

Data Structures

Lab 03 (8%)

Topics: Doubly Linked Lists

Problem 01 (3%)

(low-level mechanisms of linked lists in C++)

You have to do following operations in your C++ program using doubly linked list

- read arbitrary amount of integer numbers in linked list;
- print list in direct and reversed order;
- insert before each even element in the list value 0;
- print list in direct and reversed order;
- remove all even elements from list;
- print list in direct and reversed order;

You have to use structure:

```
struct Node
{
    int data;
    Node* next;
    Node* prev;

    Node(int aData, Node* aNext, Node* aPrev)
        : data(aData), next(aNext), prev(aPrev)
    {
    }
};
```

and implement following functions in your program:

```
void pushBack(Node*& head, Node*& tail, int elem);
void printInDirectOrder(Node *head, Node *tail);
void printInReversedOrder(Node *head, Node *tail);
void insertBefore(Node *&head, Node *cur, int elem);
Node* erase(Node *&head, Node *&tail, Node *curr);
```

Problem 02 (1%)

(standard class list)

Solve previous problem using standard `std::list`;

Problem 03 (3%)

(class `ListInt`)

You have to develop class `ListInt` which mimics several fundamental features of standard list and solve Problem 02 again using this class instead of `std::list`;

Your class has to have:

- constructor `ListInt()`: creates empty ready to use list;
- copy constructor;
- assignment operator;
- destructor;
- method `int size()`: returns the current size of the list;

- class ListInt::Iter (like iterator of std::list) with operations: ++, --, *, ==, !=
- class ListInt::RIter (like reverse_iterator of std::list) with operations: ++, --, *, ==, !=
- method Iter begin();
- method Iter end();
- method RIter rbegin();
- method RIter rend();
- method void pushBack(int e);
- method void popBack();
- method void pushFront(int e);
- method void popFront();
- method Iter insert(Iter pos, int elem);
- method Iter erase(Iter pos);

Problem 04 (1%)

(performance of lists and vectors)

You have to compare performance of std::list and std::vector for this simple problem:

create empty container

do n times:

 generate random integer number

 insert this number in container before first element in container greater or equal to this number

Solve this problem using std::vector and std::list for n = 50000, 100000, ..., 500000 and output the time of each case. Explain results. Watch video Going Native 2012 Stroustrup Keynote (from 44:40)

[link](#).