Vectors

Last updated 3/31/20

Vectors

- Motivation
  - Arrays have a limitation in that they must have a fixed size

  - The Standard Template Library has a data type called vector
    - A vector is a type of sequence container
    - Vectors hold a sequence of values or elements
    - Vectors are not limited to a fixed size
    - Vectors can use the array subscript operator [ ]
    - Vectors can have elements added to or removed at any time
    - Vectors can report their current size

    - Vectors are passed to functions by value by default
      - Unlike Arrays whitch are passed by reference by default

Vectors

- Syntax

```
#include <vector>
```

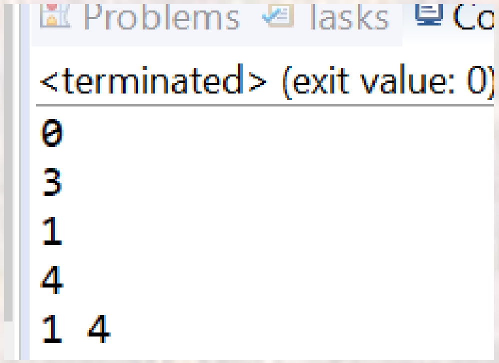
- creating vectors

```
vector<type> name;
vector<type> name(initial size);
vector<type> name(initial size, initial value);  // initializes all values
vector<type> name{list of element values};  // initializes values with {}

vector<type> name(vector);        // initializes from another vector
```

Vectors

- Accessing Vector elements
  - Just like arrays

```
#include <vector>

#include <iostream>
using namespace std;

int main(void){
	vector<int> v1(5);
	cout << v1[3] << endl;
	v1[2] = 3;
	cout << v1[2] << endl;;

	vector<int> v2(5,1);
	cout << v2[3] << endl;
	v2[0] = 4;
	cout << v2[0] << endl;

	vector<int> v3(v2);
	cout << v3[2] << " " << v3[0] << endl;
```



Problems Tasks Co

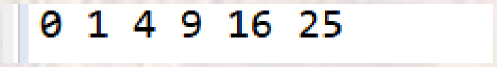
```
<terminated> (exit value: 0)
0
3
1
4
1 4
```

Vectors

- Accessing Vector elements
  - Size of the vector
    - objectName.size()

```
#include <vector>

#include <iostream>
using namespace std;

int main(void){

      vector<int> v4(6);
      for(unsigned int i=0; i < v4.size(); i++){
            v4[i] = i*i;
      }
      for(unsigned int i=0; i < v4.size(); i++){
            cout << v4[i] << " ";
      }
      cout << endl;

      return 0;
}
```

Note: unsigned int because size() returns an unsigned int and the compiler whines about a mismatch

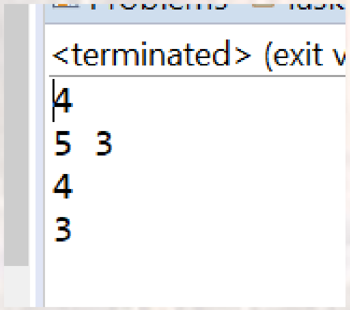


```
0 1 4 9 16 25
```

Vectors

- Add/remove elements from a vector
  - objectName.push_back() // adds one element to vector
  - objectName.push_back(value) // adds element with value
  - objectName.pop_back() // removes last element in vector

```
#include <vector>

#include <iostream>
using namespace std;

int main(void){
	vector<int> v5(4);
	cout << v5.size() << endl;
	v5.push_back(3);
	cout << v5.size() << " " << v5[4] << endl;
	v5.pop_back();
	cout << v5.size() << endl;
	v5.pop_back();
	cout << v5.size() << endl;

	return 0;
}
```

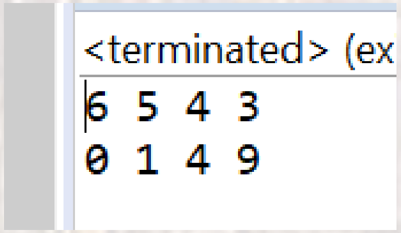


```
<terminated> (exit v
4
5 3
4
3
```

Vectors

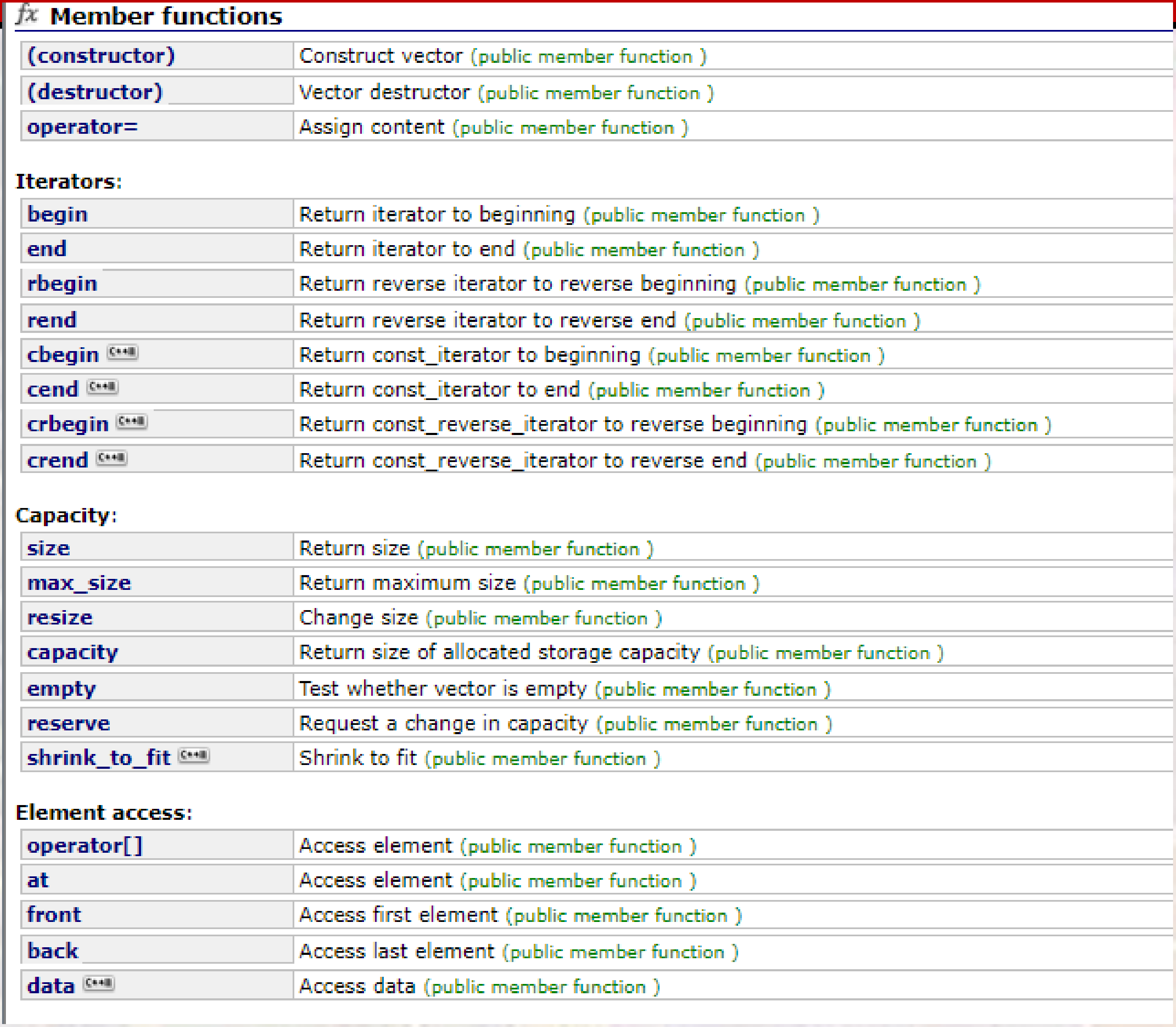
- Passing vectors to functions

```
#include <vector>

#include <iostream>
using namespace std;

void pv(const vector<int> & vec);

int main(void){
	vector<int> v1 {6,5,4,3};
	pv(v1);

	for(unsigned int i=0; i < v1.size(); i++)
		v1[i] = i*i;
	pv(v1);

	return 0;
}

void pv(const vector<int> & vec){
	for(unsigned int i = 0; i < vec.size(); i++){
		cout << vec[i] << " ";
	}
	cout << endl;
	return;
}
```



```
<terminated> (ex
6 5 4 3
0 1 4 9
```

Vectors



Member functions

(constructor)	Construct vector (public member function)
(destructor)	Vector destructor (public member function)
operator=	Assign content (public member function)

Iterators:

begin	Return iterator to beginning (public member function)
end	Return iterator to end (public member function)
rbegin	Return reverse iterator to reverse beginning (public member function)
rend	Return reverse iterator to reverse end (public member function)
cbegin C++11	Return const_iterator to beginning (public member function)
cend C++11	Return const_iterator to end (public member function)
crbegin C++11	Return const_reverse_iterator to reverse beginning (public member function)
crend C++11	Return const_reverse_iterator to reverse end (public member function)

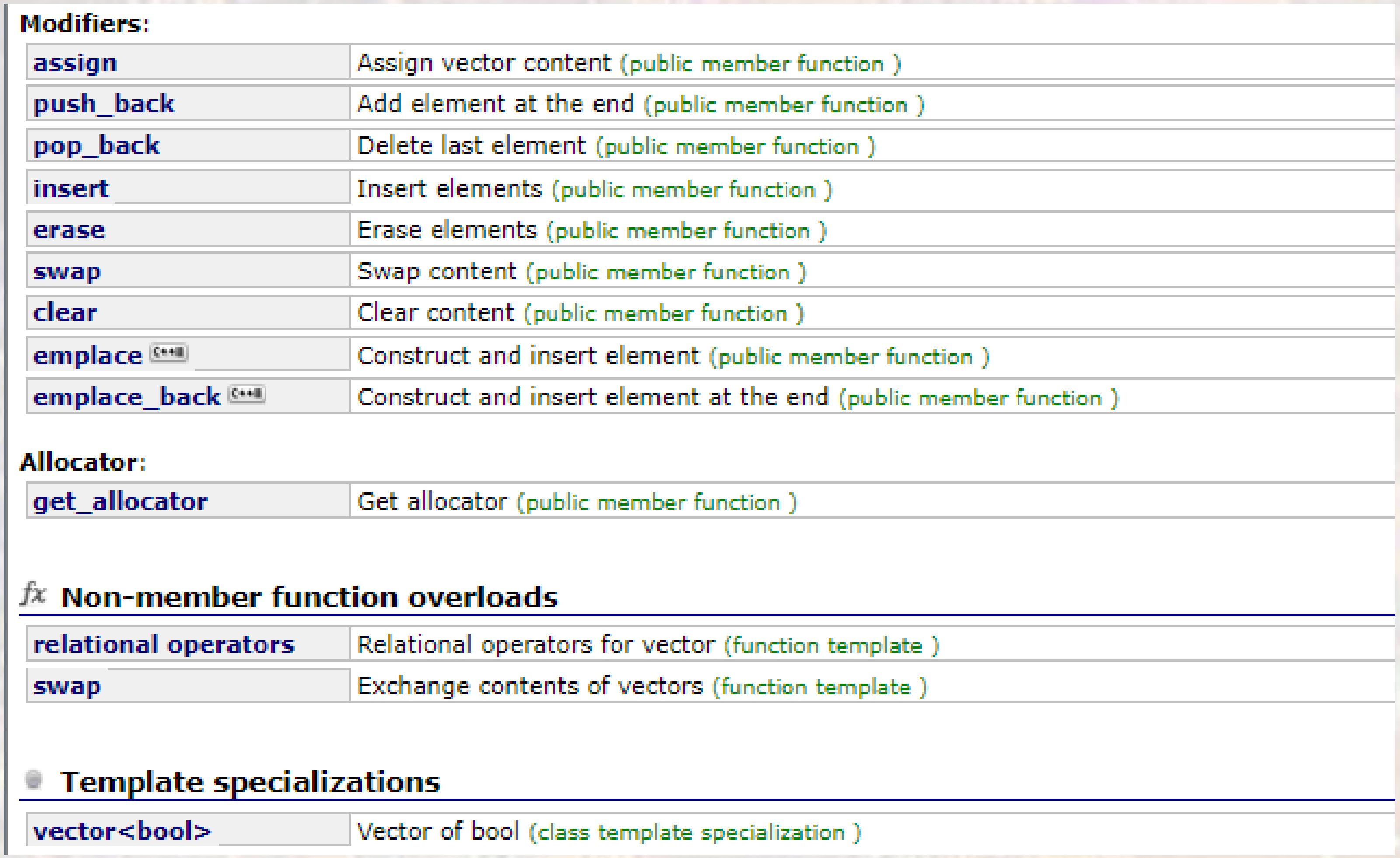
Capacity:

size	Return size (public member function)
max_size	Return maximum size (public member function)
resize	Change size (public member function)
capacity	Return size of allocated storage capacity (public member function)
empty	Test whether vector is empty (public member function)
reserve	Request a change in capacity (public member function)
shrink_to_fit C++11	Shrink to fit (public member function)

Element access:

operator[]	Access element (public member function)
at	Access element (public member function)
front	Access first element (public member function)
back	Access last element (public member function)
data C++11	Access data (public member function)

Vectors



Modifiers:

assign	Assign vector content (public member function)
push_back	Add element at the end (public member function)
pop_back	Delete last element (public member function)
insert	Insert elements (public member function)
erase	Erase elements (public member function)
swap	Swap content (public member function)
clear	Clear content (public member function)
emplace C++11	Construct and insert element (public member function)
emplace_back C++11	Construct and insert element at the end (public member function)

Allocator:

get_allocator	Get allocator (public member function)

fx Non-member function overloads

relational operators	Relational operators for vector (function template)
swap	Exchange contents of vectors (function template)

Template specializations

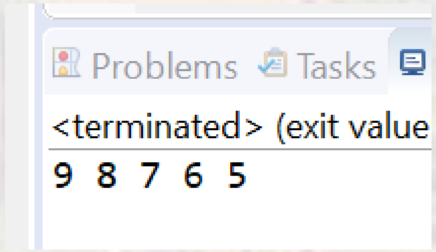
vector<bool>	Vector of bool (class template specialization)

Vectors

- Using Iterators
  - Iterators are a **type** intended to work with various container classes (like vector)
  - Operate very much like pointers

- Offer benefits when using container class objects

```
vector<int> vect1(10);
vector<int>::iterator itr;
for(itr=vect1.begin(); itr!=vect1.end(); itr++)
    cout << *itr << " ";
```

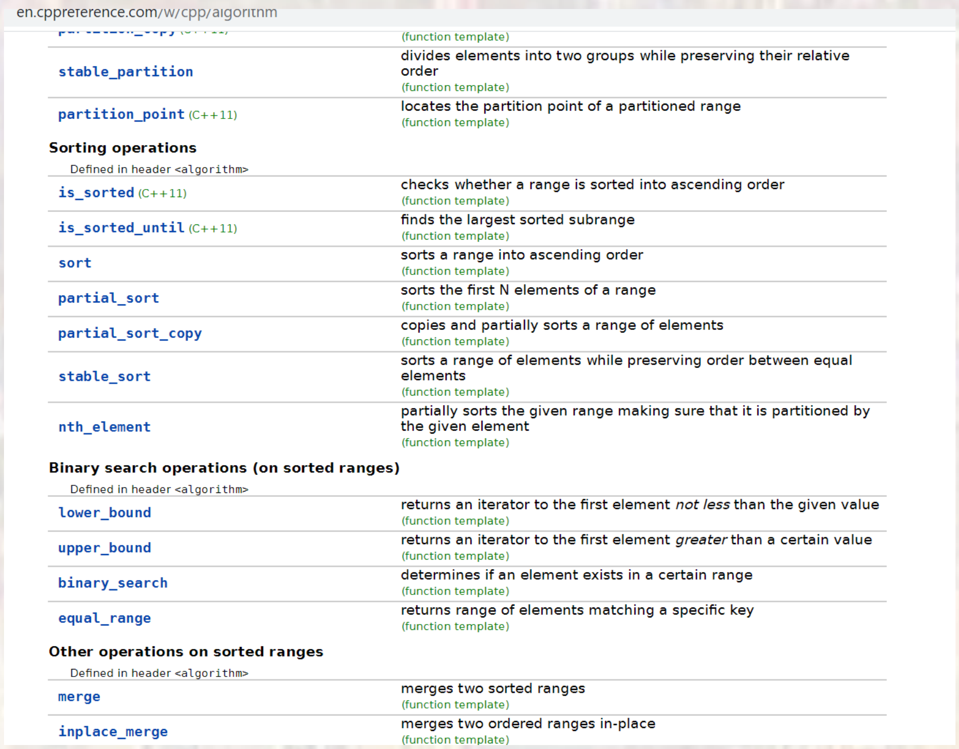


```
<terminated> (exit value
9 8 7 6 5
```

Vectors

- Using the Algorithm library
  - #include <algorithm>

…



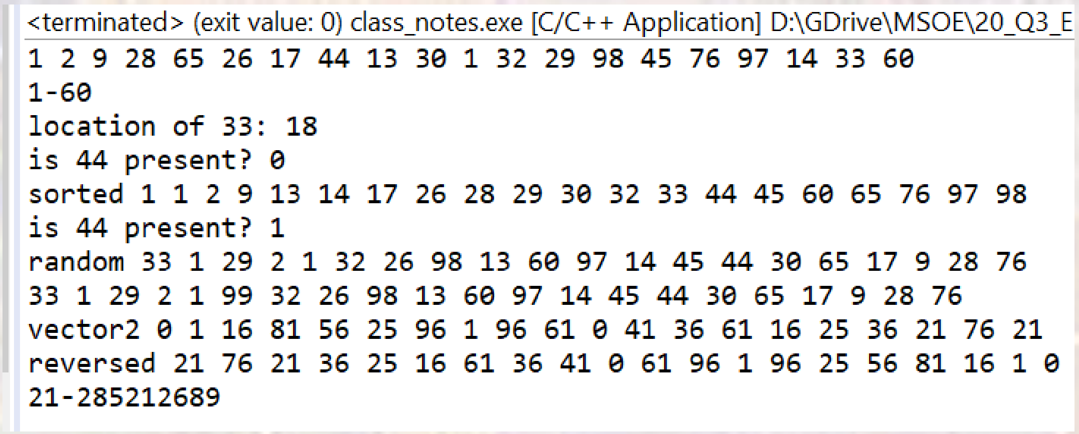
en.cppreference.com/w/cpp/algorithm

partition_copy (C++11)	(function template)
stable_partition	divides elements into two groups while preserving their relative order (function template)
partition_point (C++11)	locates the partition point of a partitioned range (function template)
Sorting operations	
Defined in header <algorithm>	
is_sorted (C++11)	checks whether a range is sorted into ascending order (function template)
is_sorted_until (C++11)	finds the largest sorted subrange (function template)
sort	sorts a range into ascending order (function template)
partial_sort	sorts the first N elements of a range (function template)
partial_sort_copy	copies and partially sorts a range of elements (function template)
stable_sort	sorts a range of elements while preserving order between equal elements (function template)
nth_element	partially sorts the given range making sure that it is partitioned by the given element (function template)
Binary search operations (on sorted ranges)	
Defined in header <algorithm>	
lower_bound	returns an iterator to the first element *not less* than the given value (function template)
upper_bound	returns an iterator to the first element *greater* than a certain value (function template)
binary_search	determines if an element exists in a certain range (function template)
equal_range	returns range of elements matching a specific key (function template)
Other operations on sorted ranges	
Defined in header <algorithm>	
merge	merges two sorted ranges (function template)
inplace_merge	merges two ordered ranges in-place (function template)

…

```
#include <vector>
#include <algorithm>

#include <iostream>
using namespace std;

void load_vector(vector<int> & myvector);
void load_vector2(vector<int> & myvector);
void print_vector(const vector<int> & myvector);

int main(void){
      vector<int> v1(20);
      vector<int> v2;

      load_vector(v1);
      print_vector(v1);

      cout << v1.front() << "-" << v1.back() << endl;
      cout << "location of 33: " << find(v1.begin(), v1.end(), 33) - v1.begin() <<
endl;

    cout << "is 44 present? " << binary_search(v1.begin(), v1.end(), 44) << endl;
    sort(v1.begin(), v1.end());
    cout << "sorted ";
    print_vector(v1);
    cout << "is 44 present? " << binary_search(v1.begin(), v1.end(), 44) << endl;

    random_shuffle(v1.begin(), v1.end());
    cout << "random ";
    print_vector(v1);

    v1.insert(v1.begin()+5, 99);
    print_vector(v1);
```

```
    load_vector2(v2);
      cout << "vector2 ";
      print_vector(v2);

      reverse(v2.begin(), v2.end());
      cout << "reversed ";
      print_vector(v2);

      cout << *v2.begin() << "-" << *v2.end() << endl;

      return 0;
}

void load_vector(vector<int> & myvector){
      for(unsigned int i=0; i< myvector.size(); i++){
            myvector[i] = (i*i*i+1)%100;
      }
      return;
}
void load_vector2(vector<int> & myvector){
      for(unsigned int i=0; i < 20; i++){
            myvector.push_back((i*i*i*i)%100);
      }
      return;
}
void print_vector(const vector<int> & myvector){
    vector<int>::const_iterator itr;
    for(itr = myvector.begin(); itr != myvector.end(); itr++){
        cout << *itr << " ";
    }
    cout << endl;
    return;
}
```



```
<terminated> (exit value: 0) class_notes.exe [C/C++ Application] D:\GDrive\MSOE\20_Q3_E
1 2 9 28 65 26 17 44 13 30 1 32 29 98 45 76 97 14 33 60
1-60
location of 33: 18
is 44 present? 0
sorted 1 1 2 9 13 14 17 26 28 29 30 32 33 44 45 60 65 76 97 98
is 44 present? 1
random 33 1 29 2 1 32 26 98 13 60 97 14 45 44 30 65 17 9 28 76
33 1 29 2 1 99 32 26 98 13 60 97 14 45 44 30 65 17 9 28 76
vector2 0 1 16 81 56 25 96 1 96 61 0 41 36 61 16 25 36 21 76 21
reversed 21 76 21 36 25 16 61 36 41 0 61 96 1 96 25 56 81 16 1 0
21-285212689
```