

Arpit Singh

19BCG10069

RenderPub – TakeHome Task

Technical Task

(Placement)

**GitHub Link:** <https://github.com/TSM-ArpitSG/RenderPub-Task>

**Video Link:** [https://drive.google.com/file/d/1AETpW4LWrz2Xz9ejV5SYMrz0RaK-i\\_pQ/view?usp=sharing](https://drive.google.com/file/d/1AETpW4LWrz2Xz9ejV5SYMrz0RaK-i_pQ/view?usp=sharing)

**Task:**

**C++ Program:**

Using C++, write a program to implement Undo/Redo for arithmetic operations on a variable based on user input.

- Initialize float x = 10.0
- Users should be able to interact with the programs by typing in commands (Strings)
- Pressing “Enter” key should execute the command
- Pressing “Escape” key should exit the application

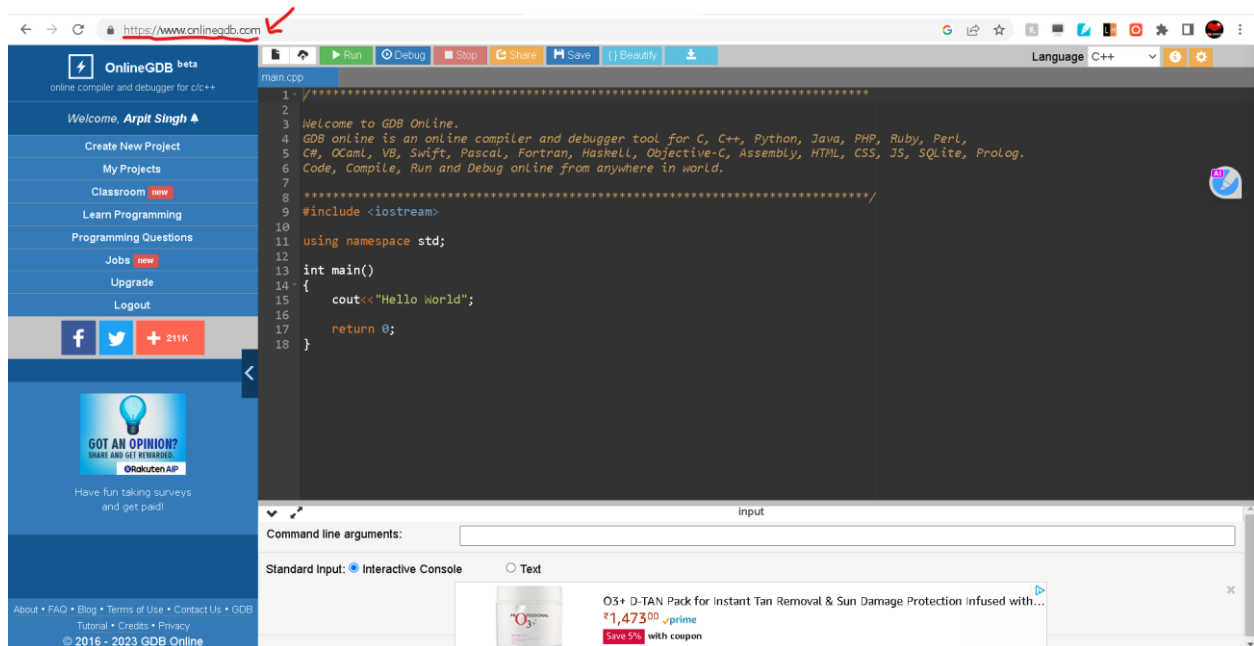
User Command Examples (String)	Expected Action
+5	Adds 5 to x
-3	Subtracts 3 from x
*7	Multiplies x by 7
/2	Divides x by 2
x	Prints the current value of x
z	Performs undo of the previous operation on x
y	Performs redo of the previous undo operation on x

**Note:** Redo operation should be allowed ONLY when it is preceded by one or more undo operations.

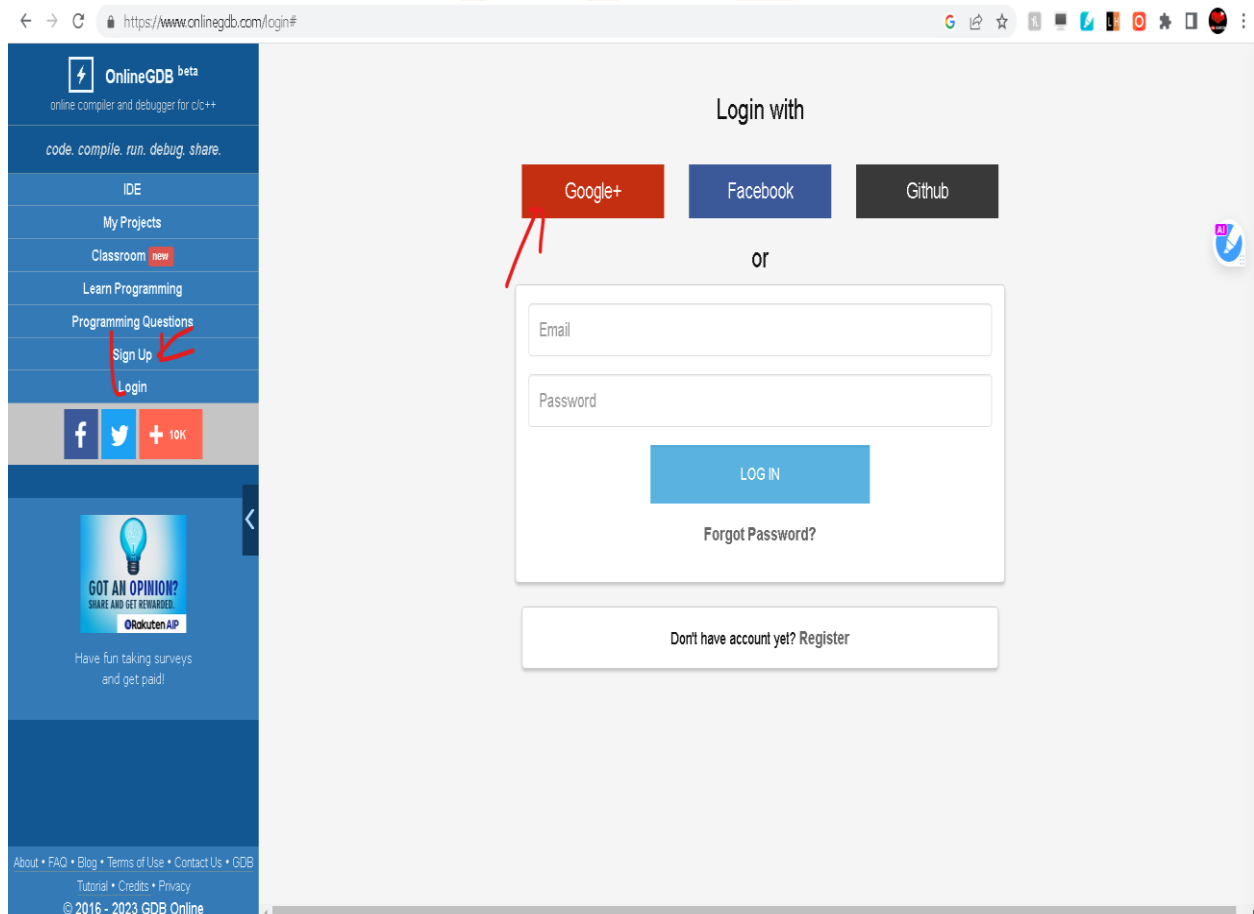
**Steps:**

1. **Go to a Online C++ Compiler or Use a IDE in Your Local Machine(I Used a Online Compiler):**

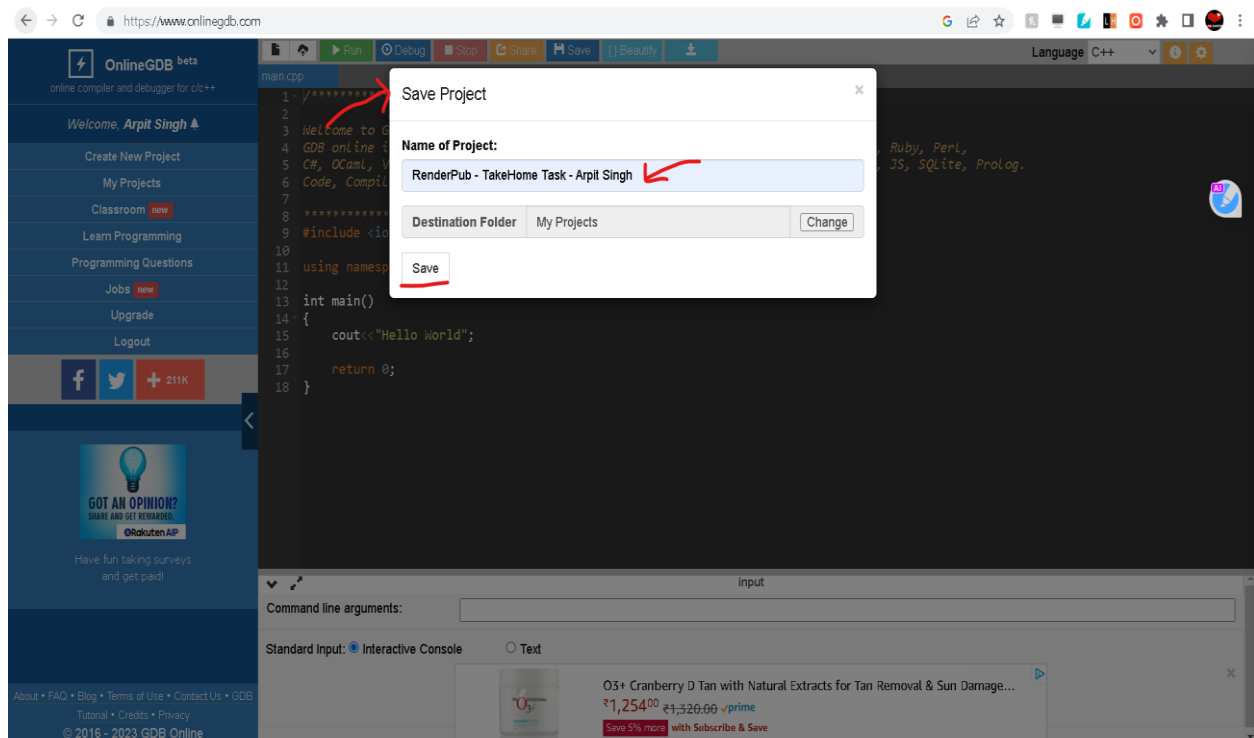
- a. Go to this **Website** which provides an **Online Compiler** - <https://www.onlinegdb.com/>



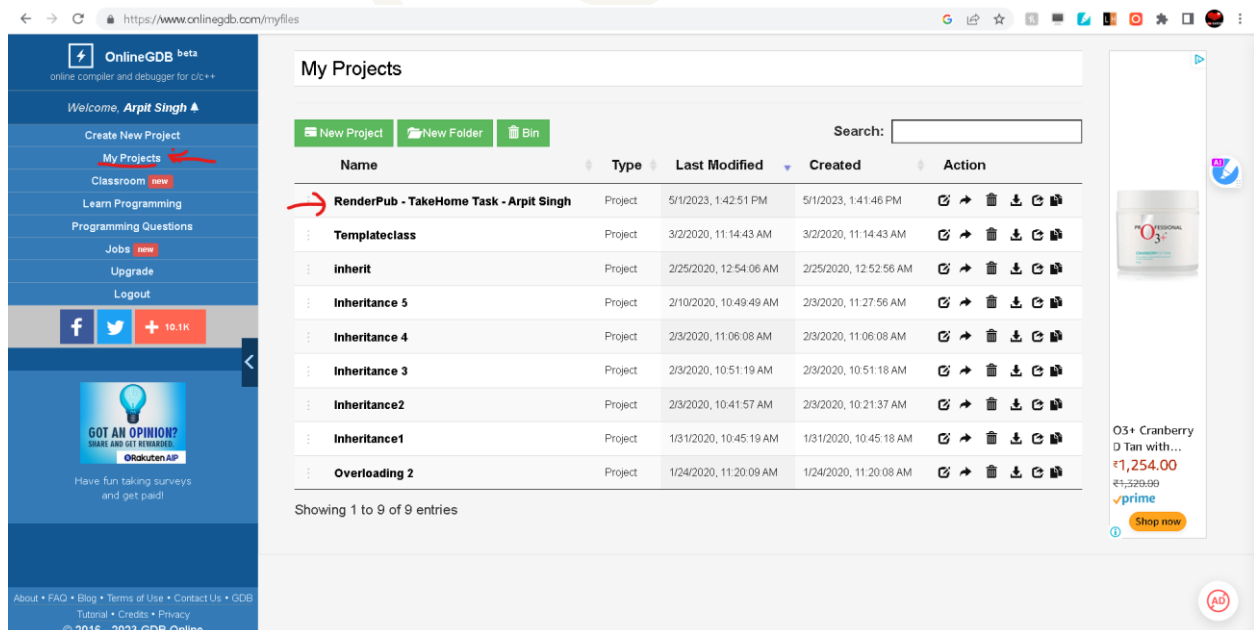
- b. **Sign Up and Log in** the website if you want to **save** your project. (Optional)



- c. **Create** a **new Project and Save** it. Give it a **name as per your preference** (Like – **RenderPub – TakeHome Task – Arpit Singh**).

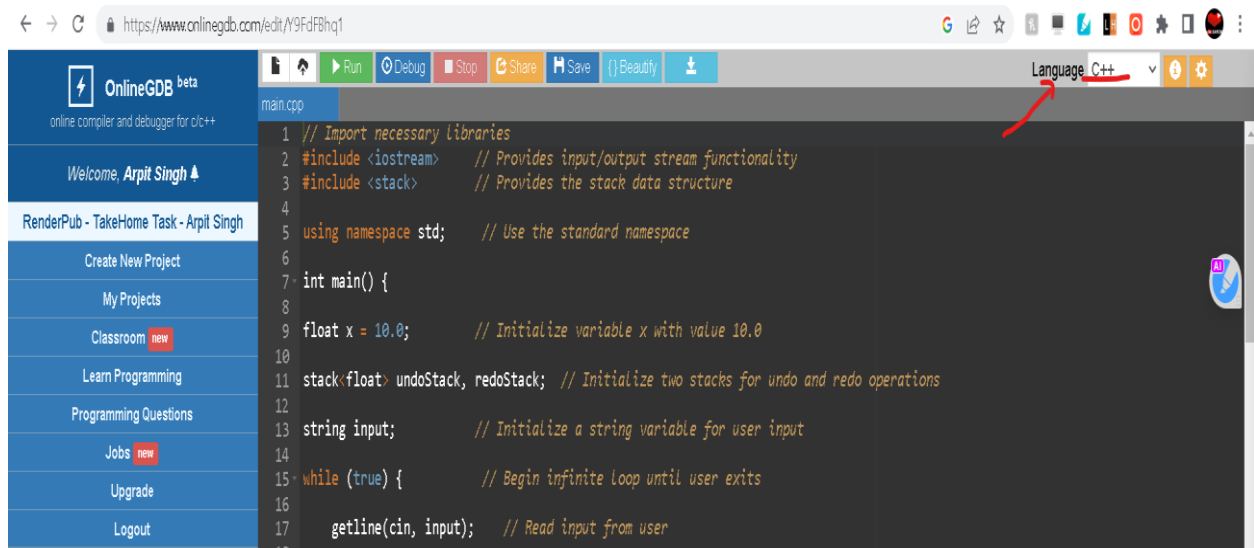


- d. Go to **My-Projects Section** available on the Left Menu and **Open the project** that you just created. (Like – **RenderPub – TakeHome Task – Arpit Singh**).



## 2. Write C++ Code:

- Make sure to set the **language as "C++"** of the compiler on the top right corner.



```
1 // Import necessary libraries
2 #include <iostream> // Provides input/output stream functionality
3 #include <stack> // Provides the stack data structure
4
5 using namespace std; // Use the standard namespace
6
7 int main() {
8
9     float x = 10.0; // Initialize variable x with value 10.0
10
11     stack<float> undoStack, redoStack; // Initialize two stacks for undo and redo operations
12
13     string input; // Initialize a string variable for user input
14
15     while (true) { // Begin infinite loop until user exits
16
17         getline(cin, input); // Read input from user
18     }
```

- Add the **Following Source Code:**



```
17         getline(cin, input); // Read input from user
18     }
19     if (input.empty() || input == "\x1B") { // Exit on empty input or Escape key press
20
21         break; // Exit loop and end program
22     }
23
24     else if (input == "x") { // If user inputs "x", print current value of x
25
26         cout << x << endl; // Output value of x to console
27     }
28
29     else if (input == "z") { // If user inputs "z", undo previous operation
30
31         if (!undoStack.empty()) { // Check if undo stack is not empty
32
33             redoStack.push(x); // Push current value of x to redo stack
34
35             x = undoStack.top(); // Set x to previous value from undo stack
36
37             undoStack.pop(); // Remove previous value from undo stack
38         }
```

```
main.cpp
37     undoStack.pop();           // Remove previous value from undo stack
38 }
39
40
41 else if (input == "y") {       // If user inputs "y", redo previous undo operation
42
43     if (!redoStack.empty()) {  // Check if redo stack is not empty
44
45         undoStack.push(x);      // Push current value of x to undo stack
46
47         x = redoStack.top();    // Set x to previous value from redo stack
48
49         redoStack.pop();       // Remove previous value from redo stack
50     }
51 }
52 else {                         // If user inputs an arithmetic operation, perform it on x and push to undo stack
53
54     float value = stof(input.substr(1)); // Convert the numeric value in the input string to a float
55
56     undoStack.push(x);         // Push current value of x to undo stack
57
58     switch (input[0]) {        // Check the operation specified by the first character in the input string
59
60         case '+':
61             x += value;        // Add the input value to x
62             break;
63
64         case '-':
65             x -= value;        // Subtract the input value from x
66             break;
67
68         case '*':
69             x *= value;        // Multiply x by the input value
70             break;
71
72         case '/':
73             x /= value;        // Divide x by the input value
74             break;
```

```
main.cpp
54     float value = stof(input.substr(1)); // Convert the numeric value in the input string to a float
55
56     undoStack.push(x);         // Push current value of x to undo stack
57
58     switch (input[0]) {        // Check the operation specified by the first character in the input string
59
60         case '+':
61             x += value;        // Add the input value to x
62             break;
63
64         case '-':
65             x -= value;        // Subtract the input value from x
66             break;
67
68         case '*':
69             x *= value;        // Multiply x by the input value
70             break;
71
72         case '/':
73             x /= value;        // Divide x by the input value
74             break;
75
76         default:                // If the first character is not a valid operation, output an error message
77             cout << "Invalid input." << endl;
78             break;
79     }
80
81     while (!redoStack.empty()) { // Clear redo stack after new operation
82
83         redoStack.pop();        // Pop values from the redo stack until it is empty
84     }
85 }
86
87 return 0; // End program
88 }
89
90 // code Written by @Arpit Singh - 19BCG10069
91
```

### Explanation (Code):

- This program uses **stacks** to keep track of **previous operations**, allowing for undo and redo functionality. The program takes **user input** in the form of **strings**, with the **first character representing the arithmetic operation** and the rest of the string representing the **value**. The program performs the specified **operation on the variable x**, and pushes the **previous value onto the undo stack** for later use. The program also **clears the redo stack** after a new operation is performed, as **redo is only allowed after undo**. The program **prints the current value of x** when the user inputs "**x**", **undoes the previous operation** when the user inputs "**z**", and **redoes the previous undo operation** when the user inputs "**y**". The program **exits on empty input** or the press of **the Escape key**.

### Algorithm (Code):

- This is a C++ program that implements an **Undo/Redo feature for arithmetic operations** on a variable. Here's how it works:
  1. First, the program initializes a float variable **x to 10.0**, and creates two stacks **undoStack and redoStack** to store **previous and future values** of x.
  2. The program enters an **infinite loop** where it waits for user input using the **getline function**. The input is stored in a **string variable input**.
  3. If the user enters an **empty input or presses the Escape key**, the loop is broken and **the program exits**.
  4. If the user **enters "x"**, the **current value of x** is printed to the console using the cout function.
  5. If the user **enters "z"**, **the previous value of x is retrieved from the undoStack** using the top and pop functions, and pushed onto the redoStack for future redo operations.

6. If the user **enters "y"**, the **previous value of x is retrieved from the redoStack** using the top and pop functions, and pushed onto the undoStack for future undo operations.
7. If the user enters **any other input**, the program checks the **first character of the input string to determine the arithmetic operation** to perform on x. The value to use in the operation is extracted from the input string using the substr function and converted to a float using the stof function.
8. The new **value of x is computed based on the operation and value**, and pushed onto the undoStack. The redoStack is cleared since a new operation has been performed.
9. The loop continues to wait for user input until the **user exits the program**.
- Overall, this program provides a simple way for users to **perform arithmetic operations on a variable and undo or redo previous operations** as needed.

c. **Save** the Source Code:

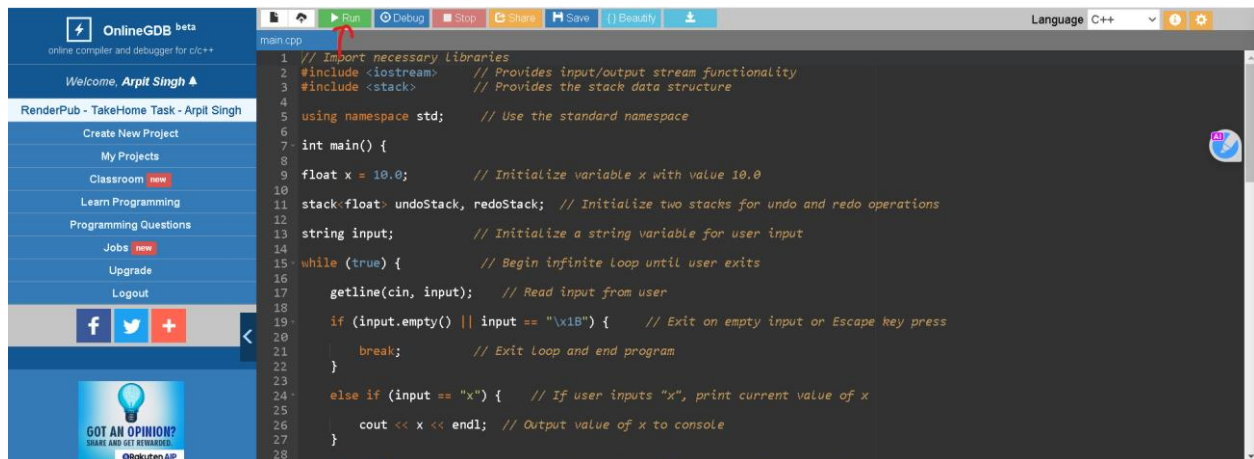
The screenshot shows the OnlineGDB web interface. The left sidebar contains navigation links like 'Welcome, Arpit Singh', 'RenderPub - TakeHome Task - Arpit Singh', 'Create New Project', 'My Projects', 'Classroom', 'Learn Programming', 'Programming Questions', 'Jobs', 'Upgrade', and 'Logout'. The main editor area displays C++ code for a calculator. The code includes comments for operations like subtraction, multiplication, and division, and uses `undoStack` and `redoStack` to manage previous states. A red arrow points to the 'Save' button in the top toolbar.

```

65     x -= value;          // Subtract the input value from x
66     break;
67
68     case '*':
69         x *= value;      // Multiply x by the input value
70         break;
71
72     case '/':
73         x /= value;      // Divide x by the input value
74         break;
75
76     default:              // If the first character is not a valid operation, output an error message
77         cout << "Invalid input." << endl;
78         break;
79     }
80
81     while (!redoStack.empty()) { // Clear redo stack after new operation
82
83         redoStack.pop();        // Pop values from the redo stack until it is empty
84     }
85 }
86
87 return 0; // End program
88 }
89
90 // code Written by @Arpit Singh - 19BCG10069
91

```

### 3. Run the Code:



The screenshot shows the OnlineGDB beta interface. On the left is a sidebar with a user profile 'Arpit Singh', navigation links like 'Create New Project', 'My Projects', 'Classroom', 'Learn Programming', 'Programming Questions', 'Jobs', 'Upgrade', and 'Logout', and social media icons. The main area displays a C++ program in a dark-themed editor. The code implements a calculator with undo and redo functionality. It uses a float variable 'x' initialized to 10.0, two stacks 'undoStack' and 'redoStack', and a string 'input'. A while loop reads user input and performs operations: adding 5, showing the value, subtracting 3, showing the value, multiplying by 7, showing the value, dividing by 2, showing the value, undoing the previous operation, redoing the previous operation, showing the value, and finally quitting on the 'Esc' key. The 'Run' button is highlighted with a red arrow.

```
1 // Import necessary libraries
2 #include <iostream> // Provides input/output stream functionality
3 #include <stack> // Provides the stack data structure
4
5 using namespace std; // Use the standard namespace
6
7 int main() {
8
9     float x = 10.0; // Initialize variable x with value 10.0
10
11     stack<float> undoStack, redoStack; // Initialize two stacks for undo and redo operations
12
13     string input; // Initialize a string variable for user input
14
15     while (true) { // Begin infinite loop until user exits
16
17         getline(cin, input); // Read input from user
18
19         if (input.empty() || input == "\\x1B") { // Exit on empty input or Escape key press
20             break; // Exit loop and end program
21         }
22
23         else if (input == "x") { // If user inputs "x", print current value of x
24             cout << x << endl; // Output value of x to console
25         }
26
27     }
28 }
```

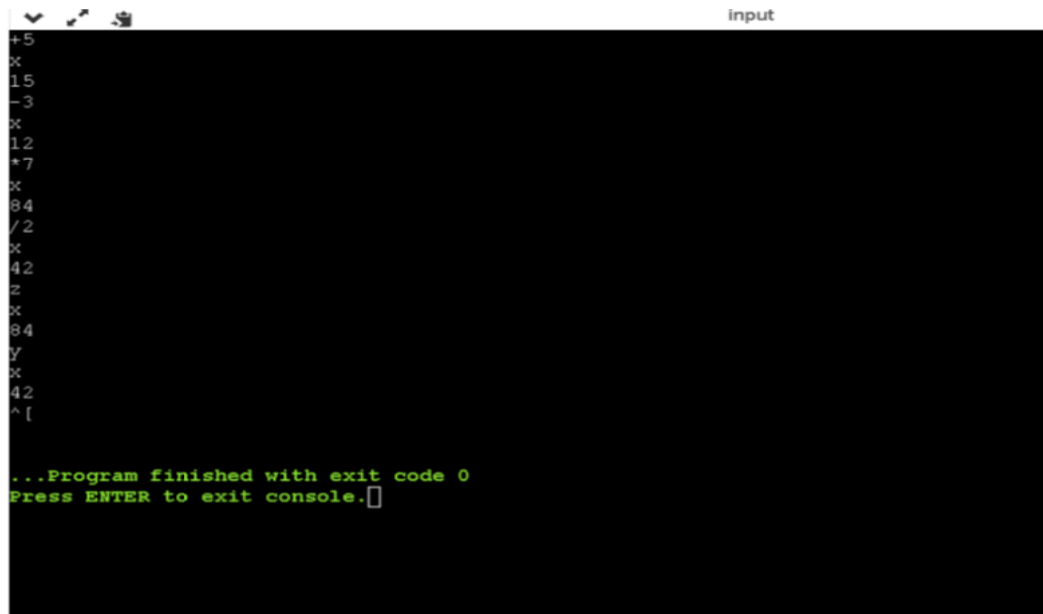
❖ That's it! We have successfully created a C++ program that implements the UNDO and REDO functionality as mentioned in our problem statement.

### Output:

- So, here *after performing each operation* I have used the “x” *command to display* that the functionality of the program is working fine.
- Flow of input is like:
  1. +5 → add 5 to the x value.
  2. X → show x value.
  3. -3 → subtract 3 from x value.
  4. X → show x value.
  5. \*7 → multiply 7 to x value.
  6. X → show x value.
  7. /2 → divide 2 from x value.
  8. X → show x value.
  9. Z → un-do the previous operation and set its last value.
  10. X → show x value.
  11. Y → re-do the previous operation and set the value after redoing the operation.
  12. X → show x value.
  13. Esc → quit the program.



- Screen Shot of O/p:



```
input
+5
x
15
-3
x
12
+7
x
84
/2
x
42
x
84
Y
x
42
^ [
...Program finished with exit code 0
Press ENTER to exit console.
```

## **Summary on how Create C++ program to implement Undo/Redo Functionality:**

*Here are the steps you can follow:*

1. Go to <https://www.onlinegdb.com/> and create a new Project. Set the language of compiler as C++.
2. Write the logic to implement Undo/Redo for a program that performs basic arithmetic operators
3. Run the code and test by providing input string.



## **Explanation on Tools/Technology(s) Used:**

### **❖ C++:**

- C++ is a commonly used programming language for implementing system software, performance-critical applications, and applications with high demands for memory efficiency. In this particular task, C++ is used to create a program that allows the user to perform arithmetic operations on a variable (float x) and undo/redo the operations.
- Using C++ provides several advantages in this case:
  - It allows for direct memory manipulation and fine-tuning, which can improve the program's performance.
  - C++ supports operator overloading, which makes it easier to define custom operators for arithmetic operations.
  - It provides standard libraries like stack, which can be used to implement undo/redo functionality efficiently.
  - C++ is a widely used language and has a large community and resources available for support.
  - Overall, C++ provides a good balance between performance, memory efficiency, and ease of use in this task.



### **❖ Github:**

- GitHub is a web-based hosting service that provides version control for software development projects. It allows you to create a repository to store your code, and provides tools for collaboration, documentation, and code management. In the context of this task, using GitHub has several benefits:
- In this project we have used GitHub to share the Source Code and document the procedure.
- Version Control: By using GitHub, you can keep track of all changes made to your code, and can easily revert back to previous versions if necessary. This is especially useful when multiple people are working

on the same code, as it helps to avoid conflicts and ensures that everyone is working with the latest version of the code.

- Collaboration: GitHub makes it easy to collaborate with others on a project. You can invite others to view or contribute to your code, and can manage access levels to ensure that only authorized individuals can make changes. This is especially useful for group projects, where multiple people are working on the same codebase.
- Documentation: GitHub provides tools for creating and maintaining documentation for your code. You can create a wiki, write documentation in the form of README files, and add comments to your code to help others understand how it works. This is especially useful for open-source projects, where anyone can contribute to the codebase.
- Code Management: GitHub provides a range of tools for managing your code, including issue tracking, pull requests, and code reviews. These tools can help you to keep track of bugs and feature requests, and ensure that changes to the code are reviewed and approved by others before being merged into the main branch.
- Overall, using GitHub provides a range of benefits for software development projects, including version control, collaboration, documentation, and code management.



#### ❖ **OnlineGDB (IDE):**

- The purpose of using OnlineGDB in this task is to have a convenient and accessible online Integrated Development Environment (IDE) for writing, compiling, and executing C++ code. With OnlineGDB, there is no need to install any software or set up a local development environment on the computer. It provides a user-friendly interface with features such as syntax highlighting, code debugging, and output visualization.

- Additionally, OnlineGDB allows for easy sharing and collaboration of code through a unique link that can be shared with others. This is especially useful for remote work or collaboration on group projects. The online platform also allows for easy access to previous versions of code and the ability to revert changes if needed. Overall, using OnlineGDB provides a convenient and efficient way to write and execute C++ code for this task.



**Thank You!**

---