Osnove inženjerske informatike II.

Uvod u programiranje

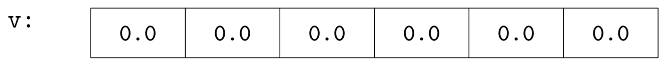
Ponešto iz standardne biblioteke

K. F. & V. B.

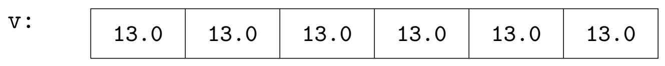
Predložak *vector*<>

- programski prikaz nizova
- #include<vector>
- definicije varijabli tipa *vector*<>:

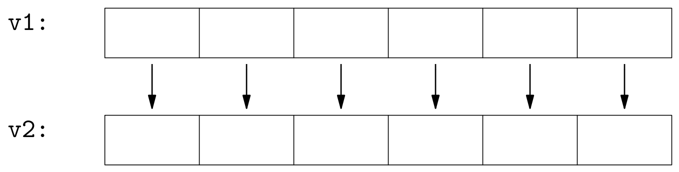
```
vector<double> v;             // niz bez elemenata
vector<double> v (n);         // npr. n = 6
```



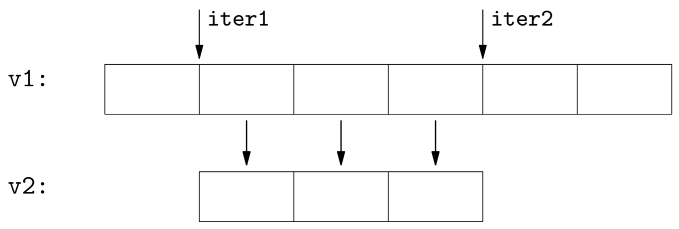
```
vector<double> v (n, c);      // npr. n = 6 i c = 13,0
```



```
vector<double> v2 (v1);
```



```
vector<double> v2 (iter1, iter2);
```

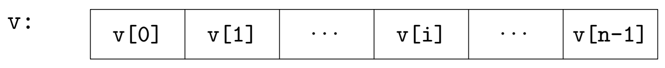


- broj elemenata niza:

  `v.size();`

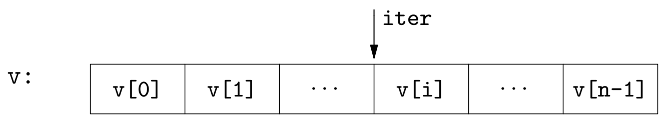
- pristup elementima niza:

  `v[i]`



  `v.at(i)`

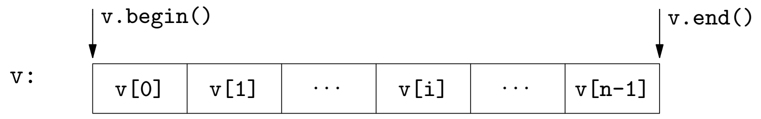
  `*iter`



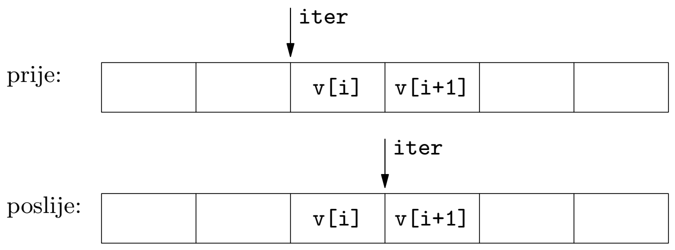
- iteratori:

`v.begin()`

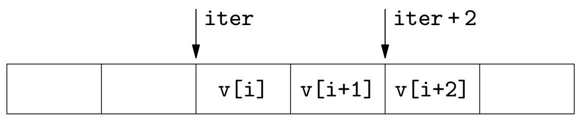
`v.end()`



`++iter`



`iter + j`

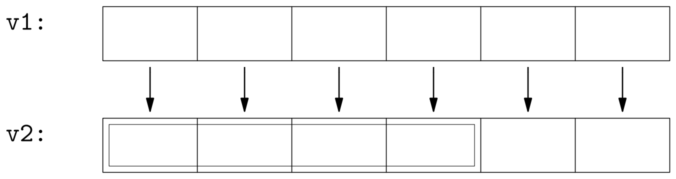


`--iter`

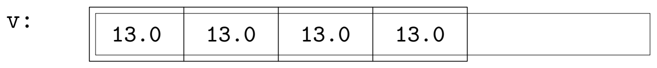
`iter - j`

- pridruživanja i promjene veličine:

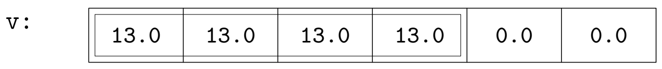
`v2 = v1;`



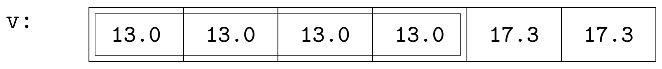
`v.assign(n, c);` // npr. $n = 4$ i $c = 13{,}0$



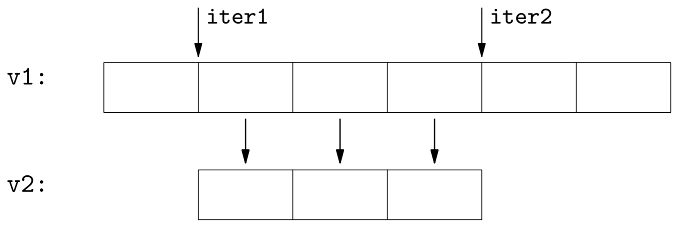
`v.resize(n);` // npr. $n = 6$



`v.resize(n, c);` // npr. $n = 6$ i $c = 17{,}3$

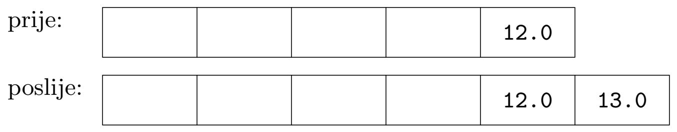


`v2.`*`assign`*` (iter1, iter2);`

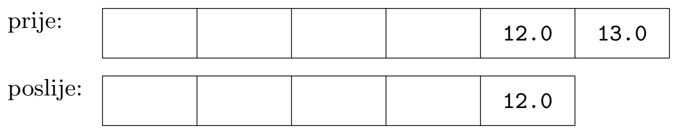


- dodavanje i uklanjanje elemenata:

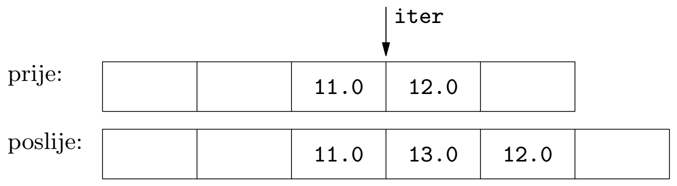
`v.`*`push_back`*` (c);` // c = 13,0



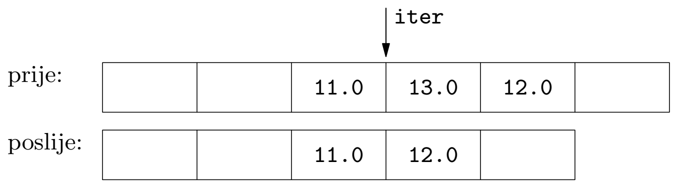
`v.`*`pop_back`*` ();`



`v.`*`insert`*` (iter, c);` // c = 13,0



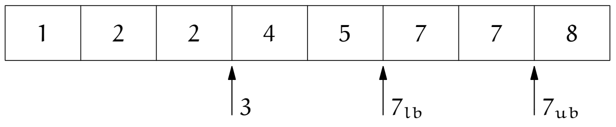
`v.`*`erase`*` (iter);`



Neki standardni algoritmi

- #include<algorithm>

  - ***bool*** *equal* (Iter1 bit1, Iter1 eit1, Iter2 bit2);
  - Iter *min_element* (Iter bit, Iter eit);
  - Iter *max_element* (Iter bit, Iter eit);
  - Iter *find* (Iter bit, Iter eit, T val);
  - Iter *sort* (Iter bit, Iter eit);
  - Iter *lower_bound* (Iter bit, Iter eit, T val);
  - Iter *upper_bound* (Iter bit, Iter eit, T val);



  -
```
OIter transform (InIter bin, InIter ein,
                 OIter bout, UnaryFunc f);
```

    $\boldsymbol{v}_2 = -\boldsymbol{v}_1$

    v2[i] = -v1[i] $\quad \forall$ i $\in [0, n-1]$

```
transform (v1.begin(), v1.end(),
           v2.begin(), negate<double>());
```

  -
```
OIter transform (InIter1 bin1, InIter1 ein1,
                 InIter2 bin2,
                 OIter bout, BinaryFunc f);
```

    $\boldsymbol{v}_3 = \boldsymbol{v}_1 - \boldsymbol{v}_2$

    v3[i] = v1[i] - v2[i] $\quad \forall$ i $\in [0, n-1]$

```
transform (v1.begin(), v1.end(),
           v2.begin(),
           v3.begin(), minus<double>());
```

- #include<numeric>

  - T *accumulate* (Iter bit, Iter eit, T init_val);

    $$v = v_{\text{init}} + \sum_{i=0}^{n} v_i$$

- `T` *`accumulate`* `(Iter bit, Iter eit, T init_val, BinaryFunc op);`

  $v = v_{\text{init}}\ \mathtt{op}\ v_0\ \mathtt{op}\ v_1\ \mathtt{op}\ \ldots\ \mathtt{op}\ v_{n-1}$

- `T` *`inner_product`* `(Iter1 bit1, Iter1 eit1,`
  `Iter2 bit2, T init_val);`

  $v = v_{\text{init}} + \sum_{i=0}^{n} x_i \cdot y_i$

- `T` *`inner_product`* `(Iter1 bit1, Iter1 eit1, Iter2 bit2,`
  `T init_val, BinaryFunc op1, BinaryFunc op2);`

  $v = v_{\text{init}}\ \mathtt{op}_1\ (x_0\ \mathtt{op}_2\ y_0)\ \mathtt{op}_1\ (x_1\ \mathtt{op}_2\ y_1)\ \mathtt{op}_1\ \ldots\ \mathtt{op}_1\ (x_{n-1}\ \mathtt{op}_2\ y_{n-1})$

Neki funkcijski objekti

- za primjenu u algoritmima *`transform`*`()`, *`accumulate`*`()`, *`inner_product`*`()`, ...
- #include<functional>
- unarni:

`negate``<T>()`	$-x_i$

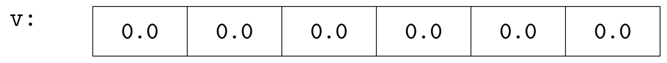
- binarni:

`plus``<T>()`	$x_i + y_i$
`minus``<T>()`	$x_i - y_i$
`multiplies``<T>()`	$x_i \cdot y_i$
`divides``<T>()`	x_i / y_i

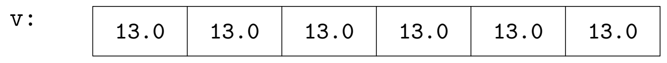
Predložak valarray<>

- programski prikaz linearnoalgebarskih vektora
- #include <valarray>
- definicije varijabli tipa valarray<>:

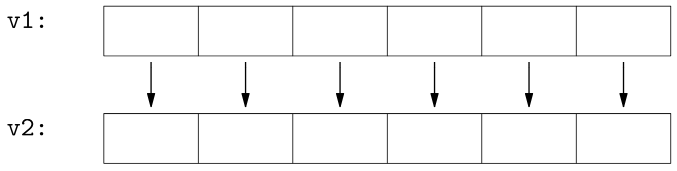
```
valarray<double> v;            // vektor bez komponenata
valarray<double> v (n);        // npr. n = 6
```



```
valarray<double> v (s, n);     // npr. s = 13,0 i n = 6
```



```
valarray<double> v2 (v1);
```



- broj elemenata:

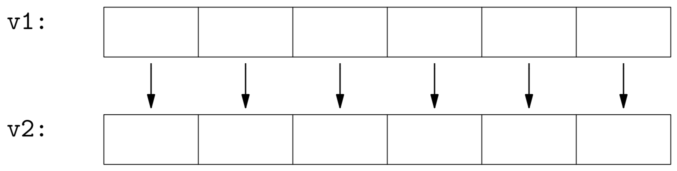
```
v.size();
```

- pristup elementima:

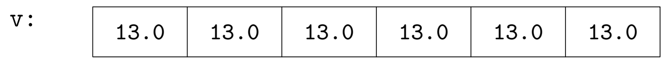
```
v[i]
```

- pridruživanja i promjene veličine:

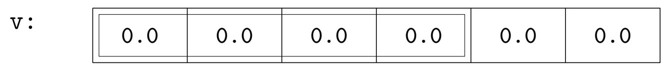
```
v2 = v1;     // v1 i v2 moraju imati isti broj elemenata!
```



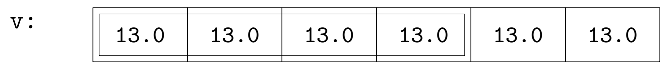
```
v = s;       // s skalar, npr. s = 13,0
```



`v.resize(n);` // npr. $n = 6$



`v.resize(n, s);` // npr. $n = 6$ i $c = 13,0$



- aritmetičke operacije:

```
v2 = -v1;           // v1 i v2 moraju imati isti broj elemenata!

v3 = v1 + v2;       // v1 i v2 moraju imati isti broj elemenata!
v2 += v1;           // v1 i v2 moraju imati isti broj elemenata!
v2 = v1 + s;        // s skalar
v2 = s + v1;
v2 += s;

v3 = v1 - v2;
v2 -= v1;
v2 = v1 - s;
v2 = s - v1;
v2 -= s;

v3 = v1 * v2;       // v3[i] = v1[i] * v2[i]
v2 *= v1;
v2 = v1 * s;
v2 = s * v1;
v2 *= s;

v3 = v1 / v2;       // v3[i] = v1[i] / v2[i]
v2 /= v1;
v2 = v1 / s;
v2 = s / v1;        // v2[i] = s / v2[i]
v2 /= s;
```

`s = v1.sum();` // $\sum_i v_i$

`s = (v1 * v2).sum();` // skalarni produkt

- relacijske operacije:

```
v_bool = (v1 == v2);     // v_bool[i] = (v1[i] == v2[i])
v_bool = (v1 != v2);
v_bool = (v1 < v2);
v_bool = (v1 <= v2);
v_bool = (v1 > v2);
v_bool = (v1 >= v2);

b = (v1 == v2).min();    // v1 ≡ v2
```

- funkcije:

```
s = v1.min();
s = v1.max();

v2 = abs(v1);
v2 = sqrt(v1);
v2 = pow(v1, s);
v2 = exp(v1);
v2 = log(v1);
v2 = sin(v1);
v2 = cos(v1);
v2 = tan(v1);
v2 = atan(v1);
```