ЗАСВОЄННЯ ЗАГАЛЬНОЇ СХЕМИ РОЗВ'ЯЗУВАННЯ ЗАДАЧ З ПРОГРАМУВАННЯ

Вакалюк Тетяна Анатоліївна,
старший викладач кафедри прикладної математики та інформатики Житомирського державного університету імені Івана Франка.

Анотація. У статті висвітлено проблему засвоєння загальної схеми розв'язування задач з програмування, пропонуються шляхи для кращого засвоєння даної схеми. Розглядається засвоєння загальної схеми розв'язування задач з програмування на прикладі однієї задачі.

Ключові слова: загальна схема розв'язування задач з програмування, постановка задачі, опис алгоритму, налагодження, тестування, програмні помилки, захисне програмування.

Застосування і розвиток сучасних інформаційних технологій у всіх сферах суспільного життя України згідно із Законом України «Про Основні засади розвитку інформаційного суспільства в Україні на 2007–2015 роки», прийнятим 09.01. 2007 р. № 537-V, зумовлює підвищення ролі освітянської галузі у підготовці й вихованні молодого покоління, а навчання школярів інформатики й інформаційно-комунікаційних технологій (ІКТ) стає одним із пріоритетних напрямів формування особистості випускника загальноосвітнього навчального закладу (ЗНЗ).

Цілі навчання шкільного курсу інформатики формулюються, базуючись на загальних цілях навчально-виховного процесу в загальноосвітніх навчальних закладах, а також на особливостях дисципліни інформатики як науки, її місці і ролі у житті сучасного інформаційного суспільства і в системі наук. На думку М. І. Жалдака: «Цілі та завдання навчання інформатики, як і будь-якого іншого шкільного предмету, пов'язані з формуванням основ наукового світогляду учнів, розвитком їх творчого та критичного мислення, здібностей та мотиваційної сфери, продовженням освіти, підготовкою до повноцінного життя у сучасному інформаційному суспільстві» [3, с. 16]. Реалізація поставлених цілей навчання інформатики неможлива без конкретних засобів навчання. До найвідоміших засобів навчання належать: підручник, засоби наочності, технічні засоби навчання і комп'ютер, у тому числі й програмне забезпечення. Отже, реалізація вищевикладених цілей навчання інформатики здійснюється також і на таких засобах навчання, як технічна база і програмне забезпечення, які розвиваються досить швидко. У складних економічних умовах неможливо регулярно купувати ліцензійне програмне забезпечення і покращувати конфігурацію комп'ютерів. Саме в цих умовах була і залишається основною темою предмету для навчання «Основи алгоритмізації та програмування», де реалізуються майже всі цілі, поставлені перед навчанням предмету «Інформатика». Спроби обійтись без цієї теми і готувати лише користувача ПК все одно призводять до необхідності деякого способу запису алгоритмів його дій, наразі цей опис може бути достатньо складним (опис логіки пошуку в базі даних).

Основною метою навчання розділу «Основи алгоритмізації та програмування» курсу «Інформатика» є формування у школярів навичок побудови алгоритмів. На думку М.І. Жалдака, одним з найважливіших компонентів основ інформаційної культури вчителя є «... володіння основами алгоритмізації, вміння добирати послідовність операцій і дій в діяльності ...».

Як свідчить досвід, базовою платформою для навчання змістової лінії основ алгоритмізації, є процедурні мови програмування, зокрема й Pascal. Це зумовлено тим, що саме ця мова програмування була створена Н. Віртом для опанування основ алгоритмізації та програмування і є оптимальною і зручною для навчання даного розділу.

Для з'ясування, якою мовою програмування для навчання теми «Основи алгоритмізації та програмування» курсу «Інформатика» користуються вчителі, було проведене опитування вчителів шкіл міста Житомира й області. Це опитування підтвердило, що найпоширенішою мовою програмування для навчання вищевказаної теми є мова програмування Pascal. Більшість учителів пояснили свій вибір тим, що ця мова є найпростішою для учнів під час навчання основ алгоритмізації.

Під час вивчення будь-якої мови програмування, у тому числі й Pascal, основним є засвоєння загальної схеми розв'язування задач із програмування.

Мета статті — опис і пояснення всіх компонентів загальної схеми розв'язування задач з програмування.

Виклад основного матеріалу. Загальну схему розв'язування задач із програмування зображено на рис. 1. Розглянемо кожний компонент даного алгоритму детальніше на прикладі однієї задачі.

Задача 1. Обчислити площу трикутника за відомими трьома сторонами [4, с. 63].

І. Постановка задачі. Для того, щоб скласти й реалізувати алгоритм обчислення площі трикутника за трьома сторонами, необхідно згідно до поставленої умови задач визначити вхідні та вихідні параметри, і, за потреби, деталізувати умову задачі: визначити, які дані допустимі; за яких умов можливе отримання допустимих результатів, а за яких — ні; які результати будуть вважатися правильними.

У даному випадку вхідними параметрами будуть сторони трикутника a, b, c, вихідними — площа S. Причому, усі (вхідні й вихідні) дані мають бути дода-

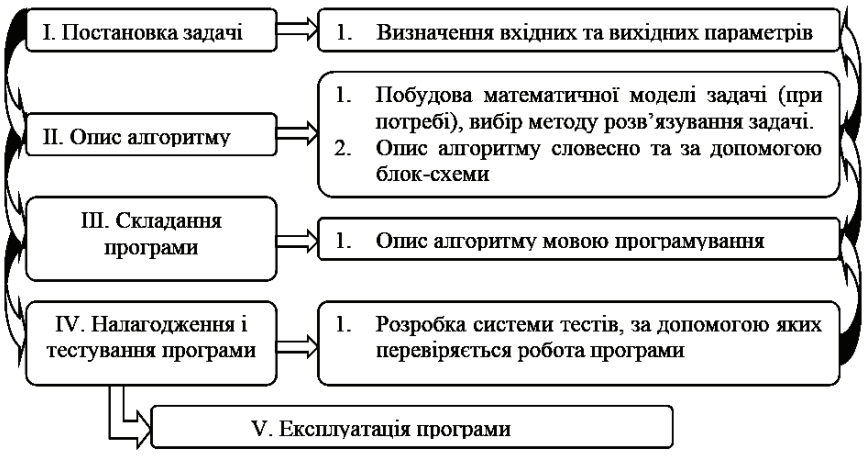

Рис. 1. Схема загального алгоритму розв'язування задач із програмування

тними, й обов'язково має виконуватись нерівність трикутника для кожної сторони: будь-яка сторона трикутника менша за суму двох інших. Чітко видно, що на даному етапі добре розвиваються такі мислительні операції, як аналіз та синтез, абстракція, а також розвиваються уміння виділення суттєвих властивостей предметів.

II. Опис алгоритму, що, у свою чергу, поділяється на два етапи.

1. Побудова математичної моделі задачі, вибір методу розв'язування задачі тощо: як відомо з курсу математики, площа трикутника за відомими трьома сторонами обчислюється за формулою Герона

$$S = \sqrt{p(p-a)(p-b)(p-c)},$$

де a, b, c — сторони трикутника, p — півпериметр, що обчислюється за формулою $p=(a+b+c)/2$.

На даному кроці розвиваються уміння знаходити головні зв'язки і відношення предметів і явищ навколишньої дійсності, що є необхідною умовою розвитку логічного мислення.

2. Опис алгоритму словесно й за допомогою блок-схеми.

Опишемо словесно алгоритм розв'язання даної задачі.

1. Початок програми.

2. Уведення вхідних даних: a, b, c — сторін трикутника.

3. Перевіряємо, чи всі дані додатні: якщо так, то перевіряємо ще одну умову — чи виконується нерівність трикутника для кожної сторони — і якщо нова умова виконується, то обчислюємо півпериметр за відомою формулою і площу за формулою Герона, після чого виводимо площу на екран; якщо ж принаймні одне з даних не є додатним або не виконується нерівність трикутника, то виводимо повідомлення на екран, що такий трикутник не існує, оскільки всі сторони мають бути додатними або не виконується нерівність трикутника.

4. Кінець програми.

Складаємо блок-схему (рис. 2) за описаним словесним алгоритмом.

Після складання блок-схеми, переходимо до наступного етапу.

III. Складання програми: написання програми мовою програмування Pascal (рис. 3). Для складання програми певною мовою програмування, потрібно володіти необхідним (базовим) обсягом знань.

IV. Налагодження і тестування програми: перевірка правильності роботи програми за допомогою тестів і виправлення виявлених помилок.

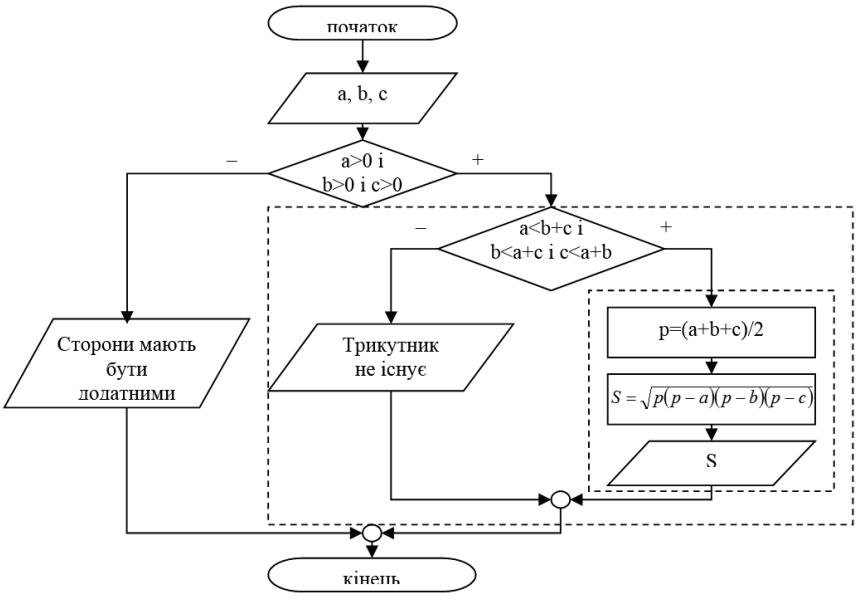

Рис. 2. Блок-схема алгоритму розв'язку задачі 1

```
var a,b,c, S, p : real;
begin {початок програми}
 write('Введіть сторони трикутника a,b,c '); {повідомлення на екран}
 readln(a,b,c); {зчитуємо вхідні дані}
 if (a>0) and (b>0) and (c>0) then {якщо кожна сторона додатна}
  if (a<b+c) and (b<c+a) and (c<a+b) then {якщо умова істинна}
    begin
      p:=(a+b+c)/2; {обчислюємо півпериметр}
      S:=sqrt(p*(p-a)*(p-b)*(p-c)); {обчислюємо площу}
      writeln('S=',S:7:2); {виводимо площу на екран}
    end
  else
    writeln('Трикутник не існує') {інакше трикутник не існує}
 else
   writeln('Сторони мають бути додатними'); {інакше виводимо
   на екран повідомлення, що сторони мають бути додатними}
end. {кінець програми
```

Рис. 3. Програма розв'язку задачі 1

Дамо визначення ключових слів даного етапу.

Налагодження (англ. debugging) — це пошук і виправлення помилок у розроблюваній програмі.

Програмні помилки, як правило, поділяються на три види (рис. 4).

Синтаксична помилка — неправильне використання синтаксичних конструкцій або помилка в написанні зарезервованих слів [5, с. 25]. Ці помилки виявляти найпростіше, адже компілятор сам виявить їх і вкаже на них.

Семантична помилка — помилка у програмі, яка пов'язана з неправильним змістом дій і використанням недопустимих значень величин [5, с. 25] (наприклад, помилки даних: ділення на 0, корінь з від'ємного числа тощо).

Логічна помилка — порушення логіки програми, яке призводить до неправильного результату [5, с. 25]. Подібні помилки ховаються в алгоритмах і потребують ретельного аналізу і всебічного тестування.

Для налагодження найчастіше використовують покрокове виконання програми, що забезпечує слідкування за значеннями змінних на різних етапах виконання програми.

Тому для зменшення ймовірності виникнення помилок, використовується *захисне програмування*. Вперше до цього поняття звертається Д. Ван Тассел [2, с. 207]. **Захисне програмування** — це стиль написання програм, за якого помилки, які з'являються, легко виявляються й ідентифікуються програмістом [2, с. 207]. Або іншими словами, *захисне програмування* — це стиль (методологія) написання програм, що зменшує ймовірність потрапляння помилок у програму.

Найпростіший метод використання захисного програмування полягає у тому, що під час написання програми потрібно передбачити опрацювання ситуацій, які не можуть статись ні за яких обставин. Для цього необхідно:

1. Перевіряти тип вхідних даних. Контролювати літерні поля, щоб переконатися, що вони не містять цифрових даних. Перевіряти цифрові поля на відсутність в них літерних даних.

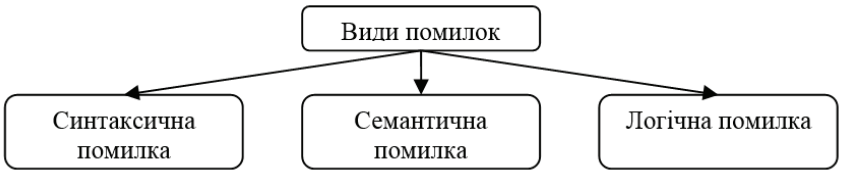

Рис. 4. Види програмних помилок

2. Здійснювати перевірку області значень змінних, щоб упевнитися, наприклад, що додатні величини завжди додатні.

3. Виконувати контроль правдоподібності значень змінних, які не повинні перевищувати деяких значень.

4. Контролювати підсумки обчислень шляхом введення всюди, де це можливо, перехресних підсумків і контрольних сум.

Тестування (англ. testing) — це випробування, перевірка.

Якщо ж вживати ширше поняття, то тестування — це виконання комплексу завдань для перевірки на правильну працездатність програми.

Наголосимо на основних принципах тестування програм, які полягають у:

1. *Використанні принципу захисного програмування* (див. вище).

2. *Тестуванні граничних умов* (так звані контрольні тести).

Основною ідеєю є те, що якщо трапиться помилка, то можна сказати з досить великою ймовірністю, що вона пов'язана з виходом за можливі граничні значення. Справедливе твердження і навпаки, коли програма працює правильно при усіх граничних значеннях тестових даних, то більш за все вона буде поводитись коректно й у нормальних умовах.

3. *Аналізі результатів тестування*. Це можна зробити кількома способами: для порівняння обчислити результат іншим способом (за допомогою іншої програми, яка виконує те ж саме, але має інший алгоритм виконання, на калькуляторі), використовуючи табличні дані тощо.

4. *Тестуванні окремих блоків незалежно один від одного*. Потрібно враховувати проміжні результати.

Описані принципи тестування програми у спільному використанні забезпечують правильне виконання програми. Виходячи з вищеописаних принципів тестування сам процес тестування програми можна розділити на два етапи перевірки: у нормальних і в екстремальних умовах (рис. 5).

Тест — це набір вхідних даних, для яких заздалегідь відомий результат.

Складемо для поставленої задачі 1 приклади тестів (табл. 1).

На цьому етапі розвивається така мислительна операція, як порівняння, розвиваються такі уміння: перевірити правильність міркування, виявити грубу логічну помилку, формуються такі здібності: робити предметом аналізу власну думку, а також уміння робити правильні висновки з фактів і перевіряти їх, доводити істин-

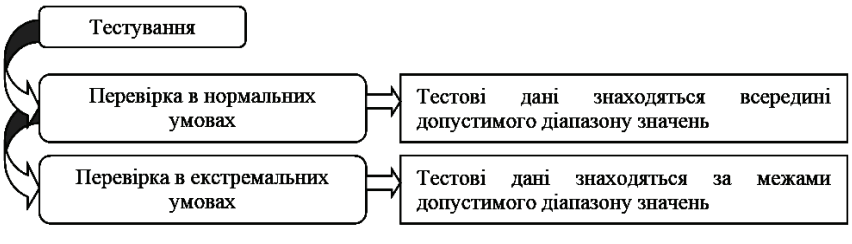

Рис. 5. Етапи процесу тестування

Таблиця 1

Приклади тестів до задачі 1

Вхідні дані	До яких умов належать вхідні дані
2 3 4	Нормальні умови
-1 2 3	Екстремальні умови (неправильні дані)
2 5 3	Екстремальні умови (неправильні дані)

ність своїх суджень і спростовувати хибні умовиводи, виробляється звичка перевіряти розв'язок задачі перед тим, як вважати його правильним.

V. Експлуатація програми: подальше використання розробленої програми і її супровід програмістами для нового налагодження за потреби, адже під час довготривалої експлуатації можуть змінитись вимоги до даної задачі або операційна система, на базі якої працює програма.

Висновки та перспективи подальших досліджень. Отже, у процесі засвоєння загального алгоритму розв'язування задач за допомогою ПК, у студентів формуються такі мислительні операції, як аналіз, синтез, порівняння, абстрагування, формуються особливості й уявлення про способи реалізації задач на практиці, окреслюються умови подальшої діяльності, формуються здібності мислити точно і послідовно, розвиваються уміння викривати логічні помилки.

Окреслені проблеми й отримані результати дають підстави стверджувати, що засвоєння загального алгоритму розв'язування задач із програмування є одним із пріоритетних напрямків досліджень, а також це питання надалі буде також актуальним і вимагатиме подальших досліджень.

★★★

Вакалюк Т.А. Усвоение общей схемы решения задач по программированию

Анотация. В статье освещена проблема усвоения общей схемы решения задач по программированию, предлагаются пути для лучшего усвоения данной схемы. Рассматривается усвоения общей схемы решения задач по программированию на примере одной задачи.

Ключевые слова: общая схема решения задач по программированию, постановка задачи, описание алгоритма, наладка, тестирование, программные ошибки, защитное программирование.

Література

1. *Вакалюк Т.А.* Підготовка майбутніх учителів інформатики до розвитку логічного мислення старшокласників : дис. … канд. пед. наук : 13.00.02 / Вакалюк Тетяна Анатоліївна. — Житомир, 2013. — 301 с.
2. *Ван Тассел Д.* Стиль, разработка, эффективность, отладка и испытание программ / Д. Ван Тассел. — [Пер. с англ.] — [2-е изд.]. — М. : Мир, 1981. — 320 с.
3. *Жалдак М.І.* Профільне навчання інформатики / М. І. Жалдак, Н. В. Морзе, О. Г. Кузьмінська // Комп'ютерно-орієнтовані системи навчання: збірник наукових праць. — [Відп. ред. М.І. Жалдак]. — 2004. — Вип. 8. — С. 13–18.
4. *Семакин И.Г.* Основы программирования : учебник / И. Г. Семакин, А. П. Шестаков. — М. : Мастерство, 2002. — 432 с.
5. *Степанченко И.В.* Методы тестирования программного обеспечения : учеб. пособие / И. В. Степанченко. — Волгоград, 2006. — 74 с.

★★★

ПРО ПЛАНУВАННЯ КУРСУ ІНФОРМАТИКИ В 5–6 КЛАСАХ ЗА НОВИМИ ПРОГРАМАМИ

Остапець Володимир Степанович,

учитель інформатики Щасливського навчально-виховного комплексу Бориспільського району Київської області, учитель-методист.

Тут мова буде про викладання інформатики згідно навчальної програми для 5–9 класів загальноосвітніх навчальних закладів з поглибленим вивченням предметів природничо-математичного циклу [1], хоча в заголовку вказано: «відповідно нових програм». Справа в тому, що ця програма була затверджена вслід за програмою [2], призначеною для загальноосвітніх навчальних закладів. По-перше, обидві програми реалізують ті самі ідеї нової концепції загальної середньої освіти та відповідного Державного стандарту базової і повної загальної середньої освіти. По-друге, вони обидві передбачають перехід до вивчення базового курсу інформатики в 5–9 класах. По-третє, у програмах [1] і [2] передбачена однакова кількість навчальних годин із деякими відмінностями у розподілі навчального часу між окремими темами і класами. По-четверте, за мету програми [1] у межах 5–6 класів не може ставитись поглиблене вивчення інформатики, адже враховуючи вік учнів та те, що програма передбачена для вивчення цього предмету в класах, де інформатика не вивчалась раніше, тут можливий лише пропедевтичний або вступний рівень розгляду навчального матеріалу.

Порівняльна характеристика програм [1] і [2] в обсязі 5-6 класів свідчить, що між ними якщо й існують відмінності у складності навчального матеріалу, то дуже незначні, адже курси 5–6 класів є фактично, як уже зазначено вище, пропедевтичними. Суттєва різниця між згаданими програмами полягає тільки в підходах до вивчення предмету інформатики, у [1] — чітко виражений змістово-компетентнісний, тоді, як у [2] — фактично чисто компетентнісний. Обидва підходи до вивчення навчальних предметів нині діють, очевидно, деякий час вони конкуруватимуть. Тут не ставиться за мету ні порівняння згаданих програм, ні їх протиставлення, але у контексті обговорення нової на даний час ідеї перенесення базового курсу інформатики з 9–11класів у 5–9 класи, цілком природно і правомірно розглянути шанси змістово-компетентнісногой підходу.

Безсумнівно, коли існує дозвіл на використання програми [1], то слід розробити й видати відповідні підручник та методичні рекомендації щодо його застосування, адже без цього програма не може бути впроваджена належно. За мету цих рядків ставиться ініціювання процесу масового впровадження названої програми.

Слід ще раз зауважити, що обидві програми, розраховані на учнів, які не вивчали інформатики до 5 класу, а в умовах, коли з 2013–2014 навчального року па-