**Objectives:**

1. Strings
   1. To become familiar with the concept of Strings.
   2. To learn how strings are implemented.
   3. To learn how to implement strings with file I/O functions

**STRINGS IN C++**

For this purpose, we need to include a library ‘cstring’ as follows.

|  |
| --- |
| #include <cstring> |

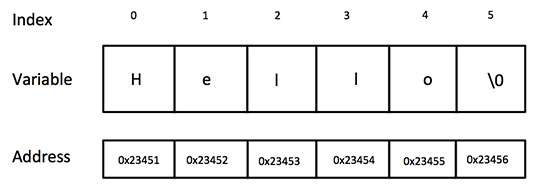
A string is a one-dimensional array of characters which is terminated by a null character '\0'. Thus, a null-terminated string contains the characters that comprise the string followed by a null. The following declaration and initialization create a string consisting of the word "Hello". To hold the null character at the end of the array, the size of the character array containing the string is one more than the number of characters in the word "Hello."

|  |
| --- |
| char greeting[6] = {'H', 'e', 'l', 'l', 'o', '\0'}; |

If you follow the rule of array initialization, then you can write the above statement as follows:

|  |
| --- |
| char greeting[] = "Hello"; |

Following is the memory presentation of above defined string in C/C++

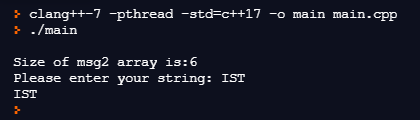


It should be noted that you do not place the null character at the end of a string constant. The C++ compiler automatically places the '\0' at the end of the string when it initializes the array.

**STRINGS AS ARRAY OF CHARACTERS**

**Example 1: Declaring and initializing strings**

|  |
| --- |
| #include <iostream>  using namespace std;  int main()  {    // Note the null character at end of definition    char msg1[9] = {'M','e','s','s','a','g','e',':','\0'};    // Following definition automatically adds null character at end    char msg2[] = "Hello";  cout<<endl<<"Size of msg2 array is:"<<sizeof(msg2)<<endl;    char userip[10];    cout<<"Please enter your string: ";  // cin also automatically adds null character at end    cin>>userip;    cout<<userip<<endl;  } |



**FUNCTIONS**

C++ supports a wide range of functions that manipulate null-terminated strings.

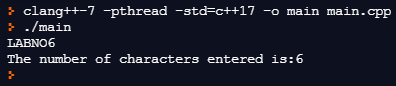
|  |  |  |
| --- | --- | --- |
| **S/No.** | **Function** | **Purpose** |
| 1 | strcpy(s1, s2); | Copies string s2 into string s1. |
| 2 | strcat(s1, s2); | Concatenates string s2 onto the end of string s1. |
| 3 | strlen(s1); | Returns the length of string s1. |
| 4 | strcmp(s1, s2); | Returns 0 if s1 and s2 are the same; less than 0 if s1<s2; greater than 0 if s1>s2. |
| 5 | strchr(s1, ch); | Returns a pointer to the first occurrence of character ch in string s1. |
| 6 | strstr(s1, s2); | Returns a pointer to the first occurrence of string s2 in string s1. |

**LENGTH OF STRING**

While you may define a very large character array, it might only contain very few characters. For example, suppose you define a character array of length 250 for storing user input. If the user inputs “Hello”, only the first 6 array elements (including null) will be used. It is often useful to be able to find the number of characters in the array. This can be done using strlen() function defined in cstring as shown below.

**Example 2: Length of String**

|  |
| --- |
| #include <iostream>  #include <cstring>  using namespace std;  int main()  {  char userip[250];  cin>>userip;  cout<<"The number of characters entered is:"<<strlen(userip)<<endl;  } |

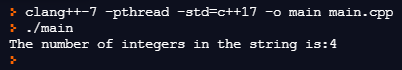


**CHARACTER COMPARISON AND ARITHMETIC**

It is often useful to be able to figure out what type of character is input by the user. For example, you may want to know whether a character is an alphabet or a number. For this purpose, one possible way is to compare individual characters in the character array. The ASCII table is shown at the end of this document. When comparing characters, keep in mind that the decimal values corresponding to characters are used. For example, character ‘a’ has ASCII value 97 and ‘A’ has ASCII value 65. This means that ‘a’ > ‘A’ will return true.

**Example 3: Counting number of numerical digits in string**

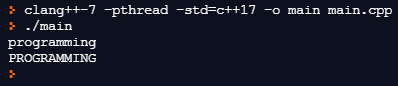
|  |
| --- |
| #include <iostream>  using namespace std;  int main()  {  char msg[9] = {'a','1','b','2','c','3','d','4','\0'};  int count = 0;  for(int i = 0; i < 9; i++)  {  if(msg[i] > 47 && msg[i] < 58)  {  count++;  }  }  cout<<"The number of integers in the string is:"<<count<<endl;  } |

****

Similarly, adding or subtracting integer values from characters results in new characters depending upon ASCII values. For example, it can be seen in ASCII table that the ASCII value of ‘a’ and ‘A’ differs by 32. Hence adding 32 in ‘A’ will yield the character ‘a’. Following simple program shows how user input can be transformed into uppercase.

**Example 4: Converting lowercase into uppercase**

|  |
| --- |
| #include <iostream>  #include <cstring>  using namespace std;  int main()  {  char userip[250];  cin>>userip;  for(int i = 0; i < strlen(userip); i++)  {  userip[i] = userip[i]-32;  }  cout<<userip<<endl;  } |

****

**ASCII Table**

Dec Char Dec Char Dec Char Dec Char

--------- --------- --------- ----------

0 NUL (null) 32 SPACE 64 @ 96 `

1 SOH (start of heading) 33 ! 65 A 97 a

2 STX (start of text) 34 " 66 B 98 b

3 ETX (end of text) 35 # 67 C 99 c

4 EOT (end of transmission) 36 $ 68 D 100 d

5 ENQ (enquiry) 37 % 69 E 101 e

6 ACK (acknowledge) 38 & 70 F 102 f

7 BEL (bell) 39 ' 71 G 103 g

8 BS (backspace) 40 ( 72 H 104 h

9 TAB (horizontal tab) 41 ) 73 I 105 i

10 LF (NL line feed, new line) 42 \* 74 J 106 j

11 VT (vertical tab) 43 + 75 K 107 k

12 FF (NP form feed, new page) 44 , 76 L 108 l

13 CR (carriage return) 45 - 77 M 109 m

14 SO (shift out) 46 . 78 N 110 n

15 SI (shift in) 47 / 79 O 111 o

16 DLE (data link escape) 48 0 80 P 112 p

17 DC1 (device control 1) 49 1 81 Q 113 q

18 DC2 (device control 2) 50 2 82 R 114 r

19 DC3 (device control 3) 51 3 83 S 115 s

20 DC4 (device control 4) 52 4 84 T 116 t

21 NAK (negative acknowledge) 53 5 85 U 117 u

22 SYN (synchronous idle) 54 6 86 V 118 v

23 ETB (end of trans. block) 55 7 87 W 119 w

24 CAN (cancel) 56 8 88 X 120 x

25 EM (end of medium) 57 9 89 Y 121 y

26 SUB (substitute) 58 : 90 Z 122 z

27 ESC (escape) 59 ; 91 [ 123 {

28 FS (file separator) 60 < 92 \ 124 |

29 GS (group separator) 61 = 93 ] 125 }

30 RS (record separator) 62 > 94 ^ 126 ~

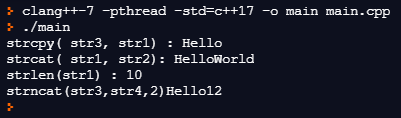
31 US (unit separator) 63 ? 95 \_ 127 DEL

**APPENDING/ COPYING TO A STRING**

There are several useful functions for string manipulation defined in cstring. For example, for appending one string with another string, it provides strcpy(), strcat() and strncat() functions. Example usage is shown in following example.

**Example 5: String copy and concatenation**

|  |
| --- |
| #include <iostream>  #include <cstring>  using namespace std;  int main()  {  char str1[10] = "Hello";  char str2[10] = "World";  char str3[10];  char str4[] = {"123"};  int len ;  // copy str1 into str3  strcpy( str3, str1);  cout << "strcpy( str3, str1) : " << str3 << endl;  // concatenates str1 and str2  strcat( str1, str2);  cout << "strcat( str1, str2): " << str1 << endl;  // total length of str1 after concatenation  len = strlen(str1);  cout << "strlen(str1) : " << len << endl;  // concatenates first two characters of str4 into str3  strncat(str3, str4, 2);  cout<< "strncat(str3,str4,2)"<<str3<<endl;  } |

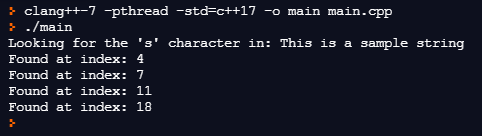


**SEARCHING IN STRINGS**

It is often required to search for characters or sub-strings within a string. For this purpose, cstring provides several functions. The following examples show how this can be done.

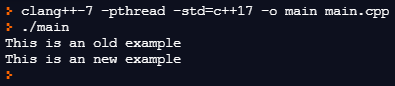
**Example 6: Locating first occurrence of a character**

|  |
| --- |
| #include <iostream>  #include <cstring>  using namespace std;  int main()  {  char str[] = "This is a sample string";  char \* pch;  cout<< "Looking for the 's' character in: "<< str<<endl;  pch=strchr(str,'s');  while (pch!=NULL)  {  cout<< "Found at index: " << pch-str+1<<endl;  pch=strchr(pch+1,'s');  }  } |

****

**Example 7: Searching and replacing a substring**

|  |
| --- |
| #include <iostream>  #include <cstring>  using namespace std;  int main()  {  char str[] ="This is an old example";  cout<<str<<endl;  char \* pch;  pch = strstr(str,"old");  strncpy (pch,"new",3);  cout<<str<<endl;  } |

****

**LAB TASKS**

Armed forces recruiting agency is collecting data of volunteers for its five main branches i.e.,

ARMY, NAVY, AIRFORCE, SSG and AMC. You are required to write a C++ program that collects

data of the volunteers and store them in a file. For each volunteer, following information needs

to be stored in a file (all details are in character array form).

* Name
* DOB (dd/mm/yyyy)
* Age

The program will read the preferences from a file “branches.txt” and display the options in the

screen for the user to select. The user may select the preference by pressing 1 for Army, 2 for

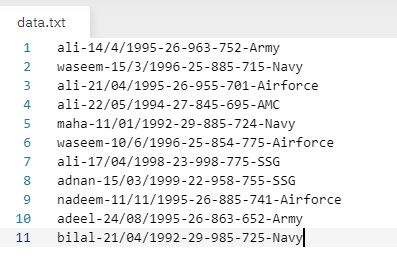
Navy etc. Once the user has entered details of the volunteer i.e., name, DOB, age, and preference,

the details of each volunteer is stored in a file. The program will ask the user if you want to enter

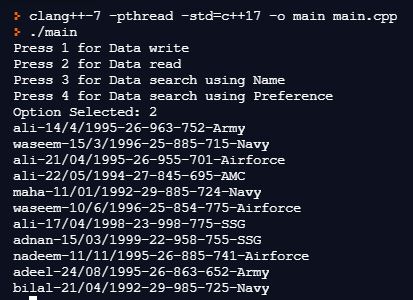
another data if YES then another set of details will be entered and the information will be

appended in the output file i.e., “data.txt”. if NO then the program will terminate.

The stored information is in the following format:



The program reads the entire contents of the file i.e., “*data.txt*” and display the entries on the screen when option 2 is selected. The program output is as follows.



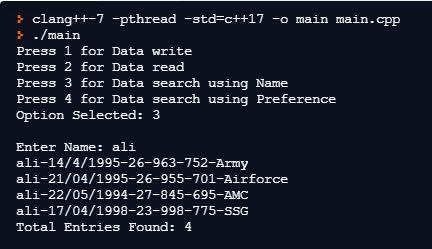
In this lab, you need to incorporate file search function in the existing code and use string functions to perform the following tasks.

**Task No 1: File Search using Applicant Name.**

Write a program that lists all volunteers with a specific name and display the list on the screen.

* The program will take the character array of the specific name from the user and use searching for substring function i.e., **strstr** to find the entries with required name and list the entries with the specific name along with the total number of entries found in the search.

**REQUIRED OUTPUT**

****

**Task No 2: File Search using Preference.**

Write a program that lists all volunteers with a specific preference and display the list on the screen.

* The program will read the preferences from a file “branches.txt” and display the options on the screen for the user to select. The user will enter the required preference (abbreviation only) and then the program will use searching for substring function i.e., **strstr** to find the entries with required preference and list the entries along with the total number of entries found in the search.

**REQUIRED OUTPUT**

