УДК 681.3.06
DOI https://doi.org/10.32782/2663-5941/2023.2.1/36

Редько І.В.
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Зилевіч М.О.
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

РЕДУКЦІЙНЕ ПРОГРАМУВАННЯ ЗАДАЧ У ТЕХНОЛОГІЧНОМУ СЕРЕДОВИЩІ ПРОГРАМУВАННЯ

У статті обґрунтовано необхідність продуктивного осучаснення розуміння програмування, оскільки відповідно до традиційної індивідно-суб'єктивної парадигми, останне розглядеться як інструмент досягнення мети при цьому програмістська діяльність є максимально суб'єктиізованою, і опирається на вміння та навички суб'єкта, який нотує програмне рішення у код за допомогою мов програмування. Мова програмування виступає лише засобом нотації наслідку і про ніяку реальну підтримку генезису програм тут не йдеться.

Показано, що застосування продуктивного редукційного програмування має ключову роль у технологізації програмування. Використання останнього дозволяє зменшити складність процесу розробки програмного продукту та збільшити продуктивність програміста. Крім того, редукційне програмування може бути використане для оптимізації коду та покращення його читабельності, що є прикладом технологізації програмування та покращенні якості програмного продукту.

Запропоновано нову парадигму програмування, що передбачає активну участь суб'єкта програмування, який розглядає програмування як діяльність, що визначається програмою. Семантика програмного продукту та його синтаксична нотація обумовлюються концептом редукції та мовою програмування, яку обирає суб'єкт програмування. Таким чином, активізація ролі суб'єкта програмування покращує процес розробки та якість програмного продукту.

Показано, що концепти програмування можуть бути використані як семантичні шаблони у програмному ланцюгу для побудови певних класів програм.

Використано програмний дефінітор для трансляції композитів та базових функцій у синтаксичне представлення в рамках технологічної системи програмування. За допомогою редукційного програмування було побудовано програмну специфікацію та згенеровано відповідний код програми. Коректність специфікації випливає з її конструкції.

Ключові слова: *концепт, монада, редукція, композиція, середовище та система програмування, сутесутнісного уподібнення, дескриптор.*

Постановка проблеми. Відповідно до традиційної, індивідно-суб'єктивної парадигми, що асоціює програмування виключно з програмою, довгий час розуміння програмування виходило з того,що домінантним є його наслідок, який частіше за все трактувався як текст у тій чи іншій мові програмування. Саме ж програмування розглядалося як інструмент досягнення мети. В такій парадигмі програмістська діяльність є максимально суб'єктиізованою, і опирається на вміння та навички суб'єкта, який нотує програмне рішення у код за допомогою мов програмування. Тобто, мова програмування виступає лише засобом нотації наслідку програмування. Таким чином про ніяку реальну підтримку генезису програм тут не йдеться. Серед усіх відомих причин такого положення, на нашу думку, основною є занадто спрощене розуміння програмування, що не відповідає сучасним вимогам. Тому продуктивне осучаснення розуміння програмування є необхідною умовою реальної підтримки програмування.

В [1] обґрунтовано, що врахування активної ролі суб'єкта у такому осучасненні є засадничим. Ключову роль у цьому відіграє наступне основоположення: програмування це діяльність, обумовлена програмою та націлена на створення програми.

Таке розуміння програмування хоча й відрізняється від традиційного своєю спрямованістю на взаємодоповнення процесу програмотворення та його результату, разом з тим є поки що надто аморфним і потребує тому подальшого продуктив-

ного збагачення. Пропонована інтерсуб'єктивна парадигма [2] виходить з тлумачення терміну програми як подоби (нарису, як результату уподібнення) суттєвої риси. У [3-5] обґрунтовано, що таке трактування є, по-перше, цілком відповідним сучасній прагматиці програмування, по-друге, дозволяє здійснити подальше продуктивне збагачення програмування. Безпосередньо програмування розуміється як взаємодоповнення двох об'єктивно незвідних один до одного модального та реального (актуального) типів абстракцій – сутності – того, що може бути предметом розгляду, та суті – об'єкту розгляду, у сенсі того, що суть це сутність, що є (у наявності). Таким чином, отримуємо продуктивне збагачення вихідного основоположення: програмування (продуктивне) – діяльність, обумовлена програмним уподібненням (ПрУ). Тут ПрУ – обумовлене програмою та націлене на створення програми продуктивне збагачення сутесутнісного уподібнення (ССУ). Зміст ПрУ у першому наближення полягає у орієнтованому на створення програми як подоби сутності взаємодоповнення сутності та суті. Останнє, з огляду на згадану об'єктивну незвідність цих типів абстракції, вимагає залучення у цей процес суб'єкта з урахуванням (об'єктивізацією) його активної ролі у цьому. З зазначеного вище випливає важливість та необхідність розвитку інтерсуб'єктивного розуміння програмування, оскільки воно є запорукою реальної технологізації програмування.

Аналіз останніх досліджень і публікацій. Практика програмування свідчить про домінування парадигми «розділяй і володарюй» при вирішенні задач. І основним прийомом тут є редукція, цілісне розуміння якої зводиться до загального методичного прийому – зведення складного до простішого [6]. Тому, роль продуктивних редукційних механізмів є засадничою для технологізації програмування. В [1, 6-8] обґрунтовано, що основу таких збагачень складає концептопрограмне активно-пасивне взаємодоповнення:

$$\begin{cases} концепт = суть, що\ обумовлює\ сутність \\ програма = сутність, що\ обумовлюється\ концептом \end{cases}$$

В роботах [9-11], концептопрограмна платформа забезпечує реальну об'єктивізацію головного чинника продуктивної технологізації – активної ролі суб'єкта програмування. Здійснюється продуктивне збагачення програмного уподібнення як спеціального виду активно-пасивного причино-наслідкового зв'язку та його релятивізація – основний чинник продуктивної технологізації. Відповідно, технологічне середовище програмування доцільно розглядати як продуктивне збагачення згаданої концептопрограмної платформи програмних уподібнень до активно-пасивного взаємодоповнення двох об'єктивно незвідних один до іншого типів абстракції: замкненої оракульної логіки – цілісного ядра середовища програмування та відкритого різноманіття її продуктивних програмних уподібнень – технологічних систем програмування. Тобто будь-яка технологічна система програмування є наслідком програмної релятивізації і носієм продуктивного розуміння редукції. Це забезпечує реальне врахування активної ролі суб'єкта програмування.

Необхідність технологізації діяльності, осучаснення методів та способів її здійснення на пряму визначається рівнем потреби в об'єктивуванні участі суб'єкта у ній. У сфері програмування це проявляється у зростанні вимог до програмних продуктів та в усвідомленні того, що головні властивості останніх формуються стадії їх генезису і як наслідок визначаються активною роллю суб'єкта у ньому [12].

Визначальні основоположення розуміння технології програмування формуються на основі принципів обумовлення, підпорядкованості та відокремлюваності [4, 5]. З вказаних принципів слідують аспекти програм у їх взаємодоповненні. Цим обумовлюється точка зору на те, як повинна виглядати продуктивна технологія програмування (ТП), щоб її продукт відповідав цим принциповим вимогам. Зокрема, ці принципи на загальному рівні чітко окреслюють місце та роль продуктивного програмування в технології програмування

У [1, 5, 6, 8] вирішення будь-якої програмістської задачі представлена як послідовність виконання певних етапів, серед яких виділяється концептування через взаємодію оракулів. Схематизація оракулів та відповідно композитні обумовлення. Завершальним є редукціювання (редукційне концептування) яке є продуктивним саме у прагматиці технологізації програмування. Таким чином, показно, що в цілому етап програмування в технології програмування складається з сукупності визначених кроків. Сукупність таких кроків визначає завершення стадії програмування, останнім серед яких є редуціювання. На ньому доцільно закценувати більше уваги, оскільки воно безпосередньо передує етапу кодування.

Постановка завдання. Метою роботи є демонстрація технологічної системи програмування (ТСП) як платформи осучасненого редукційного програмування та застосування її для вирішення репрезентативних задач.

Виклад основного матеріалу дослідження. Програмування обумовлене суб'єктно-орієнтованим замиканням ТСП до суб'єкто-орієнтованої платформи програмування. Замикання являє собою визначення композитів як концептів програмування, базових предметних операцій та композито-композиційних інтерфейсів. Для демонстрації на прикладі побудуємо арифметичну ТСП. Композиційне програмування використаємо як платформу програмування, разом із іменною моделлю даних, функцій і операцій. Композитами будуть операції мультиплікування $\circ$, розгалуження IF, циклування WD і найпростіші похідні від них композиції (у сенсі операцій аплікації Ap та n-арної суперпозиції $S^n|_{n\in N}$), що уточнюють найбільш вживані способи синтезу одних програм з інших [13-15], а в якості базових предметних операцій – арифметичні операції $+,-,0$; логічні операції $\vee,\wedge,!$, T, F; відношення $=,\langle,\rangle$; функція слідування $s(n)=n+1|_{n\in N}$. Також знадобляться параметричні операції над іменними даними – іменування $A_{\downarrow}:A_{\downarrow}(a)|_{a\in N}=\{(A,a)\}$, розіменування $A^{\uparrow}:A^{\uparrow}\left(\{(A,a)\}|_{a\in N}\right)=a$ і відкриваюча та закриваюча дужки.

Надалі, під даними, функціями та операціями, якщо не зазначено інше, розуміємо іменні дані, іменні функції та іменні операції, відповідно. Композито-композиційний інтерфейс, визначається композитами апарату послідовної або $\circ$-, галуженої або IF - та циклічної або WD-редукцій (див.[2, 4, 5] та бібліографію). Кортеж функцій $\langle f_1,f_2,\ldots,f_s\rangle$ є $^\circ$ – редукцією функції f, якщо він є рішенням рівняння $f=x_1{}^\circ x_2{}^\circ\ldots{}^\circ x_s$, тобто $f\equiv f_1{}^\circ f_2{}^\circ\ldots{}^\circ f_s$. Пара функцій f_1,f_2 є IF-редукцією функції f, якщо існує такий предикат p, що ця пара є рішенням рівняння $f=IF(p,x_1,x_2)$, тобто $f\equiv IF(p,f_1,f_2)$. Також, функція g є WD-редукцією функції f, якщо існує такий предикат p, що g є рішенням рівняння $f=WD(x,p)$, тобто $f\equiv WD(g,p)$ [6, 9]. З останнього безпосередньо випливає корисна необхідна умова WD-редукційності.

Теорема. Для того, щоб функція g була WD-редукцією функції f необхідно, щоб виконувалась наступна рівність $g^\circ f=f$.

Після побудови технологічної системи, продемонструємо спосіб програмування у ній на прикладі програмування функції цілочисельного ділення. Програмування проводитимемо виходячи з наступної властивості функції: $\forall a,b,c|_{a,b,c\in N}\ div(a,b)=c$, де div – функція цілочисельного ділення натуральних чисел, а саме: $div:N\times N\to N$, де $div(a,b)$, таке натуральне число, що $b\times div(a,b)\le a\le b\times(div(a,b)+1)$. Враховуючи зорієнтованість описаної ТСП на іменні структури даних, збагатимо функцію її іменною специфікацією у вигляді іменної функції $DIV:\{(A,a),(B,b)\}\to\{C,c\}|_{a,b,c\in N}$. $DIV:\{(A,a),(B,b)\}\to\{(C,div(a,b))\}|_{a,b,c\in N}$.

Тобто, $DIV\equiv F_1{}^\circ F_2$. Дана специфікація є оракульною схемою [6], обумовленою композитом мультиплікування. З її іменної специфікації випливає, що $F_1,F_2,$, де $F_1=0\left(C^{\uparrow}\right)^\circ C_{\downarrow}\equiv\{(C,0)\}$, $F_2:\{(A,a),(B,b),(C,c)\}\to\{(A,a-k\times b),(B,b),(C,c+div(a,b))\}$, де $a,b\in N,b\ne 0$, а $k:(k+1)\times b<a<k\times b$. Тобто, $G\equiv F_1{}^\circ F_2$. Функції F_1 очевидно не потребує деталізації, оскільки є початковим обнуліннням значення с. Функція F_2 є оракульною схемою [6], з властивості функції div випливає, що WD-редукцією функції DIV буде функція $G:\{(A,a),(B,b),(C,c)\}\to\{(A,a-b),(B,b),(C,c+1)$, де $P:\{(A,a),(B,b)\}\to\begin{cases}T,якщо\ a\ge b\\F,якщо\ a<b\end{cases}$ – іменна специфікація відповідного предикату. Тобто, $F_2=WD(G,P)$. Очевидно, потрібно здійснити її програмування. Для цього скористаємось властивістю цієї функції:

$$div(a,b)|_{a,b\in N\,\&\,b>0}=\begin{cases}div(a-b,b)+1, a\ge b,\\div(a,b)=0, a<b\ або\ a=0\end{cases}$$

Таким чином, F_2 виглядає так: $F_2=WD\left(\left(A_{\downarrow}\left(A^{\uparrow}-B^{\uparrow}\right)\right)^\circ\left(C_{\downarrow}\left(C^{\uparrow}+1\right)\right),P(A^{\uparrow},B^{\uparrow}\right.$. Звідси, $DIV\equiv\left(0\left(C^{\uparrow}\right)^\circ C_{\downarrow}\right)^\circ(WD(\left(A_{\downarrow}\left(A^{\uparrow}-B^{\uparrow}\right)\right)^\circ\left(C_{\downarrow}\left(C^{\uparrow}+1\right)\right),P\left(A^{\uparrow},B^{\uparrow}\right)\cdot$

Така спрощена форма запису обрана задля легшого розуміння виразу. Також у формулах наведених вище присутні метавирази які не є реальними виразами в описаному середовищі. Задля спрощення форми запису дані метавирази вирішено не розписувати. Описані метавирази знайдут своє спеціальне відображення у відповідній таблиці дефінітора.

У результаті першої стадії технологізації, а саме редукційного програмування у заданій системі була отримана описана вище специфікація. З побудови програми випливає її коректність. Після отримання специфікації можливо провести кодування.

Зазначалося, що більшість мов програмування є лише засобами синтаксичної нотації результатів програмування. Технологія програмування близька по своїй суті до інтерпретованої мови та є імплементацією взаємодоповнення принципів програмування. За своєю суттю представляє собою суб'єкто-об'єктну технологію створення програмного продукту [16, 17]. Будь яку програму можна представити як мікроконвейер стадій, де стадія "програмування" реалізує підпорядкованість семантики прагматиці і результатом її є програма – обумовлений суб'єктом нарис рішення задачі у вигляді відповідного семантичного терму. [18, 19]. Стадія «кодування» – стосується синтаксичного аспекту

і результатом тут є код програми у вигляді синтаксично вірно записаного тексту у визначеній мові програмування . Для автоматизації процесу використовується відповідний дефінітор мови програмування. Застосуємо це до вище розглянутого прикладу програмування функції DIV.

Для спрощення, розглянемо тут лише невелику частину дефінітору Pascal-подібної мови програмування, яка є достатньою для демонстрації. У ньому представлені відповідні композити та функції з їх синтаксичними нотаціями Pascal-подібному вигляді (таб. 1 та 2).

У представлених таблицях позначення F, можливо з індексами, $F_i, i = 1,2,3,\ldots$ і тільки вони використовуються у якості нетермінальних символів або нетерміналів. Аналогічно, термінальні символи $X^{\uparrow}$, $X_{\downarrow}$ та X також можуть використовуватись з індексами: $X_i^{\uparrow}$, $X_{i\downarrow}$ та X_i, $i = 1,2,3,\ldots$

Через них забезпечується рекурсивність побудов [20]. Концепти програмування та кодування представлені у табл. 1 представляють собою правильно записані слова у об'єднаному алфавіті термінальних символів та нетерміналів. У табл. 2 наведено термінальні символи для базових операцій та відповідні їм Pascal-подібні коди.

Звернемося до вищенаведеної програми. Використана раніше додаткова розмітка програми наочно демонструє притаманну їй ієрархічність структури. Вона обумовлена здійсненою покроковістю актуалізацій оракулів у системі програмування, починаючи від оракула DIV і закінчуючи вільним від оракулів композиційним термом. Рухаючись по цій ієрархії у відповідності до заданого у таблицях 1 та 2 фрагменту дефінітора рекурсивно будуємо Pascal-подібний код програми (див. табл. 3).

Висновки. У результаті проведеної роботи показано, що продуктивна редукція грає фундаментальну роль у забезпеченні технологізації програмування.

Обґрунтовано, що нова парадигма програмування має базуватися на активізації ролі суб'єкта програмування, у якому програмування розглядається як діяльність, детермінована програмою.

Підтверджено, що технологія програмування використовує редукційне програмування як засіб перетворення інформаційного ресурсу у програмний продукт у технологічному середовищі програмування. Обумовлена концептом редукція відіграє засадничу роль у технологізації програмування. Концепт програмного продукту обумовлює семантику програмного продукту, а обумовлене програмою синтаксичну нотацію результатів програмування завершують однією з обраних суб'єктом програмування мовою програмування.

На репрезентативному прикладі продемонстровано використання концептів програмування у вигляді семантичних шаблонів як ланок програмного ланцюга, які обумовлюють певні класи програм. Використано програмний дефінітор, який виступає у ролі засобу трансляції композитів та базових функцій технологічної системи програмування у їх синтаксичні представлення.

За допомогою редукційного програмування у заданій системі була отримана програмна специфікація, коректність якої випливає з її побудови. На основі отриманої специфікації за допомогою дефініторів отримано код програми.

Таблиця 1

Шаблони програмування та кодування

Концепт (шаблони) програмування	Концепт (шаблони)кодування
...	...
F, (F)	F
$F_1 \circ F_2$	$begin\ F_1; F_2\ end$
$IF(F_1, F_2, F_3)$	$if\ F_1\ then\ F_2\ else\ F_3$
$WD(F_1, F_2)$	$while\ F_2\ do\ F_1\ end$
$F \circ X_{\downarrow}$	$X \coloneqq F$
$X^{\uparrow} \circ S \quad X^{\uparrow} \circ S \circ Y_{\downarrow}$	$X^{\uparrow} + 1 \quad Y \coloneqq X + 1$
$P(A^{\uparrow}, B^{\uparrow})$	$(A > B)\ or\ (A = B)$
$F_1\ [F_2]$	$F_1\ ; F_2$
meta $\begin{bmatrix} 0(C^{\uparrow}) \\ A_{\downarrow}(A^{\uparrow} - B^{\uparrow}) \\ C_{\downarrow}(C^{\uparrow} + 1) \end{bmatrix}$	$\begin{bmatrix} 0(C) \\ A \coloneqq A - B \\ C \coloneqq C + 1 \end{bmatrix}$
...	...

Таблиця 2

Кодування базових функцій

Базові функції	Коди базових функцій
...	...
0	0
+	+
−	−
∧	and
∨	or
!	not
<	<
=	=
>	>
$X^{\uparrow}$	X
$X_{\downarrow}$	X
...	...

Таблиця 3

Шаблони програмування та кодування

Програма	Використовувані шаблони	Актуалізації нетерміналів
$DIV \equiv (0(C^{\uparrow})\circ C_{\downarrow})\circ(WD(\left(A_{\downarrow}(A^{\uparrow} - B^{\uparrow})\right)\circ\left(C_{\downarrow}(C^{\uparrow}+1)\right), P(A^{\uparrow}, B^{\uparrow}))$	$F_1 \circ F_2$	$F_1 \Leftarrow 0(C^{\uparrow})\circ C_{\downarrow}$ $F_2 \Leftarrow WD(\left(A_{\downarrow}(A^{\uparrow} - B^{\uparrow})\right)\circ\left(C_{\downarrow}(C^{\uparrow}+1)\right), P(A^{\uparrow}, B^{\uparrow})$ $DIV \Leftarrow begin\ F_1; F_2\ end$
$F_1 \equiv 0(C^{\uparrow})\circ C_{\downarrow}$	$F_1 \circ F_2$ $X^{\uparrow}$ $X_{\downarrow}$	$F_{11} \Leftarrow 0(\mathrm{C})$ $F_{12} \Leftarrow C_{\downarrow}$ $F_1 \Leftarrow\ begin\ F_{11}; F_{12} end$
$F_2 = WD(\left(A_{\downarrow}(A^{\uparrow} - B^{\uparrow})\right)\circ\left(C_{\downarrow}(C^{\uparrow}+1)\right), P(A^{\uparrow}, B^{\uparrow})$	$WD(F_1, F_2)$	$F_{21} \Leftarrow \left(A_{\downarrow}(A^{\uparrow} - B^{\uparrow})\right)\circ\left(C_{\downarrow}(C^{\uparrow}+1)\right)$ $F_{22} \Leftarrow P(A^{\uparrow}, B^{\uparrow})$ $F_2 \Leftarrow\ while\ F_{22}\ do\ F_{21} end$
$F_{21} \Leftarrow \left(A_{\downarrow}(A^{\uparrow} - B^{\uparrow})\right)\circ\left(C_{\downarrow}(C^{\uparrow}+1)\right)$	$F_1 \circ F_2$ (F) $X^{\uparrow}$ $X_{\downarrow}$	$F_{31} \Leftarrow A_{\downarrow}(A^{\uparrow} - B^{\uparrow})$ $F_{32} \Leftarrow C_{\downarrow}(C^{\uparrow}+1)$ $F_{21} \Leftarrow begin\ F_{31}; F_{32} end$
$F_{22} \Leftarrow P(A^{\uparrow}, B^{\uparrow})$	meta $P(F_1, F_2)$	$F_{22} \Leftarrow (A > B\ or\ A = B)$
$F_{31} \Leftarrow A_{\downarrow}(A^{\uparrow} - B^{\uparrow})$	meta $A_{\downarrow}(A^{\uparrow} - B^{\uparrow})$ (F)	$begin$ $F_{31} \Leftarrow A \coloneqq A - B$ end
$F_{32} \Leftarrow C_{\downarrow}(C^{\uparrow}+1)$	meta $C_{\downarrow}(C^{\uparrow}+1))$ (F)	$begin$ $F_{32} \Leftarrow C \coloneqq C + 1$ end
$DIV \equiv F_1\circ(WD(F_{21}\circ F_{22}))$	$F_1 \circ F_2$ $WD(F_1, F_2)$ meta $C_{\downarrow}(C^{\uparrow}+1))$ meta $A_{\downarrow}(A^{\uparrow} - B^{\uparrow})$ meta $P(F_1, F_2)$ (F)	$begin$ $while\ (A > B\ or\ A = B)\ do$ $begin$ $F_{31} \Leftarrow A \coloneqq A - B$ $F_{32} \Leftarrow C \coloneqq C + 1$ end $DIV \Leftarrow F_{32}$ end

Список літератури:

1. І.В. Редько, П.О. Яганов “Концептуальна модель технологічного середовища програмування”, *Наукові вісті КПІ*, № 1, с. 18-26, 2020.
2. E. G. Husserl, “Logical Studies. Cartesian Reflections”, Minsk, Belarusia, 2000.
3. I. Redko, “Pragmatic foundations of descriptive environments”, Programming issues, no 3, pp 2-25, (in Russian), 2005.
4. I. Basarab, N. Nykytchenko, V. Redko. “Composite databases”. Kyiv: Lybid , 1992. p. 192.
5. I. V. Redko, D. I. Redko, T. L. Zakharchenko, “Conceptual basis of programming,” Kyiv, Ukraine: Komprynt, 2016.
6. І.В Редько, П.О. Яганов, М.О. Зилевіч, "Редукційне концептування оракульних схем", Системні дослідження та інформаційні технології, №1, с.21-33, 2021.
7. I. Redko, P. Yahanov and M. Zylevich, "Concept-Monadic Model of Technological Environment of Programming," 2020 IEEE 2nd International Conference on System Analysis & Intelligent Computing (SAIC), Kyiv, Ukraine, 2020, pp. 125-130.
8. Літературознавча енциклопедія : у 2 т.авт.-уклад. Ю. І. Ковалів. Київ : ВЦ «Академія», 2007. Т. 2 : М–Я. С. 309–310.
9. Редько В.Н. Дескриптологические основания программирования. Кибернетика и системный анализ. 2002. № 1. С. 3–19.
10. Редько В.Н. Основания дескриптологии. Кибернетика и системный анализ. 2003. № 5. С. 16–36.
11. Редько В.Н., Редько И.В., Гришко Н.В. Дескриптивные системы: концептуальный базис. Проблеми програмування. 2006. № 2–3. С. 75–80.
12. В.Н. Редько “Композиции программ и композиционное программирование”, Программирование, № 5, с. 3-24, 1978.
13. В.Н. Редько “Дефиниторы и метод дефиниторного процессирования”, Кибернетика, № 6, с. 52-56, 1974.
14. И. А. Басараб , Н. С. Никитченко, В.Н. Редько, *Композиционные базы данных*. – К.:Лыбидь, 1992.
15. В.Н. Редько, “Основы програмологии”, Кибернетика и систем. анализ, № 1, с. 35-57, 2000.
16. В.Н. Редько, “Основы композиционного программирования”, Программирование, № 3, с. 3-13, 1979.

17. В.Н. Редько, Н.В. Гришко, И.В. Редько, “Экспликативное программирование в среде логико-математических спецификаций ”, УкрПРОГ98, с.71-76, 1998.

18. И.В. Редько, Н.В. Гришко, “Экспликативное программирование в среде интеграции ”, Проблемы программирования, №2, с.59-65, 2004.

19. D. I. Redko, I. V. Redko, P. O. Yahanov and T. L. Zakharchenko, "Compositional basis in programmer activity," Системні дослідження та інформаційні технології, no. 4, pp. 83-96, 2015.

20. И.В. Редько, “Прагматические основания дескриптивных сред”, Проблемы программирования, №3, с.3-25, 2005.

Redko I.V., Zylevich M.O. REDUCTIVE PROGRAMMING OF PROBLEMS IN A TECHNOLOGICAL PROGRAMMING ENVIRONMENT

The article substantiates the need for a productive modernization of the understanding of programming. According to the traditional individual-subjective paradigm, programming is considered a tool for achieving the goal, with maximally subjectivized and relies on the skills and abilities of the person who notes the software solution in code using programming languages. The programming language acts only as a means of notation of the consequence and there is no real support for the genesis of programs here.

The fundamental role of productive reduction in the technologization of programming is shown. It is justified that the new paradigm of programming should be based on the activation of the role of the programming subject, in which programming is considered an activity determined by the program.

It is confirmed that programming technology uses reductive programming as a means of transforming an information resource into a software product in a technological programming environment. The reduction plays a fundamental role in the technologization of programming. The concept of the software product determines the semantics of the software product and the syntactic notation of the programming results determined by the program.

A representative example demonstrates the use of programming concepts in the form of semantic templates as links in a program chain that determine certain classes of programs. A software definer is used, which acts as a means of translating composites and basic functions of the technological programming system into their syntactic representations.

With the help of reductive programming, a program specification was obtained in the given system, the correctness of which follows from its construction. Based on the received specification, the program code is obtained with the help of definers.

Key words: *concept, monad, reduction, composition, environment and system of programming, essence likeness, descriptor.*