List

Сложност: 10/10
Позволени езици: C++

1. Общо описание

Целта на проекта е да бъде реализиран **двусвързан** списък.

2. Описание на задачата

Трябва да бъде реализиран шаблонен клас List в **header** файл.

1. Вътре в класа List да бъде написана **структура за елемент в свързания списък (Node)**.
В структурата да има конструктор и член-променливи: m_data(int), m_next(Node*), m_prev(Node*)
За класа List да бъдат имплементирани:

2. Default-ен конструктор

3. Деструктор

4. void push_front(const T& value) - добавя елемент в началото на списъка

5. void pop_front() - премахва елемент от началото на списъка

6. void push_back(const T& value) - добавя елемент в края на списъка

7. void pop_back() - премахва елемент от края на списъка

8. T& front() - връща стойността на елемента в началото на списъка

9. T& back() - връща стойността на елемента в края на списъка

10. Вътре в класа List да бъде имплементиран **клас iterator**. Като член-променлива този клас има указател към Node. Да бъде написан конструктор за този клас и да бъдат предефинирани следните оператори:

- **T& operator*()** - връща стойността на даден Node (data)
- **iterator operator++()** – префиксен оператор за инкрементиране (it = ++v.begin())
- **iterator operator++(int)** – постфиксен оператор за инкрементиране (it = v.begin()++)
- **bool operator!=()** - проверява дали адресите на два Node-а са различни

След това в класа List да бъдат добавени следните функции:

11. iterator begin() - връща iterator към началото на списъка

12. iterator end() - връща iterator към края на списъка (един елемент след края на списъка)

13. void insert(iterator it, const T& value) - вмъква елемент със стойност value на позиция iterator

14. void erase(iterator it) - изтрива елемент на позиция iterator

15. int size() - връща броя елементи в списъка

16. void clear() - изтрива всички елементи на списъка

17. bool empty() - проверява дали списъкът е празен

За улеснение първо може да направите List да бъде едносвързан списък, работещ с int, след което да го направите двусвързан и финално да направите класа шаблонен.

3. Примерна употреба

```
int main()
{
	List<int> list1;

	list1.push_front(100);
	list1.push_front(200);
	list1.push_front(300);
	list1.push_back(777);

	cout << list1.back() << endl; //777

	list1.pop_back();

	cout << list1.back() << endl; //100
	cout << list1.front() << endl; //300

	list1.pop_front();

	cout << list1.front() << endl; //200

	List<int> list2;

	list2.push_back(616);
	list2.push_front(515);
	list2.push_front(313);
	list2.push_back(777);

	//Извежда 313 515 616 777
	for (List<int>::iterator it = list2.begin(); it != list2.end(); it++)
	{
		cout << *it << " ";
	}

	cout << endl;

	List<string> list3;
	list3.push_back("vidi");
	list3.push_back("vici");

	List<string>::iterator iter = list3.begin();

	list3.insert(iter, "Veni");

	//Извежда Veni vidi vici
	for (List<string>::iterator it = list3.begin(); it != list3.end(); it++)
	{
		cout << *it << " ";
	}

	cout << endl;

	List<string> list4;
	list4.push_back("Divide");
	list4.push_back("et");
	list4.push_back("impera");

	//Извежда Divide et impera
	for (List<string>::iterator it = list4.begin(); it != list4.end(); it++)
	{
		cout << *it << " ";
	}

	List<string>::iterator mid = ++list4.begin();
	list4.erase(mid);

	//Извежда Divide impera
	for (List<string>::iterator it = list4.begin(); it != list4.end(); it++)
	{
		cout << *it << " ";
	}

	cout << endl;
}
```