**Построение решетчатого графа в трехмерном пространстве**

Для построения решетчатого графа в трехмерном пространстве было реализовано desktop приложение на языке C# с использованием Windows Forms в среде Visual Studio.

Главное окно приложения представлено на рисунке 1.



Рисунок 1 – Главное окно приложения

Задачу построения решетчатого графа в трехмерном пространстве можно условно разделить на 4 этапа:

1. Анализ введенной программы, разбор ее на синтаксические элементы
2. Выявление зависимостей переменных в программе
3. Получение списка зависимых переменных на множестве допустимых итераций программы
4. Построение графика в трехмерном пространстве по полученному списку

Эти этапы четко прослеживаются в разделении интерфейса программы на блоки элементов.

1. Анализ введенной программы, разбор ее на синтаксические элементы

Для анализа введенной программы были созданы классы Счетчик (**CounterClass**), Переменная (**VariableClass**), Программа (**ProgramClass**).

Класс Счетчик отвечает за переменную, использующуюся в цикле. Например, для цикла

 for (int i = 0; i < 10; i++)

это будет переменная c именем **“i”**, значение ОТ – **“0”**, уравнение ДО - **“<”**, имя переменной значения ДО пусто, значение ДО – **“10”**, значение инкремента – **“1”**.

Ниже приведено краткое описание класса Счетчик.

 /// <summary>

 /// Класс Счетчик

 /// </summary>

 class CounterClass

 {

 /// <summary>

 /// имя переменной счетчика

 /// </summary>

 public string CounterName

 {

 get { return pCounterName; }

 set { pCounterName = value; }

 }

 /// <summary>

 /// значение ОТ счетчика

 /// </summary>

 public int CounterMinValue

 {

 get { return pCounterMinValue; }

 set { pCounterMinValue = value; }

 }

 /// <summary>

 /// уравнение ДО счетчика

 /// </summary>

 public string CounterMaxEquation

 {

 get { return pCounterMaxEquation; }

 set { pCounterMaxEquation = value; }

 }

 /// <summary>

 /// имя переменной - значение ДО счетчика

 /// </summary>

 public string CounterMaxName

 {

 get { return pCounterMaxName; }

 set { pCounterMaxName = value; }

 }

 /// <summary>

 /// значение ДО счетчика

 /// </summary>

 public int CounterMaxValue

 {

 get { return pCounterMaxValue; }

 set { pCounterMaxValue = value; }

 }

 /// <summary>

 /// значение инкремента счетчика

 /// </summary>

 public int CounterIncrement

 {

 get { return pCounterIncrement; }

 set { pCounterIncrement = value; }

 }

 }

При анализе введенного текста программы проверяется количество найденных счетчиков. Для трехмерного решетчатого графа оно должно быть равно 3. В противном случае программа выдаст ошибку.

Класс Переменная предназначен для хранения информации о найденных в теле программы индексных переменных. Например, для кода

 x[i-1][j+1][k] = 0

это будет переменная с именем **“x”**, имя первой индексной переменной – **“i”**, инкремент первой индексной переменной – **“-1”**, имя второй индексной переменной – **“j”**, инкремент второй индексной переменной – **“1”**, имя третьей индексной переменной – **“k”**, инкремент первой индексной переменной – **“0”**.

Ниже приведено краткое описание класса Переменная.

 /// <summary>

 /// Класс Переменная

 /// </summary>

 class VariableClass

 {

 /// <summary>

 /// Имя переменной

 /// </summary>

 public string VariableName

 {

 get { return pVariableName; }

 set { pVariableName = value; }

 }

 /// <summary>

 /// Имя первой индексной переменной (i)

 /// </summary>

 public string Index1Name

 {

 get { return pIndex1Name; }

 set { pIndex1Name = value; }

 }

 /// <summary>

 /// Инкремент первой индексной переменной (+1)

 /// </summary>

 public int Index1Increment

 {

 get { return pIndex1Increment; }

 set { pIndex1Increment = value; }

 }

 /// <summary>

 /// Значение первой индексной переменной (i)

 /// </summary>

 public int Index1Value

 {

 get { return pIndex1Value; }

 set { pIndex1Value = value; }

 }

 /// <summary>

 /// Имя второй индексной переменной (j)

 /// </summary>

 public string Index2Name

 {

 get { return pIndex2Name; }

 set { pIndex2Name = value; }

 }

 /// <summary>

 /// Инкремент первой индексной переменной (+1)

 /// </summary>

 public int Index2Increment

 {

 get { return pIndex2Increment; }

 set { pIndex2Increment = value; }

 }

 /// <summary>

 /// Значение второй индексной переменной (j)

 /// </summary>

 public int Index2Value

 {

 get { return pIndex2Value; }

 set { pIndex2Value = value; }

 }

 /// <summary>

 /// Имя третьей индексной переменной (k)

 /// </summary>

 public string Index3Name

 {

 get { return pIndex3Name; }

 set { pIndex3Name = value; }

 }

 /// <summary>

 /// Инкремент третьей индексной переменной (+3)

 /// </summary>

 public int Index3Increment

 {

 get { return pIndex3Increment; }

 set { pIndex3Increment = value; }

 }

 /// <summary>

 /// Значение третьей индексной переменной (k)

 /// </summary>

 public int Index3Value

 {

 get { return pIndex3Value; }

 set { pIndex3Value = value; }

 }

 /// <summary>

 /// Зависимости от других переменных

 /// </summary>

 public List<VariableClass> Dependencies

 {

 get { return pDependencies; }

 set { pDependencies = value; }

 }

 /// <summary>

 /// Отмечена

 /// </summary>

 public bool Checked

 {

 get { return pChecked; }

 set { pChecked = value; }

 }

 }

При анализе введенного текста программы для каждой найденной переменной проверяется количество индексных переменных. Для трехмерного решетчатого графа оно должно быть равно 3. В противном случае программа выдаст ошибку.

Свойства “Значение первой индексной переменной”, “Значение второй индексной переменной”, “Значение третьей индексной переменной” используются на этапе получения списка зависимых переменных на множестве допустимых итераций программы и хранят в себе фактические значения индексных переменных в зависимости от значений переменных, используемых в циклах.

Свойство **Dependencies** хранит список зависимых переменных.

Свойство **Checked** определяет, использовать ли переменную для анализа и построения графа. Определяется пользователем в интерфейсе.

Класс Программа хранит введенный текст программы, список найденных счетчиков и переменных, а также признак того, что программа проверена без ошибок.

Ниже приведено краткое описание класса Программа.

 /// <summary>

 /// класс - программа

 /// </summary>

 class ProgramClass

 {

 /// <summary>

 /// текст программы

 /// </summary>

 public string[] Text

 {

 get { return pText; }

 }

 /// <summary>

 /// счетчики

 /// </summary>

 public CounterClass[] Counters

 {

 get { return pCounters; }

 set { pCounters = value; }

 }

 /// <summary>

 /// переменные

 /// </summary>

 public List<VariableClass> Variables

 {

 get { return pVariables; }

 set { pVariables = value; }

 }

 /// <summary>

 /// Программа проверена?

 /// </summary>

 public bool Verified

 {

 get { return pVerified; }

 set { pVerified = value; }

 }

 }

При нажатии кнопки “Проверить программу” выполняется анализ введенной программы, выделение циклов и переменных. Выполняются необходимые проверки. В случае обнаружения ошибки, выдается соответствующее окно.

1. Выявление зависимостей переменных в программе

После проверки программы выполняется анализ зависимостей переменных. Для каждой найденной переменной заполняется свойство **Dependencies**.

Например, для кода

 x[i-1][j][k] = x[i][j][k]+a[i][j][k]

 x[i][j][k] = x[i+1][j+1][k]+b[i][j][k]

у переменной **x[i-1][j][k]** зависимыми будут переменные **x[i][j][k]** и **x[i+1][j+1][k]**, а у переменной **x[i][j][k]** – **x[i+1][j+1][k]**. Реализованный алгоритм проверяет непрямые зависимости через другие переменные, а “лишние” переменные, у которых нет зависимостей (в нашем случае это **a** и **b**) отбрасывает.

При корректном анализе, приложение выводит список найденных переменных и зависимых переменных. Пользователю предлагается отметить галочками те переменные, которые необходимо отображать на графике.

1. Получение списка зависимых переменных на множестве допустимых итераций программы

При успешном анализе текста программы, приложение выводит имена переменных – максимумов для циклов. Пользователю предлагается ввести максимальные значения этих переменных. Если границы цикла обозначены статически, выводится стандартное имя переменной (N1, N2, N3) и граница цикла заполняется автоматически.

Класс Итерация (**IterationClass**) хранит в себе информацию о конкретной итерации из множества допустимых в программе. Например, для кода

 for (i=1; i<=2; i++)

 for (j=1; j<=2; j++)

 for (k=1; k<=2; k++)

 {

 x[i-1][j+1][k] = 0

 }

будут получены 8 итераций:

1. (i = 1; j = 1; k = 1) (x[0][2][1])
2. (i = 1; j = 1; k = 2) (x[0][2][2])
3. (i = 1; j = 2; k = 1) (x[0][3][1])
4. (i = 1; j = 2; k = 2) (x[0][3][2])
5. (i = 2; j = 1; k = 1) (x[1][2][1])
6. (i = 2; j = 1; k = 2) (x[1][2][2])
7. (i = 2; j = 2; k = 1) (x[1][3][1])
8. (i = 2; j = 2; k = 2) (x[1][3][2])

Ниже приведено краткое описание класса Итерация.

 /// <summary>

 /// Класс Итерация

 /// </summary>

 class IterationClass

 {

 /// <summary>

 /// значение первого счетчика

 /// </summary>

 public int I

 {

 get { return pI; }

 set { pI = value; }

 }

 /// <summary>

 /// значение второго счетчика

 /// </summary>

 public int J

 {

 get { return pJ; }

 set { pJ = value; }

 }

 /// <summary>

 /// значение третьего счетчика

 /// </summary>

 public int K

 {

 get { return pK; }

 set { pK = value; }

 }

 /// <summary>

 /// переменные

 /// </summary>

 public List<VariableClass> Variables

 {

 get { return pVariables; }

 set { pVariables = value; }

 }

 }

При нажатии кнопки “Построить граф” формируется список доступных итераций. Для каждой итерации формируется список переменных с конкретными значениями соответствующих индексных переменных. Для каждой переменной формируется список зависимых переменных с конкретными значениями соответствующих индексных переменных.

1. Построение графика в трехмерном пространстве по полученному списку

Для отображения графика в трехмерном пространстве используется компонент **Canvas** из проекта **3D Rendering**, выполняющий автоматическую проекцию трехмерной графики на двухмерную плоскость, в зависимости от установленных значений положения и наклона виртуальной камеры (взгляда пользователя).

Компонент имеет возможность вращения камеры, ее приближение, отдаления и перемещения в пространстве. Также доступна смена проекции изображения – трехмерная, фронтальная, вид сверху, вид слева.

Для всех найденных в итерациях переменных построим на графике точки с координатами, соответствующими значениям индексных переменных. Например, переменной x[1][2][3] соответствует точка на графике с координатами X=1, Y=2, Z=3.

Для всех зависимых переменных построим на графике стрелки, соединяющие точку, соответствующую зависимой переменной, с точкой, соответствующей переменной, от которой она зависит. Например, для переменной x[1][2][3] и ее зависимой переменной x[2][3][4] это будет стрелка, соединяющая точку с координатами X=2, Y=3, Z=4 и точку с координатами X=1, Y=2, Z=3.

Искомый решетчатый граф построен (рисунок 2).



Рисунок 2 – Решетчатый граф