C++ Programming

Trainer : Rohan Paramane

Email: rohan.paramane@sunbeaminfo.com

Casting Operators

- dynamic_cast Used for conversion of polymorphic types.
  - dynamic_cast < type-id > (expression)
  - dynamic cast returns NULL if the cast to a pointer type fails
  - The **bad_cast** exception is thrown by the **dynamic_cast** operator as the result of a failed cast to a reference type.
- static_cast Used for conversion of nonpolymorphic types.
  - Risky conversion not be used, should only be used in performance-critical code when you are certain it will work correctly
  - The **static_cast** operator can be used for operations such as converting a pointer to a base class to a pointer to a derived class. Such conversions are not always safe.
- const_cast Used to remove the **const**, **volatile**, and **__unaligned** attributes.
  - const_cast<class *> (this)->membername = value;
- reinterpret_cast Used for simple reinterpretation of bits.
  - Allows any pointer to be converted into any other pointer type.
  - The **reinterpret_cast** operator can be used for conversions such as **char*** to **int***, or One_class* to Unrelated_class*, which are inherently unsafe.

Run-Time Type Information

- Run-time type information (RTTI) is a mechanism that allows the type of an object to be determined during program execution.
- There are three main C++ language elements to run-time type information:
- The dynamic_cast operator.
    - Used for conversion of polymorphic types.
- The typeid operator.
    - Used for identifying the exact type of an object.
    - typeid(Base *).name();
    - If Base ptr holds null then type_id throws bad_typeid exception
- The type_info class.
    - Used to hold the type information returned by the **typeid** operator.

STL

- The Standard Template Library (STL) 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.
- It is a generalized library and so, its components are parameterized.
- Working knowledge of template classes is a prerequisite for working with STL.

- STL has 4 components:
  - Algorithms
  - Containers
  - Functions
  - Iterators

Components of STL

1. **Algorithm**

They act on containers and provide means for various operations for the contents of the containers.

- Sorting
- Searching

2. **containers**

- Containers or container classes store objects and data.

3. **Functions**

- The STL includes classes that overload the function call operator.
- Instances of such classes are called function objects or functors.
- Functors allow the working of the associated function to be customized with the help of parameters to be passed.

4. **Iterators**

- As the name suggests, iterators are used for working upon a sequence of values. They are the major feature that allows generality in STL.
- Iterators are used to point at the memory addresses of STL containers.
- They are primarily used in sequences of numbers, characters etc.
- They reduce the complexity and execution time of the program.

vector

- Vectors are the same as dynamic arrays with the ability to resize itself automatically when an element is inserted or deleted, with their storage being handled automatically by the container.
- Vector elements are placed in contiguous storage so that they can be accessed and traversed using iterators.
- In vectors, data is inserted at the end. Inserting at the end takes differential time, as sometimes the array may need to be extended. Removing the last element takes only constant time because no resizing happens.
- Inserting and erasing at the beginning or in the middle is linear in time.
- begin() – Returns an iterator pointing to the first element in the vector
- end() – Returns an iterator pointing to the theoretical element that follows the last element in the vector
- size() – Returns the number of elements in the vector.
- empty() – Returns whether the container is empty.
- front() – Returns a reference to the first element in the vector
- back() – Returns a reference to the last element in the vector
- assign() – It assigns new value to the vector elements by replacing old ones
- push_back() – It push the elements into a vector from the back
- pop_back() – It is used to pop or remove elements from a vector from the back.
- insert() – It inserts new elements before the element at the specified position
- erase() – It is used to remove elements from a container from the specified position or range.

Thank You