



Napredne tehnike programiranja

[Studij informatike](#)

Napredne tehnike programiranja

Vježba 2



Teme

- Rad sa listom u C++-u

Lista u STL-u



- Standard Template Library (STL) sadrži gotove predloške za osnovne strukture podataka
- Između ostalog sadrži predloške za dvije vrste listi:
 - Jednostruka vezana lista (`forward_list`)
 - Dvostruko vezana lista (`list`)

Korištenje STL predloška



Kako uključiti STL predložak u naš program?

Za dvostruko vezanu listu:

```
#include<list>
```

Za jednostruko vezanu listu:

```
#include<forward_list>
```

Odabir liste



- Kad koristimo koju listu?
 - Uvijek koristimo dvostruko vezanu listu, osim ako imamo jak razlog za korištenje jednostruko vezane liste
- Koji su razlozi za korištenje jednostruko vezane liste?
 - Lista mora biti memorijski optimizirana
 - Ne moramo se kretati unazad po listi
 - Lista je prazna ili ima malo podataka

Deklaracija liste



- Listu prije korištenja moramo deklarirati:

```
list <int> moja_lista1;
```

```
list <float> moja_lista2;
```

```
list <string> moja_lista3;
```



Tip podataka u listi

Pristup elementima liste

```
list<int> L; // L=()
L.push_back(5); // L=(5)
L.push_front(7); // L=(7, 5)
L.push_back(3); // L=(7, 5, 3)
int a = L.front(); // a = 7
int b = L.back(); // b = 3
L.pop_front(); // L=(5, 3)
L.pop_back(); // L=(5)
```

Iteratori



- Kako doći do unutarnjih elemenata liste?
 - Pomoću iteratora (to su 'pokazivači' na elemente liste)

- Kreiranje iteratora:

```
list <int>::iterator pozicija;
```

- Postavljanje početne vrijednosti iteratora:

```
pozicija=moja_lista.begin();
```

```
pozicija=moja_lista.end()
```


Iteratori



- Kako doći do unutarnjih elemenata liste?
 - Pomoću iteratora (to su ‘pokazivači’ na elemente liste)
- Pomicanje iteratora udesno na slijedeći element:

```
pozicija++;
```

- Pomicanje iteratora ulijevo:

```
pozicija--;
```

Iteratori



- `v.begin()` – vraća iterator koji pokazuje na početni element liste
- `v.end()` – vraća iterator koji pokazuje na zadnji element liste
- `(v.begin() == v.end())` ako je lista prazna
- `++` (inkrement) iteratora pozicionira ga na slijedeći element liste
- `--` (inkrement) iteratora pozicionira ga na prethodni element liste

Obilazak cijele liste



```
#include <iostream>
#include <list>
using namespace std;

int main() {
    list<int> l{1,2,3};
    list<int>::iterator i;
    for (i=l.begin(); i!=l.end(); ++i) {
        cout<<*i<<endl;
    }
}
```

Brisanje liste



- `c.erase(p)` briše element na kojeg pokazuje iterator `p`
 - vraća iterator na element iza obrisanog
- `c.erase(b, e)` briše sve elemente između dva iteratora
- `c.clear()` briše sve elemente liste
- `c.pop_back()` briše zadnji element liste
- `c.pop_front()` briše prvi element liste
- Brisanje elemeneta može uništiti iteratore

Brisanje liste



- brisanje svih elemenata liste jednakih 5 :

```
list l {1,2,3,4,5,6,7,8};
list::iterator i, ltemp;
i = l.begin();
while (i != l.end())
    if (*i == 5) {
        l.erase(i); i++; //opasno!
    }
    else i++;
}
```

- Pokušali smo povečati iterator koji smo upravo “uništiti”

Brisanje liste - ispravno



- brisanje svih elemenata liste jednakih 5 :

```
list l {1,2,3,4,5,6,7,8};  
list::iterator i, ltemp;  
i = L.begin();  
while (li != L.end())  
    if (*i == 5) {  
        i = l.erase(i);  
    }  
    else i++;  
}
```

- Iskoristili smo činjenicu da erase vraća iterator na element iza obrisanog

push_back i push_front



```
#include <iostream>
#include <list>
using namespace std;

int main(){
    list<int> l={1,2,3,4,5};    //c++ 11

    l.push_back(6);
    l.push_back(7);
    /* sad je lista: 1,2,3,4,5,6,7 */

    l.push_front(8);
    l.push_front(9);
    /* sad je lista: 9,8,1,2,3,4,5,6,7 */
}
```

pop_back i pop_front



```
#include <iostream>
#include <list>
using namespace std;

int main(){
    list<int> l{1,2,3,4,5};

    l.pop_back();
    /* sad je lista 1,2,3,4 */


    l.pop_front();
    /* sad je lista 2,3,4 */
}
```


insert



- `insert(iterator, element)` : umeće `element` u listu ispred pozicije na koju pokazuje `iterator`.
- `insert(iterator, number, element)` : umeće `element` u listu ispred pozicije na koju pokazuje `iterator` `number` puta.
- `insert(iterator, start_iterator, end_iterator)` : umeće elemente od pozicije `start_iterator` do pozicije `end_iterator` prije pozicije na koju pokazuje `iterator`

insert



```
#include <iostream>
#include <list>
using namespace std;
int main(){
    list<int> l = {1,2,3,4,5};
    list<int>::iterator it = l.begin();
    l.insert (it+1, 100);    // umeće 100 prije pozicije 2
    /* sad je lista 1 100 2 3 4 5 */

    l.insert(l.begin(), 5, 10); // umeće 10 prije begin 5 puta
    /* sad je 10 10 10 10 10 10 1 100 2 3 4 5 */

    list<int> new_l = {10,20,30,40}; // nova lista
    new_l.insert (new_l.begin(), l.begin(), l.end());
    /* umeće elemente od početka do kraja liste l na početak liste new_l
       sad je lista new_l 10 10 10 10 10 1 100 2 3 4 5 10 20 30 40*/
}
```

reverse



Reverse obrće redoslijed elemenata u listi

```
#include <iostream>
#include <list>
using namespace std;

int main()
{
    list<int> l{1,2,3,4,5};

    l.reverse();
    /* sad je lista 5,4,3,2,1 */
}
```

sort

sort() – sortira listu

sort(compare_function) - sortira listu prema kriteriju u funkciji compare_function

```
#include <iostream>
```

```
#include <list>
```

```
using namespace std;
```

```
bool compare_function( string& s1 , string& s2 ){  
    return ( s1.length() < s2.length() );  
}
```

```
int main(){
```

```
    list<int> list1 = {2,4,5,6,1,3};
```

```
    list<string> list2 = {"h", "hhh", "hh"};
```

```
    list1.sort();
```

```
    /* list1 je sad 1 2 3 4 5 6 */
```

```
    list2.sort(compare_function);
```

```
    /* list2 je sad h hh hhh */
```

```
}
```

Još neke funkcije



- empty – prazni listu

```
l.empty();
```

- size – vraća broj elemenata u listi

```
cout<<l.size();
```

- swap – zamjenjuje sadržaj dvije liste

```
swap(l1,l2);
```

Zadaci



1. Napišite program koji učitava i stavlja na kraj liste stringove sve dok se ne učitava string "kraj".
2. Napišite program koji učitava i stavlja na kraj liste parne brojeve, a na početak liste neparne brojeve. Unos se završava kad se upiše 0.
3. Napišite funkciju: `void pomnozi (list <double> L, double multiplikator)` kojom se svi elementi liste L množe s vrijednošću parametra multiplikator.
4. Lista sadrži niz znakova. Napišite funkciju `bool IsSorted(const list<char>& L)` koja vraća `true` ako su elementi liste sortirani uzlazno, inače vraća `false`.
5. Napišite program koji ispred svakog stringa S u listi dodajte još onoliko čvorova koliko S ima slova, u svaki od čvorova upišite po jedno slovo od S - npr. ako je učitana lista bila ("NTP", "Jupi"), onda rezultatna lista treba biti ("N", "T", "P", "NTP", "J", "u", "p", "i", "Jupi")

Bodovanje rješenja



| Broj rješanih zadataka | Broj bodova |
|------------------------|-------------|
| 0 | 0 |
| 1 | 0 |
| 2 | 0 |
| 3 | 1 |
| 4 | 2 |
| 5 | 3 |