A brief collection of STL

Compiled by:
Soumit Salman Rahman
BUET Transient
e-mail: soumit.s.rahman@gmail.com
Frequently used container classes of STL:

1. vector
2. queue
3. string
4. stack
5. map
6. list
7. set
8. pair

**vector Class:**

**Including:**

```cpp
#include<vector>
using namespace std;
```

**Declaration:**

```cpp
vector<type> name;
```

- `type`: int, char, float, double, string, pair or any other basic variable/class/structure.
- `name`: variable name ➔ any name you want to use.

**Example:**

```cpp
vector<int> arr; /* an integer array */
```

**Frequently used functions of vector class:**

- `clear()`
- `empty()`
- `push_back(type val)`
- `begin()`
- `end()`
- `size()`
- `[]` /* this is an overloaded operator used for indexing: using process ➔ arr[i] the i-th element */
- `erase(iterator i)` /* another overloaded version available */
- `insert(iterator i, type val)` /* 2 more overloaded version available */
queue Class:

Including:

```cpp
#include<queue>
using namespace std;
```

Declaration:

```cpp
queue<type> name;
```

Example: `queue<int> q;`

Frequently used functions:

- push(type val)
- front() 
- pop()
- empty()

stack Class:

Including:

```cpp
#include<stack>
using namespace std;
```

Declaration:

```cpp
stack<type> name;
```

Example: `stack<int> stk;`

Frequently used functions:

- push(type val)
- top()
- pop()
- empty()
string Class:

Including:

```cpp
#include<string>
using namespace std;
```

Declaration:

```cpp
string name;
```

Input / Output:

```cpp
cin>>str;
cout<<str; /*you are to include the iostream header file for this*/
```

Frequently used functions:

- size()
- + /*overloaded operator used to concatenate string, character array or character*/
- empty()
- ==, <, > /*overloaded operator for comparing*/
- =, += /*overloaded operator for assigning: you can use another string object or a character array on the right side*/
- push_back(char ch)
- insert() /*same as vector insert*/
- erase() /*same as vector erase*/
- substr(int start_index, int len)

set Class:

Including:

```cpp
#include<set>
using namespace std;
```

Declaration:

```cpp
set<type> name;
```

Example: set<string> s;

Frequently used functions:

- insert(type val) /*2 other overloaded functions available*/
- find(type val)
- erase() /*2 other overloaded functions available*/
- empty()
- size()
- begin(), end()

map Class:

Including:

#include<map>
using namespace std;

Declaration:

map<type1, type2> name;

Example: map<string, long> m;

Frequently used functions:

- insert(pair<type1, type2> (key_element, value_element)) /*overloaded version available*/
- find(type1 val), count(type1 val)
- clear()
- size()
- erase(iterator it) /*overloaded version available*/
- begin(), end()

list Class:

Including:

#include<list>
using namespace std;

Declaration:

list<type> name;

Example: list<point> l; /*there is not class or structure named point so you have to declare it*/

Frequently used functions:

- push_back(type val)
- push_front(type val)
- pop_back()
- pop_front()
- sort()
- unique()
- remove(type val) /*this is not same as erase*/
Algorithm header file:

Algorithm header file has a huge collection of commonly used functions. This is not a container class rather it is a collection of functions

Including:

```cpp
#include<algorithm>
using namespace std;
```

Frequently used function:

- sort(iterator first, iterator last) /*overloaded function available*/
- unique(iterator first, iterator last)
- equal(iterator first1, iterator last1, iterator first2)
- count(iterator first, iterator last, type val)
- find( iterator first, iterator last, type val)
- binary_search(iterator first, iterator last, type val)
- for_each(iterator first, iterator last, function_name)
- transform( iterator source_first, iterator source_last, iterator destination_first , unary function_name)
- swap(type var1, type var2)
- next_permutation( iterator first, iterator last)
- prev_permutation( iterator first, iterator last) /*there is a bug in Visual Studio implementation but that bug is not present in Linux implementation. To remove the bug there was a discussion in but_contest_teams group*/
- reverse( iterator first, iterator last ) /* Sanny learnt STL just so that he doesn’t have to write the reverse function explicitly :-S */
- fill(iterator first, iterator last, type val)
- push_heap() , pop_heap()

stdlib.h Header file:

there are 2 functions from this library, those I frequently use. Those are qsort() and bsearch(). Some of you might have had difficulty in implementing the calling process of these functions. The way I use them are given below (using double array starting from index of zero):

double arr[SIZE];
int size;
/*
  other codes
*/

/* you have to declare a comparison function it can be of any name it should return any signed
  integer */

int sort_func(int *a, int *b)
{
  if(*a < *b)
    return -1;
  if(*a > *b)
    return 1;
  return 0;
}

/*now some where in the code in some function */
qsort(arr, size, sizeof(arr[0]), (int(*)(const void *, const void *))sort_func);
bsearch(key_value, arr, size, (arr[0]), (int(*)(const void *, const void *))sort_func); //returns ptr

Extra Talks:
I wrote “using namespace std;” after the inclusion of the container classes and the algorithm
header file does not mean you have to write them after every “#include”. Include the necessary files and
then write
“using namespace std;”, something like

#include<cstdio>
#include<iostream>
#include<vector>
#include<string>
#include<queue>
#include<set>
#include<algorithm>
#include<map>
#include<list>
using namespace std;

now STL is not something that you can swallow in a single day. It is some about practice. The more you
implement these stuffs the more you will get comfortable with these. I just wrote the names not the
implementations. So it is better to check the applications from any reference or source codes.

Some references those I use:
  - Complete Reference C++ - by Herbert Schildt
  - MSDN (you can see the implementations, those are really a good )
  - www.sgi.com/tech/stl/

I wrote only a small fraction of the available functions. There are lot more functions available.