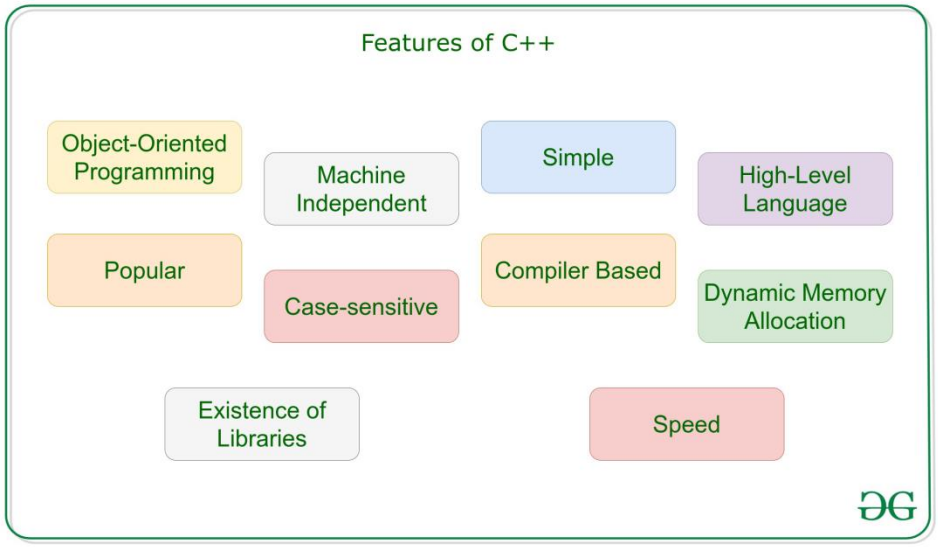
FEATURES OF C++

C++ is an upgraded version of C programming. The main idea behind creating C++ programming was to add object orientation to the C programming language. The major upgradations are object-oriented programming methodology, namespace feature, operator overloading, error & exception handling. The motivation behind object-oriented programming is to try to see the whole world in the form of classes & objects.
There are various features of C++ such as,

- Object Oriented
- Simple
- Platform Dependent
- Mid-level programming language
- Structured programming language
- Rich Library
- Memory Management
- Powerful & Fast
- Pointers
- Compiler based
- Syntax based language

Let's discuss each one of them one by one.



Machine Independent

A C++ executable is not platform-independent (compiled programs on Linux won't run on Windows), however they are machine independent. Let us understand this feature of C++ with the help of an example. Suppose you have written a piece of code which can run on Linux/Windows/Mac OSx which makes C++ Machine Independent but the executable file of the C++ cannot run on different operating systems.

Simple

It is a simple language in the sense that programs can be broken down into logical units and parts, has rich library support, and a variety of data-types. Also, the Auto Keyword of the C++ makes life easier.

The auto keyword

The idea of the auto was to form the C++ compiler deduce the data type while compiling instead of making you declare the data type every-freaking-time. Do keep in mind that you cannot declare something without an initializer. There must be some way for the compiler to deduce your type.

```
// C++ program for using auto keyword

#include <bits/stdc++.h>
using namespace std;

// Driver Code
int main()
{

// Variables
auto an_int = 26;
auto a_bool = false;
auto a_float = 26.24;
auto ptr = &a_float;

// Print typeid
cout << typeid(a_bool).name() << "\n";
cout << typeid(an_int).name() << "\n";
return 0;
}
```

Output

```
b
i
```

High-Level Language

C++ is a High-Level Language, unlike C which is a Mid-Level Programming Language. It makes life easier to work in C++ as it is a high-level language as it is closely associated with the human-comprehensible English language.

Popular

C++ can be the base language for many other programming languages that supports the feature of object-oriented programming. **Bjarne Stroustrup** found Simula 67, the first object-oriented language ever, lacking simulations and decided to develop C++.

Case-sensitive

It is clear that the C++ is a case-sensitive programming language. For example, **cin** is used to take input from the input stream. But if the **"Cin"** won't work. Other languages like HTML and MySQL are not case-sensitive language.

Compiler Based

C++ is a compiler-based language, unlike Python. That is C++ programs used to be compiled and their executable file is used to run it. Due to which C++ is a relatively faster language than Java and Python.

Dynamic Memory Allocation

When the program executes in the C++ then the variables are allocated the dynamical heap space. Inside of the functions the variables are allocated in the stack space. Many times, We are not aware in advance that how much memory is needed to store particular information in a defined variable and the size of required memory can be determined at run time.

Memory Management

C++ allows us to allocate the memory of a variable or an array in run time. This is known as Dynamic Memory Allocation.

In other programming languages such as Java and Python, the compiler automatically manages the memories allocated to variables. But this is not the case in C++.

In C++, the memory must be de-allocate dynamically allocated memory manually after it is of no use.

The allocation and deallocation of the memory can be done using the new and delete operators respectively.

```
// C++ implementation to illustrate
// the Memory Management

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

// Driver Code
int main()
{
int num = 5;
float* ptr;
// Memory allocation of
// num number of floats
ptr = new float[num];
for (int i = 0; i < num; ++i) {
*(ptr + i) = i;
}

cout << "Display the GPA of students:"
<< endl;
for (int i = 0; i < num; ++i) {
cout << "Student" << i + 1
<< ": " << *(ptr + i)
<< endl;
}
// Ptr memory is released
delete[] ptr;

return 0;
```

```
Display the
  GPA of
 students:
Student1: 0
Student2: 1
Student3: 2
Student4: 3
Student5: 4
```

```
}
```

Multi-threading

Multithreading is a specialized form of multitasking and multitasking is a feature that allows your system to execute two or more programs concurrently. In general, there are two sorts of multitasking: process-based and thread-based.

Process-based multitasking handles the concurrent execution of programs. Thread-based multitasking deals with the multiprogramming of pieces of an equivalent program.

A multithreaded program contains two or more parts that will run concurrently. Each a part of such a program is named a thread, and every thread defines a separate path of execution.

C++ doesn't contain any built-in support for multithreaded applications. Instead, it relies entirely upon the OS to supply this feature.

Below is the program to illustrate Multithreading in C++:

```
// C++ implementation to illustrate
// the working of Multi-threading

#include <cstdlib>
#include <iostream>
#include <pthread.h>

using namespace std;

#define NUM_THREADS 5

// Function to print Hello with
// the thread id
void* PrintHello(void* threadid)
{
// Thread ID
long tid;
tid = (long)threadid;

// Print the thread ID
cout << "Hello World! Thread ID, "
<< tid << endl;
```

```
pthread_exit(NULL);
}

// Driver Code
int main()
{

// Create thread
pthread_t threads[NUM_THREADS];
int rc;
int i;

for (i = 0; i < NUM_THREADS; i++) {

cout << "main() : creating thread, "
<< i << endl;

rc = pthread_create(&threads[i],
NULL,
PrintHello,
(void*)&i);

// If thread is not created
if (rc) {
cout << "Error:unable to"
<< " create thread, "
<< rc << endl;

exit(-1);
}
}

pthread_exit(NULL);
}
```

```
architdwevedi@cool:~/CP/PP$ sudo g++ -pthread multi.cpp
architdwevedi@cool:~/CP/PP$ ./a.out
main() : creating thread, 1
main() : creating thread, 2
Hello World! Thread ID, 140734220069032
main() : creating thread, 3
main() : creating thread, 4
Hello World! Thread ID, 140734220069032
Hello World! Thread ID, 140734220069032
Hello World! Thread ID, main() : creating thread, 5
140734220069032
Hello World! Thread ID, 140734220069032
```