УДК 681.3

СЕЛЕЦЬКИЙ В.С., к.т.н., провідний інженер (ІОЦ Львівської залізниці)

Мережі Петрі і мова програмування Delphi

Представив д.т.н., професор Загарій Г.І.

Розроблено автоматизоване програмне забезпечення аналізу функціонування паралельних технологічних процесів ПРСС на підставі моделі, яка побудована за допомогою математичного апарату мереж Петрі.
***Ключові слова**: мова Delphi, умовний оператор, оператор циклу, блок-схема, мережа Петрі, конструкція, модель.*

Вступ

В зв'язку з прискореним темпом розвитку багатоядерних і багатопроцесорних обчислювальних систем особливо важливим на даний час є завдання створення паралельного програмного забезпечення, що передбачає виникнення нових комп'ютерних технологій по створенню паралельних програм. Важливим питанням при цьому є неперервне підвищення продуктивності обчислювальних систем та ефективності використання апаратних ресурсів. В зв'язку з цим важливим завданням є створення автоматизованих програмних засобів аналізу паралельних програм. Ця задача найбільш вдало вирішується із застосуванням модельного підходу при використанні засобів візуального представлення топології та функціонування паралельного алгоритму у сполученні з аналітичними засобами тестування досліджуваної алгоритмічної моделі.

Для дослідження конструкцій алгоритмічної мови Delphi [4] застосовано математичний апарат мереж Петрі [1-3].

Основою інтерпретації мереж Петрі є умови й події. Стан системи (підсистеми) описується сукупністю умов. Функціонування технологічних процесів системи – це послідовність виконання подій. Для виникнення події необхідне виконання деяких умов, які називають передумовами. Виникнення подій може привести до виконання умов, які називають післяумовами. В мережі Петрі умови моделюються позиціями, події - переходами. Передумови події представляються вхідними позиціями відповідного переходу, післяумови - вихідними позиціями.

Технологічні процеси по своїй природі є динамічними. Тому побудовану модель технологічних процесів з використанням мереж Петрі будемо описувати за допомогою динамічних таблиць [1].

Мережі Петрі вдало відображають керування програмними моделями. Вони призначені для моделювання технологічних процесів і потоків інформації, але не для дійсного обчислення самих значень. В моделі дані потрібно зв'язати з певними елементами мереж. Тому перед створенням моделі паралельної програми, спершу необхідно сформувати примітиви, що відображають елементарні алгоритмічні конструкції*: умовний оператор, оператор вибору, передумовний оператор циклу, післяумовий оператор циклу* і т.д.

Конструкції алгоритмічної мови Delphi

Умовний оператор If призначений для вибору до виконання однієї з двох дій, залежно від деякої умови. Структура умовного оператора в мові Delphi має вигляд**: If** <*Умова*> T**hen** <*Твердження1*> E**lse** <*Твердження2*>.

На рис. 1 наведена блок-схема умовного оператора.

Конструкція, яка побудована за допомогою мережі Петрі (рис.1 а), відображає класичне подання структури умовного оператора.

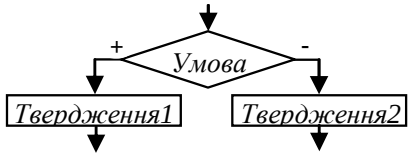

Рис. 1. Умовний оператор

Класична конструкція умовного оператора призводить до критичної ситуації – конфліктності спрацювання переходів t_1 і t_2.

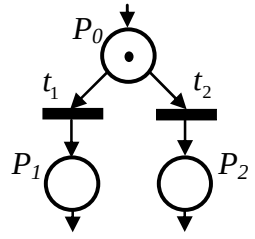

Рис.1 а. Класична конструкція If Then Else

Позиція P_0 моделює *умову* оператора If, а позиції P_1 *і* P_2 відповідно моделюють твердження1 і твердження2. Перехід t_1 моделює виконання умови - True (‘+’), а перехід t_2 моделює виконання умови - False (‘-’).

Побудована конструкція умовного оператора (рис.1б) функціонує коректно так, як в ній вирішено управління позицією P_0 . Для уникнення конфліктності між переходами t_1 і t_2 добавлено позицію $P_0^{'}$. Позиція $P_0^{'}$ моделює признак виконання умови. Якщо в позиції $P_0^{'}$ є фішка то умова виконується, інакше умова не виконується.

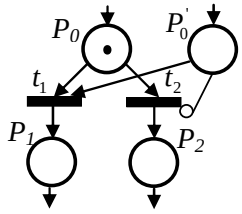

Рис.1 б. Коректна конструкція If Then Else

Складений умовний оператор в мові Delphi має вигляд**: If** <*Умова1*> T**hen** <*Твердження1*> E**lse If** <*Умова2*> T**hen** <*Твердження2*> E**lse** <*Твердження3*>.

Блок-схема складеного умовного оператора наведена на рис. 2.

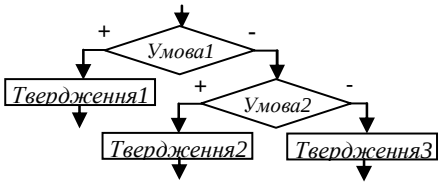

Рис. 2. Складений умовний оператор

На рис. 2 а за допомогою мережі Петрі побудована класична конструкція складеного умовного оператора.

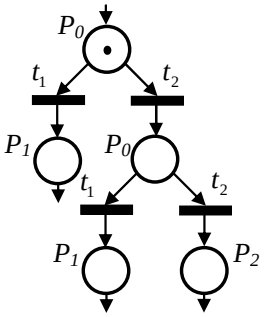

Рис. 2 а. Класична конструкція If Then Else If Then Else

В класичній конструкції складеного умовного оператора також виникають суперечності між спрацюванням переходів t_1 і t_2 для позиції P_0 і між спрацюванням переходів t_3 і t_4 для позиції P_2 .

Позиції P_0 і P_2 і моделюють відповідно *умову1* і *умову2* оператора If, а позиції P_1, P_3 і P_4 відповідно моделюють твердження1, твердження2 і твердження3. Перехід t_1 і t_3 відповідно моделюють виконання умови1 і виконання умови2 - True (‘+’), а переходи t_2 і t_4 відповідно моделюють виконання умови1 і виконання умови2 - False (‘-’).

В наведеній конструкції складеного умовного оператора на рис. 2б вибір *умови1* і *умови2* відбувається коректно. Завдяки добавленню позиції $P_0^{'}$ для позиції P_0 і позиції $P_2^{'}$ для позиції P_2 усунуто конфліктні ситуації між переходами t_1 і t_2 та між переходами t_3 і t_4 .

Позиція $P_0^{'}$ моделює признак виконання умови 1, а позиція $P_2^{'}$ моделює признак виконання умови 2.

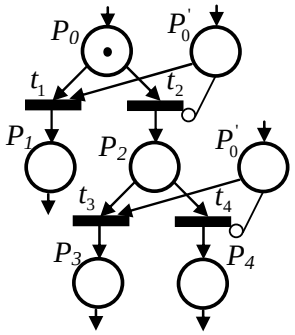

Рис. 2 б. Коректна конструкція If Then Else If Then Else

Умовний оператор Case дозволяє здійснювати вибір із будь-якої кількості варіантів. В цьому і полягає його головна відмінність від умовного оператора If. Умовний оператор Case в мові Delphi має вигляд**:**

Case *Vyb* **Of**
  Const1: Оператор 1;
  ⋮
  Const k-1: Оператор k-1;
End
  Else *Оператор k;*

Блок-схема умовного оператора вибору наведена на рис. 3.

На рис. 3 а побудована класична конструкція умовного оператора **Case** за допомогою мережі Петрі. Як в класичній конструкції умовного оператора If так і в класичній конструкції умовного оператора **Case** виникають суперечності між спрацюваннями

переходів: $t_1,\ldots,t_{k-1}$ і t_k.

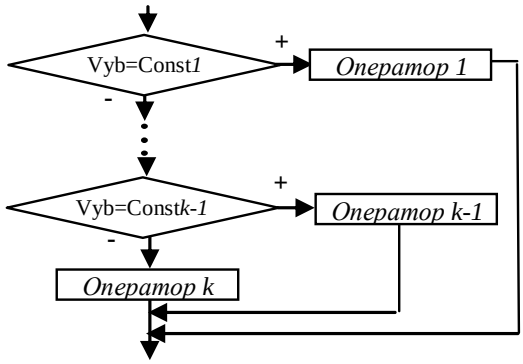

Рис.3. Оператор умовного вибору

Позиція P_0 моделює *умову вибору* оператора **Case,** а позиції P_1, …, P_{k-1} і P_k відповідно моделюють *оператор1, …, операторk-1* і *операторk*. Переходи: $t_1,\ldots,t_{k-1}$ і t_k відповідно моделюють виконання вибору, а . t_{k+1} моделює виконання наступних дій.

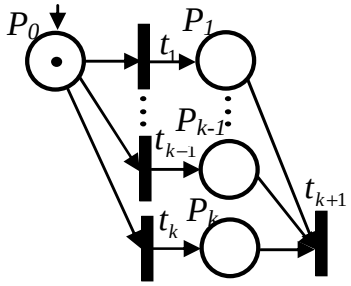

Рис. 3 а. Класична конструкція Case

В наведеній конструкції умовного оператора **Case** на рис. 3б вибір проходить коректно так, як в ній вирішено проблеми конфліктності спрацювання переходів: $t_1,\ldots,t_{k-1}$ і t_k завдяки добавленню позицій: $p_1^{'},\ldots,p_{k-1}^{'}$.

Позиція $p_1^{'}$ моделює признак вибору *оператора1,…*, позиція $p_{k-1}^{'}$ моделює признак вибору *операторak-1*. Іншими словами, якщо в позиції $p_1^{'}$ є фішка то вибирається оператор 1, …, якщо в позиції $p_{k-1}^{'}$ є фішка то вибирається оператор k-1 інакше оператор k.

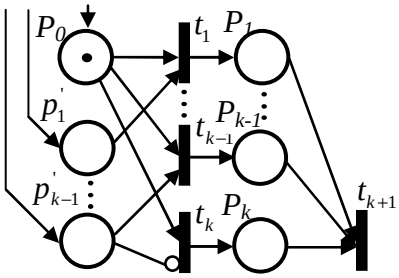

Рис.3 б. Коректна конструкція Case

Оператор циклу Repeat організовує виконання циклу, що складається з будь-якого числа операторів, з невідомим заздалегідь числом повторень. Вихід з циклу здійснюється при виконанні заданої умови. Такий цикл називають ще циклом з післяумовою. Структура цього оператора виглядає таким чином: **Repeat** *оператор1; оператор2 ; …; операторk;* **Untill** *Умова*.

Блок- схема циклу з післяумовою наведена на рис. 4, а на рис. 4 а побудовано конструкцію оператора циклу Repeat за допомогою мережі Петрі для фрагменту програми на мові Delphi з використанням циклу з післяумовою**:** *i:=1;* **Repeat** *оператор1; оператор2 ; …; операторk;* *i:=i+1;* **Until** *I > 5;. Тіло циклу.*

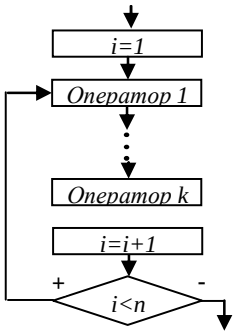

Рис. 4. Цикл з післяумовою

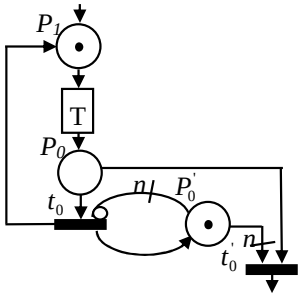

Рис. 4 а. Конструкція оператора циклу Repeat:
T - тіло циклу

Позиція P_1 моделює *оператор1*. Позиція P_0 моделює умову циклу. Позиція $P_0^{'}$ моделює признак

закінчення циклу. Тіло циклу – це підмережа Петрі, яка складається з будь-якої кількості позицій та переходів. Для даної конструкції тіло циклу складається з послідовності переходів та позицій: t_1, P_2 , …P_k , t_k. Тіло циклу назвемо макропереходом. Перехід t_0 моделює виконання повторень циклу.

Перехід $t_0^{'}$ моделює закінчення циклу. Інгібіторна дуга *n-го* порядку, де *n=5* контролює спрацювання переходу t_0, а дуга *n-го* порядку, де *n=5* контролює спрацювання переходу $t_0^{'}$.

Для дослідження роботи конструкції оператора циклу Repeat побудовано динамічну таблицю (див. табл. 1).

Таблиця 1

Стани перебування роботи конструкції оператора циклу Repeat

n/n	t_i	P_0	P_1	$P_0^{'}$
1	T	0	1	1
2	t_0	1	0	2
3	T	0	1	2
4	t_0	1	0	3
5	T	0	1	3
6	t_0	1	0	4
7	T	0	1	4
8	t_0	1	0	5
9	T	0	1	5
10	$t_0^{'}$	0	0	0

Оператор циклу While організовує виконання циклу з передумовою. Оператор циклу While записується так**: While** *Умова* **Do Begin** *оператор1; оператор2 ; …; операторk;* **End**.

На рис. 4 б наведена блок-схема циклу з передумовою.

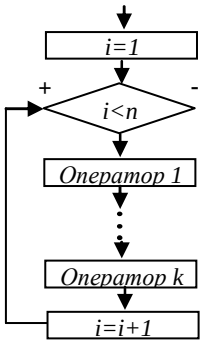

Рис. 4 б. Цикл з передумовою

Конструкція оператора циклу **While (**рис. 4 в) побудована для фрагменту програми на мові Delphi з використанням циклу з передумовою*: i:=1;* **While** *I <= 5* **Do Begin** *Тіло циклу; i:=i+1*; **End**;.

Як в конструкції оператора циклу Repeat так і в конструкції оператора циклу While позиції і переходи моделюють ці самі події.

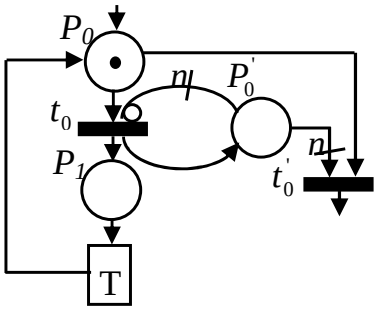

Рис. 4 в. Конструкція оператора циклу While

Роботу конструкції оператора циклу While відображено в динамічній таблиці (див. табл. 2).

Таблиця 2

Стани перебування роботи конструкції оператора циклу While

n/n	t_i	P_0	$P_0^{'}$	P_1
1	t_0	0	1	1
2	T	1	1	0
3	t_0	0	2	1
4	T	1	2	0
5	t_0	0	3	1
6	T	1	3	0
7	t_0	0	4	1
8	T	1	4	0
9	t_0	0	5	1
10	T	1	5	0
11	$t_0^{'}$	0	0	0

Оператор циклу For організовує виконання циклу, що складається з будь-якого числа операторів, з строго заданим числом повторень. Існує два варіанти оператора For.

На рис.5а побудована конструкція оператора циклу For для варіанту 1: **Fo**r *i:=1* **To** *n* **Do Begin** *Тіло циклу* **End**; а На рис.5б побудована конструкція оператора циклу For для варіанту 2: **Fo**r *i:=1* **DownTo** *n* **Do Begin** *Тіло циклу* **End.**

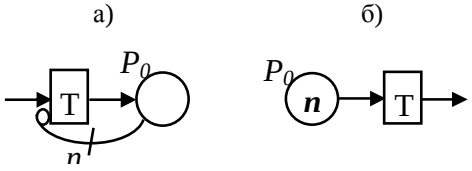

Рис.5. Конструкція оператора циклу For:
а – варіант 1, б – варіант 2

На перший погляд побудовані примітиви, що відображають алгоритмічні конструкції операторів мови Delphi*: If, Case, Repeat, While* і *For* виглядають громіздкими і незрозумілими в порівнянні з блок-схемами для даних операторів.

В подальшій роботі побудуємо модель підсистеми розформування поїздів на сортувальній станції (ПРСС), на її підставі розробимо програмне забезпечення (ПЗ) і доведемо, що побудова моделі за допомогою мереж Петрі для розробки ПЗ є доцільна.

Розробка програми на підставі моделі, яка побудована за допомогою мереж Петрі

За допомогою мережі Петрі на рис.6 побудована модель ПРСС [1, 2].

Позиція P_1 моделює дільниці, з яких прибувають поїзди в ПП (парк приймання) рівномірно. Позиція P_2 моделює кількість составів в ПП, які очікують технічне обслуговування (ТО). Позиція P_3 моделює ТО составів бригадою пункту ТО.

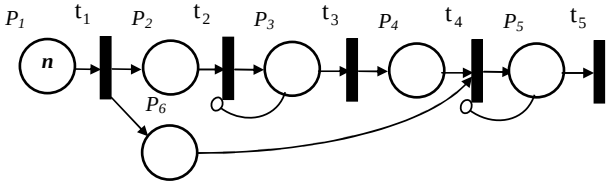

Рис. 6. Підсистема розформування поїздів на сортувальній станції

Позиція P_4 моделює кількість составів, в яких проведено ТО і які очікують розформування. Позиція P_5 моделює сортувальну гірку (СГ), яка розпускає состави. Позиція P_6 моделює лічильник составів в ПП. Перехід t_1 моделює прибуття поїзда в ПП. Перехід t_2 моделює поступання поїзда до ТО. Перехід t_3 моделює закінчення ТО. Перехід t_4 моделює поступання состава на СГ. Перехід t_5 моделює розформування состава на СГ.

Побудована модель описує функціонування паралельних технологічних процесів роботи ПРСС: прибуття поїздів в ПП, ТО составів в ПП і розпуск составів на СГ.

Нижче наведений фрагмент програми на Delphi:

```
Var {Опис змінних}
  SortSt: TSortSt;
Inp,Out,Inh:Array[1..10,1..10] Of Integer;
Tc,dt,i,ii,j,jj,k:Integer; s:String; //Tc- Умовний час;
t,f,n,n1,n2,m:Array[1..10] Of Integer;
Procedure TSortSt.FormCreate(Sender: TObject);
Begin {Початкові дані}
{Дуги: позиція – перехід}Inp[1,1]:=1;Inp[2,2]:=1;
Inp[3,3]:=1;Inp[4,4]:=1;Inp[5,5]:=1;Inp[6,4]:=1;
{Дуги: перехід - позиція } Out[1,2]:=1;Out[2,3]:=1;
Out[3,4]:=1;Out[4,5]:=1;Out[1,6]:=1;
{Інгібіторні дуги} Inh[3,2]:=-1;Inh[5,4]:=-1;
{Початкове маркування} m[1]:=3;m[2]:=0;m[3]:=0;
m[4]:=0;m[5]:=0;m[6]:=0;
{Час затримування} f[1]:=6;f[2]:=3;f[3]:=5;
f[4]:=2;f[5]:=4;
End;
Procedure TSortSt.PRClick(Sender: TObject);
Begin {Основна програма: ПРСС}
Memo1.Lines.Add('|N |У.о.ч .|tk |P1|P2|P3|P4|P5|P6|Заув.
|');
 For Tc:=0 To 80 Do
Begin{Визначення зпрацювання переходів}
For j:=1 To 5 Do Begin  n1[j]:=1;    n2[j]:=1;
For i:=1 To 6 Do
Begin If Inp[i,j]>0 Then If m[i]-Inp[i,j]>=0 Then
n1[j]:=n1[j]*1 Else n1[j]:=n1[j]*0;
 If Out[j,i]>0 Then If Inh[i,j]=0 Then n2[j]:=n2[j]*1 Else
If (Inh[i,j]=-1)And(m[i]>0) Then n2[j]:=n2[j]*0;
End;n[j]:=n1[j]*n2[j];End;
For j:=1 To 5 Do{Зпрацювання переходів}
Begin {відлік часу}
If  (n[j]=1)And(f[j]-t[j]<=0)  Then  Begin  k:=k+1;
dt:=Abs(f[j]-t[j]);
If dt>0 Then s:=IntToStr(dt)+' у.о.ч.' Else s:='  ';
t[j]:=0;
For i:=1 To 6 Do
m[i]:=m[i]-Inp[i,j]+Out[j,i];
If (n[5]=1)and(m[5]=0) Then s:=' розф.';
Memo1.Lines.Add('| '+IntToStr(k)+' |  '+IntToStr(Tc)+'  |
'+IntToStr(j)+' | '+IntToStr(m[1])+' | '+IntToStr(m[2])+' |
'+IntToStr(m[3])+'  |  '+IntToStr(m[4])+'  |
'+IntToStr(m[5])+' | '+IntToStr(m[6])+' | '+s+'|'); End;
{Зпрацювання переходу t j+1 після t j}
If  ( n[j]=1)And(f[j+1]=0)And(m[j+1]>0)And  (m[j+2]=0)
Then n[j+1]:=1;End;
{Зпрацювання переходу t j після t j+1}
For j:=1 To 5 Do  For i:=1 To 5 do
Begin  If  (Inh[i,j]=-1)And(n1[j]=1)And(m[i]=0)And
(n[i]=1) And(f[1]<=f[i])Then Begin n[j]:=1;
If(  n[j]=1)  And(f[j]-t[j]<=0)  Then  Begin  k:=k+1;
dt=Abs(f[j]-t[j]);
```

```
If dt>0 Then s:=IntToStr(dt)+'  у.о.ч.' Else s:='  ';
t[j]:=0;
For ii:=1 To 6 Do  m[ii]:=m[ii]-Inp[ii,j]+Out[j,ii];
Memo1.Lines.Add('| '+IntToStr(k)+' | '+IntToStr(Tc)+' |
'+IntToStr(j)+' | '+IntToStr(m[1])+' | '+IntToStr(m[2])+' |
'+IntToStr(m[3])+' | '+IntToStr(m[4])+' |
'+IntToStr(m[5])+' | '+IntToStr(m[6])+' | '+s+'|'); End;
End;
End; {відлік часу}
For j:=1 To 5 Do For i:=1 To 5 Do
If (Inp[i,j]>0)And(m[j]>0)Then t[j]:=t[j]+1;
End;
End;
```

Динамічна таблиця (див. табл. 3), яка отримана в результаті виконання програми, в динаміці відображає функціонування паралельних технологічних процесів роботи ПРСС.

Таблиця 3

Стани перебування роботи ПРСС

N	У.о.ч.	tk	P1	P2	P3	P4	P5	P6	Заув.
1	6	1	2	1	0	0	0	1	
2	9	2	2	0	1	0	0	1	
3	12	1	1	1	1	0	0	2	
4	14	3	1	1	0	1	0	2	
5	15	2	1	0	1	1	0	2	
6	16	4	1	0	1	0	1	1	
7	18	1	0	1	1	0	1	2	
8	20	3	0	1	0	1	1	2	
9	20	5	0	1	0	1	0	2	розф.
10	21	2	0	0	1	1	0	2	
11	22	4	0	0	1	0	1	1	
12	26	3	0	0	0	1	1	1	
13	26	5	0	0	0	1	0	1	розф.
14	28	4	0	0	0	0	1	0	
15	32	5	0	0	0	0	0	0	розф.

Як ми бачимо, побудована модель для розробки ПЗ компактна, має привабливий вигляд в порівнянні з блок-схемою і візуально відображає топологію технологічних процесів ПРСС. Переклад алгоритму з інтерпретаційної мови мереж Петрі, який описує технологічні процеси моделі на мову програмування Delphi не вимагає ніяких труднощів, а властивості мереж Петрі дозволяють проводити перевірку побудованої моделі на неконфліктність (коректне функціонування моделі), що дозволяє автоматизувати непростий етап розробки паралельних програм – тестування і відладку ПЗ.

Таким чином, розробка ПЗ на підставі моделі, яка побудована за допомогою мереж Петрі, є доцільна.

Висновки

1. За допомогою мереж Петрі побудовані примітиви, що відображають алгоритмічні конструкції операторів мови Delphi*: If, Case*, *Repeat, While* і *For*.

2. На підставі побудованої моделі ПРСС розроблено програмне забезпечення функціонування ПРСС.

3. Показано переваги моделі, яка побудована за допомогою мереж Петрі над побудовою блок-схем для розробки програмного забезпечення.

4. Цінність розробки програмного забезпечення на підставі моделі, яка побудована за допомогою мереж Петрі полягає в тому, що побудована модель дає можливість тестувати і відлагоджувати програмне забезпечення.

Література

1. Селецький В.С., Федак Я.А. Про пристрої обслуговування заявок //Іформаційно – керуючі системи на залізничному транспорті. - 2001. - № 5. – С. 31 – 34.
2. Луханін М.І., Селецький В.С. Удосконалена модель підсистеми розформування поїздів на сортувальній станції //Іформаційно – керуючі системи на залізничному транспорті. - 2000. - № 6. – С. 71 – 74.
3. Мурашко А. Г. Первое знакомство с сетями Петри: Учебное пособие. – К.:УМК ВО, 1988. – 71с.
4. Глушков С.В., Клевцов А.Л. Программирование в среде Delphi 7.0. - .: Х.: Фолио, 2003. – 520с.

Селецкий В.С. Сети Петри и язык программирования Delphi. Разработано автоматизированное программное обеспечение анализа функционирования параллельных технологических процессов ПРСС на основании модели, которая построена с помощью математического аппарата сетей Петри.

Ключевые слова: язык Delphi, условный оператор, оператор цикла, блок-схема, сеть Петри, конструкция, модель.

Selezkyj V.S. Petri nets and programming language Delphi. The automated software for the analysis of parallel technological PRSS process functioning has been developed on the basis of the model, built by means of matematical apparatus of Petri nets.

Key words: language Delphi, conditional operator, cycle statement, flow chart, Petri net, design, model.

Рецензент доктор фіз.-мат. наук, професор М.М. Зарічний (Львівський національний університет ім. І.Франка)

Поступила 28.11.2013 р.