УДК 519.681

В. Г. Акуловський, кандидат технічних наук,
доцент кафедри інформаційних систем
та технологій Університету митної справи
та фінансів
В. В. Костенко, старший викладач
кафедри інформаційних систем та технологій
Університету митної справи та фінансів
В. В. Поліщук, старший викладач
кафедри інформаційних систем та технологій
Університету митної справи та фінансів

РЕАЛІЗАЦІЯ ЗАСАД ЗАХИЩЕНОГО ПРОГРАМУВАННЯ В РАМКАХ АЛГЕБРАЇЧНОГО АПАРАТУ

Для подолання недостатньої надійності програмного забезпечення розв'язується задача реалізації засобів захищеного програмування. Застосовується спеціальний алгебраїчний апарат (алгебра алгоритмів з даними).

У рамках алгебри алгоритмів з даними використано тризначні, k-значні логічні умови, які фіксують, і похідні алгоритмічні конструкції на їх основі для реалізації концепції захищеного програмування. Зокрема, введено обмеження на використання алгоритмічних конструкцій, що перевіряються на етапі розробки алгоритмів, засоби контролю оброблюваних даних і послідовності їх обробки на етапі виконання програми.

Ключові слова: *алгебра алгоритмів; захищене програмування; Д-оператори; k-значні логічні умови; алгоритмічні конструкції; логічні умови, що фіксують.*

To overcome the lack of reliability of the software solves the problem of realization of assets protected programming. Applicable special algebraic tools (algebra algorithms to the data).

As part of the algebra algorithms to data used in the three-digit, k-digit fixing logical conditions, and derivatives algorithmic designs based on them to implement secure programming concepts. In particular, there are imposes restrictions on using algorithmic constructs scanned at design algorithms and controls data to be processed and the processing sequence at step program execution.

Key words: *algebraic tools; secure programming tools; D-operators; k-valued logic conditions; three-valued logic conditions; fixing logic conditions.*

Постановка проблеми. Кризові явища, що давно існують у програмуванні [1; 2], до нинішнього дня не подолано, а втрати від них досить істотні [3]. Одним із найперспективніших шляхів кардинального розв'язання проблеми кризи програмування є формалізація процесу розробки програмного забезпечення (далі − ПЗ) [4; 5]. У зв'язку з цим запропоновано алгебраїчний апарат (алгебра алгоритмів з даними) [6−8].

Аналіз останніх досліджень і публікацій. Згаданий вище алгебраїчний апарат побудовано в результаті модифікації відомої моделі ЕОМ Глушкова [5; 6], яка розширена за рахунок уведення носія даних. Носій даних ототожнюється з пам'яттю і пристроями введення-виведення й називається апаратним середовищем, а безліч даних, що зберігаються носієм, позначено D^{AC}.

Структурами даних, що зберігаються носієм, називається пара $D = \langle N_j, З \rangle$ (зокрема, простими даними $D = \langle N_j, з \rangle$, де $З = \langle з_1, ..., з_n \rangle$ – кортеж послідовно розташованих значень; *з* – значення), що визначає в кожен момент часу поточний стан даних, носій якого N_j, однозначно ідентифікується індексом (адресою) $j \in J$. Якщо збережене значення не визначено, то будемо говорити, що структури даних (прості дані) невизначені, й позначатимемо їх у цьому випадку $D^{\breve{}}$.

Під час побудови алгебри введено поняття Д-оператора $(D)X(D')$, на вході й виході яких специфіковано оброблювані цим Д-оператором дані, й стану обчислювального процесу (далі – ОП). Нижче наведемо визначення цих понять.

Визначення 1. Станом ОП називається множина даних $D^T = (D^{AC} \cup D^D)$, де D^{AC} – статична складова стану ОП, що $D^{AC} \subseteq D^T$ ($D^{AC} \neq \varnothing$) у будь-якому стані D^T, а D^D – динамічна складова (додаткові дані) стану ОП, що можливо як $D^D \neq \varnothing$, так і $D^D = \varnothing$. Стан ОП змінюється, і він переходить з початкового $\hat{D}^T$, де $D \subseteq \hat{D}^T$, у підсумковий стан $\breve{D}^T$, де $D' \subseteq \breve{D}^T$, в результаті й тільки в результаті вимірювання даних D', викликаного виконанням Д-оператора $(D)X(D')$. При цьому, якщо в деякому стані ОП $D^D \neq \varnothing$, то в будь-якому іншому стані для цієї множини даних виконується $D^D = \varnothing$. Множина станів ОП включає стан D^T_μ, після потрапляння в який усі наступні кроки ОП не визначені.

Визначення 2. Базовий набір Д-операторів включає:

– $(D)O(D')$ такий, що $D, D' \subseteq D^{AC}$;

– $(D)O(D^d)$ такий, що $D \subseteq D^{AC}$, а D^d – динамічні дані такі, що $D^d \subseteq D^D$, $D^d \cap D^{AC} = \varnothing$;

– $(D^\pi)P(\alpha)$, який називається предикатом, *n*-місна (у загальному випадку) логічна функція $P: D^\pi \to \alpha \in E_2$, де $E_2 = \{1,0\}$; D^π – графік деякого n-арного відношення такий, що $D^\pi = (D^d, D^c, D'')$; $D'' \subseteq D^{AC}$; D^c – множина констант така, що $D^c \subseteq D^D$, $D^c \cap D^{AC} = \varnothing$; α – це двозначна логічна умова така, що $\alpha \subseteq D^D$, $\alpha \cap D^{AC} = \varnothing$ та $\alpha = 0$, якщо умова не виконується, та $\alpha = 1$ в усіх інших випадках. Предикат аналізує множину даних D^π та продукує логічну умову α, що характеризує поточне значення даних, що зберігаються і які в результаті виконання предикату не змінюються;

– Д-оператор $(D)X(D')$, який переводить ОП зі стану $\hat{D}^T$ у стан $\breve{D}^T_\mu$, називається невизначеним і позначається *N;*

– Д-оператор $(D)X(D')$, який переводить ОП зі стану $\hat{D}^T$ у стан $\breve{D}^T$ так, що $\hat{D}^T = \breve{D}^T$ (стан ОП не змінюється), називається тотожним і позначається *Z.*

Крім того, в дослідженнях [6; 7] показано можливість побудови похідних Д-операторів, зокрема, вигляду

$$(\dot{D})O(D', \alpha),\ (D)O(D', D^d),\ (D)O(\alpha). \qquad (1)$$

На множині Д-операторів визначено такі основні та похідні операції алгебри.

Композиція Д-операторів (операція позначається "*") $(D)X(D')*(\widetilde{D})\widetilde{X}(\widetilde{D'})$ являє собою послідовне виконання спочатку Д-оператора $(D)X(D')$, після цього – Д-оператора $(\widetilde{D})\widetilde{X}(\widetilde{D'})$. Композиція предикатів записується у вигляді $(\dot{D}^{\tau})\dot{P}(\alpha)\circ(\ddot{D}^{\pi})\ddot{P}(\alpha)$, де ∘ – одна з відомих [5] логічних операцій, визначених на множині логічних умов.

– *P*-диз'юнкція

$$[(D^{\pi})P(\alpha)]((D_1)O_1(D_1')*(D_2)O_2(D_2')) = \begin{cases}(D_1)O_1(D_1'), \text{якщо } \alpha = 1;\\ D_2)O_2(D_2'), \text{якщо } \alpha = 0,\end{cases}$$

у результаті виконання якої обирається один із двох можливих Д-операторів, відповідно до значення логічної умови α.

– *P*-фільтрація $[(D^{\pi})P(\alpha P(\alpha)_1)O_1(D_1') = (D_1)O_1(D_1')$, якщо $\alpha = 1$, у результаті виконання якої Д-оператор $(D_1)O_1(D_1')$ обирається тільки за істинного значення логічної умови α;

– *P*-ітерація $[(D^{\pi})P(\alpha P(\alpha)]\{O(D')\}$ здійснює циклічне виконання Д-оператора $(D)O(D')$, якщо при $\alpha = 1$, і завершується в протилежному випадку.

– зворотня *P*-ітерація $\{(D)O(D')\}[(D^{\pi})P(\alpha P]$, в якій умова виконання перевіряється після виконання тіла циклу;

– узагальнений цикл $\{(D_1)O_1(D_1')[(D)P(\alpha[((D_2)O_2(D_2')\}$, тобто цикл з контролем умови виконання, що здійснюється в середині циклу;

– перемикач

$$\begin{pmatrix}(D_1^{\pi})P_1(\alpha_1^2),...,(D_n^{\pi})P_n(\alpha_n)\\ (D_1)O_1(D_1'),...,(D_n)O_n(D_n')\end{pmatrix},$$

який виконує *i*-й Д-оператор, котрому відповідає справжнє значення логічної умови, що продукується предикатом $(D_i^{\pi})P_i(\alpha_i^2)$, після чого виконання операції завершується.

Оскільки серед причин, що викликають згадані кризові явища, помітне місце належить недостатній надійності програмного забезпечення, попереду стоїть завдання реалізації засобів захищеного програмування [9] в рамках запропонованого алгебраїчного апарату. Виконання цього завдання – **мета статті**.

Виклад основного матеріалу. Реалізація захищеного програмування. Зазначене завдання розіб'ємо на дві підзадачі.

Слід виявити такі обмеження на використовувані програмні конструкції, які можуть бути перевірені на етапі проектування алгоритму.

Необхідно забезпечити можливості виявлення імовірних помилок та їх обробку в процесі виконання програми.

Розгляд першої підзадачі почнемо з такого визначення.

Визначення 3. Будь-яка алгоритмічна конструкція називається тотожною, якщо вона не змінює стан ОП, невизначеною, якщо вона переводить обчислювальний процес у невизначений стан, і такі алгоритмічні конструкції неприпустимі (не можуть бути

використані). Так само неприпустимі алгебраїчні конструкції, в результаті виконання яких дані безрезультатно (без використання) втрачаються або набувають невизначених значень.

Покажемо деякі обмеження, що накладаються на Д-операторів.

Твердження 1. Д-оператори, які продукують логічні умови (зокрема, предикати), можуть використовуватися тільки в операціях р-диз'юнкції, р-ітерації й похідних від них операціях.

Доведення. Оскільки логічна умова α, відповідно до визначення 1, визначення 2 і визначення операцій, існує в єдиному стані ОП, обробляється тільки перерахованими операціями, то Д-оператори, які продукують логічні умови, можуть використовуватися тільки в цих операціях, адже в протилежному випадку логічну умову буде втрачено, що, відповідно до визначення 3, неприпустимо.

Твердження доведено.

Розглянемо операцію композиції Д-операторів, на яку визначення не накладає обмежень, проте такі обмеження мають місце, що й покажемо за допомогою такого твердження.

Твердження 2. У композиції Д-операторів предикат $(D^{\pi})P(\alpha)$ може розташовуватися тільки останнім, Д-оператор $(D)O(D^{d})$ може розташовуватися тільки перед предикатом, а на вході предиката специфікуються динамічні дані $D'^{d} \subseteq D^{\pi}$ такі, що $D^{d} = D'^{d}$.

Доведення. Оскільки, логічна умова α згідно з визначенням 2 жодним з Д-операторів не обробляється, а відповідно до визначення 1 в разі композиції $(D^{\pi})P(\alpha^{3})*(D)X(D)$, де $(D)X(D)$ довільний (крім предиката, див. визначення композиції) Д-оператор, логічну умову α, за визначенням 1, буде втрачено, а така композиція відповідно до визначення 3 неприпустима. Аналогічні міркування щодо динамічних даних приведуть до результату, з якого випливає, що допустима лише композиція, в якій Д-оператор $(D)O(D^{d})$ передує предикату.

Подальші міркування побудуємо від протилежного. Припустимо, що на вході предиката динамічні дані не специфіковані. В цьому випадку $D^{d} \cap D^{\pi} = \varnothing$ і динамічні дані D^{d} предикатом не обробляються й безрезультатно втрачаються, що відповідно до визначення 3 неприпустимо. Припустимо: на вході предиката специфіковані динамічні дані $D'^{d} \subseteq D^{\pi}$ такі, що $D'^{d} \neq D^{d}$. У цьому випадку для даних $D^{d} \setminus D'^{d}$ реалізується вищерозглянутий випадок, а для даних $D'^{d} \setminus D^{d}$ виконується співвідношення $D'^{d} \setminus D^{d} = D^{\gamma}$, тобто частина даних на вході предиката виявиться невизначеною, що відповідно до визначення 3 неприпустимо.

Оскільки у зроблених припущеннях отримано неприпустимі варіанти композиції, то зроблені припущення неправильні, а $D'^{d} \subseteq D^{\pi}$ та $D^{d} = D^{d}$.

Твердження доведено.

Висновок. Доведені твердження задають обмеження на використання операції композиції, а всі можливі варіанти її використання утворюють такий набір:

$$(D)O(D')*(\tilde{D})\tilde{O}(\tilde{D}'); \quad (D)O(D')*(\tilde{D})\tilde{O}(D^{d});$$
$$(D)O(D')*(D^{\pi})P(\alpha); \quad (D)O(D^{d})*(D^{\pi})P(\alpha);$$
$$(D^{\pi})P(\alpha)*(\tilde{D}^{\pi})\tilde{P}(\tilde{\alpha}).$$

Висновок. Аналогічно доводяться такі ж обмеження для похідних Д-операторів (1).

Тепер зупинімося на операції *p*-диз'юнкції. Щодо цього зазначимо: за визначенням операції наявність логічної умови в операціях *p*-диз'юнкції є необхідною і достатньою умовою виконання. Звідси випливають обмеження на її використання

$$[(D^{\pi})P(\alpha)]((D_1)X_1(D_1')*(D_2)X_2(D_2'))=N,$$

якщо $\alpha=\varnothing$ або $\alpha=D^{\gamma}$.

Розглянемо операцію *p*-ітерації і розв'яжемо задачу знаходження необхідної умови завершення циклу, враховуючи властивості, що випливають з її визначення:

– хибне значення умови ($\alpha=0$), отримано за кінцеве число кроків, достатня умова для завершення операції (циклу);

$$-\ [(D^{\pi})P(\alpha)]\{(\mathrm{D})\mathrm{O}(\mathrm{D}')\}=\mathrm{Z},\ \text{якщо}\ \alpha=0\ \text{на першому витку циклу;} \qquad (2)$$

$$-\ [(D^{\pi})P(\alpha)]\{(\mathrm{D})\mathrm{O}(\mathrm{D}')\}=N,\ \text{якщо}\ \alpha\equiv 1, \qquad (3)$$

що звичайною мовою називається "зацикленням" ОП, і якщо $D^{\pi}=D^{\gamma}$.

$$-\ [(D^{\pi})P(\alpha)]\{(\mathrm{D})\mathrm{O}(\mathrm{D}')\}=[(D^{\pi})P(\alpha)](\mathrm{D})\mathrm{O}(\mathrm{D}'), \qquad (4)$$

якщо на другому витку циклу $\alpha=0$, тобто операція *p*-ітерація за зазначеної умови вироджується в *p*-фільтр

Теорема 1. Необхідними для завершення операції *p*-ітерації $[(D^{\pi})P(\alpha)]\{O(D')\}$, яка виконує більше двох витків, є умови $D^{\pi}\cap D'\neq\varnothing$ та $D\cap D'\neq\varnothing$.

Доведення побудуємо від протилежного, припустивши, що виконується співвідношення $D^{\pi}\cap D'=\varnothing$.

На першому витку циклу предикат $(D^{\pi})P(\alpha)$ продукує логічну умову $\alpha=1$, адже у протилежному випадку, відповідно до (2), $[(D^{\pi})P(\alpha)]\{O(D')\}=Z$.

У результаті виконання тіла циклу $(D)O(D')$ – множина даних D^{π} не змінюється, так як для значень даних D' (визначення 1), що зазнали зміни, виконується $D^{\pi}\cap D'=\varnothing$, а предикат, відповідно до його визначення, стану множини даних D^{π}, не змінюється. Таким чином, незалежно від значення виразу $D\cap D'$, другий і всі наступні витки циклу відбуватимуться за незмінного D^{π}, тобто отримуємо $\alpha\equiv 1$ і відповідно до (3) $[(D^{\pi})P(\alpha)]\{O(D')\}=N$.

Припустимо, що $D^{\pi}\cap D'\neq\varnothing$ і виконується співвідношення $D\cap D'=\varnothing$.

У результаті виконання тіла циклу $(D)O(D')$ множина даних D^{π} зміниться ($D^{\pi}\cap D'\neq\varnothing$) й ми отримаємо множину даних $D'^{\pi}\neq D^{\pi}$.

Якщо на другому витку циклу предикат $(D'^{\pi})P(\alpha)$ продукує справжню логічну умову ($\alpha=1$), то на другому і всіх наступних кроках циклу множина даних D залишиться незмінною, в силу $D\cap D'=\varnothing$. Це призведе до незмінності даних D', що своєю чергою, призведе до незмінності D'^{π}. Таким чином, отримуємо вищерозглянутий випадок $[(D^{\pi})P(\alpha)]\{O(D')\}=N$.

Якщо на другому витку циклу $\alpha = 0$, то операція *p*-ітерації завершується на другому витку циклу (вироджується в послідовний фільтр, відповідно до (4), що суперечить умові теореми.

За зроблених припущень отримуємо або тотожну операцію, або вона переводить ОП у невизначений стан (зациклюється), що відповідно до визначення 3 неприпустимо. Отже, зроблені припущення хибні; а задані умови необхідні для завершення p-ітерації.

Теорему доведено.

Висновок 1. Для похідних операцій зворотна *p*-ітерація та узагальнений цикл умови завершення циклу легко можуть бути показані аналогічно виконаному доведенню.

Наведені у твердженнях 1 та 2, теоремі 1 і висновках з них обмеження на використання операцій алгебри може бути перевірено на етапі проектування алгоритму й таким чином є розв'язком першої зі згаданих задач щодо реалізації концепції захищеного програмування.

Виконанням другого із зазначених завдань, що забезпечує виявлення і обробку можливих помилок, полягає в контролі оброблюваних даних (у тому числі одержуваних логічних умов) і послідовності виконання операцій щодо їх обробки.

Виконання цього завдання почнемо з розширення логічної складової алгебраїчного апарату. Для цього введемо тризначні логічні умови $\alpha^3 \in E_3$, де $E_3 = \{1,0,\mu, 1,0,\mu$ – логічні константи, і тризначний предикат [11]. Тризначний предикат, позначається $(D^\pi)P(\alpha^3)$, аналізує множину даних D^π і продукує логічну умову α^3, що характеризує поточні значення даних, що зберігаються, коли $\alpha = 0$, якщо умова не виконується, та $\alpha = 1$, якщо умова виконується, то $\alpha = \mu$ у всіх інших випадках.

Отриманий предикат дозволяє ввести операцію p_3-диз'юнкції

$$[(D^\pi)P(\alpha^3)]((D_1)X_1(D_1') \vee (D_2)X_2(D_2') \vee (D_3)X_3(D_3')) =$$

$$= \begin{cases} (D_1)X_1(D_1'), \text{ якщо } \alpha^3 = 1; \\ D_2)X_2(D_2'), \text{ якщо } \alpha^3 = 0; \\ (D_3)X_3(D_3'), \text{ якщо } \alpha^3 = \mu, \end{cases}$$

в результаті застосування якої обирається один із трьох можливих Д-операторів відповідно до значення логічного умови α^3. Тобто дозволяє передбачити реакцію алгоритму в разі, коли логічна умова набуває будь-якого некоректного значення. Отже, Д-оператор $(D_3)X_3(D_3')$ може служити засобом індикації помилкової ситуації і/або обробки такої ситуації.

Для подальшого розширення отриманої можливості на множині E_3 введено відношення порядку < таке, що $0<\mu<1$, а на множині логічних умов − узагальнені булеві операції, таблиці істинності яких наведено в табл. 1. Узагальнені булеві операції задовольняють усі закони булевої алгебри, крім закону виключного третього й закону суперечності. Властивості узагальнених булевих операцій і відповідних канонічних форм подання умов детально досліджено в [5]. Під час виконання композиції предикатів, яку запишемо у вигляді $(\hat{D}^\pi)\hat{P}(\hat{\alpha}^3) \circ (\check{D}^\pi)\check{P}(\check{\alpha}^3)$ (де $\circ$ – одна з логічних операцій), буде

отримано логічну умову $\alpha^3 = \hat{\alpha}^3 \circ \breve{\alpha}^3$ в результаті застосування логічної операції $\circ$ до логічних умов $\hat{\alpha}^3$ та $\breve{\alpha}^3$, які продукуються кожним із предикатів.

Таблиця 1

Таблиці істинності логічних операцій

Інверсія				Диз'юнкція	Кон'юнкція
α^3	$\overline{\alpha}^3$	α^3	β^3	$\alpha^3 \vee \beta^3$	$\alpha^3 \wedge \beta^3$
0	1	0	0	0	0
μ	μ	0	μ	μ	0
1	0	0	1	1	0
		μ	0	μ	0
		μ	μ	μ	μ
		μ	1	1	μ
		1	0	1	0
		1	μ	1	μ
		1	1	1	1

Оскільки така композиція продукує логічну умову, то вона може використовуватися в операціях p_3-диз'юнкції. У разі використання композиції предикатів операція p_3-диз'юнкції записується у такий спосіб:

$$[(\hat{D}^{\pi})\hat{P}(\hat{\alpha}^3) \circ (\breve{D}^{\pi})\breve{P}(\breve{\alpha}^3)]((D_1)X_1(D_1') \vee (D_2)X_2(D_2') \vee (D_3)X_3(D_3')) =$$

$$= \begin{cases} (D_1)X_1(D_1'), \text{ якщо } \alpha^3 = 1; \\ (D_2)X_2(D_2'), \text{ якщо } \alpha^3 = 0; \\ (D_3)X_3(D_3'), \text{ якщо } \alpha^3 = \mu, \end{cases}$$

де $\alpha^3 = \hat{\alpha}^3 \circ \breve{\alpha}^3$.

Узагальнення композиції предикатів запишемо у вигляді $(D_1^{\pi})P_1(\alpha_1^3) \circ \ldots \circ (D_n^{\pi})P_n(\alpha_1^3)$, де $\alpha^3 = \alpha_1^3 \circ \ldots \circ \alpha_n^3$, а операцію p_3-диз'юнкції у вигляді:

$$[(D_1^{\pi})P_1(\alpha_1^3) \circ \ldots \circ (D_n^{\pi})P_n(\alpha_n^3)]((D_1)X_1(D_1') \vee (D_2)X_2(D_2') \vee (D_3)X_3(D_3')).$$

У цьому випадку побудована операція може служити засобом індикації помилкової ситуації і/або обробки такої ситуації під час аналізу складних логічних умов.

Крім того, тризначні логічні умови дозволяють контролювати некоректні логічні умови за рахунок розширення можливостей похідних операцій алгебри.

Зокрема, побудуємо операцію p-фільтрації, захищену від помилкової логічної умови. Для цього запишемо операцію p_3 – диз'юнкції у вигляді

$$[(D^{\pi})P(\alpha^3)]((D_1)X_1(D_1')*Z*(D_3)X_3(D_3')) =$$

$$= \begin{cases} (D_1)X_1(D_1'), \text{ якщо } \alpha^3 = 1; \\ Z, \text{ якщо } \alpha^3 = 0; \\ (D_3)X_3(D_3'), \text{ якщо } \alpha^3 = \mu, \end{cases}$$

а отриману таким чином конструкцію у вигляді:

$$[(D^{\pi})P(\alpha^2)]((D)X(D')*(\tilde{D})\tilde{X}(\tilde{D'})), \quad (5)$$

де Д-оператор $(D)X(D')$ виконується за істинного значення логічної умови, а $(\tilde{D})\tilde{X}(\tilde{D'})$ – за будь-якого невизначеного, відмінного від 0 і 1 значення.

Організувавши вкладеність p_3-диз'юнкції

$$\begin{aligned}
&[(D_1^{\pi})P_1(\alpha_1^2)]((D_1)O_1(D_1')\vee(\tilde{D}_1)\tilde{O}_1(\tilde{D}_1')\vee\\
&\vee[(D_2^{\pi})P_2(\alpha_2^2)]((D_2)O_2(D_2')\vee(\tilde{D}_2)\tilde{O}_2(\tilde{D}_2')\vee\ldots\\
&\ldots\vee[(D_n^{\pi})P_n(\alpha_n)](D_n)O_n(D_n')\vee(\tilde{D}_n)\tilde{O}_n(\tilde{D}_n')\vee(\tilde{D}_{n+1})\tilde{O}_{n+1}(\tilde{D}_{n+1}')\ldots)=\\
&=\begin{pmatrix}(D_1^{\pi})P_1(\alpha_1^3),\ldots,(D_n^{\pi})P_n(\alpha_n^3)\\(D_1)O_1(D_1'),\ldots,(D_n)O_n(D_n')\\(\tilde{D}_1)\tilde{O}_1(\tilde{D}_1'),\ldots,(\tilde{D}_n)\tilde{O}_n(\tilde{D}_n')\\(D_{n+1})O_{n+1}(D_{n+1}').\end{pmatrix},
\end{aligned}$$

отримуємо перемикач, де за істинного значення логічної умови, що продукуться предикатом $(D_i^{\pi})P_i(\alpha_i^3)$, виконується Д-оператор $(D_i)O_i(D_i')$, а за хибного – $(\tilde{D}_i)\tilde{O}_i(\tilde{D}_i')$. Коли логічна умова $\alpha_i^3=\mu$, виконується наступний предикат. Якщо логічна умова $\alpha_n^3=\mu$, то виконується Д-оператор $(D_{n+1})O_{n+1}(D_{n+1}')$.

Ще одним засобом контролю наявності можливих помилок в алгоритмах є контроль значень оброблюваних даних. Контролювати бажано практично всі дані. Зокрема, потрапляння значень оброблюваних даних у заданий діапазон, вихід індексів за межі масивів, розмір масивів, що піддаються обробці, тощо. Для цього можна користуватися вищезазначеними засобами, однак перевірка великого обсягу різноманітних даних призводить до численного розгалуження ОП, що суттєво й негативно впливає на розмір алгоритму і, головне, істотно погіршує його читабельність.

Для розв'язання зазначеної задачі та зменшення негативних наслідків її розв'язання введемо *k*-значні логічні умови [12]. Для цього розглянемо операцію *P*-диз'юнкції під час аналізу деяких *n*-відносини, заданого графіка

$$\rho^n_{A_1,\ldots,A_n}=\begin{bmatrix}a^{(1)}_{i_1},a^{(2)}_{i_1},\ldots,a^{(n)}_{i_1}\\a^{(1)}_{i_2},a^{(2)}_{i_2},\ldots,a^{(n)}_{i_2}\\\ldots\ldots\ldots\ldots\ldots\\a^{(1)}_{i_{k-1}},a^{(2)}_{i_{k-1}},\ldots,a^{(n)}_{i_{k-1}}\end{bmatrix},$$

яку запишемо у вигляді $[(\rho^n_{A_1,\ldots,A_n})P(\alpha P(\alpha)]_1)X_1(D_1')\vee(D_2)X_2(D_2'))$, де предикат – логічна функція $P:\rho^n_{A_1,\ldots,A_n}\to\alpha\in E_2=\{0,1\}$, за допомогою якої перевіряється виконання заданого

співвідношення. Ця операція, відповідно до її визначення, приведе до виконання Д-оператора $(D_1)X_1(D_1')$, якщо задане відношення виконується, і $(D_2)X_2(D_2')$ – в іншому випадку.

Уявімо графік *n*-відносин у вигляді таких підмножин $\rho_{k-1}^n \cup \rho_{k-2}^n \cup ... \cup \rho_1^n = \rho_{A_1,...,A_n}^n$, що

$$\rho_{k-1}^n = \begin{bmatrix} a_{i_1}^{(1)}, a_{i_1}^{(2)}, ..., a_{i_1}^{(n)} \\ a_{i_2}^{(1)}, a_{i_2}^{(2)}, ..., a_{i_2}^{(n)} \\ ..... \\ a_{i_{k-1}}^{(1)}, a_{i_{k-1}}^{(2)}, ..., a_{i_{k-1}}^{(n)} \end{bmatrix}, \rho_{k-2}^n = \begin{bmatrix} a_{i_1}^{(1)}, a_{i_1}^{(2)}, ..., a_{i_1}^{(n)} \\ a_{i_2}^{(1)}, a_{i_2}^{(2)}, ..., a_{i_2}^{(n)} \\ ..... \\ a_{i_{k-2}}^{(1)}, a_{i_{k-2}}^{(2)}, ..., a_{i_{k-2}}^{(n)} \end{bmatrix}, \ldots, \ \rho_1^n = [a_{i_1}^{(1)}, a_{i_1}^{(2)}, ..., a_{i_1}^{(n)}] \ .$$

Організуємо таку вкладеність p_2 -диз'юнкцій

$$[(\rho_{k-1}^n)P_{k-1}(\alpha_{k-1})]((D_{k-1})X_{k-1}(D_{k-1}') \vee [(\rho_{k-2}^n)P_{k-2}(\alpha_{k-2})]\ ((D_{k-2})X_{k-2}(D_{k-2}') \vee ...$$
$$... \vee [(\rho_1^n)P_1(\alpha_1)]((D_1)X_1(D_1') \vee (D_0)X_0(D_0'))...).$$

Результатами виконання зазначеної конструкції будуть *Д*-оператори

$(D_{k-1})X_{k-1}(D_{k-1}')$, якщо $\alpha_{k-1} = 1$;
$(D_{k-2})x_{k-2}(D_{k-2}')$, якщо $\alpha_{k-2} = 1$;
. . . . . .
$(D_1)X_1(D_1')$, якщо $\alpha_1 = 1$;
$(D_0)X_0(D_0')$, якщо $\alpha_1 = 0$.

Замінивши для предиката P_{k-1} у множині $E_2 = \{0,1\}$ константу 1 на *k*-1, для P_{k-2} – на *k*-2 і т. д., отримуємо $P_{k-1} : \rho_{k-1}^n \to \alpha_{k-1} \in E_2 = \{k-1,0\}$, $P_{k-2} : \rho_{k-2}^n \to \alpha_{k-2} \in E_2 = \{k-2,0\}$ тощо, а для предиката – $P_1 : \rho_1^n \to \alpha_1 \in E_2 = \{1,0\}$.

Після виконаної заміни логічних констант вкладеність P_2-диз'юнкцій породжує *Д*-оператори

$(D_{k-1})X_{k-1}(D_{k-1}')$, якщо $\alpha_{k-1} = k-1$;
$(D_{k-2})X_{k-2}(D_{k-2}')$, якщо $\alpha_{k-2} = k-2$;
. . . . . .
$(D_1)X_1(D_1')$, якщо $\alpha_1 = 1$;
$(D_0)X_0(D_0')$, якщо $\alpha_1 = 0$.

Ураховуючи, що значення логічної умови, отримане внаслідок виконання деякого предиката P_i, відповідно до визначення 1 є результатом виконання тільки цього предиката і втрачається в результаті виконання P_{i+1} предиката, логічні умови, що продукуються всіма предикатами в розглянутій конструкції, трактуватимемо як єдину умову, що набуває різних значень.

Виходячи з цього, введемо похідну операцію, що реалізує розглянуту алгоритмічну конструкцію в такий спосіб

$$[(D^{\pi}_{k-1};D^{\pi}_{k-2};...;D^{\pi}_{1})P(\alpha^{k})](D_{k-1})X_{k-1}(D'_{k-1})\vee(D_{k-2})X_{k-2}(D'_{k-2})\vee...$$
$$...\vee(D_{1})X_{1}(D'_{1})\vee(D_{0})X_{0}(D'_{0})=$$
$$=[(\rho^{n}_{k-1})P_{k\text{-}1}(\alpha^{2}_{k-1})]((D_{k-1})X_{k-1}(D'_{k-1})\vee[(\rho^{n}_{k-2})P_{k\text{-}2}(\alpha^{2}_{k-2})]\ ((D_{k-2})X_{k-2}(D'_{k-2})\vee...$$
$$...\vee[(\rho^{n}_{1})P_{1}(\alpha^{2}_{1})]((D_{1})X_{1}(D'_{1})\vee(D_{0})X_{0}(D'_{0}))...),$$

де $D^{\pi}_{k-1}=\rho^{n}_{k-1}, D^{\pi}_{k-2}=\rho^{n}_{k-2},\ldots, D^{\pi}_{1}=\rho^{n}_{1}$, $P(\alpha^{k})$ – предикат, який продукує логічні умови $\alpha^{k}\in E_{k}=\{k-1,k-2,...,1,0\}$. Отримана операція називається p_k-диз'юнкція й виконується так

$$[(D^{\pi}_{k-1};D^{\pi}_{k-2};...;D^{\pi}_{1})P(\alpha^{k})]((D_{k-1})X_{k-1}(D'_{k-1})\vee(D_{k-2})X_{k-2}(D'_{k-2})\vee...$$
$$...\vee(D_{1})X_{1}(D'_{1})\vee(D_{0})X_{0}(D'_{0}))=\begin{cases}(D_{k\text{-}1})X_{k\text{-}1}(D'_{k\text{-}1}),\ \text{якщо}\ \alpha^{k}=k-1;\\(D_{k-2})X_{k-2}(D'_{k-2}),\ \text{якщо}\ \alpha^{k}=k-2;\\.....\\(D_{1})X_{1}(D'_{1}),\ \text{якщо}\ \alpha^{k}=1;\\(D_{0})X_{0}(D'_{0}),\ \text{якщо}\ \alpha^{k}=0.\end{cases}$$

Таким чином, усі необхідні операції з контролю оброблюваних даних і забезпечення реакції на їх некоректність реалізуються в рамках єдиної операції p_k-диз'юнкції, що забезпечує компактність і читабельність опису алгоритму.

На множині k-значних логічних умов визначені відомі [10] узагальнені операції:

– інверсія $\overline{\alpha}^{k}=\alpha^{k}+1(\bmod k)$;

– диз'юнкція $\alpha^{k}_{1}\vee\alpha^{k}_{2}=\max(\alpha^{k}_{1},\alpha^{k}_{1})$;

– кон'юнкція $\alpha^{k}_{1}\wedge\alpha^{k}_{2}=\min(\alpha^{k}_{1},\alpha^{k}_{2})$.

Використання цих операцій розширює можливості контролю даних за рахунок виконання такого контролю для даних у різних поєднаннях, що може бути записано у вигляді

$$[({}^{1}D^{\pi}_{k-1};...;{}^{1}D^{\pi}_{1})P^{k}_{1}(\alpha^{k})\circ...\circ({}^{s}D^{\pi}_{k-1},...,{}^{s}D^{\pi}_{1})P^{k}_{s}(\alpha^{k})]((D_{k-1})X_{k-1}(D'_{k-1})\vee$$
$$\vee(D_{k-2})X_{k-2}(D'_{k-2})\vee...\vee(D_{1})X_{1}(D'_{1})\vee(D_{0})X_{0}(D'_{0})),$$

де $\circ$ – одна з можливих узагальнених логічних операцій.

Слід звернути увагу на такі особливості отриманої операції, що випливають з її побудови. Розбиття графіка відносин може виконуватися довільно, забезпечуючи, відповідно, аналіз довільних його підмножин. Аналогічне розбиття може бути виконано для одного з векторів графіка відносин. Отримана операція може бути декомпозована, наприклад, записана у такому вигляді:

$$[(D^{\pi}_{k-1})P(\alpha P(\alpha)]_{k\text{-}1})X_{k\text{-}1}(D'_{k\text{-}1})\vee([(D^{\pi}_{k-2};...;D^{\pi}_{1})P(\alpha^{k})](D_{k-2})X_{k-2}(D'_{k-2})\vee$$
$$\vee(D_{k-3})X_{k-3}(D'_{k-3})\vee...\vee(D_{1})X_{1}(D'_{1})\vee(D_{0})X_{0}(D'_{0}))).$$

При цьому очевидно, що можуть аналізуватися й бінарні відносини, а *n*-арні відносини зводяться до бінарних.

Для виконання задачі перевірки послідовності дій ОП, тобто перевірки його передісторії, продовжимо розширення логічної складової алгебри, для чого поняття фіксувальних логічних умов (фіксувальні умови) уведемо в такий спосіб.

Фіксувальними називатимемо множини логічних умов $\Phi^k = \{\varphi_1^k, \varphi_2^k, ...\}$, $\Phi^2 = \{\varphi_1^2, \varphi_2^2, ...\}$, $\Phi^2 = \{\varphi_1^2, \varphi_2^2, ...\}$, $\Phi = \{\varphi_1, \varphi_2, ...\}$), де $\varphi_i^k \in E_k = \{k-1, ..., 1, 0\}$, $\varphi_i^3 \in E_3 = \{1, \mu, 0\}$, $\varphi_i^2 \in E_2 = \{1, \mu\}$, $\varphi_i \in E_2 = \{1, 0\}$, що $\Phi^k, \Phi^3, \Phi^2, \Phi \subset D^{AC}$. Для будь-якого Д-оператора $(D)X(D')$ виконується $D' \cap \Phi^k = \varnothing$, $D' \cap \Phi^3 = \varnothing$, $D' \cap \Phi^2 = \varnothing$, $D' \cap \Phi = \varnothing$, за винятком спеціально виділених Д-операторів вигляду $(D)X(\varphi^k)$, $(D)X(\varphi^3)$, $(D)X(\varphi^2)$, $(D)X(\varphi L)$, які визначають (задають) значення фіксувальних умов.

Зазначимо, що фіксувальні умови завжди використовувалися в практичному програмуванні під іменами "прапорці", "ознаки" тощо. У даному разі вони формалізовані в контексті цієї праці.

Оскільки елементи множин Φ^k, Φ^3, Φ^2, Φ є логічними умовами, то вони можуть використовуватися в операціях алгебри.

Отже, на будь-якому етапі ОП може бути перевірено попередній його крок, на якому встановлено істинне або хибне значення фіксувальної логічної умови, й виконано дії, які є реакцією на коректність або помилковість чергового кроку ОП:

$$[\varphi\varphi]((\ _1)X_1(D_1') \vee (D_2)X_2(D_2'));$$
$$[\varphi\varphi](D)XD').$$

При цьому в результаті використання конструкції (5) може бути перевірено і коректність завдання використовуваної фіксувальної умови $[\varphi^2)]((D)X(D')*(\widetilde{D})\widetilde{X}(\widetilde{D}'))$.

Для контролю над послідовністю попередніх кроків можуть використовуватися *k*-значні фіксувальні умови, що встановлюються відповідно до необхідної послідовності виконання цих кроків. Реакція на порушення необхідної послідовності визначається отриманим значенням фіксувальної умови

$$[\varphi^k]((D_{k-1})X_{k-1}(D_{k-1}') \vee (D_{k-2})X_{k-2}(D_{k-2}') \vee ...$$
$$... \vee (D_1)X_1(D_1') \vee (D_0)X_0(D_0')).$$

Нарешті, допустиме сумісне використання фіксувальних умов і предикатів, наприклад:

$$[\varphi_1^k \circ ... \circ \varphi_p^k \circ (D_1^\pi)P_1(\alpha_1^k) \circ ... \circ (D_k^\pi)P_k(\alpha_k^k)]((D_{k-1})X_{k-1}(D_{k-1}') \vee .... \vee (D_0)X_0(D_0')),$$

що дозволяє здійснювати комплексний контроль за значеннями даних, які обробляються, та послідовністю їх обробки.

Таким чином, показано можливість розв'язання задачі перевірки наявності помилок в алгоритмі на етапі його виконання.

Висновки з даного дослідження і перспективи подальших розвідок у даному напрямку. Отримані результати, не претендуючи на реалізацію всіх аспектів захищеного програмування, дозволили розв'язати основні задачі, пов'язані з цією концепцією програмування.

Показано такі обмеження на використовувані алгоритмічні конструкції, які може бути перевірено на етапі проектування алгоритму.

Запропоновано *k*-значні, тризначні і фіксувальні логічні умови, а також побудовано на їх базі алгоритмічні конструкції, які забезпечують виявлення можливих помилок як у даних, так і в послідовності їх обробки. Ці засоби виявляють помилки в процесі функціонування програми й можуть (мусять) бути адаптовані як до етапу налагодження, так і до експлуатації програмного забезпечення.

Знаходження нових обмежень, а також розширення засобів контролю можливих помилок, зокрема з урахуванням розвитку алгебраїчного апарату й появи в його рамках нових засобів і методів розробки, є перспективним напрямом розвитку отриманих результатів.

Список використаних джерел:

1. Брукс Ф. П. Как проектируются и создаются программные комплексы / Брукс Ф. П. ; пер. с англ. – М. : Наука, 1979. – 150 с.
2. Дейкстра Э. Смиренный программист / Э. Дейкстра ; пер. с англ. // Лекции лауреатов премии Тьюринга за первые двадцать лет. 1966–1985. – М. : Мир, 1993. – 120 с.
3. Йордон Э. Путь камикадзе. Как разработчику программного обеспечения выжить в безнадежном проекте / Йордон Э. ; пер. с англ. – М. : Лори, 2000. – 272 с.
4. Глушков В. М. Перспективы автоматизации проектирования вычислительных машин / В. М. Глушков // Вестн. АН УССР. – 1967. – № 4. – С. 22–36.
5. Глушков В. М. Алгебра. Языки. Программирование / В. М. Глушков, Г. Е. Цейтлин, Е. Л. Ющенко. – 3-е изд., перераб. и доп. – К. : Наукова думка, 1989. – 376 с.
6. Акуловский В. Г. Алгебра алгоритмов, базирующаяся на данных / В. Г. Акуловский // Кибернетика и системный анализ. – 2012. – № 2. – С. 151–166.
7. Акуловский В. Г. Основы алгебры алгоритмов, базирующейся на данных / В. Г. Акуловский // Проблемы программирования : спец. выпуск по материалам седьмой Международной научно-практической конференции по программированию УкрПРОГ`2010. – 2010. – № 2–3. – С. 89–96.
8. Акуловский В. Г. Основы алгебраического аппарата для описания алгоритмов с данными / В. Г. Акуловский // Математическое и программное обеспечение систем в промышленной и социальной сферах : междунар. сб. науч. трудов. – Магнитогорск : Изд-во Магнитогорск. гос. техн. ун-та им. Носова. – 2011. – Ч. 1. – С. 9–14.
9. Макконнелл С. Совершенный код. Мастер-класс / Макконнелл С. – М. : Издательско-торговый дом "Русская Редакция"; СПб. : Питер, 2005. – 896 с.
10. Поспелов Д. А. Логические методы анализа и синтеза схем : 3-е изд., перераб. и доп. / Поспелов Д. А. – М. : Энергия, 1974. – 368 с.
11. Акуловський В. Г. Тризначна логіка в алгебрі алгоритмів / В. Г. Акуловський, В. В. Костенко // Вісник Академії митної служби України. – 2007. – № 1. – С. 83–88.
12. Акуловский В. Г. *K*-значная логика в расширенной алгебре алгоритмов / В. Г. Акуловский // Проблемы программирования : спец. выпуск по материалам шестой Международной научно-практической конференции по программированию УкрПРОГ` 2008. – 2008. – № 2–3. – С. 50–56.