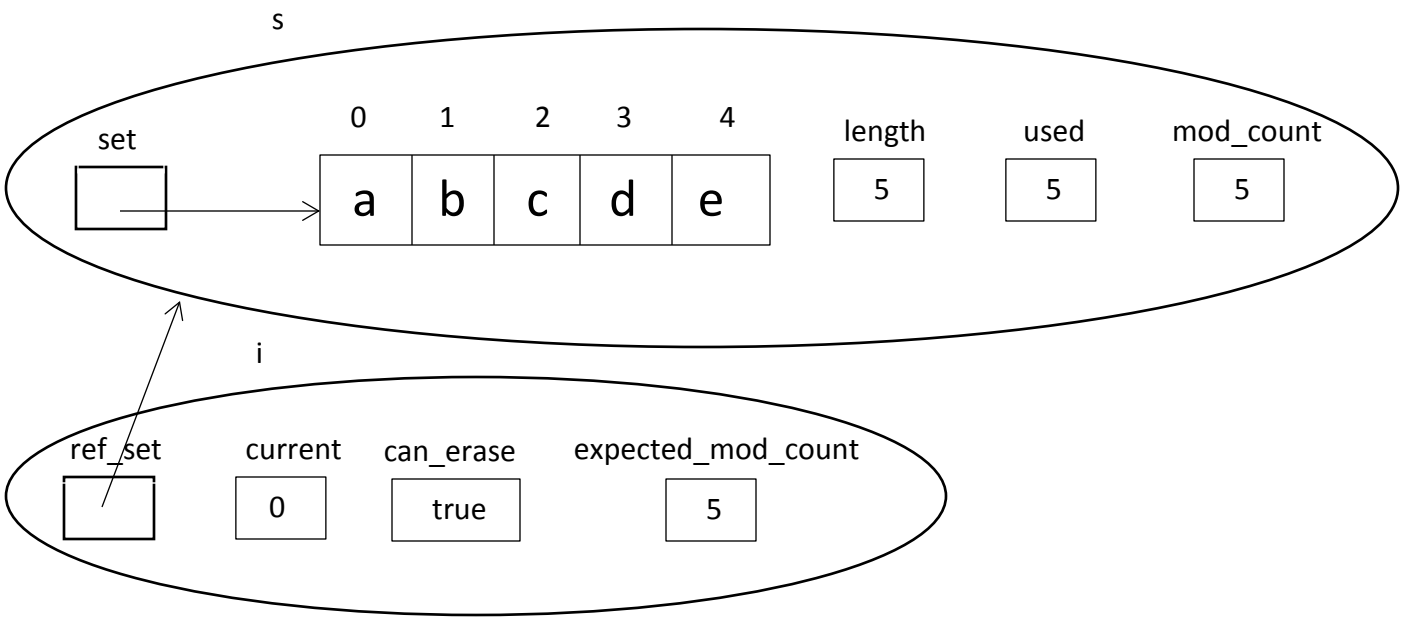


```
ics::ArraySet<std::string>::Iterator
  i =s.begin();
std::cout << *i << std::endl;
++i;
std::cout << *i << std::endl;
i.erase();
++i;
std::cout << *i << std::endl;
++i;
std::cout << *i << std::endl;
i.erase();
++i;
std::cout << *i << std::endl;
```

```
template<class T>
auto ArraySet<T>::Iterator::operator ++ () ->
  ArraySet<T>::Iterator& {
...
  if (current >= ref_set->used)
    return *this;

  if (can_erase)
    ++current;
  else
    can_erase = true;  //current already indexes
                       //"1 beyond" erased value

  return *this;
}
```

```
template<class T>
int ArraySet<T>::erase_at(int i) {
  set[i] = set[--used];
  ++mod_count;
  return 1;
}
```

```
template<class T>
T ArraySet<T>::Iterator::erase() {
   ...
  can_erase = false;
  T to_return = ref_set->set[current];
  ref_set->erase_at(current);
  expected_mod_count = ref_set->mod_count;
  return to_return;
}
```

```
template<class T>
T& ArraySet<T>::Iterator::operator *() const {
...
  return ref_set->set[current];
}
```