О. В. Копаєв
Черкаський національний університет
імені Богдана Хмельницького

Модельна сутність алгоритму

У сучасній методичній літературі, присвяченій навчанню інформатики, важко знайти матеріали, в яких би не згадувався алгоритмічний стиль мислення. Але ще важче знайти публікації, в яких би чітко та ясно тлумачилося це поняття. Разом з тим, саме формування та розвиток алгоритмічного стилю мислення є розвивальною метою навчання алгоритміки. Тому науково обґрунтоване уточнення змісту поняття «алгоритмічний стиль мислення» є одним із найважливіших завдань при з'ясуванні цілей та змісту навчання алгоритміки в сучасному курсі шкільної інформатики.

Алгоритмічний стиль мислення є невід'ємним складником алгоритмічної діяльності. Тому специфіку алгоритмічного стилю мислення, як і будь-якого іншого, принципово неможливо виявити без аналізу предметної галузі, де цей стиль себе виявляє. Власне, *алгоритмічний стиль мислення* ми визначаємо як систему мисленнєвих способів дій, прийомів, методів та відповідних їм мисленнєвих стратегій, що спрямовані на розв'язування теоретичних та практичних задач, і результатом яких є алгоритми як специфічні продукти людської діяльності. Під *алгоритмічною діяльністю* ми розуміємо діяльність, метою якої є створення, пізнання та перетворення алгоритму.

Таким чином, в контексті з'ясування сутності алгоритмічного стилю мислення та алгоритмічної діяльності постає питання про розгляд поняття *алгоритму* саме як *предмета* алгоритмічної діяльності.

Ознайомившись із інтерпретаціями поняття алгоритму у різних авторів з метою виявлення його предметної специфіки, можна помітити, що, як правило, в цих тлумаченнях алгоритм протиставляється алгоритмічному процесу. Наприклад, алгоритм розуміють як «*точний припис*, що визначає *обчислювальний процес*, який веде від вихідних даних, що можуть варіюватися, до очікуваного результату» [1, с.3] або «*текст*, який у визначених обставинах може привести до однозначного розвитку подій – *процесу* виконання алгоритму» [2, с.5], або «*правило, що указує дії*, в результаті ланцюжка яких від вихідних даних ми приходимо до шуканого результату. Такий ланцюжок дій називається *алгоритмічним процесом*, а кожна дія – його кроком»[1] [3, с.10] і т.п. Виходячи з цих тлумачень, логічно припустити, що між алгоритмом та алгоритмічним процесом існують певні відношення. Ми припустили, що ці відношення мають модельний характер.

У попередніх працях нами вже висловлювалася думка про те, що алгоритм можна розглядати як модель, а саме, – знакову модель алгоритмічного процесу [4, 5]. Проте зауважимо, що у тих публікаціях можливість такого розгляду детально не обґрунтовувалася. Думки щодо модельного характеру алгоритму трапляються й у інших дослідників методики навчання інформатики. Наприклад, у О. С Лєсневського: «Алгоритм, точніше його запис, є не що інше, як формальна модель дискретного процесу» [6, с.42]. О. А. Кузнецов стверджує, що «алгоритм – це інформаційна модель діяльності» [7, с.4]. У новій програмі середньої освіти з інформатики та інформаційних технологій Російської Федерації в основному змісті, який пропонується для засвоєння учнями, зазначено: «Алгоритм як модель діяльності» [8, с.23]. На жаль, у цих авторів, так само, як і в згаданій програмі думка про алгоритм як модель лише висловлюється, але не обґрунтовується і не розкривається.

Н. Г. Салміна, аналізуючи тлумачення поняття моделі різними авторами, приходить до висновку, що всі вони виділяють дві характеристики моделі: 1) модель – заміщувач об'єкта вивчення, 2) модель й виучуваний об'єкт перебувають у визначених відношеннях відповідності (і в цьому розумінні модель відображує об'єкт) [9, с.91]. Алгоритм і алгоритмічний процес – це різні об'єкти, причому вони мають різну природу. Разом з тим, вивчаючи алгоритм, можна зробити й певні висновки про алгоритмічний процес, який він задає. Адже алгоритм повністю детермінує алгоритмічний процес, а отже, й повністю його описує. Саме ці міркування дозволяють припустити, що алгоритм можна вважати моделлю алгоритмічного процесу.

Аналізуючи літературу, присвячену питанням моделювання як загальному методу пізнання та питанням навчання моделюванню, ми ознайомилися із працями Ю. А. Гастева , Б. В. Бірюкова та Є. С. Геллера, у яких, зокрема, зроблено висновок, що «… при всьому різноманітті змістів, в яких застосовується в літературі поняття моделі, ці змісти можуть бути охоплені в єдиному підході, в основу якого покладено поняття ізоморфізму і гомоморфізму. А саме, об'єкт (система елементів) A є моделлю об'єкта B тоді і лише тоді, коли існує такий гомоморфний образ A' об'єкта A і такий гомоморфний образ B' об'єкта B, що A' і B' між собою ізоморфні» [10, с.36].

Тому метою цієї статті є обґрунтування із залученням апарату теорії множин принципової можливості означення алгоритму як моделі алгоритмічного процесу.

Зауважимо, що, незважаючи на застосування у праці математичного апарату, обґрунтування не є строгим математичним доведенням. Основну увагу було звернено на змістовий перехід від «реальності» алгоритмів та алгоритмічних процесів до математичних абстракцій та навпаки.

Будемо вважати, що алгоритмічний процес – це процес, який є: 1) дискретним; 2) однозначно детермінованим; 3) потенційно скінченним та 4) перетворює певні конструктивні об'єкти. Він складається із елементарних дискретних актів або ж операцій. Кожна така операція вибирається з множини операцій певного алгоритмічного виконавця та актуалізується в конкретній обстановці протікання алгоритмічного процесу. Позначимо цю множину O:

$$O = \{o_1, \ldots, o_n\}, \quad n \in N.$$

Зауважимо, що ця множина є непорожньою, скінченною та задається перерахуванням усіх її елементів. У ході алгоритмічного процесу будь-яка з цих операцій може застосовуватися кілька разів, а

[1] Усі виділення наші – О. К.

тому їхні «копії» формально не розрізняються між собою. Розрізнення цих «копій» можна провести, пов'язавши з ними час початку їх виконання в межах алгоритмічного процесу, тобто утворивши пари виду $<o_l, t_i>$, де $o_l \in O$ – операція з множини операцій алгоритмічного виконавця $(1 \le l \le n)$, $t_i \in (0, T]$ – момент часу її застосування (виконання) в цьому процесі, а T – загальний час протікання всього процесу.

Оскільки усі пари такого роду є однорідними, але чітко розрізняються, а однією з характеристик алгоритмічного процесу є потенційна скінченність, то останній можна вважати скінченною множиною таких пар. Однорідність зазначених пар визначається належністю першої складової o_l до множини операцій одного й того ж алгоритмічного виконавця, а їх розрізнення визначається відмінністю другої складової t_i. При цьому, $t_{i+1} = t_i + \Delta t$, де Δt – різниця часу виконання «сусідніх» операцій. Для спрощення будемо вважати, що Δt – величина стала і дорівнює певній одиниці часу. Тоді, $t_{i+1} = t_i + 1$.

Алгоритмічний процес можна подати у вигляді графа (схема 1).

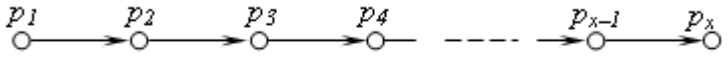

Схема 1. Подання алгоритмічного процесу у вигляді графа

Таким чином,

$$p_i = <0_1, t_i>,\ o_1 \in O,\ l = \overline{1,n},\ t_i = t_{i-1} + 1,\ i = \overline{1,x};$$
$$P = \{p_1, p_2, ..., p_x\},\ x \in N.$$

Пару виду $<o_l, t_i>$ (або p_i) назвемо *операцією алгоритмічного процесу*.

Задамо на множині P бінарне відношення $R_p = \{(p_i, p_j) \mid j \le i,\ i, j = \overline{1,x}\}$

Це відношення є:

1) транзитивне: із того, що $p_i R_P p_j$ і $p_j R_P p_k$ слідує, що $p_i R_P p_k$;
2) рефлексивне: для будь-якого p_i із P з того, що $p_i R_P p_i$ слідує, що $p_i R_P p_i$;
3) антисиметричне: із того, що $p_i R_P p_{i+1}$ і $p_{i+1} R_P p_i$ слідує, що $p_i = p_{i+1}$.

Отже, на множині операцій алгоритмічного процесу P визначене *відношення порядку*. Позначимо його R_P. Крім цього, на цій же множині визначимо *унарну операцію слідування*[2]

$$f_p : p_i \to p_{i+1};\ p_i, p_{i+1} \in P.$$

Оскільки на множині P визначені відношення порядку та операція слідування, то трійка $< P, R_P, f_P >$ є *алгебраїчною системою*[3]. Позначимо її S_P.

Якщо всі наведені вище викладки повторити для алгоритму (для спрощення вважаючи його лінійним), і першим елементом пари обрати деякий оператор opr_m $(1 \le m \le n)$ з множини операторів алгоритмічного виконавця (яку позначимо *OPR*), а у ролі другого елемента – порядковий номер n_j, ($n_j \in N$), цього оператора в алгоритмі, то отримаємо:

$$a_j = <opr_m, n_j>,\ opr_m \in OPR,\ m = \overline{1,n},\ n_j = n_{j-1} + 1;\ j = \overline{1,y};$$
$$A = \{a_1, a_2, ..., a_y\},\ y \in N.$$

Пару виду $<opr_m, n_j>$ (або ж a_j) назвемо *алгоритмічним оператором*.

На множині A теж визначене відношення порядку, яке позначимо R_A, та унарна операція

$$f_A : a_j \to a_{j+1},\ a_j, a_{j+1} \in A.$$

Тому трійка $< A, R_A, f_A >$ теж є алгебраїчною системою, яку позначимо S_A.

Алгоритм (лінійний) можна подати у вигляді графа:

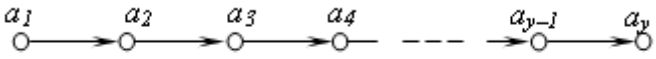

Схема 2. Подання лінійного алгоритму у вигляді графа

Як уже зазначалося, перебіг алгоритмічного процесу повністю детермінується відповідним алгоритмом. У випадку лінійного алгоритму кожному елементу (алгоритмічному оператору a_j) алгоритму A відповідає елемент (операція p_i) алгоритмічного процесу P, причому $i = j$, $x = y$:

$$a_1 \to p_1;\ a_2 \to p_2;\ \ldots\ a_y \to p_y,$$

і навпаки, кожному елементу алгоритмічного процесу P відповідає елемент алгоритму A:

$$p_1 \to a_1;\ p_2 \to a_2;\ \ldots\ p_y \to a_y.$$

Кожному елементу множини A поставлений у відповідність єдиний елемент множини P та, навпаки, – кожному елементу множини P поставлений у відповідність єдиний елемент множини A. Отже, має місце взаємооднозначне відображення зазначених множин (схема 3):

---

[2] Під операцією будемо розуміти відображення, яке зіставляє будь-якому впорядкованому набору n елементів даної множини визначений елемент іншої множини, яка в загальному випадку не співпадає з вихідною.

[3] Під алгебраїчною системою будемо розуміти множину з визначеними на ній операціями та відношеннями [11, с.126].

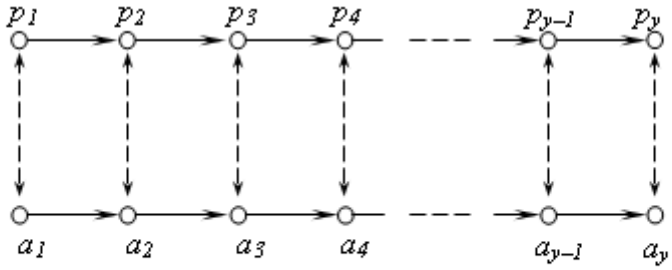

Схема 3. Взаємооднозначне відображення множини алгоритмічних операцій та множини операцій алгоритмічного процесу

$$P = \{p_1, p_2, \ldots, p_y\} \leftrightarrow A = \{a_1, a_2, \ldots, a_y\}.$$

Тому відношення порядку R_P, задане на множині елементів P, зберігається й для множини A. Також зберігається й операція слідування:

$$\text{Якщо } f_p : p_i \to p_{i+1};\ p_i, p_{i+1} \in P,$$

$$\text{то } f_A : a_i \to a_{i+1},\ a_i, a_{i+1} \in A.$$

Таким чином, множини A і P є ізоморфними[4]. Тому можна зробити висновок: *алгоритм та алгоритмічний процес є ізоморфними між собою*.

Проте цей висновок стосується лише випадку, коли кожний алгоритмічний оператор відповідає лише одній операції алгоритмічного процесу, а відношення, визначені на A, співпадають з відношеннями, визначеними на множині P, тобто для випадку лінійних алгоритмів. Насправді у переважній більшості випадків алгоритм є гомоморфним відображенням алгоритмічного процесу.

Скористаємося такими означеннями: «гомоморфізмом називають таку відповідність між двома системами об'єктів із визначеними для цих об'єктів відношеннями, при якій: 1) кожному об'єкту першої системи поставлений у відповідність лише один об'єкт другої системи, і кожному відношенню першої системи поставлене у відповідність лише одне відношення другої системи; 2) якщо для деяких об'єктів *a, b, c,*… першої системи виконується певне відношення *S* першої системи, то для об'єктів *a', b', c',*… другої системи, що відповідають об'єктам *a, b, c*, …, виконується відношення *S'* другої системи, яке відповідає відношенню *S*. Друга система об'єктів і відношень називається при цьому гомоморфним образом першої» [12, с.182]. При цьому ізоморфізм є частковим випадком гомоморфізму: «в тому частковому випадку, коли, по-перше, встановлена між двома системами, що розглядаються, відповідність взаємооднозначна і, по-друге, відношення *S'* виконується у другій системі між об'єктами *a', b', c'*… тоді й тільки тоді, коли відповідне відношення *S* виконується між відповідними об'єктами *a, b, c,*… першої системи, гомоморфізм називають ізоморфізмом» [12, с.182].

Таким чином, по-перше, ізоморфізм алгоритму та алгоритмічного процесу можливий лише у випадку лінійних алгоритмів, причому побудованих *лише з елементарних операторів*. Уже в разі застосування в алгоритмах складних операторів, а саме: повторення, розгалуження – відбувається структурне скорочення кількості елементарних алгоритмічних операторів шляхом їх групування. Одному й тому ж алгоритмічному оператору, який входить до складу групи повторення, можуть відповідати різні операції алгоритмічного процесу.

По-друге, при розробці алгоритмів часто застосовують абстракцію підпроцесів, які в алгоритмах також подаються логічно згрупованими елементарними алгоритмічними операторами (і в цьому розумінні є також складними операторами). Ці підпроцеси розглядаються на рівні алгоритму вже як *окремі логічно об'єднані й поіменовані дії* (підалгоритми або підпрограми). Тому на рівні алгоритму будь-який підалгоритм може розглядатися як абстракція елементарної дії, тобто як окремий алгоритмічний оператор. Оскільки в термінах конкретного алгоритмічного виконавця вони не є «справжніми» елементарними алгоритмічними операторами, то кожен підалгоритм завжди повинен бути «розгорнутий» в послідовність елементарних операторів. Таке «розгортання» – необхідна умова виконання алгоритму. Адже виконавець «знає» лише елементарні дії – операції.

Проілюструємо викладені вище міркування прикладом. Нехай необхідно побудувати алгоритм знаходження значення функції за формулою

$$C_n^m = \frac{n!}{m!(n-m)!},\ n, m \in N,\ m < n.$$

Фрагмент комп'ютерної програми, описаний мовою Pascal, що реалізує цей алгоритм, може мати такий вигляд:

```
C := Fact(N)/(Fact(M)*Fact(N - M));
```

Функція знаходження факторіалу Fact(x) може розглядатися як окрема елементарна дія (оператор) в межах алгоритму знаходження значення функції C_n^m. Але для практичного застосування є обов'язковою реалізація знаходження факторіалу шляхом застосування лише елементарних операторів алгоритмічного виконавця (в цьому випадку – операторів мови Pascal). Наприклад,

[4] «Ізоморфізм – взаємооднозначна відповідність між двома алгебраїчними системами, що зберігає операції й відношення між елементами» [11, с.39].

```
function Fact(const N: Byte): Integer;
var
  I: Byte;
  Result: Integer;
begin
  Result := 1;
  for I := 2 to N do
    Result := Result * I;
  Fact := Result;
end;
```

Таким чином, *хоча алгоритм за повним складом операторів алгоритмічного виконавця завжди ізоморфний алгоритмічному процесу, що здійснюється цим виконавцем, загалом його можна інтерпретувати як гомоморфний образ цього процесу*.

Досить вдалою ілюстрацією гомоморфізму алгоритму та алгоритмічного процесу з точки зору розуміння поняття алгоритму, яке прийняте в алгоритміці та програмуванні, є тлумачення гомоморфізму Д. Пойа: «Гомоморфізм є своєрідним систематичним перекладом. Оригінал не лише перекладається іншою мовою, але й скорочується таким чином, що результат перекладу і скорочення виявляється систематично рівномірно стиснутим. … Тонкощі при такому скороченні можуть бути і втрачені, але все наявне в оригіналі відображено у перекладі, й у зменшеному масштабі співвідношення зберігаються» [13, с.49].

Однак, інтерпретація алгоритму як гомоморфного образу алгоритмічного процесу аж ніяк не впливає на визначення алгоритму як моделі цього процесу. Справді, оскільки показано, що для безпосередньої реалізації алгоритмічним виконавцем алгоритм (система S_A) завжди повинен бути «розгорнутий» у лінійну послідовність алгоритмічних операторів, то завжди можна побудувати його гомоморфний образ (систему S_A'), яка, в свою чергу, є ізоморфною алгоритмічному процесу (системі S_P' як автоморфному образу системи S_P) (схема 4).

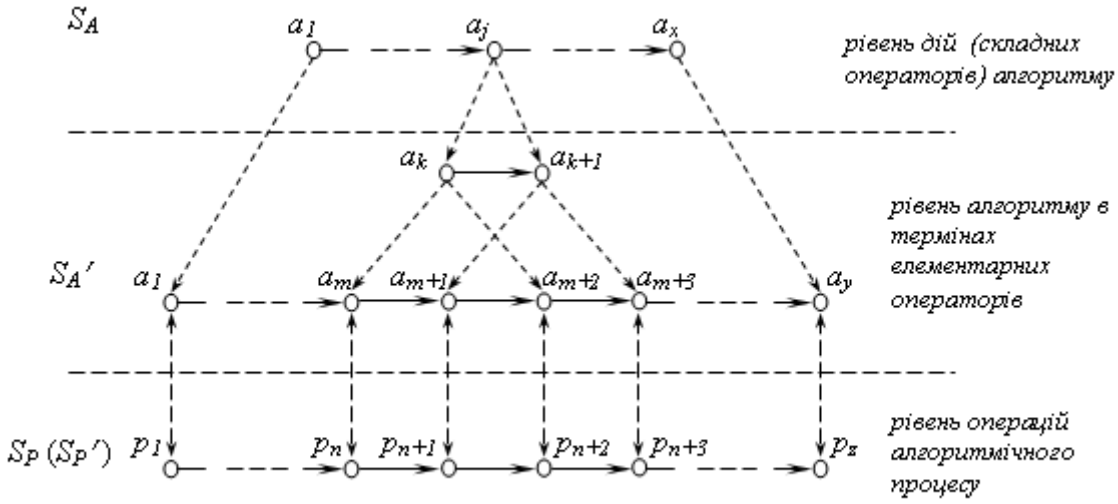

Схема 4. Гомоморфне відображення алгоритму на алгоритмічний процес

Таким чином,

$$S_A : A = \{a_1, \ldots, a_j, \ldots, a_x\},$$
$$S_A' : A' = \{a_1, \ldots, a_m, a_{m+1}, a_{m+2}, a_{m+3}, \ldots, a_y\},$$
$$S_P\ (S_P') : P = \{p_1, \ldots, p_z\} \mid y = x,$$
$$S_A' \xrightarrow{g_1} S_A \ \wedge\ S_P' \xrightarrow{g_2} S_P \ \wedge\ S_P' \xrightarrow{i} S_A',$$

де g_1, g_2 – гомоморфні відображення, а i – ізоморфне відображення.

Оскільки для системи S_A (алгоритму) можна побудувати гомоморфну їй систему S_A', а для системи S_P (алгоритмічного процесу) можна побудувати гомоморфну (в цьому випадку автоморфну[5]) систему S_P', причому S_P' і S_A' ізоморфні між собою, то система S_A є моделлю для системи S_P, тобто *алгоритм є моделлю алгоритмічного процесу*.

Єдине, що змінюється, – це взаємообернена симетрія відповідних елементів множин A та P, яка зникає у випадку гомоморфного відображення алгоритму на алгоритмічний процес. Тому алгоритм є моделлю алгоритмічного процесу, але алгоритмічний процес не може служити в ролі моделі алгоритму (що має місце у випадку їх ізоморфного відображення).

Отже, *у будь-якому випадку алгоритм можна вважати моделлю алгоритмічного процесу*. Тлумачення алгоритму як моделі дозволяє розглядати алгоритмічну діяльність як різновид діяльності моделювання. Це, в свою чергу дозволяє, по-перше, «звести» питання виявлення специфіки алгоритмічної діяльності до питань, пов'язаних із діяльністю моделювання, які значно краще досліджені в психології та педагогіці (наприклад, дослідження Давидова В. В., Салміної Н. Г., Фрідмана Л. М. і ін.), і, по-друге, застосувати в системі навчання алгоритміки поняття та методи, що прийняті в моделюванні.

ЛІТЕРАТУРА

[5] Автоморфізм – ізоморфізм певної системи об'єктів на себе [11, с.9].

1. Марков А. А. Теория алгорифмов // Тр. Мат. ин-та АН СССР им. В. А. Стеклова. – Т.42. – М.: Изд-во АН СССР, 1954. – С. 3-375.
2. Манин Ю. И. Вычислимое и невычислимое. – М.: Сов. радио, 1980. – 128 с.
3. Криницкий Н. А. Алгоритмы вокруг нас. – М.: Наука, 1977. – 224 с.
4. Копаєв О. В. Алгоритм як модель алгоритмічного процесу // Комп'ютерно-орієнтовані системи навчання. Випуск 6. – Київ, НПУ імені М. П. Драгоманова, 2003. – С. 206-213.
5. Копаєв О. В. Модельна концепція навчання алгоритміки // Інформаційні технології в науці, освіті і техніці / Матеріали IV Всеукраїнської конференції молодих науковців ІТОНТ-2004: Черкаси, 28-30 квітня 2004 р. – Черкаси: ЧНУ, 2004. – Ч 2. – С. 169-170.
6. Лесневский А. С. Об основных понятиях школьного курса информатики. – Информатика и образование. – 1994. – №2. – С. 41-44.
7. Кузнецов А. А., Бешенков С. А., Ракитина Е. А. Современный курс информатики: от концепции к содержанию // Информатика и образование. – 2004. – №2. – С. 2-6.
8. Примерная программа среднего (полного) общего образования по информатике и информационным технологиям // Информатика и образование. – 2004. – №4. – С. 20-26.
9. Салмина Н. Г. Виды и функции материализации в обучении. – М.: Изд-во Моск. ун-та, 1981. – 136 с.
10. Бирюков Б. В., Геллер Е. С. Кибернетика в гуманитарных науках. – М.: «Наука», 1973. – 383 с.
11. Микиша А. М., Орлов В. Б. Толковый математический словарь. – М.: Рус. яз., 1989. – 244 с.
12. Успенский В. А. Труды по нематематике. С приложениями семиотических посланий А. Н. Колмогорова к автору и его друзьям. В 2 т. Том 1. – М.: ОГИ, 2002. – 584 с.
13. Пойа Д. Математика и правдоподобные рассуждения. – М.: Наука, 1975. – 463 с.