File Input and Output (I/O)

Engineering 1D04, Teaching
Session 7

File I/O

- For anything in this session to work correctly, you need to include the File I/O Library.
- This can be done by adding the following line at the top of your C# program with the rest of the using statements.

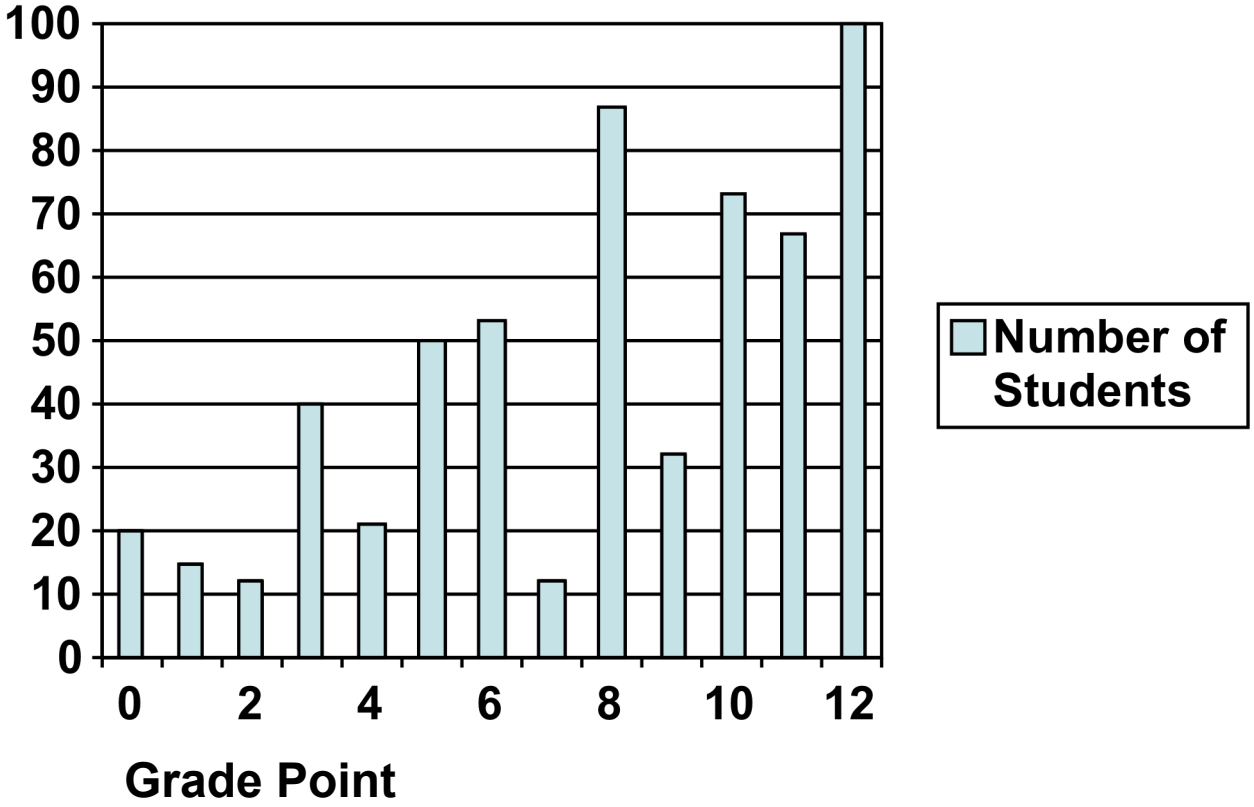
```
using System.IO;
```

File I/O

- Problem: Find the distribution of grade points from 1D04 last year, i.e. How many people received each grade point?

File I/O

- With this data, we should be able to produce a chart like this:



File I/O

- There were 800 people in 1D04 last year. This is a lot of data to enter manually - if we have the grades already stored in a computer readable form.
- When large amounts of data need to be used in a computer program it is essential to be able to import this data from a file.

File I/O

- In this example, a sample file has been prepared called *grades.yyz*.
- Each line has a number from 0-12 corresponding to the McMaster Grade Scale.
- Student names are not included.

File I/O

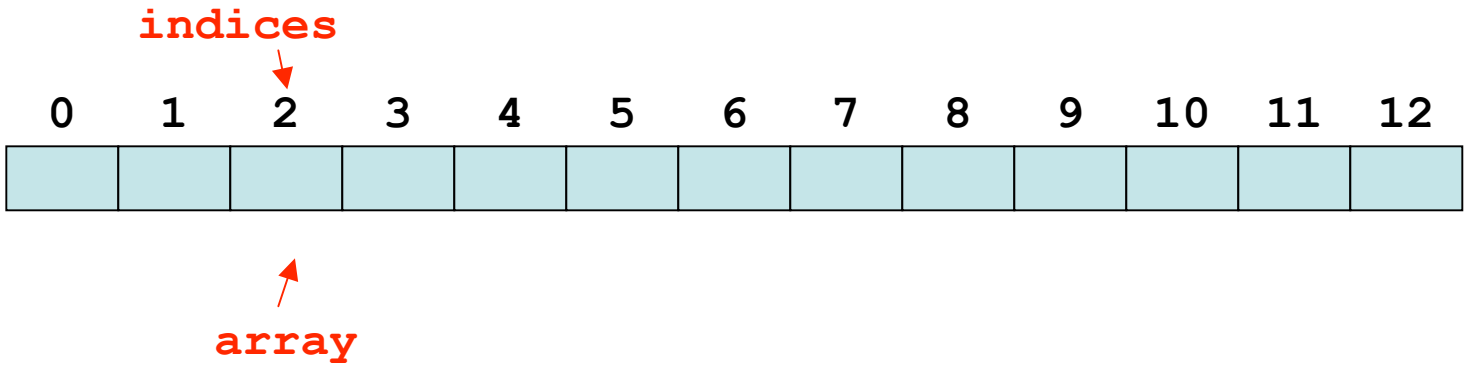
- Extract from the grade file

File I/O

- Develop an algorithm to solve this problem?

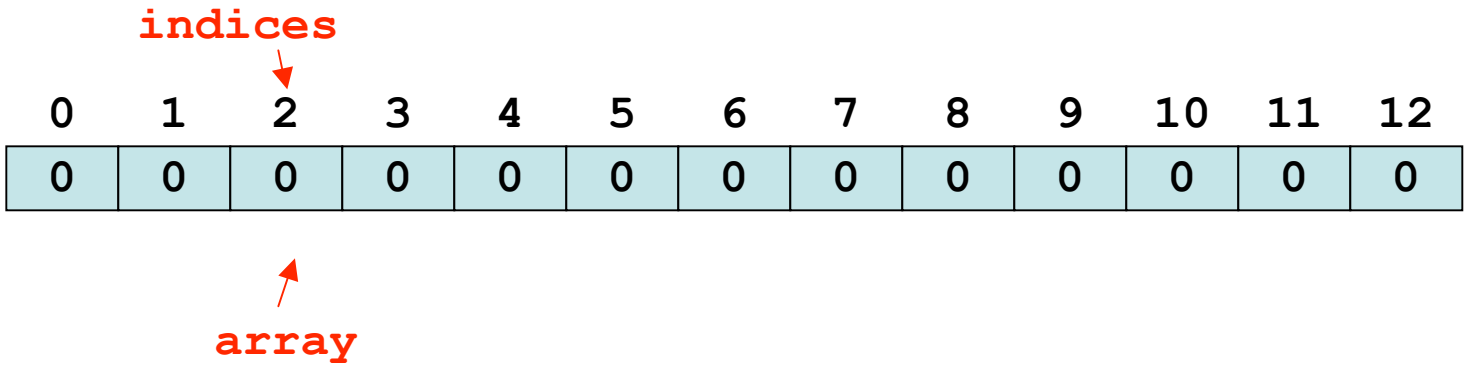
File I/O - Solution

- We can use an array, with each index in the array representing a grade point.



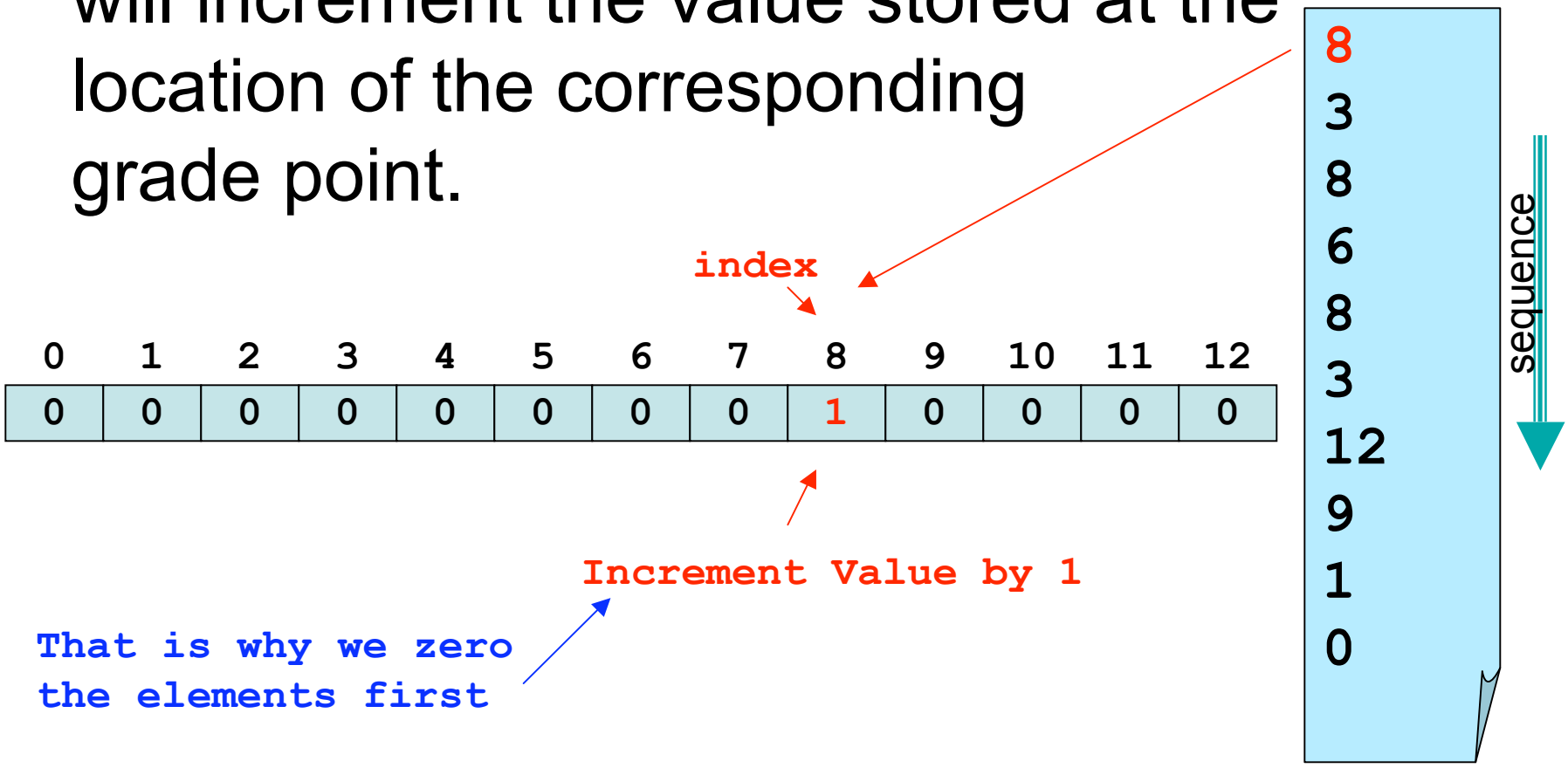
File I/O - Solution

- Each location will represent how many people scored that grade.
- We need to zero each location as we have not read in any grades yet. **Why?**



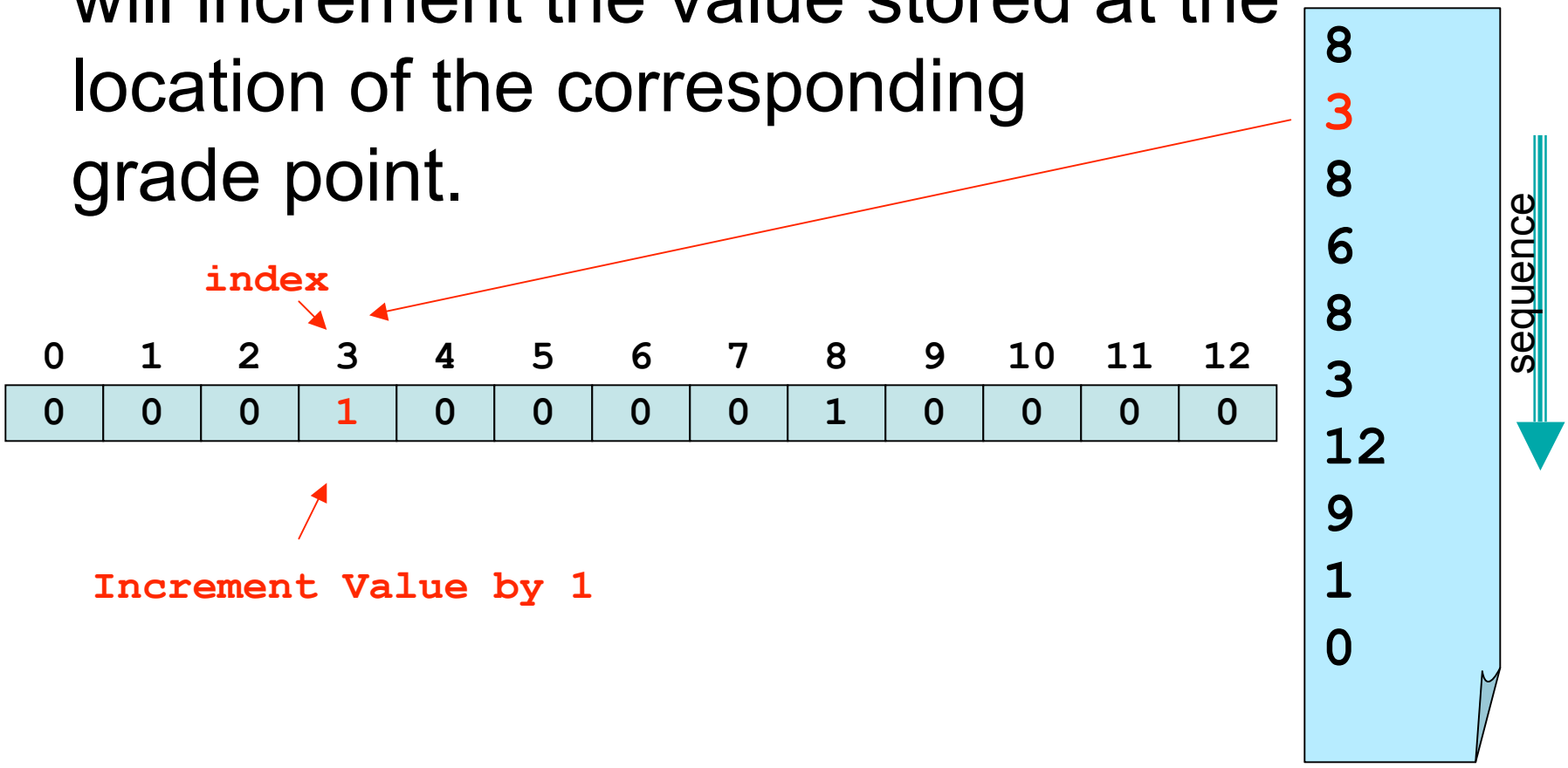
File I/O - Solution

- As we read through the file of grades, we will increment the value stored at the location of the corresponding grade point.



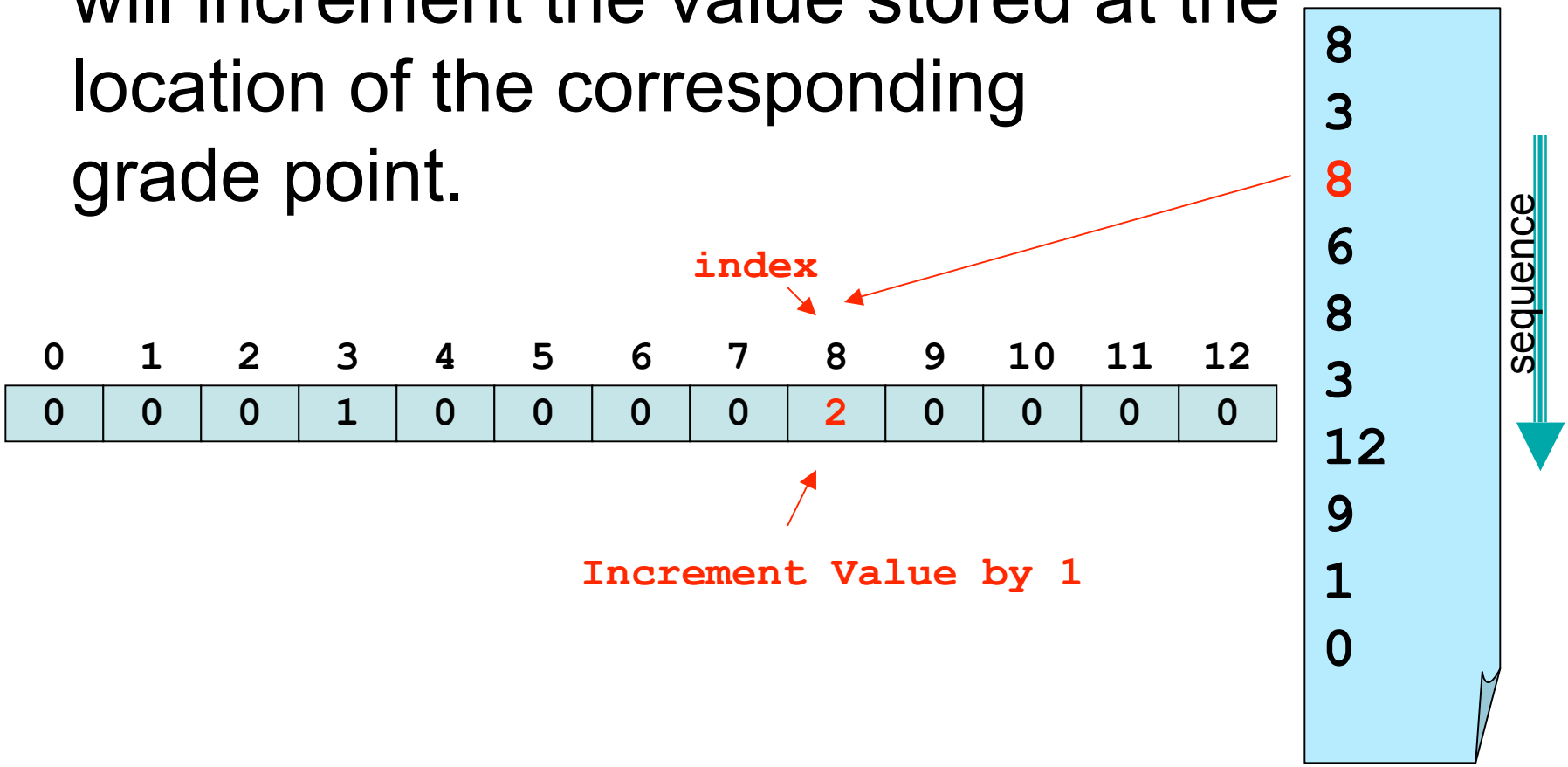
File I/O - Solution

- As we read through the file of grades, we will increment the value stored at the location of the corresponding grade point.



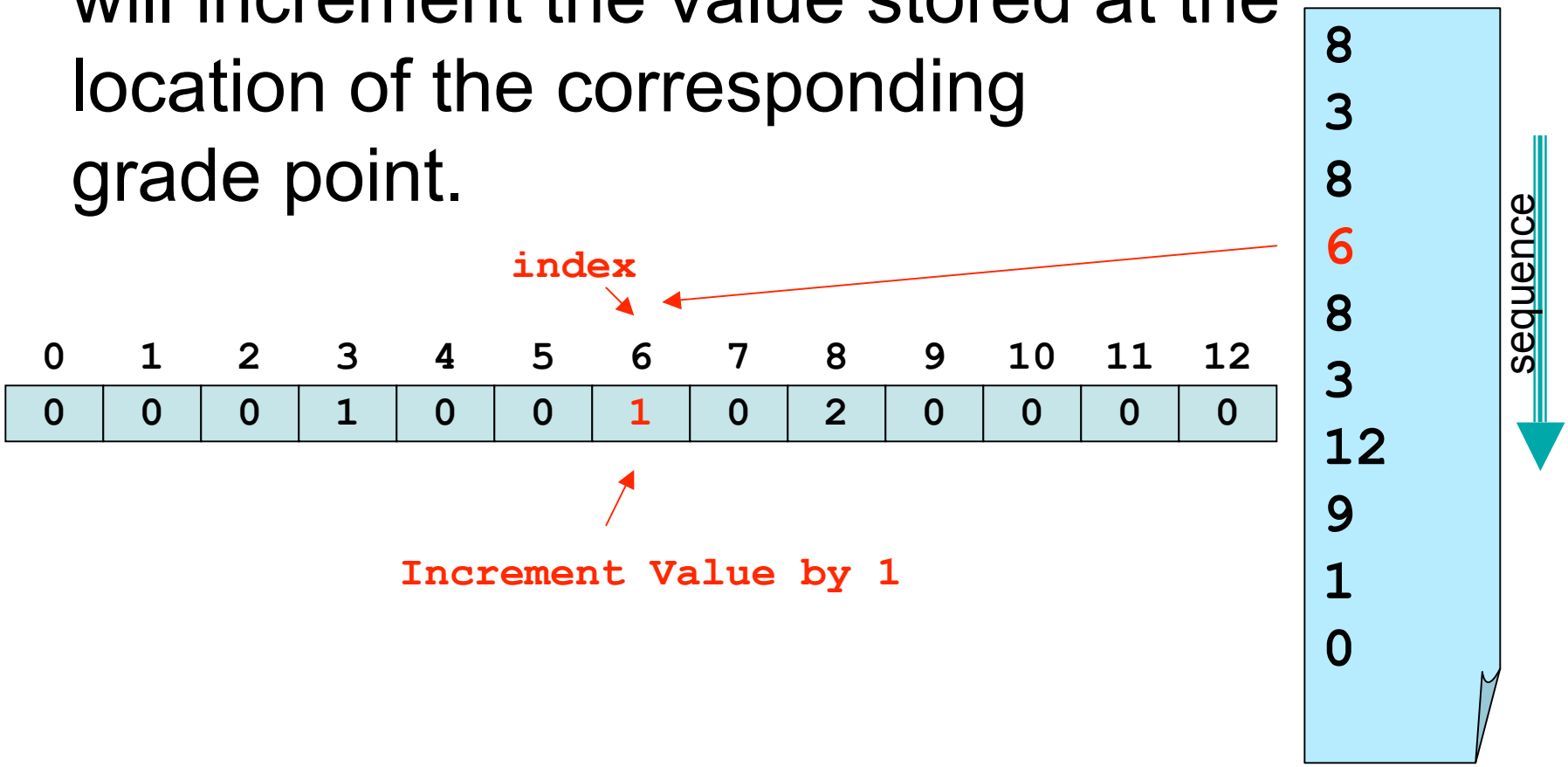
File I/O - Solution

- As we read through the file of grades, we will increment the value stored at the location of the corresponding grade point.



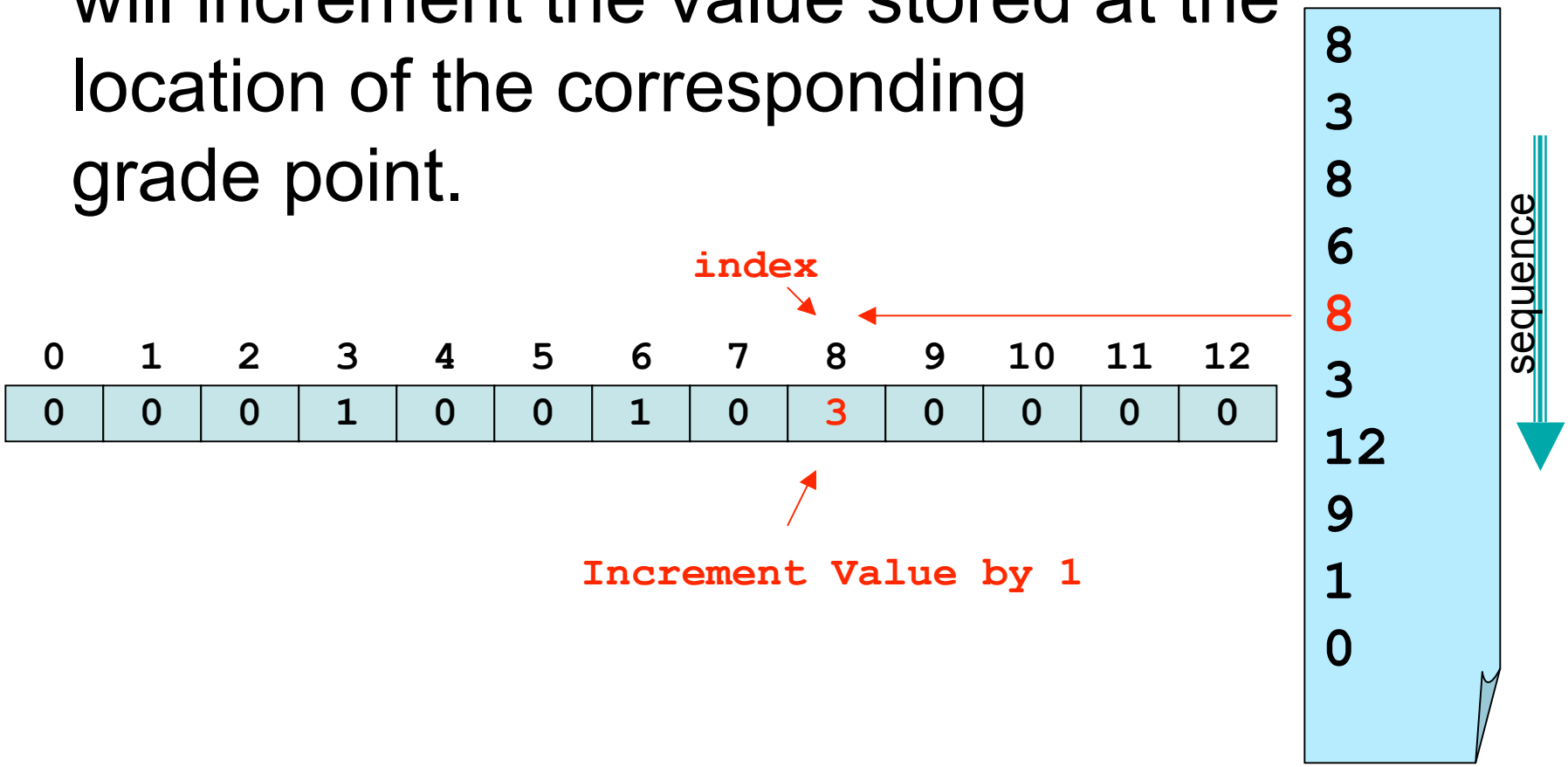
File I/O - Solution

- As we read through the file of grades, we will increment the value stored at the location of the corresponding grade point.



File I/O - Solution

- As we read through the file of grades, we will increment the value stored at the location of the corresponding grade point.



File I/O - Algorithm

- Create and zero an array of 13 elements for grade points.
- Input one grade from the grade file.
- Increment the respective location in the grade point array.
- Repeat until all grades have been used.

File I/O - Declarations

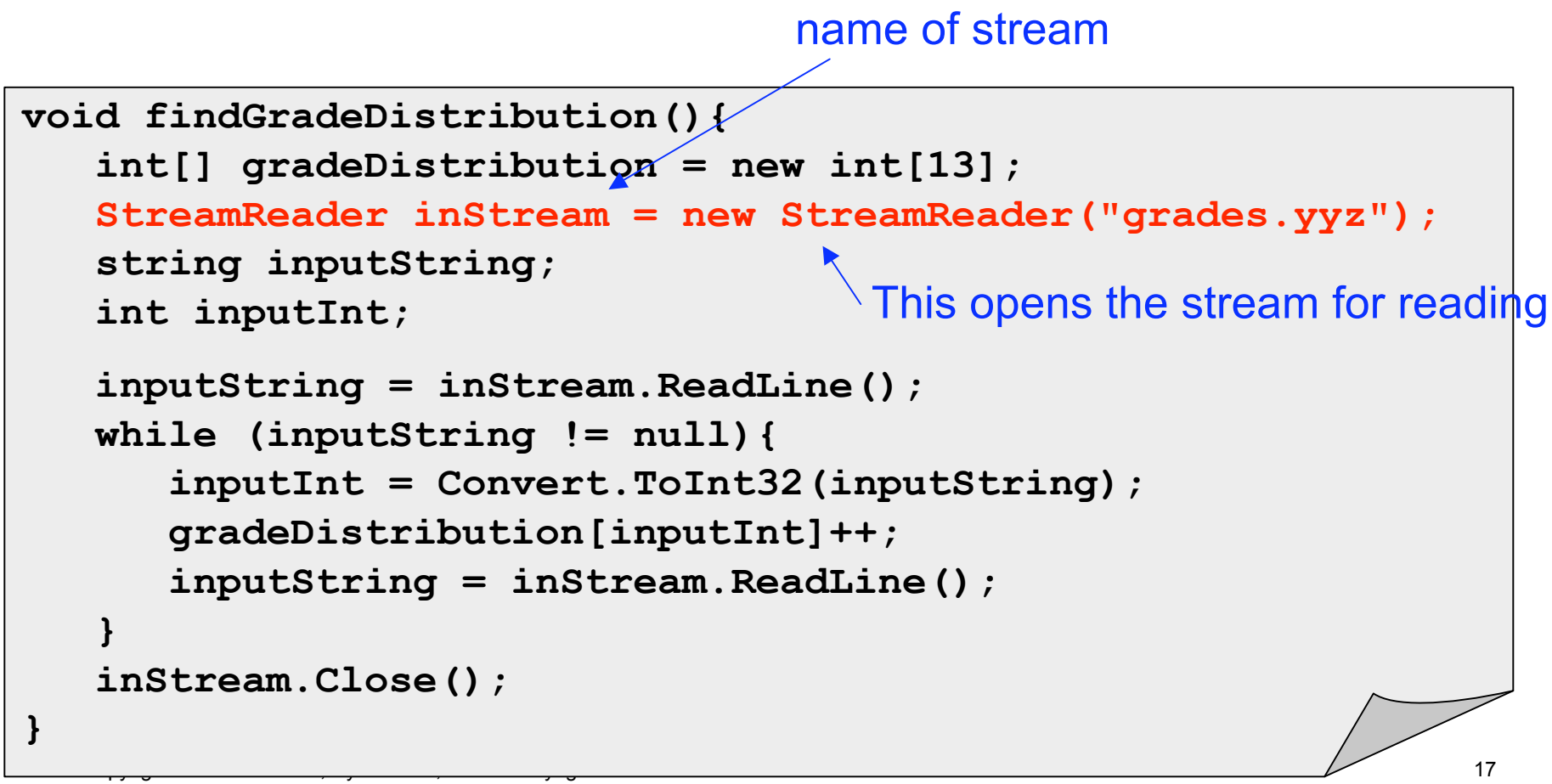
- Streams are used to work with files.
- A stream is simply a name for a flow of data.

```
void findGradeDistribution(){
   int[] gradeDistribution = new int[13];
   StreamReader inStream = new StreamReader("grades.yyz");
   string inputString;
   int inputInt;

   inputString = inStream.ReadLine();
   while (inputString != null){
      inputInt = Convert.ToInt32(inputString);
      gradeDistribution[inputInt]++;
      inputString = inStream.ReadLine();
   }
   inStream.Close();
}
```

File I/O - Declarations

- Streams are used to work with files.
- A stream is simply a name for a flow of data.




```
void findGradeDistribution(){
   int[] gradeDistribution = new int[13];
   StreamReader inStream = new StreamReader("grades.yyz");
   string inputString;
   int inputInt;

   inputString = inStream.ReadLine();
   while (inputString != null){
      inputInt = Convert.ToInt32(inputString);
      gradeDistribution[inputInt]++;
      inputString = inStream.ReadLine();
   }
   inStream.Close();
}
```

File I/O - Declarations

- The filename associated with the stream is a string passed as a parameter inside ()'s when creating a new stream.
- To make it easier right now, the file needs to be in the working directory for your program.

```
void findGradeDistribution(){
   int[] gradeDistribution = new int[13];
   StreamReader inStream = new StreamReader("grades.yyz");
   string inputString;
   int inputInt;

   inputString = inStream.ReadLine();
   while (inputString != null){
      inputInt = Convert.ToInt32(inputString);
      gradeDistribution[inputInt]++;
      inputString = inStream.ReadLine();
   }
   inStream.Close();
}
```

File I/O - Declarations

- In Visual Studio, the working directory for an application is:
  - projectFolder\ProjectName\bin\Debug

  - Example:
  - C:\Documents and Settings\username\
    My Documents\Visual Studio 2005\Projects\
    gradeThingy\gradeThingy\bin\Debug

File I/O

- Create and zero an array of 13 elements for grade points.
- Input one grade from the grade file.
- Increment the respective location in the grade point array.
- Repeat until there are no more grades in the file.

```
void findGradeDistribution(){
    int[] gradeDistribution = new int[13];
    StreamReader inStream = new StreamReader("grades.yyz");
    string inputString;
    int inputInt;

    inputString = inStream.ReadLine();
    while (inputString != null){
        inputInt = Convert.ToInt32(inputString);
        gradeDistribution[inputInt]++;
        inputString = inStream.ReadLine();
    }
    inStream.Close();
}
```

File I/O

- Create and zero an array of 13 elements for grade points.
- Input one grade from the grade file.
- Increment the respective location in the grade point array.
- Repeat until there are no more grades in the file.

```
void findGradeDistribution(){
    int[] gradeDistribution = new int[13];
    StreamReader inStream = new StreamReader("grades.yyz");
    string inputString;
    int inputInt;

    inputString = inStream.ReadLine();
    while (inputString != null){
        inputInt = Convert.ToInt32(inputString);
        gradeDistribution[inputInt]++;
        inputString = inStream.ReadLine();
    }
    inStream.Close();
}
```

why?

File I/O

- Create and zero an array of 13 elements for grade points.
- Input one grade from the grade file.
- Increment the respective location in the grade point array.
- Repeat until there are no more grades in the file.

assumption (at least 1)

```
void findGradeDistribution(){
   int[] gradeDistribution = new int[13];
   StreamReader inStream = new StreamReader("grades.yyz");
   string inputString;
   int inputInt;

   inputString = inStream.ReadLine();
   while (inputString != null){
      inputInt = Convert.ToInt32(inputString);
      gradeDistribution[inputInt]++;
      inputString = inStream.ReadLine();
   }
   inStream.Close();
}
```

why?

File I/O

- Create and zero an array of 13 elements for grade points.
- Input one grade from the grade file.
- Increment the respective location in the grade point array.
- Repeat until there are no more grades in the file.

```
void findGradeDistribution(){
   int[] gradeDistribution = new int[13];
   StreamReader inStream = new StreamReader("grades.yyz");
   string inputString;
   int inputInt;

   inputString = inStream.ReadLine();
   while (inputString != null){
      inputInt = Convert.ToInt32(inputString);
      gradeDistribution[inputInt]++;
      inputString = inStream.ReadLine();
   }
   inStream.Close();
}
```

File I/O

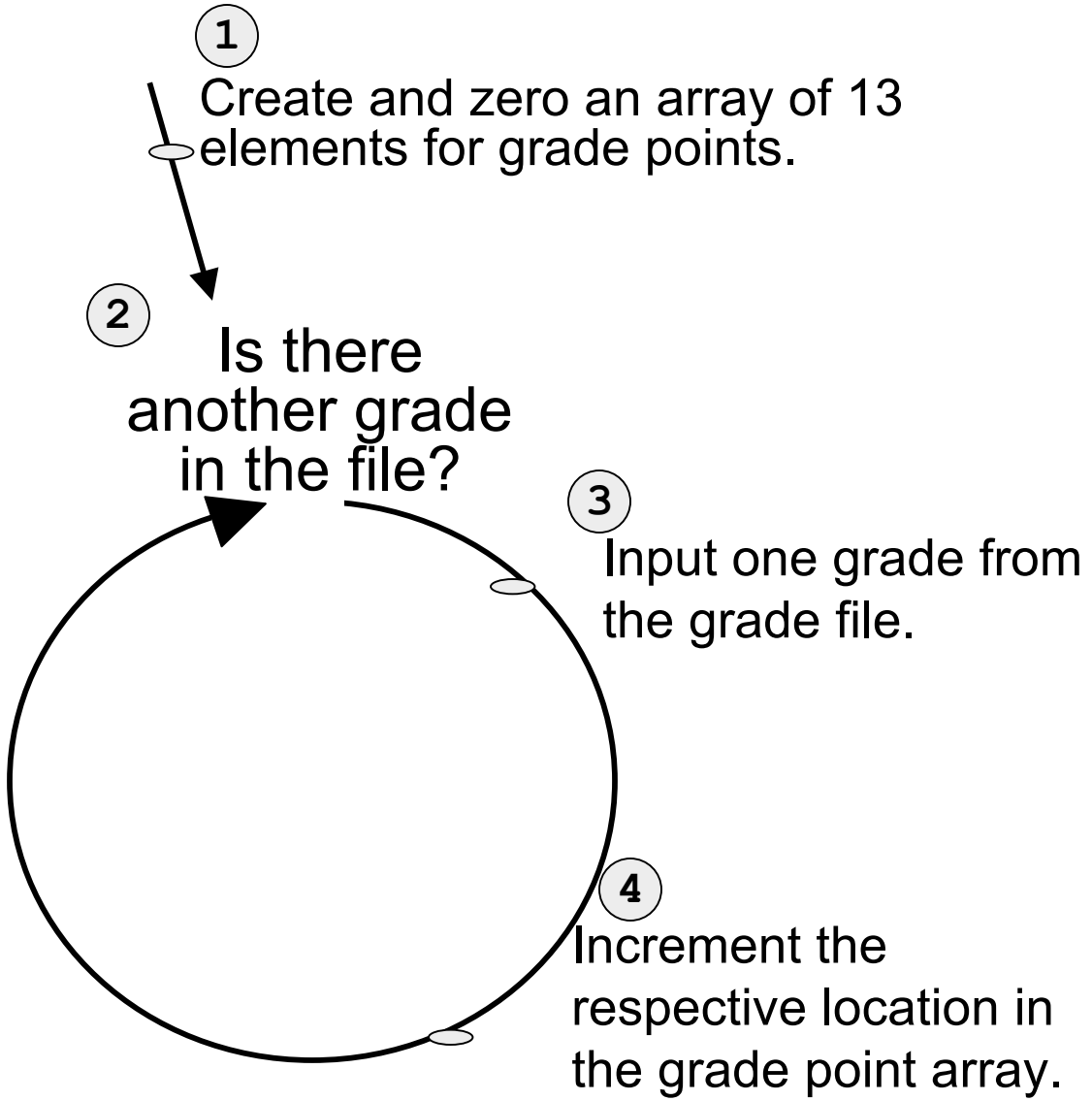
- Create and zero an array of 13 elements for grade points.
- Input one grade from the grade file.
- Increment the respective location in the grade point array.
- Repeat until there are no more grades in the file.

```
void findGradeDistribution(){
    int[] gradeDistribution = new int[13];
    StreamReader inStream = new StreamReader("grades.yyz");
    string inputString;
    int inputInt;

    inputString = inStream.ReadLine();
    while (inputString != null){
        inputInt = Convert.ToInt32(inputString);
        gradeDistribution[inputInt]++;
        inputString = inStream.ReadLine();
    }
    inStream.Close();
}
```

Loop Visualization

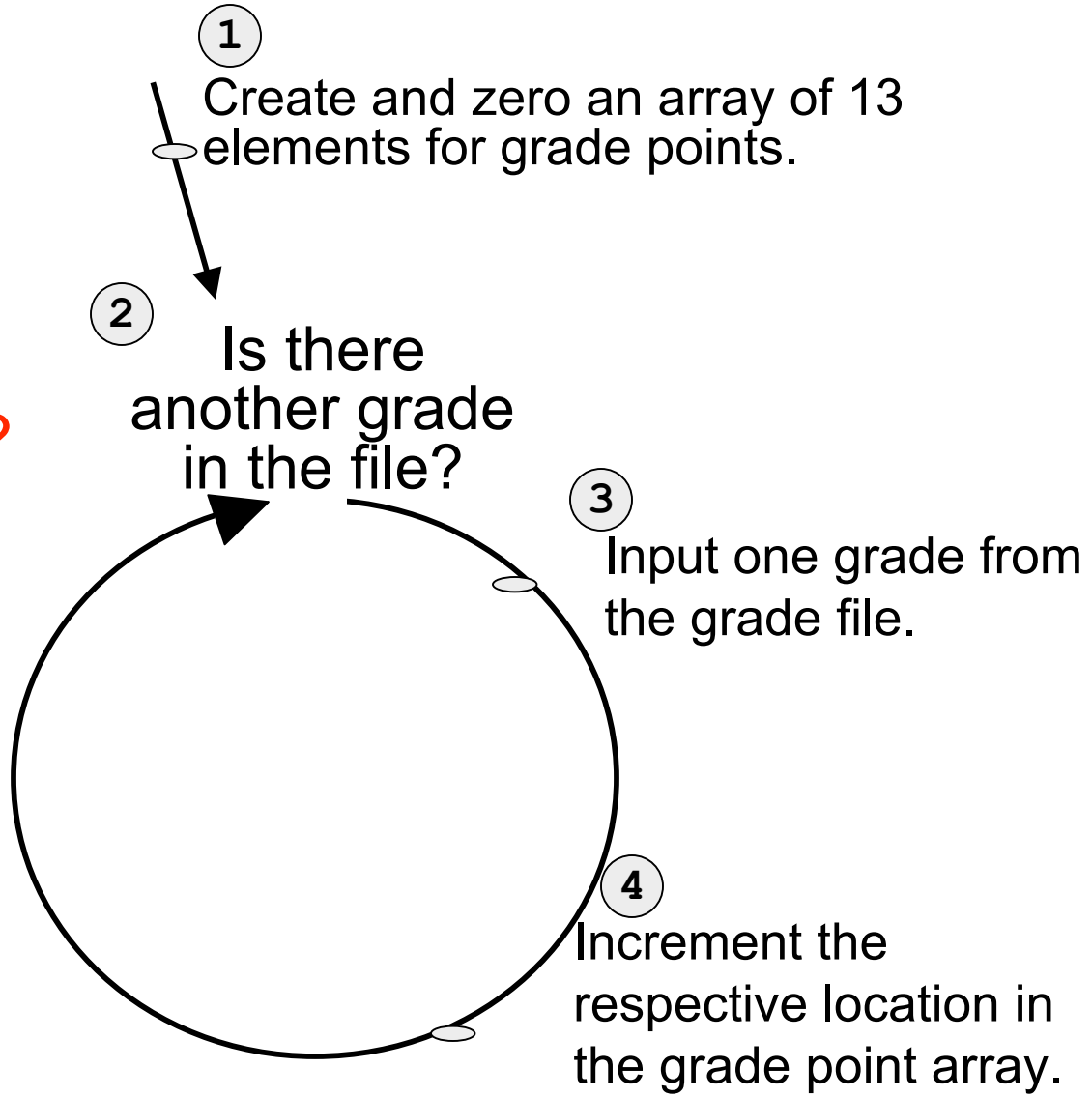
Why doesn't our C# loop look like this?



Loop Visualization

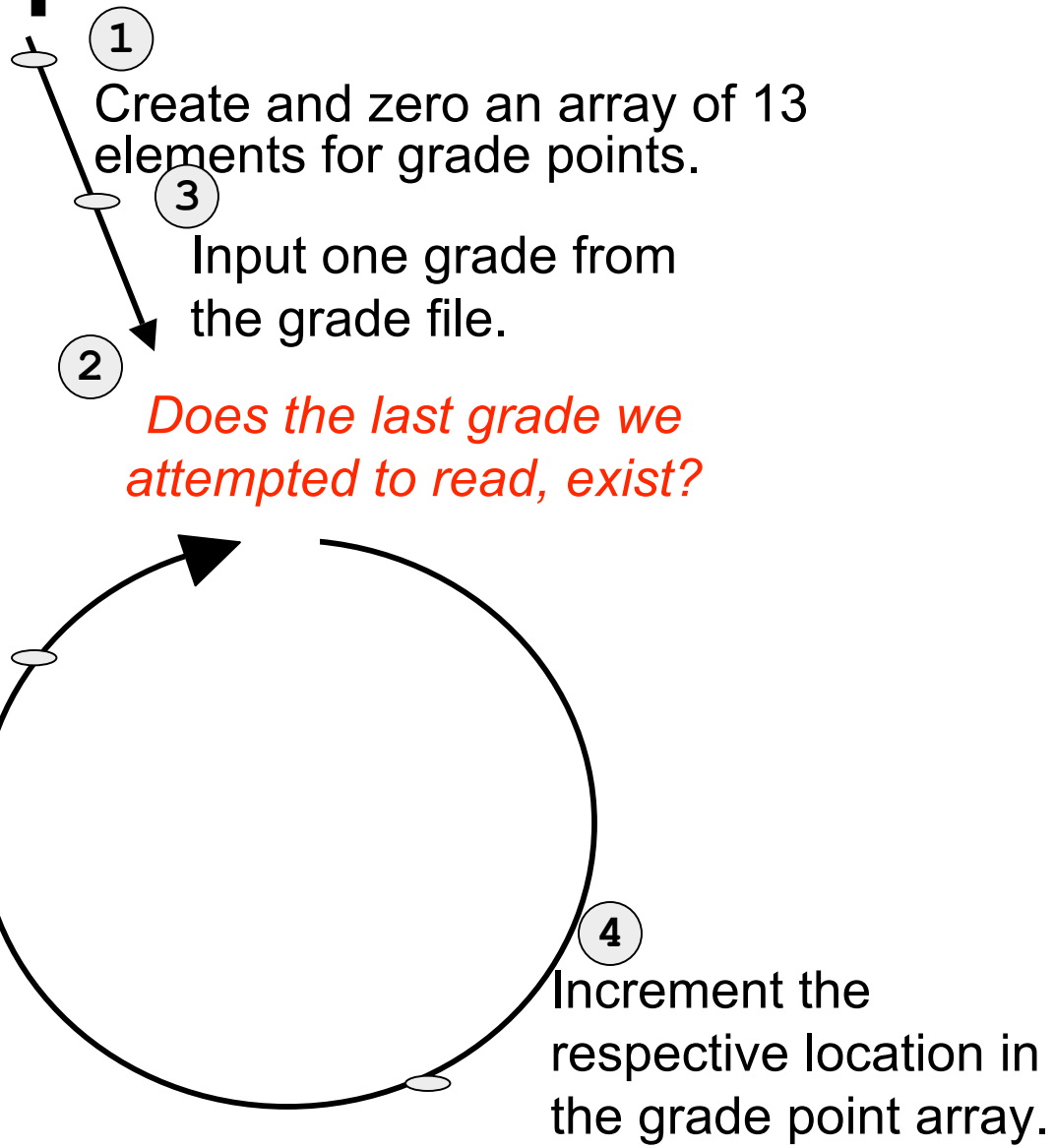
In C#, we can't ask the question *Is there another grade in the file?*

Instead, we can ask the question *Does the last grade we attempted to read, exist?*



Loop Visualization

We have to adjust the loop slightly to accommodate this new question.



File I/O

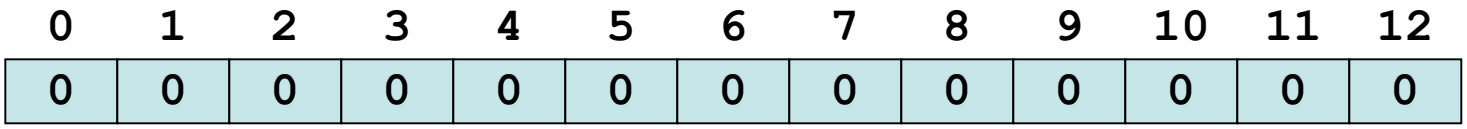
- inStream.ReadLine() is where all the magic happens.
- inStream.ReadLine() returns a string value of the next unread line unless there are no more unread lines in which case it returns null.

```
void findGradeDistribution(){
   int[] gradeDistribution = new int[13];
   StreamReader inStream = new StreamReader("grades.yyz");
   string inputString;
   int inputInt;

   inputString = inStream.ReadLine();
   while (inputString != null){
      inputInt = Convert.ToInt32(inputString);
      gradeDistribution[inputInt]++;
      inputString = inStream.ReadLine();
   }
   inStream.Close();
}
```

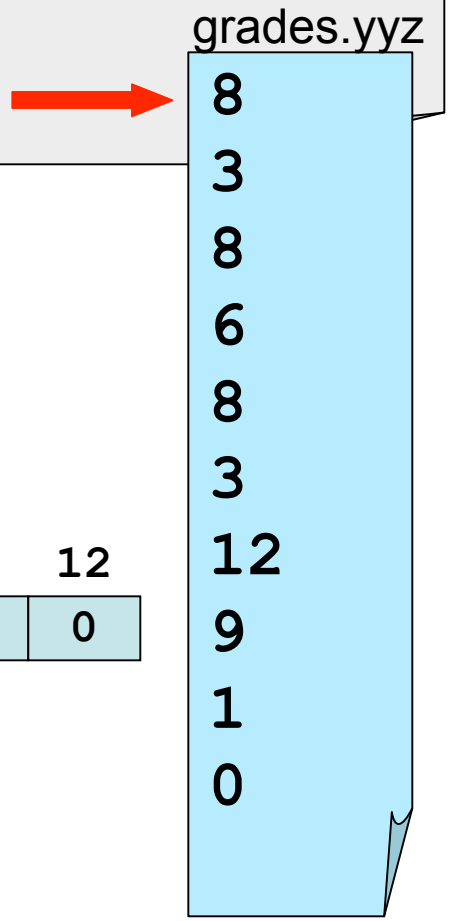
File I/O - Animation

```
. . .
   int[] gradeDistribution = new int[13];
   StreamReader inStream = new StreamReader("grades.yyz");
   string inputString;
   int inputInt;
. . .
```



File I/O - Animation

```
. . .
   int[] gradeDistribution = new int[13];
   StreamReader inStream = new StreamReader("grades.yyz");
   string inputString;
   int inputInt;
. . .
```



0	1	2	3	4	5	6	7	8	9	10	11	12
0	0	0	0	0	0	0	0	0	0	0	0	0

File I/O - Animation

```
. . .
   int[] gradeDistribution = new int[13];
   StreamReader inStream = new StreamReader("grades.yyz");
   string inputString;
   int inputInt;
. . .
```

grades.yyz

8
3
8
6
8
3
12
9
1
0

inputString

0	1	2	3	4	5	6	7	8	9	10	11	12
0	0	0	0	0	0	0	0	0	0	0	0	0

File I/O - Animation

```
. . .
   int[] gradeDistribution = new int[13];
   StreamReader inStream = new StreamReader("grades.yyz");
   string inputString;
   int inputInt;
. . .
```

grades.yyz

8
3
8
6
8
3
12
9
1
0

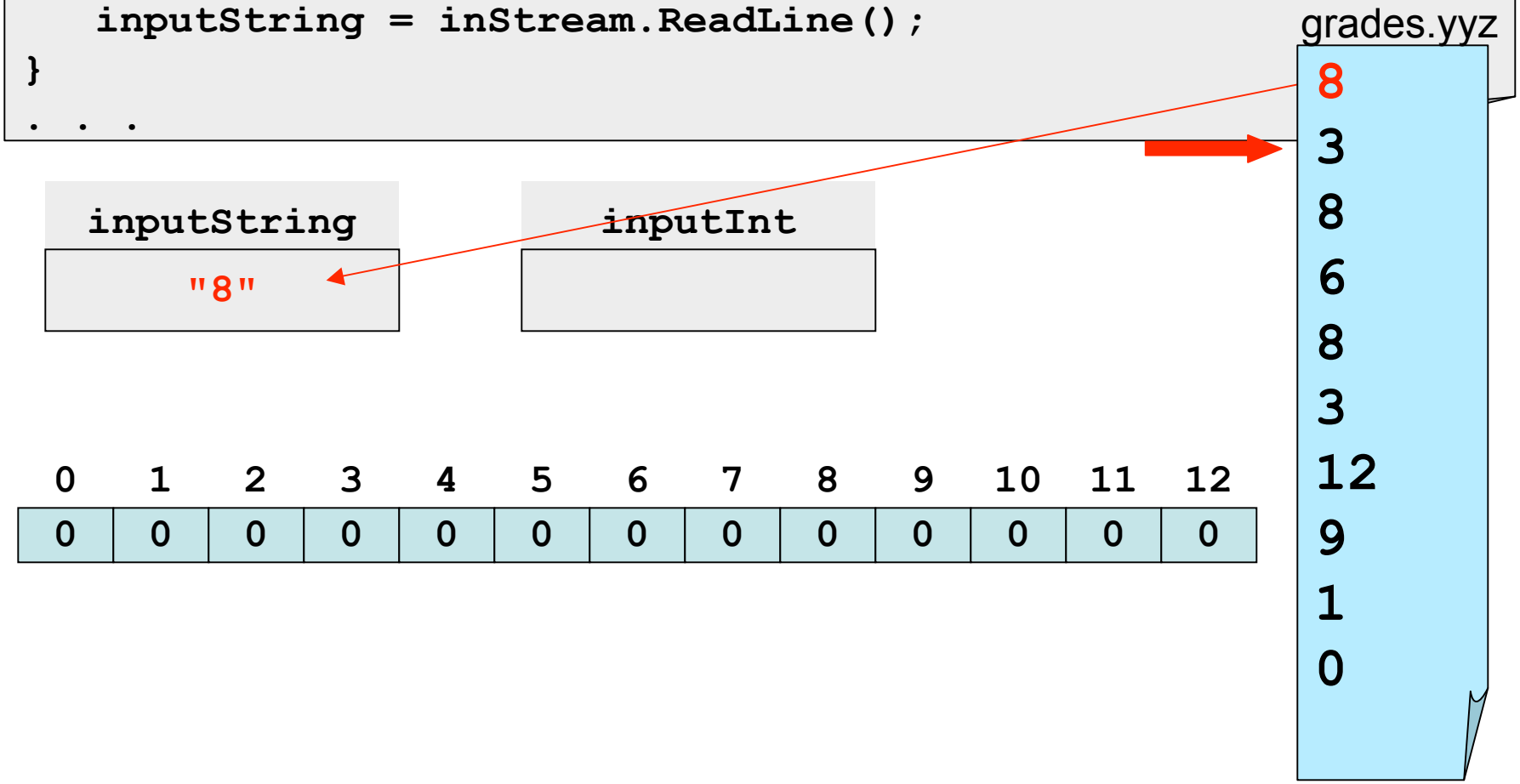
inputString

inputInt

0	1	2	3	4	5	6	7	8	9	10	11	12
0	0	0	0	0	0	0	0	0	0	0	0	0

File I/O - Animation

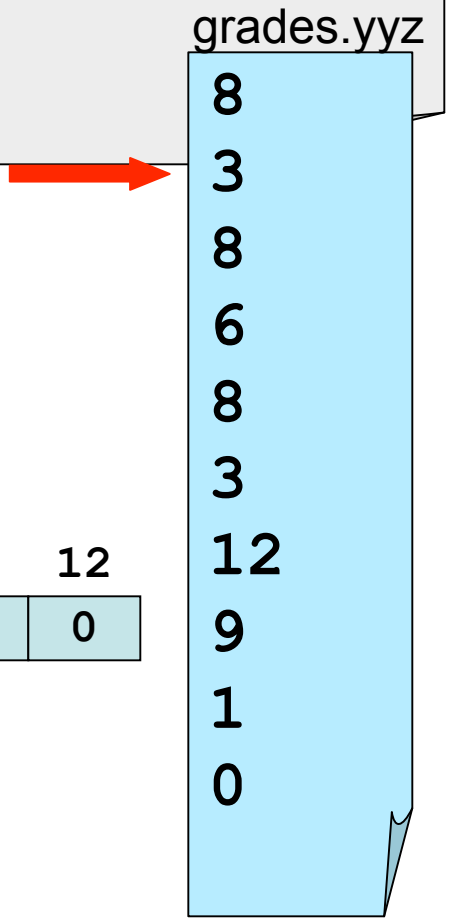
```
inputString = inStream.ReadLine();
while (inputString != null){
   inputInt = Convert.ToInt32(inputString);
   gradeDistribution[inputInt]++;
   inputString = inStream.ReadLine();
}
. . .
```



File I/O - Animation

```
inputString = inStream.ReadLine();
while (inputString != null){
   inputInt = Convert.ToInt32(inputString);
   gradeDistribution[inputInt]++;
   inputString = inStream.ReadLine();
}
. . .
```

True

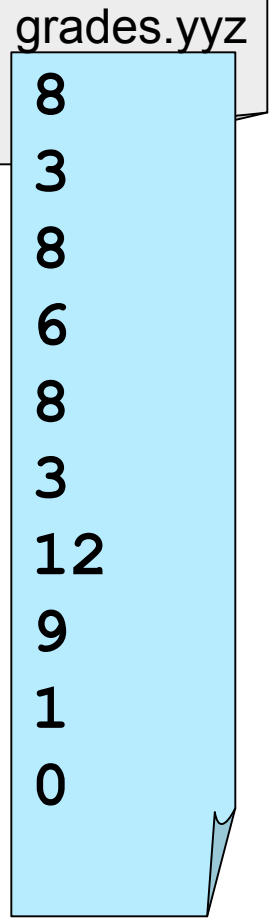


inputString	inputInt
"8"	

0	1	2	3	4	5	6	7	8	9	10	11	12
0	0	0	0	0	0	0	0	0	0	0	0	0

File I/O - Animation

```
inputString = inStream.ReadLine();
while (inputString != null){
   inputInt = Convert.ToInt32(inputString);
   gradeDistribution[inputInt]++;
   inputString = inStream.ReadLine();
}
. . .
```



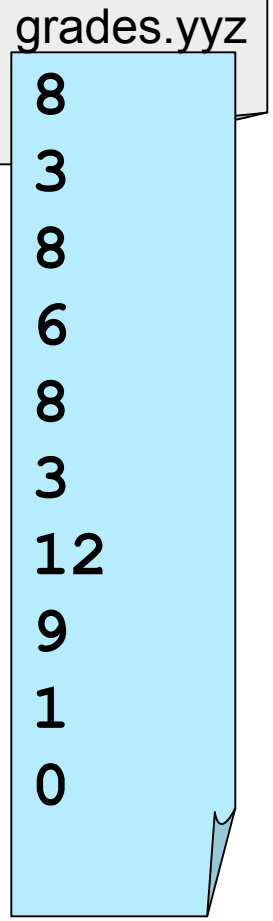
inputString
"8"

inputInt
8

0	1	2	3	4	5	6	7	8	9	10	11	12
0	0	0	0	0	0	0	0	0	0	0	0	0

File I/O - Animation

```
inputString = inStream.ReadLine();
while (inputString != null){
   inputInt = Convert.ToInt32(inputString);
   gradeDistribution[inputInt]++;
   inputString = inStream.ReadLine();
}
. . .
```



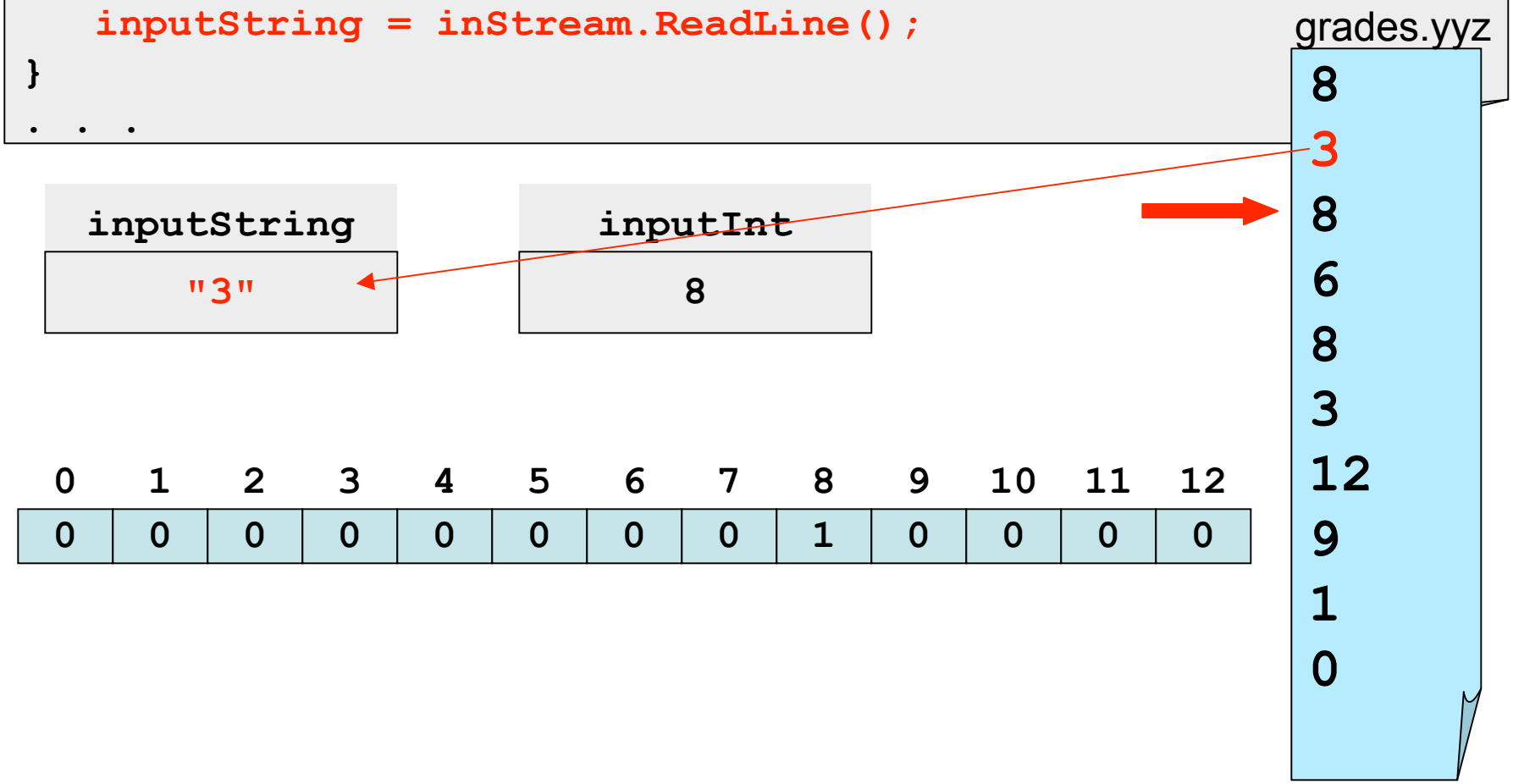
inputString
"8"

inputInt
8

0	1	2	3	4	5	6	7	8	9	10	11	12
0	0	0	0	0	0	0	0	1	0	0	0	0

File I/O - Animation

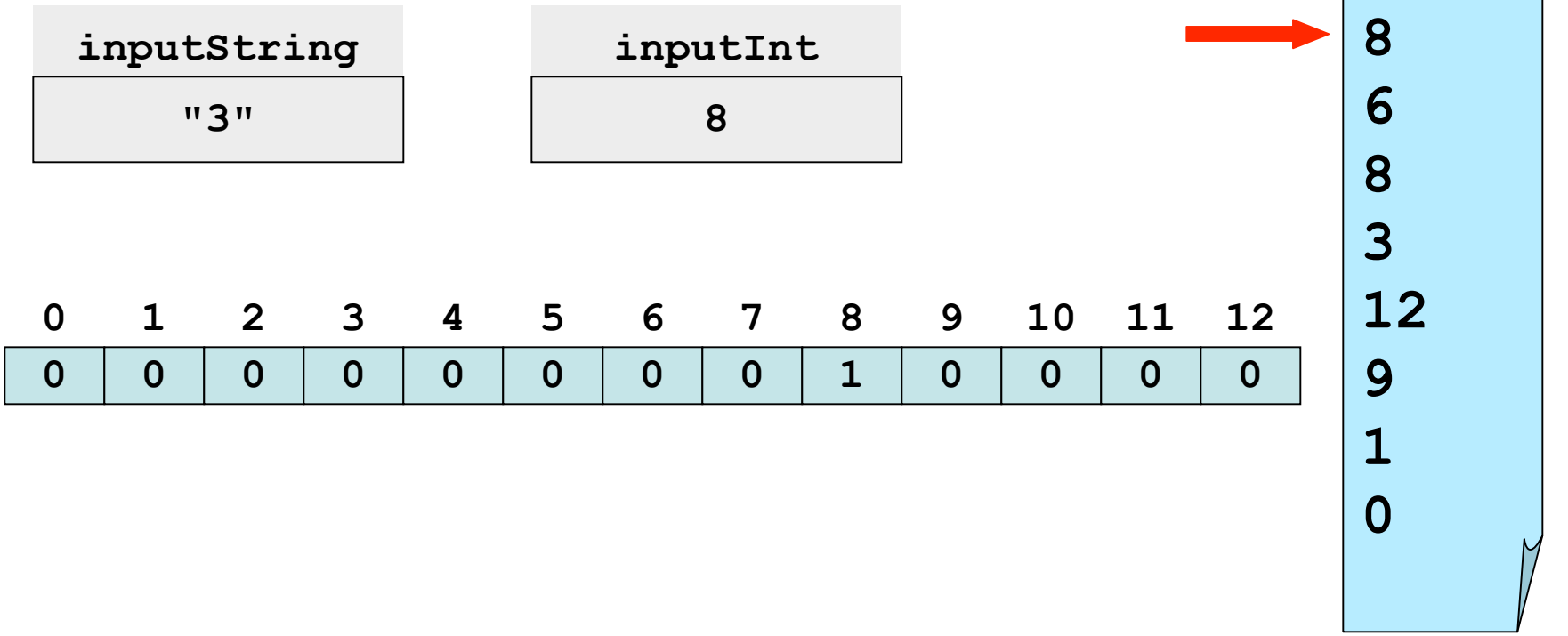
```
inputString = inStream.ReadLine();
while (inputString != null){
   inputInt = Convert.ToInt32(inputString);
   gradeDistribution[inputInt]++;
   inputString = inStream.ReadLine();
}
. . .
```



File I/O - Animation

```
inputString = inStream.ReadLine();
while (inputString != null){
   inputInt = Convert.ToInt32(inputString);
   gradeDistribution[inputInt]++;
   inputString = inStream.ReadLine();
}
. . .
```

True



File I/O - Animation

```
inputString = inStream.ReadLine();
while (inputString != null){
   inputInt = Convert.ToInt32(inputString);
   gradeDistribution[inputInt]++;
   inputString = inStream.ReadLine();
}
. . .
```

grades.yyz

8
3
8
6
8
3
12
9
1
0

inputString
"3"

inputInt
3

0	1	2	3	4	5	6	7	8	9	10	11	12
0	0	0	0	0	0	0	0	1	0	0	0	0

File I/O - Animation

```
inputString = inStream.ReadLine();
while (inputString != null){
   inputInt = Convert.ToInt32(inputString);
   gradeDistribution[inputInt]++;
   inputString = inStream.ReadLine();
}
. . .
```

grades.yyz

8
3
8
6
8
3
12
9
1
0

inputString
"3"

inputInt
3

0	1	2	3	4	5	6	7	8	9	10	11	12
0	0	0	1	0	0	0	0	1	0	0	0	0

File I/O - Animation

```
inputString = inStream.ReadLine();
while (inputString != null){
   inputInt = Convert.ToInt32(inputString);
   gradeDistribution[inputInt]++;
   inputString = inStream.ReadLine();
}
. . .
```

grades.yyz

8
3
8
6
8
3
12
9
1
0

inputString
"1"

inputInt
1

0	1	2	3	4	5	6	7	8	9	10	11	12
0	0	0	2	0	0	1	0	3	1	0	0	1

Magically Skipping Steps

File I/O - Animation

```
inputString = inStream.ReadLine();
while (inputString != null){
   inputInt = Convert.ToInt32(inputString);
   gradeDistribution[inputInt]++;
   inputString = inStream.ReadLine();
}
. . .
```

grades.yyz

8
3
8
6
8
3
12
9
1
0

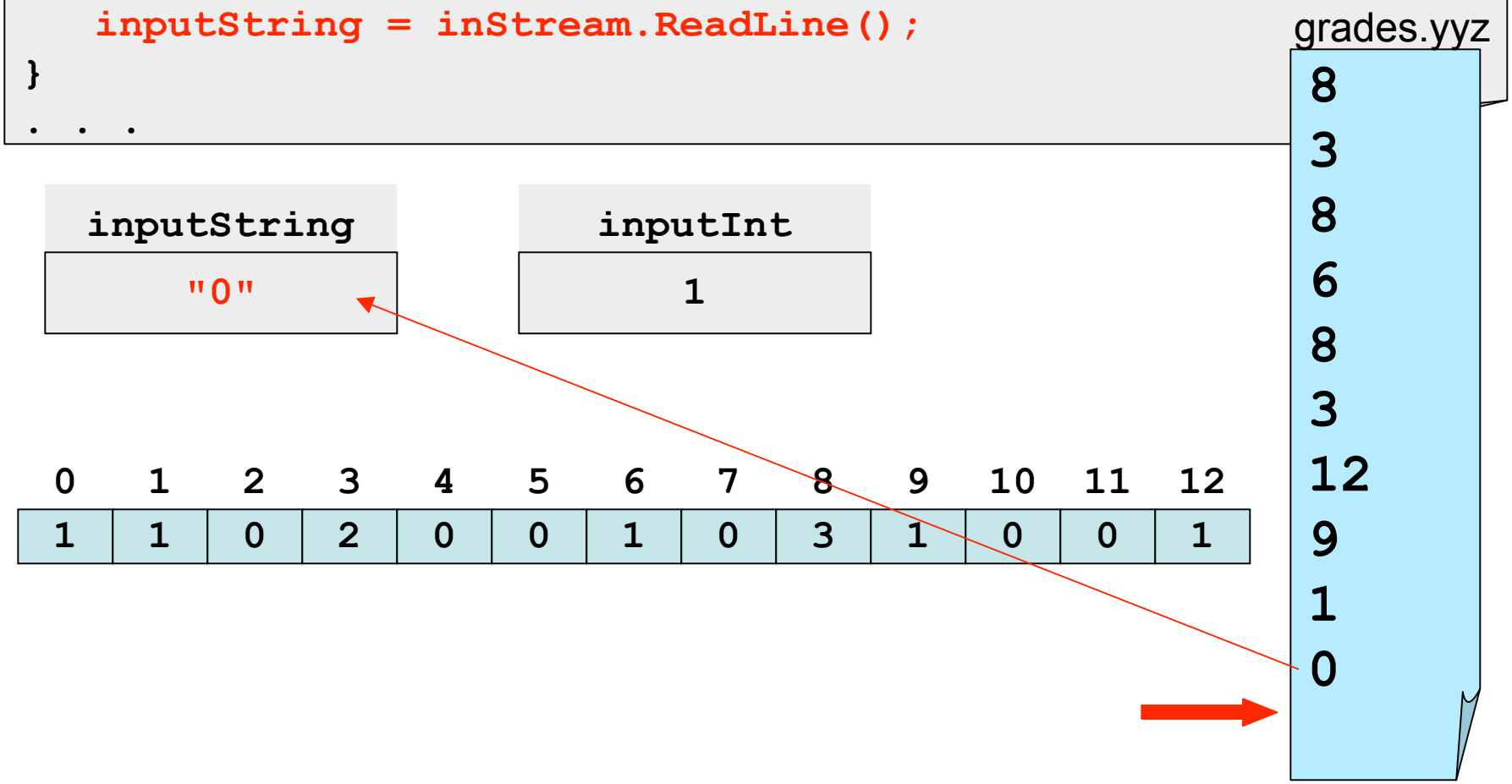
inputString
"1"

inputInt
1

0	1	2	3	4	5	6	7	8	9	10	11	12
0	1	0	2	0	0	1	0	3	1	0	0	1

File I/O - Animation

```
inputString = inStream.ReadLine();
while (inputString != null){
   inputInt = Convert.ToInt32(inputString);
   gradeDistribution[inputInt]++;
   inputString = inStream.ReadLine();
}
. . .
```



File I/O - Animation

```
inputString = inStream.ReadLine();
while (inputString != null){
   inputInt = Convert.ToInt32(inputString);
   gradeDistribution[inputInt]++;
   inputString = inStream.ReadLine();
}
. . .
```

True

grades.yyz

8
3
8
6
8
3
12
9
1
0

inputString
"0"

inputInt
1

0	1	2	3	4	5	6	7	8	9	10	11	12
0	1	0	2	0	0	1	0	3	1	0	0	1

File I/O - Animation

```
inputString = inStream.ReadLine();
while (inputString != null){
   inputInt = Convert.ToInt32(inputString);
   gradeDistribution[inputInt]++;
   inputString = inStream.ReadLine();
}
. . .
```

grades.yyz

8
3
8
6
8
3
12
9
1
0

inputString
"0"

inputInt
0

0	1	2	3	4	5	6	7	8	9	10	11	12
0	1	0	2	0	0	1	0	3	1	0	0	1

File I/O - Animation

```
inputString = inStream.ReadLine();
while (inputString != null){
   inputInt = Convert.ToInt32(inputString);
   gradeDistribution[inputInt]++;
   inputString = inStream.ReadLine();
}
. . .
```

grades.yyz

8
3
8
6
8
3
12
9
1
0

inputString
"0"

inputInt
0

0	1	2	3	4	5	6	7	8	9	10	11	12
1	1	0	2	0	0	1	0	3	1	0	0	1

File I/O - Animation

```
inputString = inStream.ReadLine();
while (inputString != null){
   inputInt = Convert.ToInt32(inputString);
   gradeDistribution[inputInt]++;
   inputString = inStream.ReadLine();
}
. . .
```

grades.yyz

8
3
8
6
8
3
12
9
1
0

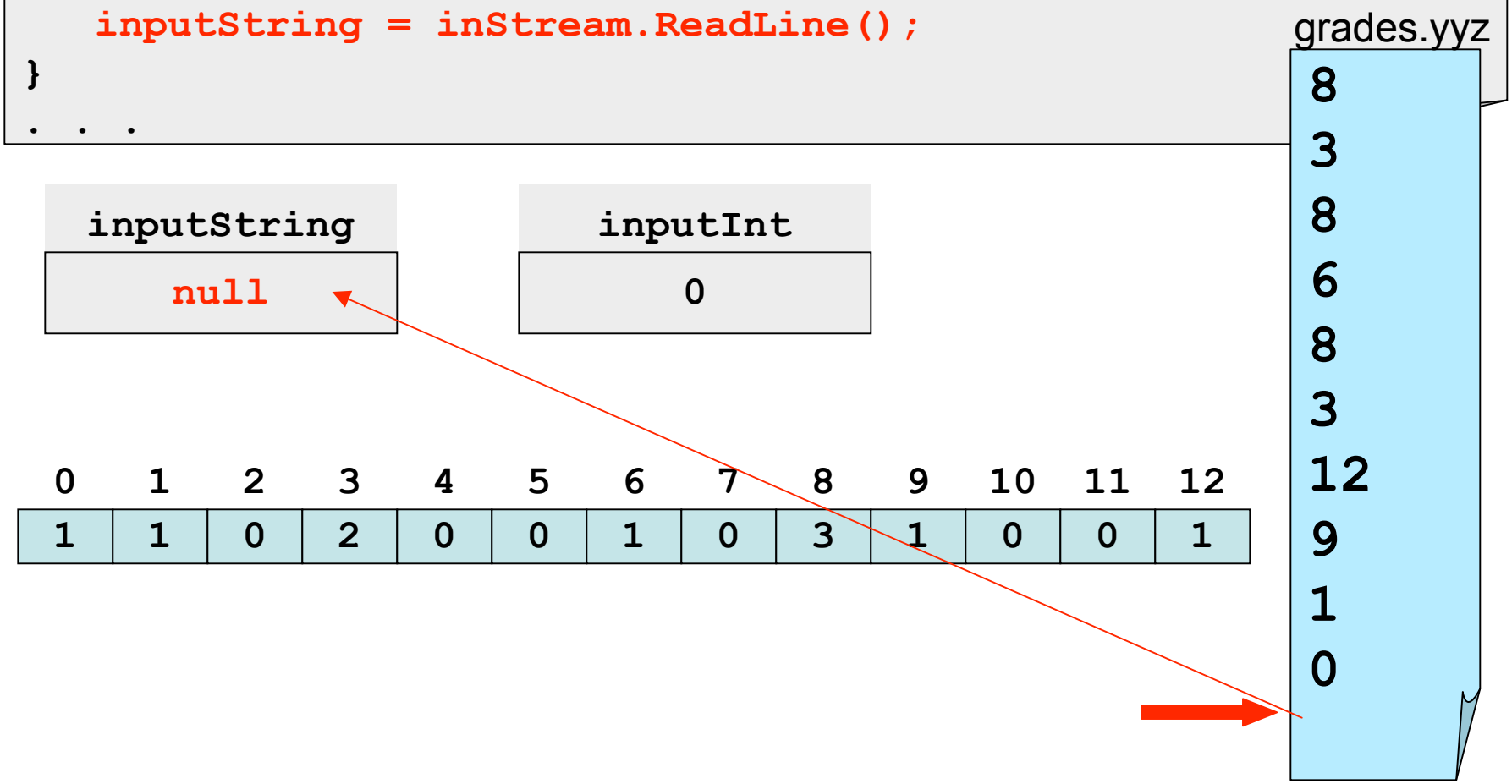
inputString
"0"

inputInt
0

0	1	2	3	4	5	6	7	8	9	10	11	12
1	1	0	2	0	0	1	0	3	1	0	0	1

File I/O - Animation

```
inputString = inStream.ReadLine();
while (inputString != null){
   inputInt = Convert.ToInt32(inputString);
   gradeDistribution[inputInt]++;
   inputString = inStream.ReadLine();
}
. . .
```



File I/O - Animation

```
inputString = inStream.ReadLine();
while (inputString != null){
   inputInt = Convert.ToInt32(inputString);
   gradeDistribution[inputInt]++;
   inputString = inStream.ReadLine();
}
. . .
```

False

grades.yyz

8
3
8
6
8
3
12
9
1
0

inputString
null

inputInt
0

0	1	2	3	4	5	6	7	8	9	10	11	12
1	1	0	2	0	0	1	0	3	1	0	0	1

File I/O

```
. . .
   inputString = inStream.ReadLine();
   while (inputString != null){
      inputInt = Convert.ToInt32(inputString);
      gradeDistribution[inputInt]++;
      inputString = inStream.ReadLine();
   }
   inStream.Close();
}
```

- Streams always need to be closed when they are finished being used.
- If they don't get closed, unexpected results may appear in the files that are being accessed.

Output to Data File

- Now that our program knows the distribution of grades from our input file, it would be nice to produce a graph.
- Graphs are not simple to produce.
- However, we can easily output our data to a file and open it in Excel. With just a few clicks in Excel, a graph can be created.

Output to Data File

- What information do I want to put in the file?
- What format do I need to use?

Output to Data File

- What information do I want to put in the file?
  - The grade points
  - The number of students with each grade point.
- What format do I need to use?
  - A file that is tab delimited is easily accessed in Excel.
    - In this case, tab delimited means that the tab character separates entries on any line,
    - i.e. grade_point_average *tab* number_of_students

Algorithm for Output to File

- Run previously developed program.
- Start at beginning of grade point array
- Print a line in a file that reads:
  - arrayIndex arrayContents
    - Where arrayIndex is the index of grade point array.
    - arrayContents is the numerical value in that index.
    - arrayIndex and arrayContents are separated by a tab.
- Loop through grade point array until all grade points have been written to the file.

Output to Data File

- Run previously developed program.
- Start at beginning of grade point array.
- Print a line in a file that reads:
  - arrayIndex arrayContents
- Loop through grade point array until all grade points have been written to the file.

```
. . Program to input / create data not shown on this slide. .

StreamWriter outStream = new StreamWriter("output.txt");
for (int i = 0; i < gradeDistribution.Length; i++){
   outStream.WriteLine(i + "\t" + gradeDistribution[i]);
}
outStream.Close();
```

Output to Data File

- This declares an output stream.
- It creates a file called "output.txt" in the application's working directory.
- We can write strings to this output stream in the same way we have used strings in message boxes for example.

```
. . Program to input / create data not shown on this slide. .

StreamWriter outStream = new StreamWriter("output.txt");
for (int i = 0; i < gradeDistribution.Length; i++){
   outStream.WriteLine(i + "\t" + gradeDistribution[i]);
}
outStream.Close();
```

Output to Data File

- Run previously developed program.
- Start at beginning of grade point array.
- Print a line in a file that reads:
  - arrayIndex arrayContents
- Loop through grade point array until all grade points have been written to the file.

```
. . Program to input / create data not shown on this slide. .

StreamWriter outStream = new StreamWriter("output.txt");
for (int i = 0; i < gradeDistribution.Length; i++){
   outStream.WriteLine(i + "\t" + gradeDistribution[i]);
}
outStream.Close();
```

Output to Data File

- Run previously developed program.
- Start at beginning of grade point array.
- Print a line in a file that reads:
  - arrayIndex arrayContents
- Loop through grade point array until all grade points have been written to the file.

```
. . Program to input / create data not shown on this slide. .

StreamWriter outStream = new StreamWriter("output.txt");
for (int i = 0; i < gradeDistribution.Length; i++){
   outStream.WriteLine(i + "\t" + gradeDistribution[i]);
}
outStream.Close();
```

Output to Data File

- WriteLine() does all the magic of writing to a file.
- It writes a single string to our output file (ends with "\n").
- \t is the code for a tab.
- "\t" is a string with a tab in it

```
. . Program to input / create data not shown on this slide. .

StreamWriter outStream = new StreamWriter("output.txt");
for (int i = 0; i < gradeDistribution.Length; i++){
   outStream.WriteLine(i + "\t" + gradeDistribution[i]);
}
outStream.Close();
```

Output to Data File

- Streams must be closed when we are finished using them.

```
. . Program to input / create data not shown on this slide. .

StreamWriter outStream = new StreamWriter("output.txt");
for (int i = 0; i < gradeDistribution.Length; i++){
   outStream.WriteLine(i + "\t" + gradeDistribution[i]);
}
outStream.Close();
```

Sample Output

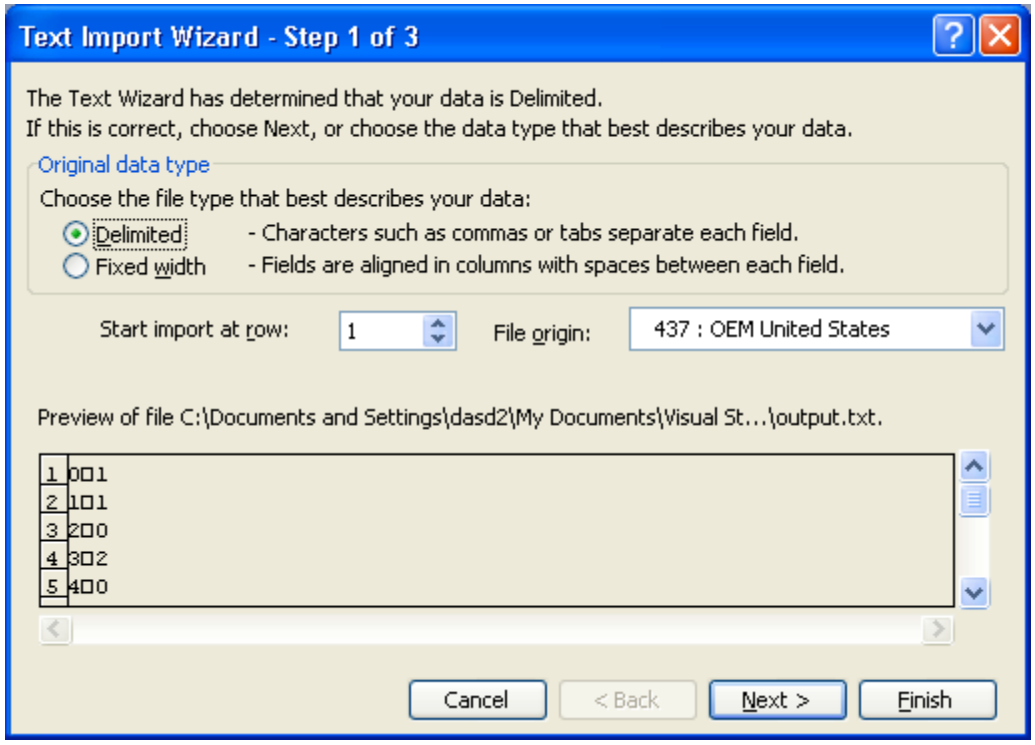
- Output from Sample Input:

output.txt

```
0    1
1    1
2    0
3    2
4    0
5    0
6    1
7    0
8    3
9    1
10   0
11   0
12   1
```

Sample Output

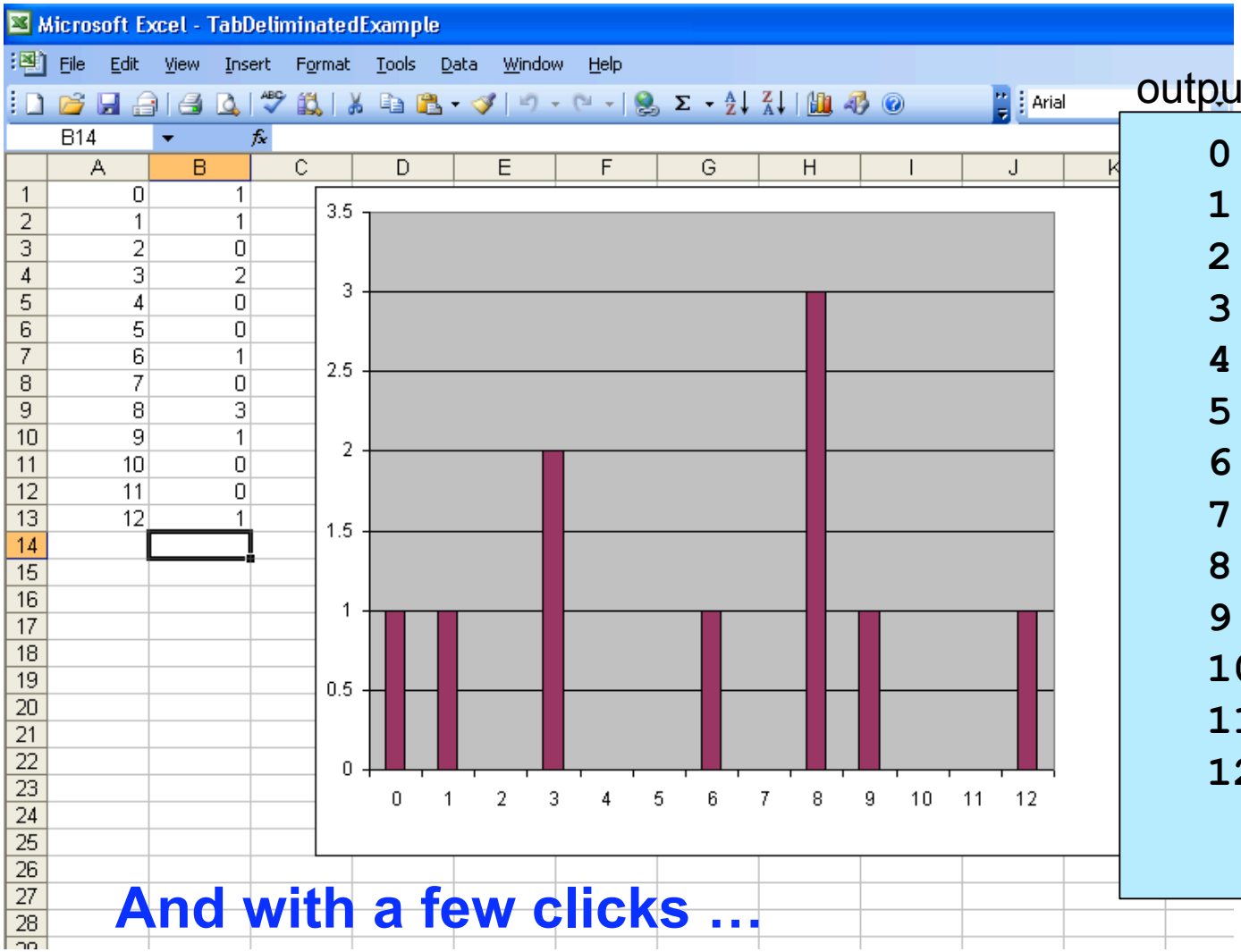
- If we open our file in Excel, the following message box will appear:



Sample Output

- Excel starts this wizard if plain text files are opened with Excel.
- Because of our explicit delimiting of our file with tabs, we can just click finish.

Sample Output



output.txt

0	1
1	1
2	0
3	2
4	0
5	0
6	1
7	0
8	3
9	1
10	0
11	0
12	1

And with a few clicks …

File I/O

- The brilliant part of file I/O like this is that it doesn't matter how big or small our input file is.
- The software will read through the entire file and create *sensible* output.

File Dialog Boxes

- Easy to include the normal Windows dialogs for browsing to a specific location to store or retrieve a file
- Can add OpenFileDialog or SaveFileDialog by dragging from Toolbox to the design view onto your form
- The ShowDialog() method shows the dialog
- The FileName property gives the filename