ВИДАВНИЦТВО
РАНОК

Олена Бондаренко
Василь Ластовецький
Олександр Пилипчук
Євген Шестопалов

НОВА УКРАЇНСЬКА ШКОЛА

ПРОЄКТ

# інформатика

## 5 клас частина 4

НОВА УКРАЇНСЬКА ШКОЛА

Олена Бондаренко
Василь Ластовецький
Олександр Пилипчук
Євген Шестопалов

# Інформатика

## 5 клас

ПРОЄКТ

**Навчальний посібник**

Частина 4

Харків
Видавництво «Ранок»
2022

УДК 004:37.016(075.3)
І-74

**Авторський колектив:**
Олена Бондаренко, Василь Ластовецький,
Олександр Пилипчук, Євген Шестопалов

**Схвалення для використання в освітньому процесі**
у закладах загальної середньої освіти, які беруть участь в інноваційному освітньому проєкті всеукраїнського рівня за темою «Розроблення і впровадження навчально-методичного забезпечення для закладів загальної середньої освіти в умовах реалізації Державного стандарту базової середньої освіти» у 2021/2022 навчальному році, підтверджується рішенням експертної комісії з інформатики (https://www.ranok.com.ua/grifi_mon.html)

Створено відповідно до модельної навчальної програми «Інформатика. 5–6 класи» для закладів загальної середньої освіти (авт. Ривкінд Й. Я., Лисенко Т. І., Чернікова Л. А., Шакотько В. В.)

І-74 **Інформатика**. 5 клас : навч. посіб. Ч. 4 / Олена Бондаренко, Василь Ластовецький, Олександр Пилипчук, Євген Шестопалов. — Харків : Вид-во «Ранок», 2022. — 80 с.

ISBN 978-617-09-7397-9

Посібник поєднує в собі функції підручника та робочого зошита й містить теоретичний матеріал, вправи для самостійного виконання, практичні роботи. Посібник є складовою навчально-методичного комплекту, який підтримується інтерактивною освітньою платформою IZZI, що забезпечує доступ до тестових завдань за кожною темою.

Призначений для використання під час уроків інформатики в 5 класі закладів загальної середньої освіти та для інших форм навчання.

**УДК 004:37.016(075.3)**

**Інтернет-підтримка на освітній платформі**

Модельна навчальна програма «Інформатика. 5–6 класи» для закладів загальної середньої освіти (автори Й. Я. Ривкінд, Т. І. Лисенко, Л. А. Чернікова, В. В. Шакотько)



ISBN 978-617-09-7397-9

# РОЗДІЛ 5
# АЛГОРИТМИ ТА ПРОГРАМИ

# ПОВТОРЮЄМО

На уроках інформатики в молодшій школі ви ознайомилися з *поняттям алгоритму* й навчилися *складати алгоритми* для розв'язування різних задач. Ви вже знаєте, що алгоритм можна подавати *словесним* і *графічним* способами, а також *у вигляді програм*, записаних певною мовою програмування.

*Мова програмування* — це штучна мова, яка є системою позначень і правил для запису алгоритмів у формі, придатній для їх виконання на комп'ютері. Для зручної розробки програм існують спеціальні засоби їх створення — *системи програмування*. Усе, що ми робимо за допомогою комп'ютера — від пошуку інформації в інтернеті до комп'ютерної гри — можливо завдяки програмному коду, написаному програмістами. Ви вже ознайомилися з основними поняттями програмування, працюючи в програмному середовищі Scratch.

1. Що таке алгоритм?
2. Що таке програмування?
3. Що таке мова програмування?
4. Які способи опису алгоритму ви знаєте?
5. З якими мовами програмування ви вже знайомі?
6. Наведіть приклади дій, які можна виконувати за допомогою комп'ютера.

*Опрацювавши цей розділ, ви ознайомитеся з популярною сучасною мовою програмування Python, якою можна створювати програми для розв'язування різних задач, таких як розробка ігор, створення сайтів, виконання обчислень тощо.*

# § 18. Алгоритм і його властивості

## Поняття алгоритму

Згадаймо вже знайоме вам поняття алгоритму.

**Алгоритм** — це чітка послідовність указівок на виконання дій, спрямованих на розв'язування певного завдання.

Слово «алгоритм» походить від імені арабського математика Аль-Хорезмі (800–847 рр.). *Абу Абдулла Абу Джафар Мухаммад ібн Муса аль-Хорезмі* (рис. 18.1), який сформулював правила 4 арифметичних дій над багатоцифровими числами. Латиною ім'я автора європейці писали як «Algorithmi». Із часом алгоритмами почали називати способи розв'язування найрізноманітніших задач.

Рис. 18.1

У повсякденному житті людина стикається з алгоритмами, що визначають послідовність дій різної природи. Так, із курсу математики вам відомі алгоритми виконання арифметичних операцій над багатоцифровими числами та ін.

**1** Алгоритмами є рецепти приготування страв, порядок підготовки автоматичного пристрою до використання, а також, наприклад, розпорядок дня (рис. 18.2) тощо.

Рис. 18.2

Для багатьох ігор існують алгоритми виграшу, якщо результат гри залежить не від випадкового збігу обставин, а від кмітливості гравця й попередніх розрахунків.

**2** Гра Баше. Є 11 предметів.
За один хід гравець може взяти 1, 2, 3 предмети.
Програє той, кому дістанеться останній предмет.
Алгоритм виграшу для першого гравця:
*1-й хід:* узяти два предмети.
*2-й хід і далі:* брати стільки предметів, щоб кількість предметів, узятих разом із суперником за черговий хід, у сумі складала 4.

Уміння складати алгоритми є дуже важливим для людини будь-якої професії.

## Властивості алгоритму

Алгоритм складається з окремих кроків, які потрібно виконати в певному порядку. Якщо порушити порядок виконання кроків або пропустити якийсь крок, то алгоритм не буде виконаний до кінця або призведе до неправильного результату.

**3** Розглянемо алгоритм відмикання дверей.

1. Дістань ключ.
2. Встав ключ у замкову шпарину (рис. 18.3).
3. Двічі поверни ключ за годинниковою стрілкою.
4. Вийми ключ.

Чи відкриємо ми двері, якщо поміняти місцями вказівки 2 і 3?


Рис. 18.3

Алгоритм має певні *властивості*. Щоб алгоритм виконав своє призначення, його необхідно будувати за певними правилами.

**Дискретність** означає, що алгоритм повинен складатися з окремих кроків, кожний з яких має завершуватися.

**Визначеність** означає однозначність тлумачення правил виконання кроків і порядку їх виконання. Алгоритм не повинен містити команди, які можуть сприйматися виконавцем неоднозначно.

**Виконуваність (зрозумілість)** означає, що алгоритм, призначений для певного виконавця, може містити лише ті команди, які виконавець здатний виконати. Так, алгоритм для виконавця Першокласник не може містити команду «Побудуй бісектрису даного кута», хоча така команда може бути в алгоритмі, призначеному для виконавця Восьмикласник.

**Скінче́нність** — обов'язкова виконуваність алгоритму. Алгоритм має складатися зі скінченної кількості кроків, кожен з яких потребує для свого виконання скінченного проміжку часу.

4 Наведена послідовність команд є нескінченною.

1. Візьміть число 2.
2. Помножте задане число на 10.
3. Додайте до результату 5.
4. Якщо одержано додатне число, то перейдіть до команди 3, якщо ні, то припиніть виконання алгоритму.

**Масовість** означає можливість виконання алгоритму для різних вхідних даних. Наприклад, ви вивчали алгоритм знаходження коренів рівняння виду $ax + b = c$ для розв'язування рівнянь $5x + 3 = 8$, $17x + 6 = 40$ та інших.

**Результативність** означає, що після виконання послідовності вказівок алгоритму має бути отримано конкретний результат. Наприклад, послідовність вказівок «налий води; увімкни плитку; вимкни плитку» не є результативною, якщо потрібно було нагріти воду.

**Формальність** означає, що будь-який виконавець, здатний сприймати й виконувати вказівки алгоритму (навіть не розуміючи його змісту), може виконати завдання за заданим алгоритмом. Як відомо, автоматизовані пристрої правильно розв'язують багато задач за заданими їм алгоритмами, хоча змісту задач вони, безумовно, розуміти не можуть.

## Питання для самоперевірки

1. Назвіть основні властивості алгоритмів і поясніть суть кожної з них.
2. Розгляньте заданий алгоритм.
   1. Прочитайте число a1.
   2. Прочитайте число a2.
   3. Поділіть число a1 на число a2.
   4. Запишіть результат.

   Чи має цей алгоритм властивості масовості та визначеності?
3. *Задача*: задане число, більше за 1, зменшити до 1 шляхом ділення на 2. Алгоритм розв'язування задачі:
   1. Поділіть задане число на 2.
   2. Якщо результат не дорівнює 1, то виконайте команду 1, інакше припиніть виконання алгоритму.

   Чи має цей алгоритм властивість скінченності?
4. Чи можна скласти алгоритми розв'язування таких задач:
   а) знайти корінь рівняння $ax + b = c$;
   б) відвідати театр;
   в) вивести новий сорт пшениці;
   г) сконструювати машину для виконання домашніх завдань?
5. Ознайомтеся з наведеним алгоритмом отримання окропу.
   1. Налийте воду в чайник.
   2. Відкрийте кран газової конфорки.
   3. Поставте чайник на плиту.
   4. Почекайте, поки вода закипить.
   5. Піднесіть запалений сірник до конфорки.
   6. Вимкніть газ.

   Визначте правильну послідовність дій, яка дозволить запобігти нещасному випадку.

6*. *Задача*: перевізнику потрібно човном переправити через річку вовка, козу й капусту по одному. Опишіть алгоритм дій перевізника, виходячи з того, що небезпечно залишати разом без нагляду козу й капусту, вовка й козу.

# Вправа 18. Алгоритм і його властивості

Завдання: скласти алгоритм для розв'язування задачі.

**1** Як, маючи два відра ємністю 2 і 5 л, набрати з водопровідного крана 1 л води? Складіть алгоритм розв'язування задачі. Впишіть у прямокутники на рисунку об'єм води, яку потрібно набрати або яка залишатиметься у відрах на кожному кроці алгоритму.

3 бали

| Крок 1 | Крок 2 | Крок 3 | Крок 4 |
|---|---|---|---|
|   |   |   |   |

**2** Як із річки принести 6 л води, якщо є лише два відра ємністю 4 і 9 л? Складіть алгоритм розв'язування задачі.
Впишіть у прямокутники на рисунку об'єм води, яку слід наливати або яка залишатиметься у відрах на кожному кроці алгоритму.

4 бали

| Крок 1 | Крок 2 | Крок 3 | Крок 4 | Крок 5 |
|---|---|---|---|---|
|   |   |   |   |   |

**3*** Хлопець, який фарбує паркан, має 12 л фарби. Є дві посудини ємністю 8 і 5 л. Яким чином налити 6 л фарби в посудину ємністю 8 л? Складіть алгоритм розв'язування задачі. Впишіть у таблицю об'єм води, наявний на кожному кроці алгоритму.

5 балів

| Відра | Крок 1 | Крок 2 | Крок 3 | Крок 4 | Крок 5 | Крок 6 |
|---|---|---|---|---|---|---|
|  | | | | | | |
|  | | | | | | |
|  | | | | | | |

**Комп'ютерне тестування**

Виконайте тестове завдання 18 з автоматичною перевіркою результату.

# § 19. Виконавець алгоритмів і система його команд

Кожен алгоритм орієнтовано на певного виконавця. **Виконавцем алгоритму** може бути людина, тварина, комп'ютер, система людина — машина, система верстат — автомат тощо, яких «навчено» виконувати вказівки алгоритму. Якщо виконавцем є деякий пристрій, то вираз «виконавця навчено виконувати вказівку» означає, що пристрій може виконати задану вказівку автоматично, без зовнішнього втручання.

Отже, **виконавець** — це об'єкт, здатний діяти за заданим алгоритмом. Щоб скласти алгоритм, орієнтований на певного виконавця, необхідно знати характеристики цього виконавця.

## Характеристики виконавця

Кожен виконавець існує в певних умовах (середовищі) і може виконувати певний набір дій (рис. 19.1). Перш ніж складати алгоритм розв'язування задачі, слід дізнатися, у якому середовищі які дії може виконувати виконавець і за допомогою яких команд.

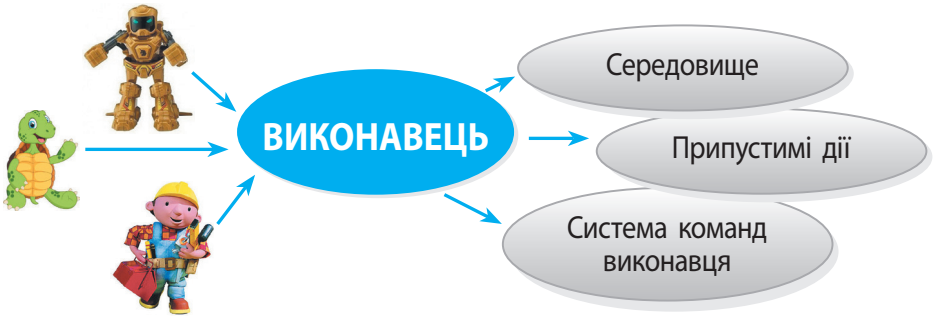



Рис. 19.1

**Середовище виконавця** — «місце існування» виконавця.

**Припустимі дії** — обмежений набір дій, що вміє виконувати певний виконавець. Описати виконавця означає вказати його припустимі дії. Досяжні цілі — результати, які виконавець може отримати при виконанні припустимих дій.

**Система команд виконавця** — повний перелік команд, за якими виконавець може виконати одну або серію припустимих дій. Виконавця можна уявити як пристрій із кнопковим керуванням. Натискання кнопки означає виклик однієї команди.

**Відмова** — подія, що виникає в разі виклику команди в неприпустимому для цієї команди стані середовища.

Розглянемо «поведінку» виконавця Кресляр, який призначений для побудови малюнків на полі розміром 5×5 клітинок (рис. 19.2) і вміє виконувати три команди: підніми перо, опусти перо, перейди до точки (*x*,*y*).

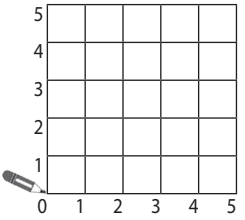



Рис. 19.2

На початку роботи Кресляр розташовується в точці (0,0) і тримає перо піднятим. Якщо перо опущене, під час пересування Кресляра за ним залишається слід.

Припустимим діям виконавця відповідає система команд:

| Система команд виконавця | Дії за командами |
|---|---|
| Підніми перо | Підготуватися до переміщення без сліду |
| Опусти перо | Підготуватися до переміщення зі слідом |
| Перейди до точки $(x,y)$ | Перейти до точки з координатами $(x,y)$ |

Відмова Кресляра виникає, якщо він отримує команду перейти до точки, яка розміщується за межами поля.

**1** Складемо для виконавця Кресляр алгоритм побудови даху будиночка (рис. 19.3).

1. Перейди до точки (0,3).
2. Опусти перо.
3. Перейди до точки (2,5).
4. Перейди до точки (3,5).
5. Перейди до точки (5,3).
6. Перейди до точки (0,3).
7. Підніми перо.

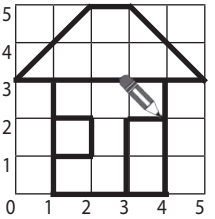



Рис. 19.3

Виконуючи алгоритм, виконавець може не розуміти смисл того, що він робить, і тим не менше отримувати потрібний результат, тобто виконавець діє формально (рис. 19.4).

**Комп'ютер — формальний виконавець**

**Помилки робить не комп'ютер, а розробник алгоритму**

Рис. 19.4

### Питання для самоперевірки

1. Поясніть поняття «виконавець алгоритму». Перелічіть характеристики виконавця.
2. Опишіть систему команд виконавця, щоб реалізувати алгоритм обчислення за формулою $p=(1+x):(1-x)$.
3. Якими припустимими діями ви оснастили б автомат, що заміняє:
   а) касира в магазині; б) двірника; в) вахтера?

4. Виконавець уміє виконувати команди: помнож число на 2; знайди суму двох чисел. Складіть для виконавця алгоритм обчислення виразу $y = 2a + 3b + 4c$ для заданих $a$, $b$ і $c$.

5*. Виконавець уміє виконувати команди: помнож число на 2; збільш число на 1. Складіть для виконавця алгоритм отримання:
а) числа 4 з 1; б) числа 5 з 1; в) числа 100 з 1.

6*. Фірма «Електронні прилади» випустила автоматизовану ванну, пульт керування якої має дві кнопки: Долити 5 л і Злити 3 л. Складіть алгоритм доливання у ванну 4 л води за якомога меншу кількість команд.

## Вправа 19. Виконавець алгоритмів і система його команд

Завдання: скласти алгоритм, використовуючи систему команд виконавця.

**1** Ознайомтесь, які команди може виконувати виконавець Кресляр: 1 бал

| Система команд виконавця | Дії за командами |
|---|---|
| Зроби крок<br>Повернися | Пройти 1 см, залишаючи слід<br>Повернутися на 90 ° ліворуч |

**2** Складіть для виконавця Кресляр алгоритм малювання (рис. 19.5). До початку малювання Кресляр спрямований вправо (→). 3 бали

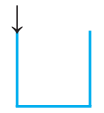


Рис. 19.5

| | |
|---|---|
| 1. | 4. |
| 2. | 5. |
| 3. | 6. |

**3** Складіть для виконавця Кресляр алгоритм малювання за рис. 19.6. 3 бали

Рис. 19.6

| | |
|---|---|
| 1. | 6. |
| 2. | 7. |
| 3. | 8. |
| 4. | 9. |
| 5. | 10. |

**4** Складіть для виконавця Кресляр алгоритм малювання за рис. 19.7. 3 бали

Рис. 19.7

| | |
|---|---|
| 1. | 8. |
| 2. | 9. |
| 3. | 10. |
| 4. | 11. |
| 5. | 12. |
| 6. | 13. |
| 7. | 14. |

**5** Запропонуйте однокласникам виконати складені алгоритми. Чи правильно відтворюються малюнки за вашими алгоритмами? 2 бали

**Комп'ютерне тестування**

Виконайте тестове завдання 19 з автоматичною перевіркою результату.

# § 20. Способи опису алгоритму. Алгоритмічні структури

## Форми подання алгоритмів

Існують *різні форми подання алгоритмів* — словесна, словесно-формульна, графічна, у вигляді програмного коду та інші — залежно від того, на якого виконавця орієнтований алгоритм.

У повсякденному житті найчастіше застосовується **словесна форма**. Алгоритм подається як послідовність окремих пронумерованих пунктів, кожен з яких містить команду на виконання певної дії. Команди виконують одну за одною в порядку зростання їх номерів, якщо немає спеціальної вказівки на перехід до виконання команди за іншим номером в алгоритмі.

Словесна форма подання алгоритму є найприйнятнішою для опису інструкцій побутового характеру, поведінки в разі виникнення певної надзвичайної ситуації, кулінарних рецептів тощо.

**1** Подамо алгоритм приготування «швидкої» піци (рис. 20.1) у словесній формі.

1. Змішайте сметану з томатною пастою, додайте спеції.
2. Отриманий соус намастіть на основу для піци.
3. Зверху викладіть шматочки помідорів і болгарського перцю, половинки маслин.
4. Посипте натертим сиром.
5. Поставте піцу в духову шафу на 5 хвилин для запікання.

Рис. 20.1

Алгоритми обчислень зручно подавати **у вигляді формул**. Так, алгоритм обчислення площі прямокутного трикутника для виконавця Учень 5 класу можна подати у вигляді $S = a \cdot b : 2$, де $a$, $b$ — катети трикутника.

Записуючи алгоритми, часто комбінують словесне та формульне подання запису команд. Для **графічного подання** алгоритмів використовують *блок-схеми*. При цьому команди подаються в окремих блоках, а послідовність виконання команд показано стрілками.

Кожна команда міститься в блоці певного вигляду:

| Назва блоку | Опис дії |
|---|---|
| Початок / Кінець | Позначає початок або кінець алгоритму |
| Дія (процес) | Позначає дію, яку потрібно виконати. Може бути позначена як указівка виконати окрему дію (додати два числа, накреслити лінію), так і послідовність логічно об'єднаних у блок дій (зробити рисунок), тобто певний процес |
| Введення / Виведення | Позначає введення вхідних даних або виведення вихідних даних |
| Умова | Позначає перевірку деякої умови. Умова може набувати одного з двох значень — TRUE (істина) або FALSE (хибність) |

Розглянемо подання алгоритму в графічному вигляді.

**2** На рис. 20.2 зображено блок-схему алгоритму знаходження середнього арифметичного трьох чисел, де:

*a, b, c* — вхідні дані;

*S* — проміжний результат;

*Sr* — вихідні дані (результат виконання алгоритму).

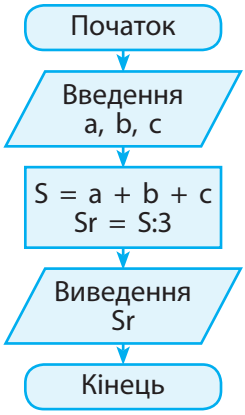



Рис. 20.2

## Базові алгоритмічні структури

Під час конструювання алгоритмів використовують три базові алгоритмічні структури: *слідування, розгалуження, повторення*. Згадаймо означення цих алгоритмічних структур.

**Слідування** — це така форма організації вказівок в алгоритмі, за якої дії виконуються послідовно одна за одною, без пропусків або повторень (рис. 20.3).

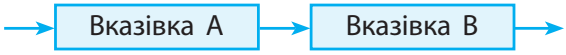



Рис. 20.3

Розглянемо алгоритм **створення комп'ютерної програми.**

1. Складіть алгоритм.
2. Напишіть програму.
3. Відлагодьте програму на комп'ютері.
4. Отримайте результат розв'язування задачі.

**Розгалуження** — це така форма організації дій, коли залежно від виконання або невиконання певної умови виконується одна з двох вказівок (рис. 20.4).

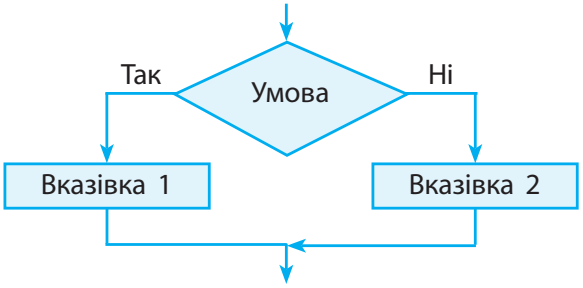



Рис. 20.4

Умову подають у вигляді запитання, сформульованого так, щоб допускалась лише одна з двох відповідей: «так» або «ні». Перевірка умови має бути допустимою дією виконавця.

Якщо умова істинна, то виконується Вказівка 1 (гілка Так); якщо умова хибна, то Вказівка 2 (гілка Ні).

Після виконання однієї з вказівок виконавець переходить до наступної після розгалуження команди.

**3** Складемо блок-схему алгоритму обирання розваги: якщо в касі є квитки, то придбати квиток і переглянути фільм, інакше — піти на прогулянку до парку (рис. 20.5).

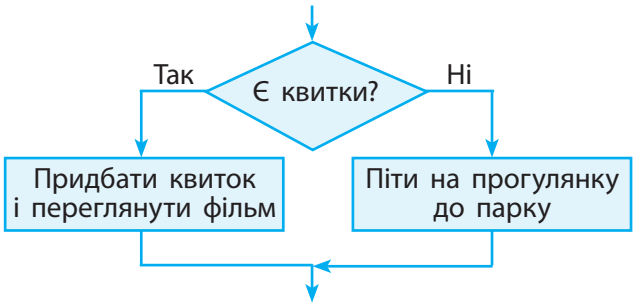



Рис. 20.5

**Повторення (цикл)** — це така форма організації дій, за якої одна й та сама послідовність дій виконується кілька разів залежно від певної умови.

Сукупність вказівок, що повторюється під час кожного проходження (ітерації) циклу, називається **тілом циклу**.

На рис. 20.6 наведено алгоритм із повторенням, у якому певні вказівки повторюються доти, поки задана умова є істинною.

Якщо умова *істинна*, то тіло циклу виконується й відбувається повернення на перевірку умови, якщо *хибна* — виконання циклу припиняється.

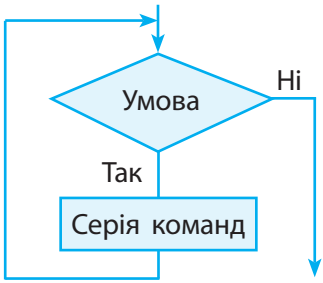



Рис. 20.6

Якщо умова в команді повторення виявиться *хибною* за першої перевірки, то тіло циклу не виконається жодного разу. Якщо під час повторення циклу умова незмінно залишається *істинною*, то цикл може повторюватися нескінченно (кажуть, що програма «зациклена»).

**4** Блок-схему алгоритму збирання яблук подано на рис. 20.7.

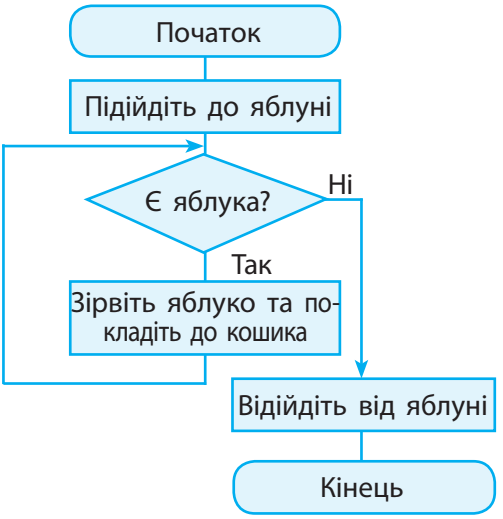



**Рис. 20.7**

Під час складання алгоритму розв'язування задачі необхідно дотримуватися чіткого плану дій.

1. Уважно прочитайте умову задачі.
2. З'ясуйте: що дано — *вхідні дані* (аргументи); що потрібно знайти — *вихідні дані* (результати).
3. Визначте спосіб розв'язування задачі та виявіть необхідні проміжні величини.
4. Складіть алгоритм.
5. Перевірте правильність алгоритму для різних значень аргументів.

## Програма та мова програмування

Програма — це алгоритм розв'язування певного завдання, записаний мовою програмування.

**Мова програмування —** це система позначень, яка використовується для запису алгоритмів для їх виконання за допомогою комп'ютера.

Існує багато мов програмування. Ви вже знайомі з мовою Scratch і основними поняттями, які будуть корисними під час вивчення інших мов.

### Питання для самоперевірки

1. Назвіть основні способи опису алгоритмів.
2. Опишіть основні блоки блок-схеми.
3. Наведіть приклади умов, які можуть бути використані для організації розгалуження.
4. Які переваги подання алгоритму у вигляді блок-схеми?
5. Наведіть приклад алгоритму з розгалуженням, поданого у словесному вигляді.
6. Наведіть приклад алгоритму з повторенням, поданого у словесному вигляді.

## Вправа 20. Способи опису алгоритму. Алгоритмічні структури

Завдання: виконати алгоритм, поданий у вигляді блок-схеми.

**1** На рис. 20.8 подано блок-схему алгоритму обчислення площі прямокутного трикутника за формулою $S=\frac{ab}{2}$. Впишіть потрібні написи.

2 бали

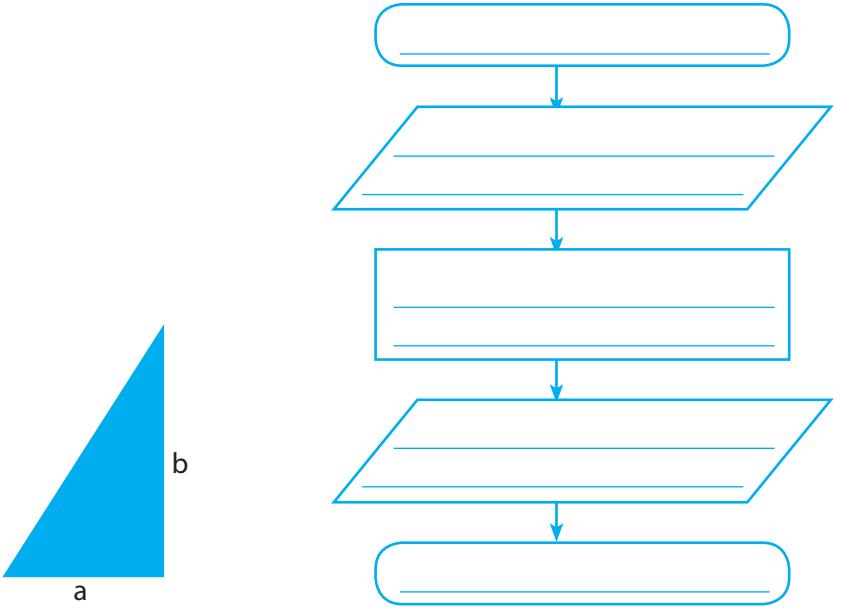
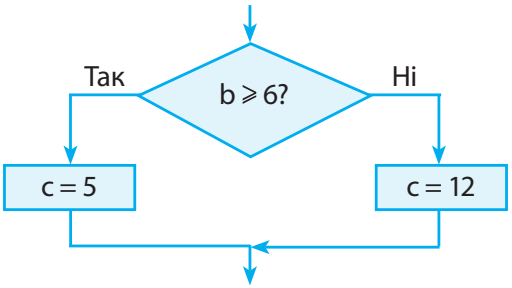



Рис. 20.8

**2** Визначте істинність умови $a > 5$ за різних значень $a$. Заповніть таблицю. 2 бали

| $a$ | 8 | 5 | 2 |
|---|---|---|---|
| $a>5$ | | | |

**3** Знайдіть значення величини $c$ після виконання команди розгалуження для різних початкових значень $b$ (рис. 20.9). Заповніть таблицю. 2 бали




| $b$ | 4 | 6 | 10 |
|---|---|---|---|
| $c$ | | | |

Рис. 20.9

**4** Впишіть потрібні написи в блок-схему команди розгалуження (рис. 20.10): якщо ціна $c$ книжки не перевищує $n$ гривень, то купити цю книжку і сувенір, в іншому випадку — купити тільки книжку. 2 бали

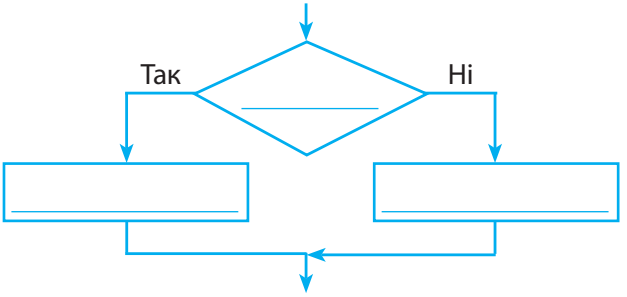
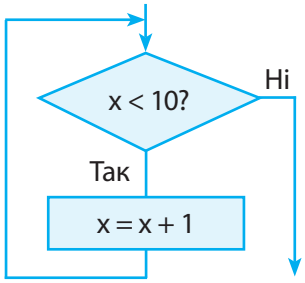



Рис. 20.10

**5** Знайдіть значення величини $x$ після виконання команди повторення для різних початкових значень $x$ (рис. 20.11). Заповніть таблицю. 2 бали




| Початкове значення $x$ | 4 | 6 | 10 |
|---|---|---|---|
| Значення $x$ після виконання циклу | | | |

Рис. 20.11

**6*** Ґра Баше.

2 бали

Є 11 предметів. За один хід гравець може взяти 1–3 предмети. Програє той, якому дістанеться останній. Існує алгоритм виграшу для гравця, який робить хід: 1-й хід: узяти два предмети. 2-й хід і далі: брати стільки предметів, щоб кількість предметів, узятих разом із суперником за черговий хід, у сумі становила 4. Впишіть потрібні написи в блок-схему виграшного алгоритму для першого гравця (рис. 20.12).

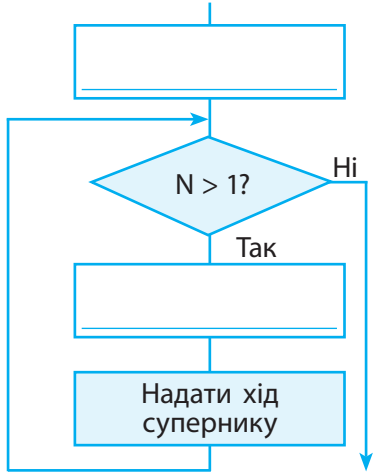



Рис. 20.12

**Комп'ютерне тестування**

Виконайте тестове завдання 20 з автоматичною перевіркою результату.

# § 21. Середовище опису та виконання алгоритмів

Починаємо знайомство з популярною мовою програмування Python, яка застосовується для розв'язування різних завдань: написання програм, створення ігор, розробки сайтів.

Мова програмування Python була створена в 1991 році нідерландським програмістом *Гвідо ван Россумом* (рис. 21.1) і названа ним на честь скетч-серіалу «Літаючий цирк Монті Пайтона» (англ. *Monty Python's Flying Circus*).

Рис. 21.1

Мова Python підтримується всіма операційними системами. Існують версії для Linux, Windows. Наразі використовують дві версії Python: 2.x і 3.x. У нашому підручнику розглядатимемо версію Python 3.

## Установлення програмного середовища Python

Перш ніж почати програмувати на Python, середовище програмування потрібно встановити на комп'ютер. Завантажити файл для інсталяції можна із сайта python.org (рис. 21.2).


Рис. 21.2

Разом із Python 3 на комп'ютер буде встановлено програму IDLE — орієнтоване на початківців середовище програмування, у якому є засоби для написання та налагодження програм мовою Python.

**Алгоритм установлення програмного середовища Python**

1. Зайдіть на сайт http://www.python.org
2. Клацніть Downloads, щоб відкрити сторінку завантаження (рис. 21.3).

3. Клацніть кнопку з версією Python 3.6.1.
4. Після завантаження інсталяційного файла двічі клацніть на ньому, щоб установити Python на своєму комп'ютері.
5. Запустіть IDLE. Для цього виберіть команди Пуск → → Всі програми → Python → IDLE.
   Відкриється вікно IDLE (рис. 21.4).

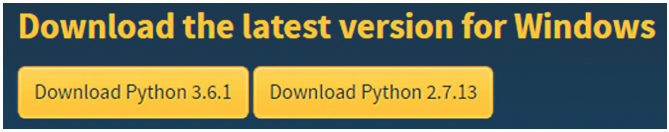



Рис. 21.3

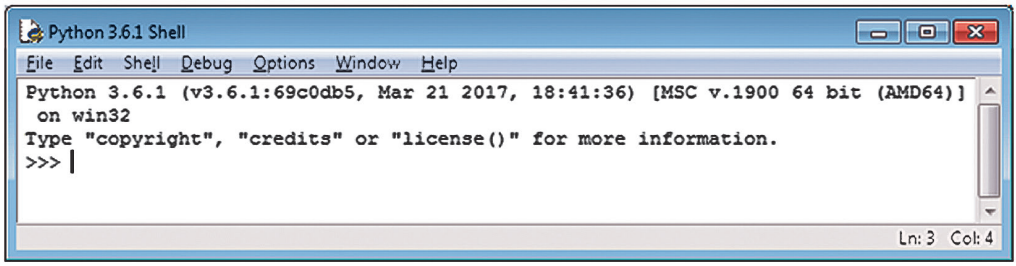



Рис. 21.4

Ми можемо починати програмувати, записуючи команди після позначки >>>.

Рядок >>> називається **запрошенням**, і його наявність означає, що комп'ютер готовий прийняти вашу першу команду.

У Python існують два види вікон: вікно програми і вікно консолі (IDLE).

> ! У **вікні програми** можна вводити й зберігати програмний код для подальшого виконання, а у **вікні консолі** — уводити команди й одразу отримувати результат виконання.

## Знайомство з IDLE

Вікно IDLE є вікном консолі. У ньому відображаються результати виконання програми й повідомлення про помилки в програмному коді.

На рис. 21.5 проілюстровано виконання команд у вікні IDLE.

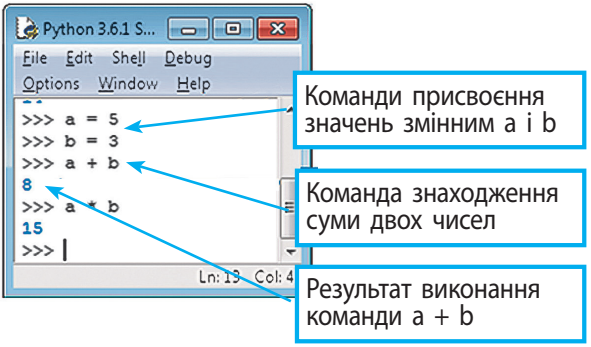



Рис. 21.5

Команди, які введено у вікні консолі, виконуються після натискання клавіші Enter, і результат одразу виводиться у вікні IDLE.

1 Щоб зрозуміти, як працює IDLE, після позначки >>> запишемо команду print ('Hello, World!') і натиснемо клавішу Enter. У вікні IDLE буде виведено привітання (рис. 21.6).

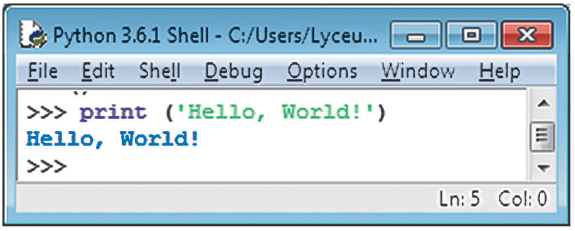



Рис. 21.6

Вікно IDLE зручно використовувати для того, щоб зрозуміти, що виконує та чи інша команда. Але для створення великих програм, які необхідно зберігати й редагувати, використовують вікно програми.

## Вікно програми

Вікно програми призначене для введення та редагування тексту комп'ютерної програми.

Щоб **відкрити вікно програми,** в IDLE виберіть команду File → New File. Відкриється окреме вікно програми, яка до першого збереження має ім'я Untitled (рис. 21.7).

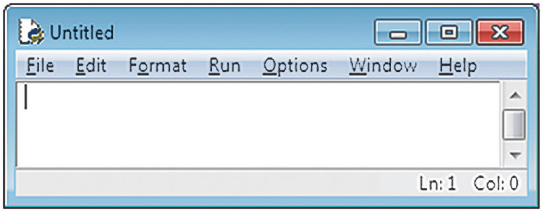



Рис. 21.7

На рис. 21.8 наведено алгоритм **роботи з програмою** в середовищі Python у вікні програми.

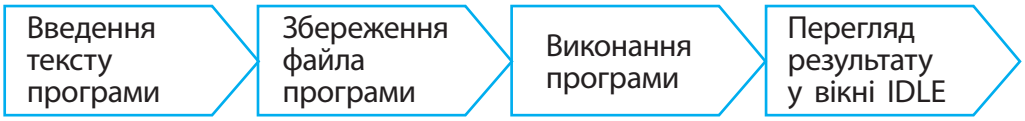



Рис. 21.8

2

Створимо програму розв'язання задачі за поданим вище алгоритмом роботи з програмою.

*Задача*: знайти суму і добуток двох чисел.

1. Уведемо код у вікні програми (рис. 21.9).
2. Виберемо команду File → Save As. Уведемо ім'я файла Перша програма і натиснемо Save. Файл збережено з розширенням .py.

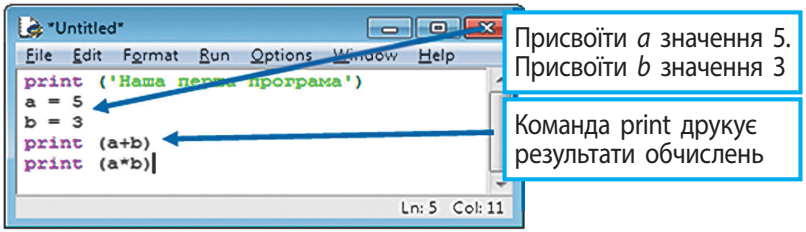



Рис. 21.9

3. Виберемо команду Run → Run Module (або натиснемо F5).

4. Переглянемо результат роботи програми у вікні консолі IDLE (рис. 21.10).

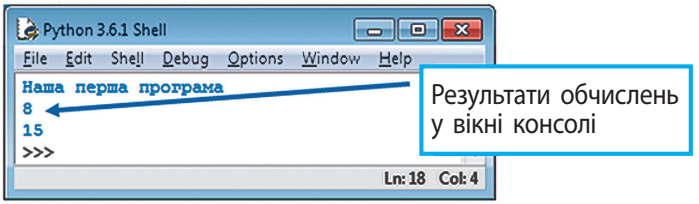



Рис. 21.10

Програму не можна запустити на виконання, поки вона не збережена. У разі спроби виконати програму без збереження з'явиться попередження.

! Зверніть увагу, що попередження виділено червоним кольором.

У Python різні складові програмного коду виділяються певними кольорами:

| | |
|---|---|
| | Команди Python, наприклад, print, — фіолетового кольору |
| | Рядки в лапках виділені зеленим кольором. Якщо дужки навколо текстового рядка теж виділені зеленим, десь не вистачає лапок |
| | Службові слова мови Python (if, while ті інші) — помаранчевого кольору |
| | Помилки у вікні програми та повідомлення про помилки у вікні консолі виділяються червоним |

Кольорові підказки допомагають уникати помилок під час уведення тексту програми.

### Питання для самоперевірки

1. Які види вікон є у програмному середовищі Python?
2. Які повідомлення відображаються у вікні консолі?
3. Як відкрити вікно програми?
4. Для чого призначене вікно програми?
5. Опишіть алгоритм роботи з програмою в середовищі Python.
6. Поясніть, чому складові програмного коду виділяються різними кольорами.

# Вправа 21. Середовище опису та виконання алгоритмів

Завдання: виконати програму у вікні програми.

*Під час роботи за комп'ютером дотримуйтесь правил безпеки.*

**1** Завантажте програму Python → IDLE. Виберіть команду File → New File (Файл → Новий файл), щоб відкрити вікно програми.
Наберіть текст програми додавання двох цілих чисел:

```
a = 5
b = 10
c = a + b
print ('c=',c)
```

2 бали

**2** Для збереження файлу виберіть команду File → Save As (Файл → Зберегти як).
У вікні збереження файлу виберіть власну папку, наберіть назву файлу Vprava19 та натисніть Save (Зберегти). Файл збережено з розширенням .py.
Запишіть повну адресу файла.

2 бали

**3** Для запуску програми на виконання виберіть команду Run → Run Module (Виконати → Виконати модуль) або натисніть клавішу F5.
Програма почне виконуватися.
Якщо з'явиться повідомлення про помилку, перевірте, чи правильно набрано текст програми.
Перегляньте результат виконання програми у вікні консолі. Закрийте вікно IDLE.

2 бали

**4** Повторно виконайте програму. Для цього ввійдіть у власну папку, установіть курсор на позначку програми й відкрийте контекстне меню, у якому виберіть команду Edit with IDLE (Змінити в IDLE).

2 бали

**5** У вікні програми змініть значення змінних *a* і *b*:
a = 10
b = 17

2 бали

**6** Збережіть файл і запустіть програму на виконання.
Запишіть результат виконання.
Закрийте вікно IDLE.

2 бали

**Комп'ютерне тестування**

Виконайте тестове завдання 21 з автоматичною перевіркою результату.

# § 22. Основні поняття мови програмування Python

Для написання програми певною мовою програмування необхідно знати алфавіт мови, правила запису команд та правила їх використання в програмі.

## Алфавіт мови Python

Як і будь-яка мова, Python має алфавіт, синтаксис, семантику.

**Алфавіт мови** — набір символів, що може використовуватись у програмному коді.

Під час створення програм мовою Python можна використовувати такі символи:

- літери латинського алфавіту: A...Z, a...z;
- цифри: 0...9;
- знаки арифметичних операцій, спеціальні символи:
  + — * / \ ^ = < > ( ) . , : ; ‘ # _.

У мові Python використовують також комбінації символів і службові слова, що мають фіксований зміст:

- комбінації символів: <=, >=, <>, = =; !=, **;
- службові слова: and, elif, if, as, else, import тощо.

**Синтаксис мови** — сукупність правил побудови команд мовою програмування.

Якщо ви неправильно напишете ключове слово, що позначає команду, Python не зможе зрозуміти введену вами команду, і виведе у відповідь повідомлення про синтаксичну помилку SyntaxError. Місце помилки у вікні консолі помічається червоним кольором.

Після команди, що містить помилку, виводиться повідомлення про помилку. Уважно читайте такі повідомлення — це допоможе зрозуміти, у чому помилка, і виправити її.

**1** Розглянемо програмний код, наведений на рис. 22.1.

У виразі допущено помилку: надруковано зайву дужку — у консолі з'являється повідомлення про синтаксичну помилку.

```
>>> a = 2*(x+5))
SyntaxError: invalid syntax
>>>
```

Рис. 22.1

**Семантика мови** — сукупність правил виконання комп'ютером команд, записаних мовою програмування. Із семантикою мови Python ми будемо знайомитись у міру вивчення команд.

## Величини в мові Python

Окремий інформаційний об'єкт (число, символ, рядок тощо) називають величиною. Кожна величина характеризується *розміром виділеної пам'яті для зберігання*, *назвою* (*ідентифікатором*), *типом* і *значенням*.

**Вид величини** визначає спосіб використання величини в програмі. Величина може бути *константою* (тобто *постійною*) або *змінною*.

**Константи** — це величини, значення яких не можуть змінюватися в ході виконання програми. Прикладом константи може бути число (5, 1.23) або рядок: «Це рядок!».

**Змінні** — це величини, значення яких можуть змінюватися в ході виконання програми. Змінні потрібні для зберігання даних. Змінним дають імена (ідентифікатори).

З'ясуємо, яких правил необхідно дотримуватися під час надання імен змінним.

- Першим символом імені має бути літера чи знак нижнього підкреслювання _.
- Решта імені може складатися з літер, чисел або знаків нижнього підкреслювання.
- Не можна використовувати спеціальні символи, такі як /, # або @. Не можна використовувати пробіли.
- Імена змінних чутливі до регістру. Наприклад, myname і myName — це різні змінні.
- Не можна надавати змінним імена команд, наприклад print.

**2** Правильні імена: i, __my_name, name_23, a1, b2.
Неправильні імена: 2things (рис. 22.2), this is spaced out, my-name.

```
>>> 2things = 15
SyntaxError: invalid syntax
>>>
```

Рис. 22.2

## Типи величин у мові Python

**Тип величини** визначається обсягом пам'яті, необхідним для її збереження, множиною припустимих значень величини та операціями, які можна над нею виконувати.

Основними типами величин є *числа* й *рядки*.

У Python є два типи числових даних: цілі числа (int) і дійсні (float) — дробові числа.

**Цілими числами** називають натуральні числа (1, 2, 3...), їм протилежні числа (–1, –2, –3...) і число 0. Кількість учнів у класі, кількість предметів ми зазвичай указуємо за допомогою цілих чисел. **Дійсні числа** — це десяткові дроби. Вони потрібні, коли ми хочемо вказати частину чого-небудь, наприклад, 3.5 м, 1.25 грн. Як роздільник між цілою та дробовою частиною дійсного числа використовується крапка.

**Рядок** — це взята в одинарні або подвійні лапки послідовність будь-яких символів: цифр, літер, розділових знаків та ін.

Рядки належать до типу str. У змінних рядкового типу зберігають фрагменти тексту.

Щоб дізнатися, до якого типу належить змінна або константа, можна скористуватися командою type (рис. 22.3).

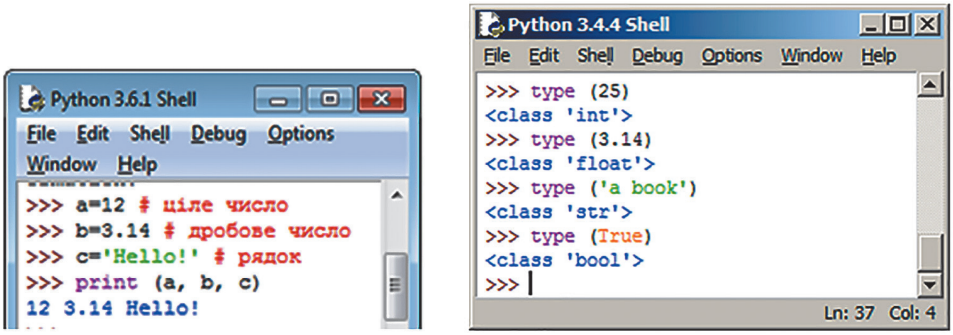



Рис. 22.3

## Указівка присвоєння

**Значення** — характеристика величини, яка може багаторазово змінюватися в процесі опрацювання інформації.

Щоб створити змінну в Python, необхідно дати їй назву й присвоїти значення (рис. 22.4).


Рис. 22.4

Загальний вигляд команди присвоєння: A = B, де A — ім'я змінної, B — константа, змінна або вираз.

Змінну часто порівнюють зі скринькою, в якій зберігається значення величини.

3 Створимо змінну *a*: a = 7

Вираз присвоювання наказує комп'ютеру запам'ятати число 7 у змінній *a*.

Схема виконання вказівки присвоєння: спочатку обчислюється значення виразу в правій частині вказівки присвоєння, потім це значення надається змінній, ім'я якої записане в лівій частині (рис. 22.5).

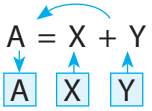



Рис. 22.5

Під час виконання команди присвоєння ділянка пам'яті, відведена для змінної, заповнюється новим значенням, одночасно знищуючи старе. Тому в правій частині виразу може бути вказане тільки ім'я змінної.

**4** Наступні команди присвоюють змінній rabbits значення 5, потім те саме значення присвоюють змінній hats:

```
>>> rabbits = 5
>>> hats = rabbits
```

У виразах можна використовувати змінні. Якщо в правій частині оператора присвоєння записати вираз, то змінна в лівій частині набуває значення виразу.

**5** Нехай $a = 10$. Тоді після виконання вказівки присвоєння

```
a = a + 5
```

змінна $a$ отримає значення 15.

При спробі виконати оператор a + 2 = b буде виведено повідомлення про синтаксичну помилку (рис. 22.6).

```
>>> a=5
>>> a+2=b
SyntaxError: can`t assign to operator
\\\ |
```

Рис. 22.6

## Питання для самоперевірки

1. Назвіть основні характеристики величини.
2. Які послідовності символів неприпустимо використовувати як імена (ідентифікатори) і чому?
   а) suma; г) (sum);
   б) w1; ґ) a–4;
   в) primer 1; д) if.
3. До яких типів належать величини:
   а) кількість учнів і учениць у класі;
   б) вага кошика з яблуками в кг;
   в) прізвище учня (учениці).
4. Чому дорівнює значення $x$ після виконання послідовності присвоювань:

а) $y=2;\ x=y;$ в) $x=5;\ x=-x;$
б) $x=8;\ x=x+2;$ г) $x=0;\ x=x+3?$

5. Після виконання оператора $x=y+x$ нові значення змінних такі: $x=10,\ y=3$. Чому дорівнювали $x$ і $y$ до виконання оператора присвоєння?
6. Після виконання оператора $x=y+x$ нові значення змінних такі: $x=3,\ y=10$. Чому дорівнювали $x$ і $y$ до виконання оператора присвоєння?

## Вправа 22. Основні поняття мови програмування Python

Завдання: виконати дії над величинами в середовищі Python.

*Під час роботи за комп'ютером дотримуйтесь правил безпеки.*

**1** Завантажте програму Python → IDLE. Після запрошення >>> запишіть команду створення змінної a:
>>>a = 10
і натисніть клавішу Enter. Інтерпретатор створить змінну a і надасть їй значення 10.
Щоб перевірити значення змінної a, після запрошення >>> запишіть назву змінної і натисніть Enter.

2 бали

**2** Щоб дізнатися тип змінної a, виконайте команду
>>>type(a)

2 бали

**3** Виконайте команду Збільшити значення змінної a на 3:
>>>a = a+3
Виведіть значення змінної a:
>>>a
Запишіть отримане значення: .

2 бали

**4** Виконайте команду Збільшити значення змінної a вдвічі і виведіть значення змінної a.

Запишіть команду:

Запишіть значення змінної: a =

2 бали

**5** Створіть змінну с рядкового типу: 2 бали
>>>c = «Вітаю!»
Дізнайтеся тип змінної с.

Запишіть команду:

Запишіть тип змінної:

**6** Виконайте команду Продублювати значення змінної с: 2 бали
>>>c = c*2
Виведіть значення змінної с.

Запишіть значення змінної: с =

Поясніть результат виконання команд.

**Комп'ютерне тестування**

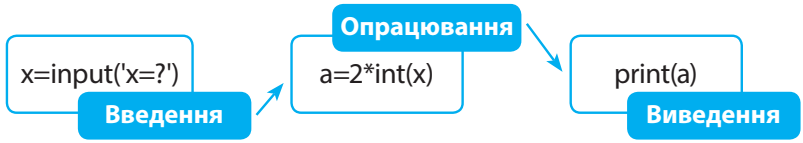
Виконайте тестове завдання 22 з автоматичною перевіркою результату.

# § 23. Лінійні алгоритми та програми

Алгоритм, у якому використовується тільки структура слідування, називається **лінійним**.

Будь-яка програма має прийняти вхідні дані (введення), опрацювати їх і повернути результат (виведення). У програмах, що реалізують лінійні алгоритми, використовуються команди введення даних, присвоєння та виведення.

На рисунку наведено логічну структуру програми, в якій реалізовано лінійний алгоритм.




## Введення даних

Команда input() призначена для введення даних із клавіатури.

Коли програма зустрічає команду input, вона припиняє роботу й очікує, поки користувач уведе дані й натисне Enter. У дужках може бути записана підказка користувачеві, що саме потрібно ввести. Ця підказка виводиться на екран.

**1** Випробуємо у вікні консолі, як працює input():

```
>>> name = input('Як тебе звати?')
Як тебе звати? Петро
>>> name
'Петро'
```

Значення, отримане від команди input, Python сприймає як рядок (послідовність літер), навіть якщо ми ввели число.

**2** У разі спроби додати до значення змінної *a* число 3 виникне помилка, оскільки Python не знає, як додати число до рядка.

```
>>> a = input('a=?')
a=?5
>>> a+3
Traceback (most recent call last):
File "<pyshell#20>", line 1, in <module>
a+3
TypeError: must be str, not int
```

Необхідно виконати перетворення введеного значення на число за допомогою функції int().

Функція int(s) перетворює рядок *s* на ціле число.

**3** Тепер помилки немає:

```
>>> a = input('a=?')
a = ?3
>>> int(a)+5
8
```

При введенні числових значень зручно застосовувати функції перетворення типів до значення, яке повертає функція input():

```
a = int(input('a=?'))
```

Під час запису дійсного числа у формі десяткового дробу для розділення цілої та дробової частин як десятковий роздільник використовується крапка «.».

Функція float(s) перетворює рядок *s* на дробове число.

**4** Введення з клавіатури дійсного значення:

```
>>> x = input('x=?')
x = ?5.25
>>> float(x)+3
8.25
```

## Виведення значень змінних

Команда print() виводить текст у вікно консолі:

print(<список виведення>)

**5** За допомогою команди print можна дізнатися значення змінної.

```
>>> print (rabbits)
5
```

У списку виведення можуть бути константи, змінні, вирази.

**6** Вивести значення змінної *x*:

```
print('x = ', x)
```

Якщо потрібно вивести значення кількох змінних або виразів, їх необхідно перелічити через кому:

```
>>> x = 4
>>> print (x, 2*x, 3*x)
4 8 12
```

За допомогою команд input і print можна організувати діалог користувача з програмою:

```
>>> name = input('Як тебе звати?')
Як тебе звати? Петро
>>> print ('Привіт, ',name)
Привіт, Петро
```

## Коментарі в програмі

**Коментар** — це текст, призначений для читання людиною, а не комп'ютером. Коментар — це підказка, яку програмісти записують у своїй програмі.

Щоб комп'ютер відрізнив команди від коментарів, у мові Python перед текстом коментаря ставиться знак #. Редактор IDLE виділяє коментарі червоним кольором, нагадуючи про те, що ці фрагменти коду будуть проігноровані.

**7** Коментар пояснює призначення наступної команди:

```
# Запит імені користувача
s = input('Як тебе звати?')
```

### Питання для самоперевірки

1. Чому виникає помилка при спробі виконання коду?
   ```
   a = input('Уведіть значення a:')
   b = a+10
   ```
2. Для чого призначена команда print()?
3. Назвіть константи та змінні у списку виведення:
   ```
   print('a = ', a, 5, '3*b', 3*b)
   ```
4. Що буде виведено у вікно консолі в результаті виконання коду?

   а)
   ```
   a = 5
   print("a = ", a)
   ```

   б)
   ```
   a = 5
   b = 7
   print("a+b = ", a+b)
   ```

   в)
   ```
   a = 5
   print("a**2 = ", a**2)
   ```

   г)
   ```
   a = 5
   b = 7
   print("Площа дорівнює", a*b, "кв. м")
   ```

5. Поясніть помилки в запису операторів
   а) a = int(input('a = ?')
   б) print("s = ", "v*t", "км")
6. Відновіть можливий варіант тексту програми, якщо після роботи програми вікно консолі має вигляд:

а)
```
a =? 3
b =? 10
a+b = 13
```

б)
```
Швидкість? 15
Час? 2
Пройдений шлях 30 км
```

## Вправа 23. Лінійні алгоритми та програми

Завдання: скласти програму-гру Нісенітниця.

*Алгоритм гри*: програма просить ввести відповіді на питання «Хто?», «Де?», «Коли?» і зберігає введені дані в трьох різних змінних. Зі створених змінних з додаванням жартівливої кінцівки формується речення і виводиться на екран.

*Під час роботи за комп'ютером дотримуйтесь правил безпеки.*

**1** Відкрийте вікно IDLE і створіть нове вікно програми. 2 бали

**2** Наберіть текст програми: 2 бали

```
c1 = input ('Хто? ')
c2 = input ('Де? ')
c3 = input ('Коли? ' )
phrase = c1+' '+c2+' '+c3+' співає пісню'
print(phrase)
```

**3** Збережіть файл у власній папці з ім'ям Вправа 23. Запустіть програму на виконання, дайте відповіді на питання програми, проаналізуйте результат виконання програми. 2 бали

**4** Додайте до програмного коду оператор для введення відповіді на питання «Що робить?».
Запишіть оператор присвоєння змінній c4 значення текстового рядка, уведеного з клавіатури:

2 бали

**5** Додайте значення змінної c4 до рядка phrase.

2 бали

phrase =

**6** Збережіть програму і запустіть її на виконання. У вікні консолі дайте відповіді на питання програми. Запишіть результат виконання програми:

2 бали

Закінчіть роботу, закривши вікно IDLE.

**Комп'ютерне тестування**

Виконайте тестове завдання 23 з автоматичною перевіркою результату.

# § 24. Математичні оператори мови Python

Майже в усіх програмах вам доведеться використовувати математичні операції, наприклад, для підрахунку балів в ігровій програмі, розв'язування рівнянь тощо.

Обчислення здійснюються за допомогою арифметичних виразів. Арифметичні вирази будуються з імен змінних, констант, знаків арифметичних операцій і круглих дужок так, як у математиці.

З арифметичними діями додавання і множення, доступними в Python, ми познайомилися в попередніх параграфах.

Символи +, –, * тощо, що використовуються для позначення арифметичних дій (операцій), прийнято називати **операторами**. Об'єкти (змінні або константи), над якими виконуються операції, називають **операндами** (рис. 24.1).


Рис. 24.1

## Математичні оператори Python

Розглянемо математичні операції та відповідні їм символи операторів, наведені в таблиці:

| Операція | Символ операції | Приклад | Результат |
|---|---|---|---|
| Додавання | + | Res = 15+3 | Res = 18 |
| Віднімання | – | A = Res–10 | A = 8 |
| Множення | * | A = A*2 | A = 16 |
| Ділення | / | Res = 5/2 | Res = 2.5 |
| Обчислення неповної частки від ділення | // | Res = 5//2 | Res = 2 |
| Обчислення остачі | % | Res = 5%2 | Res = 1 |
| Піднесення до степеня | ** | A = 4**2 | A = 16 |

При виконанні декількох операцій в одному виразі необхідно **враховувати їх пріоритет** (порядок виконання):

1) піднесення до степеня (**);
2) множення (*) і ділення (/), цілочисельне ділення (//), одержання остачі від цілочисельного ділення (%);
3) додавання (+) і віднімання (–).

Якщо операції мають однаковий пріоритет, то вони виконуються зліва направо по черзі. Для того щоб змінити порядок виконання, можна користуватися дужками.

**1** Обчислити значення змінних a і b:

```
a = 10+ 2*3**2  # a=28
b = (10+ 2)*3**2  # b=108
```

Під час запису в тексті програми арифметичних виразів потрібно дотримуватися правил лінійного запису (рис. 24.2):

Рис. 24.2

- Вираз має бути записаний у вигляді лінійного ланцюжка символів.
- Не можна опускати знак операції множення.
- Порядок виконання операцій одного пріоритету регулюється дужками.

**2** Змінній a присвоїти значення виразу $\frac{2x-5}{3+x}$:

```
a = (2*x–5)/(3+x)
```

**3** Розв'язати задачу: відомо, що деяка подія відбулась $x$ секунд тому. Виразити цей часовий період у годинах, хвилинах і секундах.

```
x = int(input('Кількість секунд?'))
h = x//3600
print(h, 'год.')
m = (x%3600)//60
print(m, 'хв.')
s = (x%3600)%60
print(s, 'сек.')
```

```
Кількість секунд?10000
2 год.
46 хв.
40 сек.
```

## Випадкові числа

При створенні ігрових або тестових програм іноді потрібно вибрати число з певного проміжку випадковим чином. У програмуванні випадкові числа отримують за допомогою спеціальних пристроїв — генераторів випадкових чисел.

Щоб **отримати випадкове число,** необхідно за допомогою команди import завантажити модуль random:

```
from random import*
```

Функція randint(x1, x2) вибирає ціле випадкове число в діапазоні від $x_1$ до $x_2$.

**4** Отримаємо випадкове число в діапазоні від 1 до 10.

```
>>> from random import*
>>> randint (1, 10)
8
>>> randint (1, 10)
6
```

**5** Для навчальної програми з вивчення таблиці множення згенерувати приклад з випадковими значеннями множників.

```
from random import*
x = randint(2, 9)
y = randint(2, 9)
print(x, '*', y, ' = ?')
vidp = int(input('Добуток:'))
```

## Питання для самоперевірки

1. Обчисліть значення виразів:
   а) 7/2; г) 123//100;
   б) 7//2; ґ) 123%10;
   в) 7%2; д) (123//10)%10.
2. Обчисліть значення, якого набуває змінна a в результаті виконання оператора присвоювання, якщо b = 4:
   а) a = 37–b**2; в) a = (b+12)/2;
   б) a = 100–2*b**2; г) a = b+12/2.
3. Запишіть вирази за правилами лінійного запису:
   а) $\frac{x-5}{2x}$; г) $\frac{x}{2+x}$;
   б) $x^5$; ґ) $\frac{35x}{x+y}-\frac{8x+5}{4}$.
   в) $2x^2-3x+5$;
4. Як отримати випадкове число в діапазоні від 1 до 100? Як отримати випадкові числа? Наведіть приклади задач, для яких потрібні випадкові числа.

5. Запишіть оператори присвоювання, які реалізують такі дії:
   а) змінній $s$ присвоїти значення суми змінних $a$ і $b$;
   б) обчислити значення добутку змінних $a$ і $b$ і результат присвоїти змінній $c$;
   в) подвоїти значення змінної $a$;
   г) змінну $a$ збільшити на 10;
   ґ) змінній $n$ присвоїти значення неповної частки від ділення 100 на 7;
   д) змінній $k$ присвоїти значення остачі від ділення значення змінної $a$ на 7.
6. Напишіть програму для обчислення площі та периметра прямокутної кімнати, ширина якої $a$ м, довжина $b$ м. Виконайте програму для $a = 3{,}9$ м, $b = 7{,}2$ м.

## Вправа 24. Математичні оператори мови Python

Завдання: розв'язати задачу в середовищі Python.

*Задача*: обчислити суму цифр двоцифрового числа $a$.

*Під час роботи за комп'ютером дотримуйтесь правил безпеки.*

**1** Завантажте програму IDLE → Python.
В IDLE виберіть команду File → New File, щоб відкрити вікно програми.
У вікні програми наберіть текст програми для обчислення суми цифр двоцифрового числа a.

2 бали

```
a = 25
a1 = a//10
a2 = a%10
sum = a1+a2
print('sum = ', sum)
```

**2** Для збереження файлу виберіть команду File → Save As. У вікні збереження файлу виберіть власну папку, наберіть ім'я файлу Вправа27 і натисніть Save.

2 бали

**3** Запустіть програму на виконання командою Run → → Run Module. Перегляньте результат виконання програми. 2 бали

**4** Змініть значення a. Збережіть файл і запустіть програму на виконання. 2 бали
Запишіть результат роботи програми для заданого значення a:

a = sum = .

**5** Змініть програму таким чином, щоб обчислювалася сума цифр трицифрового числа. Заповніть пропуски в тексті програми: 2 бали

a =

a1 =

a2 = a//10%10

a3 =

sum =

print ('sum =', sum)

**6** Збережіть файл і запустіть програму на виконання. Запишіть результат виконання програми: 2 бали

Закінчіть роботу, закривши вікно IDLE.

**Комп'ютерне тестування**

Виконайте тестове завдання 24 з автоматичною перевіркою результату.

# Практична робота 7
## Складання та виконання лінійних алгоритмів

Завдання: створити програму «Калькулятор піци».

Обладнання: комп'ютер зі встановленим середовищем програмування Python.

Програма має виконати такі дії:

- запитати, скільки піц бажає замовити покупець;
- запитати ціну піци, зазначену в меню;
- обчислити підсумкову вартість покупки, ураховуючи знижку на 10 % на честь ювілею піцерії;
- вивести ціну покупки.

### Хід роботи

*Під час роботи з комп'ютером дотримуйтеся правил безпеки.*

**1.** Відкрийте вікно IDLE і створіть нове вікно програми.

**2.** Заповніть пропуски в запису оператора для введення кількості піц:
number = int( )
Запишіть оператор у вікні програми.

**3.** Заповніть пропуски в запису оператора для введення ціни однієї піци:
cost = int( )
Запишіть оператор у вікні програми.

**4.** Заповніть пропуски в запису операторів для обчислення ціни покупки з урахуванням знижки:

total =

discount = total*0.1

total =

Запишіть оператор у вікні програми.

**5.** Додайте до програмного коду оператори виведення значень змінних discount і total.

**6.** Збережіть файл у власній папці з ім'ям Pizza.py. Запустіть програму на виконання, перевірте її для різних початкових даних. Запишіть початкові дані та отримані результати:

Кількість піц

Ціна 1 піци

Ціна без знижки

Ціна зі знижкою

Закінчіть роботу, закривши вікно IDLE.

Зробіть висновок: як складати та виконувати лінійні алгоритми для розв'язування задачі.

# § 25. «Черепашача» графіка

У середовищі Python Черепашкою зветься уявний робот — пристрій, який переміщається по екрану й повертається в заданих напрямках, водночас залишаючи (або, за вибором, не залишаючи) за собою намальований слід заданого кольору й ширини.

Положення та напрямок руху Черепашки відображає невелика чорна стрілочка, яка повільно пересувається по екрану. Це дає змогу відстежити рух Черепашки та зрозуміти, яким чином кожен рядок коду впливає на траєкторію її руху.

Рис. 25.1

Черепашка (рис. 25.1) допоможе нам вивчити основи комп'ютерної графіки, і ми будемо малювати за її допомогою цікаві рисунки.

## Система координат

Результат виконання Черепашкою команд відображається у графічному вікні Python Turtle Grafics (рис. 25.2).

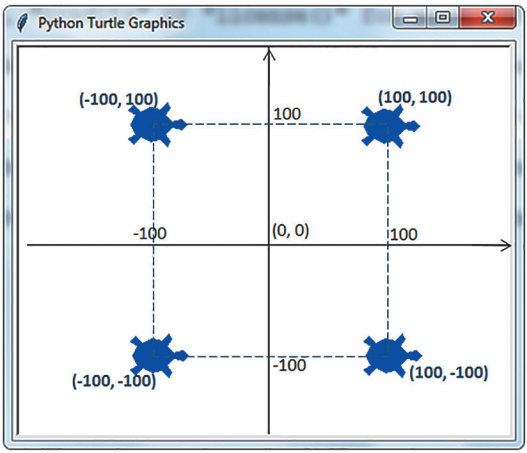



Рис. 25.2

Для визначення місцезнаходження Черепашки використовують координати. Ви вже знаєте, що таке координатна пряма, і вмієте визначати положення точки на прямій.

Але для малювання нам доведеться користуватися орієнтирами не тільки вздовж прямої, а й на площині. Будь-яка точка у вікні Python Turtle Grafics може бути задана парою чисел (X, Y).

**Координатні осі** — це дві координатні прямі, які перетинаються під прямим кутом. Центр вікна Python Turtle Grafics — точка перетину невидимих координатних осей — точка з координатами (0, 0). Вертикальна координата Y зростає знизу догори, а горизонтальна X — зліва направо.

На уроках математики ви працювали із числами, розташованими на координатній прямій праворуч від 0. Але горизонтальну числову пряму можна продовжити вліво, а вертикальну — униз від 0, а на променях ліворуч і знизу від 0 розташовані числа зі знаком мінус (–).

Повернемося до рис. 25.2. На ньому зображені черепашки і вказано координати їх розташування.

## Команди Черепашки

Для завантаження команд для роботи з Черепашкою потрібна команда:

from turtle import*

Після введення цієї команди ви можете подавати Черепашці команди малювання.

Черепашка на початку малювання знаходиться у точці з координатами (0, 0). У процесі малювання ви можете дізнатися поточні координати Черепашки за допомогою функції position().

**1** Вивести в консоль координати місцезнаходження Черепашки після виконання команд малювання. Результат виконання програми наведено на рис. 25.3:

```
from turtle import *
forward(50)
left(90)
forward(50)
right(90)
forward(50)
print(position())
```

```
=== RESTART:
(100.00, 50.00)
>>>
```

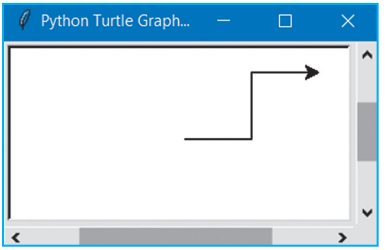



Рис. 25.3

За допомогою команди speed(n) ви можете змінювати швидкість руху Черепашки. Значення n може бути від 1 (повільно) до 10 (швидко); якщо n = 0, малюнок буде створено миттєво.

За замовчуванням «перо» Черепашки опущене на поверхню вікна, і Черепашка залишає слід при пересуванні. Для керування пером призначені команди:

- down() — опустити перо;
- up() — підняти перо.

Розглянемо команди малювання та їх призначення.

| Команда | Призначення |
|---|---|
| width(n) | Установити ширину сліду Черепашки в n пікселів |
| forward(n) | Проповзти вперед n кроків (пікселів) |
| left(angle) | Повернути ліворуч на angle градусів |

*Закінчення таблиці*

| | |
|---|---|
| right(angle) | Повернути праворуч на angle градусів |
| circle(r) | Намалювати коло радіуса r |
| circle(r, angle) | Намалювати дугу радіуса \|r\| з градусною мірою angle |
| dot(size, color) | Намалювати точку діаметра size кольору color. Параметр color необов'язковий |
| clear() | Очистити область малювання |

Розглянемо приклади застосування команд.

**2** Намалюємо трикутник, використовуючи команди малювання. Уведіть ці команди у вікні консолі:

```
>>> from turtle import *
>>> forward(100)
>>> right(120)
>>> forward(100)
>>> right(120)
>>> forward(100)
```

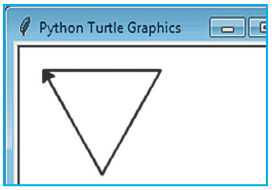



Рис. 25.4

На рис. 25.4 наведено результат виконання команд малювання.

**3** Намалюємо коло радіуса 50 з центром у точці (0,0) і товщиною лінії 3.

Команда circle(50) малює коло радіуса 50, яке знаходиться зліва від положення Черепашки. Вона на початку дивиться вправо, тому C (точка початку малювання) буде найнижчою. Знайдемо координати точка C: оскільки радіус кола за умовою дорівнює 50 пікселів, координати точка O(0; 0), то OC = 50. Точка C знаходиться на 50 пікселів нижче від осі Ox, її координати C(0; –50) (рис. 25.5).

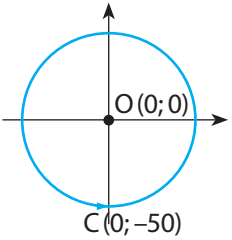



Рис. 25.5

Алгоритм малювання буде таким:

```
from turtle import *
width(3)
up()  # Підняти перо, щоб при пересуванні у т. C не залишати сліду
goto(0, –50)                  # Пересунути Черепашку в точку C
down()              # Опустити перо, щоб почати залишати слід
circle(50)          # Намалювати коло
```

**4** Намалювати смайлик (рис. 25.6).

```
from turtle import *
circle(50)          # Малювання обличчя
up()
goto(–20, 60)
down()
dot(20)             # Малювання лівого ока
up()
goto(20, 60)
down()
dot(20)             # Малювання правого ока
up()
goto(20, 40)
left(90)
down()
circle(20, –160)    # Посмішка
```

Рис. 25.6

## Питання для самоперевірки

1. Яку команду потрібно застосувати для завантаження команд роботи із Черепашкою?
2. Запишіть оператор для виведення поточних координат Черепашки.
3. Поясніть призначення команд down(), up().
4. Запишіть команди для Черепашки: 1) пройти вперед 20 кроків; 2) намалювати коло радіуса 30 пікселів; 3) підняти перо; 4) повернутися на 30° проти годинникової стрілки.
5. Черепашка знаходиться у точці (100; 100). Які координати матиме Черепашка після виконання команд малювання?

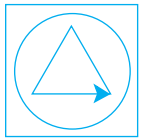
а)
```
forward(50)
right(90)
forward(50)
```

б)
```
forward(50)
left(90)
forward(50)
```

в)
```
left(90)
forward(50)
forward(50)
```

г)
```
forward(50)
forward(50)
left(90)
forward(50)
```

6. Напишіть програми для створення зображень:

а)  б)  в)  г) 

## Вправа 25. «Черепашача» графіка

Завдання: скласти програму малювання човника (рис. 25.7).

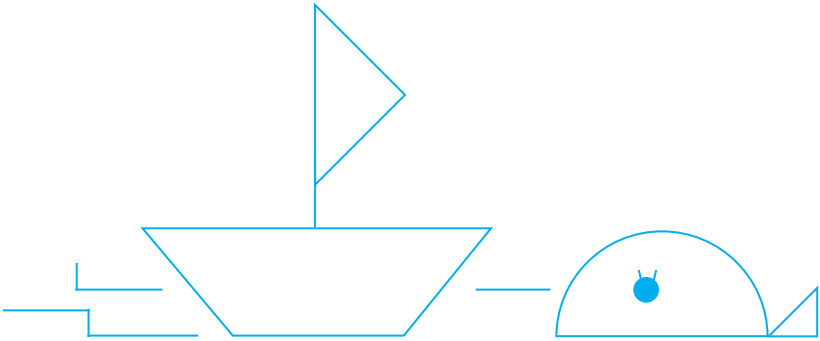


Рис. 25.7

*Під час роботи за комп'ютером дотримуйтесь правил безпеки.*

**1** Відкрийте вікно IDLE і створіть нове вікно програми. Збережіть програму у файлі Вправа28.
Завантажте модуль turtle.
Задайте ширину сліду Черепашки 3 пікселі.
Запишіть оператори, що виконують указані дії.

2 бали

**2** Визначте координати кінців відрізків, які утворюють зображення човника.

2 бали

Наприклад, якщо почати малювання з точки O (0; 0), точка **A** може мати координати (–40; 50), а точка B (120; 50) (рис. 25.8).
Запишіть координати точок C, D, E, F, G.

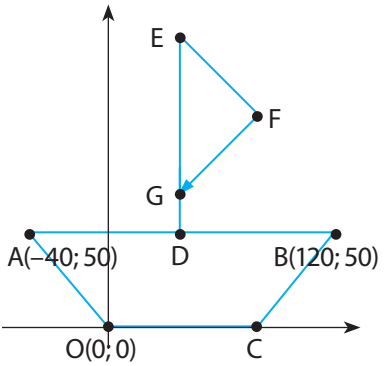



Рис. 25.8

**3** Запрограмуйте пересування Черепашки по визначених точках. Заповніть пропуски: 2 бали

```
goto(-40, 50)                    # Побудова відрізка OA
goto(120, 50)                    # Побудова відрізка AB
                                 # Побудова відрізка BC
                                 # Побудова відрізка CO
up()
goto(                          ) # Перехід у точку D
down()
                                 # Побудова відрізка DE
```

**4** Додайте зображення кита. Для цього слід перейти в точку (250,0), повернути Черепашку вліво на 90° (щоб дуга була зліва від Черепашки), і намалювати дугу з градусною мірою 180°. 2 бали

```
up()
goto(250,0)
down()
left(90)
circle(50, 180)                  # Малювання спини кита
goto(270,0)
goto(270,20)                     # Малювання хвоста кита
goto(250,0)
```

**5** За допомогою команди dot() намалюйте око кита. Запишіть оператор, що реалізує вказану дію: 2 бали

**6** Додайте кілька ліній, що зображують хвилі. 2 бали

**Комп'ютерне тестування**

Виконайте тестове завдання 25 з автоматичною перевіркою результату.

# § 26. Алгоритми створення зображень

У попередньому параграфі ви ознайомились із системою команд Черепашки та навчилися визначати її координати у графічному вікні, створювати малюнки пером чорного кольору. Модуль turtle надає можливості створення яскравих різнокольорових зображень, ми також можемо змінювати форму, розмір та колір самої Черепашки.

## Створення кольорового малюнка

Черепашка може залишати не лише чорний, а й кольоровий слід. Ви можете використовувати велику кількість різноманітних кольорів.

Назва кольору (англійською мовою) береться в одинарні лапки, наприклад 'red', 'yellow', 'green' тощо. Відвідайте сайт www.tcl.tk/man/tcl8.4/TkCmd/colors.htm, щоб переглянути повний список кольорів.

Також можна задати компоненти кольору в 16-річній системі числення, наприклад, '#31D115' — зелений колір, "#FF0000" — червоний.

Команди для **встановлення кольору сліду:**

| Команда | Призначення |
|---|---|
| color(s) | Установити s — колір лінії, яку малює Черепашка |
| color(s1, s2) | Установити s1 — колір сліду Черепашки, s2 — колір заливки замкненої фігури |

Команди для **зафарбовування замкнених фігур:**

| Команда | Призначення |
|---|---|
| fillcolor(s) | Установити s — колір заповнення. |
| begin_fill() | Почати стежити за Черепашкою для заповнення області |
| end_fill() | Заповнити кольором s2 область, пройдену Черепашкою, починаючи з begin_fill() |

1 У вікні консолі введемо команди для малювання червоного кола, зафарбованого синім кольором.

```
>>> color('red', 'blue')
>>> begin_fill()
>>> circle(50)
>>> end_fill()
```

Результат виконання команд малювання кола наведено на рис. 26.1.

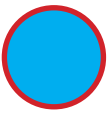


Рис. 26.1

## Змінення вигляду Черепашки

Щоб **змінити форму Черепашки**, використовуйте команду shape(<форма>), де <форма> — назва форми, узята в лапки. У модулі turtle існують такі форми Черепашки, як "arrow", "circle", "square", "triangle", "turtle", "classic". Наприклад, якщо ви бажаєте Черепашку у формі квадрата, напишіть команду shape("square").

Розмір Черепашки можна змінити за допомогою команди shapesize(n), де n — коефіцієнт збільшення розміру Черепашки. Наприклад, команда shapesize(2) збільшить Черепашку вдвічі.

Черепашка може залишати свої відбитки за допомогою команди stamp(). Після виконання цієї команди у вікні для графіки в місці, де була Черепашка, залишиться малюнок Черепашки.

Якщо потрібно, щоб після завершення малювання Черепашка не з'являлася на екрані, використовуйте команду hideturtle(). Щоб Черепашка знову показувалася, використовуйте команду showturtle().

**2** Накажемо Черепашці креслити лінію синього кольору, залишаючи сліди (рис. 26.2).

```
from turtle import*
shape('turtle')
color('blue')
stamp()
forward(50)
shapesize(2)
stamp()
forward(100)
shapesize(3)
stamp()
forward(100)
hideturtle()
```

Рис. 26.2

Повернути Черепашку на початок координат (у точку з координатами та y = 0) можна командою home(). Команда використовується без аргументів.

**3** Повернемо Черепашку в точку (0; 0) після малювання квадрата (рис. 26.3).

```
home()
print(position())
```

```
=== RESTART:
(0.00, 0.00)
>>>
```

**Рис. 26.3**

Змінювати можна не тільки вигляд Черепашки, але й деякі **властивості графічного вікна.**

| Команда | Призначення |
|---|---|
| bgcolor(<колір>) | Задати колір фону графічного вікна |
| setup(n, m) | Установити ширину вікна n пікселів, висоту m пікселів |
| reset() | Очистити вікно й перемістити Черепашку до центра вікна |

Якщо ви хочете, щоб графічне вікно закривалося по кліку миші, завершіть програму командою exitonclick().

**4** Задамо розміри вікна 200 × 100 пікселів, колір фону вікна — помаранчевий (рис. 26.4).

```
from turtle import *
setup(200, 100)
bgcolor('orange')
exitonclick()
```

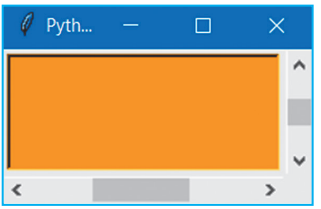



Рис. 26.4

## Додавання тексту до графічного вікна

У вікні для графіки модуля turtle можна малювати текст. Для відображення тексту використовується команда write(<текст>) — вивести текст:

write (text, move, align, font = (fontname, fontsize, fontstyle)),

де text — текст (у лапках), який буде відображатися у вікні для графіки turtle Python;

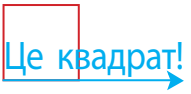
move — параметр, що відповідає появленню анімації Черепашки після відображення тексту. В анімації Черепашка підкреслює написаний текст. Параметр приймає лише логічні значення (True, False);

align — параметр, що відповідає за положення тексту щодо Черепашки; приймає значення "left", "right", "center". Зверніть увагу: якщо наявний параметр align, параметр move не спрацьовує;

font — параметри шрифту: fontname — назва шрифту (в лапках), fontsize — розмір шрифту, fontstyle — стиль тексту ("normal", "bold", "italic").

**5** Додамо текст у графічне вікно, застосовуючи різні значення параметрів тексту.

```
write('Це квадрат!', move = True, font = 'Arial 20')
```




```
write('Це квадрат!', align = 'right', font = ('Arial', 16,
'italic'))
```

## Питання для самоперевірки

1. Запишіть команди для Черепашки:
   1) установити зелений колір сліду Черепашки;
   2) установити жовтий колір фону вікна;
   3) установити синій колір заливки замкненої фігури.
2. Як виконати заливання графічного об'єкта заданим кольором? Опишіть алгоритм дій.
3. Обговоріть, які властивості графічного вікна доцільно змінювати при роботі з Черепашкою.
4. Побудуйте і зафарбуйте зеленим кольором коло радіусом 100, центр якого співпадає з центром вікна.
5. Запишіть фрагмент програми для побудови трикутника з вершинами в точках (100; 100), (150; 100), (80; 70); колір фону — сірий; колір ліній — червоний.
6. Запишіть фрагмент програми для побудови прямокутника з вершинами (80; 80), (170; 80), (80; 150). Зафарбуйте його жовтим кольором. Колір фону — синій.

## Вправа 26. Алгоритми створення зображень

Завдання: розфарбувати малюнок (рис. 26.5).

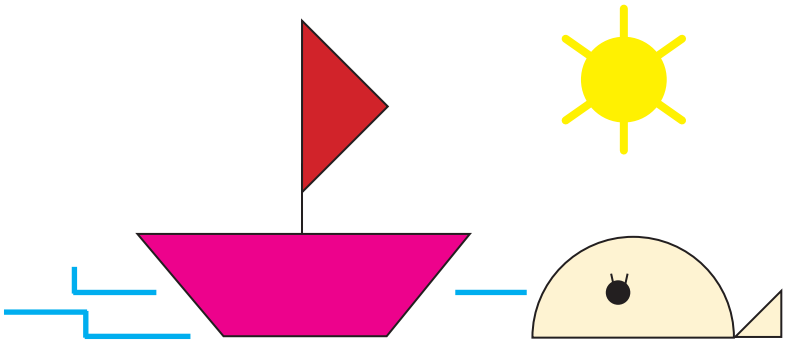


Рис. 26.5

*Під час роботи за комп'ютером дотримуйтесь правил безпеки.*

**1** Запустіть Python IDLE і відкрийте файл Вправа25. 2 бали

**2** Додайте заливку до корпусу човника: 2 бали

```
fillcolor('brown')
begin_fill()
goto(–40, 50)
goto(120, 50)
goto(80, 0)
goto(0, 0)
end_fill()
```

**3** Додайте заливку до вітрила човника. 2 бали

**4** Змініть колір хвиль на синій. 2 бали

**5** Зафарбуйте кита. 2 бали

**6** Намалюйте сонце. Заповніть пропуски в коді малювання сонця: 2 бали

```
color('orange','orange')
```

```
goto(200,200)
```

```
circle(30)
```

Додайте сонячні промені. Запишіть оператори малювання променів.

**Комп'ютерне тестування**

Виконайте тестове завдання 26 з автоматичною перевіркою результату.

# Практична робота 8
## Створення зображень за алгоритмами

Завдання: скласти програму малювання олімпійського символу.

*Задача*. Скласти програму малювання олімпійського символу.
Для малювання кожного кільця потрібно виконати такий алгоритм: задати колір, яким будемо малювати; підняти перо, що б під час переходу не малювати лінії; перейти на точку початку малювання наступного кільця; опустити перо; малювати коло з радіусом 45.

Обладнання: комп'ютер зі встановленим середовищем програмування Python.

**Хід роботи**

*Під час роботи з комп'ютером дотримуйтеся правил безпеки.*

**1.** Відкрийте вікно IDLE і створіть нове вікно програми. Напишіть оператори для завантаження команд для роботи з Черепашкою та встановлення ширини ліній:

```
from turtle import*
width(3)
```

**2.** Наберіть програмний код для малювання синього кільця:

```
color('blue')          # Встановлюємо колір синій
up()                   # Піднімаємо перо
goto(–110, –25)        # Переходимо в потрібну точку
down()                 # Опускаємо перо
circle(45)             # Малюємо коло з радіусом 45
```

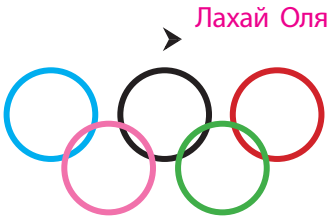
**3.** Напишіть оператори для малювання:

- чорного кільця з центром у точці (0, –25);
- червоного кільця з центром у точці (110, –25);
- жовтого кільця з центром у точці (–55, 75).

**4.** Напишіть оператори для малювання зеленого кільця.

**5.** Над олімпійським символом червоним кольором запишіть власне прізвище та ім'я (див. рисунок).

```
color('red')
up()
goto(0, 80)
down()
```




**6.** Збережіть файл у власній папці з ім'ям Практична8. Запустіть програму на виконання, проаналізуйте результат виконання програми у вікні Python Turtle Grafics. Закінчіть роботу, закривши вікно IDLE.

Зробіть висновок: як складати та виконувати лінійні алгоритми для створення зображень.

# § 27. Логічні вирази

Окрім уже відомих нам числового та рядкового типів даних, у мові Python є логічний тип bool. Змінна типу bool може набувати одного з двох значень — True (Істина) або False (Хибність).

Якщо змінній надати значення True, це буде змінна типу bool:

```
>>> type(True)
<class 'bool'>
>>> a = True
>>> print(a)
True
```

Логічний тип даних отримав свою назву на честь англійського математика Джорджа Буля (рис. 27.1), засновника математичної логіки — розділу математики, що побудований на застосуванні математичних методів для розв'язування логічних задач. Сьогодні ідеї Буля використовуються у всіх сучасних цифрових пристроях.

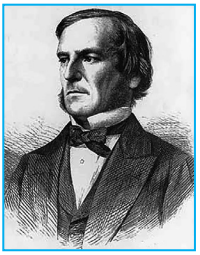


Рис. 27.1

## Логічні операції

**Логічними** є вирази, результатом яких є True або False.

Умови в програмі записуються у вигляді логічних виразів.

Простий логічний вираз — **проста умова** — утворюється за допомогою логічних операцій. Розглянемо таблицю:

| Знак операції | Значення | Логічний вираз | Результат |
|---|---|---|---|
| == | Дорівнює | 8 == 9 | False |
| > | Більше | 8 > 9 | False |
| < | Менше | 8 < 9 | True |
| >= | Більше або дорівнює | 5 >= 5 | True |
| <= | Менше або дорівнює | 5 <= 2 | False |
| != | Не дорівнює | 2 != 5 | True |

**1** Проаналізуємо результати обчислення логічних виразів у вікні консолі:

```
>>> books = 10
>>> books == 5        # Перевірка, чи дорівнює books 5
False
>>> books < 10        # Перевірка, чи є books меншим за 10
False
>>> books >= 5        # Перевірка, чи books більше або дорівнює 5
True
>>> books != 10       # Перевірка, чи не дорівнює books 10
False
```

Значення логічного виразу можна зберегти у змінній.

**2** Обчислимо значення логічних виразів:

```
x = 5
y = 2
a = x>y                # a = True
a = x<y                # a = False
a = x–4.5<y*2          # a = True
```

Бувають ситуації, коли одночасно необхідно перевірити виконання кількох умов.

**Складена умова** — це кілька простих умов, з'єднаних логічними операціями and (логічне І, інакше — логічний добуток), or (логічне АБО, інакше — логічна сума), not (логічне заперечення).

Складена умова A and B = True, тільки якщо й A, і B істинні.

Складена умова A or B = False, якщо й A, і B хибні.

**3** Розглянемо приклади складених умов:

- not a <= 3 — рівнозначне виразу $a > 3$;
- age >= 10 and age <= 18 — істинне тоді й тільки тоді, коли значення age розташовується в проміжку від 10 до 18 включно;
- age < 10 or age > 18 — істинне для всіх значень age, які не належать проміжку від 10 до 18.

**4** Проаналізуємо результати обчислення логічних виразів у вікні консолі.

```
>>> books = 10
>>> books == 10 or books == 5
# Перевірка, чи дорівнює books 10 або 5
True
>>> books == 10 and books == 5
# Перевірка, чи дорівнює books одночасно 10 і 5
False
```

**5** Визначити, чи належить точка з координатою $x$ відрізку $[-5; 5]$ (рис. 27.2).

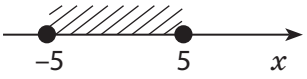


Рис. 27.2

Точка належить відрізку, якщо справджується нерівність $-5 \leqslant x \leqslant 5$. У програмуванні таку подвійну нерівність записують як складену умову:

```
x >= –5 and x <= 5
```

**6** Обчислити значення логічних виразів при $x = 1$; $y = 2$; $z = 3$.

```
a = (x < y) and (y < z)        # a = True
a = (x > y) or (y > z)         # a = False
a = not(x > y)                 # a = True
```

## Питання для самоперевірки

1. Яких значень може набувати змінна логічного типу?
2. Які операції можна виконувати над змінними логічного типу?
3. Запишіть мовою програмування прості умови:
   а) $x$ більше 10; б) $x$ — парне число.
4. Запишіть мовою програмування складені умови:
   а) $2 < x < 10$; б) $x$ не належить проміжку (2, 10).
5. Запишіть у вигляді логічних виразів подані умови:
   а) $3 \leqslant x \leqslant 10$; б) $-5 \leqslant x \leqslant 5$ за умови, що $x \neq 0$.
6. Обчисліть значення логічних виразів:
   а) (A <= B) and (A = B–1), якщо A = 2, B = 4;
   б) (A <= B) or (A = B–1), якщо A = 2, B = 4.

# Вправа 27. Логічні вирази

**Завдання:** скласти програму, в якій у залежності від значень змінних A та B обчислюються значення логічних виразів, а результат обчислення виводиться у вікно консолі (рис. 27.3).

```
a = 18
b = 12
a>b = True
b<=10 = False
x and y = False
x or y = True
not x and y = False
not (x and y) = True
>>>
```

**Рис. 27.3**

*Під час роботи за комп'ютером дотримуйтесь правил безпеки.*

**1** Запустить Python IDLE і створіть файл Вправа27. Завантажте модуль random для генерації випадкових чисел. 2 бали

**2** Надайте змінній a випадкове значення з проміжку від 1 до 20, виведіть значення a у вікно консолі: 2 бали

```
a = randint(1, 20)
print ("a= ", a)
```

Надайте змінній *b* випадкове значення з проміжку від 1 до 20, виведіть значення *b* у вікно консолі. Запишіть ці оператори.

**3** Змінній *x* присвойте значення логічного виразу $a > b$, виведіть значення *x*: 2 бали

```
x = a>b
print("a>b = ", x)
```

**4** Змінній *y* присвойте значення логічного виразу $b <= 10$, виведіть значення *y*: 2 бали

```
y = b<=10
print("b<=10 = ", y)
```

**5** Виведіть у консоль значення складної умови *x* and *y*:
print("x and y = ", x and y)

2 бали

**6** Виведіть у консоль значення логічних виразів x or y, not x and y; not(x and y).
Виконайте програму кілька разів, проаналізуйте значення логічних виразів.

2 бали

Запишіть результат виконання програми за зразком.

**Комп'ютерне тестування**

Виконайте тестове завдання 27 з автоматичною перевіркою результату.

# § 28. Алгоритми і програми з розгалуженнями. Оператор if

Програма має вміти вибирати різні дії в залежності від уведених даних. У § 27 ви ознайомилися з алгоритмічною структурою «Розгалуження» і розглянули приклади алгоритмів з розгалуженнями. Для перевірки істинності умов і організації розгалуження у програмах мовою Python призначені умовні оператори if та if-else.

**Оператори** — це команди програми.

## Умовний оператор if

Оператор if призначено для виконання деякої послідовності дій у тому випадку, якщо істинною є зазначена умова. Цей умовний оператор відповідає алгоритмічній конструкції «неповне розгалуження» (рис. 28.1).

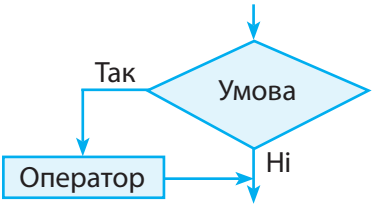



Рис. 28.1

**Синтаксис умовного оператора в неповній формі:**

if <Умова>:

____ <Оператор>

де ____ — обов'язковий відступ від лівого краю.

Після запису умови слід поставити двокрапку, яка показує, що далі має бути розташований блок дій. Блок дій записується з обов'язковим однаковим відступом від лівого краю.

Оператор if перевіряє істинність зазначеної умови. Якщо умова приймає значення True (Істина), то програма виконає дію, зазначену в частині <Оператор>. Якщо ж умова приймає значення False (Хибність), то блок <Оператор> пропускається, і керування передається оператору, що йде після оператора if.

## Приклади використання неповного розгалуження у програмах

Неповне розгалуження використовують тоді, коли деяку послідовність команд слід виконати лише за умови істинності висловлювання. Якщо ж записане в умові висловлювання хибне, то жодна з команд не виконується.

**1** Після виконання цієї програми у вікні консолі отримаємо результат Правильно.

```
books = 10
if books == 10:
    print('Правильно!')
```

**2** Вітання Вітаю! виводиться, якщо користувач уводить літеру у.

```
answer = input('Сьогодні твій день народження? (y/n)')
if answer == 'y':
    print('Вітаю!')
print('Гарного дня!')
```

Приклад виконання команди наведено на рис. 28.2.

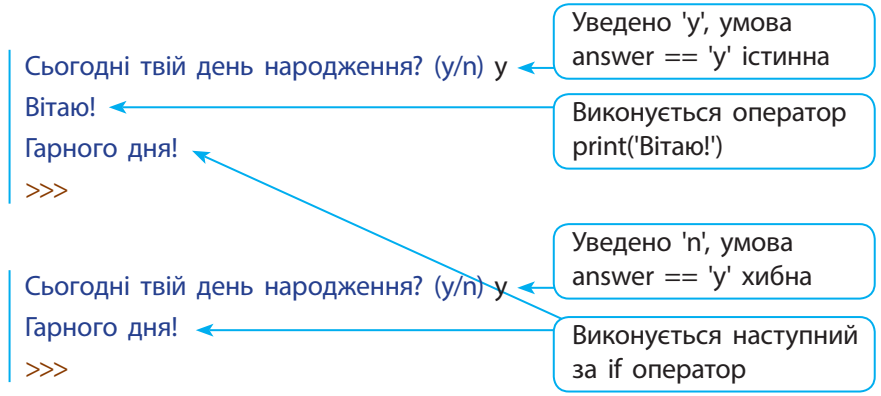



Рис. 28.2

**3** Нехай потрібно збільшити значення змінної $a$ на одиницю, якщо її поточне значення менше 5. Оператор розгалуження має вигляд:

```
if a<5:
    a = a+1
```

Оператор a = a+1 виконується тільки в тому випадку, коли істинна умова a<5:

| Початкове значення a | Значення умови | Оператор a = a+1 | Значення a після виконання оператора if |
|---|---|---|---|
| a=1 | True | виконується | 2 |
| a=5 | False | пропускається | 5 |
| a=10 | False | пропускається | 10 |

**4** Визначимо, чи є число $n$ кратним 2, 3 або 5 (рис. 28.3).

```
n = int(input("n =? "))
```

```
n =? 27
27 кратне 3
```

Рис. 28.3

```
if n%2 == 0:
    print(n, "парне число")
if n%3 == 0:
    print(n, "кратне 3")
if n%5 == 0:
    print(n, "кратне 5")
```

У програмі використовуються послідовні розгалуження, тобто розгалуження, що йдуть одне за одним. Програма по черзі перевіряє умову кожного розгалуження і, якщо умова істинна, виконує блок <оператор>, після цього програма переходить до перевірки умови наступного розгалуження і подальших операцій.

**!** Перевірка наступної умови розгалуження не залежить від результату попередньої.

## Питання для самоперевірки

1. Як записується й виконується умовний оператор у неповній формі?
2. Яких значень набудуть змінні $a$ і $b$ після виконання умовних операторів, наведених нижче, для початкових значень $a = 3$; $b = 10$?

а)
```
if a%2 == 0:
    a = a//2
```
б)
```
if a! = b:
    a = b
```
в)
```
if a<b:
    b = b–a
```
г)
```
if b >= 10:
    a = b–a
    b = 0
```

3. Виберіть оператори, виконання яких не викличе повідомлення про помилку:

а)
```
if a>b:
    print ("a>b")
```
б)
```
if a%3:
    a = a//3
```
в)
```
if a<10:
    a = a+5
```
г)
```
if b>10:
    b = b–10
```

4. Запишіть оператори розгалуження, які реалізують такі дії:

а) Якщо $a$ більше $b$, замінити $a$ нулем.

б) Якщо ціле число $a$ парне, поділити його на 2.

в) Якщо числа $a$ і $b$ не рівні, змінній $a$ надати значення $b$.

5. Скласти програму, яка запитує значення числа $n$, $1 < n < 10\,000$, і повідомляє кількість цифр у числі $n$.

6. Скласти програму, яка запитує значення температури тіла людини і визначає, здорова людина чи хвора.

- Якщо температура тіла менша 36°, то виводиться порада зігритися.
- Якщо температура від 36 до 37 градусів, людина здорова.
- Якщо температура вища 37°, виводиться порада звернутися до лікаря.

## Вправа 28. Малювання багатокутників

Завдання: скласти програму, яка запитує значення $n$ кількості кутів багатокутника і креслить фігуру, що має $n$ кутів.

*Під час роботи за комп'ютером дотримуйтесь правил безпеки.*

**1** Запустіть Python IDLE і створіть файл Вправа28. Завантажте модуль для роботи з «черепашачою» графікою. 2 бали

**2** Запишіть оператор для введення значення $n$. 2 бали

**3** Якщо $n = 3$, програма креслить трикутник. Запишіть оператор неповного розгалуження, який перевіряє умову n == 3. Блок дій, який виконується, якщо умова істинна, має містити оператори малювання трикутника. 2 бали

```
if n==3:
    forward(50)
    right(120)
    forward(50)
    right(120)
    forward(50)
    right(120)
```

**4** Якщо n = 4, програма має накреслити квадрат. Запишіть оператор неповного розгалуження, який перевіряє умову n == 4 і, якщо умова істинна, креслить квадрат. 2 бали

**5** Якщо $n = 5$, програма креслить п'ятикутник. Запишіть оператор неповного розгалуження, який перевіряє умову n == 5 і, якщо умова істинна, креслить п'ятикутник. 2 бали

**6** Запишіть оператор розгалуження, який виконує таку дію: якщо n < 3 або n > 5, вивести повідомлення: Такої фігури не передбачено. 2 бали

Перевірте роботу програми.

**Комп'ютерне тестування**

Виконайте тестове завдання 28 з автоматичною перевіркою результату.

# § 29. Алгоритми і програми з розгалуженнями. Оператор if ... else

Ви вже знаєте, що у випадках, коли нам потрібно, щоб наша програма виконувала одні дії, якщо деяка умова істинна, та інші дії, якщо ця умова хибна, використовується алгоритмічна конструкція повне розгалуження.

Зазначеній конструкції відповідає умовний оператор if ... else (рис. 29.1).

**Синтаксис умовного оператора в повній формі:**

```
if <умова> :
    <оператор 1>
else:
    <оператор 2>
```

Якщо умова істинна (True), програма виконає блок дій <Оператор 1>.

Якщо умова хибна (False), виконується блок дій <Оператор 2>, який міститься після службового слова else.

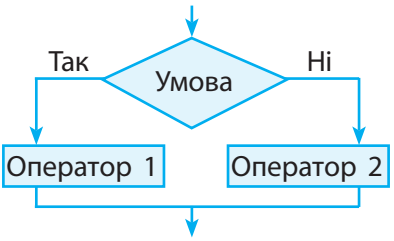


Рис. 29.1

**1** Якщо користувач або користувачка уводить літеру «у», виводиться вітання Вітаю!, інакше — текст Щасти тобі!.

Після виконання одної з гілок оператора if ... else програма переходить до виконання наступного оператора.

```
answer = input('Сьогодні твій день народження? (y/n)')
if answer == 'y':
    print('Вітаю!')
else:
    print('Щасти тобі!')
print('Гарного дня!'
```

**2** Оператор if ... else реалізує таку дію: якщо a > b, то змінні міняються значеннями так, щоб виявилося a < b.

```
if a>b :
    c = a
    a = b
    b = c
    print('a і b помінялися значеннями')
else :
    print('обміну значеннями не потрібно')
print('a = ', a, 'b = ', b)
```

Зверніть увагу на відступи від лівого краю в прикладі 2: команди, вкладені в гілки оператора if, записані на одній вертикалі. Результат виконання програми наведено на рис. 29.2.

```
a = 5 b = 2
a і b помінялися значеннями
a = 2 b = 5
>>>
a = 3 b = 8
обміну значеннями не потрібно
a = 3 b = 8
>>>
```

Рис. 29.2

**3** Оператор if ... else перевіряє, чи існує трикутник зі сторонами a, b, c. Правило трикутника: сума двох будь-яких сторін повинна бути більша за третю.

```
if (a<b+c) and (b<a+c) and (c<a+b):
    print('Трикутник з такими сторонами існує')
else: print('Трикутника з такими сторонами не існує')
```

**4** У програмі виконуються дії: значення змінної a вводиться з клавіатури; якщо значення a є парним числом, то a цілочисельно ділиться на 2, інакше значення a збільшується на 1 (рис. 29.3).

```
a = int(input('a = ?'))
if a%2 == 0:
    a = a//2
else:
    a = a+1
print('нове значення a', a)
```

```
a = ?  34
нове значення a 17
>>>
a = ?  35
нове значення a 36
>>>
```

Рис. 29.3

**!** У програмі можуть послідовно виконуватись умовні оператори повного і неповного розгалуження.

**5** Скласти програму пошуку найбільшого з трьох чисел a, b, c. Блок-схему алгоритму пошуку найбільшого з трьох чисел та результат виконання програми наведено на рис. 29.3.

```
a = int(input("a = ?"))
b = int(input("b = ?"))
c = int(input("c = ?"))
if a>b: m = a
else: m = b
if c>m: m = c
print('m = ', m)
```

```
a = ?  3
b = ? 12
c = ?  8
m =  12
>>>
```

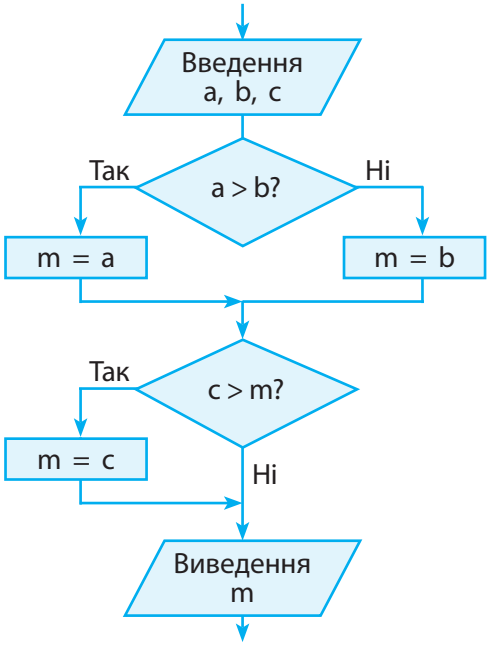



Рис. 29.3

Отже, умовні оператори if та if ... else є важливими інструментами програмування, які стануть вам у пригоді під час створення тестових та ігрових програм.

## Питання для самоперевірки

1. Як виконується умовний оператор у повній формі?
2. Як виконуються логічні операції and, or, not?
3. Яких значень набудуть змінні $a$ і $b$ після виконання умовних операторів, наведених нижче, для початкових значень $a = 3$; $b = 5$?

**4.** Яких значень набудуть змінні $a$ і $b$ після виконання умовних операторів, наведених нижче, для початкових значень $a = 3$; $b = 5$?

а)
```
if a > b:
    a = 0
else :
    b = 0
```

б)
```
if a%3 == 0:
    a = a//3
```

в)
```
if a > b:
    a = a+10
else :
    b = b+10
```

г)
```
if a! = b:
    a = b
```

ґ)
```
if a < b:
    a = 2*a
else :
    b = b*a
```

**5.** Запишіть оператори розгалуження, які реалізують такі дії:

а) Змінній $m$ присвоїти значення меншого з чисел $a$ і $b$.

б) Якщо ціле число $a$ парне, поділити його на 2, інакше збільшити a на 1.

в) Значення більшого з чисел $a$ і $b$ замінити нулем.

**6.** Складіть програму, в якій перевіряється уведене з клавіатури число:

- якщо число менше 40, то виводиться повідомлення «ВЛУЧИВ»;
- якщо число більше 40 — повідомлення «ПЕРЕЛІТ».

## Вправа 29. Алгоритми і програми з розгалуженнями. Оператор if ... else

Завдання: скласти програму, яка визначає, чи достатньо користувачеві років, щоб керувати автомобілем.

*Задача*. Потрібно ввести вік користувача та зберегти значення у змінній age; якщо age >= 18, то вивести повідомлення Ваш вік дозволяє керувати авто, інакше: змінній r присвоїти значення виразу (18–age), вивести повідомлення, через скільки років користувач зможе отримати посвідчення.

*Під час роботи за комп'ютером дотримуйтесь правил безпеки.*

**1** Відкрийте вікно IDLE і створіть нове вікно програми. Збережіть файл у власній папці з ім'ям Вправа32. 2 бали

**2** Запишіть оператор введення з клавіатури значення змінної age. 2 бали

**3** Заповніть пропуски в операторі розгалуження, який реалізує таку дію: якщо age >= 18, то вивести повідомлення Ваш вік дозволяє керувати авто, інакше: змінній r присвоїти значення виразу (18–age) та вивести повідомлення, через скільки років користувач зможе отримати водійське посвідчення. 2 бали

```
if 
print(  )
else:
r = 
print('Доведеться почекати ',  , 'років')
```

**4** Уведіть текст програми. 2 бали

**5** Запустіть програму на виконання, проаналізуйте результат виконання програми у вікні консолі. 2 бали

**6** Запишіть результати виконання програми для вказаних значень змінної age: 2 бали

age = 11 r =

age = 18 r =

age = 20 r =

Закінчіть роботу, закривши вікно IDLE.

**Комп'ютерне тестування**

Виконайте тестове завдання 29 з автоматичною перевіркою результату.

# Практична робота 9
## Складання та виконання алгоритмів із розгалуженнями

**Завдання:** скласти програму для обчислення коренів рівняння $ax+b=c$.

**Обладнання:** комп'ютер зі встановленим середовищем програмування Python.

### Хід роботи

*Під час роботи з комп'ютером дотримуйтеся правил безпеки.*

**1.** Проаналізуйте словесний алгоритм розв'язування рівняння:
якщо $a=0$, $b=c$, то коренем рівняння є будь-яке число;
якщо $a=0$, $b\neq c$, то коренів немає;
якщо $a\neq 0$, то $x=(c-b)/a$.

**2.** Відкрийте вікно IDLE і створіть нове вікно програми. Збережіть файл у власній папці з іменем Практична10.

**3.** Складіть програму на основі блок-схеми (рис. 1).
Заповніть пропуски в програмі, що складена на основі блок-схеми:

```
a = int(input(          ))
b =
c =
if                      :


if                      : print ('безліч коренів')
if                      : print ('корені відсутні')
```

**4.** Запустіть програму на виконання. Перевірте роботу програми для тестового набору значень (рис. 2).

**5.** Випробуйте програму для таких наборів коефіцієнтів:

а) $a = 5;\ b = 3;\ c = 3;$ x =

б) $a = 0;\ b = 2;\ c = 17;$ x =

в) $a = 0;\ b = 4;\ c = 4.$ x =

**6.** Закінчіть роботу, закривши вікно IDLE.

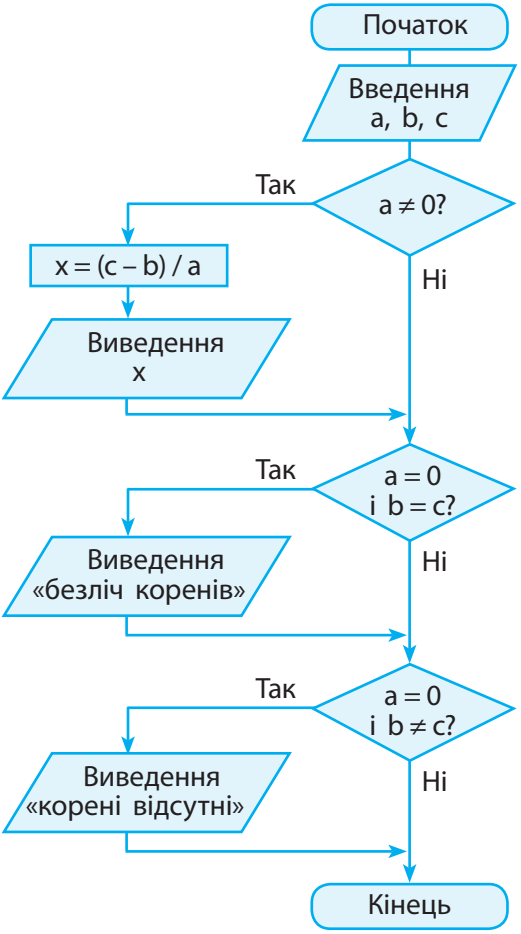



Рис. 1

```
=== RESTART:
a = ? 5
b = ? 2
c = ? 17
x =  3.0
```

Рис. 2

Зробіть висновок: як складати та виконувати алгоритми з розгалуженням для розв'язування задачі.

# Зміст

## Розділ 5. АЛГОРИТМИ ТА ПРОГРАМИ

Навчальне видання
*БОНДАРЕНКО Олена Олександрівна*
*ЛАСТОВЕЦЬКИЙ Василь Васильович*
*ПИЛИПЧУК Олександр Павлович*
*ШЕСТОПАЛОВ Євген Анатолійович*

**ІНФОРМАТИКА**
**5 КЛАС**
**Навчальний посібник**
**Частина 4**

Редактор *Л. А. Каюда*
Верстка *І. І. Пікальова*

ТИ1575003У.
Підписано до друку 24.12.2021.
Формат 70×90/16. Папір офсетний.
Гарнітура Шкільна. Друк офсетний.
Ум. друк. арк. 5,85.

ТОВ Видавництво «Ранок»,
вул. Космічна, 21а, Харків, 61145.
Свідоцтво суб'єкта видавничої справи
ДК № 7548 від 16.12.2021.

E-mail: office@ranok.com.ua
Тел. (057) 701-11-22.

Регіональні представництва видавництва «Ранок»:
Київ — тел. (073) 680-33-55, e-mail: office.kyiv@ranok.com.ua,
Львів — тел. (067) 269-00-61, e-mail: office.lviv@ranok.com.ua

З питань придбання продукції видавництва «Ранок» звертатися за тел.:
у Харкові — (057) 727-70-80;
Києві — (067) 449-39-65, (093) 177-05-04;
Вінниці — (067) 534-51-62;
Дніпрі — (067) 635-19-85;
Житомирі — (067) 122-63-60;
Львові — (067) 340-36-60;
Миколаєві та Одесі — (067) 551-10-79;
Черкасах — (0472) 51-22-51;
Чернігові — (067) 440-88-93.
E-mail: commerce@ranok.com.ua

«Книга поштою»: вул. Котельниківська, 5, Харків, 61051.
Тел. (057) 727-70-90, (067) 546-53-73.
E-mail: pochta@ranok.com.ua

**www.ranok.com.ua**

НОВА УКРАЇНСЬКА ШКОЛА

# інформатика

## 5 клас частина 4

### Переваги навчального посібника:

- Використання багатоплатформного вільного програмного забезпечення
- Зорієнтованість змісту вправ на життєвий досвід дітей, підготовку до життя в сучасному інформаційному суспільстві
- Покроковий опис виконання практичних робіт

### До навчального посібника додаються:

- календарно-тематичний план
- методичні рекомендації
- інтернет-підтримка на новітній освітній платформі IZZI

ISBN 978-617-09-7397-9
9 786170 973979
2472 9529