**Set/ Multiset Built-in Functions:**

1. **Constructor**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Details** | | **Time C** |
| Constructor | // Constructor examples      set*<*int*>* s1; // Default constructor      set*<*int*>* s2 *=* {1, 2, 3, 4}; // Initializer list constructor      set*<*int*>* s3(s2); // Copy constructor      set*<*int*>* s4(s2.begin(), s2.end()); // Range constructor | |  |
| **Assignment** | Copy Assignment | set*<*int*>* s1 *=* {1, 2, 3, 4, 5};  set*<*int*>* s2;  s2 *=* s1; |  |
| Move Assign | set*<*int*>* s3 *=* {6, 7, 8, 9, 10};  set*<*int*>* s4;  s4 *=* move(s3); |
| Initializer List Assign | set*<*int*>* s5;  s5 *=* {11, 12, 13, 14, 15}; |
| Iterators |  | set*<*int*>* s *=* {1, 2, 3, 4, 5}; |  |
|  | // begin() and end() || output: 1 2 3 4 5  *for* (auto it*=* s.begin(); it *!=* s.end(); *++*it)cout*<<\**it; |
|  | // cbegin() and cend() || output: 1 2 3 4 5  *for*(auto it*=* s.cbegin(); it *!=* s.cend(); *++*it)cout*<<\**it; |
|  | // rbegin() and rend() || output: 5 4 3 2 1  *for*(auto it*=* s.rbegin(); it *!=* s.rend(); *++*it)cout*<<\**it; |
|  | // begin() and end() || output: 5 4 3 2 1  *for*(auto it*=* s.crbegin(); it*!=*s.crend(); *++*it)cout*<<\**it; |
| Capacity | empty() | (s.empty())*?* cout*<<*"empty" *:* cout*<<*"not empty"; |  |
| size() | cout *<<* s.size(); |
| max\_size() | cout *<<* s.max\_size(); |
| Modification | insert() | set*<*int*>* s;  s.insert(1);  int val = 10;  s.insert(move(val));  set*<*int*>* s2 *=* {7, 8, 9};  s.insert(s2.begin(), s2.end()); |  |
| erase() | auto it *=* s.find(4);  *if* (it *!=* s.end()) s.erase(it);  s.erase(3); // erase t value |
| set*<*int*>* s *=* {1, 3, 4,5,6,9};  auto it1 *=* s.find(3), it2 *=* s.find(6);  *if*(it1*!=*s.end() *&&* it2*!=*s.end())s.erase(it1, it2);  output: 1 6 9 |
| explace() | s.emplace(4); |
| swap() | set*<*int*>* s *=* {1, 2, 3}; set*<*int*>* s3 *=* {14, 15, 16};  s.swap(s3);  output: s = 14 15 16 , s3 = 1 2 3 |
|  |  | |  |

1. **Look UP**

|  |  |  |
| --- | --- | --- |
| **Name** | **Details** | **Time Complexity** |
| count() | int cnt5 *=* s.count(5);  cout *<<*cnt5;  how many times 5 is in set , |  |
| lower\_bound() | // lower\_bound(const T& value)      auto lb *=* s.lower\_bound(5);  *if* (lb *!=* s.end())  cout *<<* *\**lb; |  |
| upper\_bound() | // upper\_bound(const T& value)    auto ub *=* s.upper\_bound(5);  *if* (ub *!=* s.end())   cout *<<* *\**ub; |  |
| equal\_range() | // equal\_range(const T& value)  auto range *=* s.equal\_range(5);  cout*<<*(range.first *!=* s.end() *?* *\**range.first *:* 0) *<<*" "  *<<* (range.second *!=* s.end() *?* *\**range.second *:* 0) *<<* endl;  equal\_range হল একটি ফাংশন যা একটি সেটের মধ্যে একটি নির্দিষ্ট মানের উপাদান খুঁজে বের করে এবং দুটি ইটারেটর প্রদান করে:   1. প্রথম ইটারেটরটি হল lower\_bound এর মত, যা সেটে প্রথম উপাদান খুঁজে বের করে যা নির্দিষ্ট মানের চেয়ে ছোট নয়। 2. দ্বিতীয় ইটারেটরটি হল upper\_bound এর মত, যা সেটে প্রথম উপাদান খুঁজে বের করে যা নির্দিষ্ট মানের চেয়ে বড়। |  |

1. **Ovservar**

|  |  |  |
| --- | --- | --- |
| **Name** | **Details** | **Time Complexity** |
| Key\_camp() | set*<*int*>* s *=* {1, 2, 3, 4, 5};  auto key\_comp *=* s.key\_comp();  int highest *=* *\**s.rbegin();  *for* (int c : s){  *if* (key\_comp*(*c, highest*)*)  cout *<<* c *<<* " is less than "*<<*highest*<<*endl;  *else*  cout *<<* c *<<*"is equal to or greater than "*<<*highest*<<*endl;  } | 1 is less than 5  2 is less than 5  3 is less than 5  4 is less than 5  5 is equal to or greater than 5 |
| Compare value | set*<*int*>* s *=* {1, 2, 3, 4, 5};  int highest *=* *\**s.rbegin();  auto value\_comp *=* s.value\_comp();  *for* (int c : s)  {  *if* (value\_comp*(*c, highest*)*)  cout *<<* c *<<* " is less than " *<<* highest*<<*endl;  *else*  cout *<<* c *<<* " is equal to or greater than " *<<*highest*<<*endl;  } | 1 is less than 5  2 is less than 5  3 is less than 5  4 is less than 5  5 is equal to or greater than 5 |
| operator | set*<*int*>* s1 *=* {1, 2, 3, 4, 5};  set*<*int*>* s2 *=* {6, 7, 8, 9, 10};  cout *<<* (s1 *==* s2) *?* "is equeal" *:* "is not equal";  cout*<<* (s1 *<* s2) *?* "s1 is less than s2" *:* "s1 is greater than s2";  swap(s1, s2); |  |

1. **Element access**

|  |  |  |
| --- | --- | --- |
| **Name** | **Details** | **Time Comp** |
|  |  |  |
|  |  |  |