C++

**Introduction**

**Operators**

- Operators are the symbols that tells the compiler to perform some specific operations on the corresponding operands

**Arithmetic Operators**

1 . Binary operators : Operates on double operands , ex:- + , - , \* , / , %

2 . Unary operators : Operates on single operands : ex:- pre-incrementer(++a) , post-incrementer(a++)

**STL (Standard Template Library)**

- It is an internal implemented library which is a set of C++ template classes to provide common programming data structures and functions such as lists , stacks , arrays etc

- It is a library of container classes , algorithms and iterators

**Components of STL**

- Containers - Iterators - Algorithms - Functors

**Containers**

**-** Data structures pre-implemented in STL

Sequential Containers : Sequentially implemented

**vector**

**- it is a dynamic array .**

|  |  |
| --- | --- |
| **int a[10];** | **vector<int> a ;** |
|  |  |
|  |  |
|  |  |

- a C++ class basically defined in **<utility>** header , which is used to combine two values that may be of different datatypes together and store them as a single unit

**- pair<int , string>** **p ;** Pair Declaration with different datatypes

- p = **make\_pair(2 , “abc”);** OR p = **{2 , ”abc”};** Initialization

- **p.first** and **p.second** gives the first and second values of pair respectively

- pair is used to maintain relation between two elements

- pairs can be copied to each other

like , **pair<int , string> p1 = p ; here , p1 and p are two different copies hence , changes made in one can’t hamper another**

but if **pair<int , string> &p1 = p ;** here , p1 and p are same , change in p1 affects p

Ordered Containers : values are stored in a sorted order like ascending or descending

- Maps - Multimap - Set - Multiset

Unordered Containers

- Unordered Map - Unordered Set

Nested Containers

vector<vector<int>> map <int , vector<int>> set<pair<int, string>> vector<map<int,set<int>>>

**stack Queue lists**

**Iterators**

- More like pointers but iterators are used to point the memory-addresses of container’s elements

ex: begin() , end() etc

**vector<int> :: iterator it ;** <- way of declaring the iterators

**Algorithms**

- Contains oftenly used pre-implemented algorithms

- upper bound , lower bound

- sort (comparator)

- max-element , min-element

- accumulate , reverse , count , find

- next-permutation , prev-permutation

**Functors**

- classes which can act as functions