To Check Given Character is Uppercase, Lowercase, Digit or Special Character

Uppercase Alphabet: 65-90

Lowercase Alphabet: 97-122

Digit: 48-57

_ Underscore : 95

Blank space : 32

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.

- Find the index of the first element having value just greater than **(sum – arr[i])** using upper bound.

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 and Unions. It is used with a pointer variable pointing to a structure or union. 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 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.
```

//first create a pointer of node, before using function to add values, in class

```
        Node *newnode;

        newnode=new Node(1);
```

```
//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());
```

Traversing a map

```
for (auto i : m)
     cout << i.first << "    " << i.second
```

```
            << endl;

//after inheriting class to create var of that class and use
constructor

//Write MyBook class
class MyBook : public Book {
    private:
        int price;

    public:
        MyBook(string title, string author, int price) : Book(title, author),
price(price) {

        }
};
```

```
    /*
    Create a vector containing "n"
    vectors each of size "m".
    */
    vector<vector<int>> vec( n , vector<int> (m));
```

```
/*
    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.

Ways to copy a vector in C++

Method 1: Iterative method.

Method 2: By assignment "=" operator. Simply assigning the new vector to the old one copies the vector. This way of assignment is not possible in the case of arrays.

```
// Using assignment operator to copy one vector to other
    vect2 = vect1;
```

Method 3: By passing vector as constructor. At the time of declaration of vector

```
// constructor method, Deep copy
    vector<int> vect2(vect1);
```

Method 4: copy(first_iterator_o, last_iterator_o, back_inserter()) :- This is another way to copy old vector into new one. This function takes 3 arguments, first, the first iterator of the old vector, second, the last iterator of the old vector and third is back_inserter function to insert values from the back. This also generated a deep copy.

```
// Copying vector by copy function
    copy(vect1.begin(), vect1.end(), back_inserter(vect2));
```

Method 5: assign(first_iterator_o, last_iterator_o):

This method assigns the same values to the new vector as the old one. This takes 2 arguments, the first iterator to the old vector and the last iterator to the old vector. This generates a deep copy.

```
// Copying vector by assign function
    vect2.assign(vect1.begin(), vect1.end());
```

Method 6: By using insert function. The vector class has a standard function, insert(), that can insert elements from a specified range.

```
// Copying vector by insert function
    vect2.insert(vect2.begin(), vect1.begin(), vect1.end());
```

Exceptions

Throw

```
class Calculator {
    public:
    int power(int n, int p) {
        if(n < 0 || p < 0) {
           throw runtime_error("n and p should be non-negative");
        }
        return pow(n, p);
    }
};
```

Try Catch

```
int main()
{
    string S;
    cin >> S;
    try
    {
        int no;
        no=stoi(S);
        cout<<no<<endl;
    }
    catch(exception a)
    {
        cout<<"Bad String";
    }
    return 0;
}
```

C++ The ?: Alternative to if- else

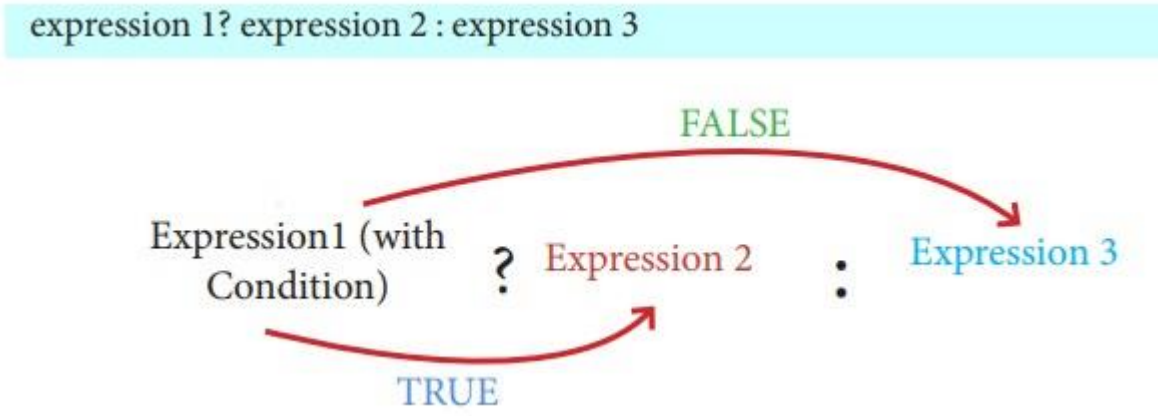
The conditional operator (or Ternary operator) is an alternative for ‘if else statement’. The conditional operator that consists of two symbols (?:). It takes three arguments.

Syntax

```
expression 1? expression 2 : expression 3
```

Ex.

```
largest = (a>b)? a : b;
```



Generics in C++

Generics is the idea to allow type (Integer, String, … etc and user-defined types) to be a parameter to methods, classes and interfaces. For example, classes like an array, map, etc, which can be used using generics very efficiently. We can use them for any type.

The method of Generic Programming is implemented to increase the efficiency of the code. Generic Programming enables the programmer to write a general algorithm which will work with all data types. It eliminates the need to create different algorithms if the data type is an integer, string or a character.

The advantages of Generic Programming are

Code Reusability

Avoid Function Overloading

Once written it can be used for multiple times and cases.

Template is a simple and yet very powerful tool in C++. The simple idea is to pass data type as a parameter so that we don't need to write the same code for different data types.

```
#include <iostream>
using namespace std;

// One function works for all data types.
// This would work even for user defined types
// if operator '>' is overloaded
template <typename T>

T myMax(T x, T y)
{
    return (x > y) ? x : y;
}

int main()
{
    // Call myMax for int
    cout << myMax<int>(3, 7) << endl;

    // call myMax for double
```

```
    cout << myMax<double>(3.0, 7.0) << endl;

    // call myMax for char
    cout << myMax<char>('g', 'e') << endl;

    return 0;
}
```

```
//reverse int x

int sum=0,temp=x;
while(temp>0){
sum=sum*10+temp%10;
temp/=10;
}
```

Queue STL

```
queue<int> g;

queue::empty()  O(1)
queue::size()   O(1)
queue::emplace()O(1)
queue::front()  O(1)
queue::back()   O(1)
queue::push(g)  O(1)
queue::pop()    O(1)
```

Roman Number to Integer

Easy

Given a string in roman no format (s) your task is to convert it to an integer . Various symbols and their values are given below.

1. Split the Roman Numeral string into Roman Symbols (character).
2. Convert each symbol of Roman Numerals into the value it represents.
3. Take symbol one by one from starting from index 0:
    - If current value of symbol is greater than or equal to the value of next symbol, then add this value to the running total.
    - else subtract this value by adding the value of next symbol to the running total.

What is the concept of taking modulo by 1000000007 (10^9 + 7)?

This is done to avoid calculations involving very large numbers.
If you take mod after the final (actual) answer it would defeat the very purpose of using the mod operator in the question.

Try this-
if you have to print (a*b*c)%m

Suppose:

a=145785635595363569532135132
b=3151635135413512165131321321
c=99987445522222200651351351
m=1000000007

now,

a*b*c=
4594054481842122908933398351488095153324400334008185667177356443070246253486015 72

(a*b*c)%m= 798848767

now use this,

i=1
i=(i*a)%m // i=508086243
i=(i*b)%m //i=144702857
i=(i*c)%m //i=798848767

i=798848767

The latter has a better complexity.
Why?
Because the complexity of multiplication of two numbers depends on the product of the number of digits in the numbers.

It also ensures that the answer remains inside a particular data type limit.

Same Number Of Set Bits As N

Approach:

1. Using __builtin_popcount() inbuilt function, count set bits in N and store into a temp variable
2. Iterate from n-1 to 1 and also count set bits in i using __builtin_popcount() function
3. Now, compare temp with __builtin_popcount(i)
4. If both are equal then increment counter variable
5. Return counter

```
int smallerNumsWithSameSetBits(int n)
{
    // __builtin_popcount function that count
    // set bits in n
    int temp = __builtin_popcount(n);

    // Iterate from n-1 to 1
    int count = 0;
    for (int i = n - 1; i > 0; i--) {

        // check if the number of set bits
        // equals to temp increment count
        if (temp == __builtin_popcount(i))
            count++;
    }
    return count;
}
```