

АЛГОРИТМИ ЗГОРТАННЯ І РОЗГОРТАННЯ ФОРМУЛ АБСТРАКТНИХ АЛГОРИТМІВ

© Василюк А.С., Басюк Т.М., 2009

Подано означення процесів згортання, розгортання й адаптації формул алгоритмів. Наведено алгоритм комп'ютерної адаптації формул алгоритмів. Синтезовано, мінімізовано, побудовано математичну модель і досліджено алгоритм адаптації базового знака операції. Наведено моделі алгоритмів згортання і розгортання формул алгоритмів. Показана верифікація розроблених алгоритмів.

Ключові слова – процеси згортання, розгортання і адаптації формул алгоритмів.

Determination of processes of rolling up, development and adaptation of formulas of algorithms is described. The algorithm of adaptation of formulas of algorithms is resulted. It is synthesized, it is minimized, a mathematical model and probed algorithm of adaptation of base sign of operation is built. The models of algorithms of rolling up and development of formulas of algorithms are resulted.

Keywords – processes of rolling up, development and adaptation of formulas of algorithms.

Вступ

Відома [1, 2] алгебра алгоритмів, яка має оригінальні операції, наприклад, секвентування, елімінування, паралелення та циклічні операції, котрі позначаються спеціальними знаками, яких немає серед відомих математичних знаків. Для набору та редагування формул абстрактних алгоритмів розроблено спеціалізовану комп'ютерну підсистему МОДАЛ [3]. Але вона не виконує такі операції, як автоматична адаптація, згортання та розгортання формул абстрактних алгоритмів.

У таких мовах програмування, як Microsoft Visual C#. Net, Borland Delphi 8.0 та інших поширена технологія згортання та розгортання підпрограм. Завдяки такій можливості значно зменшується видимий код написаної програми. Це особливо актуально, коли код програми громіздкий. Але така проблема є не тільки в мовах програмування. Вона не втрачає актуальності і при створенні алгоритмів, адже оперувати великими алгоритмами також нелегко. Використовуючи згорнуті формули, набагато зручніше створювати алгоритми, зменшується кількість механічних помилок оператора набору. Тому у цій статті розглядається проблема згортання та розгортання формул абстрактних алгоритмів.

Ілюстрація згортання та розгортання формул абстрактних алгоритмів

Нехай дано формулу абстрактного алгоритму

$$\overbrace{F(x) ; G(y)}^{\beta}, B, U-?, \quad (1)$$

α

де α – знак операції секвентування, β – знак операції елімінування, $F(x)$, $G(y)$, B , $U-?$ – унітерми. Унітермами є вирази мови, зокрема математичної мови [4].

Унітерми $F(x)$, $G(y)$ є вкладеними в знак операції секвентування – α . Відносно цих унітермів знак операції секвентування не є вкладеним. Назвемо його базовим. Кожен з унітермів чи знаків операцій має певні параметри.

Такими параметрами є: id – унікальний ідентифікаційний номер формули, $TypeObj$ – тип формули, $index_op_a$ – ідентифікаційний номер першої вкладеної формули, $index_op_b$ – ідентифікаційний номер другої вкладеної формули, $FormulaName$ – назва формули, $Text$ – текст терма [5].

$$\overline{\#_ \alpha , B , U - ?} , \quad (2)$$

де $\#_ \alpha$ – умовне позначення формули алгоритму.

Згортання – процес перетворення формули алгоритму (формула α в формулі (1)), яка складається з елементарних термів до її умовного позначення (наприклад, унітерм $\#_ \alpha$ у формулі (2)).

Розгортання – процес перетворення умовного позначення формули алгоритму (наприклад, унітерм $\#_ \alpha$ в формулі (2)) до формули алгоритму (1).

Адаптацією називатимемо процес переміщення вкладених формул відносно базового знака операції і зміни його розмірів відносно вкладених формул.

Алгоритм комп'ютерної адаптації формул алгоритмів

Виконуючи операції згортання та розгортання, необхідно автоматично адаптувати положення вкладених формул і розміри базових знаків операцій. Загальний алгоритм адаптації описується такою формулою:

$$\left(\begin{array}{l} l = 0 \\ ; \\ \left(\begin{array}{l} K \\ ; \\ \overline{\varnothing i = n l} \\ * , c_i , (i = n l) - ? \end{array} \right) \end{array} \right) ,$$

де $l=0$ – ініціалізація автоматичної адаптації формули абстрактного алгоритму, K – алгоритм адаптації базової формули абстрактного алгоритму, n_l – максимальна кількість знаків операцій та термів абстрактного алгоритму.

Синтез абстрактного алгоритму адаптації базового знака операції

Для синтезу абстрактних алгоритмів розроблено два методи [1, 2]: секвенційний та табличний. Секвенційний метод є простішим, тому синтез абстрактного алгоритму адаптації виконано власне цим методом. Він передбачає виконання синтезу абстрактного алгоритму за два етапи. На першому етапі синтезуються секвенції, а на другому – елімінування.

Перший етап – синтез секвенцій.

Абстрактний алгоритм адаптації базового знака операції передбачає такі секвенції. Перша секвенція відповідає за адаптацію першої вкладеної формули до базової формули і адаптацію базової формули до вкладеної. Назвемо її S_1 . Друга секвенція відповідає за адаптацію другої вкладеної формули до базової формули та адаптацію базової формули до вкладеної. Назвемо її S_2 . Третя секвенція описує адаптацію базової формули, коли немає першої та другої вкладених в базову формулу формул. Назвемо її S_3 .

Наступні три секвенції, на відміну від попередніх трьох, описують аналогічні процеси, тільки за умови вертикальної орієнтації знака операції. Орієнтація знака операції може бути горизонтальною або вертикальною.

Для синтезу секвенцій $S_1, S_2, S_3, S_4, S_5, S_6$ нам необхідно описати такі унітерми: $P(k,0)$ – ініціалізація алгоритму адаптації формули абстрактного алгоритму, $P(k, a), P(k, a')$ – терми адаптації першої вкладеної формули в базу залежно від типу орієнтації знака операції, $P(k, b), P(k, b')$ – терми адаптації другої вкладеної формули в базу залежно від типу орієнтації знака операції, $K_1(x, y), K_1(x', y')$ – унітерми ідентифікації підалгоритмів адаптації першої та другої вкладених формул в базу залежно від орієнтації знака операції, K_2, K_2' – унітерми ідентифікації підалгоритмів адаптації базової формули залежно від вкладених формул

$$S_1 = \left(\begin{array}{c} P(k,0) \\ ; \\ P(k,a) \\ ; \\ K1(x,y) \\ ; \\ K2 \end{array} \right) \quad S_2 = \left(\begin{array}{c} P(k,0) \\ ; \\ P(k,b) \\ ; \\ K1(x,y) \\ ; \\ K2 \end{array} \right) \quad S_3 = \left(\begin{array}{c} P(k,0) \\ ; \\ K2 \end{array} \right) \quad S_4 = \left(\begin{array}{c} P(k,0) \\ ; \\ P(k,a') \\ ; \\ K1(x',y') \\ ; \\ K2' \end{array} \right) \quad S_5 = \left(\begin{array}{c} P(k,0) \\ ; \\ P(k,b') \\ ; \\ K1(x',y') \\ ; \\ K2' \end{array} \right) \quad S_6 = \left(\begin{array}{c} P(k,0) \\ ; \\ K2' \end{array} \right) \quad (3)$$

Другий етап – синтез елімінувань.

Секвенції S_1 і S_2 елімінуємо за умовою наявності другої вкладеної формули (унітерм $U1(b, c1) - ?$). Отримане елімінування із секвенцією S_1 елімінуємо за умовою наявності першої вкладеної формули (унітерм $U1(a, c1) - ?$). Секвенції S_5 і S_6 елімінуємо за умовою наявності другої вкладеної формули (унітерм $U1(b', c1) - ?$). Одержане елімінування із секвенцією S_4 елімінуємо за умовою наявності першої вкладеної формули (унітерм $U1(a', c1) - ?$).

Для синтезу елімінування L_1 необхідно описати такі унітерми: S_1, S_2 – секвенції з формули (3), $U1(b, c1) - ?$ – умовний унітерм перевірки на наявність другої вкладеної формули. Аналогічно до того, як синтезується елімінування L_1 , синтезуються всі решта елімінувань. У результаті виконаних дій отримуємо такі елімінування:

$$L_1 = \overline{S_2, S_3, U1(b,c1) - ?} \quad L_3 = \overline{S_5, S_6, U1(b',c1) - ?} \quad (4)$$

$$L_2 = \overline{S_1, L_1, U1(a,c1) - ?} \quad L_4 = \overline{S_4, L_3, U1(a',c1) - ?}$$

$$L_5 = \overline{L_2, L_4, U1(g,c1) - ?}$$

Мінімізація абстрактного алгоритму

Підставимо в елімінування L_1 (4) секвенції S_2 та S_3 (3).

На основі властивості виносу унітерму за знак операції елімінування [3] виносимо унітерми $P(k,0)$ і K_2 за знак операції елімінування й отримаємо таку формулу:

$$\left(\begin{array}{c} \overline{P(k,0), U1(b,c1) - ?} \\ ; \\ P(k,b) \\ ; \\ K1(x,y) \\ ; \\ K2 \end{array} \right)$$

$$\left(\begin{array}{c} P(k,0) \\ ; \\ \overline{P(k,b), *, U1(b,c1) - ?} \\ ; \\ K1(x,y) \\ ; \\ K2 \end{array} \right)$$

Аналогічно мінімізуються всі решта елімінувань. Виконавши підстановку та мінімізацію, одержимо таку формулу:

$$\left(\begin{array}{c} P(k,0) \\ ; \\ \left[\begin{array}{c} \left(\begin{array}{c} P(k,a) \\ ; \\ K1(x,y) \end{array} , \left[\begin{array}{c} P(k,b) , U1(a,c1) - ? \\ , \\ * \\ , \\ U1(b,c1) - ? \end{array} \right] \\ ; \\ K2 \end{array} \right] , \left[\begin{array}{c} \left(\begin{array}{c} P(k,a') \\ ; \\ K1(x',y') \end{array} , \left[\begin{array}{c} P(k,b') , U1(a',c1) - ? \\ , \\ * \\ , \\ U1(b',c1) - ? \end{array} \right] \\ ; \\ K2' \end{array} \right] , U1(g,c1) - ? \end{array} \right| \end{array} \right)$$

Модель абстрактного алгоритму

Модель абстрактного алгоритму будують, замінивши абстрактні унітерми предметними (конкретними) і задавши секвентні області значень змінних та унітермів [1, 2].

Замінюємо абстрактні унітерми на предметні і заданням секвентних областей значень змінних й унітермів будуюмо модель абстрактного алгоритму.

Абстрактний двомісний унітерм $P(m, n)$ замінюємо на предметний унітерм присвоєння $m=n$. Тоді $P(k,0)$ замінюється на $k=0$, $P(k, a)$ на $k=a$, $P(k, b)$ на $k=b$, $P(k, a')$ на $k=a'$, $P(k, b')$ на $k=b'$. Абстрактний умовний терм $U_1(p, q)-?$ замінюємо предметним термом порівняння $(p=q)-?$. У такому разі унітерм перевірки орієнтації виразу $U_1(g, c_1)-?$ замінюємо на $(g=c_1)-?$, унітерм перевірки наявності першого вкладеного виразу в двомісну операцію $U_1(a, c_1)-?$ на $(a=c_1)-?$, $U_1(b, c_1)-?$ на

$(b=c_1)-?$, $U_1(a', c_1)-?$ на $(a'=c_1)-?$, $U_1(b', c_1)-?$ на $(b'=c_1)-?$. Для c_1 задаємо таке значення: $c_1=0$.

Абстрактні унітерми $K_1(x, y)$, $K_1(x', y')$, K_2 і K_2' є складними абстрактними виразами, які, враховуючи обмежений обсяг статті, не можуть бути розкриті у ній. Моделі цих абстрактних унітермів будуються аналогічно, як це було продемонстровано вище.

Отримана модель абстрактного алгоритму матиме такий вигляд:

$$\left(\begin{array}{c} k=0 \\ ; \\ \left[\begin{array}{c} \left(\begin{array}{c} k=a \\ ; \\ K1(x,y) \end{array} , \left[\begin{array}{c} k=b , (a=c1) - ? \\ , \\ * \\ , \\ (b=c1) - ? \end{array} \right] \\ ; \\ K2 \end{array} \right] , \left[\begin{array}{c} \left(\begin{array}{c} k=a' \\ ; \\ K1(x',y') \end{array} , \left[\begin{array}{c} k=b' , (a'=c1) - ? \\ , \\ * \\ , \\ (b'=c1) - ? \end{array} \right] \\ ; \\ K2' \end{array} \right] , (g=c1) - ? \end{array} \right| \end{array} \right)$$

Дослідження моделі алгоритму

Теорема. Математичною моделлю алгоритму описуються адаптація вкладених формул до знаків базових операцій та ідентифікація алгоритмів адаптації знаків базових операцій до вкладених формул.

Доведення. Для доведення використовуємо метод трансфінитної математичної індукції [4], оскільки доведення за всіх можливих значень змінних є аналогічним, а самі значення змінних є впорядкованими.

Доведемо твердження для змінної a за таких значень всіх решти змінних $a'=3, b=4, b'=5, g=0$. Нехай твердження справедливе за будь-яких $i < p$, де $p < n1$. При $a = p$ змінній k присвоюється значення 0. З елімінування за умовою $(g=c_1)-?$, якою ідентифікується горизонтальна конфігурація

Модель абстрактного алгоритму розгортання формул абстрактних алгоритмів

Для створення моделі абстрактного алгоритму розгортання формул абстрактних алгоритмів необхідно спочатку описати модель пошуку формули за назвою згорнутої формули.

$$\left(\begin{array}{l} f1 = 0 \\ ; \\ \text{ } \mathcal{Z} i = n1 \\ \hline * , \left(\begin{array}{l} \hline f1 = n3 , * , (t2 = t4) - ? \\ ; \\ c_i \end{array} \right) , (i = n1) - ? \end{array} \right)$$

Модель описує процес пошуку формули за його назвою. Вона містить такі унітерми: $f1 = 0$ – ініціалізація системи пошуку за назвою, $f1 = n3$ – унітерм присвоєння критерію пошуку, $(t2 = t4) - ?$ – умовний унітерм перевірки на наявність шуканої формули, $(i = n1) - ?$ – умовний унітерм закінчення пошуку формули в формулі абстрактного алгоритму.

Використовуючи вищеописану модель пошуку виразу за назвою, створюємо модель абстрактного алгоритму розгортання формул абстрактних алгоритмів.

Формула абстрактного алгоритму (11) є моделлю абстрактного алгоритму розгортання формул абстрактних алгоритмів. Для створення цієї моделі використано такі унітерми: $a = f1$ – ініціалізація згорнутої першої вкладеної формули, $(a = id1) - ?$ – унітерм перевірки на наявність першої вкладеної формули, $(i = n1) - ?$ – умовний унітерм закінчення розгортання, $b = f1$ – ініціалізація згорнутої другої вкладеної формули, $(b = id1) - ?$ – унітерм перевірки на наявність другої вкладеної формули, $K1(x, y)$ – унітерми ідентифікації підалгоритмів адаптації першої та другої вкладених формул в базовий, $x1...x12 = 0, y1...y12 = 0$ – унітерм знищення згорнутої формули абстрактного алгоритму.

$$\left(\begin{array}{l} \text{ } \mathcal{Z} i = n1 \\ \hline * , \left(\begin{array}{l} \hline a = f1 , * , (a = id1) - ? \\ ; \\ b = f1 , * , (b = id1) - ? \\ ; \\ c_i \end{array} \right) , (i = n1) - ? \\ ; \\ \left(\begin{array}{l} K1(x,y) \\ ; \\ x1...x12 = 0 \\ ; \\ y1...y12 = 0 \end{array} \right) \end{array} \right)$$

Верифікація розроблених моделей

На основі синтезованих та мінімізованих моделей згортання і розгортання формул абстрактних алгоритмів створено систему АбстрактАл, інтерфейс якої зображено нижче.

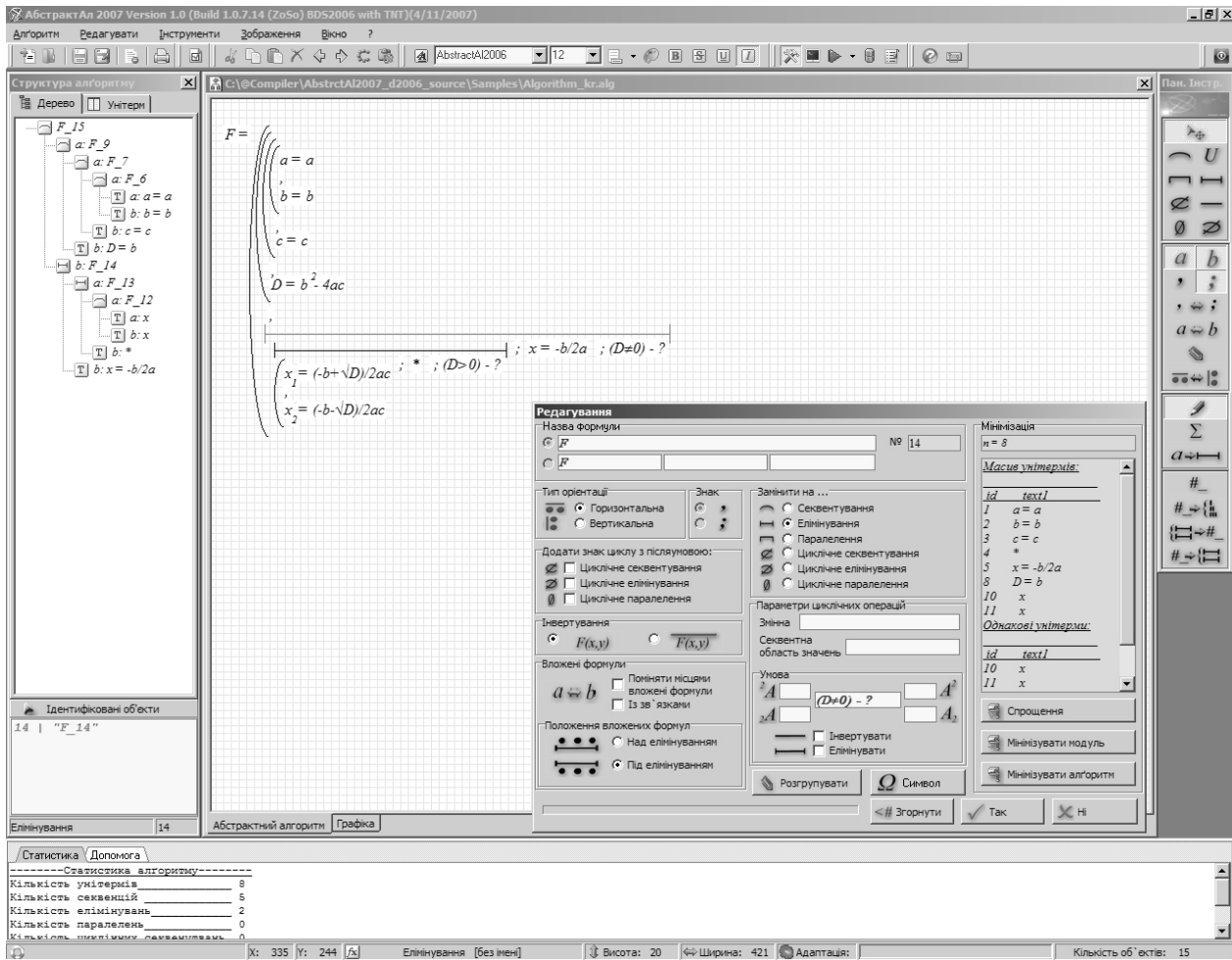


Рис. 1. Абстрактний алгоритм

На рис. 1 зображено реалізацію системи АбстрактАл, виконану засобами мови програмування Borland Delphi 7.0 [6, 7]. Ліворуч зображено формулу абстрактного алгоритму в розгорнутому вигляді, а праворуч зображено формулу абстрактного алгоритму після виконання операції згортання. Після використання операції часткового згортання (згортання вибраної формули алгоритму, а не всієї формули) формул абстрактних алгоритмів формула абстрактного алгоритму Евкліда стала лаконічнішою, читабельною і розмір формули алгоритму значно зменшився. Також важливим є те, що при повторенні згорнутої формули нема необхідності описувати всю формулу, а тільки її згорнутий вигляд.

Розглянемо приклад. Нехай необхідно згорнути, а потім розгорнути знак операції елінування.

Дії, необхідні для вищеописаних операцій з алгоритмом:

- ідентифікувати знак операції елінування;
- інструментом “Згорнути чи розгорнути формулу” здійснити згортання знака операції елінування.

Результатом вищеописаних дій буде формула, відображена в головному вікні редактора:

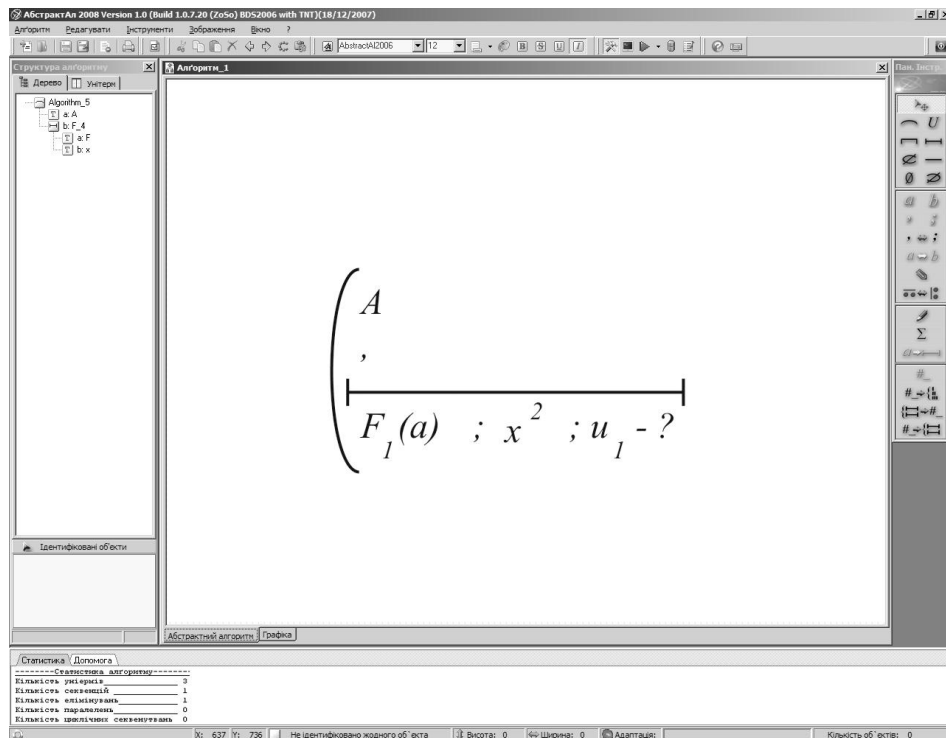


Рис. 2. Формула абстрактного алгоритму до процесу згортання

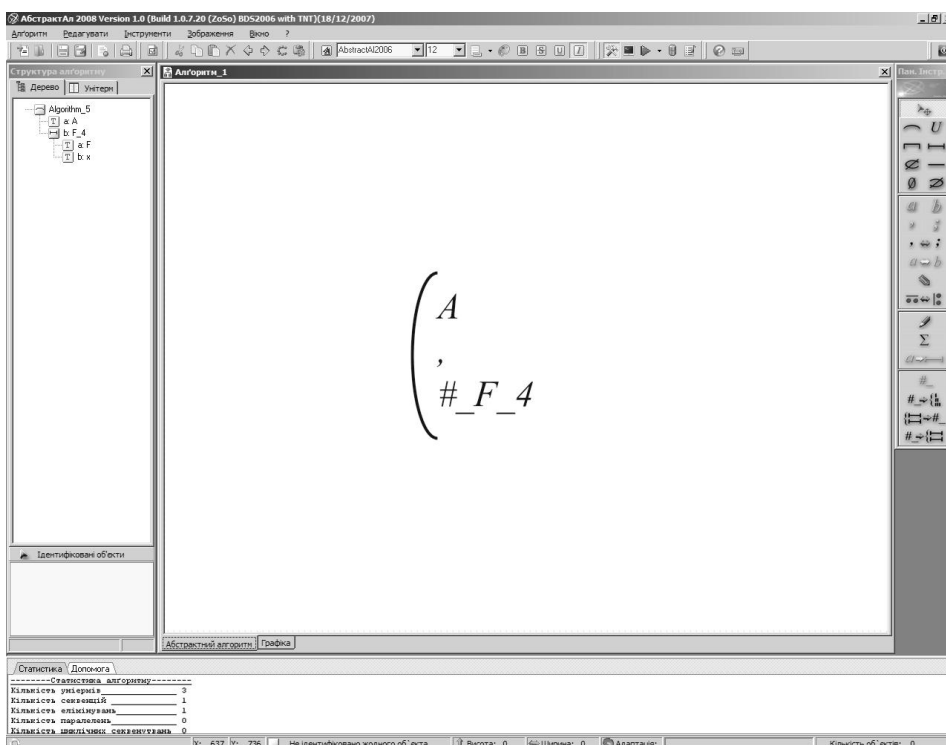


Рис. 3. Формула абстрактного алгоритму після процесу згортання

Дії, які необхідно здійснити для розгортання формули алгоритму:

- ідентифікувати згорнуту формулу;
- інструментом “Згорнути чи розгорнути формулу” здійснити розгортання згорнутої формули.

Результат роботи системи зображено на рис. 2 і 3.

Висновки

1. Синтезована, мінімізована і досліджена математична модель алгоритму адаптації формул абстрактних алгоритмів описує ідентифікацію підалгоритмів адаптації базових операцій до вкладених формул і вкладених формул до базових, реалізація якої забезпечує більшу наочність подання алгоритмів у вигляді формул теорії абстрактних алгоритмів.
2. Дослідження математичної моделі ще до її практичної реалізації та апробації забезпечило виявлення помилок, допущених під час її синтезу та доводить, що вона описує необхідні процеси.
3. Абстрактним алгоритмом адаптації формул абстрактних алгоритмів описано процеси автоматичного згортання та розгортання формул абстрактних алгоритмів.
4. Верифікацією розроблених математичних моделей алгоритмів підтверджена їх коректність. Програмна реалізація, крім автоматизованих процесів набору та редагування в автоматичному режимі, виконує адаптацію вкладених формул до базових і базових формул до вкладених.

1. Овсяк В., Бритковський В., Овсяк О., Овсяк Ю. Синтез і дослідження алгоритмів комп'ютерних систем. – Львів, 2004. – 276 с. 2. Овсяк В. АЛГОРИТМИ: методи побудови, оптимізації, дослідження вірогідності. – Львів: Світ, 2001. – 160 с. 3. Бритковський В.М. Моделювання редактора формул секвенційних алгоритмів. Автореф. дис. роб. к.т.н. – Львів: видавничо-поліграфічний відділ ЛвЦНТЕІ. – 18 с. 4. Математическая энциклопедия: Гл. ред. И.М. Виноградов, т.3. – М.: Советская Энциклопедия, 1982. – 1184 с. 5. Овсяк В., Василюк А. Принцип побудови підсистеми редагування формул абстрактних алгоритмів // Комп'ютерні технології друкарства – Львів: УАД, – 2004. – № 12. – С. 137–146. 6. Архангельский А.Я. Delphi 6. Справочное пособие – М.: ЗАО “Издательство БИНОМ”, 2001. – 1024 с. 7. Архангельский А.Я. Программирование в Delphi 6. – М.: ЗАО “Издательство БИНОМ”, 2001.