International Journal of Scientific Research in Science and Technology
Print ISSN: 2395-6011 | Online ISSN: 2395-602X (www.ijsrst.com)
doi : https://doi.org/10.32628/IJSRST229149

The C Language Mini Project to Demonstrate a Simple Progress Bar

Sakshi Satish Karanjkar, Dipali Mane, Balaji Chaugule
Department of Computer Engineering, Zeal College of Engineering and Research, Pune, Maharashtra, India

Article Info
Volume9, Issue 2
Page Number: 448-452

Publication Issue
March-April-2022

Article History
Accepted :03March2022
Published :10March2022

ABSTRACT

Creating a simple loader progress bar using some basic functions and loops in the C programming language. This mini-project makes use of basic C concepts and datatypes to print a pattern in such a way that with proper delay insertion, the end result is a progress bar that indicates the loading process of any task. In the early days, when graphics were not as appealing and fast as today, we used to have two folders on the screen separated by some distance. While our files were being copied, a piece of paper would fly off from the left folder and get inserted into the right folder. Gradually, that pattern of representation was replaced by the progress bar.

KEYWORDS: Core C Language, While Loop, Color Functions, Goto Function, Programming Plane

I. INTRODUCTION

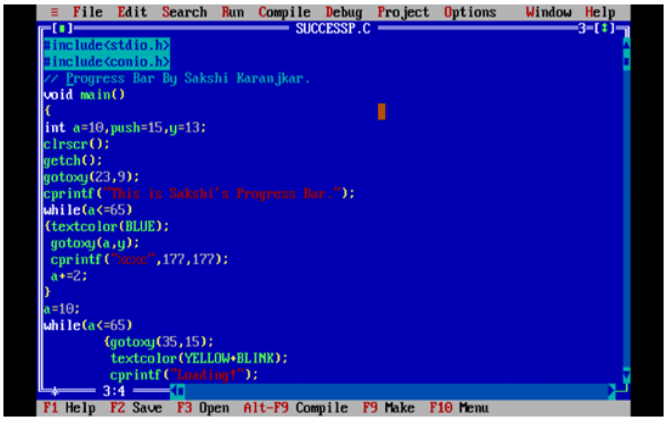
The C programming language is like a vast “sea”. The founder of this programming language is Dennis Ritchie. He has himself said that, even though he has been a pioneer of giving the gift of C to this world, he knows less than 10% of it. C language was originally created to overcome the drawbacks of B and BCPL languages. Though C is an old language, it is still regarded as one of the core fundamental programming languages. Other languages have received help from C, the tools required for problem solving in other programming languages have been provided by C. This was a brief history of C programming language.

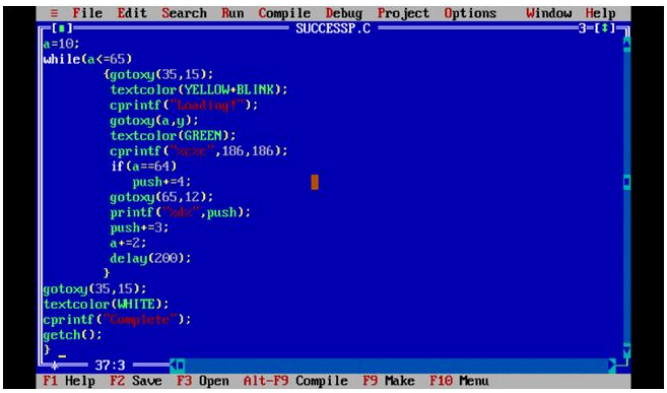
II. CORE C PROGRAMMING LANGUAGE

If a person wants a complete walkthrough of C language in a gist, here are the majority of the concepts mentioned, which are covered by the C programming language. First of all, the programmer must know how to write a pseudo code. It includes steps to approach the program in sentence format. It may also include flowcharts. Next, one must learn sequence which is nothing but proper sequential steps to write a correct program. Then comes selection which is a collection of a few conditional statements like if...else, multiple if, etc. It is followed by Iteration meaning loops or looping constructs. We mainly have while, do-while, for, and nested for loops in C language. To create menu-driven programs, the switch case concept comes into picture. To organize

records and databases as we do in stationary registers, the next useful concept of arrays and string arrays is implemented. The essence of C and CPP is pointers, a very important concept that follows arrays. Next, we have functions as a topic. By default, C has many in-built functions like printf, scanf, getch, clrscr, strcpy, etc. But we can also build our own functions in C. For e.g. If we want prime numbers to be determined, if we want factorial of a number, if we want to determine whether a number or a string is a palindrome or not, etc. The last three topics are nothing but structure, preprocessor and file handling which lay the foundations of any C mega-project. This was a brief explanation about what C actually does.

III. OVERVIEW OF EVERYTHING USED IN PROGRESS BAR PROJECT





The above two images represent the C code of the progress bar. In the first two lines, we have the hash include statements. Hash is used as a pre-processor. Whatever is written after the hash, which is usually a header file, will be pre-complied by the compiler to allow us the use of all the functionality that those header files provide. The two files have a .h extension to indicate that they are header files. Among those two header files included, stdio stands for "standard input/output" and conio stands for "Console input/output". Next, we have a text called //Progress Bar by Sakshi, which is a single-line comment used in C. Like all the other programming languages, comments will not be executed. They are just for the programmer's reference. Then, we have void main(). Here, void is the return-type of main() which is a function, main() function plays a crucial role in executing the program and terminating it. Void means nothing, we just want to display the progress bar on our console, we don't expect the function to return any value back. That is the reason void is used as the return type. The main function has a method in which we write the actual code, the space between two curly braces is called as that method.

Now we begin with the actual logic. The first step in the method is variable declaration. Variables are nothing but containers to hold a value depending upon their datatype. Here, we have the int datatype to store integer values which we have assigned to the variables a, push and y respectively. There are two functions clrscr(); and getch(); after variable declaration. Clrscr function is used to clear the screen for fresh outputs every time we compile and run the program. The output of the correct code is achieved but sometimes it gets displayed and the compilers returns to the main blue window so fast that we are unable to comprehend the output clearly. To make the compiler wait, show us the output and return to blue window only after pressing any key, getch function is used. Now, we have arrived at something called as the "gotoxy();" function. According to the applied co-ordinate geometry in C programming, the black output console is an X-Y Programming Plane with bottom of the screen being the X-axis and left side of the screen being the Y-axis. Naturally, the origin will be at the bottom-left corner of the console. In gotoxy(23,19), value of x-coordinate is 23 and that

of y-coordinate is 19. The cursor will hence move to that location on the plane. Before the while loop, the last statement is cprintf. The difference between cprintf and printf is that cprintf supports textcolor and textbackground properties whereas printf does not. The message written in cprintf will be printed as it is on the output console, by default in white textcolor.

IV. WHILE LOOP (TO PRINT A BASE DOTTED BAR OF BLUE COLOUR)

The while loop has the condition as while(a<=65). The initial value of a is 10 which is incremented by 2 using a statement in the loop as a+=2. When the value of a becomes greater than 65, while loop will be terminated. For every value between 10 and 65 which makes the while loop condition true, for as long as the loop will get executed, everything written in that loop will get executed as well. Text colour will be blue, gotoxy(a,y) will result in multiple cursor locations e.g. (10, 13) then (12,13) so on till (64,13). At each of these locations, a small part of the base bar will get printed in blue. Just because we have not inserted a delay, the entire bar is visible instantly. The way the bar is getting printed is because of cprintf function where %c and %c are two placeholders of character datatype and 177 is the ASCII value of the dotted block.

The above image shows the base blue bar printed using this while loop.

V. WHILE LOOP (TO PRINT THE FLOWING GREEN BAR INDICATING THE LOADING PROCESS)

First, let us understand which tasks we are accomplishing using this second while loop and then proceed accordingly. It is to be noted that this second while loop will not get executed until the first while loop is terminated as this loop is below the first loop in the program. First task of the second while loop is to create a loading message below the progress bar to indicate that the process is still running. The second task is to print a flowing progress bar of green colour over the previous blue dotted one. The length of the blue dotted bar and the green-lined bar will be the same. The third and the last task is to generate a percent count on the top-right corner of the previous blue bar to indicate the percent of information loading. The percent will start from 15% till 100% and overwrite on the same location.

Now, as you can see in the second image containing the code of the program, the second while loop has the same condition as the first while loop. But, the value of variable 'a' is now greater than 65 when it exited the first while loop. We want the length and position of both bars to be the same, hence, we need to bring 'a' back to 10 and increment it till 65 again. So, we write a=10 before the second while loop. Now in the second while loop, to achieve the first task, we do gotoxy(35,15) which will bring the cursor below our progress bar. Then using textcolor(YELLOW+BLINK) and cprintf function, we have created "Loading!" message below the bar which will be in yellow colour and will keep blinking till the while loop terminates. To achieve the second task, we have used the bar printing conditions from the first while loop as it is with some minor changes. We have kept the location same, just changed the colour of the bar and the ASCII value from 177 to 186 to print line bar. Also, because we have inserted delay(200), we will not get a complete bar at once but parts of it after regular intervals of time which is the entire essence of

this project. The third task of printing a percent count on top of the bar can be done by initializing the value of a push variable to 15, then incrementing it using push+=3, such that when the entire green bar is loaded, percentage count should be showing 100%. But, it might happen that the count may not reach 100 but some value before or after 100. So, to handle this condition, just before the loop terminates i.e at a=64, we use selection that is the if statement to increase the push value such that it becomes 100 before exiting the second loop. [1] After second loop is terminated, we again go to the location of 'Loading!' and overwrite it with 'Complete' to indicate completion of our task.

Finally, the pictures of how our project bar looks like are below:

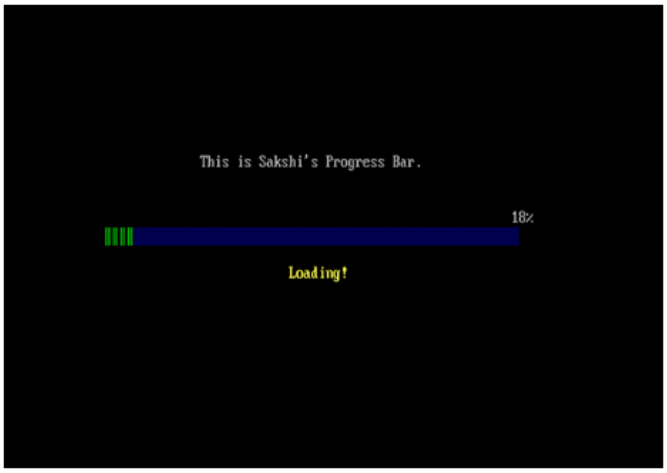


Fig 1: At 18% with Loading Message

Fig 2: At 30% without Loading Message because of Blink

Fig 1 has green bar filling the blue one only till 18%. It also has yellow colored text that is the Loading message to indicate that the bar is still flowing. In Fig 2, the yellow Loading message seems to have disappeared but it hasn't. The blink property causes it to blink until the loop finishes executing. Fig 3, Fig 4 and Fig 5 indicate the stages of progress bar completion. The last image clearly shows the Loading message being replaced by Complete message and bar percent count to be 100%

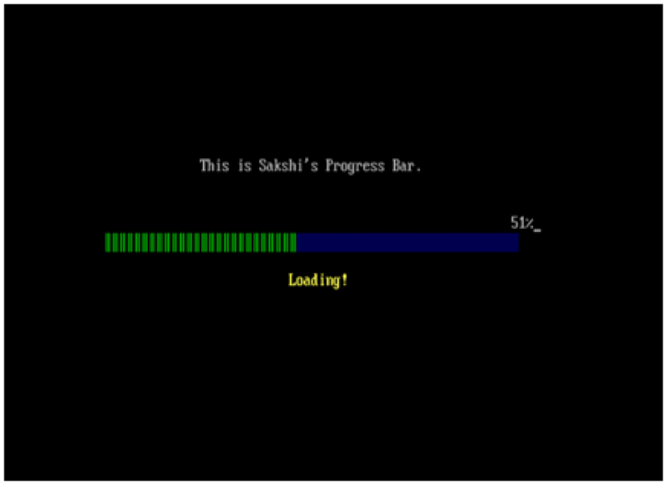


Fig 3: At 51% with Loading Message

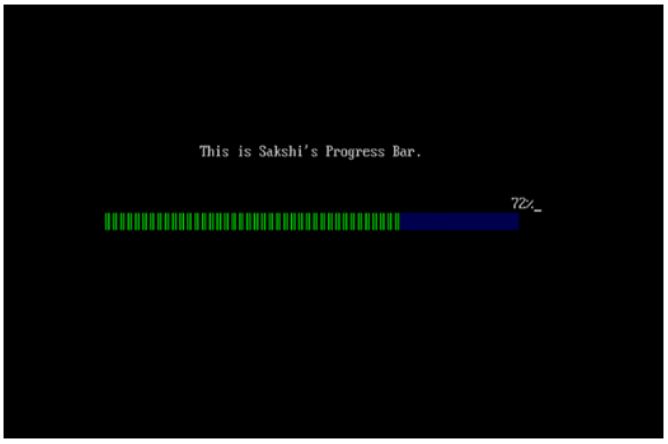


Fig 4: At 72% without Loading Message

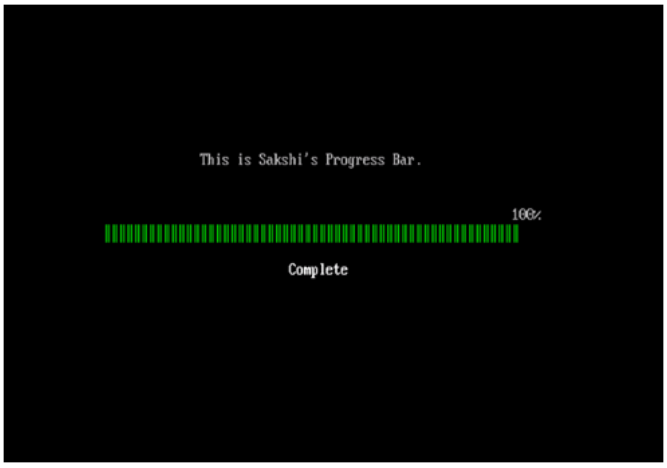


Fig 5: At 100% with Complete Message

VI. CONCLUSION

Thus, a progress bar using C programming language has been explained in this research paper. Some basic functionalities of C language have also been explained. The project is demonstrated with appropriate illustrations and images.

VII. REFERENCES

[1]. The C Programming Language Research Paper, Second Edition.