Sabancı University

Faculty of Engineering and Natural Sciences
CS204 Advanced Programming
Spring 2019

Homework 8 – Binary to Decimal Converter with GUI

Due: 17/05/2019, Friday, 21:00

PLEASE NOTE:

Your program should be a robust one such that you have to consider all relevant programmer mistakes and extreme cases; you are expected to take actions accordingly!

You can NOT collaborate with your friends and discuss solutions. You have to write down the code on your own. Plagiarism will not be tolerated!

Introduction

In this homework, you are asked to develop a simple Binary-to-Decimal Converter. This converter lets the user write a binary string (using only 0s and 1s) using the keyboard or a button for each number. It then allows him or her to convert the resulting binary string to a decimal number and then add it to a List Box.

GUI Design

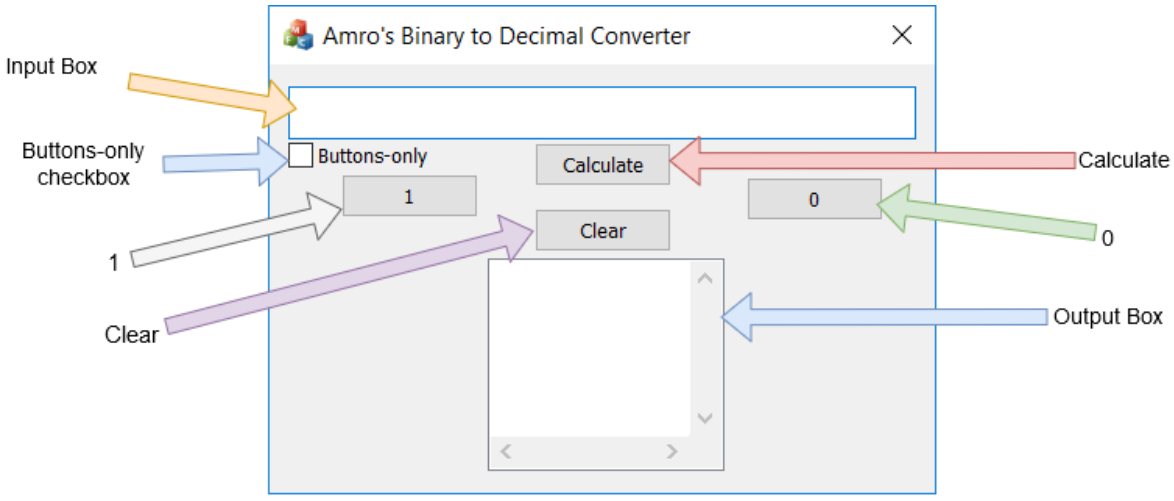

Figure 1: The GUI with each element and its name in the document

The title of the application must be "*Your name*'s Binary to Decimal Converter".

All GUI elements that you can see are available as MFC elements (objects). We now list their types and what each element should do.

***Input Box*:** This `EditControl` element is where the input of the user will be shown. *Input Box* must be enabled at the beginning of the program, meaning that the user will be able to write on it using the keyboard.

1: This `Button` element will add a *1* as the least significant bit of the number of the *Input Box* (i.e. to the right). For example, if the *Input Box* had the following value:

0010

Then, clicking the button **1** will result in the following value in the *Input Box*:

00101

0: This `Button` element will add a *0* as the least significant bit of the number of the *Input Box* (i.e. to the right). For example, if the *Input Box* had the following value:

1101

Then clicking the button **0** will result in the following value in the *Input Box*:

11010

***Clear*:** This `Button` element will clear everything that currently exists inside the *Input Box* and make the input box empty.

***Buttons-only Checkbox*:** This `CheckBox` will control the read-only property of the *Input Box* element. If it is unchecked, then the *Input Box* will be editable using keyboard. If the *Buttons-only Checkbox* is checked, the *Input Box* becomes disabled, meaning that the user will not be able to write on it using the keyboard; however, he or she will still be able to write on it using the buttons **0** and **1** and it will be clearable with the button *Clear*. When the program starts, this checkbox is unchecked by default. Changing the editability status of an EditControl object is done using `SetReadOnly` function which is explained in **GUI_document.pdf** file given together with Lab13 material on the web site.

***Calculate*:** This `Button` is where all the logic of the application is to be implemented. When pressing this button, you must calculate the decimal equivalent of the binary number written in the *Input Box* and add it to the *Output Box*. Before this calculation, there are two types of input checks that must be performed when this button is pressed. These are, (i) checking that the number of bits is between and including 2 and 32, and (ii) checking that the input is made of 0s and 1s only. If the input checks fail, appropriate messages should be displayed.

***Output Box*:** This `ListBox` element will contain the outputs of previous calculations, as well as to display any error messages. It must be vertically and horizontally scrollable. Please consult the GUI_document.pdf on the website (lab 13 link) for directions on how to create a horizontally and vertically scrollable list box. In addition, the list box **must not be sorted.** List boxes are sorted by default so you must disable the property "Sorted" manually in the properties menu of this ListBox item.

Work Flow

A user will input a binary number to the *Input Box* using the keyboard and/or by clicking on 0 and 1 buttons. The user can disable the *Input Box* using the *Buttons-only Checkbox*. If that happens, the user will only be able to insert inputs using the 0 and 1 buttons. During the program, *Buttons-only Checkbox* can be checked and unchecked multiple times.

Once the user has some input inside the *Input Box*, he or she can press the *Clear* button to empty the contents in the *Input Box*, or press the *Calculate* button to convert the binary input into a decimal number. The exact operation of calculation is broken down to following steps that must be implemented inside the *Calculate* button's **BN_CLICKED** action:

1) Read the input string inside *Input Box* and store it in a `CString` variable.
2) Make the necessary input checks mentioned below.
3) Convert the binary input to a decimal number assuming the input number is written in 2's complement format.
4) Finally, add the resulting decimal number to the *Output Box* element in the form "The number <input_in_binary> is <decimal value> in decimal". For example, if the Input box had:

   01101

   Pressing *Calculate* will add to the *Output Box* the following string:

   "The number 01101 in binary is 13 in decimal"

   As another example, entering the number 10010 to the *Input Box* and pressing *Calculate* will add to the *Output Box*:

   "The number 10010 in binary is -14 in decimal"

   However, if an error occurred, add the error message resulting from this calculation to *Output Box* as will be detailed below.

The details of carrying out these operations were explained in the labs and will be overviewed in the GUI document posted at the web site (lab 13 link).

There are two important **input checks** that you must perform when the user presses *Calculate* before doing the calculation:

a. **Number of bits in the *Input Box*:** There must be **at least two and at most 32 bits** in the *Input Box*. If a user tries to make a calculation using an empty box or with only one bit ("0" or "1"), you must stop the calculation and add the following text to *Output Box*:

"Please insert minimum of 2 bits in the input box"

If the user enters 33 or more bits in the *Input Box* and press *Calculate* button, you must stop the calculation and add the following text to *Output Box*:

"Please insert maximum of 32 bits in the input box"

b. **Only 0s and 1s are allowed:** In the *Input Box*, if the user entered any character other than '0' and '1', then you must stop the calculation and add the following error message to *Output Box*:

"You have entered an illegal character"

For binary to decimal conversion, you have to develop your own algorithm. If needed, you can use `pow` function of *cmath* library for exponentiation. Since you get the input of *Input Box* in `CString` type, but process as `int`, and display in `CString` type again, you may need to make some data conversions. The details of these data conversions are given in **GUI_document.pdf** file given together with Lab13 material on the web site.

Sample Run

Some sample snapshots are given below. However, the homework is better understood if you run the implemented version. In this homework zip package, we provide you an .exe file of the bit calculator that we developed. You can play with this sample program and see how the program operates.

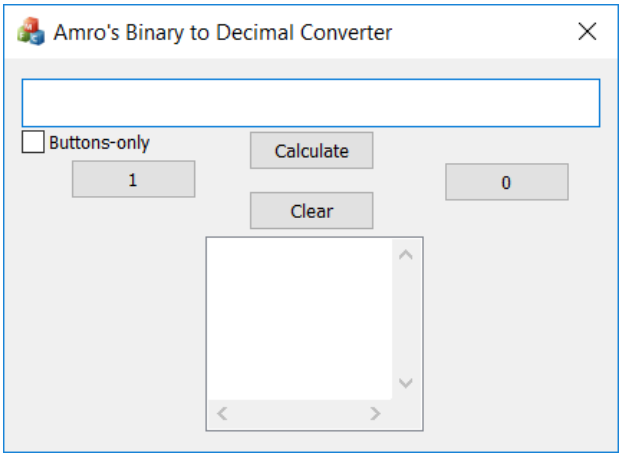

Figure 2: Initial screen

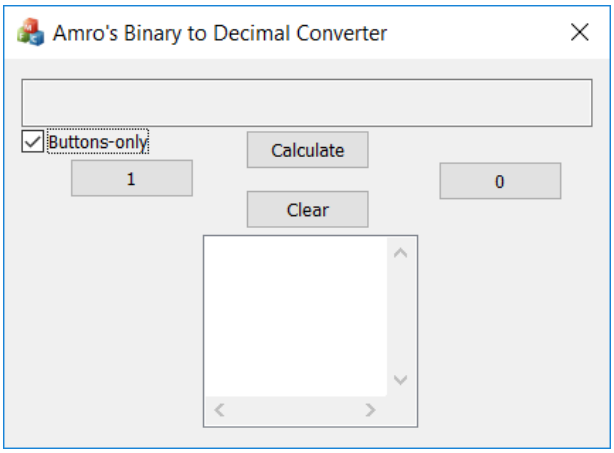

Figure 3: after checking "Buttons-only"

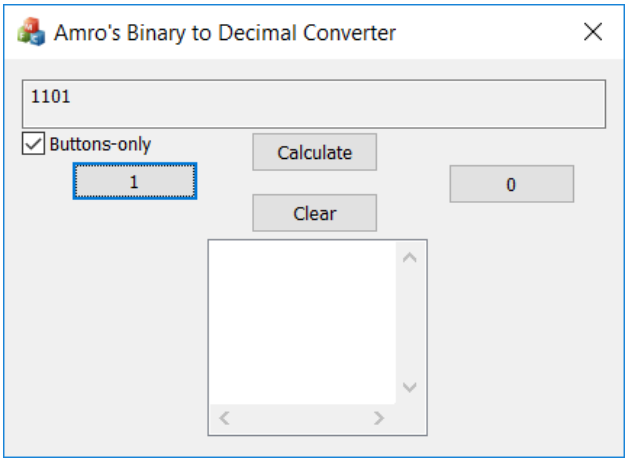

Figure 4: After pressing the buttons "1","1","0","1"

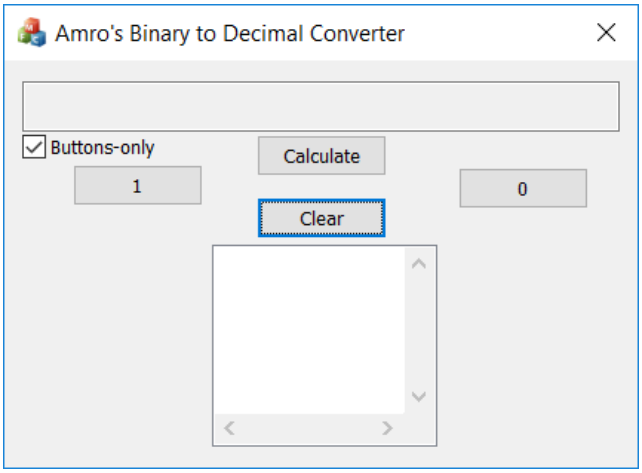

Figure 5: After pressing Clear

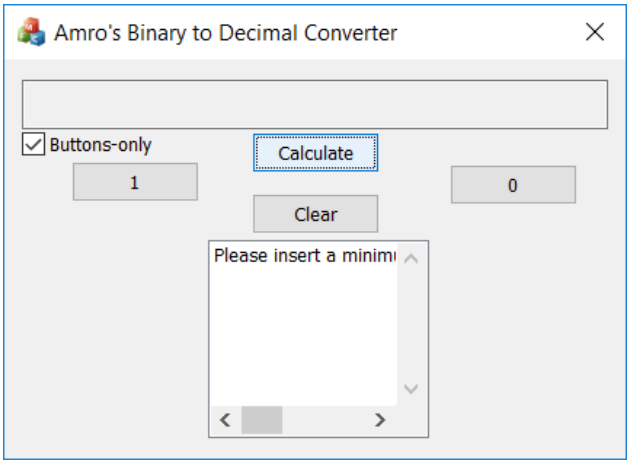

Figure 6: After pressing Calculate with an empty Input Box (notice the added error message)

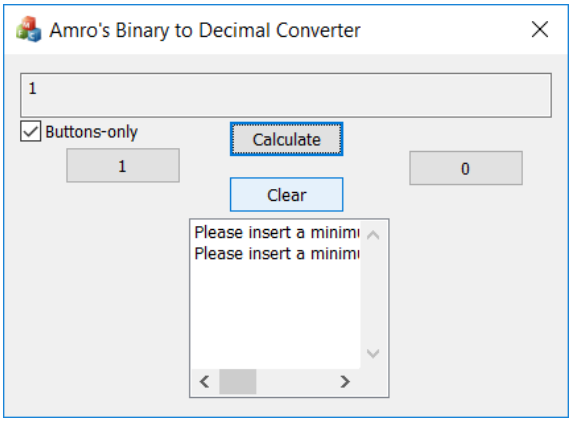

Figure 7: After clicking "1" and pressing Calculate (an error occurred because the program needs a minimum of 2 bits to calculate)

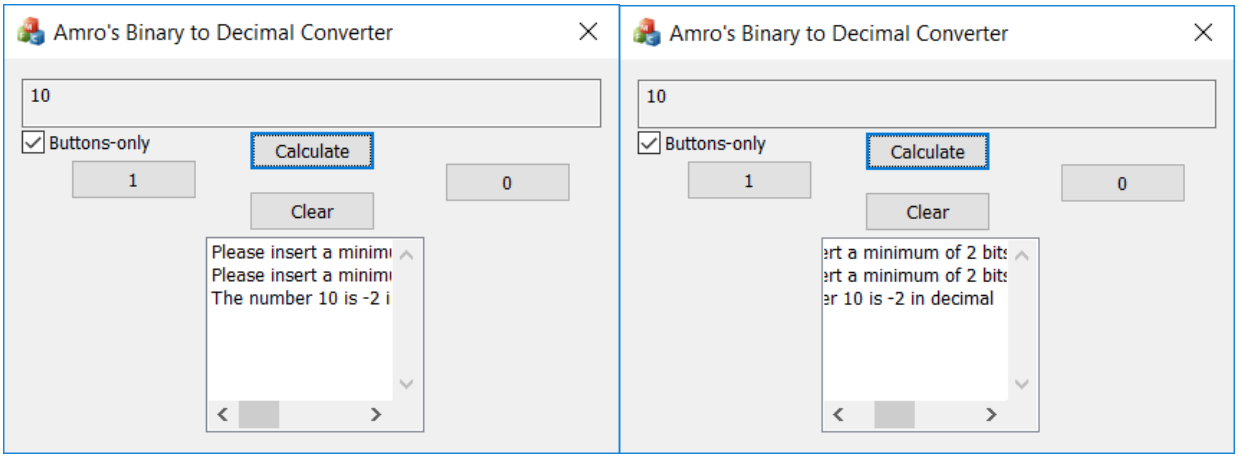

Figure 8: clicked '0' so the input box has '10' and then calculate to find -2 (left). The same output as the left picture but the Output Box is scrolled to the right (right)

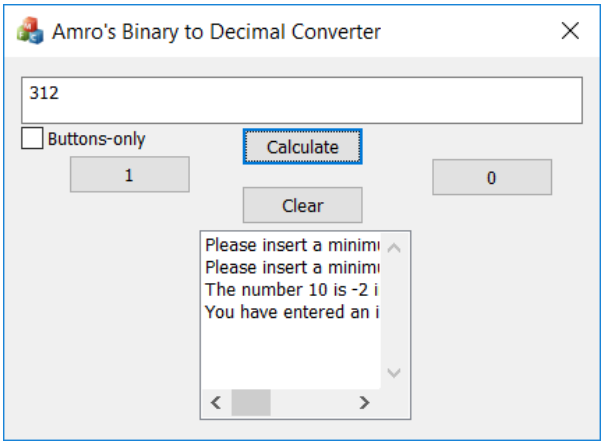

Figure 9: Unchecked Buttons-only, typed 312 and clicked Calculate. Notice the error

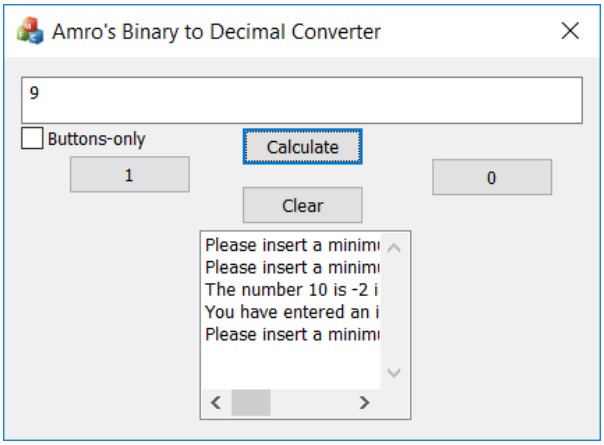

Figure 10: Used '9' as an input. Notice that 9 fails both input checks (is both < 2 bits and not 0 or 1). However, we print the error of the input size (<2 bits)

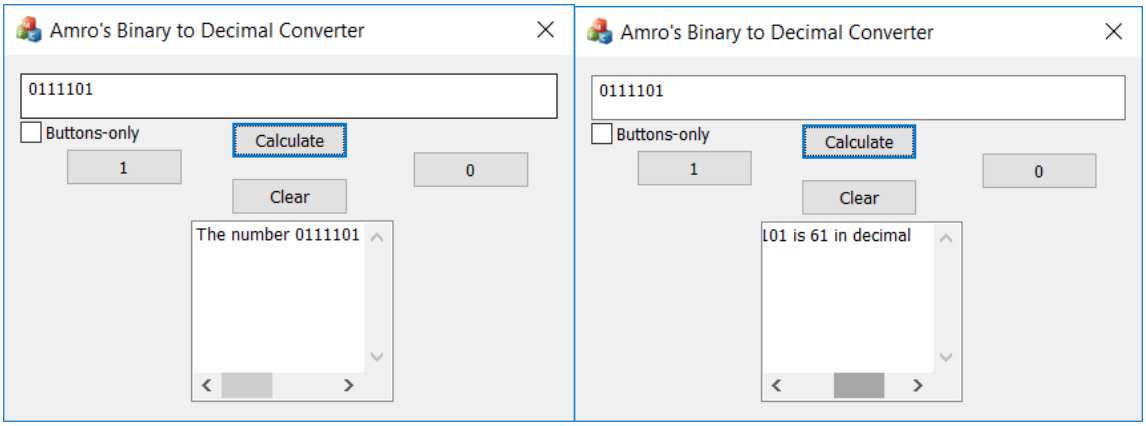

Figure 11: (New Run) typed 0111101 and pressed calculate (left). The rest of the output is shown after scrolling the Output Box (right)

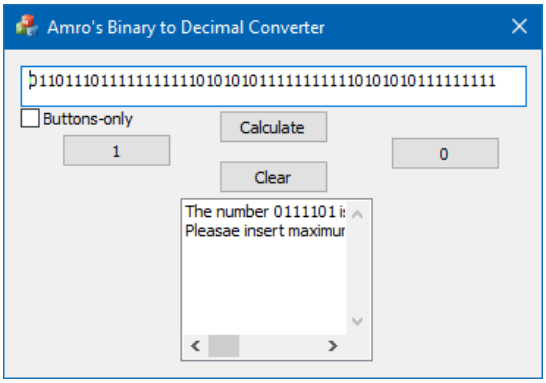

Figure 12: User typed 01101110111111111110101010111111111110101010111111111 and pressed calculate (there are more than 32 bits in the input box)

ATTENTION

Please see previous homeworks for the general rules that you have to follow.

Submission rules are similar to the previous homeworks, but there are some special considerations that you have to be very careful since this is a GUI (MFC) application.

- Use the same naming convention as in the previous homework assignments.
- Since this is not a console application, adding an existing cpp to a new project does **NOT** work. You have to develop your GUI application from scratch using the methods explained in the labs. Please check out the Lab13 link at the website for details.
- There are some other very important project files in addition to the cpp and header files. Thus please pay extreme attention that all project files exist in your zip file.
- Since the entire project would be very large (a couple of 100s of Mbytes), you will delete some files and folder while submitting. These are:
    - the large file with *.sdf* extension
    - the folder named *ipch*
    - all *debug* and *release* folders
- These files and folders, which are deleted while submitting, regenerate themselves when you open the solution again. Thus, there is no harm not to submit those. However, please make sure that without those files, your solution opens up and your program compiles and runs correctly.
- As in the other homework assignments, correct submission is part of your duty; if we cannot run your program, we cannot grade it.

Good Luck!
Albert Levi, Amro Alabsi Aljundi