УДК 004

О. Овсяк
Українська академія друкарства

ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ ОПЕРАЦІЇ ПАРАЛЕЛЕННЯ

Засобами алгебри алгоритмів описана створена математичні моделі інформаційних технологій комп'ютерного формування та утворення xml – формату операції паралелення.

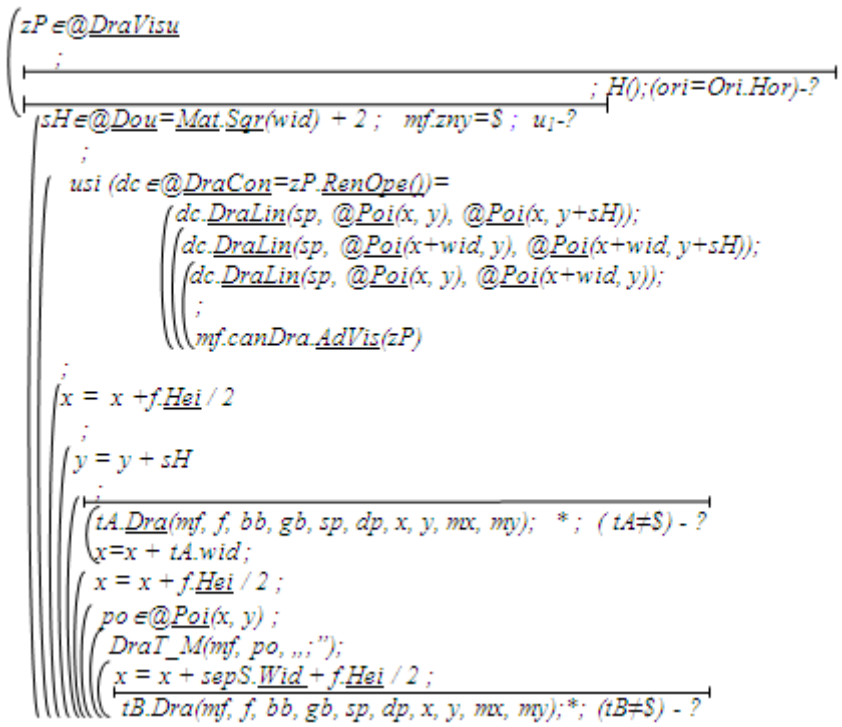
Algebra algorithms described by means of a mathematics models of computer information technologies and formation xml - format paralleling operations.

Вступ і формулювання задачі

У розширеній і класичній алгебрі алгоритмів [1, 2, 3] для опису паралельних інформаційних процесів введено операцію паралелення. Знак операції подібний до відкритої прямокутної дужки. Має він довжину і ширину. Виконується операція над двома унітермами, якими є знаки над якими виконуються операції алгебри алгоритмів. Оскільки знак операції записується над унітермами, то його розміри залежать від розмірів унітермів. У зв'язку з цим процеси набору і редагування операцій паралелення, які виконуються типовими комп'ютерними редакторами, наприклад, Word, потребують постійного виконання масштабування знаків операції паралелення. Така постійна зміна розмірів супроводжується значними затратами комп'ютерного і робочого часу. З метою зменшення затрат створено спеціалізовані редактори Модал [4] і Абстрактал [5], які призначені для набору і редагування формул алгоритмів. Ці редактори мають різні та специфічні формати даних текстового опису формул алгоритмів. Але ними не використовується мова опису даних, якою є XML [6] мова, що, як наслідок, суттєво ускладнює введення змін у структури даних і утруднює алгоритми та програмний код опрацювання структур даних. У зв'язку з цим у статті розглянуто моделі інформаційних технологій автоматичного формування виразу і xml – опису операції елімінування.

Узагальнена модель інформаційної технології формування операції паралелення

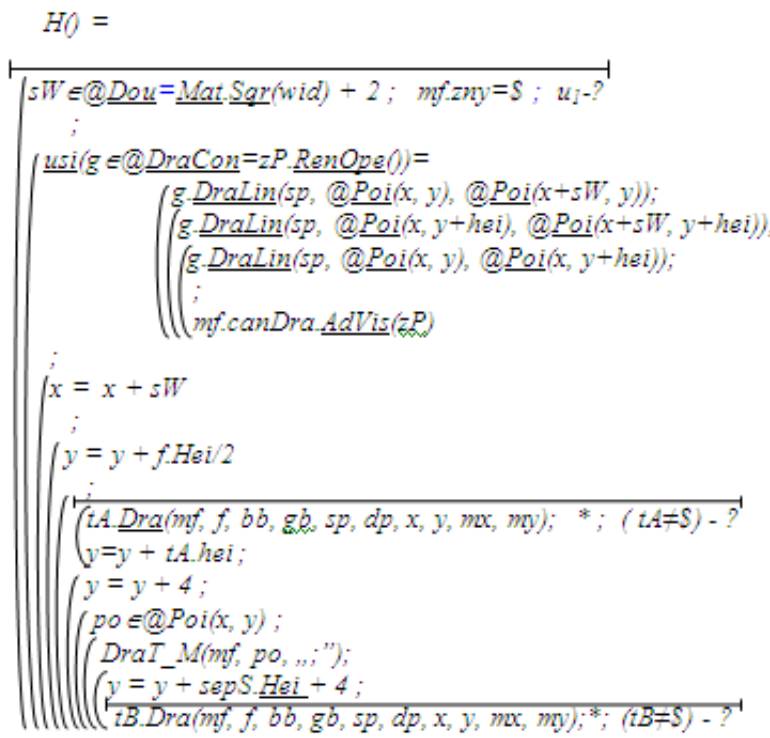
Узагальнена модель призначена для опису рисування знаків операції паралелення горизонтальної і вертикальної орієнтації та описується такою формулою:



де *zP∈@DraVisu* – декларація змінної *zP* типу стандартної підсистеми *DraVisu*, яка реалізується відомим класом `DrawingVisual()` [7, 8], *(ori=Ori.Hor)-?* – перевірка навності задання горизонтальної орієнтації операції паралелення *Ori.Hor*, u_1*-?* – умовний унітерм, який описується формулою *(mf.zny≠"Del")& (mf.zny≠"dWF")& (mf.zny≠"dWS") &(mf.zny≠"dWC") | (mf.zny=$),* у якій задано відсутність режиму видалення формули *(mf.zny≠"Del")* і режимів формування паралелення із залишеного першого *(mf.zny≠"dWF"),* другого *(mf.zny≠"dWS")* та без залишення попередніх унітермів *(mf.zny≠"dWC")* або наявність початкового значення *$* змінної *zny, sH∈@Dou=Mat.Sqr(wid) + 2* - обчислення висоти знаку *sH*, значення якого належить стандартній підсистемі *Dou* (він реалізується відомим класом `Double` [7, 8]), як корінь квадратний *Mat.Sqr()* від довжини *wid* плюс два пікселі, *mf.zny=$* - обнулення ознаки видалення або заміни унітермів, *usi(g∈@DraCon=zE.RenOpe())* - стандартний функційний унітерм *usi()*, який реалізується відомою процедурою `using` [7, 8] та описує доступ до призначених для формування і відображення графіки стандартних підсистеми *DraCon* і функційного унітерма *RenOpe()*, які реалізуються відомим класом `DrawingContext` і процедурою `RenderOpen()` [7, 8], *dc∈@DraCon* - створення змінної *dc* типу підсистеми *DraCon*, *dc.DraLin(sp, @Poi(x, y), @Poi(x, y+sH))* – рисування стандартним унітермом *DraLin()*, який реалізується відомою процедурою `DrawLine()`[7, 8], однієї із трьох ліній операції паралелення з використанням пера *sp* і кординат початкової *(x, y)* та кінцевої *(x, y+sH)* точок, які є типом стандартної підсистеми *Poi()*, що реалізується відомим класом `Point()`[7, 8], *dc.DraLin(sp, @Poi(x+wid, y), @Poi(x+wid, y+sH))* та *dc.DraLin(sp, @Poi(x, y), @Poi(x+wid, y))* - опис рисування другої та третьої ліній знаку операції паралелеання, *mf.canDra.AdVis(zP)* – до графічного елемента *canDra* головної

підсистеми *mf* додавання стандартним унітермом *AdVis()*, який реалізується відомим методом `AddVisual()`[7, 8], утвореного і збереженого в *zP* зображення знаку операції горизонтального паралелення, *x = x +f.Hei / 2* – збільшення значення абсциси на половину висоти кегля, *y = y + sH* – збільшення значення ординати, *(tA≠$)* – умовний унітерм перевірки наявності першого унітерма, *tA.Dra(mf, f, bb, gb, sp, dp, x, y, mx, my)* – вибір функційного унітерма *Dra()* для рисування першого унітерма операції елімінування, *x=x + tA.wid* – збільшення ординати знаку на довжину першого унітерма, *po∈@Poi(x, y)* – створення змінної стандартного класу і приписування їй значень поточних координат, *DraT_M(mf, po, „;”)* – функційний унітерм рисування розділювача унітермів, *x = x + sepS.Wid + f.Hei / 2* – збільшення поточного значення абсциси на довжину розділювача з врахуванням кеглю розділювача, *((tB≠$) - ?)* – перевірка наявності другого унітерма операції елімінування, *tB.Dra(mf, f, bb, gb, sp, dp, x, y, mx, my)* – рисування другого унітерма.

Модель функційного унітерму формування знаку операції вертикального паралелення *H()* аналогічна моделі операції горизонтального паралелення і описується такою формулою:

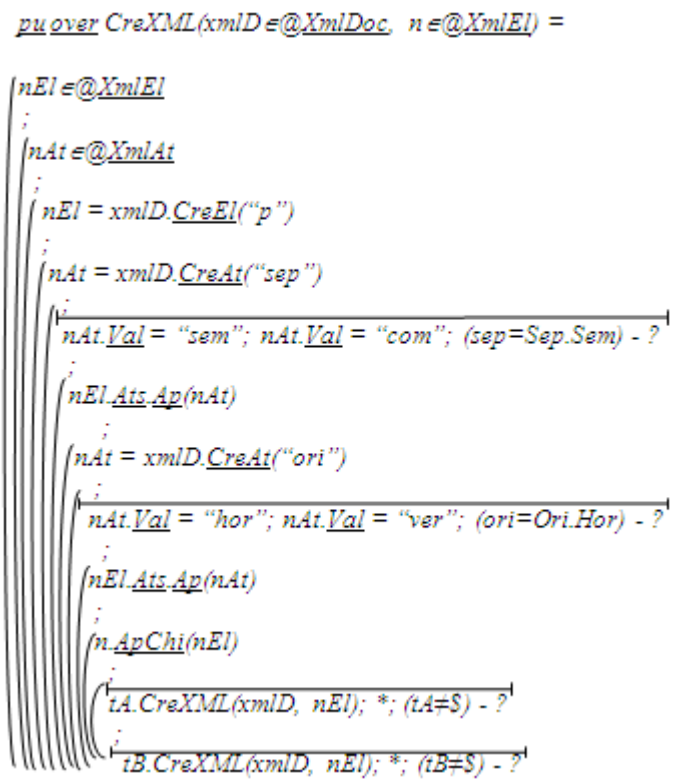


Модедь інформаційної технології формування xml-опису операції паралелення

Xml – опис операції паралелення має охоплювати ідентифікатори операції паралелення (p) та орієнтації (ori), а також значення орієнтації (hor чи ver) і самі унітерми.

Для самої назва (заголовока) функційного унітерма задаємо можливість доступу з інших підсистем (pu). Підсистема операції паралелення наслідує абстрактну підсистему Терм, у якій створено абстрактний функційний унітерм

формування xml – опису операцій алгебри алгоритмів. У звязку з цим у підсистемі паралелення необхідно описати дефініцію функційного унітерма формування xml - опису операції паралелення. Ознакою такої дифініції є over, яка платформою Microsoft Visual Studio .NET реалізується ключовим словом системи `overload` [7, 8]. Задаємо два вхідних параметри функційного унітерма. Один параметер xmlD типу стандартної підсистеми XmlDoc, яка реалізується відомим класом `XmlDocument` [7, 8]. Другий параметр n типу стандартної підсистеми XmlEl, яка реалізується відомим класом `XmlElement` [7, 8]. Формула яка описує створення xml – подібного опису з такими даними має такий вигляд:



де *xmlD*∈@XmlDoc – вхідна змінна *xmlD* типу підсистеми XmlDoc, яка реалізується відомим класом `XmlDocument` [4, 5], *n*∈@XmlEl – вхідна змінна *n* типу підсистеми XmlEl, яка реалізується відомим класом `XmlElement` [4, 5], *nEl*∈@XmlEl – створення змінної *nEl* типу XmlEl, *nAt*∈@XmlAt – створення змінної *nAt* типу XmlAt, яка реалізується відомим класом `XmlAttribute` [4, 5], *nEl* = *xmlD.CreEl("e")* – змінній *nEl* елемент *"e"* приписується типовим функційним унітермом *CreEl()*, який реалізується відомою процедурою `CreateElement()` [4, 5], *nEl.Ats.Ap(nAt)* – змінна *nEl* буде доповнена *Ap()* новими атрибутами *Ats*, *nAt* = *xmlD.CreAt("ori")* – до змінної *xmlD* введення *CreAt()* атрибуту *"ori"* і приписування значення змінної *xmlD* змінній *nAt*, *(ori=Ori.Hor)-?* – перевірка наявності задання горизонтальної орієнтації (*Ori.Hor*) операціїї секвентування, *nAt.Val* = *"hor"* та *nAt.Val* = *"ver"* - приписування атрибуту *nAt* значень *"hor"* та *"ver"*, *n.ApChi(nEl)* - ввести вложений елемент в *nEl*, *(tA≠$)-?* – перевірка наявності першого (*tA*) унітерму елімінування, *tA.CreXML(xmlD, nEl)* – формування *xml* –

елементу з першого унітерму елімінування, *(tB≠$)-?* – перевірка наявності другого (*tB*) унітерму елімінування, *tB.CreXML(xmlD, nEl)* - формування *xml* – елементу з другого унітерму елімінування, *(con≠$) - ?* – перевірка наявності унітерма-умови, *con.CreXML(xmlD, nEl)* – формування *xml* – елементу з унітерму-умови.

Моделлю інформаційної технології описано створення змінних типу елемента (*nEl∈@XmlEl*) і типу атрибута (*nAt∈@XmlAt*). Приписування цим змінним значення елемента (*nEl = xmlD.CreEl("p")*) і значення атрибута (*nAt = xmlD.CreAt("sep")*). Ними є ідентифікатор операції паралелення (*"p"*) та ідентифікатор розділювача (*"sep"*) унітермів операції паралелення. Розпізнавання розділювача крапки з комою (*Sem*) унітермів (*(sep=Sep.Sem) - ?*). Приписування змінній - атрибуту *nAt* значення розділювача крапки з комою (*nAt.Val = "sem"*), якщо значенням змінної *sep* є крапка з комою (*Sem*). Якщо ж значенням змінної *nAt* не є крапка з комою, то приписування змінній значення (*"com"*). До елемента дописування значення атрибуту (*nEl.Ats.Ap(nAt)*). Змінній – атрибуту приписування нового значення атрибуту (*nAt = xmlD.CreAt("ori")*), яким є ідентифікатор орієнтації операції паралелення. Встановлення чи є задана горизонтальна орієнтація операції паралелення (*(ori=Ori.Hor) - ?*). У випадку горизонтальної орієнтації приписування змінній – атрибуту значення горизонтальної орієнтації (*nAt.Val = "hor"*). Інакше змінній – атрибуту приписується вертикальне значення орієнтації операції паралелення (*nAt.Val = "ver"*). Приписування елементу значення атрибуту (*nEl.Ats.Ap(nAt)*). Введення нового елемента (*n.ApChi(nEl)*). Розпізнавання (*(tA≠$) - ?*) наявності унітерма *tA* операції паралелення. Якщо унітерм наявний, то формування *xml* – опису унітерма (*tA.CreXML(xmlD, nEl)*). Якщо ж унітерм відсутній, то формування *xml* – опису унітерма не відбувається. Перевірка наявності другого унітерма (*(tB≠$) - ?*). Коли унітерм є, то створення його *xml* – опису (*tB.CreXML(xmlD, nEl)*). Якщо ж унітерм відсутній, то формування *xml* – опису унітерма не відбуваєься.

Фрагмент програмної реалізації моделі інформаційної технології

Мовою обєктного програмування є C#, платформи Microsoft Visual Studio.NET, фрагмент коду узагальненої математичної моделі інформаційної технології формування операції паралелення має такий вигляд:

```
DrawingVisual znakParalelennia = new DrawingVisual();
            if (orientation == Orientation.Horizontal)
            {
                if (((mf.znyszczyty != "DeleteAll")
                      && (mf.znyszczyty != "deleteWithoutFirst")
                        && (mf.znyszczyty !=
"deleteWithoutSecond")
                         && (mf.znyszczyty !=
"deleteWithoutChild"))
                       || (mf.znyszczyty == null))
                  {
                    int symbolHeight =
((int)Math.Sqrt(width) / 2) + 2;
                    using (DrawingContext g =
znakParalelennia.RenderOpen())
                    {
                        g.DrawLine(sp, new Point(x, y),
```

```
new Point(x, y + symbolHeight));
                        g.DrawLine(sp, new Point(x +
width, y), new Point(x + width, y + mbolHeight));
                        g.DrawLine(sp, new Point(x, y),
new Point(x + width, y));
mf.canwasDraw.AddVisual(znakParalelennia);
                    }
                    x += (f.Height / 2);
                    y += symbolHeight;
                    if (termA != null)
                    {
                        termA.Draw(/*dv,*/mf, f, bb, gb,
sp, dp, x, y, marginX, marginY);
                        x += termA.width;
                    }
                    x += /*(int)*/(/*bulo:
f.Size*/f.Height / 2);
                    po = new Point(x, y);
                    DrawText_M(mf, po, ";");
                    x += separatorSize.Width;
                    x += (/f.Height / 2);
                    if (termB != null)
                        termB.Draw(/*dv,*/mf, f, bb, gb,
sp, dp, x, y, marginX, marginY);
                }
                    else
                    {
                        mf.znyszczyty = null;
                    }
            }
            else
            {
                if (((mf.znyszczyty != "DeleteAll")
                      && (mf.znyszczyty != "deleteWithoutFirst")
                        && (mf.znyszczyty !=
"deleteWithoutSecond")
                          && (mf.znyszczyty !=
"deleteWithoutChild"))
                       || (mf.znyszczyty == null))

double symbolWidth = (Math.Sqrt(height) / 2) + 2;
                using (DrawingContext g =
znakParalelennia.RenderOpen())
                    {
                     g.DrawLine(sp, new Point (x, y),
new Point (x + symbolWidth, y));
                     g.DrawLine(sp, new Point  (x, y + height),
new Point (x + symbolWidth, y + height));
                     g.DrawLine(sp, new Point (x, y),
new Point (x, y + height));
mf.canwasDraw.AddVisual(znakParalelennia);
                    }
                x += symbolWidth;
                y += (f.Height / 2);
```

```
                if (termA != null)
                {
                    termA.Draw(/*dv,*/mf, f, bb, gb, sp,
dp, x, y, marginX, marginY);
                    y += termA.height;
                }
                y += 4;
                po = new Point(x, y);
                DrawText_M(mf, po, ";");
                y += separatorSize.Height;
                y += 4;
                if (termB != null)
                   termB.Draw(/*dv,*/mf, f, bb, gb, sp,
dp, x, y, marginX, marginY);
                }
                    else
                    {
                        mf.znyszczyty = null;
                    }
            }
        }
```

Висновки

1. Практичною реалізацією і апробацією підтверджено, що загальною математичною моделлю описується інформаційна технологія автоматичного формування операції паралелення горизонтальної і вертикальної орієнтації з врахуванням розмірів унітермів.

2. Створення математичної моделі формування *xml* – опису забезпечує можливості використання стандартних процедур створення, опрацювання і модифікації *xml* – формату операції паралелення.

1. Owsiak W., Owsiak A., Owsiak J. Teoria algorytmów abstrakcyjnych i modelowanie matematyczne systemów informacyjnych. – Opole: Politechnika opolska, 2005. – 275 s.

2. Ovsyak V.K.: Computation Models and Algebra of Algorithms. http://www.nbuv.gov.ua/Portal/natural/VNULP/ISM/2008_621/01.pdf

3. Owsiak W., Owsiak A. Rozszerzenie algebry algorytmów / Pomiary, automatyka, kontrola. – № 2, 2010. – S. 184 – 188.

4. Бритковський В.М. Моделювання редактора формул секвенційних алгоритмів: автореф. дис. на здобуття наук. ступеня канд. тех. наук: спец. 01.05.02 "Математичне моделювання та обчислювальні методи" / В.М. Бритковський – Львів, 2003. – 18 с.

5. Василюк А.С. Підвищення ефективності математичного і програмного забезпечення редактора формул алгоритмів: автореф. дис. на здобуття наук. ступеня канд. тех. наук: спец. 01.05.02 "Математичне та програмне забезпечення обчислювальних машин і систем" / А.С. Василюк – Львів, 2008. – 20 с.

6. Дейтел Х., Дейтел П., Листфилд Дж., Нието Т., Йегер Ш., Златкина М. C#. – М.: СПб.: БХВ-Петербург, 2006. –1056 с.

7. Petzold C. Programowanie Windows w języku C#. – Warszawa: „RM", 2002. – 1161 s.

8. Мэтью Мак-Дональд. Windows presentation foundation в .NET 3.5 с примерами на C#2008. – Москва, Санкт-Петербург, Киев: "Apress", 2008. – 922 с.