Овсяк О. Моделі інформаційних технологій операції циклічного секвентування / Овсяк О. // Вісник ТНТУ. — 2011. — Том 16. — № 3. — С.128-141. — (приладобудування та іформаційно-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 ОПЕРАЦІЇ ЦИКЛІЧНОГО СЕКВЕНТУВАННЯ

***Резюме**. Засобами алгебри алгоритмів описано створені математичні моделі декомпозиції підсистеми циклічного секвентування, інформаційних технологій обчислення розмірів, вибору, формування горизонтальної і вертикальної орієнтацій, видалення і створення XML-формату операції циклічного секвентування. Мовою об’єктного програмування C #, платформи Microsoft Visual Studio .NET програмно реалізовано моделі функційних унітермів формування і XML-формат операції циклічного секвентування.*

***Ключові слова:** декомпозиція, модель, алгоритм, XML-формат, циклічне секвентування, функційний унітерм.*

A.Ovsyak

MODELS INFORMATIONS TECHNOLOGEIS OF OPERATION CYCLICAL SEQUENTION

***The summary.** By means of algebra algorithms described the mathematical models decomposition subsystem cyclic sequencing of information technologies computing size, choice of horizontal and vertical orientations, deletion and creating XML - format operation cycle sequencing. Object programming language C #, platform Microsoft Visual Studio .NET, software implemented and tested model of xml - description and of functional uniterm creating operation cycle sequencing.*

***Key words:** decomposition, model, algorithm, xml - format, cycle sequencing, function uniterm.*

1. Аналіз останніх досліджень і постановка проблеми. Засоби алгебри алгоритмів [1] та її розширення [2] забезпечують опис алгоритмів інформаційних технологій і систем у вигляді математичних формул над якими можуть виконуватися тотожні алгебраїчні перетворення [3, 4]. Відомо [5], що таких можливостей не забезпечують інші відомі еквівалентні методи [5], наприклад [6]. Однак система позначень операцій алгебри алгоритмів та її розширення мають специфічні знаки [1, 2, 3, 4], яких немає серед відомих математичних позначень, а їхні геометричні розміри залежать від геометричних розмірів унітермів і мають складну геометричну конфігурацію.

Є можливим створювати складну систему знаків алгебри алгоритмів засобами, наприклад, таких комп’ютерних систем, якими є такі найпоширеніші системи, як Word, Coral Draw, Page Maker, In Design та інші. Однак використання таких універсальних систем для набору і редагування формул алгоритмів потребує великих затрат праці й комп’ютерного часу [3, 4]. А створеними для набору і редагування формул алгебри алгоритмів моделями спеціалізованих комп’ютерних систем МОДАЛ [3] і АБСТРАКТАЛ [4] не забезпечуються можливості виконання автоматичної оптимізації формул алгоритмів на підставі використання *XML*-формату опису операції циклічного секвентування.

Метою роботи є інформаційні технології створення й використання *XML*-формату опису операції циклічного секвентування та їхня програмна реалізація.

Завдання дослідження. Створити й описати засобами розширеної алгебри алгоритмів моделі декомпозиції підсистеми циклічного секвентування, інформаційних технологій комп'ютерного обчислення розмірів, вибору, видалення, формування горизонтальної та вертикальної орієнтації та *XML*-опису операції циклічного секвентування, які запрограмувати мовою C# платформи Microsoft Visual Studio .NET [7, 8].

2. Декомпозиція підсистеми циклічного секвентування. Для забезпечення доступу до підсистеми циклічного секвентування (***@cS***) з інших підсистем задаємо відкритий метод доступу (***pu***). Можливість використання функційних унітермів підсистеми Терм (***@T***) [9] підсистемою циклічного секвентування запишемо як ***@cS:@T***. Підсистема циклічного секвентування використовує змінні (***Z***) і такі функційні унітерми: задавання початкових значень змінних ***cS()***; обчислення розмірів унітермів операції c***Cs()***, її вибору ***Cfc()***, зняття ознаки вибору ***Des()***, формування операції (***Dra()***) з рисуванням рамки вибору (***Ram()***) і видалення (***vF()***) операції, рисуванням її знака горизонтальної (***hcS()***) і вертикальної (***vcS()***) орієнтації; формування *XML*-опису (***cXML()***). Взаємне розташування складових системи немає значення, тому у формулі запису декомпозиції застосовуємо кому для їхнього розділення. Наявність усіх названих складових є необхідною, а їхній вибір на виконання здійснюється послідовно, тому для записування формули декомпозиції системи використовуємо операцію секвентування розширеної алгебри алгоритмів

$$pu\ @cS{:}@T = \overbrace{Z,\ cS(),\ Cs(),\ Cfc(),\ Des(),\ Dra(),cXML()}.$$

3. Опис моделей складових системи

3.1. Змінні (***Z***)

Операція циклічного секвентування має горизонтальну (***Hor***) і вертикальну (***Ver***) орієнтації. Для задавання орієнтації вводимо загальнодоступну (***pu***) секвентну область ***Ori***, яка належить ($\in$) до типу ***enu*** стандартної підсистеми enum [7, 8]

$$pu\ Ori \in enu = \overbrace{Hor;\ Ver}.$$

Операція циклічного секвентування має унітерми умови (***cond***) й області дії (***t***), які належать до підсистеми *Term* (***@T***) [10] та є загальнодоступними: ***pu cond*** $\in$ ***@T*** і ***pu t*** $\in$ ***@T***.

Для виконання рисування контуру знака операції циклічного секвентування водимо змінну ***pen***, яка належить до стандартного класу Pen [7, 8], лінією, ширина якої обчислюється як алгоритм із величини кегля (***fS***) унітермів плюс 3 одиниці, що запишемо як ***pen*** $\in$ ***@Pen*** = ***@Mat.Log(fS)***, де ***@Mat.Log(fS)*** – стандартний клас математичних функцій Math, який містить стандартний метод обчислення логарифму Log(). Змінна ***bR*** стандартного класу Brushes [7, 8] призначена для зафарбування місць розташування унітермів ***bR*** $\in$ ***@Brushes***. Змінні ***dc, sS*** і ***pO*** належні до стандартних класів DrawingContext, Size і Point [7, 8], призначені для використання стандартних методів цих класів.

3.2.Модель функційного унітерма ***cS()***

Алгоритм є загальнодоступним і описується формулою

$$\underline{pu}\ cS() = \begin{pmatrix} cond = \$ \\ ; \\ t = \$ \\ ; \\ ori = Ori.Hor, \end{pmatrix}$$

де ***$*** – початкове значення змінних типів системи і підсистеми.

3.3. Формула обчислення розмірів унітермів (с***Cs()***)

Вона описується формулою (1) у якій: змінні ***dv*** $\in$ ***@DraV*** і ***f*** $\in$ ***@Siz*** типу стандартних підсистем, які реалізуються відомими класами DravingVisual і Size [7, 8]; ***ov*** – ознака дефініції функційного унітерма у даній системі; ***tex*** $\in$ ***@FormTex*** = ***FormTex(“***W***”)*** – створення змінної ***tex*** типу стандартної підсистеми ***FormTex***, яка реалізується відомим класом FormatedTexst [7, 8] і приписування змінній сформатованого стандартним методом FormatedTexst() значення тексту W; ***sS=@Siz(tex.Wid , tex.Hei)*** – обчислення довжини (***Wid***) і ширини (***Hei***) тексту та приписування їх змінній ***sS***; ***wid = sS.Wid + f.Hei*** - обчислення довжини з врахуванням кегля; ***hei=sS.Hei*** – приписування змінній ***hei*** висоти тексту; ***(cond≠$)-?*** – перевірка наявності унітерма умови; ***cond.Cs(dv, f)*** – обчислення розмірів унітерма – умови; – обчислення загальної довжини операції з врахуванням довжини унітерма – умови; ***(hei<cond.Hei)-?*** – порівняння текучої висоти операції з висотою унітерма – умови; ***hei=cond. Hei*** – приписування поточній висоті операції висоти унітерма

```
pu ov cCs(dv∈@DraV, f∈@Siz) =
    tex∈@FormTex = FormTex("W")
        ;
    sS=@Siz(tex.Wid , tex.Hei)
        ;
        wid = sS.Wid + f.Hei
            ;
            hei=sS.Hei
                ;
                cond.Cs(dv, f); * ; (cond≠$)-?
                ;
                wid=wid+cond.Wid
                ;
                hei=cond. Hei; * ; (hei<cond.Hei)-?
                ;
                t.Cs(dv, f) ; * ; (t≠$)-?
                ;
                wid = wid + t.Wid
                ;
                hei=t.Hei ; * ; (hei<t.Hei)-?
                ;
                wid=sS.Wid+f.Hei/2
                ;
                hei=sS.Hei+4
                ;
                cond.Cs(dv, f);* ; (cond≠$)-?
                ;
                wid=wid+cond.Wid
                ;
                hei=cond.Hei;* ; (hei<cond.Hei)-?
                ;
                t.Cs(dv, f) ; * ; (t≠$)-?
                ;
                wid=t.Wid ; * ; (wid<t.Wid)-?
                ;
                hei=hei+ t.Hei
            ;
        (ori=OriHor)-?
```

(1)

– умови; ***(t≠$)*** – перевірка наявності унітерма, зв'язаного операцією циклічного секвентування; ***t.Cs(dv, f)*** – обчислення розмірів унітерма, зв'язаного операцією циклічного секвентування; ***wid = wid + t.Wid*** – обчислення поточної довжини операцій з врахуванням довжини області дії операції; ***(hei<t.Hei)-?*** – порівняння текучої висоти операції з висотою області дії операції; ***hei=t.Hei*** – поточному значенню висоти приписування значення унітерма – області дії операції циклічного секвентування; ***wid=sS.Wid+f.Hei/2*** – обчислення поточної довжини операції з врахуванням висоти кегля; ***(ori=OriHor)-?*** – перевірка наявності горизонтальної орієнтації операції циклічного секвентування.

3.4. Алгоритм вибору операції циклічного секвентування (***Cfc()***)

Нехай функційний унітерм матиме загальний доступ (***pu***), вихідну змінну ***te*** і вхідні змінні ***mf*** – типу підсистеми (*MainForm* [10]) ***@Mf***, ***f*** – типу стандартної

підсистеми Size [7, 8], ***x-y-mX-mY-moX-moY*** – типу стандартної підсистеми Double [7, 8] описується формулою

```
pu ov (te∈@T) Cfc(mf∈@Mf, f ∈@Siz, x-y-mX-mY-moX-moY ∈@Dou)=
    (tex∈@FormTex = FormTex("W")
        ;
    selT∈@T
     ;
     xT∈@Dou
      ;
      yT∈@Dou
       ;
       G
```

У ній описано приписування змінній ***tex*** відформатованого тексту і створення змінних ***selT*** та ***xT*** і ***yT*** типу стандартної підсистеми ***Dou***, яка реалізується відомим класом Double [7, 8]. Унітерм ***G*** описується формулою (2), де ***xT =xT+sS.Wid*** – обчислення і приписування змінній ***xT*** текучого значення довжини операції циклічного секвентування; ***xT=xT+f.Hei/2*** – врахування величини кегля; ***((selT=cond.Cfc(mf, f, xT, yT, mX, mY, moX, moY))≠$)-?*** – приписування змінній обчислених розмірів унітерма– умови і порівняння її значення змінної з порожнім.

```
(xT =xT+sS.Wid
 ;
 xT=xT+f.Hei/2;
  ;
  (te=selT ;
   xT=xT+cond.Wid+f.Hei;
    ((selT=cond.Cfc(mf,f,xT,yT,mX,mY,moX,moY))≠$)-? ;
   * ;
   (cond≠$)-?

   ;
   (te=selT ;
    *;
    ((selT=t.Cfc(mf,f,xT,yT,mX,mY,moX,moY))≠$)-? ;
   * ;
    (t≠$)-?
    ;
 xT=xT+sS.Wid;                                          (2)
  ;
  xT=xT+f.Hei/2;
   (te=selT;
    yT=yT+cond.Hei;
    yT=yT+sS.Hei ;
    (s$.Hei<cond.Hei)-? ;
   ((selT=cond.Cfc (mf,f,xT,yT,mX,mY,moX,moY))≠$)-? ;
   yT=yT+sS.Hei;
   (cond≠$)-?
    ;
    (yT=yT+4; xT=xT-sS.Wid-f.Hei/2 ;
     (te=selT ;
      * ;
      ((selT=t.Cfc (mf,f,xT,yT,mX,mY,moX,moY))≠$)-? ;
     * ;
      (t≠$)-?
      ;
 (ori=Ori.Hor)-?
 ;
 K
```

te=selT – приписування вихідній змінній ***te*** значення змінної ***selT***; ***xT = xT+cond.Wid+f.Hei*** – із врахуванням довжини умовного унітерма і висоти кегля обчислення текучої довжини операції; ***(cond≠$)-?*** – перевірка наявності унітерма-умови; ***yT=yT+cond.Hei*** – обчислення значення змінної ***yT*** із врахуванням висоти унітерма-умови ***cond.Hei***; ***yT=yT+sS.Hei*** – обчислення значення змінної ***yT*** із врахуванням поточної висоти операції секвентування; ***(s$.Hei<cond.Hei)-?*** – порівняння значень поточної висоти операції циклічного секвентування ***s$.Hei*** з висотою унітерма-умови ***cond.Hei***; ***yT=yT+4*** – збільшення висоти на пропуск між двома рядками операції циклічного секвентування; ***xT=xT-sS.Wid-f.Hei/2*** – зменшення довжини операції вертикального циклічного секвентування з врахуванням розміру кегля ***f.Hei/2*** та довжини унітерма ***sS.Wid***, зв'язаного операцією циклічного секвентування; ***(ori=Ori.Hor)-?*** – розпізнавання горизонтальної орієнтації операції; ***K*** – вираз, який описується такою формулою:

```
K=( re∈@Rec= Rec(x-f.Hei/4-4, y-f.Hei/8-2, wid+f.Hei/2+4,
                                              hei+f.Hei/4+4)
    ;
   |( mf.cn=tr;   ( * : (sel=re.Com(moX, moY))-?
    |  ;           ;
    |( mf.sel=fa   te=$.
    |  ;
    |( te=thi.
```

де ***re∈@Rec= Rec(x-f.Hei/4-4, y-f.Hei/8-2, wid+f.Hei/2+4, hei+f.Hei/4+ 4)*** – введення змінної типу стандартної підсистеми ***Rec***, яка реалізується відомим класом Rect [7, 8], та приписування цій змінній обчисленого значення прямокутної області операції циклічного секвентування; ***sel=re.Com(moX, moY))*** – з використанням стандартного функційного унітерма ***Com()***, який реалізується відомим методом Contains()[7, 8], перевірка попадання координат курсора в обчислену прямокутну область ***re*** та приписування логічного значення, яке видається функційним унітермом ***Com()***, змінній ***sel***; ***mf.cn=tr*** – приписування складній змінній логічного значення true []; ***mf.sel=fa*** – приписування складній змінній логічного значення false [7, 8]; ***te=thi.*** – приписування вихідній змінній значень змінних функційного унітерма ***cS()*** і завершення (.) виконання функційного унітерма ***Cfc()***; ***te=$.*** – приписування вихідній змінній значення ***$*** і завершення (.) виконання функційного унітерма ***Cfc()***.

3.5. Загальний алгоритм формування операції циклічного секвентування

Починається алгоритм форматуванням тексту ***tex∈@FormTex = FormTex("W")*** і створенням змінної ***sS=@Siz(tex.Wid, tex.Hei)***, якій приписується значення довжини і ширини відформатованого тексту з подальшим рисуванням рамки вибору (***Ram()***) і рисування операції секвентування (***RyscSe()***), якщо виконуються умови вибору (***selT=tr***) та відсутності режиму видалення (***mf.zn=$***) операції циклічного секвентування.

```
pu ov Dra(mf∈@MaFor, f∈@Siz, bb∈@Bru, gb∈@Bru, sp∈@Pen, dp∈@Pen, x:y:mx:my∈@Dou)=
                                    ( tex∈@FormTex = FormTex("W")
                                    ;
                                    |( sS=@Siz(tex.Wid , tex.Hei)
                                    ;
                                    |( Ram(); *; ((selT=tr)&(mf.zn=$))-?
                                    ;
                                    |( vF()
```

3.6. Рисування рамки вибору

Для рисування рамки вибору операції утворюємо змінні ***rvcS*** та ***dc*** типу стандартних системи ***DraVisu( )***та ***DraCon***, які реалізується відомими класами DrawingVisual() та DawingContext [7, 8] і використовуємо стандартні функційні унітерми ***RenOpe(), DraRec(), AddVisu()*** й ***RectGeo()***, які реалізуються відомими методами RenderOpen(), DrawRectangle(), AddVisual() й RectangleGeometry()[7, 8]. Формула що описуэ рисування рамки вибору операції, має вигляд

```
Ram()= rvcS∈@DraVisu()
         ;
       usi(dc∈@DraCon = rvcS.RenOpe())=dc.DraRec($, dp, Rec(x + f.Hei/4 - 4,
                                                            y - f.Hei/8 - 2,
                                                            wid + f.Hei/2 + 4,
                                                            hei + f.Hei/4 + 4))
          ;
        mf.cDra.AddVisu(rvcS)
           ;
        mf.zrvcS= rvcS
           ;
         mf.raM=RectGeo(Rec(x + f.Hei/4 - 6, y - f.Hei/8 - 4, wid + f.Hei/2 + 8,
                                                            hei + f.Hei/4 + 8))
           ;
         mf.raMz=raM
```

У ній другий зверху рядок задає створення рамки вибору, яка третім рядком долучається до висвітлюваних на екрані комп'ютера графічних об'єктів і зберігається у змінній ***zrvcS***, яка належить до головної підсистеми (MainForm). Після цього формується рамка вибору операції, яка дещо більша за попередню рамку вибору і призначена для видалення рамки вибору у разі потреби. ***mf.raMz=raM*** – описує запам'ятовування збільшеної рамки вибору у складній змінній ***mf.raMz***.

3.7. Модель видалення операції

Функційний унітерм ***vF()*** описується формулою (3). У ній ***(mf.zn=tr)-?*** описує аналіз збігу значення складної змінної з типовим значенням ***tr***, і якщо збіг немає місця, то видалення операції циклічного секвентування оминається та переходиться до виконання функційного унітерма рисування операції циклічного секвентування ***DracS()***. Інакше, змінній ***mf.zn*** приписується типове значення ***fa*** та починається процес видалення операції циклічного секвентування.

Умовний унітерм u_1, яким визначаються можливості видалення операції, описується виразом ***(mf.zn≠zO)&(mf.zn≠zU)|(mf.zn=$)***, де ***(mf.zn≠zO)& mf.zn≠zU)*** – відсутність режимів заміни операції (***mf.zn≠zO***) і заміни унітерма операції (***mf.zn≠zU***) або (|) відсутності режиму знищення (***mf.zn=$***).

Виконання умовного унітерма u_1 супроводжується обчисленням прямокутника (***rg=new RecGeo(new Rec(x - f.Hei/4 - 4, y - f.Hei/8 - 2, wid + f.Hei/2 +6, hei + f.Hei/4 + 4))***), з якого має видалитися операція. Після цього перевіряється виконання умовного унітерма u_2, який утворено виразом ***(mf.rp≠$)&(rg.Rec.Hei< mf.rp.Rec.Hei)& (rg.Rec.Wid< mf.rp.Rec.Wid)***. У ньому перевіряється наявність (***mf.rp≠$***) попередньої рамки (***mf.rp***) операції й порівняння висоти (***rg.Rec.Hei< mf.rp.Rec.Hei***) і довжини

(***rg.Rec.Wid*** < ***mf.rp.Rec.Wid***) рамки ***rg*** з рамкою ***rp***. Для виділення з рамок їхніх висоти і ширини використовується типовий унітерм ***Rec***, який реалізується стандартною властивістю Rect [7, 8].

Якщо умовний унітерм u_2 виконується, то розміри рамки ***rg*** заміняються (***rg.Rec=mf.rp.Rec***) розмірами рамки ***rp***, які після цього заміняються (***mf.rp=mf.zrv***) розмірами вище утвореної рамки ***zrv***. Невиконання умовного унітерма u_2 призводить до введення (***mf.fig=mf.cDra.GetVis(rg)***) рамки ***rg*** до всіх інших раніше утворених фігур та обчислення кількості всіх фігур ***mf.fig.Cou*** і приписування її складній змінній ***mf.kf***. Обчислення кількості фігур описується типовим унітермом ***Cou***, який реалізується стандартною властивістю Count [7, 8]. Введення рамки ***rg*** до фігур, які висвітлюються на ***mf.cDra***, описується типовим функційним унітермом ***GetVis()***, що реалізується відомим методом GetVisuals()[7, 8]. Після цього перевіряється наявність фігур (***(mf.kf>0)-?***). Якщо вони наявні, то відкривається цикл ($\lhd i$) для здійснення видалення фігур функційним унітермом ***SpaFig(i, mf)*** з поверненням у цикл (c_i), якщо виконується умова ***i<kf***.

vF() =

```
mf.zn = fa ; *; (mf.zn = tr)-?
    ;
  rg=new RecGeo(new Rec(x - f.Hei/4 - 4, y - f.Hei/8 - 2, wid + f.Hei/2 +6, hei + f.Hei/4 + 4))
  ;
  rg=new RecGeo(new Rec(x - f.Hei/4 - 6, y - f.Hei/8 - 6, wid + f.Hei/2 +10, hei + f.Hei/4 + 10))
  ;
  u1-?
    ;                                   ;
   (rg.Rec=mf.rp.Rec;  mf.fig=mf.cDra.GetVis(rg); u2-?
        ;                    ;
    mf.rp=mf.zrv        mf.kf=mf.fig.Cou                                   (3)
        ;
    ◁ i                                 ; *; (mf.kf>0)-?
     (SpaFig(i, mf); *; (i<kf)-?
      ;
      ci
  ;
DracS()
```

Невиконання умовного унітерма u_1 супроводжується обчисленням (***mf.geo=new RecGeo(new Rec(x - f.Hei/4 - 6, y - f.Hei/8 - 6, wid + f.Hei/2 +10, hei + f.Hei/4 + 10))***) прямокутної області операції.

3.8. Модель формування операції

Загальна модель алгоритму рисування операції циклічного секвентування включає функційні унітерми і створення змінної ***zcS*** типу стандартної підсистеми ***DraVisu***, яка реалізується відомим класом DrawingVisual [7, 8] та описана формулою (4). Вона утворена формуванням горизонтальної і вертикальної (***H()***) орієнтації операції, які розпізнаються умовним унітермом ***ori=Ori.Hor***. Значення ***Ori.Hor*** змінної ***ori*** відповідає горизонтальній орієнтації операції.

3.8.1.Алгоритм формування операції горизонтальної орієнтації

Починається алгоритм функційним унітермом ***usi***, який реалізується типовою директивою using [7, 8]. Нею вводиться змінна ***dc*** стандартної підсистеми ***DraCon***, яка реалізується відомим класом DrawingContext [7, 8]. Для виконання контекстного рисування з використанням змінних ***dc*** і ***zcS,*** типовим функційним унітермом ***RenOpe()***, який реалізується відомою процедурою RenderOpen()[7, 8], відкривається (***usi (dc∈@DraCon=zcS.RenOpe())***) можливість введення складових операції циклічного секвентування на ***mf.cDra***. Змінній ***po*** типу стандартної підсистеми ***Poi***, яка реалізується відомим класом Point()[7, 8], приписуються обчисленні значення координат (***x+sS.Wid/2, y+sS.Hei*0.2***) початкової точки рисування дуги знака операції циклічного секвентування. ***DracSeH(mf, dc, po)*** – функційний унітерм рисування дуги знака операції горизонтальної орієнтації. Типовим функційним унітермом ***DraLin()***, який реалізується відомою процедурою DrawLine() [7, 8], для обчислених координат початкової (***Poi(x+sS.Wid*0.5, y+sS.Hei*0.2)***) і кінцевої ***Poi(x+sS.Wid, y+sS.Hei*0.2)***) точок рисування однієї з трьох ліній операції циклічного секвентування горизонтальної орієнтації. ***mf.cDra.AdVis(zcS)*** – опис відображення на ***mf.cDra*** введеної операції. ***x=x+sS.Wid+f.Hei/2*** – збільшення довжини операції на знак операції секвентування з врахуванням поправки на висоту кегля шрифта. ***(cond≠$) - ?*** – перевірка наявності унітерма-умови. ***cond..Dra(mf, f, bb, gb, sp, dp, x, y, mx, my)*** – функційний унітерм створення унітерма-умови. ***x=x + cond.wid*** – збільшення довжини операції на довжину умовного унітерма операції. ***x = x + f.Hei / 2*** – врахування через розмір кегля відступу між знаком операції і умовним унітермом операції, ***(t≠$) - ?*** – перевірка наявності значення функційного унітерма операції циклічного секвентування, ***t.Dra(mf, f, bb, gb, sp, dp, x, y, mx, my)*** – формування функційного унітерма операції, ***mf.zn=$*** – обнулення ознаки видалення, u_1***-?*** – перевірка виконання умовного унітерма u_1.

3.8.2. Модель циклічного секвентування вертикальної орієнтації

Опис операції з вертикальною орієнтацією наведена формулою (5). Вона від формули (4) відрізняється тим, що невиконання умовного унітерма ***cond≠$) - ?*** призводить до виконання унітерма ***y=y+sS.Hei***, яким збільшується висота виразу операції циклічного секвентування. Також введено елімінування за умовою ***sS.Hei<cond.hei***, якою порівнюється поточне значення висоти операції з висотою унітерма-умови. Виконання умови збільшує висоту операції, яка обчислюється виразом ***y=y+cond.Hei***. Невиконання цієї умови призводить до збільшення висоти операції, яке обчислюється ***y+sS.Hei***. Після цього зменшується довжина виразу операції ***x=x - sS.Wid - f.Hei/2*** і збільшується її висота ***y = y + 4***.

4. Модель інформаційної технології формування *xml*-опису. *XML*-опис операції секвентування має охоплювати ідентифікатори операції секвентування (**cs**), ідентифікатор орієнтації (***ori***) і значення орієнтації (***hor*** чи ***ver***) та самі унітерми умови і функційний унітерм. Формула (6) описує створення *XML*-опису операції циклічного секвентування з такими параметрами.

```
DracS()=
  zcS ∈@DraVisu
   ;
    usi (dc ∈@DraCon=zcS.RenOpe())= po ∈@Poi(x+sS.Wid/2, y+sS.Hei*0.2)
                                    ;
                                    DracSeH(mf, dc, po)
                                    ;
                                    dc.DraLin(sp, new Poi(x+sS.Wid*0.5, y+sS.Hei*0.2),
                                                  new Poi(x+sS.Wid, y+sS.Hei*0.2))
                                     ;
                                    dc.DraLin(sp, new Poi(x+sS.Wid*0.5, y+sS.Hei*0.8),
                                                  new Poi(x+sS.Wid, y+sS.Hei*0.8))
                                    ;
                                    dc.DraLin(sp, new Poi(x+sS.Wid*0.3, y+sS.Hei),
                                                  new Poi(x+sS.Wid*08, y))
                                    ;
                                     mf.cDra.AdVis(zcS)
     ;
     x=x+sS.Wid+f.Hei/2
     ;
      cond..Dra(mf, f, bb, gb, sp, dp, x, y, mx, my);  * ;  (cond≠$) - ?
        ;
       x=x + cond.wid
        ;
       x = x + f.Hei / 2
        ;
       t.Dra(mf, f, bb, gb, sp, dp, x, y, mx, my); mf.zn=$; (t≠$) - ?
    ;
   mf.zn=$
    ;
   u1-?
    ;
  vcS()
    ;
  (ori=Ori.Hor)-?
```

(4)

Функційний унітерм формули (6) ***pu over cXML(xmlD ∈@XmlDoc, n ∈@XmlEl)*** є доступним (***pu***) з інших підсистем, заміняє його означення (***over***), дане у спадковій підсистемі (***Term***), має два вхідних параметри ***xmlD*** і ***n***, які є типами стандартних підсистем ***XmlDoc*** і ***XmlEl***, реалізованих відомими класами XmlDocument та XmlElement [7, 8]. Створення ***XML***-опису починається введенням змінних ***nEl*** і ***nAt*** типових підсистем ***XmlEl*** і ***XmlAt***, які реалізуються відомими класами XmlElement і XmlAttribute [7, 8]. ***nEl = xmlD.CreEl(“cs”)*** – створення елемента ***cs*** і приписування його типовим унітермом ***CreEl()***, який реалізується відомою процедурою CreateElement() [7, 8], змінній ***nEl***. ***nAt = xmlD. CreAt(“ori”)*** – створення атрибута ***ori*** типовим функційним унітермом ***CreAt()***, який реалізується відомою процедурою CreateAttribute() [7, 8], і приписування його значення змінній ***nAt***. (***(ori=Ori.Hor) - ?***) – розпізнавання горизонтальної (***Ori.Hor***) орієнтації операції циклічного секвентування. У випадку заданої горизонтальної орієнтації операції, типовим унітермом ***Val***, який реалізується відомою властивістю Value [7, 8], змінній ***nAt*** присвоєння значення ***hor***. Інакше цій змінній приписується значення ***ver***. ***nEl.Atr.Ap(nAt)*** – дописування типовим функційним унітермом ***Ap()***, який реалізується відомою процедурою Append()[7, 8], до змінної ***nEl*** значення атрибута ***nAt*** з використанням типового унітерма ***Atr***, який реалізується відомою властивістю Attributes [7, 8]. ***n.ApChi(nEl)*** – приписування типовим функційним унітермом ***ApChi()***, який реалізується відомою процедурою

AppendChild()[7, 8], змінній ***n*** значення змінної ***nEl***. (***(cond≠$) - ?***) – перевірка наявності унітерма-умови. ***cond.cXML(xmlD, nEl)*** – формування з використанням функційного унітерма ***cXML()*** підсистеми ***Uniterm***, ***XML*** – опису унітерма-умови, яка є значенням змінної ***cond***. ***(t≠$) - ?*** – перевірка наявності функційного унітерма операції. ***t.cXML(xmlD, nEl)*** – формування з використанням функційного унітерма ***cXML()***, ***XML***-опису функційного унітерма операції секвентування.

```
vcS()=

    usi (dc ∈@DraCon=zcS.RenOpe())= po ∈@Poi(x+sS.Wid/2, y+sS.Hei*0.2)
                                    ;
                                    DracSeW (mf, dc, po)
                                     ;
                                     dc.DraLin(sp, new Poi(x+sS.Wid*0.5, y+sS.Hei*0.2),
                                                   new Poi(x+sS.Wid, y+sS.Hei*0.2))
                                      ;
                                      dc.DraLin(sp, new Poi(x+sS.Wid*0.5, y+sS.Hei*0.8),
                                                    new Poi(x+sS.Wid, y+sS.Hei*0.8))
                                     ;
                                      dc.DraLin(sp, new Poi(x+sS.Wid*0.3, y+sS.Hei),
                                                    new Poi(x+sS.Wid*08, y))
                                      ;
                                       mf.cDra.AdVis(zcS)
     ;
     x=x+sS.Wid+f.Hei/2
      ;
      cond..Dra(mf, f, bb, gb, sp, dp, x, y, mx, my);  y=y+sS.Hei ;  (cond≠$) - ?
        ;
        y=y+cond.Hei; y=y+sS.Hei; (sS.Hei<cond.hei)-?
       ;
       x=x - sS.Wid - f.Hei/2
        ;
        y = y + 4
         ;
         t.Dra(mf, f, bb, gb, sp, dp, x, y, mx, my); mf.zn=$; (t≠$) - ?
```

(5)

```
pu over cXML(xmlD ∈@XmlDoc, n ∈@XmlEl) =

                                    nEl∈@XmlEl
                                    ;
                                    nAt∈@XmlAt
                                     ;
                                     nEl = xmlD.CreEl("cs")
                                      ;
                                      nAt = xmlD.CreAt("ori")
                                            ;
                                            nAt.Val = "hor"; nAt.Val = "ver"; (ori=Ori.Hor) - ?
                                     ;
                                      nEl.Atr.Ap(nAt)
                                     ;
                                      n.ApChi(nEl)
                                       ;
                                       cond.cXML(xmlD, nEl); *; (cond≠$) - ?
                                         ;
                                         t.cXML(xmlD, nEl); *; (t≠$) - ? ,
```

(6)

5. Фрагменти програмної реалізації

5.1. Програмна реалізація моделі формування операції циклічного секвентування

Написаний мовою об'єктного програмування **C#** [7, 8] код інформаційної технології рисування операції з горизонтальною (***hcS()***) і вертикальною (***vcS()***) орієнтаціями має такий вигляд:

*//Програма функційного унфтерма **hcS()**-рисування горизонтального циклічного секвентування*

```
DrawingVisual dv_3 = new DrawingVisual();
if (orientation == Orientation.Horizontal)
 {
  if ((mf.znyszczyty!="InsertReplace")&& (mf.znyszczyty != "InsertWithChild")
                ||(mf.znyszczyty == null))
   {
    using (DrawingContext g = dv_3.RenderOpen())
     {
      po = new Point(x+symbolSize.Width/2, y+symbolSize.Height*0.2);
      DrawSek_Goryzontalne(mf, g, po);
      g.DrawLine(sp, new Point((x+symbolSize.Width*0.5,y+symbolSize.Height*0.2)),
                   new Point(x+symbolSize.Width, y+symbolSize.Height*0.2));
      g.DrawLine(sp, new Point((x+symbolSize.Width*0.5,y+symbolSize.Height*0.8)),
                    new Point(x+symbolSize.Width,y+symbolSize.Height*0.8));
      g.DrawLine(sp, new Point((x+symbolSize.Width*0.3), y+symbolSize.Height),
                       new Point(x + symbolSize.Width * 0.8), y));
      mf.canwasDraw.AddVisual(dv_3);
     }
       x += symbolSize.Width;
       x += f.Height / 2);

       if (condition != null)
         {
          condition.Draw(mf, f, bb, gb, sp, dp, x, y, marginX, marginY);
          x += condition.width;
          }
       x +=f.Height/2;
       if(term!=null) term.Draw(mf, f, bb, gb, sp, dp, x, y, marginX, marginY);
    }
   else
      {
       mf.znyszczyty = null;
      }
 }
```

*//Програма функційного унфтерма **vcS()**-рисування вертикального циклічного секвентування*

```
    else
 {
  if((mf.znyszczyty!="InsertReplace")&&(mf.znyszczyty != "InsertWithChild")
                ||(mf.znyszczyty == null))
   {
    using (DrawingContext g = dv_3.RenderOpen())
     {
      po = new Point(x+symbolSize.Width/2,y+symbolSize.Height*0.2);
```

```
        DrawSek_Wertykalne(mf, g, po);
        g.DrawLine(sp, new Point((x+symbolSize.Width*0.5,y+symbolSize.Height*0.2),
                  new Point (x+symbolSize.Width,y+symbolSize.Height*0.2));
        g.DrawLine(sp, new Point((x+symbolSize.Width*0.5,y+symbolSize.Height*0.8),
                    new Point(x+symbolSize.Width, y+symbolSize.Height*0.8));
        g.DrawLine(sp, new Point((x+symbolSize.Width*0.3,y+ symbolSize.Height),
                              new Point(x+symbolSize.Width * 0.8, y));
          mf.canwasDraw.AddVisual(dv_3);
         }
        x += symbolSize.Width;
        x += (f.Height / 2);
        if (condition != null)
         {
          condition.Draw(mf, f, bb, gb, sp, dp, x, y, marginX, marginY);
          if(symbolSize.Height<condition.height) y += condition.height;
          else y += symbolSize.Height;
          }
        else
           {
            y += symbolSize.Height;
           }
        x -= f.Height / 2);
        x -= symbolSize.Width;
        y += 4;
        if(term!=null) term.Draw(mf, f, bb, gb, sp, dp, x, y, marginX, marginY);
        }
     else
        {
         mf.znyszczyty = null;
        }
    }
    mf.znyszczyty = null;
}
```

5.2. Програмна реалізація моделі формування XML-опису операції

Написаний мовою **C#** код є таким:

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

    newElement = xmlDoc.CreateElement("cyclic-sequence");

    newAttribute = xmlDoc.CreateAttribute("orientation");

    if (orientation == Orientation.Horizontal)
       newAttribute.Value = "horizontal";
    else
       newAttribute.Value = "vertical";

    newElement.Attributes.Append(newAttribute);
```

```
    node.AppendChild(newElement);
    if (condition != null)
      condition.CreateXML(xmlDoc, newElement);
    if (term != null)
      term.CreateXML(xmlDoc, newElement);
  }
```

Результати дослідження

1. Засобами розширеної алгебри алгоритмів побудовано математичні моделі декомпозиції підсистеми циклічного секвентування на функційні унітерми і математичні моделі інформаційних технологій обчислення розмірів, вибору, видалення, формування горизонтальної й вертикальної орієнтації та формування *XML* - опису операції циклічного секвентування.

2. Створені математичні моделі програмно реалізовано й апробовано.

Висновки

1. Програмна реалізація математичних моделей інформаційних технологій забезпечує автоматизоване виконання процесів обчислення розмірів, вибору, видалення, формування горизонтальної й вертикальної орієнтації та формування *XML*-опису операції циклічного секвентування.

2. Створені моделі і їхня програмна реалізація можуть бути використані для реалізації процесів автоматичної оптимізації формул алгебри алгоритмів і формування програмних кодів з формул алгоритмів.

Література

1. Овсяк, В.К. Засоби еквівалентних перетворень алгоритмів інформаційно-технологічних систем [Текст] / В.К. Овсяк //Доповіді Національної академії наук України. – № 9, 1996. – С. 83 – 89.
2. Owsiak, W. Rozszerzenie algebry algorytmów / W. Owsiak, A. Owsiak //Pomiary, automatyka, kontrola. – № 2, 2010. – S. 184–188.
3. Бритковський, В.М. Моделювання редактора формул секвенційних алгоритмів: автореф. дис. ... канд. техн. наук: спец. 01.05.02 “Математичне моделювання та обчислювальні методи” [Текст] / В.М. Бритковський. – Львів, 2003. – 18 с.
4. Василюк, А.С. Підвищення ефективності математичного і програмного забезпечення редактора формул алгоритмів: автореф. дис. ... канд. техн. наук: спец. 01.05.02 “Математичне та програмне забезпечення обчислювальних машин і систем” [Текст] / А.С. Василюк. – Львів, 2008. – 20 с.
5. Детловс, В.К. Нормальные алгоритмы и рекурсивные функции [Текст] / В.К. Детловс //Докл. АН СССР. – 1953. – 90, № 4. – С.723–725.
6. Turing, A.M. On computable numbers, with an application to the Entscheidungsproblem / Turing A.M. // *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).
7. Petzold, C. Programowanie Windows w języku C#. – Warszawa: Rm, 2002. – 1161 s.
8. Мак-Дональд, М. WPF: Windows Prezentation Foundation в NET 3.5 с примерами на C# 2008. Для профессионалов [Текст] / Мак-Дональд М.; пер. с англ. Я.П. Волковой, Д.Я. Иваненко, Н.А. Мухана. – Москва, Санкт-Петербург, Киев: И.Д.Вильямс, 2008. – 928 с.
9. 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.
10. Овсяк, О. Класи інформаційної системи генерування коду [Текст] / О. Овсяк //Вісник Тернопільського державного технічного університету, 2010. – № 3. – С.106–110.

Отримано 23.08.2011