**to Check Given Character is Uppercase, Lowercase, Digit or Special Character**

Uppercase Alphabet: 65-90

Lowercase Alphabet: 97-122

Digit: 48-57

\_ Underscore : 95

If ASCII value of character is other then the values mentioned above then it is a special character.

get ascii value of char in cpp

C++ code: Here int() is used to convert character to its ASCII value.

cout << "The ASCII value of " << c << " is " << int(c);

get char from ascii value c++

1. Use Assignment Operator to Convert ASCII Value to Char in C++
2. Use the sprintf() Function to Convert ASCII Value to Char in C++
3. Use char() to Convert ASCII Value to Char

**inserting characters in a string c++**

**insert()**is used to insert characters in string at specified position.

    // Inserts str2 in str1 starting

    // from 6th index of str1

    str1.insert(6, str2);

**Caeser Cipher :**

I would encourage the use of the modulus operator (%) over subtracting 26 because any input that is greater than 26 would break a program using subtraction.

Vector Sort

sort(v.begin(), v.end());

sort in descending order?

sort(v.begin(), v.end(), greater<int>());

Pair

    // defining a pair

    pair<**int**, **char**> PAIR1;

    // first part of the pair

    PAIR1.first = 100;

    // second part of the pair

    PAIR1.second = 'G';

the extraction operator >> on cin to display a string entered by a user:

string firstName;  
cout << "Type your first name: ";  
cin >> firstName; // get user input from the keyboard  
cout << "Your name is: " << firstName;  
  
// Type your first name: John  
// Your name is: John

when working with strings, we often use the getline() function to read a line of text. It takes cin as the first parameter, and the string variable as second

string fullName;  
cout << "Type your full name: ";  
getline (cin, fullName);  
cout << "Your name is: " << fullName;  
  
// Type your full name: John Doe  
// Your name is: John Doe

# C++ cstdlib abs()

// get absolute value of -5

cout << abs(-5);

Palindrome

For digits we can calculate reverse of number and compare it.

String

str.size()

// Deletes character at position 4

    str.erase(str.begin() + 4);

// Deletes 4 characters from index number 1

    str.erase(1, 4);

//convert the string str variable to have an int value

//place the new value in a new variable that holds int values, named num

int num = stoi(str);

**Functions in c++**

Max()

Min()

Sort()

* Find the index of the first element having value same or just greater than **(sum – arr[i])** using [lower bound](https://www.geeksforgeeks.org/lower_bound-in-cpp/).
* Find the index of the first element having value just greater than **(sum – arr[i])** using [upper bound](https://www.geeksforgeeks.org/upper_bound-in-cpp/).

# Sieve of Eratosthenes

The sieve of Eratosthenes is one of the most efficient ways to find all primes smaller than n when n is smaller than 10 million or so

we move to our next unmarked number and mark all the numbers which are multiples of the number and are greater than or equal to the square of it

// Create a boolean array

"prime[0..n]" and initialize all entries it as true. A value in prime[i] will finally be false if i is Not a prime, else true.

// If prime[p] is not changed, then it is a prime

**for** (**int** p = 2; p \* p <= n; p++)

**if** (prime[p] == **true**)

// Update all multiples of p greater than or equal to the square of it numbers which are multiple of p and are less than p^2 are already been marked.   
  **for** (**int** i = p \* p; i <= n; i += p)

        prime[i] = **false**;

**When to use new operator in C++ and when it should not be used?**

Use of the new operator signifies a request for the memory allocation on the heap. If the sufficient memory is available, it initializes the memory and returns its address to the pointer variable.

The new operator should only be used if the data object should remain in memory until delete is called. Otherwise if the new operator is not used, the object is automatically destroyed when it goes out of scope. In other words, the objects using new are cleaned up manually while other objects are automatically cleaned when they go out of scope.

pointer\_variable = new datatype;

# Arrow operator -> in C/C++

An **Arrow operator in C/C++** allows to access elements in [Structures](https://www.geeksforgeeks.org/structures-in-cpp/) and [Unions](https://www.geeksforgeeks.org/union-c/). It is used with a [pointer variable pointing to a structure or union](https://www.geeksforgeeks.org/self-referential-structures/). The arrow operator is formed by using a minus sign, followed by the greater than symbol as shown below.   
**Syntax:**

(pointer\_name)->(variable\_name)

**Operation:** The -> operator in C or C++ gives the value held by variable\_name to structure or union variable pointer\_name.  
**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.

# Floyd’s Cycle Finding Algorithm

[Floyd’s cycle finding algorithm](https://www.geeksforgeeks.org/detect-loop-in-a-linked-list/) or Hare-Tortoise algorithm is a pointer algorithm that uses only two pointers, moving through the sequence at different speeds. This algorithm is used to find a loop in a linked list. It uses two pointers one moving twice as fast as the other one. The faster one is called the faster pointer and the other one is called the slow pointer.

**How Does Floyd’s Cycle Finding Algorithm Works?**

While traversing the linked list one of these things will occur-

* The Fast pointer may reach the end (NULL) this shows that there is no loop in the linked list.
* The Fast pointer again catches the slow pointer at some time therefore a loop exists in the linked list.

while(fast->next!=NULL || fast!=NULL)

let's say fast is equal to null, then what would happen if we try to find the next node of fast ? ERROR!

Because if there is nothing ( null ) at fast, there won't be anything at its next. So all you have to do is swap the condition.

while(fast!=null || fast->next!=null)

but wait, there's one more thing. Now's let say this while condition checks if fast == null and says it's true but because of OR (||) it would not check the second condition and will give the error from second mistake if it tries to find fast->next->next and first next comes out as null.

for eg. fast->null->(???) there is nothing next to null

therefore it will be:

while(fast!=null && fast->next!=null)

Also the if condition will be :

if(fast == slow ) you don't know if fast will ever be one behind than the slow but you know for sure that fast will be same as slow at one point.

//Decimal to Binary

// array to store binary number

**int** binaryNum[32];

    // counter for binary array

**int** i = 0;

**while** (n > 0) {

        // storing remainder in binary array

        binaryNum[i] = n % 2;

        n = n / 2;

        i++;

    }

Rev(binaryNum)

# C++ String to int and vice versa

string str = "123";

int num;

// using stoi() to store the value of str1 to x

num = std::stoi(str);

int num = 123;

string str = to\_string(num);

std::cout << str;

# Find min or max value in a vector in C++

std::min\_element and std::max\_element return an iterator to the minimum and the maximum value;

    int max = \*max\_element(v.begin(), v.end());

    int min = \*min\_element(v.begin(), v.end());

    /\*

    We create a 2D vector containing "n"

    elements each having the value "vector<int> (m, 0)".

    "vector<int> (m, 0)" means a vector having "m"

    elements each of value "0".

    Here these elements are vectors.

    \*/

    vector<vector<**int**>> vec( n , vector<**int**> (m, 0));

**class** Person{

**protected**:

        string firstName;

        string lastName;

**int** id;

**public**:

        Person(string firstName, string lastName, **int** identification){

**this**->firstName = firstName;

**this**->lastName = lastName;

**this**->id = identification;

        }

};

**class** Student :  **public** Person{

**private**:

        vector<**int**> testScores;

**public**:

        Student(string firstName,string lastName,**int** id,vector<**int**> scores): Person(firstName, lastName, id) {

**this**->testScores=scores;

        }

**Traversing using begin() and end()**

map<**char**,**int**>::iterator it=m.begin();

**while**(it!=m.end(){

it->first;

it->second;

it++;

}

**Using a range based for loop**

cout << "Element  Frequency" << endl;

**for** (**auto** i : m)

        cout << i.first <<""<< i.second

 //declaring vector of pairs

    vector< pair <**int**,**int**> > vect;

vect.push\_back( make\_pair(arr[i],arr1[i]) );

By default the sort function sorts the vector elements on basis of first element of pairs

    // Using simple sort() function to sort

    sort(vect.begin(), vect.end());

    // Using sort() function to sort by 2nd element

    // of pair

    sort(vect.begin(), vect.end(), sortbysec);

Errors:

" Abort signal from abort(3) (SIGABRT)"

This is due to the stoi function. This does not work with a very large number.  
For a string to int conversion try the following approach.

Algorithm to manually converting a string to int:

**int x = 0; // may take long long**

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

**x = x \* 10 + s[i] - '0';**

**Compilation Error: void value not ignored as it ought to be in std::queue::pop()**

error is quite simple; STL based containers that have pop and push algorithms do not return the value during the pop. Thus, you have to call front() (Or back(), depending on what you need) on the container to get the value before popping it.