Contents

Todo List

Description

Purpose

This project is designed to teach you how to devise, implement, and submit solutions to the simple programming problem of constructing a "todo list software". It aims at making sure that you master the simple concepts of control structures and data manipulation before introducing more advanced concepts.

Challenge

In short Develop a simple program that asks the user to provide their todo list, and then tracks the completion of the items (or "tasks") on that list.

In more details

1. Your program should start by asking the user to provide items for their todo list, one by one.
2. Once the user is done providing the items, it should display the todo list, with a number associated to each item, and its status (done or not done).
3. Then, your program should ask the user to enter the number of the item they have just completed. There are three cases:
   (a) If the user enters something that is not a number, your program should simply ask again.

(b) If the user enters an "invalid" number (that is, that does not correspond to the number of an item), your program should ask again.
(c) If the user enters the number of an item that is not done, its status should become "done".

4. Once the user entered the number of item, the updated todo list should be displayed, and the user should be asked for another number of an item.
5. Once the user completed all the items in the list, the program should display a celebratory message about being done.

Submission

Please, follow our guideline on project submission. In particular, make sure you write your name and the date in a delimited comment at the beginning of your file.

Example

Here is an example of execution, where the user input is underlined, and hitting "enter" is represented by "↵":

```
What is on your todo list? Enter "done" when you are done.
Make sure my IDE is still
↪ working.↵
What is on your todo list? Enter "done" when you are done.
Compile a simple "Hello World"
↪ program.↵
What is on your todo list? Enter "done" when you are done.
Start working on this project.↵
What is on your todo list? Enter "done" when you are done.
done↵
Here is your current todo list:
| # | Status | Task |
| 1 |   ☐    | Make sure my IDE is still working.
| 2 |   ☐    | Compile a simple "Hello World" program.
| 3 |   ☐    | Start working on this project.
Enter the number of the task you completed.
Not yet.↵
Enter the number of the task you completed.
1↵
Here is your current todo list:
| # | Status | Task |
| 1 |   ☑    | Make sure my IDE is still working.
| 2 |   ☐    | Compile a simple "Hello World" program.
```

```
| 3 |   ☐    | Start working on this project.
Enter the number of the task you completed.
3 ↵
Here is your current todo list:
| # | Status | Task |
| 1 |   ☑    | Make sure my IDE is still working.
| 2 |   ☐    | Compile a simple "Hello World" program.
| 3 |   ☑    | Start working on this project.
Enter the number of the task you completed.
4 ↵
Enter the number of the task you completed.
2 ↵
You're all done, congratulations!

Press any key to continue...
```

Bonuses

- The behaviour of the program if the user enters the number of an item whose status is "done" is not specified above. Write (as a comment) in your program which behaviour you implemented, and test it.
- Complete the project without resizing arrays.
- Improve the way the todo list is displayed using string formatting.
- Display, along with the list of items, the completion rate: for example, after the user completed the first of their list of 4 items, the program should display "You are 25% done!".

Submission

Please, follow our guideline on project submission. In particular, make sure you write your name and the date in a delimited comment at the beginning of your file.

Solution

Simplest Solution

A possible solution, using arrays but not resizing them, is as follows:

```
using System;

public class Program
{
  public static void Main(string[] args)
  {
```

```
// Variable declarations.

string[] todo = new string[100]; // This will hold the
 ↪  items in the todo list.
// Note that we are arbitrarily deciding that the
 ↪  maximum number of items is 100.
bool[] status = new bool[100]; // This will hold the
 ↪  status of each item.
// true means "done", false means "not done".
string uInput; // This will hold user input.
int todoSize = 0; // This will hold the actual number
 ↪  of items in the list.
int completed = 0; // This will hold the number of
 ↪  items done.
int justdone; // This will hold the number of the last
 ↪  item completed.
bool valid; // This will hold true if the user input
 ↪  is valid (a positive number
// less than the number of items in the list), false
 ↪  otherwise. Used for user-input
// validation.
char itemStatus; // This will hold '☑' if the current
 ↪  item is done,
// '☐' otherwise.

// We start by populating the list with items.
do
{
  Console.WriteLine(
    "What is on your todo list? Enter \"done\" when
     ↪  you are done."
  );
  uInput = Console.ReadLine();
  if (uInput != "done")
  {
    todo[todoSize] = uInput; // We can store the first
↪  item at index todoSize
    // since its initial value is 0.
    todoSize++; // We increment the number of items in
↪  the list.
  }
} while (uInput != "done"); // When the user enters
 ↪  "done", we exit this loop.

// We now display the todo list, and ask the user to
 ↪  indicate which item they
```

```
    // completed, as long as there are some items left in
     ↪  their list.

    while (completed != todoSize)
    {
      // We display the todo list.
      Console.WriteLine("Here is your current todo
↪   list:");
      Console.WriteLine("| # | Status | Task |");
      for (int i = 0; i < todoSize; i++)
      {
        if (status[i])
        {
          itemStatus = '☑';
        }
        else
        {
          itemStatus = '☐';
        }
        Console.WriteLine(
          "| "
            + (i + 1)
            + " |   "
            + itemStatus
            + "    | "
            + todo[i]
        );
      }
      // We now ask the user to enter the number of the
       ↪  completed item.
      valid = false; // We assume that the user has not
↪   given a valid value yet.
      do
      {
        Console.WriteLine(
          "Enter the number of the task you completed."
        );
        valid =
          int.TryParse(Console.ReadLine(), out justdone)
          && 0 < justdone
          && justdone <= todoSize;
      } while (!valid);
      status[justdone - 1] = true; // We indicate that the
↪   item was completed by setting its value to true.
      completed++; // We increment the number of items
↪   completed.
```

```
      Console.WriteLine(
        $"You are {completed / (double)todoSize:P} done!"
      );
      // Note that we force double division using casting,
       ↪  and use the :P format speficier.
    }
    Console.WriteLine("Congratulations!");
  }
}
```

You can download it here

Using Classes

Another solution is to create a class for "todo list items" and to create an array of them. That is, have a class file `Todo.cs` along the lines of

```
class Todo{
    public string Description{get; set;}
    public bool Status{get; set;}
}
```

and then to create and manipulate arrays of `Todo` objects, for example as follows:

```
Todo[] todoList = new Todo[100];
todoList[0] = new Todo();
todoList[0].Description = "My first item";
todoList[0].Status = false;
Console.Write(todoList[0].Description +
 ↪  (todoList[0].Status ? " done" : " not done"));
```