**FORMAN CHRISTIAN COLLEGE (A CHARTERED UNIVERSITY)**

****

**COMP 451 (Compiler Construction)**

**2022 FALL**

**Lab # 4**

**Gulraiz Noor Bari (231-525536)**

**Muhammad Hamza (231-450349)**

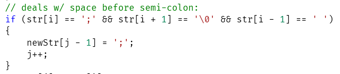
**INTRODUCTION:**

* **stdio.h** (standard input/output)is a header file that contains declarations for functions like, printf, scanf, etc.
* **stdlib.h** (standard library) is a header file that contains declarations of functions that involves memory allocation and process control. In our program we used:
  + **exit(0)**: which means to successfully terminate the program
  + **malloc():** which is referred as Dynamic Memory Allocation is used to dynamically allocate a single large block of memory of a specified size. It returns a pointer of void type which can be casted to any other datatype.
* **string.h** is a header file that contains declarations of functions that are used for working with strings. In our program we have used **strlen** function to get the length of string.
* **main():** The main function serves as the starting point of the program execution in C language. User can pass any number of parameters depending upon the requirements of the program logic or structure. We have passed two parameters:
* **int argc:** Keeps count of the number of command-line arguments entered by the user that also includes the name of the program
* **char \* argv[]:** Here argv (stores command-line arguments) is an array of pointers that is pointing to the characters of the command-line arguments entered by the user
* **MAX** is a macro (user defined variable) that is assigned value 100

**LOGIC/ALGORITHM:**

The code is a program that reads a string entered by the user, identifies and removes all the unnecessary spaces in the instruction string and displays the updated string on the command-line.

* The program starts by checking whether two arguments were passed by the user in command line, which is done by **argc** (which has the count of arguments entered in the command-line). If the number of arguments is not two the function is not proceeded and displays a message “Invalid argument”, otherwise if user has entered two arguments; in our case the executable and file name that is to be read, the program proceeds.
* Two **char** datatype pointers are initialized:
* **char \*str:** Takes and stores the string (from argv[1]), entered by the user in command-line
* **char \*newStr:** Stores a pointer to a memory block of size 100 (in our case) or any other specified by the user
* A for-loop runs from 0 till the length of **str** (input string), in this loop first it initiates a while-loop that runs till the string terminator symbol **“\0”** is detected at the end of **str** (input string). Following checks and operations are performed in the while-loop:
  + First it checks at every index **“i”** of str for a space, if it finds one then it adds a single space in **newStr** (output string) at the same index as **str**, using **j** iterator and moves **j** iterator on the next index. Now to ignore more than one consecutive space it iterates on next indices in a while-loop and till it encounters spaces it keeps on incrementing **i** iterator.
  + In the next if-statement it checks for a space before the semi-colon that ends the instruction line. It checks if the current index **i** of the **str** (original string) is equal to semi-colon and there is empty space on left and to verify if it is the ending semi-colon of the instruction line we check if it has a **“\0”** (string terminating symbol) on its right. If all these conditions are **True**, then itshifts the position of semi-colon to the left in **newStr** (output string) and increments **j** iterator by 1



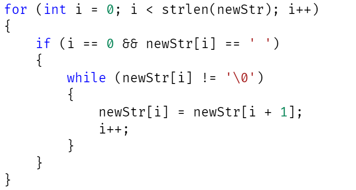
* After both conditions are checked then the valid character present at the index **i** of **str** (input string) is copied to the current index **j** of **newStr** (output string) and **i** and **j** iterators are incremented by 1.

This process repeats till the string terminating symbol **“\0”** is encountered and the loop breaks and adds a string terminating symbol at the end index **(j)** of the **newStr** (output string).

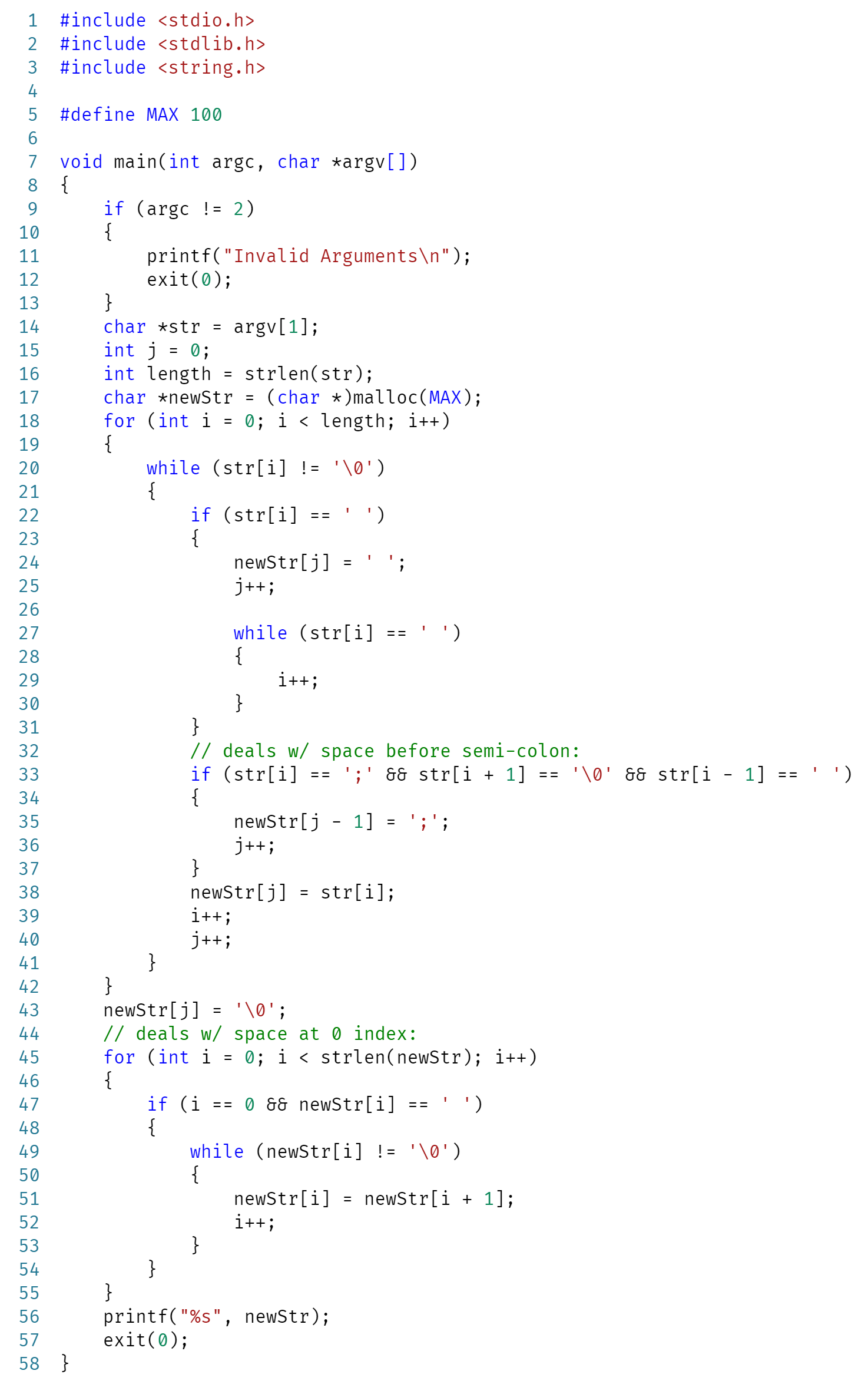
* But there is a space at the 0th index of **newStr** (output string) which is removed by:
  + For-loop that runs starting from 0 index till length of **newStr,** and checks:
  + If iterator **i** is on 0th index and there is a space at 0th index, then initiate a while-loop till it doesn’t encounter **“\0”** and in the loop at every ith index of **newStr** offset it by +1, which will be the next index. So,

**“ int myVal = 200;”**

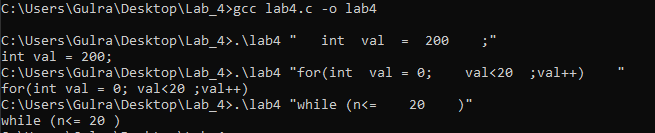
Becomes

 **“int myVal = 200;”**

* After this for-loop ends program prints the **newStr** on the console using **“%s”** (string formatter) and terminates the program successfully using **exit(0)**.

**CODE:**

**OUTPUT:**

****