ARRAY

Sort(a.begin(),a.end()) sort from iterator begin and end

abs(a-b) gives the absolute number always positive Number

\_\_gcd(m,n) give the gcd of m and n

sort(array , array + size) sort array in ascending

sort(array , array + size, greater<int>()) sort array in decsending

int n = sizeof(A) / sizeof(A[0]); finding size of array

Finding Minimum and Maximum in array:

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

cout<<\*min\_element(arr,arr+size)<<"\n";

cout<<\*max\_element(arr,arr+size)<<"\n";

work with array and vector

CONVERSION

Convetring a full string to int, this will take the whole string and will change it to string e.g (123+4) if we convert this

it will be (123) in integer the other will be ignored.

ways to convert:

1. Using stringstream class or sscanf()

stringstream() : This is an easy way to convert strings of digits into ints, floats or doubles. Following is a sample program

using a stringstream to convert string to int.

#include<sstream>

#include<iostream>

string s = "12345";

int x = 0;

stringstream geek(s);

//or istringstream(str) >> s;

geek >> x;

cout << "Value of x : " << x;

2. String conversion using stoi() or atoi()

stoi() : The stoi() function takes a string as an argument and returns its value. Following is a simple implementation:

#include<string>

#include<iostream>

string str1 = "45";

string str2 = "3.14159";

string str3 = "31337 geek";

int myint1 = stoi(str1);

int myint2 = stoi(str2);

int myint3 = stoi(str3);

cout << "stoi(\"" << str1 << "\") is "

<< myint1 << '\n';

cout << "stoi(\"" << str2 << "\") is "

<< myint2 << '\n';

cout << "stoi(\"" << str3 << "\") is "

<< myint3 << '\n';

atoi() : The atoi() function takes a character array or string literal as an argument and returns its value. Following is a simple implementation:

#include <iostream>

#include <cstdlib>

const char \*str1 = "42";

const char \*str2 = "3.14159";

const char \*str3 = "31337 geek";

int num1 = atoi(str1);

int num2 = atoi(str2);

int num3 = atoi(str3);

cout << "atoi(\"" << str1

<< "\") is " << num1 << '\n';

cout << "atoi(\"" << str2

<< "\") is " << num2 << '\n';

cout << "atoi(\"" << str3

<< "\") is " << num3 << '\n';

stoi() vs atoi()

atoi() is a legacy C-style function. stoi() is added in C++ 11.

atoi() works only for C-style strings (character array and string literal), stoi() works for both C++ strings and C style strings

atoi() takes only one parameter and returns integer value.

CONVERTING ARRAY OF STRING TO ARRAY OF INTEGER

we can convert any index of a string to integer array by this code:

arr[j] = (Str[i] – 48)

this code will take the string at index i change it to int and store it in index j of array which is integer.

the sample code;

the is input string like 1+2+3+4+5 i take this input as string and convert it to integer array;

string str;

cin>>str;

int size = (str.length()/2)+1;

int arr[size] = {0};

int index = 0;

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

{

arr[index] = (str[i] - 48);

index++;

i++;

}

according to the given sample in array there will be 1 2 3 4 5 so according to this example the string with seperators can also be converted using this technique.

LIST

List

Lists are sequence containers that allow non-contiguous memory allocation.

As compared to vector, list has slow traversal, but once a position has been found,

insertion and deletion are quick. Normally, when we say a List, we talk about doubly linked list.

For implementing a singly linked list, we use forward list

Functions:

front() – Returns the value of the first element in the list.

back() – Returns the value of the last element in the list .

push\_front(g) – Adds a new element ‘g’ at the beginning of the list .

push\_back(g) – Adds a new element ‘g’ at the end of the list.

pop\_front() – Removes the first element of the list, and reduces size of the list by 1.

pop\_back() – Removes the last element of the list, and reduces size of the list by 1

list::begin() and list::end() in C++ STL– begin() function returns an iterator pointing to the first element of the list

end()– end() function returns an iterator pointing to the theoretical last element which follows the last element.

list rbegin() and rend() function in C++ STL– rbegin() returns a reverse iterator which points to the last element of the list. rend() returns a reverse iterator which points to the position before the beginning of the list.

list cbegin() and cend() function in C++ STL– cbegin() returns a constant random access iterator which points to the beginning of the list. cend() returns a constant random access iterator which points to the end of the list.

list crbegin() and crend() function in C++ STL– crbegin() returns a constant reverse iterator which points to the last element of the list i.e reversed beginning of container. crend() returns a constant reverse iterator which points to the theoretical element preceding the first element in the list i.e. the reverse end of the list.

empty() – Returns whether the list is empty(1) or not(0).

insert() – Inserts new elements in the list before the element at a specified position.

erase() – Removes a single element or a range of elements from the list.

assign() – Assigns new elements to list by replacing current elements and resizes the list.

remove() – Removes all the elements from the list, which are equal to given element.

list::remove\_if() in C++ STL– Used to remove all the values from the list that correspond true to the predicate or condition given as parameter to the function.

reverse() – Reverses the list.

size() – Returns the number of elements in the list.

list resize()function in C++ STL– Used to resize a list container.

sort() – Sorts the list in increasing order.

list max\_size() function in C++ STL– Returns the maximum number of elements a list container can hold.

list unique() in C++ STL– Removes all duplicate consecutive elements from the list.

list::emplace\_front() and list::emplace\_back() in C++ STL– emplace\_front() function is used to insert a new element into the list container, the new element is added to the beginning of the list. emplace\_back() function is used to insert a new element into the list container, the new element is added to the end of the list.

list::clear() in C++ STL– clear() function is used to remove all the elements of the list container, thus making it size 0.

list::operator= in C++ STL– This operator is used to assign new contents to the container by replacing the existing contents.

list::swap() in C++ STL– This function is used to swap the contents of one list with another list of same type and size.

list splice() function in C++ STL– Used to transfer elements from one list to another.

list merge() function in C++ STL– Merges two sorted lists into one

list emplace() function in C++ STL– Extends list by inserting new element at a given position.

SET

For Displaying set items

for(auto i=num.begin(); i != num.end(); ++i){

cout<<\*i<<" ";

}

set.insert(num);

set<int> s(array, array+size); insert and array to the set

To give output to specific index in set:

set <int> arr; This is set

set <int>::iterator it = arr.begin(); Initialize an iterator

advance(it, index) give specific index

cout<<\*it<<"\n"; it is displayed

STRING

C++ has in its definition a way to represent sequence of characters as an object of class. This class is called std:: string. String class stores the characters as a sequence of bytes with a functionality of allowing access to single byte character.

Functions of 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.

4. capacity() :- This function returns the capacity allocated to the string, which can be equal to or more than the size of the string. Additional space is allocated so that when the new characters are added to the string, the operations can be done efficiently.

5. resize() :- This function changes the size of string, the size can be increased or decreased.

6.length():-This function finds the length of the string

7.shrink\_to\_fit() :- This function decreases the capacity of the string and makes it equal to the minimum capacity of the string. This operation is useful to save additional memory if we are sure that no further addition of characters have to be made.

8. begin() :- This function returns an iterator to beginning of the string.

9. end() :- This function returns an iterator to end of the string.

10. rbegin() :- This function returns a reverse iterator pointing at the end of string.

11. rend() :- This function returns a reverse iterator pointing at beginning of string.

12. copy(“char array”, len, pos) :- This function copies the substring in target character array mentioned in its arguments. It takes 3 arguments, target char array, length to be copied and starting position in string to start copying.

13. swap() :- This function swaps one string with other.

14. erase() :- erase all sring --- erase(string.begin() + i) erase char at position i --- erase(string.begin()+a, string.end()-i) erase from range a to i

15. sort() :- sort the string sort(str.begin(), str.end()) sort all of the string -- sort(str.begin()+a, str.end()-i) sort from a to i

16. compare() :- str1.compare(str2) if equal returns 0 if str1 < str2 returns -1 if str1 > str2 return 1

functions in Code

#include<iostream>

#include<string> // for string class

using namespace std;

int main()

{

// Declaring string

string str;

// Taking string input using getline()

// "geeksforgeek" in givin output

getline(cin,str);

// Displaying string

cout << "The initial string is : ";

cout << str << endl;

// Using push\_back() to insert a character

// at end

// pushes 's' in this case

str.push\_back('s');

// Displaying string

cout << "The string after push\_back operation is : ";

cout << str << endl;

// Using pop\_back() to delete a character

// from end

// pops 's' in this case

str.pop\_back();

// Displaying string

cout << "The string after pop\_back operation is : ";

cout << str << endl;

// Initializing string

string str = "geeksforgeeks is for geeks";

// Displaying string

cout << "The initial string is : ";

cout << str << endl;

// Resizing string using resize()

str.resize(13);

// Displaying string

cout << "The string after resize operation is : ";

cout << str << endl;

// Displaying capacity of string

cout << "The capacity of string is : ";

cout << str.capacity() << endl;

//Displaying length of the string

cout<<"The length of the string is :"<<str.length()<<endl;

// Decreasing the capacity of string

// using shrink\_to\_fit()

str.shrink\_to\_fit();

// Displaying string

cout << "The new capacity after shrinking is : ";

cout << str.capacity() << endl;

// Initializing string`

string str = "geeksforgeeks";

// Declaring iterator

std::string::iterator it;

// Declaring reverse iterator

std::string::reverse\_iterator it1;

// Displaying string

cout << "The string using forward iterators is : ";

for (it=str.begin(); it!=str.end(); it++)

cout << \*it;

cout << endl;

// Displaying reverse string

cout << "The reverse string using reverse iterators is : ";

for (it1=str.rbegin(); it1!=str.rend(); it1++)

cout << \*it1;

cout << endl;

// Initializing 1st string

string str1 = "geeksforgeeks is for geeks";

// Declaring 2nd string

string str2 = "geeksforgeeks rocks";

// Declaring character array

char ch[80];

// using copy() to copy elements into char array

// copies "geeksforgeeks"

str1.copy(ch,13,0);

// Diplaying char array

cout << "The new copied character array is : ";

cout << ch << endl << endl;

// Displaying strings before swapping

cout << "The 1st string before swapping is : ";

cout << str1 << endl;

cout << "The 2nd string before swapping is : ";

cout << str2 << endl;

// using swap() to swap string content

str1.swap(str2);

// Displaying strings after swapping

cout << "The 1st string after swapping is : ";

cout << str1 << endl;

cout << "The 2nd string after swapping is : ";

cout << str2 << endl;

return 0;

}

when you cin an int and after that you use getline function you have to do it twise because getline funtion also read "\n" this as an input.

when you getline you can seperate the word by using this:

string S, T;

getline(cin, S);

stringstream X(S);

while (getline(X, T, ' ')) {

cout << T << endl;

}

this code will cout each word of the sentence.

if you want to use char array and use getline function you have to write it like this

char arr[maxLen]

cin.getline(arr, maxlen, '$')

it will read till max length or till $ this sign and the third parameter is optional

this is how when you are given a line of number you get it as string and convert it to integer one by one;

getline(cin,s);

string T;

int total = 0;

stringstream x(s);

while(getline(x, T, ' ')){

int num = stoi(T);

total += num;

}

Count Number of accurance of a substring in a string

// Simple C++ program to count occurrences

// of pat in txt.

#include<bits/stdc++.h>

using namespace std;

int countFreq(string &pat, string &txt)

{

int M = pat.length();

int N = txt.length();

int res = 0;

/\* A loop to slide pat[] one by one \*/

for (int i = 0; i <= N - M; i++)

{

/\* For current index i, check for

pattern match \*/

int j;

for (j = 0; j < M; j++)

if (txt[i+j] != pat[j])

break;

// if pat[0...M-1] = txt[i, i+1, ...i+M-1]

if (j == M)

{

res++;

j = 0;

}

}

return res;

}

/\* Driver program to test above function \*/

int main()

{

string txt = "dhimanman";

string pat = "man";

cout << countFreq(pat, txt);

return 0;

}

Second Method for counting the occurrence of a substring in a string

int countFreq(string &pat, string &txt)

{

// pat is substring

// txt is string

int occurrences = 0;

std::string::size\_type pos = 0;

while ((pos = txt.find(pat, pos )) != std::string::npos) {

++ occurrences;

pos += pat.length();

}

return occurrences;

}

TIPS

the maximum GCD from 1 to n is always n/2 if n is odd then n-1/2.

find take 2 number for l up to r and find two x and y which its LCM is

between l and r, for that if 2l > r it is no such LCM if not x=l and y=2l will do.

given x find LCD(x,y) + GCD(x,y) = x. The x=1 and y=n-1 will always do.

if u want an array which arr[i] + arr[j] != arr[a] and 1<i,j,a<size. so put

1 in all index of array it will always do.

int digits = (int)log10(num); this will give the number of digits in num variable

int first = num / pow(10, digits) this give the first digit

int last = num % 10 this gives the last digit

cout << (ans ? "YES" : "NO") << endl; if ans = true output YES otherwise NO

For fobinaci sequence 1+2+3+...+n

if n is odd n + (n\*(n/2)

if n is even n + (n\*(n/2) - (n/2)

want binary of an int

just take a loop modules number by 2 store it in the array then divide number by 2 until num == 0

the array printed from last index till index 0 will be the binary number

while(n > 0){

int num = n % 2;

arr.push\_back(num);

n = n/2;

}

for(int i= arr.size()-1; i>=0; i--){

if(arr[i] == 1)cout<<i+1<<" ";

}

is\_sorted() this function checks the array that is it sorted or not

is\_sorted(arr.begin(), arr.end());

VECTOR

Vectors are same as dynamic arrays with the ability to resize itself automatically when an element is inserted or deleted,

with their storage being handled automatically by the container. Vector elements are placed in contiguous storage so that

they can be accessed and traversed using iterators. In vectors, data is inserted at the end. Inserting at the end takes differential time,

as sometimes there may be a need of extending the array. Removing the last element takes only constant time because no resizing happens.

Inserting and erasing at the beginning or in the middle is linear in time.

Fucntions:

begin() – Returns an iterator pointing to the first element in the vector

end() – Returns an iterator pointing to the theoretical element that follows the last element in the vector

rbegin() – Returns a reverse iterator pointing to the last element in the vector (reverse beginning). It moves from last to first element

rend() – Returns a reverse iterator pointing to the theoretical element preceding the first element in the vector (considered as reverse end)

cbegin() – Returns a constant iterator pointing to the first element in the vector.

cend() – Returns a constant iterator pointing to the theoretical element that follows the last element in the vector.

crbegin() – Returns a constant reverse iterator pointing to the last element in the vector (reverse beginning). It moves from last to first element

crend() – Returns a constant reverse iterator pointing to the theoretical element preceding the first element in the vector (considered as reverse end)

size() – Returns the number of elements in the vector.

max\_size() – Returns the maximum number of elements that the vector can hold.

capacity() – Returns the size of the storage space currently allocated to the vector expressed as number of elements.

resize(n) – Resizes the container so that it contains ‘n’ elements.

empty() – Returns whether the container is empty.

shrink\_to\_fit() – Reduces the capacity of the container to fit its size and destroys all elements beyond the capacity.

reserve() – Requests that the vector capacity be at least enough to contain n elements.

reference operator [g] – Returns a reference to the element at position ‘g’ in the vector

at(g) – Returns a reference to the element at position ‘g’ in the vector

front() – Returns a reference to the first element in the vector

back() – Returns a reference to the last element in the vector

data() – Returns a direct pointer to the memory array used internally by the vector to store its owned elements.

assign() – It assigns new value to the vector elements by replacing old ones

push\_back() – It push the elements into a vector from the back

pop\_back() – It is used to pop or remove elements from a vector from the back.

insert() – It inserts new elements before the element at the specified position

erase() – It is used to remove elements from a container from the specified position or range.

swap() – It is used to swap the contents of one vector with another vector of same type. Sizes may differ.

clear() – It is used to remove all the elements of the vector container

emplace() – It extends the container by inserting new element at position

emplace\_back() – It is used to insert a new element into the vector container, the new element is added to the end of the vector

sort(v.begin(),v.end()) sort the vector in ascending

sort(v.begin(), v.end(), greater <datatype>()) sort vector in descending

vector of Pairs

vector<pair<int,string>> name;

can be initialize like this = {{1,"string},

{2,"string"},

{3,"string"}};

can be cout like this

for(auto i=0; i<vec.size(); ++i){

cout<<vec.at(i).first<<vec.at(i).second; }

can store value in it like this:

vec.push\_back(make\_pair(1,"sdf"));

vec.push\_back(pair(1,"sfdf"));

vec.emplace\_back(1,"dasdf");

cout element of vector in for loop;

for(int &v : ans) cout<< v <<" ";

C++ Library Functions

C++ <cmath>

C++ cos()Returns Cosine of the Argument

C++ sin()Returns Sine of the Argument

C++ asin()Returns Inverse Sine a Number

C++ atan()Returns Inverse tangent a Number

C++ tan()Returns Tangent of the Argument

C++ atan2()Returns Inverse Tangent of a Coordinate

C++ acos()Returns Inverse cosine a Number

C++ sinh()returns hyperbolic sine of an angle

C++ ceil()Return ceiling value of number

C++ floor()Returns floor value of decimal number

C++ tanh()returns hyperbolic tangent of an angle

C++ fmod()Computes floating point remainder of division

C++ acosh()returns hyperbolic cosine of a number

C++ asinh()returns arc hyperbolic sine of a number

C++ atanh()returns arc hyperbolic tangent of a number

C++ log()Returns Natural Logarithm of a Number

C++ lround()Returns the long int value nearest to the argument

C++ trunc()Truncates the demical part of a number

C++ round()Returns integral value nearest to argument

C++ llround()Rounds argument to nearest long long int value

C++ rint()Rounds argument using current rounding mode

C++ lrint()Rounds argument using current rounding mode

C++ log10()Returns Base 10 Logarithm of a Number

C++ modf()Breaks Number Into Integral and Fractional Part

C++ exp()returns exponential (e) raised to a number

C++ exp2()Returns 2 raised to a Number

C++ ldexp()returns product of x and 2 raised to the power e

C++ expm1()Returns e raised to Power Minus 1

C++ nearbyint()Rounds argument to using current rounding mode

C++ ilogb()returns integral part of logarithm of |x|

C++ frexp()breaks float to its binary significand

C++ scalbn()Scales x by FLT\_RADIX to the power n

C++ log2()returns base2 logarithm of a number

C++ scalbln()Scales x by FLT\_RADIX to the power n

C++ log1p()returns natural logarithm of x+1.

C++ logb()returns logarithm of |x|

C++ remquo()Computer remainder and stores quotient of x/y

C++ sqrt()Computes Square Root of A Number

C++ cbrt()Computes Cube Root of a Number

C++ hypot()Returns Square Root of sum of square of Arguments

C++ fmax()returns largest among two arguments passed

C++ nextafter()returns next value after x in direction of y

C++ nexttoward()returns next value after x in direction of y

C++ nan()returns a quiet NaN value

C++ cmath abs()Returns absolute value of an argument

C++ fma()Returns Fused Multiply–Accumulate

C++ fmin()returns smallest among two given arguments

C++ copysign()returns num with value of first and sign of second

C++ fdim()Returns Positive Different Between Arguments

C++ cosh()Returns Hyperbolic Cosine of an Angle

C++ fabs()Returns absolute value of argument

C++ remainder()Returns remainder of x/y

C++ llrint()Rounds argument using current rounding mode

C++ pow()Computes Power a Number

C++ <cstdlib>

C++ strtod()returns string float to double

C++ atof()Converts String to Double

C++ strtol()Converts a string to number

C++ atol()Converts String to Integer

C++ strtoll()converts string to long long int in C++

C++ strtoull()converts string to unsigned long long int

C++ malloc()Allocates a block of unitialized memory

C++ realloc()reallocates a block of previously allocated memory

C++ srand()Seeds pseudo random number for rand()

C++ free()deallocates a block of memory

C++ atexit()registers function to be called on termination

C++ at\_quick\_exit()registers function and calls on quick termination

C++ getenv()returns pointer to environment variable passed

C++ quick\_exit()causes termination without cleaning resources

C++ \_Exit()causes termination without cleanup tasks

C++ bsearch()performs binary search on sorted array

C++ qsort()sorts array using quick-sort algorithm

C++ div()computes integral quotient and remainder of number

C++ cstdlib abs()Returns absolute value of an integer

C++ lldiv()computes integral division of two long long int.

C++ labs()returns absolute value of long or long int number

C++ ldiv()computes integral division of long int numbers

C++ llabs()returns absolute value of a long long int data

C++ mblen()determines size of a multibyte character

C++ mbtowc()converts multibyte character to a wide character

C++ wctomb()converts wide character to a multibyte character

C++ mbstowcs()converts multibyte char string to wide char seq

C++ wcstombs()converts wide character string to multibyte seq

C++ calloc()allocates block of memory and initializes to zero

C++ <cstring>

C++ memcpy()Copies block of memory from source to destination

C++ strcat()appends copy of string to end of another string

C++ memmove()copies memory even if there is overlapping blocks

C++ strcpy()Copies character string from source to destination

C++ strncpy()copies character string from source to destination

C++ strncat()appends string to end of another string

C++ memcmp()compares two pointer objects

C++ strncmp()compares two strings lexographically

C++ strcmp()Compare two strings

C++ memchr()searches for character in string

C++ strchr()searches for character in string

C++ strrchr()searches last occurence of a character in string

C++ strspn()gives length of maximum initial segment

C++ strcspn()searches a string for characters in another string

C++ strpbrk()search characters in one string in another string

C++ memset()copies character to beginning of string n times

C++ strstr()finds first occurrence of a substring in string

C++ strtok()Split string based on delimiter

C++ strerror()gives description of system error code

C++ strlen()Returns length of given string

C++ strcoll()compares two null terminated string

C++ strxfrm()transform byte string into implementation def form\

C++ <cctype>

C++ isalpha()checks if given character is alphabet or not

C++ isblank()checks if given character is a blank character

C++ iscntrl()checks if given character is control character

C++ isdigit()Checks if given character is a digit or not

C++ isgraph()checks if given character is graphic or not

C++ islower()checks if given character is lowercase

C++ isspace()check if given character is whitespace character

C++ isupper()check if given character is uppercase or not

C++ isprint()check if given character is printable or not

C++ ispunct()check if given character is punctuation character

C++ isxdigit()Checks if given character is hexadecimal character

C++ tolower()Converts a given character to lowercase

C++ toupper()Converts a given character to uppercase

C++ <cwctype>

C++ iswspace()checks if given wide character is wide whitespace

C++ iswgraph()checks if wide char has graphical representation

C++ iswupper()checks if given wide character is uppercase

C++ iswalnum()checks if given wide character is alphanumeric

C++ iswlower()checks if given wide character is lowercase

C++ iswalpha()checks if given wide character is an alphabet

C++ iswprint()checks if given wide character can be printed

C++ iswblank()checks if given wide character is blank character

C++ iswpunct()checks if given wide character is punctuation

C++ iswcntrl()checks if given wide char is control character

C++ iswctype()checks if given wide char has certain property

C++ towctrans()transforms a given wide character

C++ iswxdigit()checks if given wide character is hexadecimal num

C++ wctrans()returns current transformation for wide character

C++ towlower()converts given wide character to lowercase

C++ wctype()returns wide character classification

C++ towupper()converts given wide character to uppercase

C++ iswdigit()checks if given wide character is digit or not

C++ <cstdio>

C++ remove()Deletes the specified file

C++ rename()renames or moves specified file

C++ tmpfile()creates temporary file with auto-generated name

C++ fclose()closes given file stream

C++ fopen()opens specified file

C++ setbuf()sets the internal buffer to be used for I/O

C++ tmpnam()generates unique filename

C++ fscanf()read data from file stream

C++ printf()Write formatted string to stdout

C++ scanf()Read data form stdin

C++ fprintf()write a formatted string to file stream

C++ sscanf()read data from string buffer

C++ snprintf()write formatted string to character string buffer

C++ sprintf()Write a formatted string to buffer

C++ vfprintf()write formatted string to file stream

C++ vsprintf()write formatted string to a string buffer

C++ vprintf()printf but takes args from vlist instead

C++ vsnprintf()write formatted string to string buffer

C++ fgetc()reads the next character from given input stream

C++ fgets()reads n number of characters from file stream

C++ fputc()writes character to given output stream

C++ fputs()writes string to file stream

C++ getchar()reads next character from stdin

C++ gets()reads line from stdin

C++ putc()writes character to given output stream

C++ putchar()writes a character to stdout

C++ puts()writes string to stdout

C++ fread()reads specified no. of characters from stream

C++ fwrite()writes specified number of characters to stream

C++ fgetpos()gets current file position

C++ fsetpos()sets stream file pointer to given position

C++ ftell()returns current position of file pointer

C++ perror()prints error to stderr

C++ rewind()sets file position to beginning of stream

C++ clearerr()resets error flags and EOF indicator for stream

C++ feof() functionchecks if file stream EOF has been reached or not

C++ ferror()checks for errors in given stream

C++ setvbuf()change or specify buffering mode and buffer size

C++ vfscanf()read data from a file stream

C++ fflush()flushes any buffered data to the respective device

C++ freopen()opens a new file with stream associated to another

C++ vscanf()read data from stdin

C++ vsscanf()read data from a string buffer

C++ ungetc()push previously read character back to the stream

C++ fseek()sets file position indicator for given file stream

C++ getc()reads next character from input stream

C++ <cwchar>

C++ fwide()set or query orientation of given file stream

C++ fgetwc()reads next wide character from given input stream

C++ fgetws()reads specified num of wide characters from stream

C++ fputwc()writes wide character to the given output stream

C++ fputws()writes wide string except null wide char to output

C++ fwprintf()write formatted wide string to a file stream

C++ fwscanf()reads wide character from file stream

C++ getwc()reads next wide character from input stream

C++ getwchar()reads next wide character from stdin

C++ ungetwc()push previously read wide character back to stream

C++ putwc()writes wide character to the given output stream

C++ vfwprintf()write formatted wide string to a file stream

C++ putwchar()writes wide character to stdout

C++ vfwscanf()read wide character string from a file stream

C++ swprintf()write formatted wide string to wide string buffer

C++ vswprintf()write formatted wide string to wide string buffer

C++ swscanf()reads wide character from wide string buffer

C++ vswscanf()read wide character string from wide string buffer

C++ vwprintf()write formatted wide string to stdout

C++ vwscanf()read wide character from stdin

C++ wprintf()write formatted wide string to stdout

C++ wscanf()reads wide character from stdin

C++ wcstok()returns next token in null terminated wide string

C++ wcstod()converts wide string float number to double

C++ wcstof()converts wide string float number to float

C++ wcstol()converts wide string float number to long int

C++ wcstold()converts wide string float number to long double

C++ mbsrtowcs()convert narrow multibyte char seq to wide char seq

C++ wcschr(searches for a wide character in a wide string

C++ btowc()converts character to its wide character

C++ wcrtomb()convert wide character to its narrow multibyte rep

C++ mbrlen()determines size in bytes of a multibyte character

C++ wctob()converts wide character to single byte character

C++ mbrtowc()converts narrow multibyte char to wide char

C++ wcsrtombs()convert wide char seq to narrow multibyte char seq

C++ mbsinit()describe initial conversion state of mbstate\_t obj

C++ wcscat()appends copy of wide string to the end of another

C++ wcslen()returns length of the given wide string

C++ wcsncat()appends specified num of wide char to another str

C++ wcscmp()lexicographically compares two wide string

C++ wcsncmp()compares specified number of wide char of strings

C++ wcscpy()copies wide character string from source to dest

C++ wcscspn()returns number of wide char before first occurence

C++ wcsrchr()searches last occurrence of wide char in string

C++ wcsspn()returns length of maximum initial segment

C++ wcsstr()finds first occurrence of wide substring in a str

C++ wcsncpy()copies specified number of wide characters

C++ wcsxfrm()transforms wide string to implementation defined

C++ wcspbrk() searches for set of wide char in given wide string

C++ wmemchr() searches for first occurrence of wide char

C++ wcsftime() converts given date and time to wide character str

C++ wmemcmp() compares wide chars of two wide strings

C++ wmemcpy() copies specified num of wide char from src to dest

C++ wmemmove() moves wide chars from src to dest

C++ wmemset() copies single wide char for a certain num of time

C++ wcstoll() converts wide string of specified base to int

C++ wcstoul() converts wide str of given base to unsigned long

C++ wcstoull() converts wide string num to unsigned long long

C++ wcscoll() compares two null terminated wide string