Science, Services and Agents on the World Wide Web 2006, Vol. 4, No. 1. P. 14–28. 2. G. Xu, Ya. Zhang, L. Li. Web Mining and Social Networking: Techniques and Applications. –Springer: Web Information Systems Engineering and Internet Technologies Book Series, 2011. – 210 p. 3. Resource Description Framework: Overview. [Електронний ресурс]. – Режим доступу: http://www.w3.org/RDF/. 4. OWL Web Ontology Language: Overview. [Електронний ресурс]. – Режим доступу: http://www.w3.org/TR/owl-features/. 5. D.F. Specht Probabilistic neural network. Neural Networks 1990, 3. P. 109–118. 6. Бодянский Е.В., Шубкина О.В. Семантическое аннотирование текстовых документов с использованием модифицированной вероятностной нейронной сети // Системные технологи: Региональный межвузовский сборник научных трудов. – Днепропетровск, 2011. – Вып. 4 (75). – С. 48–55. 7. L.H. Tsoukalas, R.E. Uhrig. Fuzzy and Neural Approaches in Engineering. – N.Y.: John Willey and Sons Inc., 1997. – 587 p. 8. D.F. Spech. A general regression neural network. IEEE Trans. on Neural Networks 1991, 2. P. 568–576. 9. O. Nelles. Nonlinear System Identification. – Berlin: Springer, 2001. – 785 p. 10. D.R. Zahirniak, R. Chapman, S.K. Rogers, B.W. Suter, M. Kabriski, V. Pyatti. Pattern recognition using radial basis function network. Proc: 6-th Ann. Aerospace Application of AI Conf. Dayton, OH, 1990, Pp. 249–260. 11. R. Callan. The Essence of Neural Networks. – London: Prentice Hall Europe, 1999. – 248 p.

УДК 004.4’232

О. Овсяк
Київський національний університет культури і мистецтв,
Українська академія друкарства

МОДЕЛЬ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ ОПРАЦЮВАННЯ ФОРМУЛ АЛГОРИТМІВ

Модель інформаційної технології опрацювання формул алгоритмів описано розширеною алгеброю алгоритмів у вигляді рекурентно-декомпозиційної моделі. За ознакою функціонального призначення модель декомпоновано на субмоделі. Описано візуальну і функціональну підмоделі інформаційної технології. Наведено фрагмент програмної реалізації моделі.

Ключові слова: модель, інформаційна технологія, декомпозиція, підмодель, секвентування, елімінування, паралелення, функційний унітерм, унітерм.

Described the extended algebra algorithms recurrent – decomposition model of information technology for synthesis and processing algorithms formulas. Model by functional appointment of the decomposing at sub models. Described visual and functional sub models information technology. An piece of software implementation model.

Keywords: model, information technology, decomposition, sub models, sequention, elimination, parallelization, function uniterm, uniterm.

Вступ і формулювання задачі

Модель декомпозиції комп’ютерної системи генерування програмного коду з формул алгоритмів [1–3] описана у статті [4]. Але у цій моделі не подано інформації про наслідування підмоделей і декомпозиції підмоделей на графічну та функціональну субмоделі. Графічна субмодель реалізується у вигляді графічних вікон, які використовуються для отримання інформації, вибору і задання параметрів, введення і виведення графічно-текстових даних тощо. Тоді як функціональна підмодель призначена для опису функціонування. Графічна і функціональна субмоделі мають бути поєднані між собою. Наприклад, таке поєднання субмоделей може бути виконано через вибір функційних унітермів, який здійснюється при виконанні певних умов.

У згаданій статті [4] також не описано модель інформаційної технології опрацювання формул алгоритмів. Наявність такої моделі є доцільною з багатьох поглядів. Перш за все буде отримано її точний математичний опис. По-друге, буде можливо здійснити її якісне і кількісне оцінювання, виявити недоліки і створити передумови їхнього подолання.

Для розв'язання задач побудови субмоделей і моделі інформаційної технології опрацювання формул алгоритмів доцільно застосувати рекурентно-декомпозиційну методологію, яка полягає у використанні методів декомпозиції і алгебри алгоритмів [1–3]. Обґрунтованість такого розв'язання зумовлена потребою отримання математичної моделі інформаційної технології синтезу та опрацювання формул алгоритмів.

Декомпозиційна модель інформаційної технології опрацювання формул алгоритмів

Модель за функціональним призначенням декомпозуємо на підмоделі кореневу (*@MF*), оптимізації формул алгоритмів (*@OA*), бази алгоритмів (*@DC*), унітерму (*@U*), терму (*@T*), візуалізації (*@V*), контексту (*@MD* і *@MC*), генерування коду (*@GC*), операцій секвентування (*@S*), елімінування (*@E*), паралелення (*@P*), циклічних секвентування (*@CS*), елімінування (*@CE*) і паралелення (*@CP*), а також задання параметрів унітермів (*@UG*), заміни (*@IF*), видалення (*@DF*), шрифтів (*@FF*) та операцій секвентування (*@SF*), елімінування (*@EF*), паралелення (*@PF*), циклічних секвентування (*@CSF*), елімінування (*@CEF*), паралелення (*@CPF*) і доступу до бази алгоритмів (*@DCF*) й зв'язку з операційним середовищем (*@Ap*). Підмоделі кореневa, зв'язку, візуалізації, задання параметрів унітермів, заміни, видалення, шрифтів та операцій і доступу до бази алгоритмів поділені на графічну (*@%MF, @%Ap, @%V, @%UG, @%IF, @%ICF, @%DF, @%FF, @%SF, @%EF, @%PF, @%CSF, @%CEF, @%CPF, @%DCF*) й функціональну (*@MF, @UG, @IF, @ICF, @DF, @FF, @SF, @EF, @PF, @CSF, @CEF, @CPF, @DCF*) субмоделі, відповідно. Декомпозиційна модель іформаційної технології описується формулою:

де *@* – ідентифікатор підмоделі; *:* – ознака наслідування моделі, яка записана після двох крапок; *@%* – ідентифікатор графічної підмоделі; __ – знак підкреслення моделей з відомими реалізаціями; *@Win* і *@Can* – моделі, які реалізуються відомими класами `Window` і `Canvas` [5, 6].

У графічних субмоделях задаються потрібні параметри операцій (горизонтальна чи вертикальна орієнтація, кома чи крапка з комою як розділювачі унітермів) створюваних і редагованих формул алгоритмів.

Коренева підмодель утворена графічною і функціональною субмоделями. Коренева графічна субмодель описує створення кореневого вікна системи і задання вибору кореневою функційною субмоделлю усіх інших підмоделей і субмоделей.

Підмодель терм абстрактна, має змінні та абстрактні й конкретні функційні унітерми і є складовою всіх операційних підмоделей, якими абстрактні функційні унітерми, з врахуванням

особливостей кожної операції розширеної алгебри алгоритмів, замінюються предметними унітермами. Моделі абстрактних функційних унітермів зорієнтовані на рисування унітермів і знаків операцій алгебри алгоритмів, селекцію і деселекцію унітермів і формул алгоритмів, обчислення розмірів унітермів і знаків операцій, створення XML-опису формул створюваних алгоритмів. Моделями предметних унітермів описуються аналіз коректності XML-опису формул алгоритмів з метою перетворення до графічного вигляду, розпізнавання типу описаної операції, заміни унітерму операціями алгебри алгоритмів і вибору терма.

На функціональне призначення усіх решти підмоделей вказують їхні назви. Наприклад, підмодель секвентування описує формування операції секвентування.

Технологічний процес починається запуском в операційній системі з подальшим виходом на кореневу підсистему, якою починається і закінчується функціонування усіх інших підсистем та субсистем.

Візуальна модель графічного вікна

Модель інформаційної технології, з метою зменшення складності її створення, декомпозуємо на функційні унітерми. Модель декомпозиції описується такою формулою:

$$@\%Mf_0{:}\underline{Win}=\left[\begin{array}{l} Ti=\overline{C_0;\ C_1;\ C_2;\ C_3;\ C_4}\\ ;\\ M=\overline{M_0;\ M_1;\ M_2;\ M_3;\ M_4;\ M_5}\\ ;\\ \underline{K}=\overline{C_5;\ C_6;\ K_s;\ K_e;\ K_p;\ K_{cs};\ K_{ce};\ K_{cp};\ C_7;\ K_r;\ K_d;\ K_u;\ K_b;\ C_8;\ F_t;\ F_s;K_g},\\ S_v=\overline{S_v^g;\ S_v^n};\\ D \end{array}\right.$$

де @ – ідентифікатор підсистеми; % – ідентифікатор опису графічних елементів; Mf_0 – головне графічне вікно комп'ютерної системи; *:* – ідентифікатор наслідування; *Win* –підсистема з типовими можливостями підтримки графічних вікон, яка реалізується відомим класом `Window` [5, 6]; *Ti* – рядок титулу вікна; *M* – рядок менню *K* – рядок кнопок з умовними графічними позначеннями; S_v – скролінг вікна; *D* – поле для набору і редагування формул алгоритмів; C_0 є позначенням іконки головного вікна комп'ютерної системи; C_1 = "***Генератор формул алгоритмів***"; C_2, C_3 і C_4 – типові іконки згортання, зменшення/збільшення і закриття вікна, відповідно [5, 6]; M_0, M_1, M_3, M_4 та M_5 – унітерми, які описуються формулами:

$$M_0=\%M(\text{"Файл"}),\ \overline{C_c(\%M(\text{"Файл"}))=\left[\begin{array}{l}\%Mi(\text{"Новий"});\ *;\ u_0\text{-?}\\ \%Mi(\text{"Відкрити"});\\ \%Mi(\text{"Записати"});\\ \%Mi(\text{"Записати як"});\\ C=\_\_\_\_\_;\\ \%Mi(\text{"Закінчити"}),\end{array}\right.}$$

де u_0 – умова вибору унітерма M_0, $C_c()$ – функційний унітерм, який цоком лівою клавішею маніпулятора "миш" графічного елемента, на якому встановлено курсор (подія), описує вибір функційного унітерма опрацювання події; $C_c(\%Mi(\text{"Новий"}))=Mf.n\_C();\ *;\ u_n\text{-?}$ – вибір функційного унітерма опрацювання події на графічному елементі *%Mi("Новий");*

$$\overline{C_c(\%Mi(\text{"Відкрити"}))=Mf.o\_C();\ *;\ u_v\text{-?}};\ \overline{C_c(\%Mi(\text{"Записати"}))=Mf.za\_C();\ *;\ u_z\text{-?}}$$
$$\overline{C_c(\%Mi(\text{"Записати як"}))=Mf.zJ\_C();*;\ u_j\text{-?}};\ \overline{C_c(\%Mi(\text{"Закінчити"}))=Mf.z\_C();\ *;\ u_{za}\text{-?}};\ C_c(C))=C;$$

$$M_1=\%M(\text{"Операції"}),\ \overline{C_c(\%M(\text{"Операції"}))=\left[\begin{array}{l}\%Mi(\text{"Секвентування"});\ *;\ u_1\text{-?}\\ \%Mi(\text{"Елімінування"});\\ \%Mi(\text{"Паралелення"});\\ C;\\ \%Mi(\text{"Циклічне секвентування"});\\ \%Mi(\text{"Циклічне Елімінування"});\\ \%Mi(\text{"Циклічне Паралелення"}),\end{array}\right.}$$

C_c(%Mi(“Секвентування”)) = Mf.s_C();*; u_s-?; C_c(%Mi(“Елімінування”)) = Mf.e_C(); *; u_e-?;
C_c(%Mi(“Паралелення”))=Mf.zR_C();*;u_p-?; C_c(%Mi(“Циклічне секвен-вання”))=Mf.cS_C();*;u_{cs}-?
C_c(%Mi(“Ци-не Елі-ання”)=Mf.cE_C();*;u_{ce}-?; C_c(%Mi(“Ци-не Паралелення”))= Mf.cZ_C();*;u_{cp}-?

M_2=%M(“Дії”), C_c(%M(“Дії”))= %Mi(“Замінити”); *; u_2-?
%Mi(“Видалити”)=Mf.uD_C();
%M(“Властивості”)=Mf.WC();
C;
%M(“Експорт/Імпорт”),

C_c(%Mi(“Замінити”))= Mf.za_C();*; u_{zam}-?; C_c(%Mi(“Видалити”)) = Mf.uD_C(); *; u_{vy}-?;
C_c(%M(“Властивості”)) = Mf.WC(); *; u_{vs}-?; C_c(%M(“Експорт/Імпорт”)) = Mf.eI_C(); *; u_{ei}-?;

M_3=%M(“Налаштування”), C_c(%M(“Налаштування”)) = %Mi(“Кегель”); *; u_3-?
C;
%Mi(“База алгоритмів”),

C_c(%Mi(“Кегель”))= Mf.cZ_C(); *; u_k-? ; C_c(%Mi(“База алгоритмів”)) = Mf.bD_C(); *; u_b-?;

M_4= %M(“Програма”), C_c(%Mi(“Згенерувати код”))= Mf.zK_C(); *; u_4-?;

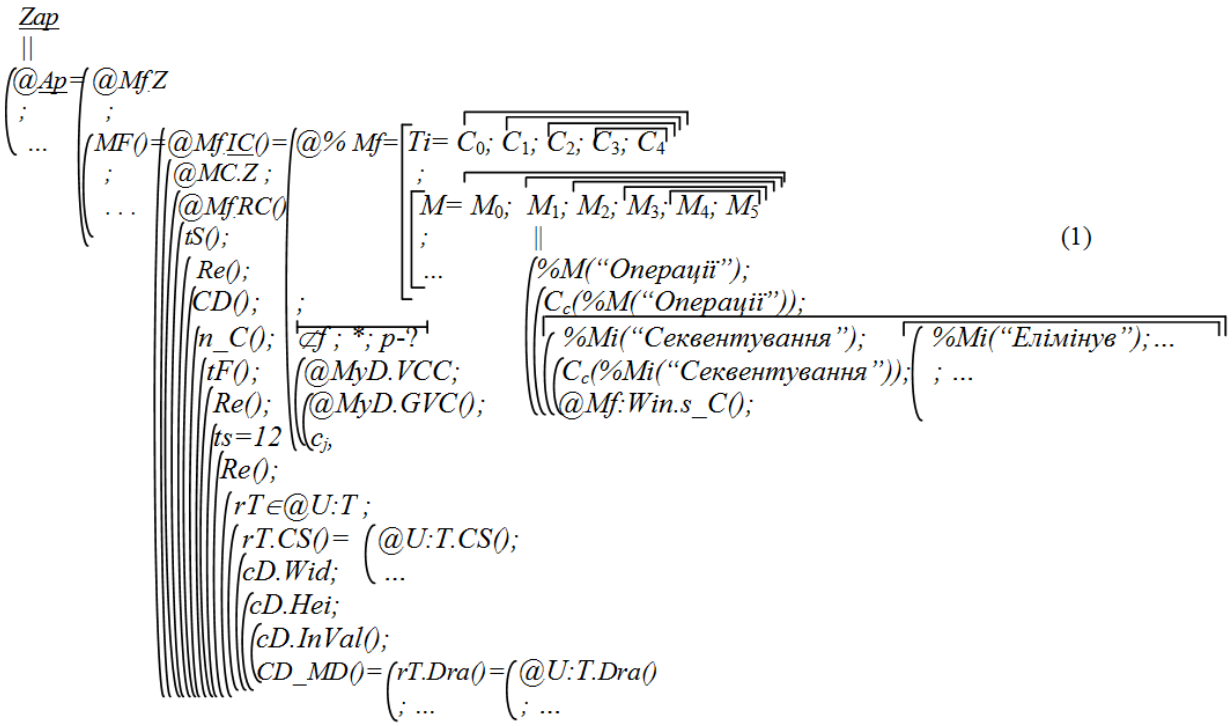
M_5=%M(“Допомога”), C_c(%Mi(“Інформація”)) = Mf.i_C(); *; u_5-?,

у яких u_0, u_n, u_v, u_z, u_1, u_2, u_3, u_4, u_5, u_j, u_{za}, u_s, u_e, u_p, u_{cs}, u_{ce}, u_{cp}, u_{zam}, u_{vy}, u_{vs}, u_{ei}, u_k, u_b, u_b,– умовні унітерми вибору графічних елементів “*Файл*”, “*Новий*”, ”*Відкрити*”, ”*Записати*”, ”*Операції*”, ”*Дії*”, ”*Налаштування*”, “*Програма*”, “*Допомога*”, “*Записати як*”, “*Закінчити*”, “*Секвентування*”, “*Елімінування*”, “*Паралелення*”, “*Циклічне секвентування*”, “*Циклічне Елімінування*”, “*Циклічне Паралелення*”, “*Замінити*”, “*Видалити*”, “*Властивості*”, “*Експорт/Імпорт*”, “*Кегель*”, “*База алгоритмів*”, ”” *%M()* – графічний елемент, який реалізується відомим елементом `Menu` [5, 6]; *%Mi()* –графічний елемент, який реалізується відомим елементом `MenuItem` [5, 6]; *%M(“Файл”), %M(“Операції”), %M(“Дії”), %M(“Налаштування”), %M(“Програма”), %M(“Допомога”)* – опис команд меню *“Файл”*, *“Операції”*, *“Дії”*, *“Налаштування”*, *“Програма”*, *“Допомога”*; *%Mi(“Новий”), %Mi(“Відкрити”), %Mi(“Записати”), %Mi(“Записати як”), %Mi(“Закінчити”), %Mi(“Секвентування”), %Mi(“Елімінування”), %Mi(“Паралелення”), %Mi(“Циклічне секвентування”), %Mi(“Циклічне Елімінування”), %Mi(“Циклічне Паралелення”), %Mi(“Замінити”), %Mi(“Видалити”), %M(“Властивості”), %M(“Експорт/Імпорт”), %Mi(“Кегель”), %Mi(“База алгоритмів”), %Mi(“Згенерувати код”), %Mi(“Інформація”)* – опис підменю *“Новий”*, *“Відкрити”*, *“Записати”*, *“Записати як”*, *“Закінчити”*, *“Секвентування”*, *“Елімінування”*, *“Паралелення”*, *“Циклічне секвентування”*, *“Циклічне Елімінування”*, *“Циклічне Паралелення”*, *“Замінити”*, *“Видалити”*, *“Властивості”*, *“Експорт/Імпорт”*, *“Кегель”*, *“База алгоритмів”*, *“Згенерувати код”*, *“Інформація”*; *Mf.n_C()* – цоком правою клавішею “мишки” на *%M(“Файл”)* і *%Mi(“Новий”)* відбувається вибір курсором з головного вікна системи кореневої підсистеми (*Mf*) функційного унітерма *n_C()*, призначеного для створення нового поля *D* набору і редагування формули алгоритму; *Mf.o_C()* – функційним унітермом *o_C()* відкриття у головному вікні системи вибраного XML-файла опису формули алгоритму; *Mf.za_C()* – запису XML-опису формули алгоритму в пам'ять комп’ютера у вигляді файла з розширенням *.xml*; *Mf.zJ_C()* – запису XML-опису формули алгоритму в пам'ять комп’ютера у вигляді файла з розширенням *.xml* та задаваною назвою файла; *C* – типовий графічний елемент-розділювач; *Mf.z_C()* – завершення функціонування комп'ютерної системи; *Mf.s_C()* – утворення на *D* операції абстрактного

секвентування; *Mf.e_C()* – операції абстрактного елімінування; *Mf.zR_C()* – операції абстрактного паралелення; *Mf.cS_C()* – операції абстрактного циклічного секвентування; *Mf.cE_C()* – операції абстрактного циклічного елімінування; *Mf.cZ_C()* – операції абстрактного циклічного паралелення; *Mf.za_C()* – заміни попередньо вибраного унітерма формули алгоритму; *Mf.uD_C()* – видалення попередньо вибраного унітерма формули алгоритму; *Mf.WC()* – отримання властивостей попередньо вибраного унітерма формули алгоритму; *Mf.eI_C()* – експорту/імпорту попередньо вибраного фрагменту або всієї формули алгоритму; *Mf.cZ_C()* – задання налаштування кеглю шрифта; *Mf.bD_C()* – задання налаштування бази алгоритмів; *Mf.zK_C()* – генерування програмного коду з формули алгоритму; *Mf.i_C()* – надання допомоги користувачам комп'ютерної системи; C_5 – стандартний розділювач *C*; C_6 – поле з написом "**Операції**"; K_s – кнопка з графікою операції секвентування і вибору з базової підсистеми функційного унітерма формування операції секвентування *Mf.se_C()*; K_e – операції елімінування і вибору з базової підсистеми функційного унітерма формування операції елімінування *Mf.el_C()*; K_p – операції паралелення і вибору з базової підсистеми функційного унітерма формування операції паралелення *Mf.pa_C()*; K_{cs} – операції циклічного секвентування і вибору з базової підсистеми функційного унітерма формування операції циклічного секвентування *Mf.cse_C()*; K_{ce} – операції циклічного елімінування і вибору з базової підсистеми функційного унітерма формування операції циклічного елімінування *Mf.cel_C()*; K_{cp} – операції циклічного паралелення і вибору з базової підсистеми функційного унітерма формування операції циклічного паралелення *Mf.cpa_C()*; C_7 – поле з написом "**Дії**"; K_r – заміни вибраного унітерму *Mf.r_C()*; K_d – видалення вибраного унітерма *Mf.d_C()*; K_u – формування унітерма *Mf.u_C()*; K_b – доступу до бази алгоритмів *Mf.ei_C()*; C_8 – поле з написом "**Шрифт**"; F_t – вибору шрифта з використанням функційного унітерма *tF()*; F_s – задання розміру кегля з використанням функційного унітерма *tS()*; K_g – вибору функційного унітерма *Gk()* генерування програмного коду з формули алгоритму; S_v^g – горизонтальна складова прокрутки; S_v^v – вертикальна складова прокрутки.

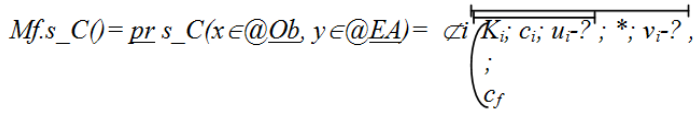
Функціональна модель графічного вікна

Процес починається запуском *Zap* (див. формулу (1)), який виконується її користувачем у середовищі Microsoft Visual Studio .NET. Зв'язок між середовищем і прикладною технологією описується підмоделлю *@Ap*, яка на платформі Microsoft Visual Studio .NET реалізується відомим класом Application [5, 6]. Підмодель *@Ap* описує і вибір підмоделі *@Mf*, якою задаються унітерми-змінні (*Z*) і функційний унітерми *MF()* та інші, які з огляду наявності значної кількості функційних унітермів і обмеження обсягу статті, не наведені у формулі (1). Для ініціалізації графічних елементів (*@% Mf*) інтерфейсу користувача функційним унітермом *MF()* вибирається функційний унітерм *IC()*, який реалізується відомою процедурою `InitializeComponent()`[5, 6]. Після ініціалізації графічних елементів кореневого вікна встановлюються значення змінних підмоделі відображення на екрані монітора контексту кореневого вікна (*@MC.Z*). *@Mf.RC()* – зчитування параметрів доступу до бази алгоритмів. Функційним унітермом *tS()* задається початковий шрифт для опису унітермів формул алгоритмів. Обчислення початкових координат формули алгоритму описується функцій ним унітермом *Re()*. Встановлення наявності унітерму описується у функційному унітермі *CD()* (скорочений запис *CD_MD()*). Запис поточних змін описується *m_C()*. *tF()* – функційний унітерм опрацювання розміру кегля. Нове обчислення розмірів унітерма (*Re()*). Піся чого задається початковий розмір кегля *ts=12* і описується перерисовування екрану (*Re()*). *rT Î @U:T* – створення змінної *rT* типу підмоделі *@U*, яка наслідує підмодель *T*. Вибір функційного унітерма (*rT.CS()*) обчислення розмірів початкового унітерма, яке описується унітермом *@U:T.CS()*. Обчислення довжини (*cD.Wid*) і ширини (*cD.Hei*) поля синтезу формул алгоритмів (*cD*). *cD.InVal()* – візуалізація робочого поля. Зозпізнавання наявності унітерма (*CD_MD()*) і вибір підмоделі (*rT.Dra()*) опису рисування унітерму (*@U:T.Dra()*).

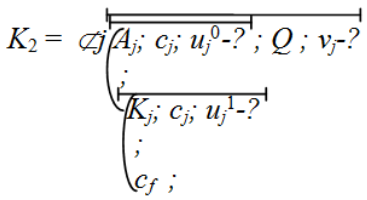


(1)

Загальна модель опрацювання операції секвентування *(Mf.s_C())* кореневою підсистемою описується формулою



де *pr* – типовий специфікатор доступу до функційного унітерма, який на платформі Microsoft Visual Studio .NET реалізується ключовим словом `private` [5, 5]; *x* та *y* – унітерми-змінні; *@Ob* та *@EA* – типові підсистеми, які реалізуються відомими класами `object` та `EventArgs` [5, 6]; – змінна циклу ($i \hat{I}$ 0; 1; 2) ; K_0 = *K(“Зазначте місце вставляння секвентування. Помилка”)* – функційний унітерм висвітлення на екрані комп’ютера повідомлення, який реалізується відомою процедурою `MessageBox.Show()` [5, 6]; K_1 = *K(“Виберіть унітерм для його заміни секвентуванням. Помилка”)*;



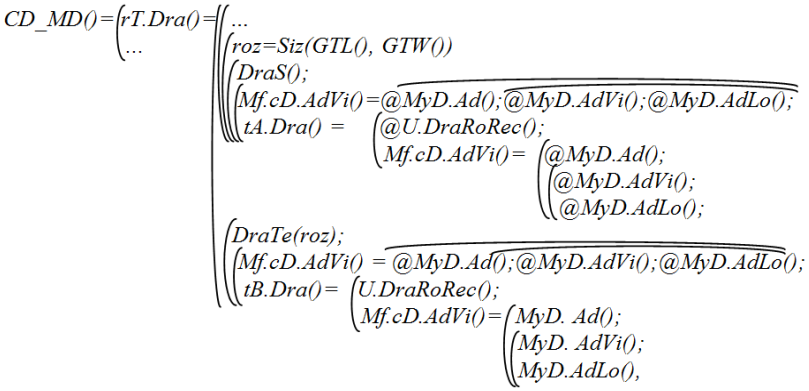
c_i – ознака повернення у цикл за змінною *i*; u_i – умова елімінування (u_0 є умовою *(c_cD=0)-?* – перевірка відсутності події на *D*; u_1 є умовою *(sT=$)-?* – перевірки відсутності вибору унітерма; u_2 є умовою *(sT.par≠$)-?* – перевірка наявності умовного унітерму; (v_i-?) – умова *(i<3)-?*; c_f – повернення у цикл за змінною *f*; *j* – змінна циклу (*j* $\hat{I}$ 0; 1; 2; 3); A_j – творення унітерму-змінної та приписування їй значення (A_0 то *el* $\hat{I}$ *@E=sT.par*, A_1 то *cs* $\hat{I}$ *@cS=sT.par*, A_2 то *ce* $\hat{I}$ *@cE=sT.par*, A_3 то *cp* $\hat{I}$ *@cP=sT.par*,); c_j – повернення у цикл за змінною *j*; u_j^0 – умовний унітерм (u_0^0 то *(sT.par=E)-?*, u_1^0 то *(sT.par=cS)-?*, u_2^0 то *(sT.par=cE)-?*, u_3^0 то *(sT.par=cP)-?*); v_j – умова (*j<4*); K_0 = *K(“Недопустимо умову елімінування заміняти операцією”)*; K_1 = *K(“Недопустимо умову циклічного секвентування заміняти операцією”)*; K_2 = *K(“Недопустимо умову циклічного елімінування заміняти операцією”)*; K_3 = *K(“Недопустимо умову циклічного паралелення заміняти операцією”)*; u_j^1-? – умовний унітерм (u_0^1 то *(el.cond=sT)-?*, u_1^1 то *(cs.cond=sT)-?*, u_2^1 то

(ce.cond=sT)-?, u_3^1 то *(cp.cond=sT)-?*); *Q* – описується формулою (1) у якій: *if Î @IF* – створення змінного унітерму *if* типу підсистеми *@iF* (InsertForm) [4]; *if.ShDia()* – відкриття графічного вікна підсистеми *@IF*; *((((sT.GT().TS().Eq("unit")=0)|(sT.val≠$))-?* – дві перевірки наявності унітерма (*sT.val≠$* – наявність не порожнього (*$*) значення змінної *val* або *(sT.GT().TS().Eq("unit")=0)* – наявність значення *"unit"*змінної *sT*, яке вибирається функційними унітермами вибору типу (*GT()*), перетворення на текст (*TS()*) і порівняння (*Eq()*), які реалізуються відомими процедурами `GetType()`, `ToString()` і `Equals()` [5, 6], відповідно; *(if.DiaRez=0)-?)* – перевірка відсутності вибору будь-якого із режимів заміни унітермів; *sf Î @sF* – створення унітерму-змінної типу підсистеми SequenceForm (*@sF*) [ ]; *s Î @S* – створення змінної типу підсистеми Sequence (*@S*); *s.sep=sf.Sep* – приписування змінній *s.sep* значення розділювача унітермів, заданого у графічному вікні підсистеми *@sF*; *s.ori=sf.Ori* – приписування змінній значення орієнтації, заданого у графічному вікні підсистеми *@sF*; *rT= T.Rt(rT, sT, s)* – приписування змінній *rT* значення, яке вибрано функційним унітермом *Rt()* підсистеми *T* [4]; *s.tA=@U* – приписування змінній *s.tA* значення, яке формується у підсистемі *U* [4] функційним унітермом *U()*; *s.tA.par=s* – приписування змінній *s.tA.par* значення *s*; *s.tA=sT* – приписування змінній *s.tA* значення змінної *sT*; (*(if.Met=iF.IM.IF)-?)* – перевірка наявності збігу значення змінної *if.Met* зі значенням *IF*, заданим у графічному вікні підсистеми *iF*; (*(if.Met=iF.IM.IR)-?*) – перевірка наявності збігу значення змінної *if.Met* зі значенням *IR*, заданим у графічному вікні підсистеми *iF*;

```
Q=(if∈@iF:W;
   if.ShDia();*;(((sT.GT().TS().Eq("unit")=0)|(sT.val≠$))-?
   *;cf;(if.DiaRez=0)-?
   (sf∈@sF;
    sf.ShDia();
    s∈@S;cf;(sf.DiaRez=1)-?
     ;
    s.sep=sf.Sep;
    s.ori=sf.Ori;
    rT= T.Rt(rT, sT, s);
    (s.tA=@U;                 s.tA=sT;                 s.tA=@U;  (if.Met=iF.IM.IF)-?; (if.Met=iF.IM.IR)-? ;   (1)
       ;                      ;                        ;
     (s.tA.par=s;             (s.tA.par=s;             (s.tA.par=s;
      s.tB=@U;                 s.tB=@U;                 s.tB=sT;
       (s.tB.par=s;             (s.tB.par=s;             (s.tB.par=s;
        zN="IR";                 zN="IF";                 zN="IS";
        oC=0                     oC=0                     oC=0
    rT.Des(thi);
    sT=s;
    sT.sel=1;
    dv∈@DV;
    rT.Cs($, fon);
     cD.Wid=cD.Wid+rT.wid+mX;
     cD.Hei=cD.Hei+rT.Hei+mY;
     IV();
     mod=1;
     thi.Tit="ГенКод";
     rT.Des(thi)=@S:T.Des(); ...
     rT.CFC(thi, fon, xP, yP, mX, mY, scrol.AW, scrol.AH)=(@S:T.CFC();
     CD_MD(rT, $)=(@S:T.Dra();
                   ...
```

s.tB=@U – приписування змінній *s.tB* значення, яке формується у підсистемі *U* [4] функційним унітермом *U()*; *zN="IR"* – приписування змінній *zN* значення *"IR"*; *oC=0* – приписування змінній *zN* значення 0; *s.tA=sT* – приписування змінній *s.tA* значення *sT*; *s.tB.par=* – приписування змінній

s.tB.par значення *s*; *zN=”IF”* – приписування змінній *zN* значення *”IF”*; *rT.Des(thi)* – вибір з кореневої підсистеми (*thi*) функційного унітерма *Des(thi)* для опрацювання значення змінної *rT*; *sT=s* – приписування змінній *sT* значення змінної *s*; *sT.sel=1* – приписування змінній *sT.sel* значення 1; *dv Î @DV* – створення змінної *dv* типу підсистеми *DV*, яка реалізується відомим класом `DrawingVisual` [5, 6]; *rT.Cs($, fon)* – вибір функційного унітерма *Cs($, fon)* для обчислення розмірів унітерма-змінної *rT*; *cD.Wid=cD.Wid+rT.wid+mX* – обчислення довжини *Wid* графічного поля *cD*, враховуючи його наявну довжину *cD.Wid* і довжину унітерма-змінної *rT.wid* та відступ по вісі абсцис від краю форми *mX*; *cD.Hei=cD.Hei+rT.Hei+mY* – обчислення ширини *Hei* графічного поля *cD*; *IV()* – функцій ний унітерм нового рисування графічної області, який реалізується відомою процедурою `InvalidateVisual()` [5, 6]; *mod=1* – запис значення змінної, яким ідентифікується наявність виконаних змін; *thi.Tit=”ГенКод”* – заадання назви заголовку (*Tit*) кореневого (*thi*) вікна; *rT.CFC(thi, fon, xP,yP, mX, mY, scrol.AW, scroll.AH)* – вибір функційного унітерма для встановлення попадання координат курсора в область унітерма змінної *rT* (*thi* – ознака кореневого вікна, *xP* і *yP* – абсциса і ордината початку формування формули алгоритму на графічному полі, *mX* і *mY* – відступи від краю кореневої форми до початку графічого поля по осях абсцис і ординат, *scrol.AW* і *scrol.AH* – значення актуальних довжини (*AW*) і ширини (*AH*) змінної полів прокрутки (*scrol*); *CD_MD(rT, $)* – вибір функційного унітерма для опрацювання заданої у кореневому вікні операції алгебри алгоритмів, яка зберігається в унітермі *rT*, а сам функцій ний унітерм описується формулою:



у якій функційним унітермом *Dra()* вибираються функційні унітерми обчислення розмірів розділювача унітермів *Siz(GTL(), GTW())* операції секвентування, застосуванням функційних унітермів обчислення довжини *GTL()* і ширини *GTW()* знаку-розділювача унітермів; *DraS()*-рисування знаку операції секвентування; *Mf.cD.AdVi()* – долучення знаку операції секвентування до графічного елемента його візуалізації; *MyD.Ad()* – висвітлення знаку операції секвентування; *MyD.AdVi()* та *MyD.AdLo()* – віалізація та створення логічного номера графічного елемента; *tA.Dra()* – вибір функційного унітерма рисування унітерма *tA*; *@U.DraRoRec()* – рисування унітерма; *tB.Dra()* – вибір функційного унітерма рисування унітерма *tB*.

Фрагмент програмної реалізації інформаційної технології

Загальна модель опрацювання операції секвентування (*Mf.s_C()*) кореневої підсистеми описується кодом мови C# [5, 6]:

```
private void sequence_Click(object sender, EventArgs e)
        {
            if(cok_Na_canwasDraw == false)
            {
                MessageBox.Show("Зазначте місце встановлення об'єкта.", "Помилка",
                   MessageBoxButton.OK, MessageBoxImage.Exclamation);     return;
            }
            if (selectedTerm == null)
            {
```

```
            MessageBox.Show("Зазначте місце встановлення об'єкта.", "Помилка",
             MessageBoxButton.OK, MessageBoxImage.Exclamation);     return;
          }
          if (selectedTerm.parent != null)
          {
           if

(selectedTerm.parent.GetType().ToString().Equals("TermEdit.Elimination"))
              {
                Elimination elimi= (Elimination)selectedTerm.parent;
                  if (elimi/*nation*/.condition == selectedTerm)
                  {
                   MessageBox.Show("Заборонено вставляння об'єкта на місце умови
                    елімінування.", "Помилка",  MessageBoxButton.OK,
                     MessageBoxImage.Exclamation);       return;
                  }
              }
       if (selectedTerm.parent.GetType().ToString().Equals("TermEdit.CyclicSequence"))
              {
               CyclicSequence cyclicSequence = (CyclicSequence)selectedTerm.parent;

                  if (cyclicSequence.condition == selectedTerm)
                  {
                    MessageBox.Show("Заборонено вставляння об'єкта на місце умови
                     циклічного секвентування.", "Помилка", MessageBoxButton.OK,
                      MessageBoxImage.Exclamation);      return;
                  }
              }
     if
(selectedTerm.parent.GetType().ToString().Equals("TermEdit.CyclicElimination"))
              {
         CyclicElimination cyclicElimination = (CyclicElimination)selectedTerm.parent;

                  if (cyclicElimination.condition == selectedTerm)
                  {
                   MessageBox.Show("Заборонено вставляння об'єкта на місце умови
                    циклічного елімінування.", "Помилка", MessageBoxButton.OK,
                     MessageBoxImage.Exclamation);
                        return;
                  }
              }
  if
(selectedTerm.parent.GetType().ToString().Equals("TermEdit.CyclicParallelisation"))
              {
                  CyclicParallelisation cyclicParallelisation =
                                  (CyclicParallelisation)selectedTerm.parent;
                  if (cyclicParallelisation.condition == selectedTerm)
                  {
                   MessageBox.Show("Заборонено вставляння об'єкта на місце умови
                       циклічного паралелення.", "Помилка", MessageBoxButton.OK,
                         MessageBoxImage.Exclamation);          return;
                  }
              }
          }
          InsertForm insertForm = new InsertForm();

         if (!selectedTerm.GetType().ToString().Equals("TermEdit.Uniterm") ||
              ((Uniterm)selectedTerm).value != null)
          {
              insertForm.ShowDialog();
```

```
                if (insertForm.DialogResult == false)
                    return;
            }
            SequenceForm sf = new SequenceForm();
            sf.ShowDialog();
            if (sf.DialogResult == false)
                return;
                Sequence s = new Sequence();
                s.separator = sf.Separator;
                s.orientation = sf.Orientation;
                rootTerm = Term.ReplaceTerm(rootTerm, selectedTerm, s);
                if (insertForm.Method == InsertForm.InsertMethod.InsertReplace)
                {
                    s.termA = new Uniterm();
                    s.termA.parent = s;
                    s.termB = new Uniterm();
                    s.termB.parent = s;
                    znyszczyty = "InsertReplace";
                    operacijaCykliczna = false;                    }
                else if (insertForm.Method == InsertForm.InsertMethod.InsertWithFirst)
                {
                    s.termA = selectedTerm;
                    s.termA.parent = s;
                    s.termB = new Uniterm();
                    s.termB.parent = s;
                    znyszczyty = "InsertWithFirst";
                    operacijaCykliczna = false;
                }
                else
                {
                    s.termA = new Uniterm();
                    s.termA.parent = s;
                    s.termB = selectedTerm;
                    s.termB.parent = s;
                    znyszczyty = "InsertWithSecond";
                    operacijaCykliczna = false;
                }
                rootTerm.Deselect(this);
                selectedTerm = s;
                selectedTerm.selected = true;
                DrawingVisual dv = new DrawingVisual();
                rootTerm.CalculateSize(null, font);
                canwasDraw.Width = canwasDraw.Width + rootTerm.width + marginX;
                canwasDraw.Height = canwasDraw.Height + rootTerm.height +  marginY;
                InvalidateVisual();
                modified = true;
                if (fileName == null)
                    this.Title = "Генератор коду формул алгоритмів";
                else
     this.Title = "Генератор коду формул
лгоритмів"+System.IO.Path.GetFileName(fileName);
                rootTerm.Deselect(this);
                rootTerm.CheckForClick(this, font, xPos, yPos, marginX, marginY,
                Scroll_X_Y.ActualWidth, Scroll_X_Y.ActualHeight);
                canwasDraw_MouseDown(rootTerm, null);
        }
```

Висновки

Декомпозицією моделі інформаційної технології опрацювання формул розширеної алгебри алгоритмів на підмоделі і субмоделі зменшено складність її побудови.

Створена інформаційна технологія забезпечує побудову математичних моделей візуальних і функціональних складових інформаційних технологій і систем.

Побудована математична модель описує опрацювання формул алгебри алгоритмів.

Програмною реалізацією і апробацією виконано верифікацію побудованої математичної моделі інформаційної технології опрацювання формул алгоритмів.

1.Owsiak W., Owsiak A. Rozszerzenie algebry algorytmów /Pomiary, automatyka, kontrola. – 2010. – № 2. – S. 184 – 188. 2.Owsiak W., Owsiak A., Owsiak J. Teoria algorytmów abstrakcyjnych i modelowanie matematyczne systemów informacyjnych. – Opole: Politechnika opolska, 2005. – 275 s. 3.Ovsyak V.K.: Computation Models and Algebra of Algorithms. http://www.nbuv.gov.ua/Portal/natural/ VNULP/ISM/ 2008_621/01.pdf 4.Овсяк О. Класи інформаційної системи генерування коду /О. Овсяк // Вісник Тернопільського державного технічного університету: "Тернопільський національний технічний університет імені Івана Пулюя". – № 1, 2010. – С. 171 – 176. 5.Petzold C. Programowanie Windows w języku C#. –Warszawa: „RM", 2002. – 1161 s. 6.Мэтью Мак-Дональд. Windows presentation foundation в .NET 3.5 с примерами на C# 2008. – М.– СПб.–К.: Apress, 2008. – 922 с.

УДК 622.691.4:622.692.4

Н. Притула[1,2], М. Притула[1,2], В. Ямнич[2], А. Дацюк[3], С. Гладун[3], О.Химко[4]
[1]Центр математичного моделювання ІППММ ім. Я.С.Підстригача НАН України,
[2]ТзОВ "Математичний центр",
[3]Об'єднане диспетчерське управління ДК „Укртрансгаз",
[4]Національний університет "Львівська політехніка"

ТЕПЛОВИЙ РЕЖИМ ТРАНСПОРТУВАННЯ ГАЗУ

Проаналізовано проблему розрахунку теплового режиму транспортування газу для задач планування режимів роботи магістральних газопроводів та результати числових експериментів. Показано на реальних даних, що при плануванні режимів існуючі формули розрахунку параметрів стаціонарного температурного режиму транспортування газу вносять значну похибку. Для підвищення точності розрахунку запропоновано поправкові емпіричні коефіцієнти.

Ключові слова: тепловий режим, магістральний газопровід, газотранспортна система, компресорна станція, потенціал оптимізації.

The calculation problem of thermal conditions of gas transportation for tasks of working conditions of trunk gas pipelines planning is analyzed and shown in the results of numerical experiments. When planning the conditions the existing calculation formulae of stationary temperature conditions of gas transportation parameters cause a considerable error. It's shown on real data. In order to increase the calculation accuracy an empirical correction factors are suggested.

Key words: thermal conditions, trunk gas pipeline, gas–transport system, compressed air plant, optimization potential.

Вступ

На компресорних станціях магістральних газопроводів після газоперекачувальних агрегатів (ГПА) встановлені апарати повітряного охолодження (АПО) газу. Вони використовуються переважно влітку для виконання технологічних умов (температура газу на виході із станції повинна бути нижчою за 40 ^{0}C). У холодну пору року вентилятори АПО відімкнені для економії електро-