully!");
            }

            Console.WriteLine("\nAfter removal:");
            foreach (Student s in students)
            {
                Console.WriteLine(s);
            }

            // Remove all students with Age > 22
            Console.WriteLine("\n\nRemoving all students with Age > 22...");
```

```
            for (int i = students.Count - 1; i >= 0; i--)
            {
                if (students[i].Age > 22)
                {
                    students.RemoveAt(i);
                }
            }

            Console.WriteLine("After removal:");
            foreach (Student s in students)
            {
                Console.WriteLine(s);
            }

            Console.WriteLine("\n");
        }
        #endregion

        #region Method 7: Remove Range of Elements
        static void RemoveRange()
        {
            Console.WriteLine("Method 7: Remove Range of Elements");
            Console.WriteLine("===================================");

            List<int> numbers = new List<int>()
            {
                10, 20, 30, 40, 50, 60, 70, 80
            };

            Console.WriteLine("Original List:");
            PrintList(numbers);

            // Remove 3 elements starting from index 2
            Console.WriteLine("\nRemoving 3 elements starting from index
2...");
            numbers.RemoveRange(2, 3);  // Remove elements at index 2, 3, 4

            Console.WriteLine("After RemoveRange(2, 3):");
            PrintList(numbers);

            Console.WriteLine("\n");
        }
        #endregion

        #region Method 8: Clear Entire List
        static void ClearList()
        {
            Console.WriteLine("Method 8: Clear Entire List");
            Console.WriteLine("============================");

            List<string> names = new List<string>()
            {
                "Ali", "Sara", "Omar"
            };

            Console.WriteLine("Original List:");
            PrintList(names);
            Console.WriteLine($"Count: {names.Count}");

            // Clear all elements
            names.Clear();

            Console.WriteLine("\nAfter Clear():");
            PrintList(names);
            Console.WriteLine($"Count: {names.Count}");

            Console.WriteLine("\n");
        }
        #endregion

        #region Method 9: Remove First/Last Element
        static void RemoveFirstLast()
        {
            Console.WriteLine("Method 9: Remove First/Last Element");
            Console.WriteLine("====================================");

            List<int> numbers = new List<int>()
            {
                10, 20, 30, 40, 50
            };

            Console.WriteLine("Original List:");
            PrintList(numbers);

            // Remove first element
            Console.WriteLine("\nRemoving first element...");
            numbers.RemoveAt(0);

            Console.WriteLine("After removing first:");
            PrintList(numbers);

            // Remove last element
            Console.WriteLine("\nRemoving last element...");
            numbers.RemoveAt(numbers.Count - 1);

            Console.WriteLine("After removing last:");
            PrintList(numbers);

            Console.WriteLine("\n");
        }
        #endregion

        #region Method 10: Remove Duplicates
        static void RemoveDuplicates()
        {
            Console.WriteLine("Method 10: Remove Duplicates");
            Console.WriteLine("============================");

            List<int> numbers = new List<int>()
            {
                10, 20, 10, 30, 20, 40, 10, 50, 30
            };

            Console.WriteLine("Original List:");
            PrintList(numbers);

            // Remove duplicates - keep first occurrence
            Console.WriteLine("\nRemoving duplicates...");

            List<int> uniqueNumbers = new List<int>();

            foreach (int num in numbers)
            {
                // Check if number already exists
                bool exists = false;
                foreach (int unique in uniqueNumbers)
                {
                    if (unique == num)
                    {
                        exists = true;
                        break;
                    }
                }

                // Add only if it doesn't exist
                if (!exists)
                {
                    uniqueNumbers.Add(num);
                }
            }

            Console.WriteLine("After removing duplicates:");
            PrintList(uniqueNumbers);

            Console.WriteLine("\n");
        }
        #endregion

        #region Helper Method
        static void PrintList<T>(List<T> list)
        {
            Console.Write("[ ");
            for (int i = 0; i < list.Count; i++)
            {
                Console.Write(list[i]);
                if (i < list.Count - 1)
                    Console.Write(", ");
            }
            Console.WriteLine(" ]");
        }
        #endregion
    }
}
```

Important Notes & Common Mistakes

❌ WRONG: Removing While Looping Forward

```
// ❌ DON'T DO THIS - Will skip elements!
List<int> numbers = new List<int>() { 10, 20, 30, 20, 40 };

for (int i = 0; i < numbers.Count; i++)
{
    if (numbers[i] == 20)
    {
        numbers.RemoveAt(i);  // ❌ Index shifts, skips next element!
    }
}

// Result: Only first 20 is removed, second 20 is skipped!
```

Why it fails:

```
Initial:  [10, 20, 30, 20, 40]
           i=0  i=1  i=2  i=3 i=4

Step 1: i=0, value=10, no removal
Step 2: i=1, value=20, REMOVE!
        [10, 30, 20, 40]  ← List shifts left
         i=0 i=1  i=2 i=3
Step 3: i=2, value=20  ← We skipped 30!
```

✅ CORRECT: Loop Backwards

```
// ✅ CORRECT WAY - Loop backwards
List<int> numbers = new List<int>() { 10, 20, 30, 20, 40 };

for (int i = numbers.Count - 1; i >= 0; i--)
{
    if (numbers[i] == 20)
    {
        numbers.RemoveAt(i);  // ✅ Safe - no skip!
    }
}

// Result: Both 20s are removed correctly!
```

Why it works:

```
Initial:  [10, 20, 30, 20, 40]
           i=0  i=1  i=2  i=3 i=4

Step 1: i=4, value=40, no removal
Step 2: i=3, value=20, REMOVE!
        [10, 20, 30, 40]  ← We already processed right side
         i=0 i=1  i=2 i=3
Step 3: i=2, value=30, no removal
Step 4: i=1, value=20, REMOVE!
        [10, 30, 40]  ← All processed correctly!
```

Additional Examples for Your Code

Example 1: Remove Student by Name

```
List<Student> students = new List<Student>()
{
    new Student(1, "Ali", 22),
    new Student(2, "Sara", 20),
    new Student(3, "Omar", 25),
    new Student(4, "Ali", 19)  // Duplicate name
};

Console.WriteLine("Original List:");
foreach (Student s in students)
{
    Console.WriteLine(s);
}

// Remove all students named "Ali"
Console.WriteLine("\nRemoving all students named 'Ali'...");

for (int i = students.Count - 1; i >= 0; i--)
{
    if (students[i].Name == "Ali")
    {
        students.RemoveAt(i);
    }
}

Console.WriteLine("\nAfter removal:");
foreach (Student s in students)
{
    Console.WriteLine(s);
}

// Output:
// 2-Sara-20 years old
// 3-Omar-25 years old
```

Example 2: Remove from Dictionary-List Structure

```
Dictionary<Student, List<Subject>> dic = new Dictionary<Student,
List<Subject>>();

Student student1 = new Student(1, "Ali", 22);
Student student2 = new Student(2, "Sara", 20);
Student student3 = new Student(3, "Omar", 25);
```

```
dic.Add(student1, new List<Subject>()
{
    new Subject(1, "C#", 60),
    new Subject(2, "DB", 30)
});

dic.Add(student2, new List<Subject>()
{
    new Subject(1, "C#", 60),
    new Subject(3, "DS", 24)
});

dic.Add(student3, new List<Subject>()
{
    new Subject(2, "DB", 30)
});

// Remove subject with Id = 2 from all students
Console.WriteLine("Removing subject with Id=2 from all students...");

foreach (KeyValuePair<Student, List<Subject>> item in dic)
{
    List<Subject> subjects = item.Value;

    // Remove subject from this student's list
    for (int i = subjects.Count - 1; i >= 0; i--)
    {
        if (subjects[i].id == 2)
        {
            subjects.RemoveAt(i);
        }
    }
}

// Display results
foreach (KeyValuePair<Student, List<Subject>> item in dic)
{
    Console.WriteLine($"\n{item.Key}");
    foreach (Subject subj in item.Value)
    {
        Console.WriteLine($"  {subj}");
    }
}
```

Example 3: Remove Empty/Null Entries

```
List<string> names = new List<string>()
{
    "Ali",
    "",
    "Sara",
    null,
    "Omar",
    "   ",  // Whitespace
    "Ahmed"
};

Console.WriteLine("Original List:");
foreach (string name in names)
{
    Console.WriteLine($"'{name}'");
}

// Remove null, empty, or whitespace strings
Console.WriteLine("\nRemoving empty/null/whitespace entries...");

for (int i = names.Count - 1; i >= 0; i--)
{
    if (string.IsNullOrWhiteSpace(names[i]))
    {
        names.RemoveAt(i);
    }
}

Console.WriteLine("\nAfter removal:");
foreach (string name in names)
{
    Console.WriteLine($"'{name}'");
}

// Output:
// 'Ali'
// 'Sara'
// 'Omar'
// 'Ahmed'
```

Example 4: Remove Based on Multiple Conditions

```
List<Student> students = new List<Student>()
{
    new Student(1, "Ali", 17),     // Minor
    new Student(2, "Sara", 22),     // Adult
    new Student(3, "Omar", 15),     // Minor
    new Student(4, "Fatma", 25),    // Adult
    new Student(5, "Ahmed", 19)     // Adult
};

Console.WriteLine("Original List:");
foreach (Student s in students)
{
    Console.WriteLine(s);
}

// Remove students who are minors (age < 18) OR have Id > 4
Console.WriteLine("\nRemoving students: Age < 18 OR Id > 4...");

for (int i = students.Count - 1; i >= 0; i--)
{
    Student currentStudent = students[i];

    // Check multiple conditions
    bool shouldRemove = false;

    if (currentStudent.Age < 18)
    {
        shouldRemove = true;
    }

    if (currentStudent.Id > 4)
    {
        shouldRemove = true;
    }

    if (shouldRemove)
    {
        students.RemoveAt(i);
    }
}

Console.WriteLine("\nAfter removal:");
foreach (Student s in students)
{
    Console.WriteLine(s);
}

// Output:
```

```
// 2-Sara-22 years old
// 4-Fatma-25 years old
```

Example 5: Remove Using Helper Method

```
// Helper method to check if student should be removed
static bool ShouldRemoveStudent(Student student)
{
    // Remove if age is not in range 18-30
    if (student.Age < 18 || student.Age > 30)
        return true;

    // Remove if name starts with 'A'
    if (student.Name.StartsWith("A"))
        return true;

    return false;
}

// Main code
List<Student> students = new List<Student>()
{
    new Student(1, "Ali", 22),      // Starts with A - Remove
    new Student(2, "Sara", 20),     // Keep
    new Student(3, "Omar", 35),     // Age > 30 - Remove
    new Student(4, "Ahmed", 25),    // Starts with A - Remove
    new Student(5, "Fatma", 19)     // Keep
};

Console.WriteLine("Original List:");
foreach (Student s in students)
{
    Console.WriteLine(s);
}

Console.WriteLine("\nRemoving based on custom conditions...");

for (int i = students.Count - 1; i >= 0; i--)
{
    if (ShouldRemoveStudent(students[i]))
    {
        students.RemoveAt(i);
    }
}
```

```
Console.WriteLine("\nAfter removal:");
foreach (Student s in students)
{
    Console.WriteLine(s);
}

// Output:
// 2-Sara-20 years old
// 5-Fatma-19 years old
```

Performance Comparison

Method Performance

```
Operation               | Time Complexity | Notes
-----------------------|-----------------|---------------------------
Remove(value)          | O(n)            | Searches then removes
RemoveAt(index)        | O(n)            | Must shift elements
RemoveRange(idx, cnt)  | O(n)            | Single shift operation
Clear()                | O(1)            | Fast - just resets count
```

Best Practices

1. **Loop Backwards** when removing during iteration
2. **Use RemoveAt()** when you know the index
3. **Use Remove()** for single value removal
4. **Use RemoveRange()** for contiguous elements
5. **Use Clear()** to remove everything

Complete Working Example in Main

```
static void Main(string[] args)
{
    // Example from your code - removing students
    List<Student> students = new List<Student>()
    {
        new Student(1, "Ali", 22),
        new Student(2, "Sara", 20),
        new Student(3, "Omar", 25),
        new Student(4, "Fatma", 19),
        new Student(5, "Ahmed", 23)
    };

    Console.WriteLine("=== Original Student List ===");
    foreach (Student s in students)
    {
        Console.WriteLine(s);
    }

    // Method 1: Remove specific student by Id (without reference)
    Console.WriteLine("\n=== Method 1: Remove by Id ===");
    int idToRemove = 3;

    for (int i = students.Count - 1; i >= 0; i--)
    {
        if (students[i].Id == idToRemove)
        {
            Console.WriteLine($"Removing: {students[i]}");
            students.RemoveAt(i);
            break;  // Remove only first match
        }
    }

    Console.WriteLine("\nAfter removal:");
    foreach (Student s in students)
    {
        Console.WriteLine(s);
    }

    // Method 2: Remove all students with specific age
    Console.WriteLine("\n=== Method 2: Remove by Age ===");
    int ageToRemove = 22;

    for (int i = students.Count - 1; i >= 0; i--)
    {
        if (students[i].Age == ageToRemove)
        {
            Console.WriteLine($"Removing: {students[i]}");
            students.RemoveAt(i);
        }
    }

    Console.WriteLine("\nAfter removal:");
    foreach (Student s in students)
    {
```

```
        Console.WriteLine(s);
    }

    // Method 3: Remove students in age range
    Console.WriteLine("\n=== Method 3: Remove Age Range (18-20) ===");

    for (int i = students.Count - 1; i >= 0; i--)
    {
        if (students[i].Age >= 18 && students[i].Age <= 20)
        {
            Console.WriteLine($"Removing: {students[i]}");
            students.RemoveAt(i);
        }
    }

    Console.WriteLine("\nFinal list:");
    foreach (Student s in students)
    {
        Console.WriteLine(s);
    }
}
```

By Abdullah Ali

Contact : +201012613453