1. Write a program to find the length of a string without using strlen().

**Input**

* Reads a full line of input from the user (including spaces).
* Stops at newline or when 99 characters are read.

**Process**

* Loops through each character.
* Stops when it reaches the **end of the string** ('\0') or a **newline** ('\n'), which fgets() includes.

**Output**

* Displays the final length calculated.

#include <stdio.h>

void main()

{

char str[] = "Hello, World!";

int length = 0;

while (str[length] != '\0')

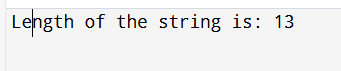
{

length++;

}

printf("Length of the string is: %d\n", length);

}



1. Write a program to copy one string to another.

|  |  |
| --- | --- |
| **Input** | Read a string from the user (source) |
| **Process** | Copy each character from source to destination |
| **Output** | Print both source and destination strings |

#include <stdio.h>

int main()

{

char source[100], destination[100];

printf("Enter a string: ");

fgets(source, sizeof(source), stdin);

int i = 0;

while (source[i] != '\0')

{

destination[i] = source[i];

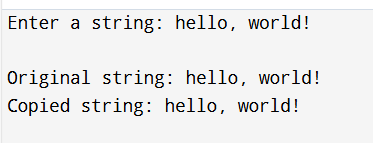
}

destination[i] = '\0';

printf("\nOriginal string: %s", source);

printf("Copied string: %s", destination);

}



1. Write a program to concatenate two strings.

**Input:** The user enters two strings.

**Process:**

* 1. First string (str1) is copied into result.
  2. Second string (str2) is appended after that.

fgets() is used to allow space-containing input.

i-- is used to overwrite the newline character inserted by fgets() from str1.

#include <stdio.h>

void main()

{

char str1[100], str2[100], result[200];

int i = 0, j = 0;

printf("Enter first string: ");

fgets(str1, sizeof(str1), stdin);

printf("Enter second string: ");

fgets(str2, sizeof(str2), stdin);

while (str1[i] != '\0')

{

result[i] = str1[i];

i++;

}

i--;

while (str2[j] != '\0')

{

result[i] = str2[j];

i++;

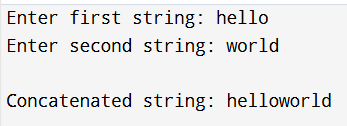
j++;

}

result[i] = '\0';

printf("\nConcatenated string: %s", result);

}



1. Write a program to compare two strings.

**Input**

* Two strings entered by the user.
  + Example:
    - First string: "hello"
    - Second string: "hello”

**Process**

* The program reads the two input strings.
* Compares the two strings character by character (using == in Python or strcmp in C).
* Checks if both strings are exactly the same or not.

**Output**

* Displays whether the two strings are equal or not.
  + If equal: "The strings are equal."
  + If not equal: "The strings are not equal."

#include <stdio.h>

Void main()

{

char string1[100], string2[100];

printf("Enter the first string: ");

fgets(string1, sizeof(string1), stdin);

printf("Enter the second string: ");

fgets(string2, sizeof(string2), stdin);

string1[strcspn(string1, "\n")] = '\0';

string2[strcspn(string2, "\n")] = '\0';

if (strcmp(string1, string2) == 0)

{

printf("The strings are equal.\n");

}

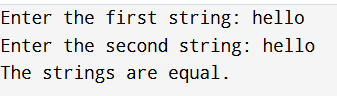
Else

{

printf("The strings are not equal.\n");

}

}



1. Write a program to count vowels and consonants in a string.

**Input:** A string from the user

**Process:** Traverse the string and count vowels and consonants

**Output:** Number of vowels and consonants

#include <stdio.h>

void main()

{

char str[100];

int vowels = 0, consonants = 0;

printf("Enter a string: ");

fgets(str, sizeof(str), stdin);

for (int i = 0; str[i] != '\0'; i++)

{

char ch = tolower(str[i]);

if (ch >= 'a' && ch <= 'z')

{

if (ch == 'a' || ch == 'e' || ch == 'i' ||

ch == 'o' || ch == 'u')

{

vowels++;

}

Else

{

consonants++;

}

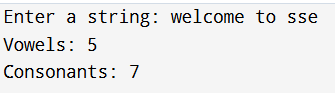
}

}

printf("Vowels: %d\n", vowels);

printf("Consonants: %d\n", consonants);

}



1. Write a program to convert lowercase to uppercase and vice versa.

|  |  |
| --- | --- |
| **Input** | - A string entered by the user (can contain letters, numbers, spaces, symbols) |
|  | Example: "Hello World 123" |
| **Process** | - Loop through each character in the string |
|  | - If a character is lowercase (a–z), convert it to uppercase (A–Z) |
|  | - If a character is uppercase (A–Z), convert it to lowercase (a–z) |
|  | - If it's not a letter (e.g., number, space, punctuation), leave it unchanged |
| **Output** | - A new string with case of all letters flipped |
|  | Example Output: "hELLO wORLD 123" |

#include <stdio.h>

void main()

{

char str[100];

int i;

printf("Enter a string: ");

fgets(str, sizeof(str), stdin);

for (i = 0; str[i] != '\0'; i++)

{

if (islower(str[i]))

{

str[i] = toupper(str[i]);

}

else if (isupper(str[i]))

{

str[i] = tolower(str[i]);

}

}

printf("Converted string: %s", str);

}

1. Write a program to check if a string is palindrome.

|  |  |
| --- | --- |
| **Input** | A string entered by the user |
| **Process** | 1. Remove newline if present   2. Compare characters from start and end moving towards the center   3. Ignore case and non-alphanumeric characters (optional)   4. Determine if the string is a palindrome |
| **Output** | A message indicating whether the string is a palindrome or not |

#include<stdio.h>

void main()

{

char str[100];

printf("Enter a string: ");

fgets(str, sizeof(str), stdin);

size\_t len = strlen(str);

if (len > 0 && str[len - 1] == '\n')

{

str[len - 1] = '\0';

}

if (isPalindrome(str))

{

printf("The string is a palindrome.\n");

}

Else

{

printf("The string is not a palindrome.\n");

}

}

1. Write a program to reverse a string.

**🔹 Input**

* Read a string from the user using fgets().

**🔹 Processing**

* Remove the newline character (if any).
* Use a loop to reverse the string in place by swapping characters.

**🔹 Output**

* Display the reversed string using printf().

#include <stdio.h>

void main()

{

char str[100], temp;

int i, j;

printf("Enter a string: ");

fgets(str, sizeof(str), stdin);

str[strcspn(str, "\n")] = '\0';

int len = strlen(str);

for (i = 0, j = len - 1; i < j; i++, j--)

{

temp = str[i];

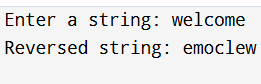
str[i] = str[j];

str[j] = temp;

}

printf("Reversed string: %s\n", str);

}



1. Write a program to count words in a string.

|  |  |  |
| --- | --- | --- |
| **Input** | Read a string from the user |
| **Processing** | Loop through the string and count word  boundaries |
| **Output** | Display the total number of words |

#include <stdio.h>

void main()

{

char str[100];

int i, wordCount = 0;

int inWord = 0;

printf("Enter a string: ");

fgets(str, sizeof(str), stdin);

str[strcspn(str, "\n")] = '\0';

for (i = 0; str[i] != '\0'; i++)

{

if (isspace(str[i]))

{

inWord = 0; // We're outside a word

}

else if (inWord == 0)

{

inWord = 1; // We've found the start of a word

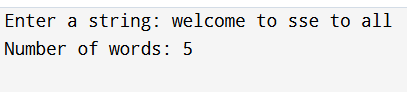
wordCount++; // Count the word

}

}

printf("Number of words: %d\n", wordCount);

}



10. Write a program to find the frequency of each character in a string.

|  |  |
| --- | --- |
| **Input** | :Read a string from the user |

|  |  |  |
| --- | --- | --- |
| **Processing** | :Loop through each character and count its frequency |  |

|  |  |
| --- | --- |
| **Output** | :Display each character and its frequency (if frequency > 0) |

#include <stdio.h>

void main()

{

char str[1000];

int freq[256] = {0};

printf("Enter a string: ");

fgets(str, sizeof(str), stdin);

str[strcspn(str, "\n")] = '\0';

for (int i = 0; str[i] != '\0'; i++)

{

freq[(unsigned char)str[i]]++;

}

printf("\nCharacter Frequencies:\n");

for (int i = 0; i < 256; i++)

{

if (freq[i] > 0)

{

printf("'%c' = %d\n", i, freq[i]);

}

}

}

