

VECTORS

Vector is a sequence of objects with direct access. It supports constant time for adding / deleting elements from the end of sequence and linear time for adding / deleting elements from middle or beginning. Size of vector varies dynamically, memory management is automatic. For using **vector** you need to include the following library:

```
#include <vector>
```

Constructors

Create an empty vector v (array of length 0, with no element):

```
vector<int> v;  
v = ()
```

Create a vector of length 5:

```
vector<int> v(5);  
// v = (0, 0, 0, 0, 0)
```

Create a vector of length 5 with all elements equal to 2:

```
vector<int> v(5,2);  
// v = (2, 2, 2, 2, 2)
```

Let m be an array of integers. To create a vector containing the same numbers as in m you need a **copy constructor**:

```
int m[] = {1,2,3,4,5};  
vector<int> v(m,m+5);  
// v = (1, 2, 3, 4, 5)
```

Example. Create vector. Print the vector elements in a line.

```
#include <cstdio>  
#include <vector>  
using namespace std;  
  
int m[] = {1,2,3,4,5};  
vector<int> v(m,m+5);  
  
int main(void)  
{  
    for(int i = 0; i < v.size(); i++)  
        printf("%d ",v[i]);  
    printf("\n");  
    return 0;  
}
```

After declaration

```
vector<int> v;
```

vector v is empty, it contains no element: $v = ()$. Method **resize()** changes the number of vector elements and fills them with 0. For example, after

```
v.resize(5);
```

vector will look like $v = (0, 0, 0, 0, 0)$.

Method **size()** returns the number of elements in the vector.

```
#include <cstdio>
```

```

#include <vector>
using namespace std;

vector<int> v;

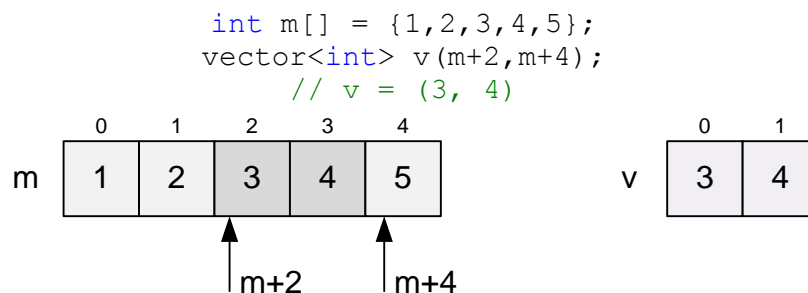
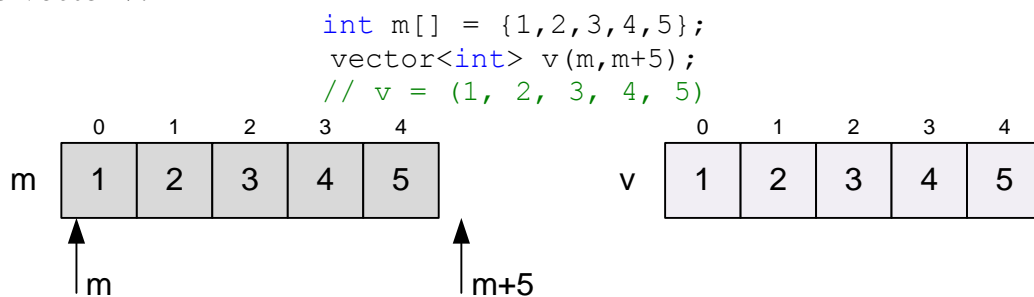
int main(void)
{
    v.resize(5);
    for(int i = 0; i < v.size(); i++)
        printf("%d ", v[i]);
    printf("\n");
    return 0;
}

```

Following methods are created to work with vectors:

method	method description
void clear()	deleting all elements vector becomes empty
size_type size() const	calculates the size of vector
bool empty() const	returns true if vector is empty
reference operator[](size_type n)	indexation operator, returns n-th element of vector
reference front()	returns first element of vector
reference back()	returns last element of vector

Example. Let m be an array of integers. Let's create vector v by copying all data from m to vector v :

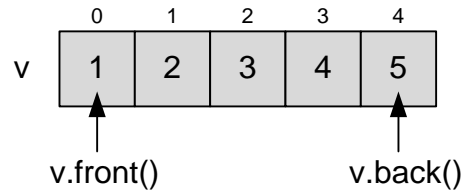


Let's find the size of vector v :

```
printf("Size: %d\n", v.size());
```

Let's find the first and the last elements:

```
printf("First: %d, Last: %d\n",v.front(),v.back());
```



Example. Create vector and print its elements.

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

int m[] = {1,2,3,4,5};
vector<int> v(m+1,m+4); // v = (2, 3, 4)

int main(void)
{
    printf("Size of array = %d\n",v.size());
    printf("First = %d, last = %d\n",v.front(),v.back());

    for(int i = 0; i < v.size(); i++)
        printf("%d ",v[i]);
    printf("\n");
    return 0;
}
```

method	method description
void push_back(const T& x)	add element to the end of vector
void pop_back()	delete last element

Example. Push and pop operations with vector.

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

vector<int> v;

int main(void)
{
    v.push_back(3); v.push_back(6); // v = (3, 6)
    v.push_back(8); v.push_back(1); // v = (3, 6, 8, 1)

    for(int i = 0; i < v.size(); i++)
        printf("%d ",v[i]);
    printf("\n");

    v.pop_back(); // v = (3, 6, 8)

    for(int i = 0; i < v.size(); i++)
        printf("%d ",v[i]);
    printf("\n");
}
```

```
    return 0;
}
```

E-OLYMP 928. The sum of the largest and the smallest Find the sum of the largest and the smallest element in array.

► Use *vector* to solve the problem. Use two ways:

- **resize** the vector according to the number n of its elements (if this value is known);
- use **push_back** to insert the values to the vector;

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

int i, n, mn, mx, res;
vector<int> v;

int main(void)
{
    scanf("%d", &n);
    v.resize(n); // v = (0, 0, . . . , 0)

    for (i = 0; i < n; i++)
        scanf("%d", &v[i]);

    mn = 1e9; mx = -1e9; // 1e9 = 1000000000
    for (i = 0; i < v.size(); i++)
    {
        if (v[i] < mn) mn = v[i];
        if (v[i] > mx) mx = v[i];
    }

    res = mn + mx;
    printf("%d\n", res);
    return 0;
}
```

Use **push_back** to insert the values to the vector:

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

int x, i, n, mn, mx, res;
vector<int> v;

int main(void)
{
    scanf("%d", &n);
    for (i = 0; i < n; i++)
    {
        scanf("%d", &x);
        v.push_back(x);
    }

    mn = 1e9; mx = -1e9;
```

```

for (i = 0; i < v.size(); i++)
{
    if (v[i] < mn) mn = v[i];
    if (v[i] > mx) mx = v[i];
}

res = mn + mx;
printf("%d\n", res);
return 0;
}

```

E-OLYMP 8685. Arithmetic mean Find the arithmetic mean of the given sequence of integers.

► Read the given sequence into *vector*. As we do not know the initial size of the sequence, read it till the end of file and use **push_back**.

```

#include <cstdio>
#include <vector>
using namespace std;

int i, x, s;
vector<int> v;

int main(void)
{
    while (scanf("%d", &x) == 1)
        v.push_back(x);

    s = 0;
    for (i = 0; i < v.size(); i++)
        s = s + v[i];

    printf("%.4lf\n", 1.0 * s / v.size());
    return 0;
}

```