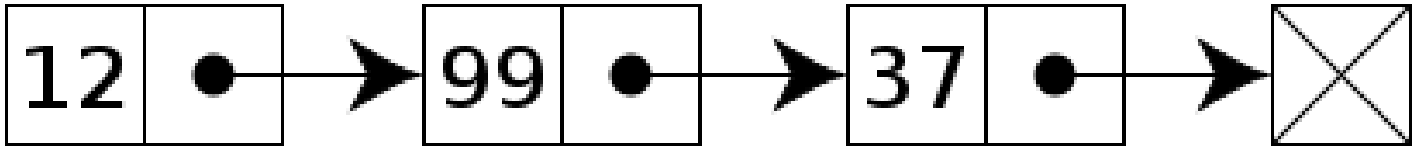
Programowanie C++

Wykład 5 (30.03.2016)

Lista łączona (*linked list)*




```
struct ElementListy {
     ElementListy * nast;
     int wart;
};

typedef ElementListy *Lista;
```

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

struct ElementListy {
        ElementListy * nast;
        int wart;
};

typedef ElementListy *Lista;


void dodaj_element(Lista &lista, int wartosc) {
 ElementListy * nowy = new ElementListy();
 nowy->wart = wartosc;
 nowy->nast = lista;
 lista = nowy;
}


void wypisz_liste(Lista lista) {
 ElementListy * akt = lista;
 while(akt != nullptr) {
  cout << akt->wart << "\n";
  akt = akt->nast;
 }
}
```

```
int main() {
  Lista lista = nullptr;

  dodaj_element(lista, 10);
  dodaj_element(lista, 25);
  dodaj_element(lista, 1);
  dodaj_element(lista, 5);

  wypisz_liste(lista);
  return 0;
}
```

```
class Lista {
  public:
    Lista();    // konstruktor
    ~Lista();   // destruktor

    void dodaj_element(int wartosc);
    void wypisz_liste();

  private:
    ElementListy * pierwszy;
};
```

```
class Lista {
  public:
    Lista();    // konstruktor
    ~Lista();   // destruktor

    void dodaj_element(int wartosc) {
      ElementListy * nowy = new ElementListy();
      nowy->wart = wartosc;
      nowy->nast = pierwszy;
      pierwszy = nowy;
    }


    void wypisz_liste();

  private:
    ElementListy * pierwszy;
};
```

```
class Lista {
  public:
    Lista();    // konstruktor
    ~Lista();   // destruktor

    void dodaj_element(int wartosc);
    void wypisz_liste();

  private:
    ElementListy * pierwszy;
};
```

```
void Lista::wypisz_liste() {
 ElementListy * akt = lista;
 while(akt != nullptr) {
  cout << akt->wart << "\n";
  akt = akt->nast;
 }
```

```
class Lista {
  public:
    Lista();    // konstruktor
    ~Lista();   // destruktor

    void dodaj_element(int wartosc);
    void wypisz_liste();

  private:
    ElementListy * pierwszy;
};
```

```
Lista::Lista() {
  pierwszy = nullptr;
}

Lista::~Lista() {
    while(pierwszy != nullptr) {
      ElementListy *  pomocniczy = pierwszy->nast;
      delete pierwszy;
      pierwszy = pomocniczy;
   }
```

Przykład klasy: liczby zespolone

Kontenery STL

```
#include <forward_list>
#include <iostream>
using namespace std;
int main() {

  forward_list<int> lista;
  lista.push_front(5);
  lista.push_front(3);
  lista.push_front(1);

  cout << lista.pop_front() << "\n";

  if(lista.empty())
        cout << "Lista jest pusta\n";

  return 0;
}
```

Kontenery STL

```
#include <list>
#include <iostream>
using namespace std;
int main() {

  list<int> lista;
  lista.push_front(5);
  lista.push_front(3);
  lista.push_front(1);
  lista.push_back(8);

  cout << lista.pop_front() << "\n";

  if(lista.empty())
        cout << "Lista jest pusta\n";

  return 0;
}
```

Container	Insert Head	Insert Tail	Insert	Remove Head	Remove Tail	Remove	Index Search	Find
vector	n/a	O(1)	O(n)	O(1)	O(1)	O(n)	O(1)	O(log n)
list	O(1)	O(1)	O(1)	O(1)	O(1)	O(1)	n/a	O(n)
deque	O(1)	O(1)	n/a	O(1)	O(1)	O(n)	n/a	n/a
queue	n/a	O(1)	n/a	O(1)	n/a	n/a	O(1)	O(log n)
stack	O(1)	n/a	n/a	O(1)	n/a	n/a	n/a	n/a
map	n/a	n/a	O(log n)	n/a	n/a	O(log n)	O(1)	O(log n)
multimap	n/a	n/a	O(log n)	n/a	n/a	O(log n)	O(1)*	O(log n)
set	n/a	n/a	O(log n)	n/a	n/a	O(log n)	O(1)	O(log n)
multiset	n/a	n/a	O(log n)	n/a	n/a	O(log n)	O(1)*	O(log n)

http://devmentor.org/references/stl/stl.php