# List Introduction

## What is std::list ?

std::list is sequential STL container that is internally implemented as doubly linked list.  
i.e. every element in the list is stored at a separate memory location i.e. called node and it also contains a pointer to the previous and next node.

### [Vector vs List](https://thispointer.com/difference-between-vector-and-list-in-c/)

Creating a std::list of int and pushing elements in front and back

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | std::list<int> listOfNumbers;    //Inserting elements at end in list  listOfNumbers.push\_back(5);  listOfNumbers.push\_back(6);    //Inserting elements at front in list  listOfNumbers.push\_front(2);  listOfNumbers.push\_front(1); |

List Contents are now,  
1 2 5 6

### [Different ways to Initialise a List](https://thispointer.com/different-ways-to-initialize-a-list-in-c/)

## Inserting elements in the std::list

std::list has several overloaded member function insert() to insert the elements in between the list. One is,

|  |  |
| --- | --- |
| 1 | insert(iterator\_position,elem) |

It allocates a new node and copy the passed element to it and then inserts that node before iterator position iterator\_position and returns the position of the newly added element.

It has several other overloaded implementations i.e.

|  |  |
| --- | --- |
| 1 | insert(iterator\_position,other\_list\_begin,other\_list\_end) |

etc.

Now lets insert an element at 3rd position in the list.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | //Inserting elements in between the list using  // insert(pos,elem) member function. Let's iterate to  // 3rd position  it = listOfNumbers.begin();  it++;  it++;  // Iterator 'it' is at 3rd position.  listOfNumbers.insert(it, 4); |

Now List contents will be,  
1 2 4 5 6

### [C++ : How to get element by index in List](https://thispointer.com/c-how-to-get-element-by-index-in-list/)

### [How to search an element in std::list ?](https://thispointer.com/c-list-find-contains-how-to-search-an-element-in-stdlist/)

Iterating over a std::list of elements and display them

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | // Iterating over list elements and display them  it = listOfNumbers.begin();  while(it != listOfNumbers.end())  {  std::cout<<(\*it)<<"  ";  it++;  }  std::cout<<std::endl; |

### [Different Ways to iterate over a std::list of objects](https://thispointer.com/c-different-ways-to-iterate-over-a-list-of-objects/)

## Erasing elements from std::list

std::list has two overloaded member function erase() to erase the elements in between the list. These are,

|  |  |
| --- | --- |
| 1 | erase(iterator\_position,elem) |

It erases the element at the passed node and change the left and right pointers of previous and next nodes. Also returns iterator of next node element. This opeartion takes O(1) time, as only swapping of pointers is required not like vector and deque, where other elements need to be shifted.

It has an other overloaded implementations i.e.

|  |  |
| --- | --- |
| 1 | erase(other\_list\_begin,other\_list\_end) |

It erases the elements in given range.

Now lets erase an element at 3rd position in the list.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | //Erasing elements in between the list using  // erase(position) member function. Let's iterate to  // 3rd position  it = listOfNumbers.begin();  it++;  it++;  // Iterator 'it' is at 3rd position. Now erase this element.  listOfNumbers.erase(it); |

Now List contents will be,  
1 2 5 6

### [Erase elements from a List using iterators](https://thispointer.com/how-to-erase-elements-from-a-list-in-c-using-iterators/)

### [Remove Elements from std::list while Iterating](https://thispointer.com/how-to-remove-elements-from-a-list-while-iterating/)

## Removing elements from a std::list based on some condition

::remove(value) member function of std::list removes all the elements whose value is equivalent to passed value.  
::remove\_if(predicate) member function of std::list removes all the elements for which predicate(element) returns true.

Lets remove all elements with value greater than 3.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | //Lets remove all elements with value greater than 3.  listOfNumbers.remove\_if([](int elem)  {  if(elem > 3)  return true;  else  return false;  }); |

Now List contents will be,  
1 2

### [Remove elements from std::list based on External Criterion](https://thispointer.com/how-to-remove-elements-from-a-list-based-on-value/)

Complete executable code is as follows,

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82 | #include <iostream>  #include <list>  #include <iterator>      int main()  {  std::list<int> listOfNumbers;    //Inserting elements at end in list  listOfNumbers.push\_back(5);  listOfNumbers.push\_back(6);    //Inserting elements at front in list  listOfNumbers.push\_front(2);  listOfNumbers.push\_front(1);    // Iterating over list elements and display them  std::list<int>::iterator it = listOfNumbers.begin();  while(it != listOfNumbers.end())  {  std::cout<<(\*it)<<"  ";  it++;  }  std::cout<<std::endl;      //Inserting elements in between the list using  // insert(pos,elem) member function. Let's iterate to  // 3rd position  it = listOfNumbers.begin();  it++;  it++;  // Iterator 'it' is at 3rd position.  listOfNumbers.insert(it, 4);      // Iterating over list elements and display them  it = listOfNumbers.begin();  while(it != listOfNumbers.end())  {  std::cout<<(\*it)<<"  ";  it++;  }  std::cout<<std::endl;      //Erasing elements in between the list using  // erase(position) member function. Let's iterate to  // 3rd position  it = listOfNumbers.begin();  it++;  it++;  // Iterator 'it' is at 3rd position. Now erase this element.  listOfNumbers.erase(it);      // Iterating over list elements and display them  it = listOfNumbers.begin();  while(it != listOfNumbers.end())  {  std::cout<<(\*it)<<"  ";  it++;  }  std::cout<<std::endl;  //Lets remove all elements with value greater than 3.  listOfNumbers.remove\_if([](int elem) { if(elem > 3)  return true;  else  return false;  });    it = listOfNumbers.begin();  while(it != listOfNumbers.end())  {  std::cout<<(\*it)<<"  ";  it++;  }  std::cout<<std::endl;    return 0;  } |