Овсяк О. Моделі інформаційних технологій видалення, формування XML-опису і деселекції операції секвентування / Овсяк О. // Вісник ТНТУ. — 2011. — Том 17. — № 2. — С.150-157. — (приладобудування та інформаційно-вимірювальні технології).

УДК 004.4’232

О. Овсяк, канд. техн. наук

Львівська філія Київського національного університету культури і мистецтв
Українська академія друкарства

МОДЕЛІ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ВИДАЛЕННЯ, ФОРМУВАННЯ *XML*-ОПИСУ І ДЕСЕЛЕКЦІЇ ОПЕРАЦІЇ СЕКВЕНТУВАННЯ

***Резюме**. Засобами алгебри алгоритмів описано синтезовані математичні моделі інформаційних технологій видалення, створення xml-формату і зняття виділення операції секвентування. Модель видалення базується на обчисленні розмірів прямокутної області операції секвентування. У моделі xml-опису використовуються стандартні підсистеми створення елементів і атрибутів. Мовою об’єктного програмування C#, платформи Microsoft Visual Studio.NET програмно реалізовано й апробовано моделі формування xml-опису і функційного унітерму деселекції операції секвентування.*

***Ключові слова:** модель, елімінування, xml-формат, операція секвентування, деселекція.*

A. Ovsyak

MODEL ELIMINATION, *xml* - DESCRIPTION AND DESELECTION THE SEQUENCING OPERATION OF THE INFORMATION TECHNOLOGY

***The summary.** By means of algebra algorithms described mathematical models synthesized elimination, creating xml-format and deselect operations sequencing of information technology. The model is based on removal calculating the size of rectangular area operations sequencing. The model xml-standard descriptions used to create components and subsystems attributes. Object programming language C #, platform Microsoft Visual Studio.NET, software implemented and tested model of xml-description and of functional uniterm deselect operation sequencing.*

***Key words:** model, elimination, xml-format, operation sequenction, deselection.*

Аналіз останніх досліджень і постановка проблеми. Найбільш розповсюдженими методами опису алгоритмів є вербальний і блок-схемний. Крім них відомі методи Поста [1], Т’юрінґа [2], Ахо-Ульмана-Хопкрофта [3], Шенґаґе [4], рекурсивних функцій [5], Маркова [6], Колмогорова [7], Криницького [8]. Очевидно, що вербальний і блок-схемний методи забезпечують текстовий і текстово-графічний опис алгоритмів відповідно. Також відомо [9], що існуючі методи [1–8] не забезпечують математичного опису алгоритмів.

У дослідженні [10], яке набуло подальшого розвитку в роботі [11], аксіоматичним методом введено систему операцій, використання яких забезпечує отримання опису алгоритмів у вигляді математичних формул, над якими можуть бути виконані тотожні алгебраїчні перетворення. Система позначень алгебри алгоритмів має специфічні знаки [10–13], яких немає серед типових математичних позначень. Крім того, геометричні розміри таких знаків операцій, як секвентування, елімінування, паралелення і реверсування залежать від геометричних розмірів унітермів, над якими вони виконуються. Знаки операцій секвентування, елімінування, паралелення і циклічних секвентування, елімінування і паралелення мають складну геометричну конфігурацію.

Є можливість створювати складну систему знаків алгебри алгоритмів засобами найпоширеніших універсальних комп'ютерних систем, таких, як Word, Coral Draw, Page Maker, In Design та ін. Однак використання таких універсальних систем для набору й редагування формул алгоритмів потребує великих затрат праці та комп'ютерного часу і тому є малоефективним [12, 13].

Для підвищення рівня автоматизації та зменшення затрат часу для набору й редагування формул алгоритмів створено спеціалізовані редактори МОДАЛ [12] і АБСТРАКТАЛ [13]. Однак моделі цих систем не забезпечують можливості автоматичної оптимізації формул алгоритмів. Виконання автоматичних перетворень формул алгоритмів передбачено моделлю спеціалізованої комп'ютерної системи [14]. Але для забезпечення можливості виконання оптимізації формул алгоритмів перш за все необхідно створити й реалізувати моделі видалення, формування *XML*-опису та зняття (деселекції) вибору операції секвентування.

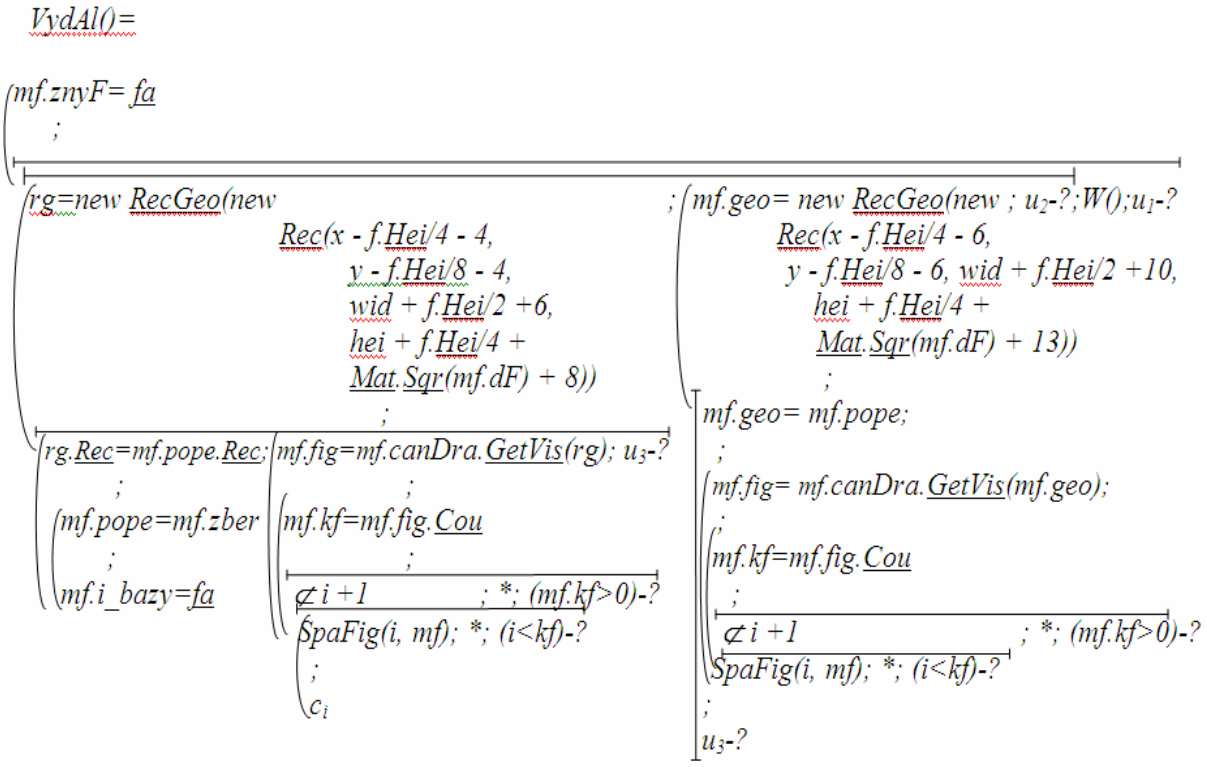
Метою роботи є автоматизація процесів видалення, створення *XML*-опису й деселекції операції секвентування.

Завдання дослідження. Створити й описати засобами алгебри алгоритмів моделі інформаційних технологій комп'ютерного видалення, формування *XML*-опису й деселекції операції секвентування, які запрограмувати мовою C# платформи Microsoft Visual Studio.NET [15, 16].

1. Модель інформаційної технології видалення

1.1. Модель інформаційної технології видалення секвентування за відсутності режимів заміни

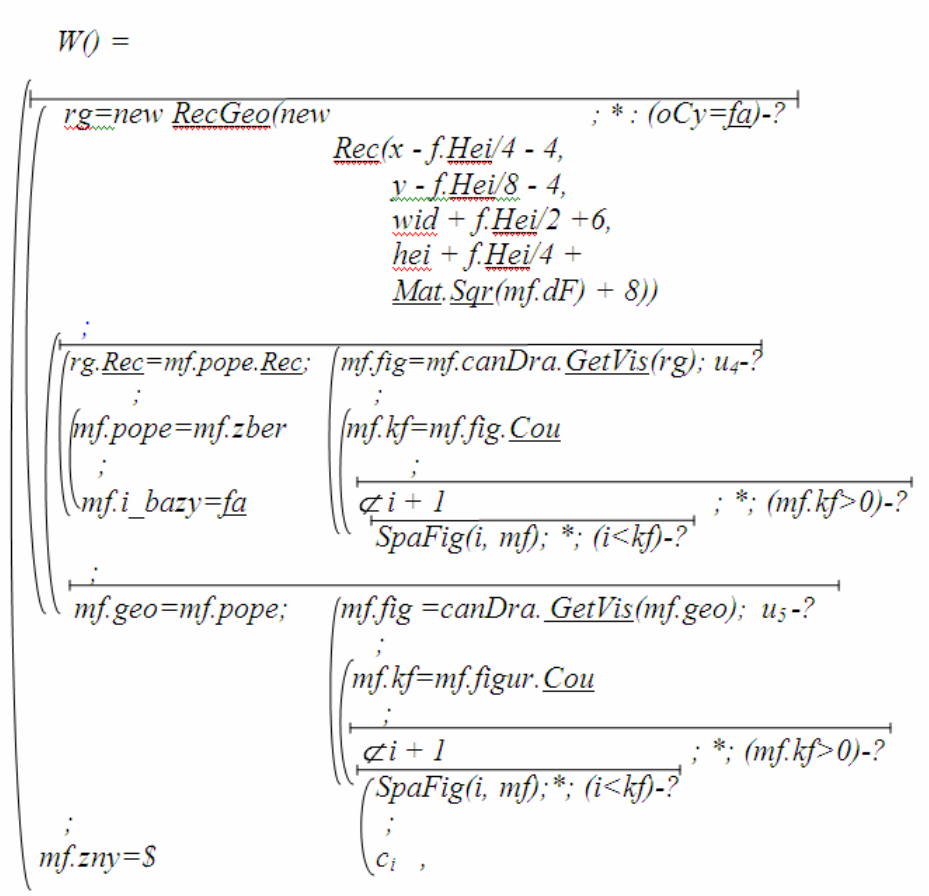
У загальному випадку видалення графічних об'єктів описується такою формулою:



де *mf.znyF= fa* – приписування змінній *znyF* через складну змінну *mf* підсистеми *MaFor* стандартного значення *fa*; *M()* – функційні унітерми рисування знака операції секвентування; *(mf.znyF=tr) - ?* – порівняння значення змінної з типовим значенням *tr*; u_1 – складна умова, яка описується виразом *(mf.zny≠"R")&(mf.zny≠"F")&(mf.zny≠"S")*, у якому *(mf.zny≠"R")* – порівняння значення змінної *zny* на нерівність *"R"* (ідентифікатор режиму заміни операції секвентування); *(mf.zny≠"F")* – відсутність режиму заміни операції секвентування залишенням її першого унітерму; *(mf.zny≠"S")* – відсутність режиму заміни операції секвентування залишенням її другого унітерму; *W()* – функційний унітерм видалення алгоритму за наявності одного із режимів заміщення операції секвентування; u_2 – складний умовний унітерм, який описується виразом *(mf.zny≠"dA")&(mf.zny≠"dF")&(mf.zny≠"dS")&(mf.zny≠"dC")v(mf.zny=$)*, у якому *(mf.zny≠"dA")* – відсутність режиму видалення операціїї секвентування; *(mf.zny≠"dF")* – залишення її першого унітерму; *(mf.zny≠"dS")* – залишення її другого унітерму; *(mf.zny≠"dC")* – залишення секвентування як вкладеної операції; *(mf.zny=$)* – відсутність задавання режимів видалення; *rg=new RecGeo(new Rec(x - f.Hei/4 – 4, y - f.Hei/8 - 4, wid + f.Hei/2 +6, hei + f.Hei/4 + Mat.Sqr(mf.dF) + 8))* – у змінній *rg* збереження координат прямокутника (*x - f.Hei/4 – 4, y - f.Hei/8 - 4, wid + f.Hei/2 +6, hei + f.Hei/4 + Mat.Sqr(mf.dF) + 8*), обчислених функційним унітермом *Rec()* стандартної підсистеми *RecGeo*, яка реалізується класом RectangleGeometry [15, 16], а *Rec()* – процедурою Rect( ) [15, 16]; *Sqr()* – процедурою Sqrt() системного класу Math [15, 16]; *wid* , *hei* – текучі значення сумарних довжини і висоти унітермів з розділювачем; u_3 – складний умовний унітерм *(mf.pope≠$) &(rg.Rec.Hei <mf.pope.Rec.Hei)&(rg.Rec.Wid<mf.pope.Rec.Wid)*, у якому *(mf.pope≠$)* - перевірка наявності попередньої (*pope*) рамки вибору операції; *(rg.Rec.Hei<mf.pope.Rec.Hei)* – порівняння висоти (*rg.Rec.Hei*) текучої рамки вибору секвентування з висотою (*mf.pope.Rec.He*) попередньої рамки; *(rg.Rec.Wid<mf.pope.Rec.Wid)* – порівняння довжини (*rg.Rec.Wid*) текучої рамки вибору секвентування з довжиною (*mf.pope.Rec.Wid*) попередньої рамки; *rg.Rec=mf.pope.Rec* – переписування розмірів попередньої рамки вибору у змінну зберігаючу розміри текучої рамки вибору, яке виконується унітермом *Rec*, який реалізується властивістю Rect [15, 16]; *mf.pope=mf.zber* – переписування розмірів збереженої рамки вибору у змінну з попередніми розмірами; *mf.i_bazy=fa* – запис в ідентифікатор (*i_bazy*) вибору формули алгоритму з бази алгоритмів типового значення *fa*; *mf.fig=mf.canDra.GetVis(rg)* – графічних обєктів, які знаходяться на *canDra*, переписування у змінну *fig* стандартним функційним унітермом *GetVis()*, який реалізується відомою процедурою GetVisuals()[15, 16]; *mf.kf=mf.fig.Cou* – змінній *kf* приписується кількість графічних фігур, яка обчислена стандартним унітермом *Cou*, який реалізується відомою властивістю Count [15, 16]; *(mf.kf>0)-?* – порівняння кількості фігур з нулем; $\alpha i + 1$ – цикл за змінною кількості фігур *i* та задавання кроку зміни значень змінної; *SpaFig(i, mf)* – фукційний унітерм усунення графічних обєктів; c_i*+1* – ознака повернення у цикл за змінною *i*; *mf.geo= mf.pope* – збереження у комірці пам'яті *geo* попередньої рамки вибору операції секвентування *pope*; *mf.fig= mf.canDra.GetVis(mf.geo)* – графічні фігури *canDra*, вміщені у рамку *geo* записуються у *fig* після їхнього вибору функційним унітермом *GetVis()*.

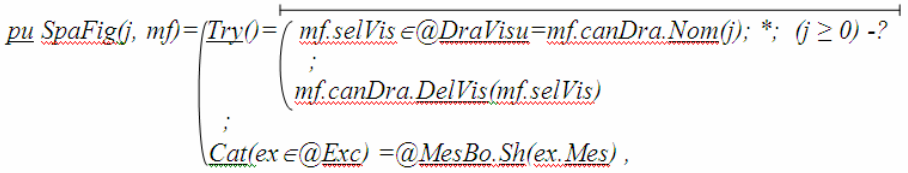
1.2. Видалення за наявності режимів заміни і відсутності циклічних операцій

Формула має такий вигляд:



де *oCy* – змінна зберігача ідентифікатора циклічних операцій; *rg=new RecGeo(new Rec(x - f.Hei/4 – 4, y - f.Hei/8 - 4, wid + f.Hei/2 +6, hei + f.Hei/4 + Mat.Sqr(mf.dF) + 8))* – запамятовуваня у *rg* рамки вибору; u_4 – складний унітерм *(mf.pope≠$) &(mf.iBaz=tr)&(rg.Rec.Hei<mf.pope.Rec.Hei)&(rg.Rec.Wid < mf.pope.Rec.Wid)* умови, у якому *(mf.iBaz=tr)* – порівняння значення ідентифікатора зчитування з бази алгоритмів *iBaz* зі стандартним значенням *tr* унітерму; *rg.Rec=mf.pope.Rec* – переписування в *rg* передньої рамки вибору *pope*; *mf.pope=mf.zber* – запис у *pope* значення збереженої рамки вибору *zber*; *mf.i_bazy=fa* – запис у *i_bazy* типового значення *fa*; *mf.fig=mf.canDra.GetVis(rg)* – запис у *fig* кількості графічних обєктів, які є у рамці *rg* розміщеній на *canDra*; *mf.kf=mf.fig.Cou* – обчислення кількості графічних об'єктів *fig*; *∝ i + 1* – цикл за змінною з модифікатором її зміни (*+1*) після повернення у цикл; *SpaFig(i, mf)* – функційний унітерм усунення графічних об'єктів; u_5 – складний умовний унітерм *(mf.pope≠$) &(mf.geo.Rec.Hei <mf.pope.Rec.Hei) &(mf.geo.Rec.Wid <mf.pope.Rec.Wid),* у якому перевіряється наявність рамки попереднього вибору і порівняння її розмірів з розмірами рамки, записаної у *geo*.

Функційний унітерм усунення графічних об'єктів описується формулою

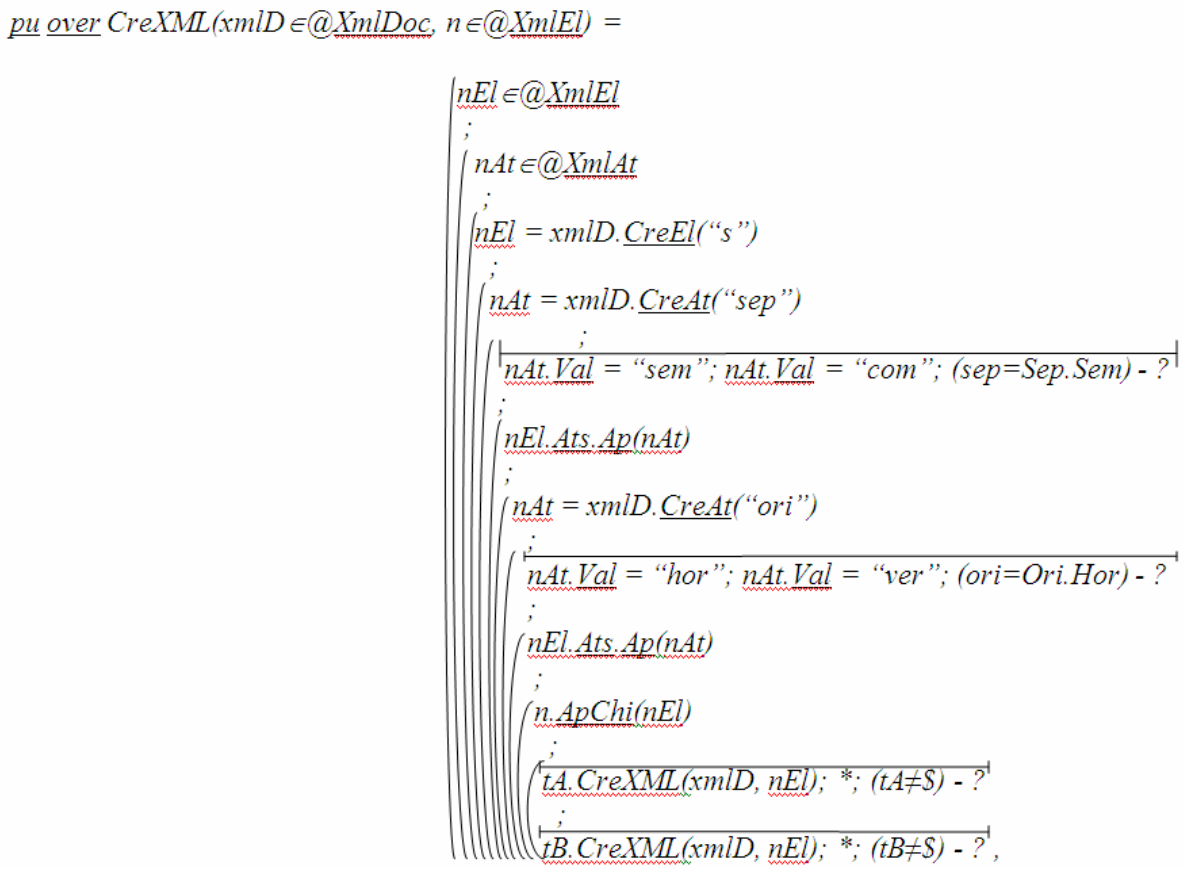


де *Try()* – функційний унітерм перевірки коректності формули, яка йому дорівнює, який описується відомою процедурою try [15, 16]; *mf.selVis ∈@DraVisu* – змінна

mf.selVis типу стандартної підсистеми *DraVisu*, яка реалізується відомим класом DrawingVisual [15, 16]; графічний об'єкт з номером *j* вибирається з *canDra* типовим функційним унітермом *Nom(i)*, який реалізується відомою процедурою Nomer()[15, 16]; *mf.selVis∈@DraVisu=mf.canDra.Nom(j)* – приписування змінній графічного об'єкта з номером *j*; *mf.canDra.DelVis(mf.selVis)* – розміщений на *canDra* графічний об'єкт змінної *selVis* усувається стандартним функційним унітермом *DelVis()*, який реалізується відомою процедурою DeleteVisual()[15, 16]; *ex∈@Exc* – змінна стандартної підсистеми винятків *Exc*, яка реалізується відомим класом Exception [15, 16]; *Cat()* – стандартний функційний унітерм, який реалізується процедурою catch ()[15, 16]; *MesBo* – підсистема створення графічного вікна, яка реалізується відомим класом MessageBox [15, 16]; *Sh()* – функційний унітерм висвітлювання повідомлення, що реалізується процедурою Show()[15, 16]; *Mes* – унітерм задавання повідомлення про виняток, який реалізується властивістю Message [15, 16]; *Cat(ex∈@Exc) =@MesBo.Sh(ex.Mes)* – опис виведення повідомлення про виняток.

2. Модель інформаційної технології *xml*-опису операції секвентування

Xml-опис операції секвентування має охоплювати ідентифікатори операції секвентування (*s*) і розділювача (*sep*) унітермів, сам розділювач (*sem* чи *com*), ідентифікатор орієнтації (*ori*) і значення орієнтації (*hor* чи *ver*), й самі унітерми. Формула яка описує створення подібного опису з такими даними, має такий вигляд:

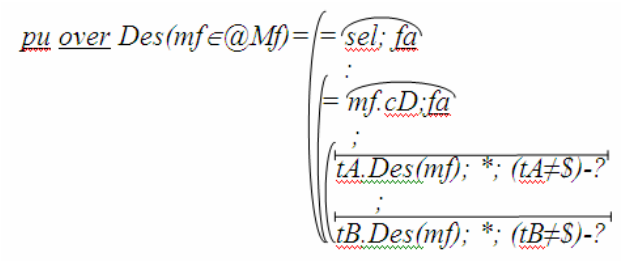


де *xmlD∈@XmlDoc* – вхідна змінна *xmlD* типу підсистеми *XmlDoc*, яка реалізується відомим класом XmlDocument [15, 16]; *n∈@XmlEl* – вхідна змінна *n* типу підсистеми *XmlEl*, яка реалізується відомим класом XmlElement [15, 16]; *nEl∈@XmlEl* – створення змінної *nEl* типу *XmlEl*; *nAt∈@XmlAt* – створення змінної *nAt* типу *XmlAt*, яка реалізується відомим класом XmlAttribute [15, 16]; *nEl = xmlD.CreEl("s")* – змінній *nEl* елемент *"s"* приписується типовим функційним унітермом *CreEl()*, який реалізується

відомою процедурою CreateElement() [15, 16]; *nAt = xmlD.CreAt("sep")* – змінній *nAt* атрибут *"sep"* приписується типовим функційним унітермом *CreAt()*, який реалізується відомою процедурою CreateAttribute() [15, 16]; *(sep=Sep.Sem)-?* – перевірка значення розділювача на наявність значення *Sem*, *nAt.Val = "sem"* та *nAt.Val = "com"* – приписування атрибуту значень *"sem"* та *"com"*; *nEl.Ats.Ap(nAt)* – змінна *nEl* буде доповнена *Ap()* новими атрибутами *Ats*; *nAt = xmlD.CreAt("ori")* – до змінної *xmlD* введення *CreAt()* атрибута *"ori"* і приписування значення змінної *xmlD* змінній *nAt*; *(ori=Ori.Hor)-?* – перевірка наявності задання горизонтальної орієнтації (*Ori.Hor*) операції секвентування; *nAt.Val = "hor"* та *nAt.Val = "ver"* – приписування атрибуту *nAt* значень *"hor"* та *"ver"*; *n.ApChi(nEl)* – ввести вкладений елемент у *nEl*; *(tA≠$)-?* – перевірка наявності першого (*tA*) унітерму секвенції; *tA.CreXML(xmlD, nEl)* – формування *xml*-елемента з першого унітерму секвенції; *(tB≠$)-?* – перевірка наявності другого (*tB*) унітерму секвенції; *tB.CreXML(xmlD, nEl)* – формування *xml*-елемента з другого унітерму секвенції.

3. Модель інформаційної технології деселекції

Модель функційного унітерму деселекції описується формулою



де *over* – ідентифікатор прикриття даним функційним унітермом такого самого функційного унітерму базової підсистеми, яку наслідує дана підсистема (ідентифікатор реалізується модифікатором override [15, 16]); = *sel; fa* – приписування змінній селективного вибору (*sel*) значення *fa*, яке відповідає стандартному значенню false [15, 16]; = *mf.cD;fa* – секвенційний унітерм приписування змінній *cD* підсистеми *Mf* значення *fa*.

4. Програмна реалізація моделі формування xml-опису операції секвентування

Написаний **C#** [15, 16] мовою об'єктного програмування код моделі має такий вигляд:

```
public override void CreateXML(XmlDocument xmlDoc, XmlElement node)
 {
 XmlElement newElement;
 XmlAttribute newAttribute;

 newElement = xmlDoc.CreateElement("sequence");

 newAttribute = xmlDoc.CreateAttribute("separator");

 if (separator == Separator.Semicolon)
 newAttribute.Value = "semicolon";
 else
 newAttribute.Value = "comma";
```

```
newElement.Attributes.Append(newAttribute);
newAttribute = xmlDoc.CreateAttribute("orientation");
if (orientation == Orientation.Horizontal)
newAttribute.Value = "horizontal";
else
newAttribute.Value = "vertical";
newElement.Attributes.Append(newAttribute);
node.AppendChild(newElement);
if (termA != null)
termA.CreateXML(xmlDoc, newElement);
if (termB != null)
termB.CreateXML(xmlDoc, newElement);
}
```

5. Програмна реалізація унітерму деселекції

Написаний мовою **C#** код є таким:

```
public override void Deselect(MainForm mf)
{
selected = false;
mf.cok_Na_canwasDraw = false;
if (termA != null)
termA.Deselect(mf);
if (termB != null)
termB.Deselect(mf);
    }
```

Результати дослідження. Засобами розширеної алгебри алгоритмів побудовано математичні моделі інформаційних технологій елімінування, формування *xml*-опису і деселекції операції секвентування формул алгоритмів. Створені математичні моделі програмно реалізовано й апробовано.

Висновки. Програмна реалізація математичних моделей забезпечує виконання процесів видалення, формування *XML*-опису і зняття вибору операції секвентування. Створені моделі і їхня програмна реалізація можуть бути використані для реалізації процесів автоматичної оптимізації формул алгебри алгоритмів.

Література

1. Post, E.L. Finite Combinatory Processes - Formulation 1 / E.L. Post // *Journal of Symbolic Logic*. – 1936. – № 1. – P. 103 – 105. (Reprinted in The Undecidable, pp. 289ff).
2. Turing, A.M. On computable numbers, with an application to the Entscheidungsproblem / A.M.Turing // *Proceedings of London Mathematical Society*. – 1936 – 1937. – Series 2, vol. 42. – P. 230–265. (Correction, ibidem, vol. 43, pp. 544–546. Reprinted in [13 Davis M., pp. 155–222] and available online at http://www.abelard.org/turpap2/tp2-ie.asp).
3. Aho, A.V. The design and analysis of computer algorithms / A.V. Aho, J.E. Hopcroft, J.D. Ullman // Addison-Wesley Publishing Company. – 1974.
4. Schönhage, A. Universelle Turing Speicherung. In J. Dörr and G. Hotz, Editors / A.Schönhage // *Automatentheorie und Formale Sprachen, Bibliogr. Institut, Mannheim*. – 1970. – P. 369–383.
5. Church, A. An unsolvable problem of elementary number theory / A. Church // *American Journal of Mathematics*. – 1936. – Vol. 58. – P. 345–363.
6. Марков, А.А. Теория алгоритмов [Текст] / А.А. Марков // Труды МИАН. – 1951. – Т. № 38. – С. 176–189.

7. Колмогоров А.Н. О понятии алгоритма [Текст] / А.Н. Колмогоров // УМН. – 1953. – Т. № 8, Вып. 4 (56).– С. 175–176.
8. Криницкий А.А. Алгоритмы вокруг нас [Текст] / А.А. Криницкий. – М.: Наука, 1984. – 224 с.
9. Детловс, В.К. Нормальные алгоритмы и рекурсивные функции [Текст] / В.К. Детловс // Докл. АН СССР. – 1953. – Т. 90, № 4. – С.723–725.
10. Овсяк, В.К. Засоби еквівалентних перетворень алгоритмів інформаційно-технологічних систем [Текст] / В.К. Овсяк // Доповіді Національної академії наук України, 1996. – № 9. – С. 83–89.
11. Owsiak, A. Rozszerzenie algebry algorytmów / W. Owsiak, A. Owsiak // Pomiary, automatyka, kontrola. 2010. – № 2. – S. 184–188.
12. Бритковський, В.М. Моделювання редактора формул секвенційних алгоритмів [Текст]: автореф. дис. ... канд. тех. наук: 01.05.02 "Математичне моделювання та обчислювальні методи" / В.М. Бритковський. – Львів, 2003. – 18 с.
13. Василюк, А.С. Підвищення ефективності математичного і програмного забезпечення редактора формул алгоритмів [Текст]: автореф. дис. ... канд. тех. наук: 01.05.02 "Математичне та програмне забезпечення обчислювальних машин і систем" / А.С. Василюк. – Львів, 2008. – 20 с.
14. Овсяк, О.В. Модель абстрактної підсистеми комп'ютерної інформаційної системи генерування коду [Текст] / О.В. Овсяк // Комп'ютерні науки та інформаційні технології. Вісник національного університету "Львівська політехніка". – 2010. – С.127–136.
15. Petzold, C. Programowanie Windows w języku C# / C. Petzold. – Warszawa: Rm, 2002. – 1161 s.
16. Мак-Дональд, М. WPF: Windows Prezentation Foundation в NET 3.5 с примерами на C# 2008. Для профессионалов / М. Мак-Дональд; [пер. с англ. Я.П. Волковой, Д.Я. Иваненко, Н.А. Мухана]. – Москва, Санкт-Петербург, Киев: И.Д. „Вильямс", 2008. – 928 с.

Отримано 01.02.2011