14. **Zimmermann, H.** Die Beredhnung des Eibahn-oberbaues [Текст] / H. Zimmermann.- Berlin, 1988 - 186 p.
15. **Маруфий, А.Т.** Математическое моделирование задач изгиба различных осей плит на деформируемом основании с особенностью в оснований [Текст] / А.Т. Маруфий, К.Т. Мансуров. - Б.: Илим, НАН КР, 2014. - 145 с.

Поступила в редакцию 24.02.2021 г.

УДК 681.3+519.1

Ормонова Э. М.
аспирант Кыргызско -Узбекского Междун. универ., Кыргызская Республика

ГРАФТЫН ТЕОРИЯСЫНЫН НЕГИЗИНДЕ ПРОГРАММАЛЫК КАРАЖАТТЫН САПАТЫН АНЫКТОО

Бул макалада изилдөөнүн предмети катары компьютердик программа, тактап айтканда Pascal ABC программалоо тили алынды. Изилдөөнүн максаты графтын теориясын колдонуу менен программалык каражаттын сапатын аныктоо, графтын теориясы үчүн А. Н. Колмогоровдун дифференциалды теңдемесин түзүү жана квадраттык матрицанын жардамында ал теңдемени чыгаруу болуп эсептелет. Жумушту аткарууда программалык каражаттардын сапатын аныктоо үчүн графтын теориясы жана Марковдун чынжыры колдонулат. Багытталган графты колдонуу менен программанын ишенимдүүлүгү аныкталды, ал эми алгебралык теңдемелер системасы квадраттык матрицаны колдонуу менен чыгарылды. Изилденип жаткан, багытталган граф үчүн Колмогоровдун дифференциалдык теңдемелер системасы алынды жана математикалык модели түзүлдү, Марковдун чынжыры жана графтын теориясы методдору менен программалык каражаттын сапатын баалоо жолдору иштелип чыкты. Бул макаланын өзгөчөлүгү компьютердик программага граф түзүү болуп эсептелет. Багытталган графтын негизинде түзүлгөн, Колмогоровдун теңдемесинин жардамында компьютердик программанын сапатынын ыктымалдуу баалоосунун модели сунушталды.

Негизги сөздөр: *графтын теориясы; марковдун чынжыры; программалык каражаттын сапаты; дифференциалдык теңдеме; программалоо тили.*

ОПРЕДЕЛЕНИЕ КАЧЕСТВА ПРОГРАММНОГО ПРОДУКТА НА ОСНОВЕ ТЕОРИИ ГРАФОВ

В данной статье предметом исследования является компьютерная программа, а именно язык программирования Pascal ABC. Поставлена цель определения качества программного продукта с использованием теории графов, составления дифференциальных уравнений А.Н. Колмогорова для теории графа и решение этих уравнений с помощью квадратной матрицы. В работе для определения качества программного продукта используется теории графов и цепи Маркова. Надежность программы определено с использованием ориентированного графа, система алгебраических уравнений решена с использованием квадратной матрицы. Создана математическая модель и получена система дифференциальных уравнений Колмогорова для исследуемого, ориентированного графа, разработаны способы оценки качества програмного средства методами

теории графов и цепи Маркова. Особенностью этой статьи является составление графа для компьютерных программ. Предложены модели вероятностной оценки качества компьютерных программ с помощью уравнений Колмогорова, составленные на основе ориентированного графа.

Ключевые слова: *теория графа; марковские цепи; качества программного продукта; дифференциальные уравнения; язык программирования.*

DETERMINING THE QUALITY OF THE SOFTWARE PRODUCT BASED ON THE THEORY OF GRAPHS

In this article, the subject of research is a computer program, namely the Pascal ABC programming language. The goal is to determine the quality of the software product using graph theory, to compose Kolmogorov's differential equations for graph theory and to solve these equations using a square matrix. The work uses graph theory and Markov chains to determine the quality of a software product. The reliability of the program was determined using a directed graph, the system of algebraic equations was solved using a square matrix. A mathematical model is created and a system of Kolmogorov differential equations for the investigated oriented graph is obtained, methods for assessing the quality of a software tool using the methods of graph theory and Markov chains are developed. The features of this article are considered to be the compilation of a graph for computer programs. Models of probabilistic assessment of the quality of computer programs using the Kolmogorov equations, based on a directed graph, are proposed.

Keywords: *graph theory; Markov chains; software product quality; differential equations; programming language.*

В современном мире очень много разнообразных компьютерных программ. Чтобы эти программы были качественными, надо определить качество программного средства в контексте международных и межгосударственных стандартов [1].

При определении качества программного обеспечения (программного продукта) актуальной является задача эффективного прогнозирования надежности программного средства и определение вероятностного распределения ошибок в программе [2]. Поэтому для определения отдельных показателей качества программного продукта применяются стохастические методы, теории графов и марковские цепи, позволяющие найти количественные значения параметра качества программы [3].

В настоящее время теория граф востребованы во многих областях: в химии, информатике, программировании, экономике, логистике, схемотехнике и пр. В математике, например, граф — это абстрактное представление множества объектов и связей между ними. Графом называют пару (V, E) где V это множество вершин, а E множество пар, каждая из которых представляет собой связь (эти пары называют рёбрами).

Граф может быть ориентированным или неориентированным. В ориентированном графе, связи являются направленными (то есть пары в E являются упорядоченными, например, пары (a, b) и (b, a) это две разные связи). В свою очередь в неориентированном графе, связи ненаправленные, и поэтому если существует связь (a, b) то значит, что существует связь (b, a) [4].

Задачей наших исследований является определение надёжности программного продукта на языке программирования *Pascal ABC* с использованием ориентированного графа.

Рассмотрим язык программирование *Pascal ABC* и требуется определить, является ли заданное трехзначное число палиндромом (палиндром читается одинаково слева направо и справа налево, например, палиндромами являются числа 121, 282, слова «шалаш», «наган»):

```
Program palindrom;
uses crt;
var x : integer;
begin
clrscr;
write('Введите целое число:');
readln(x);
if x mod 10 = x div 160 then write('Введенное число является палиндромом')
else write('Введенное число не является палиндромом');
readln;
end.
```

Составляем граф программы, представленной на рисунке 1.

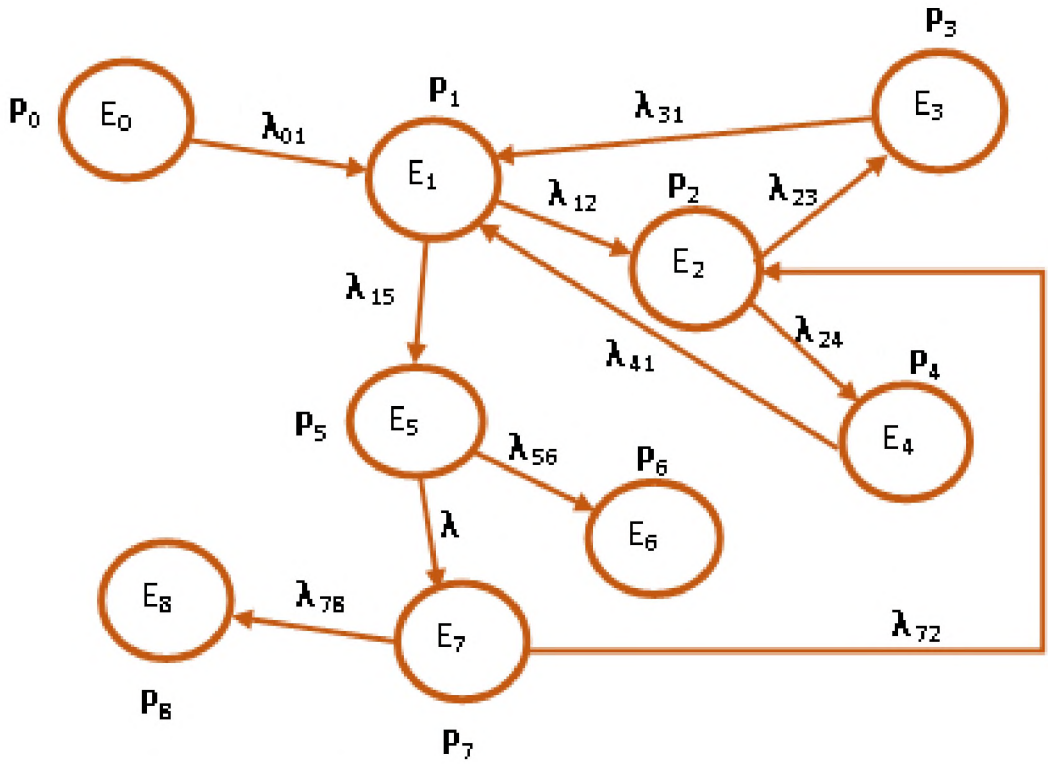

Рисунок 1 – Граф программы:

E_0 – запуск программы; E_1 – ввод данных; E_2 – контроль вводимых данных; E_3 – обнаружение ошибок и ее исправление; E_4–пользователь не обнаружил ошибку ; E_5 – программа выполняет свою функцию; E_6 – программный отказ и восстановление программы; E_7 – контроль результатов, выход данных; E_8 – завершение работы программы.

Используя полученный граф составим таблицу инцидентности (таблица 1).

Таблица 1- Таблица инцидентности

	0	1	2	3	4	5	6	7	8
P_0	-1	0	0	0	0	0	0	0	0
P_1	1	0	-1	1	1	-1	0	0	0
P_2	0	1	0	-1	-1	0	0	-1	0
P_3	0	-1	1	0	0	0	0	0	0
P_4	0	-1	1	0	0	0	0	0	0
P_5	0	1	0	0	0	0	-1	-1	0
P_6	0	0	0	0	0	1	0	0	0
P_7	0	0	1	0	0	1	0	0	-1
P_8	0	0	0	0	0	0	0	1	0

Здесь *Po, P1, P2,P3....P8* – вероятности переходов из одного состояния в другое, *а* $+\lambda$ коэффициенты переходов. Матрица инцидентности для определения качества программного продукта является инструментом качественного анализа программного продукта [5].

Для аналитического определения параметров качества программ нужно составить уравнения Колмогорова. Используя общее правило составления уравнений Колмогорова для нашего графа (цепи Маркова)получим дифференциальные уравнения.

Используя данный алгоритм получили уравнение Колмогорова для данного графа 1 (схема1) и на основе условия стационарности окончательно получим:

$$
\begin{gathered}
-\lambda_{01}P_0 = 0; \\
\lambda_{31}P_3 + \lambda_{41}P_4 + \lambda_{01}P_0 - (\lambda_{12} + \lambda_{15})P_1 += 0; \\
\lambda_{72}P_7 + \lambda_{12}P_1 - (\lambda_{24} + \lambda_{23})P_2 += 0; \\
\lambda_{23}P_2 - \lambda_{31}P_3 = 0; \\
\lambda_{24}P_2 - \lambda_{41}P_4 = 0; \\
\lambda_{65}P_6 + \lambda_{15}P_1 - (\lambda_{56} + \lambda_{57})P_5 = 0; \\
\lambda_{65}P_6 - \lambda_{65}P_6 = 0; \\
\lambda_{57}P_5 - (\lambda_{78} + \lambda_{72})P_7 = 0; \\
\lambda_{78}P_7 = 0;
\end{gathered}
\qquad (1)
$$

Из системы уравнений (1) получим следующую квадратную матрицу.

Введем нижеследующее определение для определителя квадратной матрицы любого порядка. Определитель квадратной матрицы A будем обозначать $|A|$ или det A .

Определение 1. *Определителем* квадратной матрицы

$$A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \quad (2)$$

второго порядка называется число

$$|A| = a_{11}a_{22} - a_{12}a_{21} \quad (3)$$

Определителем

$$A = \begin{vmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{vmatrix} \quad (4)$$

квадратной матрицы порядка n, $n \geq 3$, называется число

$$|A| = \sum_{k=1}^{n} (-1)^{k+1} a_{1k} M_k, \quad (5)$$

где M_k - определитель матрицы порядка $n-1$, полученной из матрицы A вычеркиванием первой строки и столбца с номером k [6].

В нашем случае определитель квадратной матрицы порядка 9. С помощью квадратной матрицы решим полученные нами таблицу инцидентности.

$$|A| = \sum_{k=1}^{9} (-1)^{k+1} a_{1k} M_k \quad (6)$$

$$A = \begin{vmatrix} -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & -1 & 1 & 1 & -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 & -1 & 0 & 0 & -1 & 0 \\ 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & -1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{vmatrix}$$

$$= - \begin{vmatrix} 1 & 0 & -1 & 1 & 1 & -1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 & -1 & 0 & 0 & -1 & 0 \\ 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & -1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{vmatrix}$$

$$= -\begin{vmatrix} 1 & 0 & -1 & 1 & 1 & -1 & 0 & 0 & 0 \\ 0 & 0 & -2 & 2 & 2 & -2 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 & -1 & 0 & 0 & -1 & 0 \\ 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & -1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{vmatrix}$$

$$= -1 * (-1)^2 \begin{vmatrix} 0 & -2 & 2 & 2 & -2 & 0 & 0 & 0 \\ 1 & 0 & -1 & -1 & 0 & 0 & -1 & 0 \\ -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & -1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{vmatrix}$$

$$= -(-)\begin{vmatrix} 1 & 0 & -1 & -1 & 0 & 0 & -1 & 0 \\ 0 & -2 & 2 & 2 & -2 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & -1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{vmatrix}$$

$$= \begin{vmatrix} 1 & 0 & -1 & -1 & 0 & 0 & -1 & 0 \\ 0 & -2 & 2 & 2 & -2 & 0 & 0 & 0 \\ 0 & 1 & -1 & -1 & 0 & 0 & -1 & 0 \\ 0 & 1 & -1 & -1 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{vmatrix}$$

$$= -\begin{vmatrix} 1 & -1 & -1 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -2 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & -1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{vmatrix}$$

$$= -1 * (-1)^4 \begin{vmatrix} -1 & -1 & 0 & 0 & -1 & 0 \\ 0 & 0 & -2 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & -1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{vmatrix}$$

$$= -(-) \begin{vmatrix} 1 & 1 & 1 & 0 & 1 & -1 \\ 0 & 0 & -2 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{vmatrix}$$