Container Classes

- Container Class
- typedef versus alias
- BagFixed Class
- SequenceFixed Class
- Standard Template Library (STL)

Container Class

- A container class is an ADT capable of holding a collection of items
- C++ implements containers as classes with member functions to:
  - Add item
  - Remove item
  - Examine item
- Container class *should* be capable of holding *any* type of item

BagFixed

- Container that holds fixed number of data items in *any* order with *duplicates allowed*
  - Initially empty
  - Add items to bag
    - If room available
  - Remove items from bag
    - If item in bag
  - Check number of items
    - Count number of items
  - Check number of specific item occurrences
    - Check for item equality

Container Data Items

- Manage item data types with <u>typedef</u>
  - Synonym for an existing type
    - Often used to
      - create shorter, or more meaningful names, for types already defined
      - create 'generic' data types whose underlying details can change with new compilation
      - hide platform specific details such as data type byte differences
    - *Does not* introduce new types
    - *Cannot* change meaning of existing types
    - Usable within its defined scope

typedef Declaration

- Example

```
--format--
typedef existing_type alias_to_type;

--example--
typedef int value_type;
```

```
--instead of--
int myValue;

--use--
value_type myValue;
```

alias Declaration

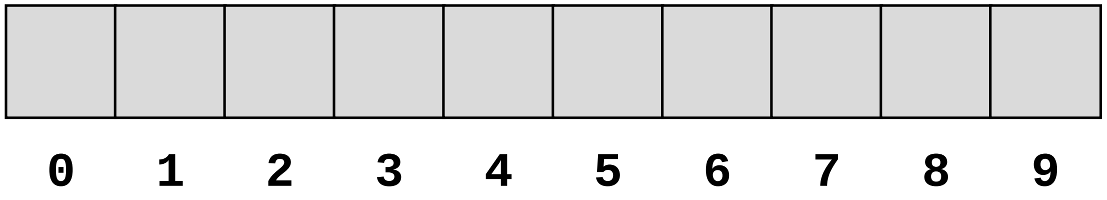
- Introduced with C++ 11 standard to overcome typedef limitations with templates
- Simplest form equivalent to C++ 03 typedef

```
--format--
using alias_to_type = existing_type;

--example--
using value_type = int;
```

```
--instead of--
int myValue;

--use--
value_type myValue;
```

Container Data Items

- Manage number of data items with partially filled array
  - Fixed size
  - Elements used <= fixed size
  - Index < fixed size



Invariant of a Class

- Invariant defined as property that remains unchanged during object transformations
- Class containers must define rules required for correct class implementation
  - Member functions *depend* on valid invariant when called
  - Member functions *ensure* invariant is valid when complete
- Critical part of class implementation but no effect on class use

BagFixed Class

- Implemented as C++ Class
- Rules for implementation
  - Bag *items* stored in array member variable **`data`**
  - *Number* of items stored in member variable **`used`**
  - *Relevant* items stored in **`data[0]`** to **`data[used-1]`**
    - Contents of other **`data`** array elements not important
- Item data type
  - Must have defined operations for = , ==, and !=
  - If class, must have default constructor
- Static member constant for **`CAPACITY`**

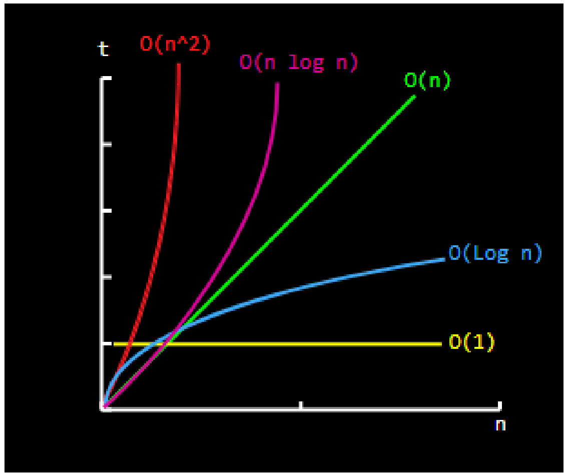
BagFixed Class

- Member functions
  - Default constructor → create empty bag
  - size() → return number of items in bag
  - count → count number of item occurrences
  - insert → insert an item, if space available
  - erase_one → remove an item, if found
  - += → copy items from one bag to another
- Non-member function
  - + → create new **`bagFixed`** object from two added **`bagFixed`** objects

bagFixed Class Analysis

- O(n) : linear time → time required by function depends upon size of input
- O(1) : constant time → time required by function *does not* depend on size of input

bagFixed Operation	**Time Analysis**
default constructor	O(1)
count	O(n)
insert	O(1)
erase_one	O(n)
+= bagFixed	O(n)
b1 + b2	$O(n_1 + n_2)$



SequenceFixed Class

- Container that holds fixed number of data items in *sequential* order with *duplicates allowed*
- Rules for implementation
  - Bag *items* stored in array member variable `data`
  - *Number* of items stored in member variable `used`
  - *Relevant* items stored in `data[0]` to `data[used-1]`
    - Contents of other `data` array elements not important
  - Current item in sequence is in `data[current_index]`
    - No current item if `current_index = used`

sequence internal iterator

SequenceFixed Class

- Member functions
  - Default constructor → create empty sequence
  - size() → return number of items in sequence
  - is_item() → boolean indicates if `current_index` is valid

SequenceFixed Class

- In order retrieval of container items enforced through *member* functions:
  - start() → position iterator at beginning of sequence
  - current() → return current item in sequence
  - advance() →position iterator at next item in sequence, if `current_index` is valid

```
for (numbers.start(); numbers.is_item(); numbers.advance())
   cout << numbers.current() << endl;
```

SequenceFixed Class

- Additional *member* functions
  - insert → places new item before current
    - new item becomes current
  - attach → places new item after current
    - new item becomes current
  - remove_current() → current item is removed
    - item after removed becomes current, if valid

Standard Template Library (STL)

- Software library of common C++ classes which began as a generic programming initiative first released by HP in 1994
- STL includes
  - Containers
  - Iterators
  - Algorithms
  - Functions

STL Multiset Class

- STL template associative container that allows duplicates
  - #include <set>
  - Specify data type upon variable creation
    - multiset<data_type> ms_name;
  - < operator must be defined for data_type
- Container size limited by amount of memory available
- External iterators enable traversal of all items in multiset
  - const_iterator prevents changing items in container to which it refers

STL Copy Algorithm

- Function to easily copy items from one location to another
  - `#include <algorithm>`

left-inclusive pattern
copies elements in range [begin, end)

- Usage:

```
copy(<begin>, <end>, <dest>);
```

  - Copies items from source <begin> to, but not including, <end> to the target <dest>
- External iterators used to identify locations