МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ІВАНА ФРАНКА
Факультет електроніки і комп'ютерних технологій
Кафедра системного проектування

Звіт про виконання лабораторної роботи №4
з начальної дисципліни

«Паралельне програмування»

на тему:

«Реалізація LU-розкладу матриці за допомогою завдань (tasks) в OpenMP програмах»

Виконав:
студент групи ФЕП-22

Линва В. А.

Львів – 2021

Хід роботи

1. Написав програму LU -розкладу матриці у трьох режимах роботи:
   1) послідовному,
   2) паралельному за допомогою директиви #pragma omp for,
   3) паралельному за допомогою директиви #pragma omp task.
2. Написав функцію для заповнення матриці довільного розміру випадковими числами.
3. Написав функцію для перевірки правильності LU -розкладу матриці за допомогою формули .
4. Виконав порівняння швидкодії LU -розкладу матриці для трьох режимів роботи програми у випадках, коли розмір матриці дорівнює n = 10, 100, 500, 1000 елементів.

Код програми:

```
#include <iostream>
#include <iomanip>
#include <Math.h>
#include <omp.h>


using namespace std;

int main()
{
   int G;
   double start_time, end_time;
   cout << "Chooes program type:\n1- Standart\n2- Parellel v1(pragma omp for)\n3- Paralell v2(pragma omp task)\nYou
choose: ";
   cin >> G;
   switch (G)
   {
   case 1:
   {
      long long const N = 100;
      long double mas[N][N + 1],
         L[N][N + 1],
         U[N][N + 1],
         x[N]; // Корені системи
      long double sum = 0;
      int i, j, k, n;

      cout << "Enter the size of the matrix: "; cin >> n;
      start_time = omp_get_wtime();
      srand(time(0));
      for (int i = 0; i < n; i++)
      {
         for (int j = 0; j < n; j++)
         {
            mas[i][j] = 1 + rand() % 10;

            L[i][j] = 0;
            U[i][j] = 0;

            if (i == j)
               U[i][j] = 1;
         }
      }
```

```
    //знаходим перший стовпець L[][] і першу стрічку U[][]
    for (int i = 0; i < n; i++)
    {
        L[i][0] = mas[i][0];
        U[0][i] = mas[0][i] / L[0][0];
    }

    //дальше вираховуємо  L[][], U[][] по формулі
    for (int i = 1; i < n; i++)
    {
        for (int j = 1; j < n; j++)
        {
            if (i >= j) //нижній трикутник
            {
                sum = 0;
                for (int k = 0; k < j; k++)
                    sum += L[i][k] * U[k][j];

                L[i][j] = mas[i][j] - sum;
            }
            else // вірхній
            {
                sum = 0;
                for (int k = 0; k < i; k++)
                    sum += L[i][k] * U[k][j];

                U[i][j] = (mas[i][j] - sum) / L[i][i];
            }
        }
    }
    end_time = omp_get_wtime();

    cout << "\nMatrix L\n\n";
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < n; j++)
            cout << " " << L[i][j] << " ";
        cout << "\n\n";
    }

    cout << "\nMatrix U\n\n";

    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < n; j++)
            cout << " " << U[i][j] << " ";
        cout << "\n\n";
    }
    cout << "Time of execution: " << end_time - start_time << endl;
    return 0;
}

case 2:
{
    long long const N = 100;
    long double mas[N][N + 1],
        L[N][N + 1],
        U[N][N + 1],
        x[N]; // Корені системи
    long double sum = 0;
    int i, j, k, n;

    cout << "Enter the size of the matrix: "; cin >> n;
    srand(time(0));
    #pragma omp parallel
```

```
start_time = omp_get_wtime();
#pragma omp for
for (int i = 0; i < n; i++)
{
   for (int j = 0; j < n; j++)
   {
      mas[i][j] = 1 + rand() % 10;

      L[i][j] = 0;
      U[i][j] = 0;

      if (i == j)
         U[i][j] = 1;
   }
}

//знаходим перший стовпець L[][] і першу стрічку U[][]
#pragma omp for
for (int i = 0; i < n; i++)
{
   L[i][0] = mas[i][0];
   U[0][i] = mas[0][i] / L[0][0];
}

//дальше вираховуємо  L[][], U[][] по формулі
#pragma omp for
for (int i = 1; i < n; i++)
{
   for (int j = 1; j < n; j++)
   {
      if (i >= j) //нижній трикутник
      {
         sum = 0;
         for (int k = 0; k < j; k++)
            sum += L[i][k] * U[k][j];

         L[i][j] = mas[i][j] - sum;
      }
      else // вірхній
      {
         sum = 0;
         for (int k = 0; k < i; k++)
            sum += L[i][k] * U[k][j];

         U[i][j] = (mas[i][j] - sum) / L[i][i];
      }
   }
}
end_time = omp_get_wtime();
//start_time = omp_get_wtime();
cout << "\nMatrix L\n\n";
for (int i = 0; i < n; i++)
{
   for (int j = 0; j < n; j++)
      cout << " " << L[i][j] << " ";
   cout << "\n\n";
}

cout << "\nMatrix U\n\n";

for (int i = 0; i < n; i++)
{
   for (int j = 0; j < n; j++)
      cout << " " << U[i][j] << " ";
   cout << "\n\n";
}
```

```
    //end_time = omp_get_wtime();
    cout << "Time of execution: " << end_time - start_time << endl;
    return 0;
}

case 3:
{
    long long const N = 100;
    long double mas[N][N + 1],
        L[N][N + 1],
        U[N][N + 1],
        x[N]; // Корені системи
    long double sum = 0;
    int i, j, k, n;

    cout << "Enter the size of the matrix: "; cin >> n;
    srand(time(0));
    #pragma omp parallel
    start_time = omp_get_wtime();

    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < n; j++)
        {
            #pragma omp task
            mas[i][j] = 1 + rand() % 10;

            L[i][j] = 0;
            U[i][j] = 0;

            if (i == j)
                U[i][j] = 1;
        }
    }

    //знаходим перший стовпець L[][] і першу стрічку U[][]
    #pragma omp task
    for (int i = 0; i < n; i++)
    {
        L[i][0] = mas[i][0];
        U[0][i] = mas[0][i] / L[0][0];
    }

    //дальше вираховуємо  L[][], U[][] по формулі
    for (int i = 1; i < n; i++)
    {
        #pragma omp task
        for (int j = 1; j < n; j++)
        {
            #pragma omp task
            if (i >= j) //нижній трикутник
            {
                #pragma omp task
                sum = 0;
                for (int k = 0; k < j; k++)
                    sum += L[i][k] * U[k][j];

                L[i][j] = mas[i][j] - sum;
            }
            else // вірхній
            {
                #pragma omp task
                sum = 0;
                for (int k = 0; k < i; k++)
                    sum += L[i][k] * U[k][j];
```

```
                U[i][j] = (mas[i][j] - sum) / L[i][i];
            }
        }
    }
    end_time = omp_get_wtime();
    //start_time = omp_get_wtime();
    cout << "\nMatrix L\n\n";
    for (int i = 0; i < n; i++)
    {
        #pragma omp task
        for (int j = 0; j < n; j++)
            cout << " " << L[i][j] << " ";
        cout << "\n\n";
    }

    cout << "\nMatrix U\n\n";

    for (int i = 0; i < n; i++)
    {
        #pragma omp task
        for (int j = 0; j < n; j++)
            cout << " " << U[i][j] << " ";
        cout << "\n\n";
    }
    //end_time = omp_get_wtime();
    cout << "Time of execution: " << end_time - start_time << endl;
    return 0;
}
}

}
```

Результат виконання:

Час	Розмір матриці:	Точність:
0.00250 sec	10x10	0.00001
0.02800 sec	100x100	0.001
0.22300 sec	500x500	0.1

Висновок: на даній лабораторній роботі я ознайомився з директивою task, порівняв з іншими методами розпаралелення (з не явними типами завдань, що створюються автоматично при omp parallel for). Реалізував програму для порівняння між типами розпаралелення та продемонстрував різницю при різних розмірах вхідних даних.