Лабораторная работа 5. Третий уровень. Варианты 1 и 2

Классы для работы с файлами. Сериализация. Взаимодействие управляемого и неуправляемого кода

Требования к программе

В лабораторной работе требуется на языках **C#** и **C++** определить классы для решения систем линейных алгебраических уравнений с блочной трехдиагональной матрицей.
Классы используется для сравнения времени выполнения управляемого кода **C#** и неуправляемого кода **C++**. Код **C++** компилируется в DLL-библиотеку. Код **C#** вызывает методы из DLL-библиотеки **C++** с помощью сервиса **PInvoke**. Для сохранения в файле результатов сравнения времени выполнения кодов **C#** и **C++** используется сериализация.

Определения и формулы для решения систем линейных уравнений с блочной трехдиагональной матрицей приведены в приложении в конце текста.
Варианты отличаются типом матриц, которые являются элементами блочной матрицы.

Вариант	Тип элементов блочной матрицы
1	**Диагональная матрица, порядок которой задается в конструкторе блочной матрицы**
2	**Матрица третьего порядка**

Для матриц, которые являются элементами блочной матрицы, надо определить класс (или структуру).
В первом варианте это класс диагональных матриц. Во втором варианте - класс для матриц третьего порядка. Элементы матриц имеют тип **double**.

В классах матриц-блоков надо определить

- конструкторы для инициализации матриц;
- операции (или статические методы) для сложения и вычитания матриц;
- операцию (или статический метод) для умножения матриц;
- операцию (или статический метод) для умножения матрицы на вектор;
- статический метод для вычисления обратной матрицы.

Можно определить класс для вектора, элементами которого являются массивы **double[]**, длина которых равна порядку матриц-блоков.

Определить класс для блочной трехдиагональной матрицы.
Элементы блочной трехдиагональной матрицы хранятся в массиве (или массивах), элементы которых имеют тип, определенный для матриц-блоков. Если блочный порядок матрицы равен N, то число элементов-блоков для хранения матрицы не должно быть больше, чем 3N.

В классе для блочной трехдиагональной матрицы надо определить

- конструкторы для инициализации матрицы;
- конструктор для инициализации матрицы большого порядка, который используется при сравнении скорости выполнения кодов **C#** и **C++:** в варианте 1 конструктор с двумя параметрами - один параметр для блочного порядка матрицы, другой – для порядка матриц-блоков; в варианте 2 конструктор с одним параметром, который определяет блочный порядок матрицы;

- метод для умножения блочной матрицы на вектор;
- метод для решения системы линейных уравнений;
- метод, который сохраняет в файле в текстовом виде матрицу, правую часть и решение системы линейных уравнений.

При создании матриц большого порядка для сравнения скорости выполнения кодов **C#** и **C++** следует использовать матрицы простой структуры, например, во втором варианте можно использовать диагональные матрицы-блоки, так как время решения системы линейных уравнений определяется только порядком матрицы и не зависит от конкретной матрицы и правой части.
Не следует использовать методы случайной генерации элементов матрицы, так как этот процесс значительно медленнее, чем простое присваивание значений, кроме того, матрица может получиться вырожденной.

В коде **C++** не должно быть утечки памяти, т.е. **вся память**, выделенная динамически с помощью оператора **new**, **должна быть освобождена** с помощью оператора **delete**.

В среде VisualStudio надо создать решение (solution) с тремя проектами:
- тип первого проекта – Dll-библиотека **C++**, которая содержит классы для матриц и глобальную экспортируемую функцию, которая вызывается из управляемого кода **C#**;
- тип второго проекта – Dll-библиотека **C#** (сборка), в которой находятся классы для матриц и класс для накопления результатов сравнения времени выполнения управляемого кода **C#** и неуправляемого кода **C++**;
- тип третьего проекта – консольное приложение **C#**, которое принимает ввод пользователя и в режиме накопления сохраняет результаты сравнения времени выполнения кодов **C#** и **C++**.

Библиотеки классов C# и C++

В библиотеку классов **C#** надо скомпилировать классы для матрицы и класс для накопления результатов сравнения времени выполнения управляемого кода **C#** и неуправляемого кода **C++**.

DLL-библиотека **C++** содержит классы для матрицы и глобальную экспортируемую функцию, которая вызывается из кода **C#**.

В коде глобальной экспортируемой функции создается объект класса для блочной трехдиагональной матрицы, решается система линейных алгебраических уравнений и измеряется время решения системы. Полученное решение и время решения системы надо через параметры глобальной экспортируемой функции вернуть в вызывающий код на C#.

Через параметры глобальной экспортируемой функции надо передать всю необходимую информацию для того, чтобы матрицу, которая была создана в коде на C#, можно было полностью восстановить в коде на C++.

Тестирование решения системы линейных уравнений

В консольном приложении тестируется метод решения системы линейных уравнений и в режиме накопления сохраняются результаты сравнения времени выполнения кодов **C#** и **C++**.

Для проверки, что метод решения системы линейных алгебраических уравнений с блочной треугольной матрицей работает правильно, в консольном приложении **C#** надо определить метод **Test()**.

В методе **Test()** надо

- задать блочную матрицу небольшого порядка и правую часть; как блочная матрица, так и матрицы-элементы не должны быть единичными матрицами; во втором варианте матрицы-блоки не должны быть диагональными;
- решить систему линейных уравнений и сохранить в файле матрицу, правую часть и решение системы линейных уравнений;
- умножить матрицу на полученное решение и вывести результат;
- через параметры глобальной экспортируемой функции передать в код C++ данные для матрицы и правую часть, решить систему линейных уравнений, полученное решение вывести в коде C#.

Сравнение времени выполнения кодов C# и C++.

Для накопления результатов сравнения времени выполнения кодов C# и C++ надо определить тип **TestTime**.

Класс **TestTime** содержит поле (или автореализуемое свойство) **List<string>**. Каждая строка списка содержит следующую информацию

- порядок блочной матрицы и матриц-элементов в первом варианте; порядок блочной матрицы во втором варианте;
- время решения системы линейных уравнений в коде на **C#**;
- время решения системы линейных уравнений в коде на **C++**;
- коэффициент (тип **double)**, который равен отношению времени выполнения кодов **C#** и **C++;**

Информация в строке должна быть подписана.

Класс **TestTime** содержит

- открытый метод **Add(string record)**, добавляющий новую строку в список **List<string>**.;
- открытый статический метод **bool Save(string filename, TestTime obj)** для сохранения объекта **obj** в файле с именем **filename** с использованием сериализации;
- открытый статический метод **bool Load(string filename, ref TestTime obj)** для восстановления объекта **obj** из файла с именем **filename** с использованием сериализации;
- перегруженную версию виртуального метода **string ToString()**.

Статический метод **bool Save(string filename, TestTime obj)**

- сериализует объект **obj** в файл с именем **filename**;
- если файл с именем **filename** существует, приложение его перезаписывает; если такого файла нет, приложение его создает;
- метод возвращает значение **true**, если сериализация завершилась успешно, и значение **false** в противном случае.

Статический метод **bool Load(string filename, ref TestTime obj)**

- десериализует объект **obj** из файла с именем **filename**;
- метод возвращает значение **true**, если десериализация завершилась успешно, и значение **false** в противном случае.

Решение системы линейных уравнений с блочной трехдиагональной матрицей порядка N с матрицами-блоками порядка M требует $O(N \cdot M^3)$ арифметических операций. Время решения системы линейных уравнений с блочной трехдиагональной матрицей должно возрастать в 2 раза при увеличении порядка матрицы в 2 раза. Это соотношение выполняется только для матриц достаточно большого порядка, так как для матриц малого порядка накладные расходы на вызов

метода, распределение и освобождение памяти в методе решения системы линейных уравнений будут сравнимы со временем выполнения арифметических операций.

В методе **Main()**

1. Вызвать метод **Test()** для проверки, что метод решения системы линейных уравнений работает правильно.
2. Создать объект типа **TestTime**.
3. Вызвать статический метод **bool Load(string filename, ref TestTime obj)** класса **TestTime** для инициализации объекта **TestTime.** Если файла с заданным именем нет, то в объекте **TestTime** создается пустая коллекция **List<string>**.
4. Затем пользователь получает приглашение ввести порядок блочной матрицы и порядок матриц-элементов в первом варианте и порядок блочной матрицы во втором варианте или завершить работу приложения:
    - если пользователь корректно ввел данные, выполняются вычисления (на **C#** и **C++**) и в список **List< string>** добавляется новая строка; затем пользователь снова получает приглашение ввести значения или завершить работу приложения.
    - если пользователь ввел данные с ошибками, он снова получает приглашение ввести данные или завершить работу приложения.
5. Когда пользователь завершает работу приложения
    - выводится вся коллекция **List<string>** из объекта **TestTime;**
    - вызывается статический метод **bool Save(string filename, TestTime obj)** класса **TestTime** для сохранения в файле объекта **TestTime** с использованием сериализации.
6. Все исключения, которые могут возникнуть при работе приложения, в том числе из-за ошибок при вводе данных, должны быть обработаны. Все операции открытия файла, сериализации и десериализации данных должны находиться в блоках **try-catch-finally**. Приложение должно обрабатывать все исключения, которые могут возникнуть из-за ошибок при вводе данных. Независимо от того, корректно были введены данные или при вводе были допущены ошибки, все файловые потоки должны быть закрыты.

Лабораторную работу надо прислать на почту berezina@cs.msu.su **до 17 декабря 2019 (последний день отправки).**

Надо прислать все решение (solution) с тремя проектами. Перед тем, как отправить работу, можно удалить каталоги и файлы, которые будут созданы при компиляции решения:

- каталоги obj и bin в проектах C#;
- каталог DEBUG(RELEASE), который относится к проекту C++ и находится в каталоге с решением.

Приложение

Метод матричной прогонки для решения системы линейных уравнений с блочной трехдиагональной матрицей

Система линеных алгебраических уравнений для блочной трехдиагональной матрицы имеет вид

$$\begin{pmatrix} C_0 & B_0 & 0 & 0 & . & . & 0 & 0 & 0 & 0 \\ A_1 & C_1 & B_1 & 0 & . & . & 0 & 0 & 0 & 0 \\ 0 & A_2 & C_2 & B_2 & . & . & 0 & 0 & 0 & 0 \\ . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . \\ 0 & 0 & 0 & 0 & . & . & A_{N-3} & C_{N-3} & B_{N-3} & 0 \\ 0 & 0 & 0 & 0 & . & . & 0 & A_{N-2} & C_{N-2} & B_{N-2} \\ 0 & 0 & 0 & 0 & . & . & 0 & 0 & A_{N-1} & C_{N-1} \end{pmatrix} * \begin{pmatrix} X_0 \\ X_1 \\ X_2 \\ . \\ . \\ X_{N-3} \\ X_{N-2} \\ X_{N-1} \end{pmatrix} = \begin{pmatrix} F_0 \\ F_1 \\ F_2 \\ . \\ . \\ F_{N-3} \\ F_{N-2} \\ F_{N-1} \end{pmatrix}$$

A_i, B_i, C_i - квадратные матрицы порядка M; X_i, F_i - векторы длины M. Порядок матрицы системы линейных уравнений как блочной матрицы равен N, точечный порядок матрицы системы равен $N \times M$.

В методе матричной прогонки для решения системы линейных уравнений с блочной трехдиагональной матрицей сначала вычисляются прогоночные коэффициенты – матрицы $\alpha_i, i = 1,...N-2$ порядка M и векторы $\beta_i, i = 1,...N-1$ длины M.

Формулы для вычисления прогоночных коэффициентов:

$$\alpha_1 = C_0^{-1} B_0$$
$$\alpha_{i+1} = \left(C_i - A_i \alpha_i\right)^{-1} B_i, \; i = 1, 2, ...N-2$$

$$\beta_1 = C_0^{-1} F_0$$
$$\beta_{i+1} = \left(C_i - A_i \alpha_i\right)^{-1} \left(F_i - A_i \beta_i\right), \quad i = 1, 2, ...N-1$$

Формулы для вычисления вектора решения системы уравнений

$$X_{N-1} = \beta_N$$
$$X_i = \beta_{i+1} - \alpha_{i+1} X_{i+1}, \quad i = \left(N-2\right), ...0.$$

Литература.

1. А.А.Самарский, А.В.Гулин. Численные методы. - М.: Наука, 1989.