1. SP Übung

20.10.2020

Strukturierte Programmierung Übung 1

Prescher Marco

1.) Spezifikation: (Angabe siehe Angabeblatt)

- Ist eine min./max. erlaubte Länge für die Namen/Liste vorhanden?
- Welches Format/Darstellung hat die Liste?
- Wird eine Verschlisselung für die Namen/Liste eingesetzt?
- Welche Zeichencodierung (UTF-8, ASCII, Latin, ...) haben die Namen?
- Ist die List vorsortiert?
  Wenn JA => auf/ absteigend sortiert?
- Gibt es eine maximal Dauer bei der Suche der Namen?
  Wenn JA => Was soll passieren bei überschreitung der max Dauer?
  ↳ Abbruch? → neu Suchen?
- Doppelte Namen vorhanden?
- Können Sonderzeichen / Nummern in Namen vorkommen? (z.B X Æ A-12 Musk)

2.) Multiplikation durch Addition

a.) x … Multiplikant

y … Multiplikator

p … Produktwert

$$\Rightarrow x \cdot y = p$$

Bsp.: $5 \cdot 6 = 30 = \overbrace{6+6+6+6+6}^{5\text{ mal}} = 30$

Die Multiplikation stellt eine wiederholte Addition dar

indem der Multiplikator x mal (Multiplikant)

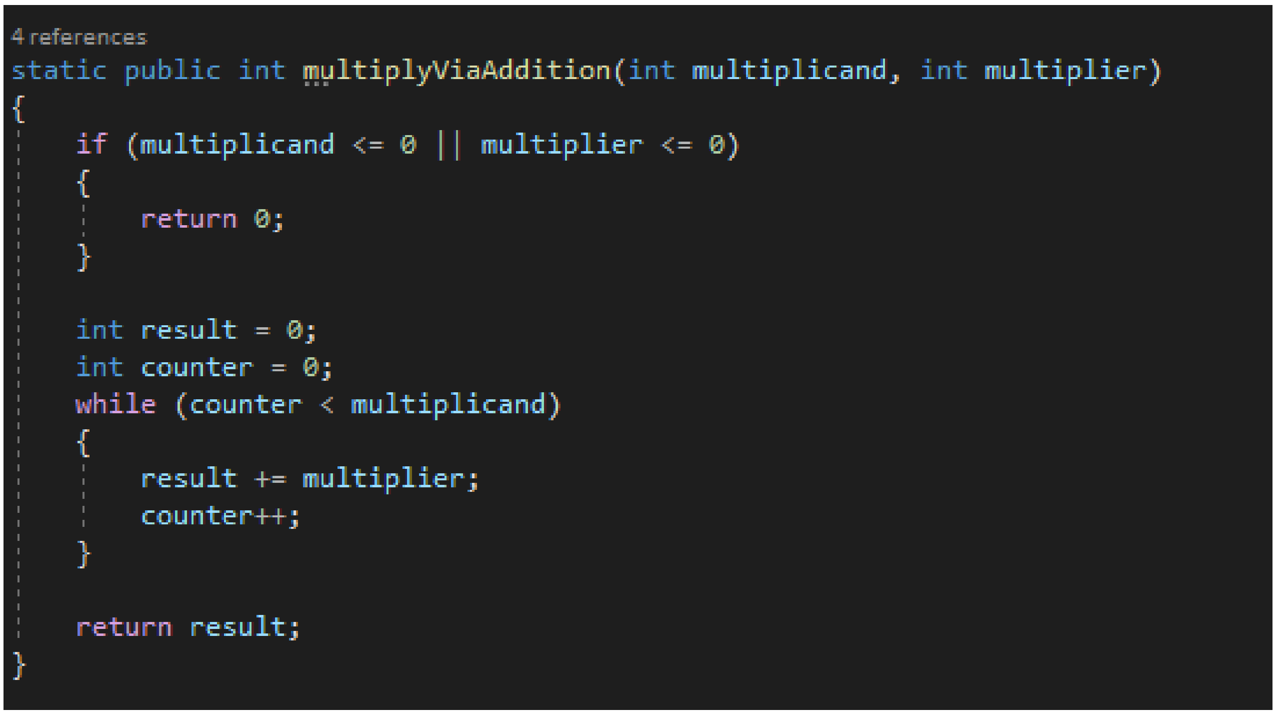
mit sich selbst Addiert wird.

b.) Algorithmus (Pseudocode) (Nur Natürliche Zahlen $\Rightarrow \mathbb{N} = \{0,1,2,3,\dots\}$)

```
MultipliziereMitAddition:
    integer multiplikator = 6
    integer multiplikant = 5
    integer ergebnis = 0
    integer zähler = 0
    WHILE (zähler < multiplikant) do
        ergebnis = ergebnis + multiplikator
        zähler = zähler + 1
    end WHILE
    ergebnis ausgeben / weitergeben
end MultipliziereMitAddition
```

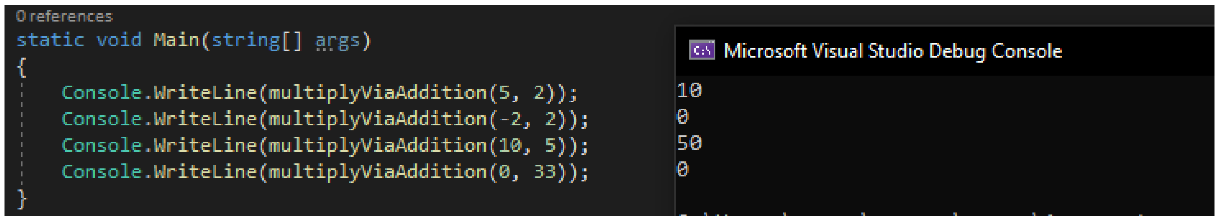
* Tests mit C# durchgeführt => siehe nächste Seite.

Mit C# eine “multiplyViaAddition” app geschrieben und dann 4 mal mit verschiedenen Werten getestet. (nur Natürliche Zahlen erlaubt)



```
4 references
static public int multiplyViaAddition(int multiplicand, int multiplier)
{
    if (multiplicand <= 0 || multiplier <= 0)
    {
        return 0;
    }

    int result = 0;
    int counter = 0;
    while (counter < multiplicand)
    {
        result += multiplier;
        counter++;
    }

    return result;
}
```

Ausgabe durch console:



```
0 references
static void Main(string[] args)
{
    Console.WriteLine(multiplyViaAddition(5, 2));
    Console.WriteLine(multiplyViaAddition(-2, 2));
    Console.WriteLine(multiplyViaAddition(10, 5));
    Console.WriteLine(multiplyViaAddition(0, 33));
}
```

Microsoft Visual Studio Debug Console

```
10
0
50
0
```

1. SP Übung

3.) Eine wird gewinnen!

Algorithmus: (Pseudo Code)

```
getWinner (n as a List, m as an integer):
        IF (size_of_List <= 0   or   m <= 0) then
            return -1
        else IF ( size_of_List == 1) then
            return List[0]

        integer removeIndex = 0
        WHILE (size_of_List > 1) do
            removeIndex += m - 1
            WHILE (removeIndex >= size_of_List)
                removeIndex -= size_of_List
            end WHILE
            List.RemoveAt(removeIndex)
        end WHILE
        return List[0]
end getWinner
```

removeIndex %= size_of_List ⇐ auch mit modulo Lösbar!

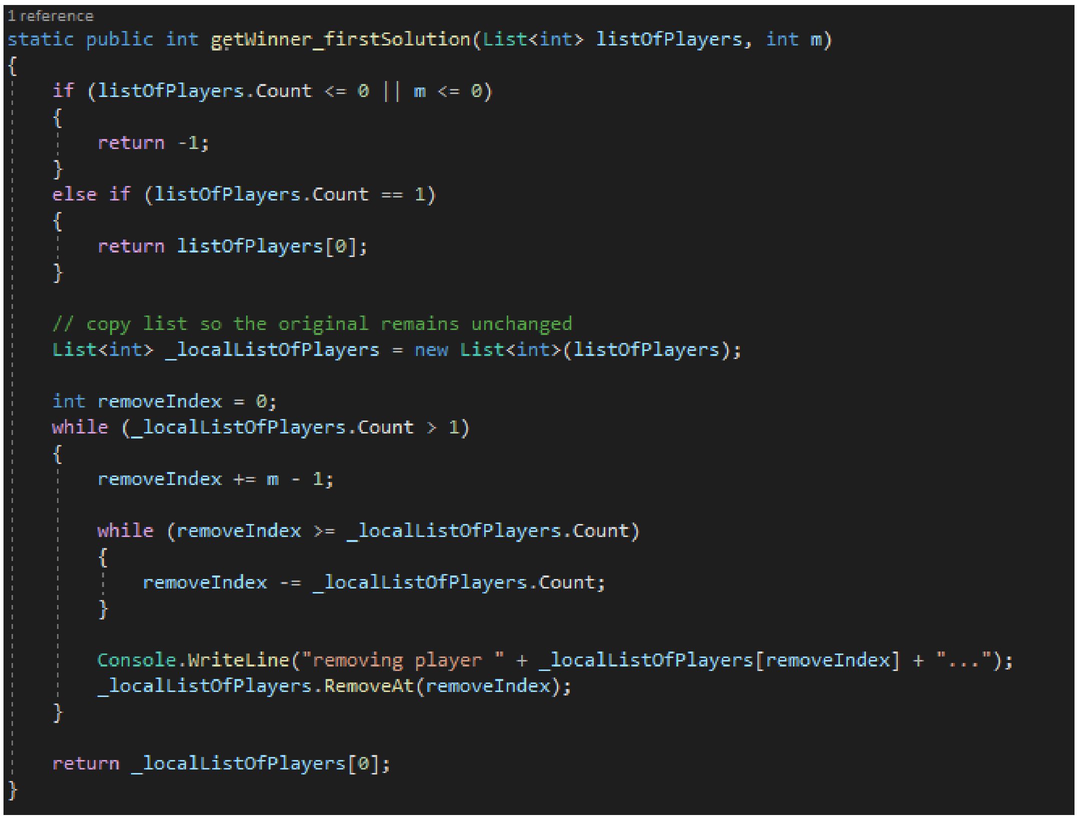
(dann braucht man die zweite WHILE schleife nicht)

* Tests mit C# durchgeführt => siehe nächste Seite.

(Test mit dem obigen realisierten Pseudo Code + einmal mit der modulo Lösung)

Mit C# eine “winner” app geschrieben und mit zwei verschiedenen Lösungen getestet. Habe „int Lists“ verwendet, da für die Spieler, in der Angabe, nur Ganzzahlen verwendet worden sind.

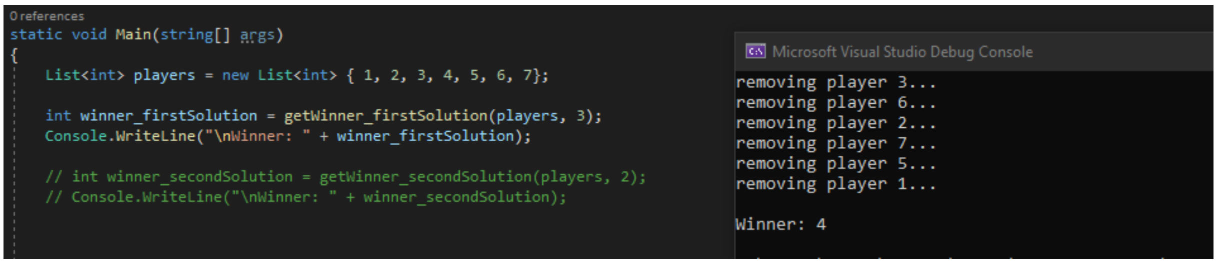
Erste Lösung:



```
1 reference
static public int getWinner_firstSolution(List<int> listOfPlayers, int m)
{
    if (listOfPlayers.Count <= 0 || m <= 0)
    {
        return -1;
    }
    else if (listOfPlayers.Count == 1)
    {
        return listOfPlayers[0];
    }

    // copy list so the original remains unchanged
    List<int> _localListOfPlayers = new List<int>(listOfPlayers);

    int removeIndex = 0;
    while (_localListOfPlayers.Count > 1)
    {
        removeIndex += m - 1;

        while (removeIndex >= _localListOfPlayers.Count)
        {
            removeIndex -= _localListOfPlayers.Count;
        }

        Console.WriteLine("removing player " + _localListOfPlayers[removeIndex] + "...");
        _localListOfPlayers.RemoveAt(removeIndex);
    }

    return _localListOfPlayers[0];
}
```

Ausgabe durch console:

1 Test (1 zu 1 wie in der angabe)

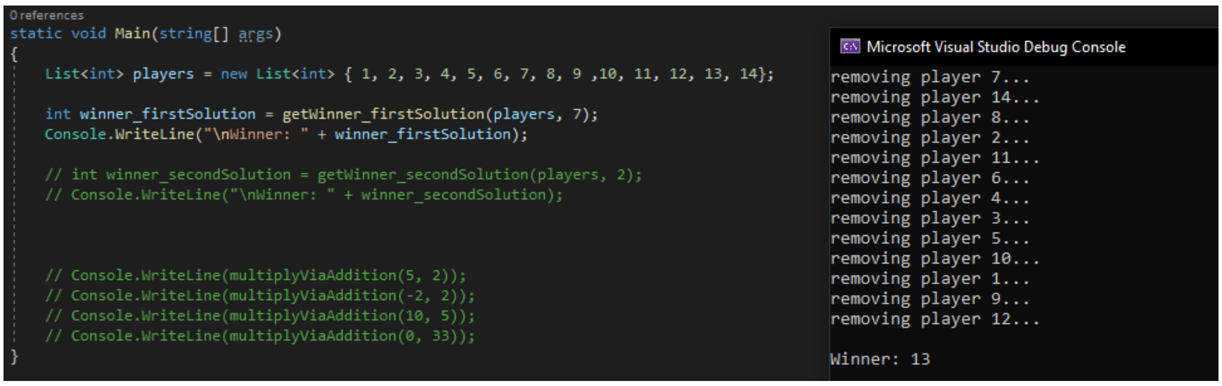


```
0 references
static void Main(string[] args)
{
    List<int> players = new List<int> { 1, 2, 3, 4, 5, 6, 7};

    int winner_firstSolution = getWinner_firstSolution(players, 3);
    Console.WriteLine("\nWinner: " + winner_firstSolution);

    // int winner_secondSolution = getWinner_secondSolution(players, 2);
    // Console.WriteLine("\nWinner: " + winner_secondSolution);
```

Microsoft Visual Studio Debug Console

```
removing player 3...
removing player 6...
removing player 2...
removing player 7...
removing player 5...
removing player 1...

Winner: 4
```

2 Test (andere ausscheidungs Zahl + mehr players in der Liste)

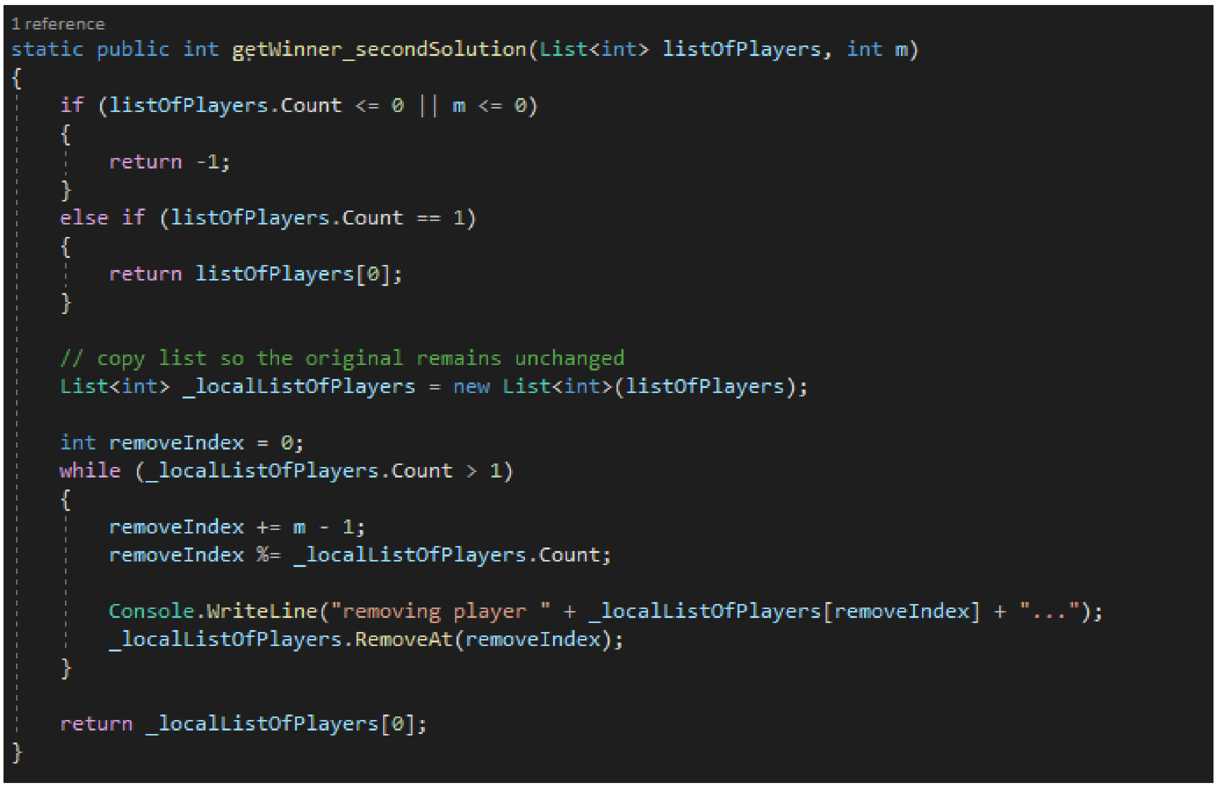


```
0 references
static void Main(string[] args)
{
    List<int> players = new List<int> { 1, 2, 3, 4, 5, 6, 7, 8, 9 ,10, 11, 12, 13, 14};

    int winner_firstSolution = getWinner_firstSolution(players, 7);
    Console.WriteLine("\nWinner: " + winner_firstSolution);

    // int winner_secondSolution = getWinner_secondSolution(players, 2);
    // Console.WriteLine("\nWinner: " + winner_secondSolution);



    // Console.WriteLine(multiplyViaAddition(5, 2));
    // Console.WriteLine(multiplyViaAddition(-2, 2));
    // Console.WriteLine(multiplyViaAddition(10, 5));
    // Console.WriteLine(multiplyViaAddition(0, 33));
}
```

Microsoft Visual Studio Debug Console

```
removing player 7...
removing player 14...
removing player 8...
removing player 2...
removing player 11...
removing player 6...
removing player 4...
removing player 3...
removing player 5...
removing player 10...
removing player 1...
removing player 9...
removing player 12...

Winner: 13
```

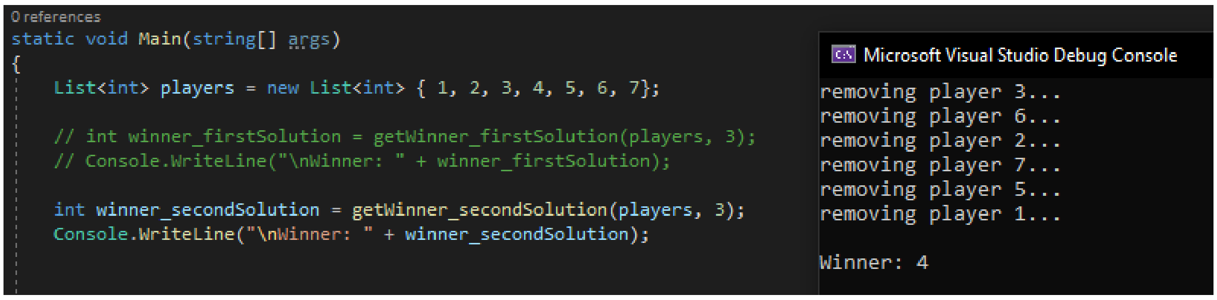
Zweite Lösung:



```
1 reference
static public int getWinner_secondSolution(List<int> listOfPlayers, int m)
{
    if (listOfPlayers.Count <= 0 || m <= 0)
    {
        return -1;
    }
    else if (listOfPlayers.Count == 1)
    {
        return listOfPlayers[0];
    }

    // copy list so the original remains unchanged
    List<int> _localListOfPlayers = new List<int>(listOfPlayers);

    int removeIndex = 0;
    while (_localListOfPlayers.Count > 1)
    {
        removeIndex += m - 1;
        removeIndex %= _localListOfPlayers.Count;

        Console.WriteLine("removing player " + _localListOfPlayers[removeIndex] + "...");
        _localListOfPlayers.RemoveAt(removeIndex);
    }

    return _localListOfPlayers[0];
}
```

Ausgabe durch console:

1 Test (1 zu 1 wie in der angabe)

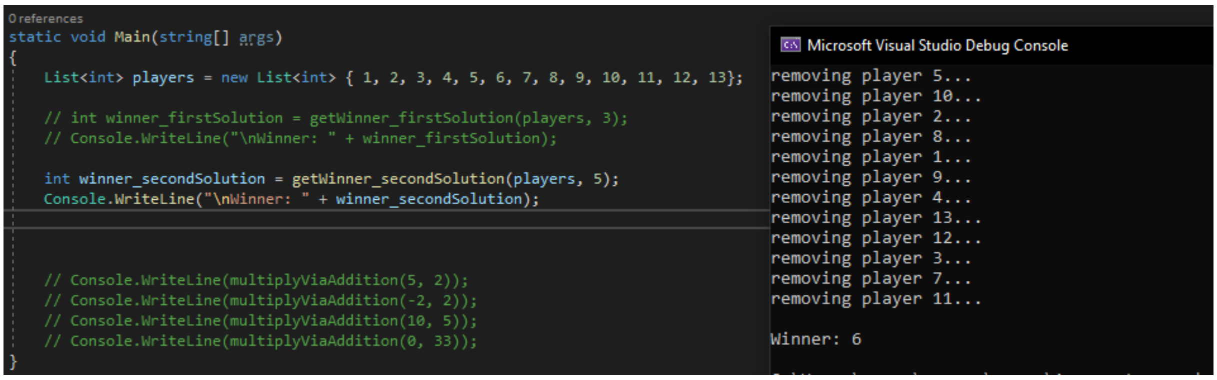


```
0 references
static void Main(string[] args)
{
   List<int> players = new List<int> { 1, 2, 3, 4, 5, 6, 7};

   // int winner_firstSolution = getWinner_firstSolution(players, 3);
   // Console.WriteLine("\nWinner: " + winner_firstSolution);

   int winner_secondSolution = getWinner_secondSolution(players, 3);
   Console.WriteLine("\nWinner: " + winner_secondSolution);
```

Microsoft Visual Studio Debug Console

```
removing player 3...
removing player 6...
removing player 2...
removing player 7...
removing player 5...
removing player 1...

Winner: 4
```

2 Test (andere ausscheidungs Zahl + mehr players in der Liste)



```
0 references
static void Main(string[] args)
{
   List<int> players = new List<int> { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13};

   // int winner_firstSolution = getWinner_firstSolution(players, 3);
   // Console.WriteLine("\nWinner: " + winner_firstSolution);

   int winner_secondSolution = getWinner_secondSolution(players, 5);
   Console.WriteLine("\nWinner: " + winner_secondSolution);



   // Console.WriteLine(multiplyViaAddition(5, 2));
   // Console.WriteLine(multiplyViaAddition(-2, 2));
   // Console.WriteLine(multiplyViaAddition(10, 5));
   // Console.WriteLine(multiplyViaAddition(0, 33));
}
```

Microsoft Visual Studio Debug Console

```
removing player 5...
removing player 10...
removing player 2...
removing player 8...
removing player 1...
removing player 9...
removing player 4...
removing player 13...
removing player 12...
removing player 3...
removing player 7...
removing player 11...

Winner: 6
```