Algorithms

Algorithm for sort in ascending

```
#include<iostream>
#include<string>
#include<vector>
#include<algorithm>
using namespace std;
int main()
{
	vector<int> test;
	test.push_back(1);
	test.push_back(8);
	test.push_back(4);
	vector<int>::iterator itr1,itr2;
	itr1 = test.begin();
	itr2 = test.end();
	sort(itr1, itr2);
	for (int i = 0; i < test.size(); i++)
	{
		cout << test[i];
	}
	system("pause");
	return 0;
}
```

Algorithm for sort in descending

```
#include<iostream>
#include<string>
#include<vector>
#include<algorithm>
using namespace std;
bool wayToSort(int i, int j) { return i > j; }
int main()
{
	vector<int> test;
	test.push_back(1);
	test.push_back(8);
	test.push_back(4);
	vector<int>::iterator itr1,itr2;
	itr1 = test.begin();
	itr2 = test.end();
	sort(itr1, itr2,wayToSort);
	for (int i = 0; i < test.size(); i++)
	{
		cout << test[i];
	}
	system("pause");
	return 0;
}
```

Searching a Vector using Algorithm

```
#include<iostream>
#include<string>
#include<vector>
#include<algorithm>
using namespace std;
```

```
int main()
{
	vector<int> test;
	test.push_back(1);
	test.push_back(8);
	test.push_back(4);
	vector<int>::iterator itr;
	int k = 8;
	itr = find(test.begin(), test.end(), k);
	if (itr == test.end())
	{
		cout << "element not found";
	}
	else
	{
		cout << "element found";
	}
	system("pause");
	return 0;
}
```

Searching for an element based on certain constraint [searching if an element greater than 4 exists using find_if]

```
#include<iostream>
#include<string>
#include<vector>
#include<algorithm>
using namespace std;
bool greater(int i)
{
	return i > 4;
}
int main()
{
	vector<int> test;
	test.push_back(1);
	test.push_back(8);
	test.push_back(4);
	vector<int>::iterator itr;
	itr = find_if(test.begin(), test.end(), greater);
	if (itr == test.end())
	{
		cout << "element not found";
	}
	else
	{
		cout << "element found";
	}
	system("pause");
	return 0;
}
```

Counting number of elements in a vector

```
#include<iostream>
#include<string>
#include<vector>
#include<algorithm>
using namespace std;
int main()
```

```
{
	vector<int> test;
	test.push_back(1);
	test.push_back(8);
	test.push_back(4);
	test.push_back(1);
	vector<int>::iterator itr;
	int ct = count(test.begin(), test.end(), 1);
	cout << ct;
	system("pause");
	return 0;
}
```

Counting based on a condition [condition is number of elements > 1]

```
#include<iostream>
#include<string>
#include<vector>
#include<algorithm>
using namespace std;
bool greater(int i)
{
	return i > 1;
}
int main()
{
	vector<int> test;
	test.push_back(1);
	test.push_back(8);
	test.push_back(4);
	test.push_back(1);
	vector<int>::iterator itr;
	int ct = count_if(test.begin(), test.end(), greater);
	cout << ct;
	system("pause");
	return 0;
}
```

Use of ACCUMULATE

```
#include<iostream>
#include<string>
#include<vector>
#include<algorithm>
#include<numeric>
using namespace std;
int main()
{
	vector<int> test;
	test.push_back(1);
	test.push_back(8);
	test.push_back(4);
	test.push_back(1);
	vector<int>::iterator itr;
	int sum = accumulate(test.begin(), test.end(), 0.0);
	cout << sum;
	system("pause");
	return 0;
}
```