**Cpp**

**C++ Variables**

In programming, a variable is a container (storage area) to hold data.

**Static variables in a Function**: When a variable is declared as static, space for **it gets allocated for the lifetime of the program**. Even if the function is called multiple times, space for the static variable is allocated only once and the value of variable in the previous call gets carried through the next function call.

**C++ Literals**

Literals are data used for representing fixed values.

Here's a list of different literals in C++ programming.

1. Integers

2. Floating-point Literals

3. Characters

4. String Literals

**C++ Constants**

In C++, we can create variables whose value cannot be changed.

**C++ Data Types**

In C++, data types are declarations for variables. In C++, data types are declarations for variables.

**C++ Fundamental Data Types**

The table below shows the fundamental data types, their meaning, and their sizes (in bytes):

|  |  |  |
| --- | --- | --- |
| Data Type | Meaning | Size (in Bytes) |
| int | Integer | 2 or 4 |
| float | Floating-point | 4 |
| double | Double Floating-point | 8 |
| char | Character | 1 |
| wchar\_t | Wide Character | 2 |
| bool | Boolean | 1 |
| void | Empty | 0 |

**C++ Type Modifiers**

We can further modify some of the fundamental data types by using type modifiers. There are 4 type modifiers in C++. They are:

1. signed

2. unsigned

3. short

4. long

**Derived Data Types**

Data types that are derived from fundamental data types are derived types. For example: arrays, pointers, function types, structures, etc.

**Memory allocation in c++**

 Memory in your C++ program is divided into two parts

**The stack** − All variables declared inside the function will take up memory from the stack.

**The heap** − This is unused memory of the program and can be used to allocate the memory dynamically when program runs.

You can allocate memory at run time within the heap by using new operator.

If you are not in need of dynamically allocated memory anymore, you can use **delete** operator, which de-allocates memory that was previously allocated by new operator.

#include<iostream>

Using namespace std;

Int main() {

Int a = 10; // stored in stack

Int \*p = new int(); // allocate memory in heap

\*p = 10;

delete(p); // deallocate memory in heap

p = null;

}

**Array:**

An array is **a collection of similar types of data**. For example, if we want to store the names of 100 people then we can create an array of the string type that can store 100 names.

Sting is a collection of character.

**C++ Structures**

**Structures in C++** are user defined data types which are used to store group of items of non-similar data types.

struct Person

{

char name[50];

int age;

float salary;

};

**C++ Pointers**

In C++, pointers are variables that store the memory addresses of other variables.

# **References in C++**

When a variable is declared as a reference, it becomes an alternative name for an existing variable.

Different types of Member functions:

1. Simple functions

2. Static functions

3. Const functions

4. Inline functions

5. Friend functions

**Static function**

We can define class members static using **static** keyword. When we declare a member of a class as static it means no matter how many objects of the class are created, there is only one copy of the static member. A static member is shared by all objects of the class.

A static member function can be called even if no objects of the class exist and the **static** functions are accessed using only the class name and the scope resolution operator **::**.

Example:

#include<bits/stdc++.h>

using namespace std;

class person{

   public:

   int static objcount;

};

int person :: objcount = 4;

int main()

{

    cout<<person :: objcount<<endl;

    return 0;

}

**Inline Function**

C++ provides an inline functions to reduce the function call overhead. Inline function is a function that is expanded in line when it is called. When the inline function is called whole code of the inline function gets inserted or substituted at the point of inline function call.

**Friends function**

A friend class can access private and protected members of other class in which it is declared as friend. It is sometimes useful to allow a particular class to access private members of other class.

**Const Function:**

Constant member functions are those functions which are denied permission to change the values of the data members of their class.

#include<bits/stdc++.h>

using namespace std;

class base{

  public:

  mutable int x;

  base(int a){

    x = a;

  }

  int get() const{

    x = 20;

    return x;

  }

};

int main(){

    int n = 10;

    base a(n);

    cout<<a.get()<<endl;

    return 0;

}

**This Poniter**

This pointer is **a pointer accessible only within the nonstatic member functions of a class, struct, or union type**. It points to the object for which the member function is called. Static member functions don't have a this pointer.