Polja vs vektori

Popuniti jednodimenzionalno polje od 100 elemenata nulama. Ispisati polje.

Popuniti vektor od 100 elemenata nulama. Ispisati ga.

```
#include <iostream>
#include <vector>

using namespace std;

int main() {
  vector <int> v(100,0);
  vector<int>::iterator i;
  for (i  = v.begin(); i != v.end(); ++i) {
        cout << *i << " ";
  }
  cout << endl;
  return 0;
}
```

Popuniti vektor od 100 elemenata slovom x. Ispisati ga.

```
#include <iostream>
#include <vector>

using namespace std;

int main() {
vector<char> v(10,’x’);

for (vector<char>::iterator i=v.begin(); i!=v.end();++i) { cout<<*i<<" ";}

cout << endl;
return 0;
}
```

Učitati riječ (bez razmaka). Ispisati riječ naopako. Provjeriti da li je palindrom.

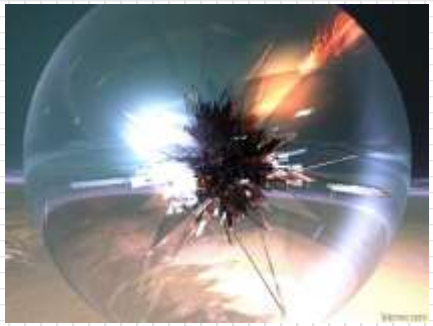
Učitati riječ (bez razmaka). Uz pomoć vektora ispisati riječ naopako i provjeriti da li je palindrom.

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

int main(void) {
	string ime;
	cin >> ime;
	vector<char> vector1(ime.begin(), ime.end()), vector2=vector1;
	vector<char>::iterator i;
	reverse(vector1.begin(), vector1.end());
	cout << "\nPreokrenuta rijec je: ";
	for (i = vector1.begin(); i != vector1.end(); i++) cout << *i;
	(vector1==vector2) ? cout << "\nPalindrom.\n“ : cout << "\nNije palindrom.";
	return 0;}
```

Primjeri inicijalizacije.
Ispisati označene strukture na ekran.

```
#include <vector>
#include <deque>
#include <list>
#include <string>
```



```
vector<int>v1; list<string>l1; deque<float>d1;

vector<int> v2(100);
deque<int> d2(5);
vector<int> v3(100, -5);
vector<string> v4(4, "dog");
list<string> l2(5, "cat");
string s1("Hello World");
vector<char> v5(s1.begin(), s1.end());
list<int> l3(v3.begin(), v3.end());
// deque<int> d3(v3.begin(), v3.end());
int numbers[5] = {9, 3, 2, 5, 6};
vector<int> v6(numbers, numbers + 3);
vector<int> v7(v6);
list<int> l4(l3);
deque<int> d4(d2);
```

Učitati n i zatim n cijelih brojeva, te jedan cijeli broj čiju poziciju u nizu želimo naći. Ispisati poz.

Učitati n i zatim n cijelih brojeva, te jedan cijeli broj čiju poziciju u nizu želimo naći. Ispisati poz.

(uz pomoć vektora)

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

int main() {
	int n, broj;
	vector<int> v1;
	vector<int>::iterator pos;
	cin >> n;
	for (int j=1;j<=n;j++) {cin >> broj; v1.push_back(broj);}
	cin >> broj;
	pos = find(v1.begin(), v1.end(), broj);
	(pos == v1.end())?cout<< "nema ":cout << "na poziciji "<<pos-v1.begin()+1<< endl;
	return 0;
}
```

Učitati n i zatim n znakova (bez praznina), te jedan znak čiju poziciju u nizu želimo naći. Ispisati poziciju (gledano od kraja niza).

Učitati n i zatim n slova, te jedno slovo čiju prvu poziciju u nizu želimo naći. Ispisati poz.

(uz pomoć vektora)

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

int main() {
	int n;
	char slovo;
	vector<char> v1;
	vector<char>::iterator pos;
	cin >> n;
	for (int j=1;j<=n;j++){ cin >> slovo; v1.push_back(slovo);}
	cin >> slovo;
	pos = find(v1.begin(), v1.end(), slovo);
	(pos == v1.end())?cout<<"nema ":cout <<"na poziciji "<<pos-v1.begin()+1 << endl;
	return 0;}
```

Učitati n i zatim n slova, te slovo za koje želimo izbrojiti koliko puta se u nizu nalazi. Ispisati broj pojav.

(uz pomoć vektora)

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

int main() {
	int n, bslova;
	char slovo;
	vector<char> v1;
	cin >> n;
	for (int j=1;j<=n;j++){ cin >> slovo; v1.push_back(slovo);}
	cin >> slovo;
	bslova=count(v1.begin(), v1.end(), slovo);
	cout << "Slovo " << slovo << " pojavljuje se " << bslova << " puta.\n";
return 0;}
```

Učitati n i zatim n cijelih brojeva. Ispisati učitani niz brojeva sortirano od najvećeg prema najmanjem.

Učitati n i zatim n cijelih brojeva. Ispisati učitani niz brojeva sortirano od najmanjeg prema najvećem.

(uz pomoć vektora)

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

int main() {
	int n, broj;
	vector<int> v1;
	cin >> n;
	for (int j=1;j<=n;j++){ cin >> broj; v1.push_back(broj);}
	ostream_iterator<int> output(cout, " ");
	sort(v1.begin(), v1.end());
	copy(v1.begin(), v1.end(), output);
return 0;}
```

Učitati n i zatim n cijelih brojeva. Ispisati učitani niz brojeva sortirano od najvećeg prema najmanjem.

(uz pomoć vektora)

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

int main() {
	int n, broj;
	vector<int> v1;
	cin >> n;
	for (int j=1;j<=n;j++){ cin >> broj; v1.push_back(broj);}
	ostream_iterator<int> output(cout, " ");
	sort(v1.begin(), v1.end());
	reverse(v1.begin(), v1.end());                    //ili sa greater<int> ()
	copy(v1.begin(), v1.end(), output);
return 0;}
```

Učitati n i zatim n imena. Ispisati učitani niz imena sortirano.

(uz pomoć vektora)

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

int main() {
	int n;
	string ime;
	vector<string> v1;
	cin >> n;
	for (int j=1;j<=n;j++){ cin >> ime; v1.push_back(ime);}
	ostream_iterator<string> output(cout, "\n");
	sort(v1.begin(), v1.end());
	copy(v1.begin(), v1.end(), output);
return 0;}
```

Učitati n i zatim n cijelih brojeva. Ispisati učitani niz brojeva sortirano tako da svaki broj bude samo jedan.

Učitati n i zatim n cijelih brojeva. Ispisati učitani niz brojeva sortirano tako da svaki broj bude samo jedan.

(uz pomoć vektora)

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

int main() {
        vector<int> v1;
        int n, broj;
        vector<int>::iterator it1;
        cin >> n;
        for (int j=1;j<=n;j++){ cin >> broj; v1.push_back(broj);}
        ostream_iterator<int> output(cout, " ");
        sort(v1.begin(), v1.end());
         it1 = unique(v1.begin(), v1.end());
        copy(v1.begin(), it1, output);
        cout << endl;
   return 0;}
```

Algoritmi

STL algoritmi pokrivaju većinu uobičajeno korištenih operacija na nizovima elemenata

(traversal, searching, sorting, insertion/removal of elements)

Non modifying operations

- **for_each** Do specified operation for each element in a sequence
- **find** Find the first occurence of a specified value in a sequence
- **find_if** Find the first match of a predicate in a sequence
- **find_first_of** Find the first occurence of a value from one sequence in another
- **adjacent_find** Find the first occurence of an adjacent pair of values
- **count** Count occurences of a value in a sequence
- **count_if** Count matches of a predicate in a sequence
- **accumulate** Accumulate (i.e., obtain the sum of) the elements of a sequence
- **equal** Compare two ranges
- **max_element** Find the highest element in a sequence
- **min_element** Find the lowest element in a sequence

Modifying operations

transform	Apply an operation to each element in an input sequence and store the result in an output sequence (possibly the same input sequence)
copy	Copy a sequence
replace	Replace elements in a sequence with a specified value
replace_if	Replace elements matching a predicate
remove	Remove elements with a specified value
remove_if	Remove elements matching a predicate
reverse	Reverses a sequence
random_shuffle	Randomly reorganize elements using a uniform distribution
fill	Fill a sequence with a given value
generate	Fill a sequence with the result of a given operation

Sorting

- **sort** — Sort elements
- **stable_sort** — Sort maintaining the order of equal elements
- **nth_element** — Put nth element in its place
- **binary_search** — Find a value in a sequence, performing binary search