SET

Set is implemented by Red-Black Tree

In gnu , the header is stl\_tree.h

|  |
| --- |
| template<typename \_Key, typename \_Val, typename \_KeyOfValue,  typename \_Compare, typename \_Alloc = allocator<\_Val> >  class \_Rb\_tree  the class has most important member:\_Rb\_tree\_impl<\_Compare> \_M\_impl;  the class is an inner class  struct \_Rb\_tree\_impl : public \_Node\_allocator  {  \_Key\_compare \_M\_key\_compare;  \_Rb\_tree\_node\_base \_M\_header;  size\_type \_M\_node\_count; // Keeps track of size of tree.  ….  Many operations on tree are implemented by this **impl** class  Initialized as  \_M\_header |

In set definition, it's the member

\_Rep\_type \_M\_t; //\_Rep\_type is \_Rb\_tree above.

As for node

|  |
| --- |
| enum \_Rb\_tree\_color { \_S\_red = false, \_S\_black = true };  struct \_Rb\_tree\_node\_base  {  typedef \_Rb\_tree\_node\_base\* \_Base\_ptr;  typedef const \_Rb\_tree\_node\_base\* \_Const\_Base\_ptr;  \_Rb\_tree\_color \_M\_color;  \_Base\_ptr \_M\_parent;  \_Base\_ptr \_M\_left;  \_Base\_ptr \_M\_right; |

M\_begin() -> \_header->Parent

M\_end() -> \_header

Begin() -> header->left

template<typename \_Value,

typename \_Hash = hash<\_Value>,

typename \_Pred = std::equal\_to<\_Value>,

typename \_Alloc = std::allocator<\_Value>,

typename \_Tr = \_\_uset\_traits<\_\_cache\_default<\_Value, \_Hash>::value>>

using \_\_uset\_hashtable = \_Hashtable<\_Value, \_Value, \_Alloc,

\_\_detail::\_Identity, \_Pred, \_Hash,

\_\_detail::\_Mod\_range\_hashing,

\_\_detail::\_Default\_ranged\_hash,

\_\_detail::\_Prime\_rehash\_policy, \_Tr>;