Old Keypad Phone Converter

Description

The program asks the user to input an old keypad phone number sequence to create a text message and returns a text message that shows on screen.

Usage

Described in README.

Code Explanation

Create a Dictionary to map keys to the characters.

```
private static readonly Dictionary<char, string> Keypad = new()
{
    { '1', "!@#$%^&*()" },
    { '2', "ABC" },
    { '3', "DEF" },
    { '4', "GHI" },
    { '5', "JKL" },
    { '6', "MNO" },
    { '7', "PQRS" },
    { '8', "TUV" },
    { '9', "WXYZ" },
    { '0', " " }
};
```

Create a method to convert to text message. Set output to be returned later and currentKey to track which key to get the character. Loop each character in the input string and handle different conditions.

```
public static string OldPhonePad(string input)
{
    string output = "";
    string currentKey = "";
    foreach (char c in input)
    {
        //handles each condition
    }
    return output;
}
```

If the character is '#', this block handles the last tracked key to convert to a letter.

```
if (c == '#')
{
    if (!string.IsNullOrEmpty(currentKey))
    {
        ConvertKeyToLetter(currentKey, ref output);
    }
}
```

If the character is '*', this block performs as a backspace function. The first 'if' block is for the case when currentKey is not empty (eg. 2233*#). It clears the currentKey so there is no key to convert. The second 'if' block is for the case when current key is empty but current accumulated output is not empty (eg. 2233 *#). The block to handle whitespace characters clears the currentKey so the second block handles such cases.

```
else if (c == '*')
{
    if (!string.IsNullOrEmpty(currentKey))
    {
        currentKey = "";
    }
    else if (!string.IsNullOrEmpty(output))
    {
        output = output[..^1];
    }
}
```

If the character is ' ', firstly the current tracked key is converted to a character then it clears the currentKey to track what comes after the whitespace.

```
else if (c == ' ')
{
    if (!string.IsNullOrEmpty(currentKey))
    {
        ConvertKeyToLetter(currentKey, ref output);
        currentKey = "";
    }
}
```

This block handles all the normal cases. If there is a change in number without space (eg. 2233#), it first converts the currentKey then clears the currentKey. After that, it concats the new number to currentKey. If there is no change in the number, it keeps adding to currentKey until '#' character comes to convert to letter.

```
else
{
    if (!string.IsNullOrEmpty(currentKey) && c != currentKey[0])
    {
        ConvertKeyToLetter(currentKey, ref output);
        currentKey = "";
    }
    currentKey += c;
}
```

This is the method to convert keys to corresponding letters. It sets the key and count of that key in the input. Then, it tries to get value from the dictionary and adds to the output. I use modulus here to start from the beginning if the user overtypes (eg. 2222# returns A because A has only 3 letters).

```
private static void ConvertKeyToLetter(string currentKey, ref string
output)
{
    char key = currentKey[0];
    int keyCount = currentKey.Length;
    if (Keypad.TryGetValue(key, out string? letters))
    {
        output += letters[(keyCount - 1) % letters.Length];
    }
}
```

Conclusion

This is the detailed code explanation of the program. Email to minnmoeyanoo@gmail.com if you have any questions.