Programowanie i metody numeryczne

Zadania – seria 10.

Algebra liniowa.

Zadanie. `lusolver` – Rozwiązywanie układów równań liniowych metodą LU.

Napisz szablon klasy

```
template <typename T>
class Vector
```

reprezentującej wektor n–wymiarowy, którego składowe są liczbami reprezentowanymi przez typ `T`.

W klasie tej zaimplementuj:

- konstruktor jednoargumentowy, ustalający wartość n równą liczbie przekazanej mu jako argument,
- metodę `Length` zwracającą długość wektora,
- operator `==` porównywania wektorów,
- jednoargumentowy operator zmiany znaku `-`,
- operator `+` dodawania wektorów,
- operator `*` mnożenia wektora przez liczbę (reprezentowaną przez typ `T`),
- operator `*` iloczynu skalarnego wektorów (wynik powinien być typu `T`),
- operator `<<`, wypisujący do strumienia typu `ostream` wektor reprezentowany przez obiekt, oraz operator `>>`, wczytujący odpowiedni wektor ze strumienia typu `istream`; przyjmij dowolny, wygodny dla Ciebie sposób tekstowego reprezentowania wektora,
- statyczną metodę `RandomVector`, zwracającą wektor wypełniony losowymi liczbami całkowitymi z przedziału określonego argumentami tej metody.

Napisz ponadato szablon klasy

```
template <typename T>
class Matrix
```

reprezentującej macierz kwadratową stopnia n, której współczynniki są liczbami reprezentowanymi przez typ `T`.

W klasie tej zaimplementuj:

- konstruktor jednoargumentowy, ustalający wartość n równą liczbie przekazanej mu jako argument,
- operator `==` porównywania macierzy,
- jednoargumentowy operator zmiany znaku `-`,
- operator `+` dodawania macierzy,

- operator `*` mnożenia macierzy przez liczbę (reprezentowaną przez typ `T`),
- operator `*` mnożenia macierzy przez wektor, czyli obiekt napisanej przez Ciebie klasy `Vector<T>` (wynik powinien być typu `Vector<T>`),
- operator `*` iloczynu macierzy,
- operator `<<`, wypisujący do strumienia typu `ostream` macierz reprezentowaną przez obiekt, oraz operator `>>`, wczytujący odpowiednią macierz ze strumienia typu `istream`; przyjmij dowolny, wygodny dla Ciebie sposób tekstowego reprezentowania macierzy,
- metodę

  ```
  std::pair<Matrix<T>, Matrix<T>> DecomposeLU()
  ```

  zwracającą parę macierzy złożoną z macierzy górnotrójkątnej i dolnotrójkątnej, których iloczyn jest równy macierzy reprezentowanej przez instancję klasy; posłuż się metodą Doolittle'a,
- statyczną metodę `RandomMatrix`, zwracającą macierz wypełnioną losowymi liczbami całkowitymi z przedziału określonego argumentami tej metody.

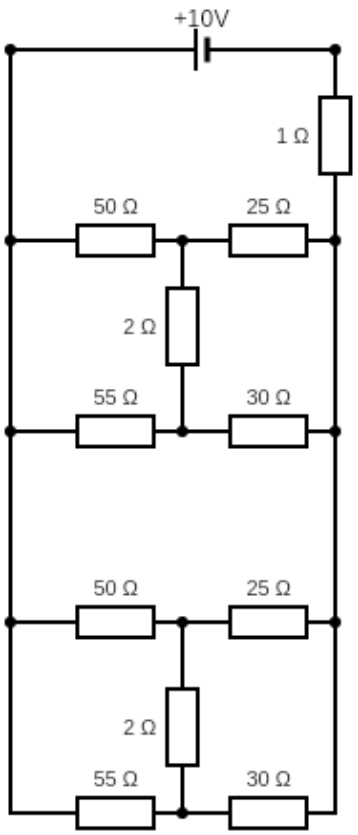
Napisz także szablony następujących funkcji:

- ```
  template <typename T>
  Vector<T> SolveU(const Matrix<T> &U,
                   const Vector<T> &y)
  ```

  zwracającej wektor x stanowiący rozwiązanie równania $Ux = y$ przy założeniu, że macierz U jest górnotrójkątna,
- ```
  template <typename T>
  Vector<T> SolveL(const Matrix<T> &L,
                   const Vector<T> &y)
  ```

  zwracającej wektor x stanowiący rozwiązanie równania $Lx = y$ przy założeniu, że macierz L jest dolnotrójkątna,
- ```
  template <typename T>
  Vector<T> Solve(const Matrix<T> &C,
                  const Vector<T> &y)
  ```

  zwracającej wektor x stanowiący rozwiązanie równania $Cx = y$ dla dowolnej macierzy C; funkcja ta powinna wykorzystywać rozkład LU macierzy C.



Korzystając z tych szablonów napisz program `lusolver`, który wczytuje ze standardowego wejścia macierz C oraz wektor y, a następnie wypisuje na standardowe wyjście wektor x stanowiący rozwiązanie równania $Cx = y$.

Wykorzystaj napisany przez siebie program do znalezienia wartości prądów płynących w obwodzie przedstawionym na rysunku obok. W tym celu zapisz układ równań wynikających z praw Kirchhoffa w postaci macierzowej, umieszczając wartości prądów w wektorze niewiadomych.

Opracowanie: Bartłomiej Zglinicki.
  ```