STRINGS,, CHAR ARRAY ,C-STYLE STRING(CSTRING) , ARRAY OF STRINGS

In C++, a string is a sequence of characters represented as a contiguous array of characters terminated by a null character '\0'. It is a standard library class that is defined in the header file "string".

C++ provides two types of strings:

1. C-style strings: These are arrays of characters terminated by a null character '\0', and are commonly used in C programming.
2. C++ strings: These are objects of the std::string class, which provides several methods and features for working with strings.

Difference between cstyle string and cstring in c++

Eg: char str1[] = { 'h', 'e', 'l', 'l', 'o', '\0' }; // C-style string

char str2[] = "world"; // C-style string

C-style strings are represented as arrays of characters terminated by a null character (**'\0'**). They are a fundamental data type in C and C++, and are widely used in C++ code that interfaces with C libraries or APIs. C-style strings have a fixed size and do not support dynamic resizing or many of the string manipulation functions provided by the **std::string** class. They are also prone to buffer overflow errors if not used carefully.

The **std::string** class, on the other hand, is a C++ class that provides many powerful string manipulation functions and supports dynamic resizing. It is a part of the C++ Standard Library and provides a safer and more flexible alternative to C-style strings. **std::string** objects can be easily concatenated, searched, and modified, and can be resized as needed.

Here are some differences between C-style strings and the **std::string** class:

1. C-style strings are represented as arrays of characters, while **std::string** objects are C++ objects.
2. C-style strings have a fixed size, while **std::string** objects can be resized dynamically.
3. C-style strings are terminated by a null character (**'\0'**), while **std::string** objects keep track of their own length.
4. C-style strings do not provide many built-in string manipulation functions, while **std::string** objects provide many powerful functions, such as **substr()**, **find()**, **replace()**, and **insert()**.
5. C-style strings can be prone to buffer overflow errors, while **std::string** objects automatically manage memory allocation and deallocation, and are less prone to such errors.

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |

Differnece between c-sytle string and character qrray in c++

In C++, a character array and a C-style string are not exactly the same, although they are closely related.

A character array is simply an array of characters, which can be defined as follows

Char arr[10];

A C-style string, on the other hand, is also an array of characters, but it is terminated by a null character (**'\0'**). This means that the last element of a C-style string is always **'\0'**. C-style strings are often used to represent text in C++ programs.

Char arr[] = “djdjd”;

This defines a C-style string with the contents "Hello". Note that the size of the array is not specified explicitly. Instead, the compiler automatically determines the size of the array based on the size of the string literal plus one for the terminating null character.

So, while a character array is simply an array of characters, a C-style string is a special kind of character array that is terminated by a null character.

In C++, a character array and a C-style string are closely related, but there are some differences between them.

A character array is a fixed-size array of characters, declared using the `char` keyword followed by the array name and its size in brackets. Character arrays can be used to store and manipulate strings of characters, but they do not have any built-in functionality for string manipulation. Character arrays are typically manipulated using functions such as `strcpy()`, `strcat()`, and `strlen()`, which are defined in the `cstring` header.

On the other hand, a C-style string is a sequence of characters terminated by a null character `'\0'`. C-style strings are typically declared using a character array initialized with a string literal. For example:

```cpp

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

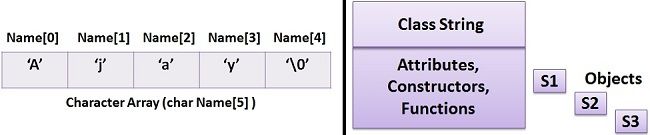
```

In this case, `str` is a character array initialized with the string literal "Hello, world!", including the null terminator.

C-style strings have built-in functionality for string manipulation, including functions such as `strcpy()`, `strcat()`, `strlen()`, `strcmp()`, and many more. These functions are defined in the `cstring` header and operate on null-terminated character arrays.

The main difference between a character array and a C-style string is that a C-style string includes a null terminator, while a character array may or may not include a null terminator. This means that C-style strings can be used with a wide range of string manipulation functions, while character arrays need to be manipulated using specialized functions defined in the `cstring` header.

Difference Between Character Array and String

C++ supports both, **Character array and string**, as C++ has considerable benefits in using both of them. But, inability to operate on character array raises the development of class string. Both a character array and string contain the sequence of characters

Access to the string is slow as compared to a character array or null terminated string.

### **Key Differences Between Character Array and String**

1. A character array is a collection of variables which are of character datatype. String is a class that is instantiated to declare strings.
2. Using index value you can access a character from a character array. On the other hand, if you want to access a particular character in a string, you can access it by function string’s\_name.charAt(index).
3. As an array is not a datatype similarly a character also is not a datatype. On the other hand, String being a class act as a reference type hence, it can be said String is a data type.
4. You can not apply any operator on a character array whereas, you can apply operators on String.
5. Being an array character array has a fixed length and its boundaries can be easily overrun. Where String does not have any boundaries.
6. Array elements are stored in a contiguous memory location hence that can be accessed faster than string variable.

Sting functions and operations

|  |  |
| --- | --- |
| **Sr.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. |

An example to take input in string and array

char str[100];

cout << "Enter a string: ";

cin.get(str,6);

string str = “ghjfe”

getline (cin,str);

Following example makes use of few of the above-mentioned functions −

[Live Demo](http://tpcg.io/oA6mP3)

#include <iostream>

#include <cstring>

using namespace std;

int main () {

char str1[10] = "Hello";

char str2[10] = "World";

char str3[10];

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 lenghth of str1 after concatenation

len = strlen(str1);

cout << "strlen(str1) : " << len << endl;

return 0;

}

STRINGS::

A string variable contains a collection of characters surrounded by double quotes:

IN c++ a single ‘\’ will always be ignored.so if u want to print any special character use \ and then add ur special character.

Properties of strings:

<https://www.geeksforgeeks.org/c-string-class-and-its-applications/>

Operations on strings:

1. ‘+’
2. Append
3. Length() / Size()
4. Str[0] (also str can be manipulated)
5. Push\_back
6. Pop\_back()
7. Resize()
8. Length()
9. Begin(), end()
10. Rbegin(), rend()

11)copy(“char array”, len, pos) = to copy an string into an array

12)swap() – to swap two strings()

13) find()

14) substr()

15) erase

16)replace

17)at()

18) reverse()

19)

CSTRING

<https://www.geeksforgeeks.org/cpp-cstring/>

differnec between char array and cstring in c++

In C++, `char` array and `cstring` are both used to represent strings, but there are some key differences between them.

1. Size:

`char` array has a fixed size, whereas `cstring` can dynamically resize itself according to the length of the string it holds.

2. Null-terminated:

A `cstring` always ends with a null character (`\0`), which indicates the end of the string. On the other hand, a `char` array may or may not be null-terminated, and it's the programmer's responsibility to ensure that the last element is a null character.

3. Functions:

`cstring` provides several functions to manipulate strings, such as `strlen`, `strcpy`, `strcat`, `strcmp`, etc., while `char` array doesn't provide any built-in functions for string manipulation.

Here's an example to illustrate the difference between `char` array and `cstring`:

```

#include <iostream>

#include <cstring>

int main() {

char char\_array[] = {'H', 'e', 'l', 'l', 'o'}; // Not null-terminated

char cstring\_array[] = "Hello"; // Null-terminated

std::cout << strlen(char\_array) << std::endl; // Undefined behavior

std::cout << strlen(cstring\_array) << std::endl; // Outputs 5

char new\_char\_array[10];

strcpy(new\_char\_array, char\_array); // Undefined behavior

char new\_cstring\_array[10];

strcpy(new\_cstring\_array, cstring\_array); // Copies "Hello"

std::cout << new\_char\_array << std::endl; // Undefined behavior

std::cout << new\_cstring\_array << std::endl; // Outputs "Hello"

strcat(char\_array, " World"); // Undefined behavior

strcat(cstring\_array, " World"); // Appends " World" to "Hello"

std::cout << char\_array << std::endl; // Undefined behavior

std::cout << cstring\_array << std::endl; // Outputs "Hello World"

return 0;

}

```

ARRAY OF STRINGS

<https://www.geeksforgeeks.org/array-of-strings-in-cpp-5-different-ways-to-create/>

A stringstream associates a string object with a stream allowing you to read from the string as if it were a stream (like cin).

1. For finding the unique elements in array use xor , because all the elements that are repeated twice will get deleted. Its order is o(n);

For splitting a string in c++

### 1->Use strtok() function to split strings

**strtok():** A strtok() function is used to split the original string into pieces or tokens based on the delimiter passed.

**Syntax**

1. **char** \*ptr = strtok( str, delim)

In the above syntax, a strtok() has two parameters, the **str**, and the **delim**

**str**: A str is an original string from which strtok() function split strings.

**delim**: It is a character that is used to split a string. For example, comma (,), space ( ), hyphen (-), etc.

**Return**: It returns a pointer that references the next character tokens. Initially, it points to the first token of the strings.

#include <iostream>

#include <cstring>

**using** **namespace** std;

**int** main()

{

**char** str[100]; // declare the size of string

    cout << " Enter a string: " <<endl;

    cin.getline(str, 100); // use getline() function to read a string from input stream

**char** \*ptr; // declare a ptr pointer

    ptr = strtok(str, " , "); // use strtok() function to separate string using comma (,) delimier.

    cout << " \n Split string using strtok() function: " << endl;

    // use while loop to check ptr is not null

**while** (ptr != NULL)

    {

        cout << ptr  << endl; // print the string token

        ptr = strtok (NULL, " , ");

    }

**return** 0;

}

In C++, std::substr() is a predefined function used for string handling. **string.h** is the header file required for string functions.

This function takes two values **pos** and **len** as an argument and returns a newly constructed string object with its value initialized to a copy of a sub-string of this object. Copying of string start from *pos* and done till *pos+len* means **[pos, pos+len) .**

// CPP program to illustrate substr()

#include <string.h>

#include <iostream>

using namespace std;

int main()

{

    // Take any string

    string s1 = "Geeks"; o/p🡪 eek

    // Copy three characters of s1 (starting

    // from position 1)

    string r = s1.substr(1, 3);

    // prints the result

    cout << "String is: " << r;

    return 0;

}

To get a substring after a particular value

#include <string.h>

#include <iostream>

using namespace std;

int main()

{

    // Take any string

    string s = "dog:cat";

    // Find position of ':' using find() /// o/p ->cat

    int pos = s.find(":");

    // Copy substring after pos

    string sub = s.substr(pos + 1);

    // prints the result

    cout << "String is: " << sub;

    return 0;

}

Important operations on strings ;;

**1. getline()** :- This function is used to**store a stream of characters** as entered by the user in the object memory.  
**2. push\_back()** :- This function is used to **input** a character at the**end** of the string.  
**3. pop\_back()**:- Introduced from C++11(for strings), this function is used to**delete the last character** from the string.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

some important functionalities of char array in c++ --------------------------

strchr :

#include <iostream> // to find the first occurence

#include<cstring>

using namespace std;

int main(){

char sr[]="afghg";

char \*ch;

ch=strchr(sr, 'g');

cout<<ch-sr+1<<endl;

cout<<ch; // prints all characters from that point

}

2) also to find whether charater is present or not

#include <iostream>

#include<cstring>

using namespace std;

int main(){

char str[] = "My name is Ayush";

char ch = 'A', ch2 = 'z';

if (strchr(str, ch) != NULL)

cout << ch << " "

<< "is present in string" << endl;

else

cout << ch << " "

<< "is not present in string" << endl;}

STRSTR ( similar to strchr)

1-🡪 #include <iostream>

#include<cstring>

using namespace std; // find the occurenece of one string in another (number)

int main()

{

char s1[] = "GeeksforGeeks";

char s2[] = "for";

char\* p;

// Find first occurrence of s2 in s1

p = strstr(s1, s2);

cout<<p-s1+1 ; }

2) #include<cstring> // to copy one string to another

#include<iostream>

using namespace std;

int main()

{

// Take any two strings

char s1[] = "Fun with STL";

char s2[] = "STL";

char\* p;

// Find first occurrence of s2 in s1

p = strstr(s1, s2);

cout<<p;

if (p) {

strcpy(p, "Strings");

cout<<s1;

} }

Append- --------------------------------------------------------

string str1("GeeksforGeeks ");

string str2("Hello World! ");

// Appends 5 characters from 0th index of str2 to str1

cout<<str1.append(str2,0,5); // (str, index, no\_of\_char)

Assign ----------------------------------------------------------------------------------------

  // Assigns 4 characters from

    // 5th index of str2 to str1 // str1= “geeeekss” ,str2=”ijklmnopqrs”

    str1.assign(str2, 5, 4); // str1= nopq

// Assigns first 5 characters of

    // GeeksforGeeks to str

    str.assign("GeeksforGeeks", 5);

 // Assigns 10 occurrences of 'x'

    // to str

    str.assign(10, 'x');

 // Assigns all characters between

    // str.begin()+6 and str.end()-0 to str1

    str1.assign(str.begin()+6, str.end()-0);

// applicable for all

void assignDemo(string str)

{

    string str1;

    // Assigns all characters between

    // str.begin()+6 and str.end()-0 to str1

    str1.assign(str.begin()+6, str.end()-0);

    cout << "After assign() : ";

    cout << str1;

}

// Driver code

int main()

{

    string str("Hello World!");

    cout << "Original String : " << str << endl;

    assignDemo(str);

    return 0;

}

ERASE:

    string str="This is a java tutorial";   // This is java tutorial

    str.erase(8,1);  // ( index , yast )

  string str="java programming";

JavaTpoint

|  |
| --- |
| **C++ String erase()** This function removes the characters as specified, reducing its length by one. Syntax Consider a string str. Syntax would be:   1. str.erase(pos,len); 2. str.erase(itr); 3. str.erase(first,last);  Parameter  * **pos** :It defines the position of the character which is to be removed. * **len** :It defines the number of characters to be erased. * **Itr** : It is an iterator to the character to be removed. * **Range(first,last)**: It defines the range within the string to be removed.  Return value It returns \*this. Example 1 Let's see a simple example when pos and len are given:   1. #include<iostream> 2. using namespace std; 3. **int** main() 4. { 5. string str="This is a java tutorial"; 6. str.erase(8,1);  // it should have two parames, one index and other how many characters 7. cout<<str; 8. **return** 0; 9. }   **Output:**  This is java tutorial Example 2 Let's see a simple example when iterator is passed in a parameter:   1. #include<iostream> 2. using namespace std; 3. **int** main() 4. { 5. string str="java programming"; 6. str.erase(str.begin()+11); 7. cout<<str; 8. **return** 0; 9. }   **Output:**  java programing |

    str.erase(str.begin()+11);

  str.erase(str.begin()+24,str.end());

The datatype size\_t is unsigned integral type. It represents the size of any object in bytes and returned by sizeof operator. It is used for array indexing and counting.

#include <iostream>

#include <string>

using namespace std;

int main()

{

string str = "geeksforgeeks a computer science";

string str1 = "geeks";

// Find first occurrence of "geeks"

size\_t found = str.find(str1);

if (found !=string::npos)

cout << "First occurrence is " << found << endl;

// Find next occurrence of "geeks". Note here we pass

// "geeks" as C style string.

char arr[] = "geks";

found = str.find(arr, found+1);

if (found != string::npos)

cout << "Next occurrence is " << found << endl;

return 0;}

  // Find first occurrence of 'g'

    size\_t found = str.find(c);

    if (found != string::npos)

        cout << "First occurrence is " << found << endl;

    // Find next occurrence of 'g'

    found = str.find(c, found+1);

**const char\*** and **char const\*** (both are same) says that the pointer can point to a constant char and value of char pointed by this pointer cannot be changed. But we can change the value of pointer as it is not constant and it can point to another constant char.

**char\* const** says that the pointer can point to a char and value of char pointed by this pointer can be changed. But we cannot change the value of pointer as it is now constant and it cannot point to another char.

**const char\* const** says that the pointer can point to a constant char and value of int pointed by this pointer cannot be changed. And we cannot change the value of pointer as well it is now constant and it cannot point to another constant char.

Thumb rule is to naming syntax from right to left.

// constant pointer to constant char

const char \* const

// constant pointer to char

char \* const

// pointer to constant char

const char \*

1. **int** k= str1.compare(str2);

* k==0 : If k contains value zero, it means both the strings are equal.
* k!=0 : If k does contain value zero, it means both the strings are unequal.
* k>0 : If k contains value more than zero, either the value of the first character is greater in the compared string or all the compared characters match but the compared string is longer.
* k<0 : If k contains value less than zero, either the value of the first character is lower in the compared string or all the compared characters match but the compared string is shorter.

   string str1="Hello";

   string str2="javatpoint";

**int** k= str1.compare(str2);

**if**(k==0)

         cout<<"Both the strings are equal";

**else**         cout<<"Both the strings are unequal";  }

swap fn :[]

string r = "10";

string m = "20"

r.swap(m);

SIZE -- > gives length()

string str ="Welcome to the javatpoint tutorial";

**int** size = str.size();

cout << "length of the string is :" <<size;

REPLACE ------------------------------------------

#include<iostream>

using namespace std;

int main()

{

string str1 ="This is C language";

string str3= "java";

cout <<"Before replacement, String is "<<str1<<'\n';

str1.replace(8,3,str3); //(where ,yast, str , yast from str)

cout<<"After replacement,String is "<<str1<<'\n';

return 0;

}

string str1="This is C language";

cout<<"Before replacement,string is"<<str1<<'\n';

str1.replace(8,1,"C##",2); // This is C ## language

KEYWORD :: Count in c++

#include <bits/stdc++.h>

using namespace std;

int main()

{

    string str = "geeksforgeeks";

    cout << "Number of times 'e' appears : "

         << count(str.begin(), str.end(), 'e');

    return 0;

}

#include <bits/stdc++.h>

using namespace std;

int main()

{

    int arr[] = { 3, 2, 1, 3, 3, 5, 3 };

    int n = sizeof(arr) / sizeof(arr[0]);

    cout << "Number of times 3 appears : "

         << count(arr, arr + n, 3);

    return 0;

}

1. **Replace**

This function replaces the portion of string that begins at character position pos and spans len characters.

Syntax

Consider two strings str1 and str2. Syntax would be:

1. str1.replace(pos,len,str2);

Parameters

* **str :** str is a string object, whose value to be copied in another string object.
* **pos :** pos defines the position, whose character to be replaced.
* **len :** Number of characters to be replaced by another string object.
* **subpos :** It defines the position of the first character of string object that is to be copied to another object as replacement.
* **sublen :** Number of characters of string object to be copied into another string object.
* **n :** Number of characters to be copied into an another string object.

#include<iostream>

using namespace std;

**int**  main()

{

string str1 = "This is C language";

string str2 = "C++";

cout << "Before replacement, string is :"<<str1<<'\n';

str1.replace(8,1,str2);

cout << "After replacement, string is :"<<str1<<'\n';

**return** 0;

}

//2 nd programme

#include<iostream>

#include<cstring>

using namespace std;

int main()

{

string str1 = "This is C language";

string str2 = "C++";

cout << "Before replacement, string is :"<<str1<<'\n';

str1.replace((str1.find('C')),1,str2);

cout << "After replacement, string is :"<<str1<<'\n';

return 0;

}

BASICAAAS : 🡪

To print last character

#include<iostream>

#include<cstring>

using namespace std;

int main()

{

string s="hello";

char a[]="assjd";

int x=s.length();

cout<<s[x-1]; // prints the last character

cout<<s.length()-1; // prints the count

cout<<s[x-2]; // prints the second last character

int y=strlen(a);

cout<<a[y-1]; // pritnt slast charater

cout<<strlen(a)-1; // print len-1

}

// to count the no of occurences of “ aa ” in given string::🡪

#include <iostream>

using namespace std;

int test(string s)

{

int ctr\_aa = 0;

for (int i = 0; i < s.length() - 1; i++)

{

if (s.substr(i, 2) == "aa")

{

ctr\_aa++;

}

}

return ctr\_aa;

}

int main()

{

cout << test("bbaaccaag") << endl;

cout << test("jjkiaaasew") << endl;

cout << test("JSaaakoiaa") << endl;

return 0;

}

Pgm to print aababcabcd fron abcd (only function)

string test(string str)

#include <iostream>

using namespace std;

string test(string str)

{

string result = "";

for (int i = 0; i < str.length(); i++)

{

result += str.substr(0, i + 1);

}

return result;

}

int main()

{

cout << test("abcd") << endl;

cout << test("abc") << endl;

cout << test("a") << endl;

return 0;

}

Write a C++ program to count a substring of length 2 appears in a given string and also as the last 2 characters of the string. Do not count the end substring.

**Sample Solution**:

**C++ Code :**

#include <iostream>

using namespace std;

int test(string str)

{

string last\_two\_char = str.substr(str.length()-2);

int ctr = 0;

for (int i = 0; i < str.length()-2; i++)

{

if (str.substr(i, 2) == (last\_two\_char)) ctr++;

}

return ctr;

}

int main()

{

cout << test("abcdsab") << endl;

cout << test("abcdabab") << endl;

cout << test("abcabdabab") << endl;

cout << test("abcabd") << endl;

return 0;

}

SORT function to sort the given array

#include<iostream>

#include<algorithm>

using namespace std;

int main()

{

int a[]={3,5,1,2};

int l=sizeof(a)/sizeof(a[0]);

sort(a,a+l);

for(int i=0;i<l;i++){

cout<<a[i]<<" ";

}

}

# **C++ String erase()**

This function removes the characters as specified, reducing its length by one.

## Syntax

Consider a string str. Syntax would be:

1. str.erase(pos,len);
2. str.erase(itr);
3. str.erase(first,last);

## Parameter

* **pos** :It defines the position of the character which is to be removed.
* **len** :It defines the number of characters to be erased.
* **Itr** : It is an iterator to the character to be removed.
* **Range(first,last)**: It defines the range within the string to be removed.

#include<iostream>

using namespace std;

**int** main()

{

    string str="This is a java tutorial";

    str.erase(8,1);

    cout<<str;

**return** 0;

}

 C++ program to create a new string of the characters at indexes 0,1, 4,5, 8,9 ... from a given string

string s="abcdefghijk";

cout<<s.erase(5); // deletes everything after index

cout<<s.erase(3,2); // from poition 3 , 2 items will be deleted

cout<<s.erase(3,1); // to remove any particular element

int main(){

string s="python";

int l=s.length();

for(int i=0;i<l;i++){

if(i%2==0){

cout<<s[i]<<" ";

}

else{

cout<<s[i]<<" ";

i+=2;

}

} }

Pgm to print tables upto given number

#include <iostream>

#include<algorithm>

using namespace std;

int main(){

int res;

cout<<"enter no:";

cin>>res;

for(int i=1;i<=10;i++){

for(int j=1;j<res;j++){

cout<<j<<"\*"<<i<<"="<<j\*i<<" ";

}

cout<<endl;

}

}

Pgm to print patttternn (1 to 10)

#include <iostream>

#include<algorithm>

using namespace std;

int main(){

int n;

for(int i=0;i<5;i++){

static int k=0;

for(int j=0;j<i;j++){

cout<<k+1<<" ";

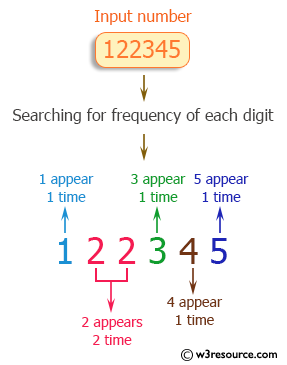
k++;

}

cout<<endl;

}

}

  
C++ Exercises: Find the frequency of each digit in a given integer

**Sample Solution**:-

**C++ Code :**

#include <iostream>

using namespace std;

int main()

{

int n, i, j, ctr, r;

cout << "\n\n Find frequency of each digit in a given integer:\n";

cout << "-----------------------------------------------------\n";

cout << " Input any number: ";

cin >> n;

for (i = 0; i < 10; i++)

{

cout << "The frequency of " << i << " = ";

ctr = 0;

for (j = n; j > 0; j = j / 10)

{

r = j % 10;

if (r == i)

{

ctr++;

}

}

cout << ctr << endl;

}

}

**C++ pgm to print checkboard**

#include <iostream>

using namespace std;

int main()

{

int i, j, rows;

string b, w, t;

b = "black";

w = "white";

cout << "\n\n Display checkerboard pattern with the words 'black' and 'white':\n";

cout << "---------------------------------------------------------------------\n";

cout << " Input number of rows: ";

cin >> rows;

for (i = 1; i <= rows; i++)

{

for (j = 1; j <= rows; j++)

{

if (j % 2 != 0)

{

cout << b;

if (j < rows)

{

cout << "-";

}

}

else if (j % 2 == 0)

{

cout << w;

if (j < rows)

{

cout << "-";

}

}

}

t = b;

b = w;

w = t;

cout << endl;

}

}

C++ pgm to sort the words in a given string

#include <iostream>

using namespace std;

int main(){

string s="python";

char temp;

for(int i=0;i<s.length();i++){

for(int j=i+1;j<s.length();j++){

if (s[i]>s[j]){

temp=s[i];

s[i]=s[j] ;

s[j]=temp;

}

}

}

cout<<s;

}

Pgmmm to reverse a whole string ;

#include <iostream>

#include<algorithm>

using namespace std;

int main(){

int c=0;

string t;

string s="python is a great language";

for(int i=s.length();i>0;i--)

{

if (s[i]!=' '){

t+=s[i];

}

else

break;

}

reverse(t.begin(),t.end());

cout<<t;

}

Pgm to print the words in reverseeeee

#include <iostream> /// my name is Vinayak

#include<algorithm> //// vinayak is name my

using namespace std;

int main(){

int c=0;

string t,r;

string s="python is a great language";

for(int i=s.length()-1;i>=0;i--){

if(s[i]==' '){

r+=s.substr(i);

s.erase(i);

}

}

cout<<r;

cout<<" "<<"python";

}

**PPPgm to insert hyphen (-) in between two odd numbers**

#include <iostream>

#include<algorithm>

using namespace std;

int main(){

int n=1345789;

string s=to\_string(n);

for(int i=0;i<s.length();i++){

int x=int(s[i]);

int y=int(s[i+1]);

if(x%2!=0 && y%2!=0){

cout<<s[i]<<'-';

}

else

cout<<s[i];

}

}

To find the elemnts which are present only in array by I.e. unique elements in an array using xor operation

#include <iostream>

using namespace std;

int main()

{

int a1[]={1,2,3,4,5,6};

int a2[]={3,2,1};

for(int i=0;i<6;i++){

int f=0;

for(int j=0;j<3;j++){

if((a1[i] ^a2[j])!=0)

continue;

else

f=1;

}

if(f==0)

cout<<a1[i]<<" ";

}

To find only the common elements in both arrays i.e. the elemnts present only in both arrys ,common to both arrays

#include <iostream>

using namespace std;

int main()

{

int a1[]={1,2,4,5,6};

int a2[]={3,2,1};

for(int i=0;i<5;i++){

for(int j=0;j<3;j++){

if((a1[i]^a2[j])==0)

{ cout<<a1[i]<<" ";}

}

}

}

Pgm to count the number of ones 1’s or any other number in a given range

#include <iostream>

#include<cmath>

using namespace std;

int main(){

int ct=0;

for(int i=1;i<=12;i++) {

string s;

s=to\_string(i);

for(int j=0;j<s.length();j++){

if(s[j]=='1')

{ct++;

break;

}}

}

cout<<ct;

}

C++ Algorithm **max()** function can be used in following 3 ways:

* It compares the two values passed in its arguments and **returns the larger between them**. If both are equal, then it returns the first one.
* It also compares the two values using a **binary function** which is defined by the user, and then passed as an argument in std::max().
* It is also used to find the **largest element in a given list**, and it returns the first one if there are more than one are largest in the list.

#include <iostream>     // std::cout

#include <algorithm>    // std::max

**bool** comp(**int** a, **int** b)

{

**return** (a < b);

}

**using** **namespace** std;

**int** main () {

  cout << "max(1,2)==" << max(1,2) << '\n';

  cout << "max(2,1)==" << max(2,1) << '\n';

  cout << "max('a','z')==" << max('a','z') << '\n';

  cout << "max(3.14,2.73)==" << max(3.14,2.73) << '\n';

 cout << max(a,b,comp);

cout << "Maximum element is: "<< max({1, 2, 3, 4, 5, 10, -1, 7},comp) << "\n";

**return** 0;

}

|  |  |
| --- | --- |
| **char s[10]= “abcd”** | **char \*s= “stuv”** |
| s[] is an array | \*s is a pointer |
| sizeof(s) = 10 bytes | sizeof(\*s) = 4 bytes |
| ‘abcd’ is stored in stack section of memory | ‘stuv’ is stored in code section of memory, whereas \*s is stored in a stack section of memory |
| char s[10] = “abcd”  s= “hello”; //  This code is invalid as s is an address and string constant is also an address, so it is impossible. | char \*s = “stuv”;  s=” Ninja”; // This code is valid |
| s++ in invalid in array | s++ is valid in pointer |
| char s[10] = “abcd”;  S[0] = ‘b’ // This code is valid. | char \*s = “stuv”;  s[0] =”r” // This code is invalid |
| We can edit the elements in s[] array | We cannot edit the elements in \*s pointer |

**Difference between Dot(.) and Arrow(->) operator:**

* The Dot(.) operator is used to normally access members of a structure or union.
* The Arrow(->) operator exists to access the members of the structure or the unions using pointers.