Бондаренко О.О., Ластовецький В.В.,
Пилипчук О.П., Шестопалов Є.А.

Інформатика
6 клас

Тимчасовий підручник

Відповідає вимогам програми МОН України для 5-9 класів
від 07.06.2017 року

Робочий зошит і календарний план
можна безкоштовно звантажити з сайту https://aspekt.in.ua/

Розділи 1 і 2 відповідають програмі для 6 класу, розділ 3 повторено з 5 класу
для вивчення нового середовища програмування мовою Python

У наступному 2018-2019 н.р. буде забезпечена
повна відповідність програмі для 6 класу

Шепетівка
«Аспект»
2017

Рекомендується для 6-х класів загальноосвітніх навчальних закладів різних профілів, відповідає вимогам діючих програм з інформатики, базується на використанні операційної системи Windows'7 і 10, офісного пакета Office 2007, 2010 або 2013 та середовища програмування Python.

Просто і доступно описані: комп'ютерна графіка, створення презентацій, основи алгоритмізації та програмування мовою Python. У кінці кожної теми є питання для самоперевірки.

Вправи до кожного уроку і практичні роботи для закріплення набутих знань та формування практичних навичок зібрані в окремому робочому зошиті

Любі п'ятикласники!

У 6-му класі ви будете розвивати та доповнювати одержані у 5-му класі і в молодшій школі знання та навички з інформатики – науки про цікавий та захоплюючий світ інформації та комп'ютерних технологій.

Старанно вивчаючи теоретичний матеріал, виконуючи вправи і практичні роботи, ви будете більш впевнено працювати з комп'ютером, збільшуючи з кожним уроком свої знання та навички.

Слово до вчителя

Навчальний матеріал підручника розрахований на вивчення інформатики в обсязі 1 години на тиждень (35 годин за навчальний рік) та орієнтований на практичне використання комп'ютерів, починаючи з першого уроку.

Навчальний матеріал кожного параграфу вивчається, закріплюється практичними навичками та оцінюється протягом одного уроку за методикою «перевернутого навчання» (детальніше див. http://aspekt-edu.kiev.ua/).

Кожен параграф посібника вивчається протягом одного уроку, має структуру, що відповідає санітарним нормам: освоєння нового матеріалу – 20 хв., виконання письмових завдань без використання комп'ютерів і практична робота з комп'ютером – 25 хв.

Сім обов'язкових до виконання практичних робіт виконуються на окремих уроках після вивчення розділу.

Пробний випуск для апробації
в умовах навчального процесу

Зауваження та пропозиції щодо змісту підручника надсилайте

на адресу: aspekt@aspekt.in.ua

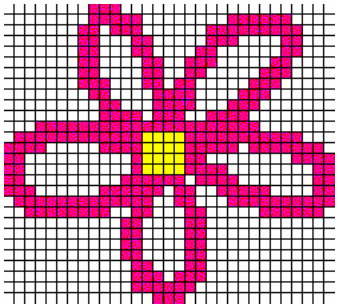
Зміст

1. Комп'ютерна графіка

Урок 1. Основні поняття комп'ютерної графіки

Поняття комп'ютерної графіки

Якщо придивитись до екрана комп'ютера, можна помітити, що зображення на ньому складається з окремих крапок (їх ще називають пікселями, від англійських слів PICture ELement – елемент малюнка).



Кожен з пікселів може набувати одного з кольорів.
Неможливо змінити колір частини пікселя

Оскільки пікселі дуже малі, то розглядаючи екран ми не помічаємо окремих крапок і бачимо суцільне зображення.

Як відомо, вся інформація, яку обробляє комп'ютер, кодується за допомогою чисел. Але як закодувати числами малюнок? Для цього є два способи: растровий і векторний.

Растрові зображення, їх властивості

Спробуйте намалювати квітку на листку в клітинку, зафарбовуючи кожним кольором лише цілі клітинки. Ви отримаєте зображення, подібне до наведеного на малюнку. Позначивши кожен з кольорів числом, отримаємо кодову таблицю кольорів:

Колір	Код	Колір	Код	Колір	Код	Колір	Код
чорний	0	зелений	2	червоний	4	жовтий	6
синій	1	коричневий	3	фіолетовий	5	білий	7

Тепер є можливість закодувати малюнок послідовністю чисел: перші два числа – кількість клітинок по довжині і ширині малюнка, наступні – коди кольорів клітинок, перелічені зліва направо рядок за рядком: 31, 30, 7, 7, 7, 7, 7, 7, 7, 7, 4, 4, 4, 7, 7, 7 і т.д. Зображення, закодоване таким чином, називають точковим або растровим.

Щоб перенести растрове зображення на екран, досить повідомити системі, які кольори повинні мати пікселі у певному місці екрану.

Растрове кодування використовується для зберігання фотографій, малюнків, сканованих зображень тощо.

Основна перевага растрової графіки – можливість роботи з фотографічними та сканованими зображеннями.

Для роботи з растровою графікою розроблено багато програм, які відрізняються набором засобів для редагування малюнків: для початківців – Paint, KolourPaint; для професіоналів – Adobe Photoshop та GIMP тощо.

Параметри растрових зображень

Кожний растровий малюнок має певну кількість крапок по горизонталі й вертикалі. Від кількості крапок залежить якість зображення, адже, якщо крапок більше, то ними можна зобразити більше окремих деталей зображення. Наприклад, типові розміри екрана монітора в пікселях – 800x600, 1024x768, 1240x1024 і більше. Цифрове фото може мати 2048x1536, 4320x3240 тощо.

Крім того, важливим є те, скільки крапок розміщується на одиниці довжини зображення, бо від цього залежить розмір самих крапок. Цю величину називають роздільною здатністю (або роздільністю) і, здебільшого, вимірюють у «крапках на дюйм» (dpi – dots per inch).

Довжина, ширина і роздільність разом визначають розмір малюнка.

На малюнку відтворено три зображення однакових геометричних розмірів, але з різною роздільністю: 96, 48 та 24 крапок на дюйм. Очевидно, що чим більше пікселів припадає на одиницю довжини, тим вища

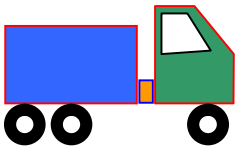
якість малюнка.

Крім розмірів, малюнок характеризується кольором кожного пікселя. Чим більше різних кольорів містить зображення, тим краще воно може відтворювати вигляд реальних об'єктів. Але для кодування більшої кількості кольорів потрібні довші кодові послідовності, що тягне за собою збільшення файлу.

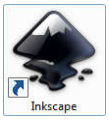
Таким чином, для створення або збереження растрового малюнка необхідно зазначити його розміри та колір кожного пікселя. Для точнішого відтворення слід вказати також роздільність.

Векторні зображення, їх властивості

У багатьох випадках виявляється зручнішим зазначати не колір окремих точок малюнка, а властивості та розміщення окремих його елементів (ліній, многокутників, криволінійних фігур).



Наприклад, зображення автомобіля (див. малюнок) складається з двох прямокутників, трьох кіл, двох неправильних многокутників. Опишемо властивості одного з елементів: прямокутник довжиною 1,74 см та шириною 1,02 см, колір заливки – 1, колір лінії – 4, товщина лінії 0,5 мм, координати лівого верхнього кута (50; 120). Якщо таким чином описати всі фігури – отримаємо векторний код даного малюнка.



Основна перевага векторної графіки – можливість довільного масштабування та повертання зображень без втрати якості

Для роботи з векторними зображеннями застосовують програми Inkscape, Corel Draw, Adobe Illustrator та інші.

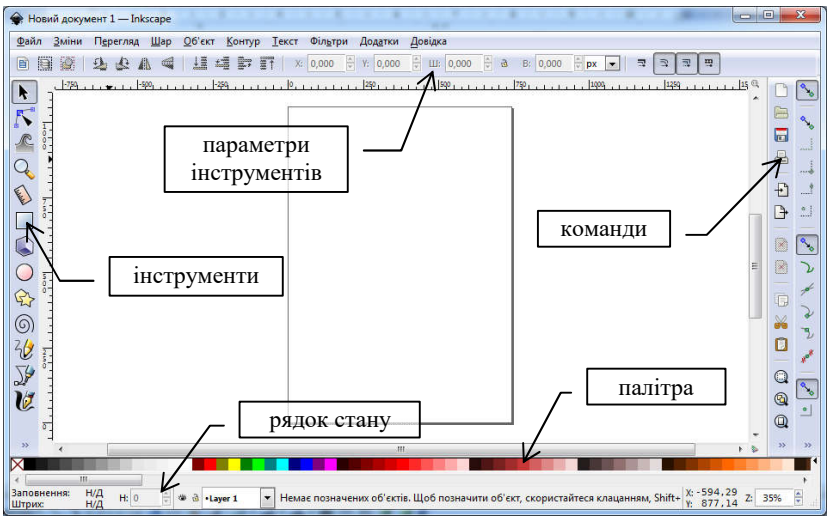
Параметри векторних зображень

Розмір файлу векторного зображення перш за все залежить від кількості окремих елементів, з яких утворено малюнок.

При формуванні зображення сучасні векторні редактори дозволяють використовувати замість суцільних кольорів градієнти, тобто плавні переходи від одного кольору до іншого. Розмивання окремих елементів, імітація тіні та інші засоби дозволяють отримати реалістичні зображення (див. малюнок1).

Векторний графічний редактор Inkscape

Редактор векторної графіки Inkscape розробляється під вільною ліцензією GNU GPL. Він є багатоплатформним, тобто одночасно виходять версії для різних операційних систем (Linux, Windows тощо). Піктограму програми показано на малюнку.



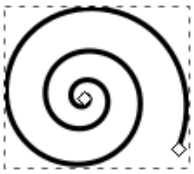
Після завантаження редактора відкривається його головне вікно (див. малюнок). У ньому крім рядків заголовка та меню бачимо робоче поле з чистим полотном для побудови малюнка та декілька панелей: панель команд, панель

[1] Векторне зображення з сайту http://luciano.kurumin.com.br/images_dicas/

інструментів, панель параметрів інструментів, палітра.

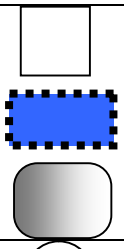
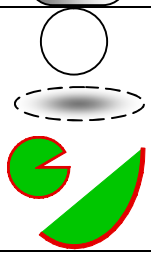
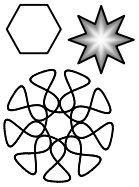
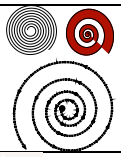
Інструменти

Більшість кнопок, розміщених на панелі інструментів, призначені для побудови окремих частин малюнка. Якщо клацнути кнопку інструмента, то на панелі параметрів інструментів з'являться значення відповідних параметрів. Ознайомимось детальніше з деякими інструментами та їхніми параметрами.



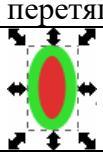
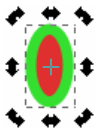
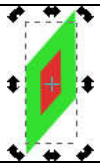
Щоб скористатись інструментами Прямокутник, Еліпс, Многокутник та Спіраль потрібно, клацнувши кнопку на панелі, перемістити курсор на робоче поле, натиснути ліву кнопку миші і, не відпускаючи, пересунути вказівник в інше місце. При цьому на екрані буде зображуватись форма майбутнього об'єкта. Якщо відпустити кнопку миші, об'єкт буде додано до малюнка. Форму новоствореного об'єкта змінюють, перетягуючи маркери у вигляді маленьких білих квадратів (див. малюнок). Дія маркерів у об'єктів різних типів різна.

Утримування натиснутими клавіш Ctrl, Shift або Alt дає додаткові можливості при зміні форми об'єктів. Їх дію описують підказки внизу екрана.

– інструмент Прямокутник. Якщо при його використанні утримувати натиснутою клавішу Ctrl, то сторони прямокутника будуть відноситись як цілі числа (наприклад 1:1, 2:1, 3:1). При відношенні 1:1 отримаємо квадрат. Перетягуючи верхній правий маркер отримуємо округлені кути. Два інші маркери – для зміни розмірів.	
– інструмент Еліпс. Якщо при його використанні утримувати натиснутою клавішу Ctrl, то довжини осей еліпса будуть відноситись як цілі числа (наприклад 1:1, 2:1, 3:1). При відношенні 1:1 отримаємо круг. Якщо правий маркер перетягнути вправо, то „виріжемо" сектор, вліво – отримаємо сегмент. Два інші маркери – для зміни розмірів.	
– інструмент Многокутник. Завдяки додатковим параметрам можна будувати правильні і неправильні, опуклі і зірчасті, з округленими кутами та інші многокутники. Опуклий многокутник має один маркер, яким можна змінювати його розмір і повертати. У зірчастого многокутника два маркери: один – для зміни внутрішнього радіусу і форми; другий – для зміни зовнішнього радіусу і повертання.	
– інструмент Спіраль. Отримана спіраль має на кінцях два маркери, обертаючи які за допомогою миші можна додавати або вилучати витки зовні та всередині.	

Для керування програмою користуються інструментом Вибір (Стрілка) . За його допомогою можна виконувати операції над об'єктами і групами обєктів:

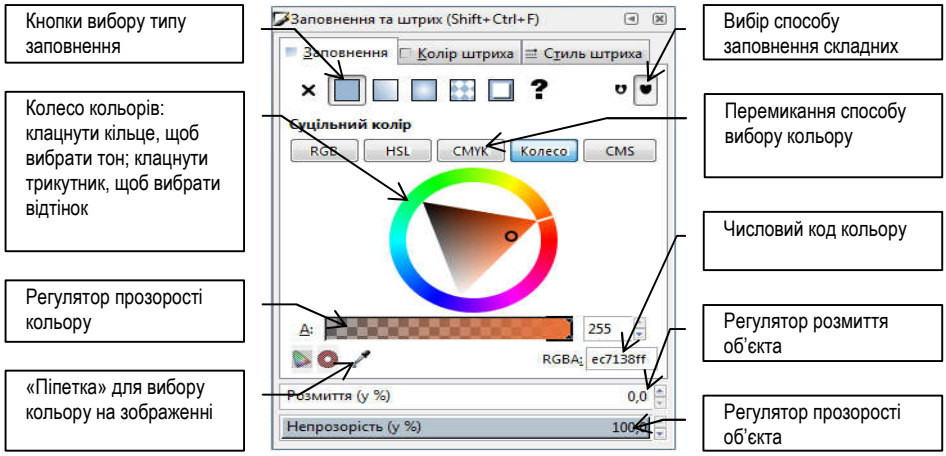
вибрати для подальших дій весь малюнок або його частину	1) Щоб вибрати окремий об'єкт – клацніть на ньому. Навколо об'єкта з'явиться штрихова прямокутна рамка, а біля неї – стрілки для зміни розмірів (див. нижче). 2) Щоб вибрати декілька об'єктів – клацніть їх по черзі, утримуючи натиснутою клавішу Shift. 3) Щоб вибрати декілька об'єктів, розміщених поряд – поставте стрілку лівіше і вище від групи; натисніть ліву кнопку миші і, не відпускаючи, перетягніть стрілку правіше і нижче від групи; відпустіть кнопку. В процесі роботи буде видно прямокутник, який охоплює об'єкти, що будуть вибрані.

перемістити об’єкти	Навести на об’єкт стрілку і перетягнути мишею. Щоб одночасно перетягнути кілька об’єктів, їх треба перед тим вибрати.
змінити розміри об’єктів	 Для зміни розмірів слід перетягувати відповідні стрілки. Кутовими стрілками змінюють одночасно довжину і ширину, а бічними – лише один з розмірів.
повернути об’єкти	 Якщо клацнути на вибраному об’єкті – стрілки змінять вигляд і з’явиться хрестик, що позначає центр обертання. При перетягуванні кутових стрілок об’єкт буде обертатись. Центр також можна перетягти в інше місце.
нахилити об’єкти	 Стрілками біля сторін прямокутника можна зсувати одні частини об’єкта відносно інших. Центр залишається нерухомим, як і при обертанні.

Швидко змінити положення об’єкта допоможуть кнопки на панелі параметрів інструмента ***Стрілка***:  ,  – повернути об’єкт на 90° за чи проти годинникової стрілки;  ,  – відобразити його відносно горизонтальної або вертикальної осі.

<u>Контури та заповнення фігур</u>

Щоб змінити кольори ліній та заповнення фігур потрібно на панелі команд натиснути кнопку  , яка відкриє вікно *Заповнення* та штрих (*див. малюнок*).



Розглянемо елементи керування на вкладці *Заповнення*. Кнопками у верхній частині вибираємо тип заповнення:

✕ – заповнення відсутнє, зображатимуться лише лінії;	
– суцільне заповнення вибраним кольором з урахуванням прозорості;	
– лінійний градієнт – плавний перехід від одного кольору до іншого або від зафарбованої до прозорої ділянки;	
– радіальний градієнт – плавний перехід кольорів від центра до країв фігури.	
– візерунок.	

На малюнку зображено вигляд вікна під час вибору кольору для суцільного ***заповнення*** з використанням *колеса кольорів*. В цьому режимі дуже просто підібрати бажаний колір: клацнувши в потрібне місце різнобарвного кільця обираємо тон, після чого, клацнувши в трикутник, обираємо відтінок кольору. Щоб заповнення фігури було напівпрозорим, треба клацнути у відповідне місце регулятора під кільцем. Якщо вікно не закриває малюнка, то всі зміни кольору відразу видно на екрані, що полегшує роботу.

Вкладка *Колір штриха* містить подібні елементи, але призначені для вибору ***кольору ліній*** об'єкта.

На вкладці *Стиль штриха* задають товщину ліній у вибраних одиницях (наприклад, у міліметрах) та інші параметри ліній.

Особливості побудови й опрацювання векторних зображень

Настав час випробувати редактор *Inkscape* в роботі. Створення комп'ютерних графічних зображень, так само, як і звичайне малювання, – процес творчий. Не кожен зможе з першої спроби створити шедевр. Тому починати треба з простих малюнків, які містять небагато окремих елементів. На таких тренувальних зображеннях можна добре випробувати інструменти, щоб потім впевнено користуватися ними.

Щоб вилучити невдалий фрагмент малюнка, треба вибрати його інструментом

Стрілка  *і натиснути клавішу Delete*

Це не обов'язково робити зразу:

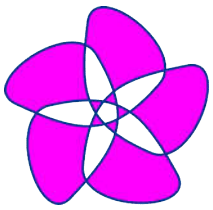
всі частини векторного зображення є окремими об'єктами і їх можна змінити або стерти в будь-який момент

Інструмент ***Лінза*** допоможе краще роздивитись дрібні деталі малюнка. Щоб скористатись ним досить клацнути кнопку  , а потім – потрібне місце на зображенні. Масштаб буде ***збільшено***. Керувати масштабом зручно також кнопками, що на панелі команд:  – показати вибрані об'єкти,  – показати весь малюнок,  – показати весь аркуш. Додаткові можливості масштабування, зокрема, кнопка для ***зменшення*** масштабу  доступні на панелі параметрів інструмента *Лінза*.

Зміни масштабу ніяк не впливають на малюнок, а змінюється лише його розмір на екрані.

Питання для самоперевірки

1. *Що таке комп'ютерна графіка?*
2. *У чому полягає растрове кодування зображень? Які його переваги і недоліки?*
3. *Які зображення обробляються у растровому форматі?*
4. *Поясніть суть векторного кодування зображень. Які його переваги і недоліки?*
5. *Для роботи з якими зображеннями призначений графічний редактор Inkscape?*
6. *Як змінити розмір фігури?*
7. *Як побудувати круг?*
8. *Як повернути прямокутник відносно одного з кутів?*
9. *Яким інструментом можна намалювати фігуру, зображену на малюнку?*
10. *Як відкрити вікно «Заповнення та штрих»?*
11. *Як користуватись колесом кольорів?*
12. *Що таке градієнт?*
13. *Як зробити заповнення фігури напівпрозорим?*
14. *Як змінити товщину лінії фігури?*
15. *Як можна змінювати масштаб перегляду зображення на екрані?*



Урок 2. Формати графічних файлів. Побудова ліній

Формати графічних файлів

Форматом файлу називають набір правил запису даних. Існує багато форматів для запису графічних зображень у файл, які відрізняються розміром файлу, способом розміщення даних у файлі, якістю зображення тощо. Є формати, які підтримують тільки растрову або векторну графіку. Деякі формати підтримують обидва види графіки. Розглянемо коротко найпоширеніші графічні формати.

Растровий формат BMP підтримується будь-якими Windows-сумісними програмами. Формат дає змогу використовувати повну палітру з 16 млн. кольорів. При цьому не використовується стискання даних, тому файли формату BMP мають великий обсяг.

Растровий формат PNG широко використовується в Інтернеті. Його палітра підтримує, залежно від типу, від 65536 до більше ніж 4 млрд. кольорів, а також прозорість. Використовує стискання без втрати якості.

Растровий формат JPG використовує стискання графічних даних за рахунок втрати якості зображення: менший файл – нижча якість. Це означає, що зберігши зображення у цьому форматі практично неможливо відтворити його початковий вигляд. Проте, завдяки малому розміру файлів, формат JPG набув поширення при роботі з фотографіями та в Інтернеті.

Растровий формат GIF підтримує палітру всього з 256 кольорів, зате дозволяє зберігати прозорість окремих ділянок зображення і анімацію.

Векторний формат SVG – масштабована векторна графіка –використовується багатьма програмами, зокрема, графічним редактором Inkscape.

Популярні програми для роботи з векторною графікою мають власні формати файлів: ***CDR*** – CorelDRAW, ***AI*** – Adobe Illustrator тощо.

Формати CGM, WMF, EPS підтримують як векторну, так і растрову графіку.

Робота з файлами в Inkscape

Ви вже вмієте зберігати результат своєї роботи в деяких програмах. Малюнок, виконаний в Inkscape, також можна зберегти у файл для подальшого використання.

Кнопка  , що на панелі команд, відкриває стандартне діалогове вікно збереження файлу. У ньому потрібно відкрити папку для збереження, зазначити ім'я файлу та його тип. Якщо робота з малюнком не закінчена, то слід вибрати тип файлу *Inkscape SVG*. При повторному натисканні файл зберігається без відкриття діалогового вікна.

 – *створити новий документ*. За цією командою відкривається нове вікно Inkscape, в якому можна працювати з іншим малюнком.

 – *відкрити* (завантажити з диска) існуючий документ. Команда викликає стандартний діалог вибору файлу, у правій частині якого є область перегляду вибраного у папці файлу.

 – *надрукувати документ*. Відкривається стандартне діалогове вікно друку документа, в якому слід вказати: на який з принтерів надіслати документ; скільки надрукувати примірників тощо.

Побудова зображення з графічних примітивів

Придивившись до малюнків у книжках, журналах, комп'ютерних іграх, помічаємо, що більшість з них складаються не лише з простих геометричних фігур, про які йшлося на попередньому уроці, а мають багато складних за формою криволінійних елементів.

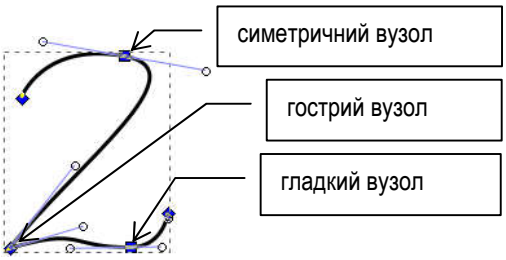
Розглянемо інструменти, якими малюють криві.

 – ***Крива***. Інструмент для малювання довільних кривих ліній і відрізків. Після вибору інструмента слід натиснути ліву кнопку миші і, не відпускаючи, рухом миші малювати лінію. Якщо початок і кінець кривої збігаються, вона стає замкненою. Щоб закінчити криву досить відпустити кнопку миші. Якщо ж клацнути мишею в початковій і кінцевій точках, то отримаємо прямолінійний відрізок.

 – ***Крива Безьє***. Інструмент для малювання ламаних ліній та ліній особливого виду – кривих Безьє. Якщо клацати мишкою без перетягування, то отримаємо ламану з прямолінійних відрізків. При перетягуванні з натиснутою лівою кнопкою будуть з'являтись криві лінії. Щоб закінчити малювання треба двічі клацнути або зробити лінію замкненою, клацнувши початкову точку.

Як вже було сказано, кожна з намальованих кривих є самостійним об'єктом. Їх можна пересувати за допомогою миші, змінювати колір тощо.

Зміна контурів



Розглянемо детальніше, як можна змінити форму кривої. Для цього призначений інструмент *Редагування вузлів* . Якщо вибрати цей інструмент і клацнути одну з кривих, на ньому будуть показані квадратиками *вузлові точки* (***вузли***). Перетягуючи вузли можна змінювати форму кривої.

Частину кривої між двома вузлами називають ***сегментом***. Сегменти бувають двох типів: прямолінійні і криволінійні.

Щоб вибрати вузол досить клацнути його. Утримуючи клавішу Shift можна вибрати декілька вузлів.

Щоб вибрати сегмент треба вибрати два вузли, між якими він розміщений.

Якщо вибрати вузол, що належить криволінійному сегменту, то біля нього з'являться один або два ***вуси***, кожен з круглим маркером на кінці. Вуси також з'являться у двох сусідніх вузлів (див. малюнок). Перетягуючи маркери можна довільно змінювати форму сегментів.

Якщо ж вуси у вибраного вузла не з'явились, то це означає, що обидва сусідні сегменти – прямолінійні. Крім того, маркери можуть бути суміщені з вузлом. Тоді їх називають втягнутими, а витягти їх можна мишею, утримуючи клавішу ***Shift***.

Якщо два сегменти плавно переходять один в інший, то вузол називають ***гладким****, а якщо ні –* ***гострим****. Гладкий вузол може бути* ***симетричним****.*

Відрізнити тип вузла можна за „поведінкою" його вусів (*див. малюнок*): у гострого – обидва маркери можна переміщувати незалежно; у гладкого – маркери завжди розміщені на одній прямій з вузлом; у симетричного – на одній прямій з вузлом і на однаковій відстані від нього.

Робота з вузлами

На панелі параметрів інструмента ***Редагування вузлів*** розміщені кнопки додаткових операцій з вузлами. Ознайомимося з ними детальніше.

– *додавання нових вузлів*. Потрібно вибрати один або більше сегментів і клацнути цю кнопку. Посередині кожного з них з'явиться новий вузол.

– *видалення вузлів*. Слід вибрати один або більше вузлів і клацнути цю кнопку або клавішу ***Delete***: вузли зникнуть, а контур залишиться нерозривним, якщо він був таким до цього. Форма контуру пр цьому може трохи змінитись.

З'єднання двох кінцевих вузлів. Для з'єднання треба вибрати два кінцеві вузли і клацнути одну з кнопок: – сегменти викривляться і сполучаться одним вузлом; – між вузлами з'явиться новий сегмент.

Якщо треба з'єднати два різні контури, їх попередньо об'єднують в один об'єкт.

Щоб ***об'єднати*** *два або більше контурів в один об'єкт, слід вибрати ці контури і вибрати команду меню* ***Контур ⇨ Об'єднати*** *(гарячі клавіші* ***Ctrl+K****).*

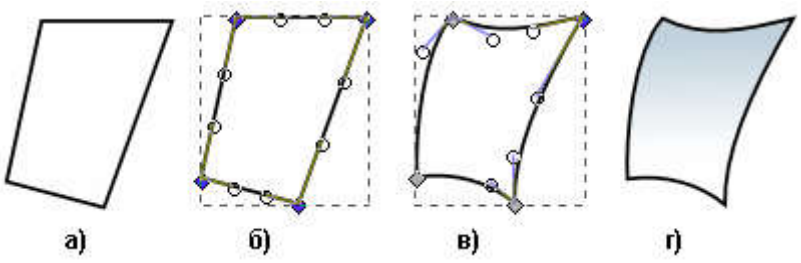
Розрив контуру можна виконати двома способами: вибрати сегмент і вилучити його кнопкою ; вибрати вузол і зробити розрив кнопкою . Якщо утвориться два контури, то вони будуть об'єднані в один об'єкт.

Щоб ***розділити*** *об'єкт на окремі контури, слід вибрати його і викликати команду меню* ***Контур ⇨ Розділити****. Після цього кожен контур стає окремим об'єктом.*

Щоб змінити тип вузла, треба його вибрати і клацнути одну з кнопок: – зробити вузол гострим; – зробити вузол гладким; – зробити вузол симетричним.

Щоб *змінити тип сегмента* потрібно його вибрати і викликати одну з команд: – зробити сегмент прямолінійним; – зробити сегмент криволінійним.

Розглянемо, наприклад, як можна отримати зображення вітрила для малюнка, наведеного на початку параграфа.



Інструментом *Крива Безьє* клацаємо по черзі у вершинах чотирикутника (мал. а). Першу вершину в кінці побудови треба клацнути ще раз, щоб контур був замкненим. Інструментом Редагування вузлів вибираємо чотирикутник, а потім, утримуючи ***Shift*** всі 4 його вузли. Клацнувши кнопку перетворюємо всі сегменти в криволінійні: з'являються круглі маркери вусів (мал. б). Перетягуємо маркери, щоб отримати бажану форму контуру (мал. в). Відкривши кнопкою вікно Заповнення та штрих підбираємо спочатку колір (блакитний), а потім вид заповнення (лінійний градієнт) (мал. г).

Питання для самоперевірки

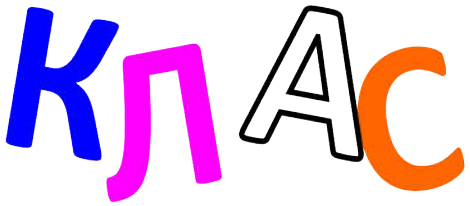
1. *Як отримують різні кольори на екрані монітора?*
2. *Як кодуються кольори у моделі RGB? CMYK? HSB?*
3. *Опишіть особливості растрових форматів файлів.*
4. *Які ви знаєте векторні формати файлів?*
5. *Яка особливість вирізняє формат GIF серед розглянутих растрових форматів?*
6. *Якими інструментами можна створити на екрані довільні контури?*
7. *Як зробити криву Безьє замкненою?*
8. *Як за допомогою інструмента **Крива Безьє** отримати ламану лінію?*
9. *Які операції виконують інструментом **Редагування вузлів** ?*
10. *Як змінити тип сегмента контуру з прямолінійного на криволінійний?*
11. *Які є типи вузлів контуру?*
12. *Як додати на контурі новий вузол?*
13. *Як з'єднати незамкнутий контур новим сегментом?*

Урок 3. Текстові об'єкти. Копіювання об'єктів

Додавання тексту до графічних зображень

Написи на малюнок додають за допомогою інструмента ***Текст*** . Вибравши його, потрібно клацнути в те місце, де має починатись напис: там з'явиться текстовий курсор. Користуючись клавіатурою слід виконати необхідний напис.

Не треба зважати на розмір літер, тому що розмір готового напису можна змінювати як завгодно, користуючись інструментом ***Стрілка***. Колір контурів та заповнення для літер змінюють так само, як і для інших об'єктів. Зокрема, виділивши окремий знак, можна змістити його вгору чи вниз, або повернути на певний кут відносно інших. Також для кожного знака окремо можна задати стиль заливки і контуру (*див. малюнок*).

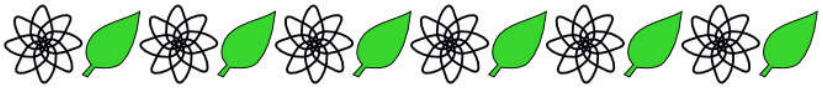


На панелі параметрів інструмента для введеного напису можна змінити гарнітуру, накреслення, вирівнювання абзаців тощо. Додаткові можливості для обробки напису надає меню *Текст*. Команда меню *Текст* ⇨ *Текст* і шрифт відкриває діалогове вікно для більш детальної роботи з написом.

Якщо додати до малюнка напис і криву, вибрати їх, а потім вибрати команду меню *Текст* ⇨ *Розмістити по контуру*, то текст буде розміщено вздовж кривої. При зміні форми кривої буде відповідно змінюватися форма тексту (*див. малюнок*).

Копіювання та клонування об'єктів

Досить часто на малюнках використовуються однакові елементи. Наприклад, орнамент (*див. малюнок*) може складатись з однакових зображень квіток та листків; емблеми для учасників шкільного конкурсу теж однакові, а на аркуші їх для друку розміщують декілька.



Для операцій з копіями об'єктів використовують групу кнопок  *на панелі команд*

Отримати копію об'єкта дуже просто: потрібно його вибрати і клацнути кнопку *Дублювання*  на панелі команд або команду меню *Зміни* ⇨ *Дублювати*. Копія розміщується на тому ж місці, де знаходиться оригінал і стає вибраною. Тепер її можна перетягти на потрібне місце. Якщо копій треба декілька, то досить кілька разів клацнути кнопку  .

Кожна копія, відразу після створення, стає самостійним об'єктом і ніяк не пов'язана з оригіналом. Але іноді буває корисно мати можливість швидко змінити відразу багато об'єктів. Наприклад, змоделювавши візерунок для вишивання хрестиком червоними, зеленими, і жовтими нитками, ви можете захотіти замінити колір зелених хрестиків на синій.

В такому разі слід створити по одному зображенню різнокольорових хрестиків, а далі робити не копії, а ***клони***.

Клон, це копія, яка зберігає зв'язок з оригінальним об'єктом. Колір, розмір, положення, товщина ліній та інші властивості клону змінюються разом з оригіналом

Для клонування слід вибрати об'єкт і викликати команду меню Зміни ⇨ Клонувати ⇨ Створити клон або клацнути кнопку  на панелі команд.

Щоб розірвати зв'язок між клоном та оригіналом потрібно вибрати його і клацнути кнопку  на панелі команд. Після цього він стає звичайним об'єктом.

Вирівнювання об'єктів

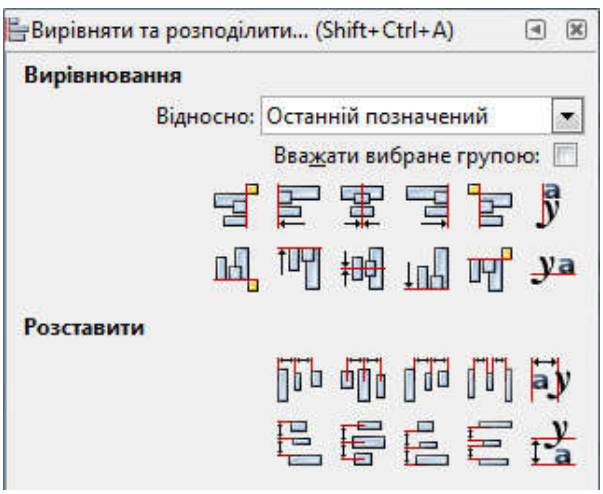
Працюючи над малюнком ви будете створювати багато об'єктів: прямокутників, кругів, ліній, складніших фігур. При цьому часто буде потрібно впорядковувати їх взаємне розміщення. Наприклад, щоб отримати простий орнамент, можна створити спочатку кілька його елементів (квітка і квадрат на малюнку). Потім скопіювати потрібну кількість разів і розсунути приблизно на свої місця (мал. 1).

А для того, щоб об'єкти вирівнялися, потрібно їх всі вибрати і скористатися командою меню *Об'єкт* ⇨ *Вирівняти*… або кнопкою  на панелі команд. З'явиться вікно *Вирівняти та розподілити* з відповідними командами.

Призначення команд зрозуміле з малюнків-піктограм, а детальніша інформація з'являється в спливних підказках.

Є такі команди вирівнювання:

- вирівняти ліві сторони;
- вирівняти праві сторони;
- вирівняти верх;
- вирівняти низ;
- центрувати на вертикальній осі;
- центрувати на горизонтальній осі.



Можливе також вирівнювання відносно певного об'єкта (*якоря*), який потрібно вибрати у полі *Відносно*, що у верхній частині вікна (*див. малюнок*):

- ліва сторона об'єктів до правої сторони якоря;

- права сторона об’єктів до лівої сторони якоря;
- нижня сторона об’єктів до верхньої сторони якоря;
- верхня сторона об’єктів до нижньої сторони якоря.

Якорем можуть бути: останній або перший з вибраних об’єктів, найбільший або найменший з вибраних об’єктів, сторінка, весь малюнок або прямокутник, що обмежує вибране. Об’єкт-якір залишається нерухомим, а всі інші зміщуються так, як сказано в команді.

Нижче у вікні *Вирівняти* розміщені команди для розстановки об’єктів (група *Розставити*). Вони дають можливість різними способами розподіляти об’єкти по горизонталі, по вертикалі або на аркуші. Можна, наприклад, розставити центри об’єктів на однаковій відстані по горизонталі або по вертикалі. Призначення цих кнопок добре зрозуміле зі спливних підказок.

Рівно розмістити об’єкти безпосередньо під час побудови допоможе сітка. Вмикають сітку командою меню *Перегляд → Сітка сторінки*. Після цього, під час побудови об’єктів, при клацанні і перетягуванні мишею відбуватиметься прилипання вказівника до найближчого вузла сітки. Тимчасово вимкнути прилипання, а також керувати прилипанням до інших елементів (вузлів, контурів, рамок об’єктів тощо) допоможуть кнопки Панелі керування прилипанням. Якщо її не видно, то слід вибрати команду меню *Перегляд → Показати/Сховати* і клацнути назву панелі.

Питання для самоконтролю:

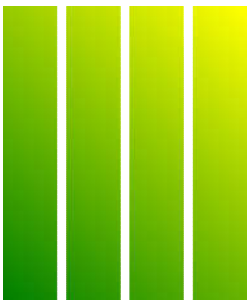
1. *Для чого призначений інструмент Текст ?*
2. *Як змінити колір ліній, якими обведені літери в тексті?*
3. *В яких випадках використовують копіювання об’єктів?*
4. *Що називають клоном? Чим клон відрізняється від інших об’єктів?*
5. *Яке призначення кнопки на панелі команд?*
6. *Опишіть засоби, доступні на панелі Вирівняти та розподілити.*
7. *Як розірвати зв’язок між клоном і оригіналом? До чого це призводить?*
8. *Для чого призначена сітка?*
9. *Як побудувати наведений на малюнку орнамент?*

Урок 4. Практична робота №1 «Створення простих векторних зображень»

Див. робочий зошит «Інформатика. 6 клас» / Бондаренко О.О., Ластовецький В.В., Пилипчук О.П., Шестопалов Є.А. – Шепетівка: «Аспект», 2017.

Урок 5. Складені векторні зображення

Об’єднання контурів. Операції над об’єктами



Іноді буває потрібно обробляти декілька об’єктів як одне ціле. Наприклад, на малюнку показано чотири прямокутники, які мають спільну градієнтну заливку. Такого ефекту можна досягти просто вибравши всі прямокутники і скориставшись інструментом ***Градієнт***. Але для того, щоб потім його можна було регулювати, потрібно спочатку об’єднати контури відповідною командою меню *Контур* ( ).

Надалі об’єднані контури обробляються як єдиний контур, тобто вони мають спільні колір ліній заливку.

Щоб ***розділити*** раніше об’єднані контури слід вибрати об’єкт, що їх містить, і застосувати команду меню *Контур* ⇨ *Розділити* ( ).

Подібно працює і операція групування об’єктів. Щоб утворити групу, треба вибрати всі об’єкти, що в неї увійдуть, і скористатись командою меню *Об’єкт* ⇨ *Згрупувати* або кнопкою панелі команд. Згруповані об’єкти можна пересувати і масштабувати як єдине ціле, хоча при цьому кожен з них має власні кольори та стилі заливки. Але при групуванні не

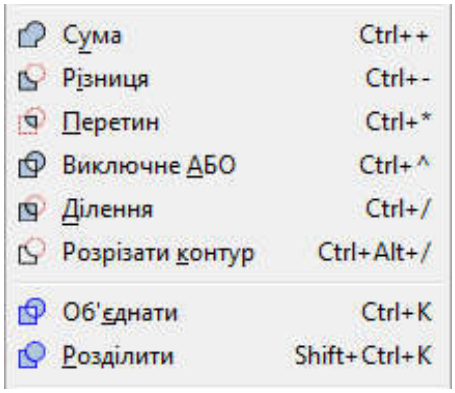
утворюється єдиний контур, тому для операцій над контурами потрібно увійти в групу, двічі клацнувши на ній. Після внесення змін слід вийти з групи, двічі клацнувши за її межами.

Якщо потрібно, об'єкти розгруповують відповідною командою меню *Об'єкт* або кнопкою .

Операції над контурами

Ви вже навчилися створювати різні об'єкти: прямокутники, кола, криві Безьє, текст, криві тощо. Кожен графічний об'єкт має контур, тобто лінію, яка його обмежує. І саме виконуючи різні операції над контурами об'єктів отримуємо додаткові можливості для створення малюнків.

На малюнку показано білий квадрат, а в ньому «вирізано» зірчастий отвір, через який видно інші об'єкти. Для того, щоб отримати такий об'єкт потрібно спочатку створити квадрат і зірку. Після цього зірку слід розмістити над прямокутником там, де має бути отвір. Потім треба вибрати, утримуючи натисненою клавішу ***Shift***, прямокутник і зірку, і вибрати команду меню *Контур* ⇨ *Різниця* (*див. малюнок*) – прямокутник з отвором готовий.



Ознайомимося з усіма операціями, які можна виконати над контурами. Піктограми (малюнки) в меню зображають результат застосування команди до прямокутника і круга, що накладаються. За ними можна швидко зорієнтуватись, яку з операцій вибрати.

Сума. Для цієї операції слід вибрати два або більше контурів. Контури в точках перетину з'єднуються і новий об'єкт включає всі ділянки, що належали хоча б одному з них.

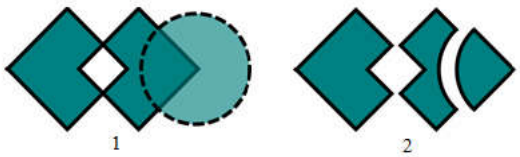
Різниця. Мають бути вибрані два контури. Внаслідок операції зникає верхній об'єкт, а також ті частини нижнього, які він закривав.

Перетин. Треба вибрати два або більше контурів. Після виконання операції залишиться тільки та ділянка, яка належала одночасно всім об'єктам. Якщо ж такої спільної ділянки немає, то всі вибрані контури зникнуть.

Виключне Або. Ця операція застосовується до двох вибраних контурів, що перекриваються. Після її виконання зникає та частина, що була спільною для обох контурів.

Ділення. Нижній з двох вибраних об'єктів розрізається лініями верхнього та власними точками самоперетинів на окремі об'єкти, кожен з яких має замкнутий контур. Верхній об'єкт зникає.

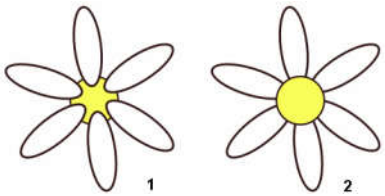
Розгляньте малюнок. Спочатку до двох квадратів, що накладаються, була застосована операція *Виключне АБО* . Внаслідок цього з'явився один об'єкт з квадратним отвором (мал. 1). Потім намалювали круг і, вибравши обидва об'єкти, використали операцію *Ділення* . Круг зник, а основний об'єкт виявився поділеним на три частини (мал. 2): одна – яку закривав круг, а інші дві утворилися з квадрата точками перетину його контуру.



Розрізати контур. Ця операція застосовується до двох вибраних об'єктів. В результаті контур нижнього об'єкта виявляється розрізаним на частини в точках перетину з контуром верхнього та точках самоперетину. У всіх отриманих об'єктів відсутня заливка і окремі частини можуть бути незамкненими.

Упорядкування перекриття об'єктів

Об'єкти, з яких складається малюнок можуть закривати один одного. При цьому, той, що намальований пізніше, розміщується найвище, тобто закриває всі попередні. Працюючи



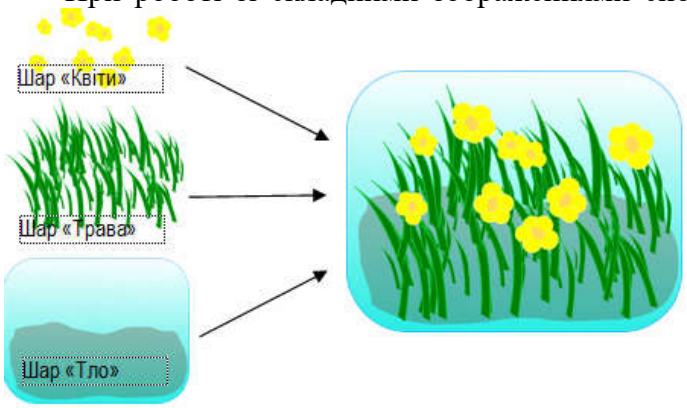
над складним малюнком, не завжди вдається передбачити послідовність появи об'єктів, і, як наслідок, частини малюнка, які мають бути видимі повністю, виявляються закритими іншими елементами (див. малюнок 1).

Змінити порядок об'єктів можна так: вибрати інструмент ***Стрілка***; вибрати об'єкти, які треба перемістити; на панелі параметрів інструмента вибрати потрібну команду: – підняти вибране на один рівень; опустити вибране на один рівень; – опустити вибране на задній план; – підняти вибране на передній план.

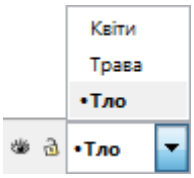
Наприклад, щоб виправити малюнок 1, треба вибрати круг – середину квітки – і подати команду (підняти на передній план).

<u>Багатошарові зображення, розміщення об'єктів у шарах</u>

При роботі зі складними зображеннями споріднені об'єкти, з яких вони складаються, зручно розміщувати в окремих шарах. Це дозволяє вимикати видимість частини зображення, блокувати об'єкти шару від випадкових змін тощо.



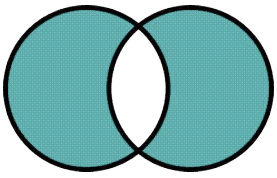
В меню Шар зібрані команди для: додавання і перейменування, показу і приховування, блокування і розблокування, зміни взаємного розміщення шарів, а також переміщення вибраних об'єктів з одного шару на інший.



У рядку стану розміщено список шарів та кнопки для керування видимістю і блокуванням поточного шару (*див. малюнок*).

Питання для самоконтролю:

1. *Для чого об'єднують контури?*
2. *Як розділити складний контур на окремі частини?*
3. *Як вирівняти вибрані об'єкти за верхнім краєм?*
4. *Як працюють команди вирівнювання за об'єктом-якорем?*
5. *Вибрано 10 прямокутників різних розмірів. Як вирівняти правий край великих за лівим краєм найменшого?*
6. *Як встановити між вибраними об'єктами однакові проміжки по горизонталі?*
7. *Яким чином у кругові можна зробити трикутний отвір?*
8. *Як отримати доступ до операцій над контурами?*
9. *В чому різниця між операціями Ділення та Розрізати контур?*
10. *Як побудувати зображення, наведене на малюнку?*
11. *Чим відрізняються операції Групування об'єктів та Об'єднання контурів?*



Урок 6. Практична робота №2 «Створення складених векторних зображень»

Див. робочий зошит «Інформатика. 6 клас» / Бондаренко О.О., Ластовецький В.В., Пилипчук О.П., Шестопалов Є.А. – Шепетівка: «Аспект», 2017.

2. Комп’ютерні презентації

Урок 7. Програмне забезпечення для створення й відтворення комп’ютерних презентацій

??????? (можна взяти з нашого підручника для 9-го класу)

Урок 8. Етапи створення презентації та вимоги до її оформлення

Як і будь-яка інша робота, за умови чіткого планування, презентація може бути розроблена порівняно швидко. Процес розробки презентації включає такі етапи:

1. ***Визначення теми презентації***. Чітко сформульована тема дозволить уникнути пошуку зайвих матеріалів, а отже прискорить розробку презентації.

2. ***Збирання матеріалу*** з теми (текст, малюнки, анімація, відео, звук) та розподіл його за слайдами (структурування).

Подальша робота проводиться у середовищі програми для створення презентацій.

3. ***Вибір оформлення слайдів***. Залежно від мети розробки, можуть бути використані зразки оформлення з колекції Power Point або розроблене власне оформлення.

4. ***Розміщення на слайдах матеріалів***. При цьому, залежно від виду та обсягу матеріалів, для кожного слайда вибирають готову розмітку або налаштовують власну.

5. ***Додавання анімаційних ефектів і/або музичного супроводу***.

6. ***Перегляд презентації***. При першому перегляді презентації можуть бути виявлені недоліки: невдало підібране оформлення; пропуск слайдів; перевантаженість окремих слайдів текстом, анімаційними ефектами тощо.

7. ***Збереження презентації***. Завершену презентацію зберігають у форматі **Демонстрація PowerPoint** (*.ppts).

Такі файли відкриваються зразу в режимі перегляду. Якщо ж планується продовження роботи над презентацією, то слід вибрати формат файлу **Презентація PowerPoint** (*.pptx).

Для того, щоб презентацію можна було переглянути навіть на комп’ютері, на якому не встановлено **PowerPoint**, слід, натиснувши кнопку Office, в розділі **Опублікувати** вибрати команду **Підготувати для компакт-диска**. В діалоговому вікні, що з’явиться, треба натиснути кнопку **Копіювати в папку**. Після вибору папки і підтвердження дій у зазначеній папці з’являться файл, необхідні для показу презентації без програми **PowerPoint**.

Залежно від призначення та обсягу презентації кожен з етапів може вимагати більше або менше часу і зусиль. Якщо передбачається презентація результатів розробки проекту, то робота може бути розподілена між учасниками проекту і тоді деякі етапи відбуватимуться одночасно.

Вимоги до оформлення презентації

Для того, щоб презентація добре сприймалася і допомагала реалізувати задум доповідача, потрібно дотримуватися певних вимог.

Презентація не є конспектом виступу, тому текстове наповнення повинне бути лаконічним. Слід пам’ятати, що, можливо, презентація демонструватиметься на екрані у залі великій кількості людей, і увага при цьому повинна бути сконцентрована на доповідачі. Великі текстові фрагменти відвертають увагу слухачів, а дрібний шрифт вимагає при читанні додаткових зусиль.

Якщо у виступі оголошуються певні ряди числових даних, то на слайді презентації слід продемонструвати діаграму, яка їх ілюструє.

Для всієї презентації здебільшого використовують однакове оформлення слайдів. Проте, якщо потрібно привернути увагу до окремих ідей, проблем, то для певних слайдів можна застосувати особливе оформлення (інший колір, графічні елементи, анімацію).

Оформлення слайдів не повинне заважати сприйняттю доповіді і змісту презентації.

Це зауваження стосується й анімаційних ефектів. Потреба у них залежить від призначення презентації: якщо вона має розважальний характер, то музичне і анімаційне оформлення буде доречним, проте для ділового виступу варто обмежитись невеликою кількістю анімаційних ефектів, які підкреслюють найважливіші думки, факти, і, здебільшого, зовсім відмовитись від звуків і музики.

Якщо планується опублікування презентації в Інтернеті, то слід дотримуватися авторських прав на використані зображення, звукові та відеоматеріали.

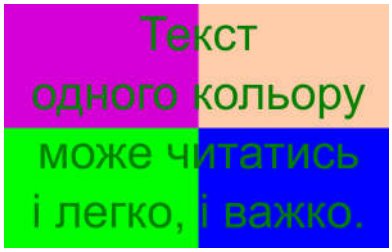
Структура презентації

Якщо на етапі збирання матеріалу він був чітко розподілений за слайдами, то швидко підготувати основу майбутньої презентації можна в *режимі структури* (рис. ??). Замість додавання слайдів вручну, достатньо набирати з клавіатури заголовки слайдів і натискати **Enter**: одночасно з цим будуть створюватись слайди. В режимі структури кнопки керування відступом діють подібно до багаторівневих списків:  – понижує рівень заголовка, перетворюючи його на пункт списку на попередньому слайді;  – перетворює пункт списку на заголовок окремого слайду.

Елементи дизайну презентацій

Робота над дизайном презентації передбачає налаштування таких елементів: *параметри сторінки*, *колірна гама*, *шрифти*, *ефекти для фігур*, *фон*. Засоби, які дозволяють це зробити, зібрані на вкладці **Дизайн**.

Залежно від мети, презентація може демонструватись на екрані або друкуватись на папері. Тому є можливість у вікні **Параметри сторінки** вибрати зі списку як один з типових розмірів екрана, так і один зі стандартних форматів паперу.



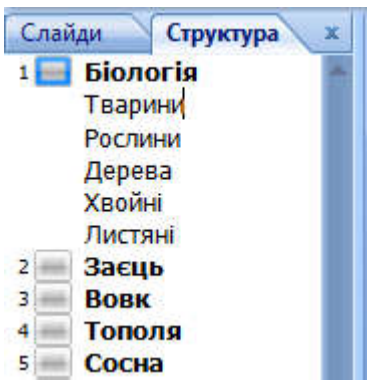
Важливою складовою дизайну слайда є *колірна гама*. Кольорів у презентації не слід використовувати занадто багато, а їх гармонійне поєднання, за умови вдалого вибору кольорів, сприяє кращому сприйманню матеріалу доповіді. Особливої уваги вимагає тло, на якому розміщено текстові блоки: текст легше читати, якщо кольори тексту і тла відрізняються за тоном і яскравістю (рис. ??).

Підбираючи шрифт потрібно мати на увазі, що не всі шрифти однаково добре читаються. Тому у презентаціях, призначених для виступу слід уникати використання декоративних шрифтів зі складною формою літер (рис.??). Проте, наприклад, у презентації-фотоальбомі такі шрифти можуть бути доречними.

Оцінювання презентацій

Таким чином, щоб оцінити якість презентації, слід звернути увагу на:

- відповідність темі;
- повноту охоплення теми;
- доцільність макетування слайдів та форматування вмісту;
- доречність дизайну та використання анімації, звукового оформлення;
- зручність користування (у випадку застосування елементів керування переглядом; див. далі).



Питання для самоперевірки

1. *В чому відмінність презентації від текстового документа?*
2. *Об'єкти яких типів може містити презентація?*
3. *З яких етапів складається процес розробки презентації?*
4. *Які недоліки можна виявити при перегляді презентації?*
5. *В чому різниця між форматами файлів *.ppts і *.pptx?*

6. Як підготувати презентацію до перегляду на іншому комп'ютері?
7. Чому текстове наповнення презентації має бути лаконічним?
8. Для чого використовують діаграми?
9. З яких міркувань слід добирати оформлення слайдів?
10. В яких випадках слід уникати в презентації музичних фрагментів?
11. На що слід зважати, використовуючи зображення, отримані з Інтернету?
12. Як швидко створити слайди, якщо відомі їхні заголовки?

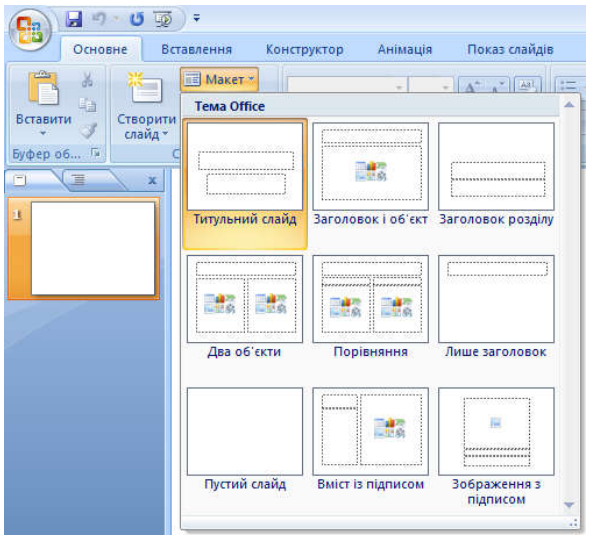
Урок 9. Макети слайдів. Діаграми

На слайдах презентації можуть використовуватись різноманітні матеріали, тому не завжди вдається швидко підібрати вдалий варіант розміщення. Розв'язати цю проблему допоможе використання готових *макетів слайдів*, вбудованих у PowerPoint.

Макети слайдів

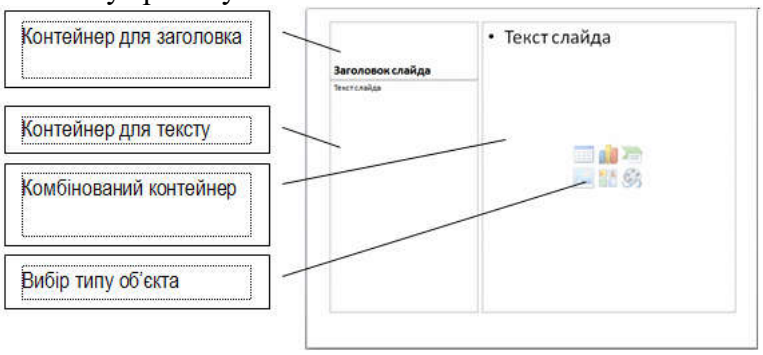
Щоб вибрати один з готових макетів слайдів, потрібно:

- виділити один або декілька слайдів;
- на вкладці Основне в групі Слайди розкрити список Макет (рис. ??);
- клацнути один зі зразків макетів.



На всіх виділених слайдах з'являться контейнери у вигляді штрихових рамок для додання матеріалів різних типів: заголовка, тексту, графічних та інших об'єктів (рис. ??). Кожен з типів контейнерів має налаштовані параметри форматування тексту, що дозволяє уникнути невдалих варіантів оформлення слайдів. Контейнер можна вилучити, клацнувши на штриховій рамці, а потім натиснувши кнопку **Delete**.

Контейнер *заголовка* вилучати небажано, оскільки пізніше, при налаштуванні гіперпосилань, цей заголовок буде присутній у списку слайдів, що значно полегшує роботу.



Комбінований контейнер дозволяє розміщувати дані різних типів. Для введення тексту, достатньо клацнути на вільному місці контейнера і розпочати набирати текст. Піктограми вибору типу об'єкта, розташовані в центрі контейнера, дозволяють додати об'єкти інших типів (див. таблицю).

Піктограма	Тип об'єкта і дія
	*Таблиця*. Відкривається вікно для введення числа рядків і стовпчиків.
	*Діаграма*. Відкривається вікно Вставка діаграми
	*Рисунок Smartart*. Додає графічний об'єкт для наочного подання даних

	*Малюнок з файла.* Відкривається стандартне вікно вибору файла.
	*Кліп з колекції.* Відкривається область завдань **Кліпи** для вибору кліпу з колекції.
	*Відео з файла*. Відкривається стандартне вікно вибору файла.

Якщо вставлений об’єкт видалити, то знову з’явиться контейнер і можна буде додати інший об’єкт.

Стильове оформлення слайдів презентації

Вивчаючи текстовий процесор ви дізналися, що стилі допомагають автоматизувати значну частину роботи. Є подібні засоби і в програмі **PowerPoint**.

Теми оформлення слайдів

На вкладці **Дизайн** в групі **Теми** доступна колекція готових наборів властивостей об’єктів презентації.

Піктограми в колекції ілюструють фон слайда, колірну гаму та стиль тексту кожної з тем оформлення. Достатньо клацнути одну з піктограм, щоб всі слайди презентації набули відповідного стилю оформлення. Якщо ж потрібно змінити тему оформлення тільки вибраних слайдів, то слід відкрити на піктограмі теми контекстне меню і вибрати команду **Застосувати до виділених слайдів** (рис. ??).

Слайди, яким призначена особлива тема оформлення, надалі не змінюватимуть вигляду при застосуванні теми оформлення до всіх слайдів клацанням на піктограмі.

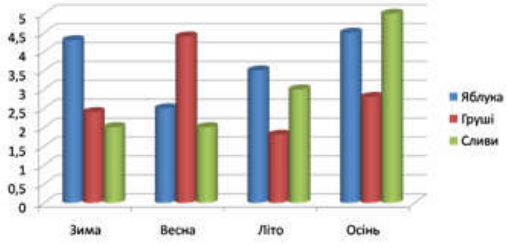
Використання діаграм у презентаціях

На слайд презентації можна додати:

- діаграми призначені для унаочнення рядів числових даних, подібні до тих, які ви створювали в електронних таблицях;
- об’єкти **SmartArt**, що полегшують сприйняття іншої структурованої інформації.

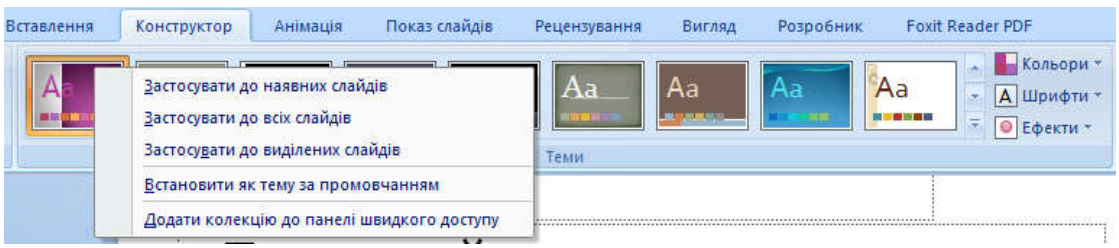
	A	B	C	D
1		Яблука	Груші	Сливи
2	Зима	4,3	2,4	2
3	Весна	2,5	4,4	2
4	Літо	3,5	1,8	3
5	Осінь	4,5	2,8	5

а б



Діаграми

Для додавання на поточний слайд діаграми потрібно перейти на вкладку **Вставлення** і в групі **Зображення** вибрати команду **Діаграма** або в комбінованому контейнері клацнути значок  – відкриється вікно **Вставлення діаграми**, в якому слід вибрати тип діаграми. Після підтвердження вибору кнопкою **ОК** відкриється вікно **Excel** з таблицею даних (рис. ??, а), а безпосередньо на слайді або в контейнері з’явиться побудована на цих даних діаграма

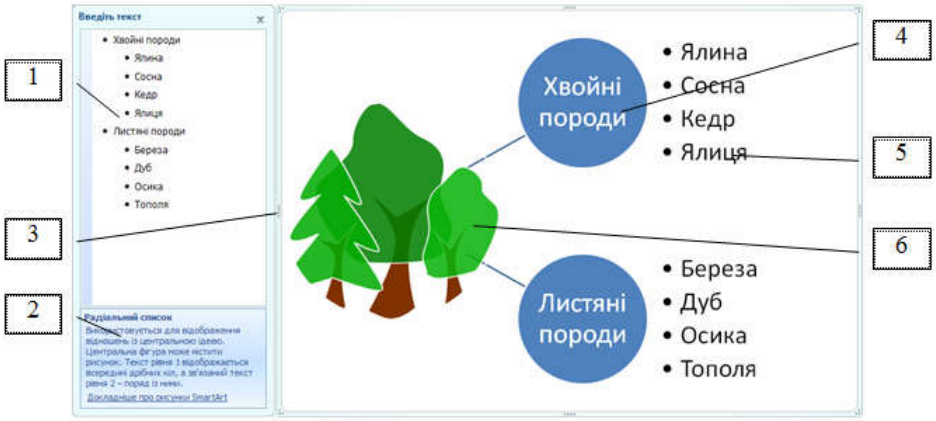


(рис. ??, б). Спочатку таблиця містить фіктивні дані, які потрібно замінити реальними. При редагуванні даних діаграма зразу ж змінює вигляд.

Об'єкти SmartArt

Починаючи з версії 2007 у програмах **PowerPoint**, **Word** і **Excel** з'явилась можливість створювати об'єкти **SmartArt** – діаграми, які дозволяють унаочнити різноманітну структуровану інформацію (наприклад, подану у вигляді багаторівневого списку).

Щоб додати на слайд об'єкт **SmartArt** потрібно на вкладці **Вставлення** в групі **Зображення** вибрати команду **SmartArt** або клацнути значок  в контейнері – відкриється вікно **Вибір рисунка SmartArt**, в якому слід вибрати тип і вид об'єкта. Після підтвердження вибору кнопкою **ОК,** на слайді з'явиться шаблон об'єкта з полями для введення тексту або вставки малюнків (рис. ??).



1 – текстова форма	4 – текстове поле 1-го рівня
2 – опис шаблона	5 – текстове поле 2-го рівня
3 – перемикач видимості текстової форми	6 – графічне поле

Якщо шаблон має складну структуру, то зручніше вводити текст у вікні, яке вмикається перемикачем 3. При цьому на вкладці **Конструктор** в групі **Створити графіку** стають доступними команди для підвищення і пониження рівня елементів списку, віддзеркалення шаблона в цілому тощо.

В групі **Стилі SmartArt** вибирають бажаний стиль оформлення об'єкта і його колірну гаму.

В групі **Макети** можна вибрати зовсім інший вид об'єкта. При цьому, залежно від виду, частина даних списку може бути відсутня на діаграмі. Проте зі списку вони не зникають, а лише позначаються хрестиками. При наступній зміні виду об'єкта ці дані можуть знову з'явитись у полях відповідного рівня.

Питання для самоперевірки

1. *Що таке макет слайда? (додати визначення)*
2. *Які об'єкти може містити макет слайда?*
3. *Як використати один із готових макетів слайдів?*
4. *Чому небажано вилучати зі слайда контейнер заголовка?*
5. *Що таке тема оформлення слайда? (додати визначення)*
6. *Як змінити тему оформлення частини слайдів?*
7. *При спробі змінити тему оформлення презентації деякі слайди не змінили свого вигляду. Що може бути причиною цього?*
8. *В чому різниця між діаграмами і об'єктами SmartArt?*
9. *Опишіть порядок додавання на слайд діаграми.*
10. *Як додати на слайд об'єкт SmartArt?*

Урок 10. Мультимедійний вміст у презентаціях. Анімаційні ефекти

Монотонна мова і споглядання статичних зображень і тексту швидко викликають втому слухачів. У багатьох випадках виправити ситуацію можна, додавши анімаційні ефекти і мультимедійний вміст.

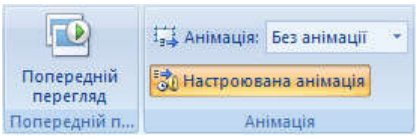
Елементи анімації

Є два напрямки використання анімації у презентаціях: анімація об'єктів і анімація зміни слайдів.

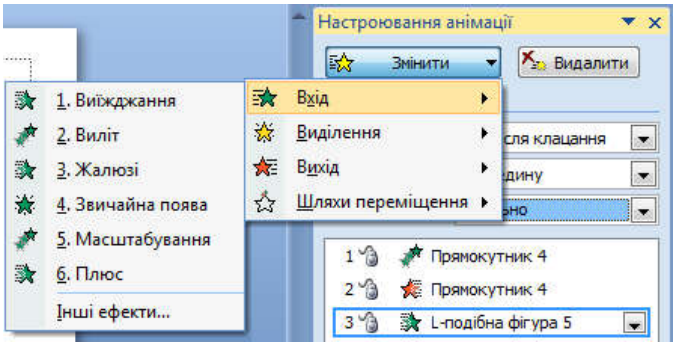
Анімація об'єктів

Анімаційні ефекти, які можна додати до об'єкта поділяються на три групи: **Вхід**, **Виділення**, **Вихід** і **Шляхи переміщення**.

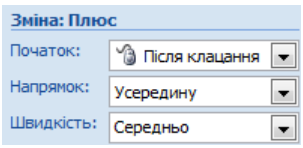
До кожного об'єкта можна додати декілька ефектів з кожної групи, програмуючи цим самим досить складну його поведінку.

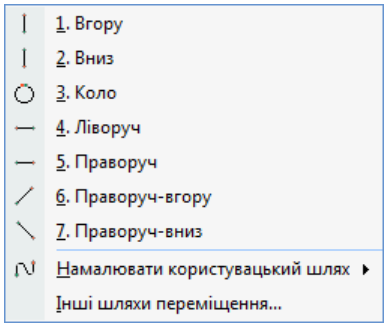


Щоб додати до розміщеного на слайді об'єкта анімаційний ефект з груп **Вхід**, **Виділення** або **Вихід**, потрібно його виділити, а потім виконати такі дії:



- на вкладці **Анімація** в групі **Анімація** вибрати команду **Настроювана анімація** (рис. ??) – з'явиться область завдань **Настроювання анімації**;
- в області завдань (рис. ??) розкрити список **Додати ефект**, потім один зі списків, що відповідають типам ефектів (Вхід, Виділення тощо) і вибрати потрібний ефект;
- за допомогою списків в області завдань (рис. ??) налаштувати параметри ефекту, а саме момент його початку (після клацання мишею, разом з попереднім у списку ефектом чи після попереднього), напрямок і швидкість;





Більш гнучко налаштувати рух об'єкта по слайду можна, задавши *шлях переміщення* по слайду. Для цього зі списку вибирають групу **Шляхи переміщення**, а зі списку, що розкриється, – один з готових шляхів (рис. ??) або команду **Намалювати користувацький шлях**. Готові шляхи (Вгору, Коло, Ліворуч тощо) можна додатково налаштувати перетягуючи круглі маркери, що з'являються, якщо виділити шлях на слайді.

ПРИКЛАД. До об'єкта **Хмара** спочатку було додано шлях **Праворуч**, а потім напрям було трохи змінено (рис. ??).

Також шлях можна довільно переміщувати по слайду. Зелений трикутний маркер позначає початок шляху, а червоний – кінець. Якщо початковий і кінцевий маркер розмістити за межами слайда, то об'єкт при перегляді з'явиться з-за краю екрана, переміститься шляхом і зникне з екрана.

Якщо шлях перемістити від об'єкта, то перед виконанням анімаційного ефекту об'єкт, де б він не був, «перескочить» на початок шляху.

У підменю **Намалювати користувацький шлях** можна вибрати варіант малювання шляху для переміщення: **Лінія**, **Крива**, **Полілінія**, **Мальована крива**. Порядок побудови шляху в цих випадках не відрізняється від користування однойменними інструментами для вставлення фігур.

Кожен з доданих до об’єкта ефектів з’являється у списку в області завдань, а також позначається на слайді номером у квадратику ліворуч від об’єкта.

ПРИКЛАД. На рис. ?? показано, що до об’єкта **Усміхнене обличчя** додано 4 ефекти.

Для змінення порядку виконання анімаційних ефектів достатньо перетягти їх на потрібні місця у списку в області завдань.

Як у процесі роботи, так і по її завершенні результат налаштування анімації можна переглянути на зображенні слайда, натиснувши кнопку **Перегляд** внизу області завдань або запустивши повноекранний перегляд з поточного слайда.

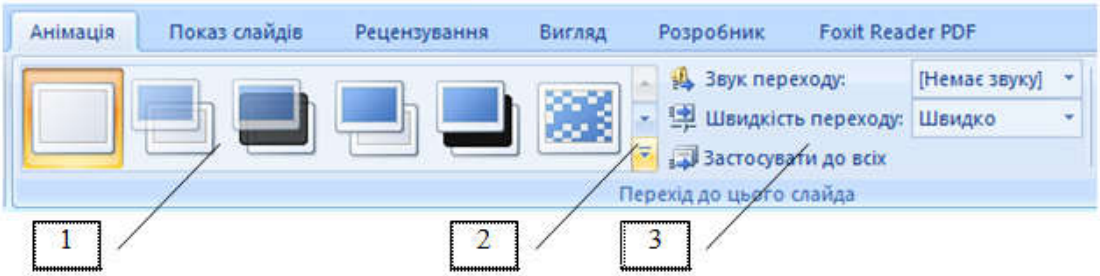
Щоб детальніше налаштувати параметри анімаційного ефекту, потрібно викликати контекстне меню на його рядку в списку і вибрати команду **Параметри ефектів**. З’явиться діалогове вікно, в якому можна:

- змінити часові параметри вибраного ефекту;
- додати до нього звуковий супровід;
- увімкнути затінення або зникнення об’єкта після завершення анімації;
- налаштувати запуск ефекту клацанням на іншому об’єкті тощо.

Анімація зміни слайдів

Анімаційний ефект, доданий до слайда, визначає, як він буде з’являтись на екрані під час перегляду презентації.

Засоби для налаштування анімації зміни слайдів знаходяться на вкладці **Анімація** в групі **Перехід до цього слайда** (рис. ??). Щоб додати до вибраних слайдів анімаційний

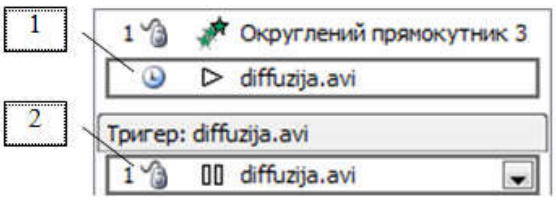


ефект, потрібно клацнути його піктограму в полі 1. Кнопка 2 розкриває вікно, в якому доступні значно більше різних ефектів. Після вибору ефекту слід у відповідних списках 3 налаштувати його додаткові параметри: звук переходу і швидкість. За допомогою відповідної кнопки можна застосувати цей ефект до всіх слайдів презентації.

Додавання відеокліпів

Для додання на слайд презентації відеокліпа потрібно:

- на вкладці **Вставлення** в групі **Медіакліпи** вибрати **Фільм** (рис. ??) або в контейнері клацнути піктограму  ;
- в діалоговому вікні вибрати відеофайл і натиснути ОК;
- у наступному діалоговому вікні зазначити, в який спосіб починатиметься відтворення відео: автоматично чи після клацання мишею.

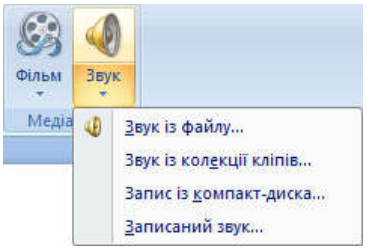


Після цього вибраний відеокліп з’явиться на слайді. Одночасно в списку ефектів анімації з’являються записи (рис. ??), що відповідають запуску кліпа (1) і керуванню переглядом клацанням миші (2). Запис для запуску кліпа можна переміщувати так само, як записи анімаційних ефектів, змінюючи таким чином момент початку відтворення відео. Якщо

видалити тригер, то при клацанні на вікні відеоролика під час відтворення, відбуватиметься перезапуск перегляду.

Звуковий супровід презентації

До презентації можна також додати загальне звукове тло (наприклад, мелодію, яка звучить постійно або мовний супровід).



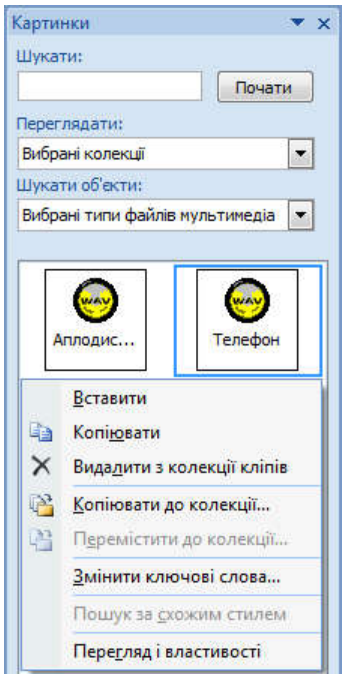
Меню вибору способу додання на слайд звукового файлу з'являється при натисканні нижньої кнопки **Звук** в групі **Медіакліпи** на вкладці **Вставлення** (рис. ??).

Звук із файлу

Щоб додати звук із файла, потрібно:

- вибрати відповідну команду;
- у стандартному вікні вибору файла, що з'явиться, зазначити, який файл додається і підтвердити вибір;
- у наступному діалоговому вікні вибрати, коли починатиметься відтворення звуку: автоматично чи після клацання.

Звук із колекції кліпів



При виборі цієї команди з'являється область завдань **Картинки** (рис. ??), в якій слід знайти бажаний звуковий фрагмент. Для прискорення пошуку можна:

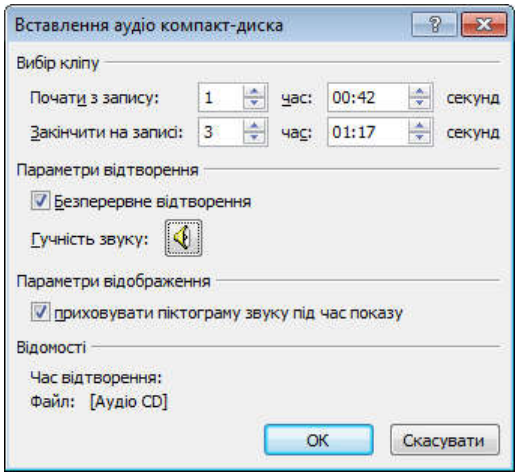
- у поле **Шукати** ввести ключове слово (наприклад, «телефон», «свист») ;
- у списку **Переглядати** вибрати, в яких колекціях здійснюється пошук;
- у списку **Шукати** об'єкти зазначити, об'єкти яких типів потрібно шукати (картинки, фотографії, фільми, звуки).

Після натискання кнопки **Почати**, якщо пошук був успішним, з'являться піктограми знайдених звукових фрагментів. Далі слід клацнути на одному з них і в діалоговому вікні вибрати, коли починатиметься звук. Якщо ж клацнути праворуч від нього, то з'явиться меню для детальнішого керування даним фрагментом.

Запис із компакт-диска

При виборі цієї команди з'явиться діалогове вікно **Вставлення аудіо компакт-диска** (рис. ??) для налаштування відтворення. У ньому можна:

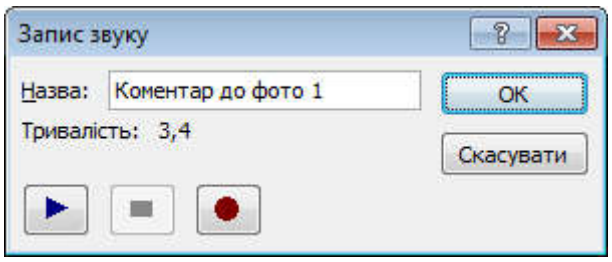
- задати номер запису та момент час у ньому для початку і закінчення відтворення диска;
- увімкнути безперервне відтворення;
- відрегулювати гучність звуку;
- приховати піктограму звуку при показі презентації.



Записаний звук

Вибір цієї команди дозволяє записати за допомогою мікрофона мовний супровід презентації. У діалоговому вікні, що з'явиться (рис. ??) слід ввести назву звукового фрагменту, а потім клацнути кнопку  для початку записування звуку. Кнопка  призупиняє запис, після чого його можна продовжити, клацнувши знову кнопку запису. Кнопка  призначена для прослуховування записаного звуку.

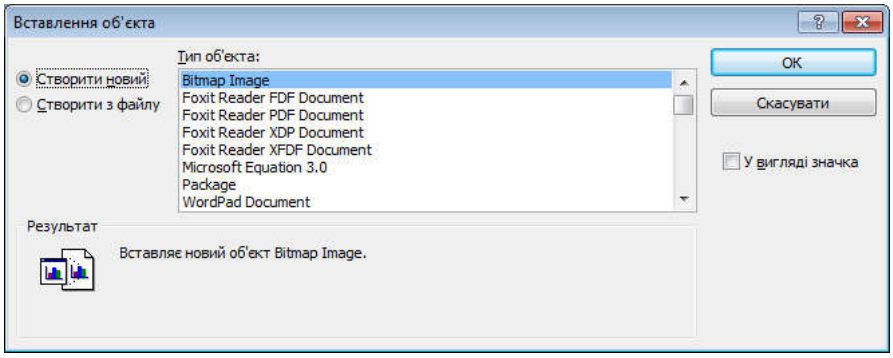
В усіх перелічених випадках, як наслідок, на слайді з'явиться піктограма, що зображає гучномовець або компакт-диск.



Вбудовування об'єктів

Завдяки технології OLE (англ. Objects Linking and Embedding – і вбудовування об'єктів) є можливість додати на слайд об'єкт, розроблений в інших програмах. Ви вже робили це, вставляючи малюнки, відеокліпи тощо.

При вставлянні об'єкта, записаного в іншому файлі, є можливість *зберегти* зв'язок з цим файлом. Якщо після цього змінити початковий файл, то відповідно зміниться й об'єкт на слайді.



Для того, щоб вбудувати об'єкт, потрібно на вкладці **Вставлення** в групі **Текст** вибрати команду **Об'єкт** – відкриється вікно **Вставлення об'єкта** (рис. ??).

Щоб створити об'єкт, не зберігаючи його до файла, слід:

- перемикачем ліворуч вибрати **Створити новий**;
- вибрати зі списку тип об'єкта (на малюнку – Bitmap Image);
- натиснути ОК – запуститься програма, якою розробляють об'єкти вибраного типу (наприклад, графічний редактор);
- розробити об'єкт у середовищі цієї програми;
- закрити програму – об'єкт з'явиться на слайді.

При подвійному клацанні на об'єкті він буде відкриватись цією ж програмою для редагування. За такого варіанту об'єкт вбудовується у файл презентації, збільшуючи його обсяг.

Якщо ж перемикачем вибрати варіант Створити з файлу (рис. ??), то клацнувши кнопку Огляд, потрібно буде вибрати файл, в якому зберігається об'єкт. Щоб при зміненні об'єкта у файлі його вигляд оновлювався й у презентації, потрібно перед натисканням ОК поставити прапорець Зв'язати. Завдяки цьому при відкриванні презентації оновлюватимуться зв'язки з об'єктами, вбудованими з файлів. Якщо ж об'єкт змінився під час розробки презентації, то для його оновлення слід викликати його контекстне меню і вибрати команду Оновити зв'язок.

При перенесенні презентації на інший комп'ютер або при видаленні зв'язаного об'єкта зв'язок з ним втрачається.

Питання для самоперевірки

1. *Що означає термін "анімація"?*
2. *На які групи поділяються анімаційні ефекти об'єкта?*
3. *Опишіть послідовність додавання анімаційного ефекту до об'єкта?*
4. *Як створити для об'єкта довільний шлях переміщення?*
5. *При перегляді слайду об'єкт спочатку стрибає в інше місце, а потім рухається по кривій лінії. Яка причина такої поведінки?*
6. *Що означають квадратики з цифрами біля об'єкта на слайді?*
7. *Як змінити порядок виконання анімаційних ефектів?*

8. Як додати до звукового ефекту звуковий супровід?
9. Як додати анімаційний ефект для зміни слайдів?
10. Опишіть порядок додання на слайд відеокліпу?
11. Які є варіанти звукового супроводу презентації?
12. Як додати до слайда звуковий фрагмент, записаний з мікрофона?
13. Які переваги надає технологія OLE?

Урок 11. Керування показом презентації

Розглянуті раніше приклади презентацій мали лінійну структуру, тобто при перегляді слайди виводились на екран послідовно один за одним. Для більш гнучкого керування показом слайдів їх групують, створюючи *довільні покази*.

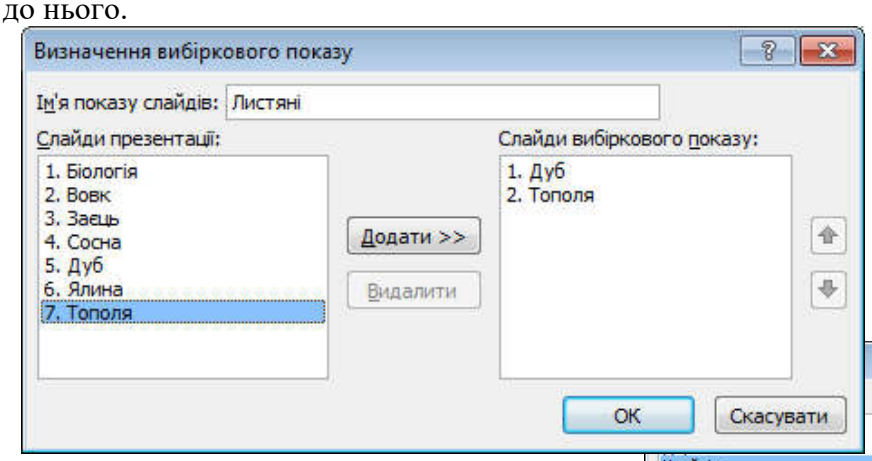
Довільні покази слайдів

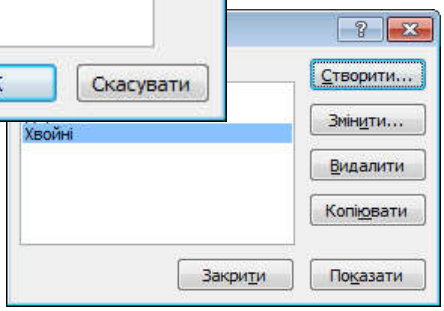
Довільний показ – це іменована послідовність слайдів презентації. Один слайд може входити одночасно в декілька довільних показів.

ПРИКЛАД. У презентації для захисту проекту з біології можуть бути створені довільні покази «Тварини», «Дерева», «Хвойні», «Листяні» тощо.

Довільні покази створюють тоді, коли слайди презентації вже створено. Щоб перейти до створення нового довільного показу, потрібно:

- на вкладці **Показ слайдів** в групі **Розпочати** показ слайдів натиснути кнопку **Настроюваний показ слайдів** і вибрати команду **Довільний показ**;
- у діалоговому вікні **Довільний показ** (рис. ??) натиснути кнопку **Створити**;
- у наступному вікні (рис. ??) зазначити назву показу, а також сформувати список слайдів, які увійдуть до нього.





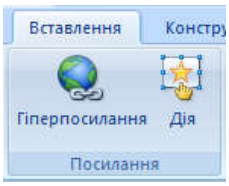
Кнопкою **Додати** назва слайда додається до списку вибраних слайдів. При натисканні кнопки **Видалити** назва слайда вилучається зі списку. Кнопки зі стрілками в правій частині вікна призначені для зміни порядку слайдів у створюваному довільному показі.

Гіперпосилання

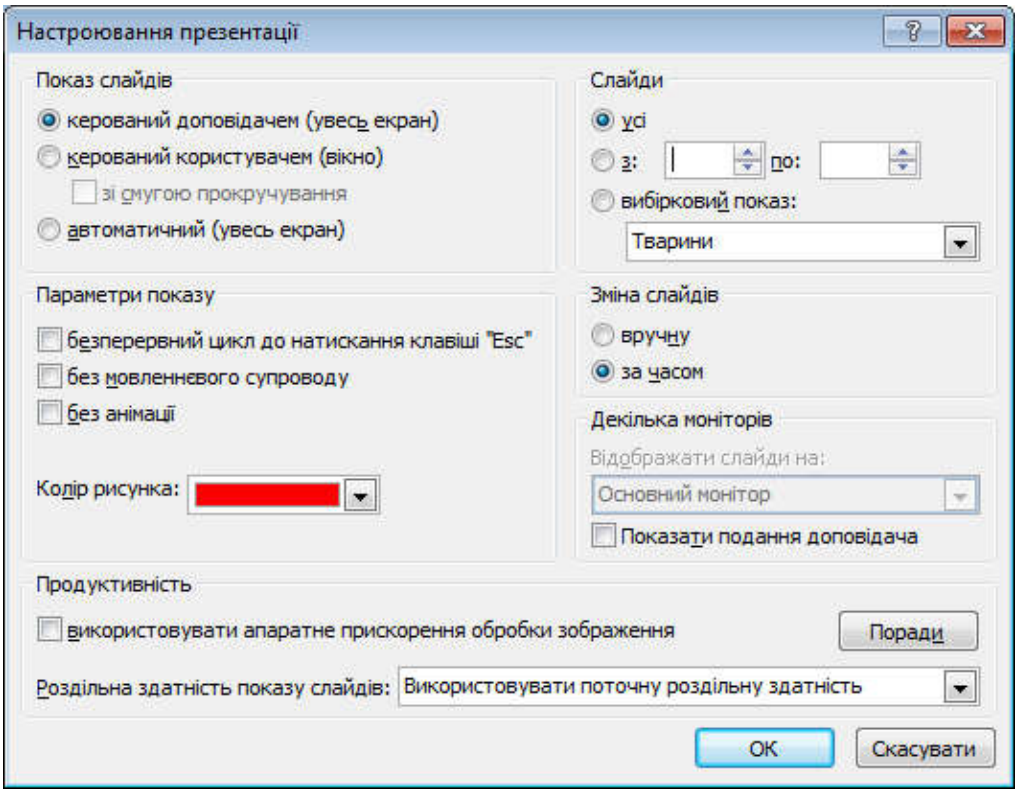
Подібно до того, як це робиться на веб-сторінках, гіперпосилання дозволяють під час перегляду презентації переходити до певних її частин або на веб-сторінки, відкривати інші документи.

Гіперпосилання може бути додане як до текстового фрагмента, так і до графічного об’єкта. Щоб створити гіперпосилання потрібно:

- виділити фрагмент тексту або малюнок;
- на вкладці **Вставлення** в групі **Посилання** вибрати команду **Гіперпосилання** (рис. ??).

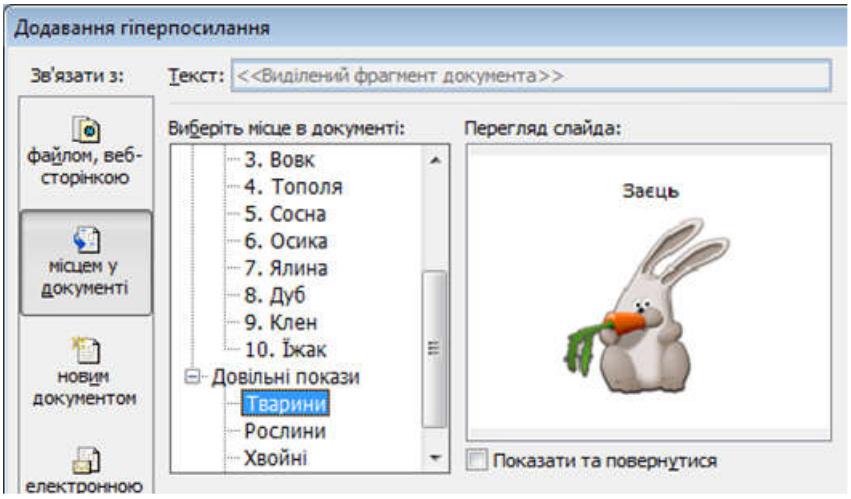


У вікні, що з’явиться, зазначають, з чим буде пов’язане гіперпосилання: файлом або веб-сторінкою; місцем у документі; новим документом; електронною поштою.

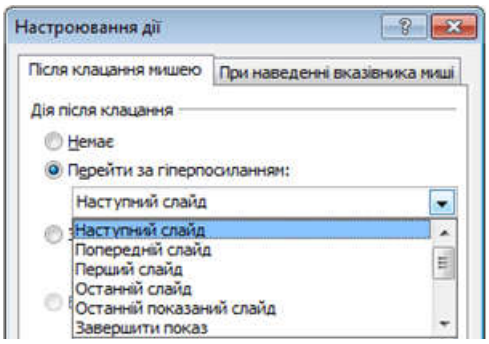


Щоб додати посилання на певний слайд презентації або на довільний показ, вибрають варіант **місцем у документі**. У діалоговому вікні, фрагмент якого показано на рис. ??, слід вибрати назву слайда або довільного показу, який відкриватиметься при клацанні на посиланні.

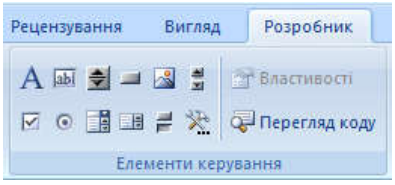
Якщо потрібно, щоб після перегляду довільного показу презентація продовжувалася зі слайду, на якому клацнули гіперпосилання, потрібно встановити прапорець Показати та повернутися.



Елементи керування в презентаціях



Окрім гіперпосилань, з фрагментом тексту або графічним об'єктом можна пов'язати певну дію, яка буде виконуватися при



клацанні. Для цього слід виділити фрагмент або об'єкт і вибрати на вкладці **Вставлення** в групі **Посилання** команду **Дія** (рис. ??).

У вікні **Настроювання дії**, яке з'явиться, залежно від того, коли має виконуватись дія, вибирають одну зі вкладок: **Після клацання мишею** чи **При наведенні вказівника миші**. Далі потрібно налаштувати дію, вибравши її зі списку **Перейти за гіперпосиланням** (рис. ??) або призначити запуск певної програми, встановленої на комп'ютері.

ПРИКЛАД. На вкладці **Вставлення** в групі **Зображення** серед фігур, які можна додати на слайд, є так звані **Кнопки дій** (рис. ??) – низка фігур, що зображають кнопки, для яких вже призначено одну з часто вживаних дій.

Управління показом презентації

Плануючи розробку презентацію слід урахувати, як буде здійснюватися керування переглядом. Можливі декілька варіантів:

- автоматичний (наприклад, на рекламному екрані на вулиці), пр. якому зміна слайдів відбувається через певний проміжок часу, заданий під час розробки;
- керований доповідачем (повноекранний), за якого доповідач змінює слайди, користуючись пультом дистанційного керування, мишею чи так званим поданням доповідача, зображеним на іншому моніторі;
- керований користувачем (віконний), за якого презентація виводиться у вікні, користувач змінює слайди за допомогою смуги прокрутки.

Вибирають один з варіантів, а також уточнюють додаткові параметри у вікні, яке відкривається командою **Показ слайдів → Настроювання показу**. **Слайдів**

Друк презентації

Попри те, що презентація призначається для перегляду на екрані, може виникнути необхідність роздрукувати її. Список для вибору режиму друку знаходиться у вікні **Друк**. Залежно від потреби, можна роздрукувати:

- **слайди** – друкується по одному слайду на аркуш;
- **видачі** – друкується 1 або більше слайдів на аркуш з датою і номером сторінки;
- **нотатки** – друкується 1 слайд на аркуш і додані до нього нотатки;
- **структуру** – друкується структура презентації у вигляді багаторівневого списку.

Питання для самоперевірки

1. *Що таке довільний показ? Наведіть приклад.*
2. *Як створити довільний показ?*
3. *Що таке гіперпосилання?*
4. *До яких елементів презентації можна додати гіперпосилання?*
5. *Як додати гіперпосилання до малюнка?*
6. *Як зробити, щоб після перегляду довільного показу презентація продовжилась з того ж місця?*
7. *Як зробити, щоб при клацанні на прямокутнику, розташованому на слайді, відбувався перехід на перший слайд?*
8. *Які дії можна призначати об'єктам слайду?*
9. *Що таке кнопки дій?*
10. *Які є варіанти керування переглядом презентації в цілому?*

Урок 12. Практична робота 3. Проектування та розробка презентацій за визначеними критеріями

Див. робочий зошит «Інформатика. 6 клас» / Бондаренко О.О., Ластовецький В.В., Пилипчук О.П., Шестопалов Є.А. – Шепетівка: «Аспект», 2017.

Урок 13. Практична робота 4. Розробка презентацій з елементами мультимедіа

Див. робочий зошит «Інформатика. 6 клас» / Бондаренко О.О., Ластовецький В.В., Пилипчук О.П., Шестопалов Є.А. – Шепетівка: «Аспект», 2017.

3. Алгоритми та програми

Урок 14. Алгоритм та його властивості

Поняття алгоритму

Згадаємо поняття алгоритму, знайоме вам з курсу інформатики 2-4 класу.

Алгоритм – це скінчена послідовність вказівок виконання дій, спрямована на розв'язування задачі.

Слово «алгоритм» походить від імені арабського математика Аль-Хорезмі (800-847 рр)., який сформулював правила чотирьох арифметичних дій над багатозначними числами. Латиною ім'я автора європейці писали як «Algorithmi», і спочатку алгоритмами називали саме ці 4 правила дій. З часом алгоритмами стали називати способи розв'язування самих різних задач.

У повсякденному житті людина зустрічається з алгоритмами, що визначають послідовність дій різноманітної природи. З курсу математики вам відомі алгоритми виконання арифметичних операцій над багатоцифровими числами; алгоритми знаходження коренів лінійного рівняння та ін. Рецепти для приготування їжі, ліків, правила користування автоматичними пристроями та інші послідовності розпоряджень, виконання яких дає змогу досягти поставленої мети, теж є алгоритмами.

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

Приклад 20.1. Для багатьох ігор, в яких результат гри залежить не від випадкового збігу обставин, а від кмітливості гравця і попереднього розрахунку, існують алгоритми виграшу.

Гра Баше. Є 11 предметів. За один хід гравець може взяти 1, 2, 3 предмети. Програє той, хто змушений взяти останній предмет.

Алгоритм виграшу для 1-го гравця:

1 хід: узяти 2 предмети.

2 хід і далі: брати стільки предметів, щоб кількість предметів, узятих разом із суперником за черговий хід, у сумі складала 4.

Властивості алгоритму

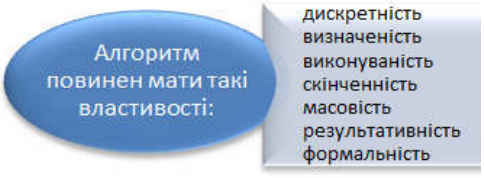
Алгоритм складається з окремих кроків у певному порядку. Якщо порушити порядок виконання кроків або загубити якийсь крок, алгоритм стане неправильним, тобто або не виконається до кінця, або призведе до неправильного результату.

Приклад 20.2. Дано алгоритм відкривання двері:

1. Дістати ключ.
2. Вставити в замкову шпару.
3. Повернути 2 рази за годинниковою стрілкою.
4. Вийняти ключ.

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

Щоб алгоритм виконав своє призначення, його необхідно будувати за певними правилами.



Дискретність означає, що алгоритм повинен складатися з окремих завершених дій. Розпорядження повинні мати форму «виконати», «зробити», а не «виконувати», «робити».

Визначеність – однозначність тлумачення правил виконання дій і порядку їхнього виконання. Алгоритм не повинен містити команди, які можуть сприйматися виконавцем неоднозначно, наприклад, «Порівняти числа А і В», «Вимкнути світло через кілька хвилин» тощо.

Виконуваність алгоритму означає, що алгоритм, призначений для певного виконавця, може містити тільки команди, які виконавець здатний виконати. Так, алгоритм для вико-

навця «Першокласник» не може містити команду «Побудуй бісектрису даного кута», хоча така команда може бути в алгоритмі, який призначений для виконавця «Восьмикласник».

Скінченність – потенційна виконуваність алгоритму. Алгоритм повинен складатися зі скінченого числа кроків, кожний з яких вимагає для свого виконання скінченого проміжку часу.

Приклад 20.3. Наступна послідовність команд є нескінченною:

- Взяти число 2.
- Помножити взяте число на 10.
- Додати до одержаного числа 5.
- Якщо одержане число додатне, то виконати команду 3, якщо ні, то припинити виконання алгоритму.

Масовість – придатність до використання для великої кількості варіантів вхідних даних. Наприклад, алгоритм знаходження коренів лінійного рівняння повинен бути придатним для розв′язування будь-якого рівняння виду ax+b=c.

Результативність означає, що виконання послідовності вказівок алгоритму повинне приводити до цілком конкретного результату. Наприклад, алгоритм розв'язування рівняння повинен містити перевірку на випадок, коли коренів не існує, і передбачати видання повідомлення про відсутність коренів.

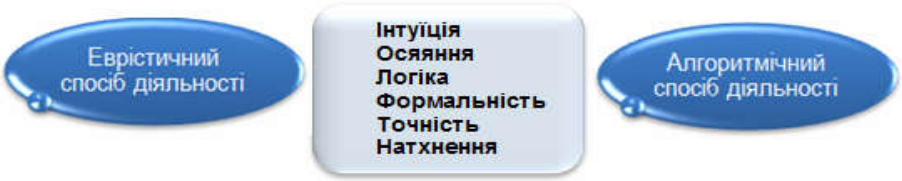
Формальність означає, що будь-який виконавець, здатний сприймати і виконувати вказівки алгоритму (навіть не розуміючи їх змісту), діючи за алгоритмом, може виконати поставлене завдання. Як відомо, автомати правильно розв'язують багато задач за заданими їм алгоритмами, хоча суть задач, безумовно, автомати розуміти не можуть.

Питання для самоперевірки

1. *Що таке алгоритм?*
2. *Назвіть основні вимоги до алгоритмів і поясніть суть кожної з них.*
3. *Нехай маємо такий алгоритм:*
   a) прочитати перше число – a1;
   b) прочитати друге число – a2;
   c) поділити число a1 на a2;
   d) записати результат.

   Чи має даний алгоритм властивості масовості та визначеності?
4. *Нехай маємо такий алгоритм для розв'язування задачі: деяке задане число, більше за 1, зменшити до 1 шляхом ділення на 2:*
   a) поділити число на 2;
   b) якщо одержане число не дорівнює 1, то виконати команду 1, інакше припинити виконання алгоритму.

   Чи має даний алгоритм властивість скінченності?
5. *Вкажіть стрілками, до якого способу діяльності відносяться терміни:*



6. *Чи можна скласти алгоритм для розв'язування таких задач:*
   a) знайти корінь рівняння ax+b=c;
   b) відвідати театр;
   c) вивести новий сорт пшениці;
   d) сконструювати машину для виконання домашніх завдань.

Урок 15. Виконавець алгоритмів та система його команд

Виконавці алгоритмів

Кожен з виконуваних на практиці алгоритмів орієнтовано на певного виконавця. Виконавцем алгоритму може бути людина, комп'ютер, система людина-машина, верстат-автомат, робот 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.

Виконавець – людина, тварина чи пристрій, здатні діяти за алгоритмом.

Щоб скласти орієнтований на конкретного виконавця алгоритм, потрібно знати характеристики виконавця.

Характеристики виконавця

Кожен виконавець працює або живе в певних умовах, середовищі; і може виконувати певний набір дій. Перш ніж складати алгоритм розв'язування задачі, потрібно дізнатися, які д її виконавець може виконувати.

Середовище – «місце існування» виконавця.

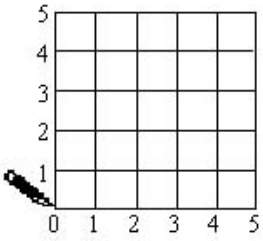
Припустимі дії – обмежений набір дій, що вміє виконувати даний виконавець.

Описати виконавця – значить указати його припустимі дії. Досяжні цілі – результати, що виконавець може одержати за допомогою своїх припустимих дій.

Система команд виконавця – повний перелік команд, які «розуміє» виконавець. Виконавця можна уявити у вигляді пристрою з кнопковим управлінням, де кожна кнопка відповідає одній команді, натискання кнопки означає виклик команди.

Відмова – подія, що виникає при виклику команди в неприпустимому для даної команди стані середовища.

Приклад 21.1. Виконавець Кресляр призначений для побудови малюнків та креслень на полі розміром 5×5 клітин і вміє виконувати 3 команди:



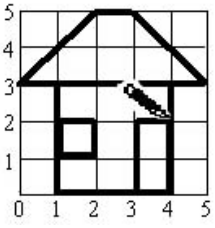
- підняти перо;
- опустити перо;
- перейти до точки з координатами (x,y).

Якщо перо опущене, при пересуванні Кресляра за ним залишається слід. На початку роботи Кресляр знаходиться в точці (0,0) і тримає перо піднятим. Припустимим діям виконавця відповідає система команд:

Припустимі дії виконавця	**Система команд виконавця**
Підняти перо	Підніми перо
Опустити перо	Опусти перо
Перейти до точки з координатами (x,y)	Перейди до точки (x,y)

Відмова Кресляра виникає, якщо він отримує команду перейти в точку, яка знаходиться за межами поля.

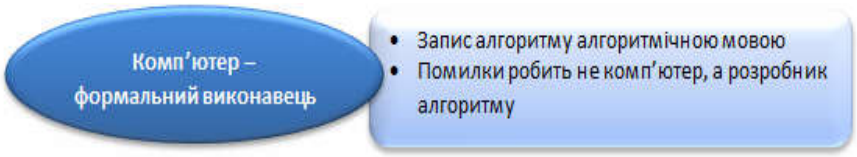
Приклад 21.2. Складемо для виконавця Кресляра алгоритм побудови даху будиночка на полі 5×5 клітин:



- Перейди до точки (0,3).
- Опусти перо.
- Перейди до точки (2,5).

- Перейди до точки (3,5).
- Перейди до точки (5,3).
- Перейди до точки (0,3).
- Підніми перо.

Виконуючи алгоритм, виконавець може не розуміти сенс того, що він робить, і тим не менше отримувати потрібний результат, тобто виконавець діє формально.



Питання для самоперевірки

7. *Поясніть поняття «виконавець алгоритму». Перерахуйте характеристики виконавця.*
8. *Яку систему команд повинен мати виконавець, щоб реалізувати алгоритм обчислення за формулою: p=(1+x)/(1-x) ?*
9. *Якими припустимими діями ви оснастили б автомат, що заміняє:*
   a) касира в магазині;
   b) двірника;
   c) вахтера;
   d) директора школи?
10. *Виконавець вміє: помножити число на 2; знайти суму двох чисел. Складіть для виконавця алгоритм обчислення виразу y=2a+3b+4c.*
11. *Виконавець вміє: помножити число на 2; збільшити число на 1. Складіть для виконавця алгоритм одержання:*
    a) числа 4 з 1;
    b) числа 5 з 1;
    c) числа 100 з 1.
12. *Фірма "Електронні прилади" випустила автоматизовану ванну, пульт управління якої має дві кнопки Долити 5 л і Злити 3 л. Складіть алгоритм, що дозволяє долити в ванну 4 л води за якомога меншу кількість команд.*

Урок 16. Способи опису алгоритму. Алгоритмічні структури

Форми подання алгоритмів

Існують різні форми подання алгоритмів – словесні, словесно-формульні, графічні, у вигляді програмного коду та інші – залежно від того, на якого виконавця орієнтований алгоритм.

У повсякденному житті найчастіше застосовується словесний спосіб. Алгоритм подається як послідовність окремих занумерованих пунктів, кожний з яких містить команду на виконання певної дії. Команди записуються словами. Пункти виконуються один за одним у порядку зростання їх номерів, якщо немає спеціальної вказівки на перехід до виконання іншого пункту, номер якого задається.

Словесний спосіб подання алгоритму є найбільш прийнятним для опису інструкцій побутового характеру, дій на випадок надзвичайної ситуації, кулінарних рецептів тощо.

Приклад 22.1. Алгоритм швидкого виготовлення піци:

5. Змішайте сметану з томатною пастою, додайте спеції.
6. Отриманий соус намажте на хліб.
7. Зверху викладіть порізані на шматочки помідори та болгарський перець, половинки маслин.
8. Осипте натертим сиром.
9. На 5 хвилин поставте піцу в духовку.

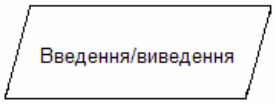
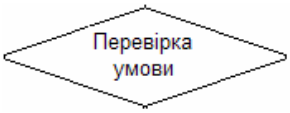
Обчислювальні алгоритми можна задавати й у вигляді формул. Так, алгоритм обчислення площі прямокутного трикутника можна подати у вигляді формули $S = \frac{a \cdot b}{2}$.

При запису алгоритмів часто комбінують словесне і формульне подання запису вказівок.

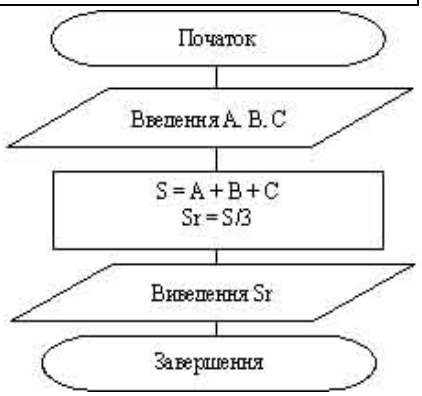
Для графічного представлення алгоритмів використовують блок-схеми.

Блок-схема алгоритму – графічне зображення алгоритму у вигляді організованої послідовності блоків.

Графічне зображення базових алгоритмічних структур

Назва блоку	Опис дії
	Позначає початок та завершення алгоритму
	Позначає дію, яку потрібно виконати. Прямокутником може бути позначена як вказівка виконати окрему дію (додати два числа, накреслити лінію), так і послідовність логічно об'єднаних у блок дій (виконати розрахунки за заданими формулами, намалювати малюнок), тобто певний процес.
	Позначає введення вхідної інформації і виведення проміжної і результуючої інформації.
	Позначає перевірку значення логічного виразу деякої умови. Логічний вираз може набувати одного з двох значень – *TRUE* (істина) або *FALSE* (хибність).

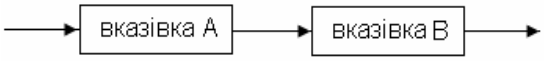
Приклад 22.2. Блок-схема алгоритму знаходження середнього арифметичного трьох чисел.



Алгоритмічні структури

При конструюванні алгоритмів використовуються три базові алгоритмічні структури: слідування, розгалуження, повторення. Згадаємо означення алгоритмічних структур.

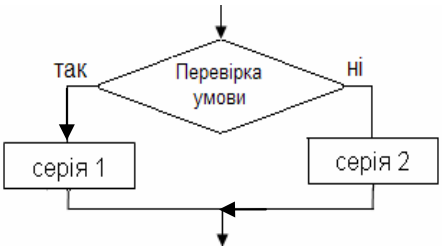
Слідування – це така організація дій в алгоритмі, при якій дії виконуються послідовно, одна за другою, без пропусків або повторень.



Приклад 22.3. Алгоритм створення комп'ютерної програми:

10. скласти алгоритм;
11. написати програму;
12. відлагодити програму на ПК;
13. отримати рішення задачі.

Розгалуження – це така форма організації дій, при якій, в залежності від виконання або невиконання певної умови, виконується одна з двох послідовностей дій.



Умова **–** це питання, сформульоване так, що допускає лише одну з двох відповідей: «так» або «ні». Перевірка умови повинна бути допустимою дією виконавця.

Якщо умова є істинною, то виконуються команди серії 1 (гілка «Так»), якщо ж умова не виконується (умова хибна), – команди серії 2 (гілка «Ні»). Після виконання однієї з серій команд виконавець переходить до наступної після розгалуження команди.

Приклад 22.4. Блок-схема вказівки розгалуження: якщо у касі є квитки, то придбати квиток і переглянути фільм, інакше піти на прогулянку до парку (рис.22.4).

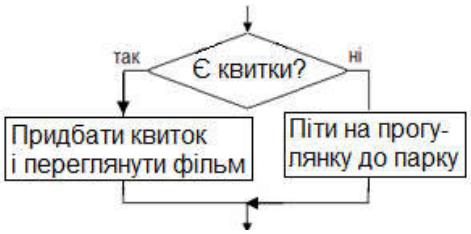


Рис.22.4

Повторення (цикл) – це форма організації дій, за якою одна і та ж послідовність дій виконується кілька разів доти, поки виконується деяка умова. Серія команд, що повторюється при кожному проході (ітерації) циклу, називається ***тілом циклу***.

При виконанні алгоритму спочатку перевіряється умова, і якщо вона істинна, то тіло циклу виконується черговий раз, і відбувається повернення на перевірку умови. Якщо умова хибна, то виконання циклу припиняється.

Якщо умова у вказівці повторення виявиться хибною при першій перевірці, то тіло циклу не виконається жодного разу.

Якщо ж при повторенні циклу умова незмінно залишається істинною, то цикл може повторюватися нескінченно (кажуть, програма «зациклена).

Приклад 22.5. Блок-схема алгоритму збору врожаю (рис.22.5).

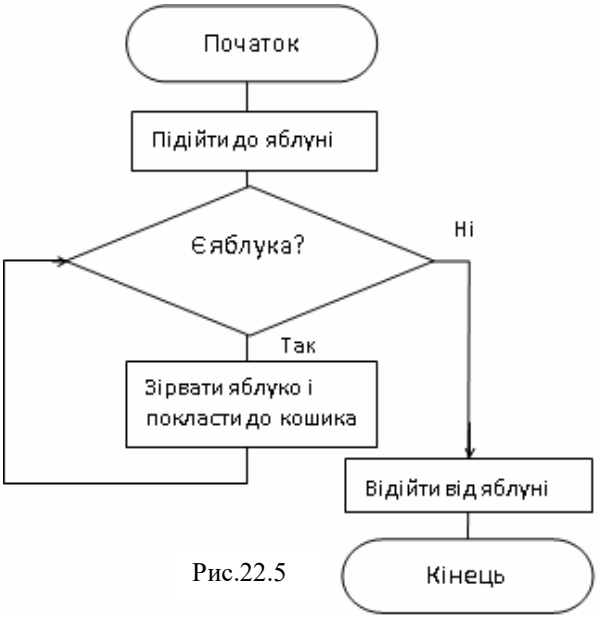


Рис.22.5

При складанні алгоритму розв'язування задачі потрібно дотримуватись чіткого плану дій:

14. Уважно прочитати умову задачі.
    Визначити:
    a) що дано (аргументи);
    b) що потрібно знайти (результати);

    Визначити спосіб розв'язування задачі та виявити необхідні проміжні величини.
15. Скласти блок-схему алгоритму.
16. Перевірити правильність складання алгоритму при конкретних значеннях аргументів

<u>Програма та мова програмування</u>

Програма – впорядкована послідовність команд для комп'ютера, виконання якої реалізує алгоритм розв'язування певної задачі. Команди в програмі (програмному коді) записуються мовою програмування.

Мова програмування – це система позначень, яка використовується для запису алгоритмів для реалізації (виконання) їх за допомогою комп'ютера.

Програма – це алгоритм, записаний мовою програмування.

Ви вже знайомі з візуальною мовою програмування Scratch і основними поняттями, які будуть корисними при вивченні інших мов програмування.

Питання для самоконтролю

1. *Назвіть основні способи опису алгоритмів.*
2. *Як зображується базова структура «слідування»?*
3. *Як зображується і що означає базова структура «розгалуження»?*
4. *Дайте визначення розгалуження як алгоритмічної конструкції.*
5. *Дайте означення циклу як алгоритмічної конструкції.*
6. *У якому випадку цикл не виконується жодного разу?*
7. *У якому випадку виконання тіла циклу повторюється нескінченно?*
8. *Поясніть переваги подання алгоритму у вигляді блок-схеми.*

Урок 17. Середовище опису й виконання алгоритмів

Почнемо знайомство з популярною сучасною мовою Python, яка підходить для розв'язування різних задач. Мова програмування Python була створена у 1991 році голландцем Гвідо ван Россумом. Своє ім'я Python отримав від назви телесеріалу ("Monty Python"), а не плазуна.

Установка Python 3

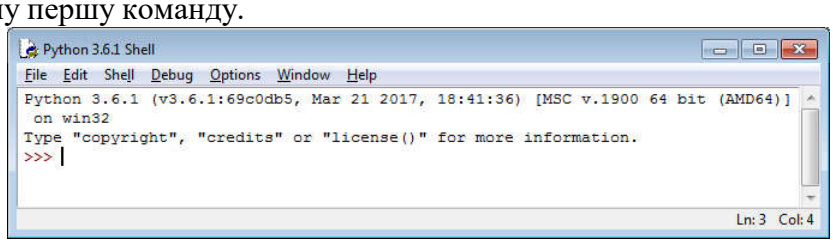
Перш ніж почати програмувати на Python, програмне середовище потрібно встановити на комп'ютер. Завантажити файл для інсталяції можна з сайту python.org.

Разом з Python 3 на комп'ютер буде встановлена програма IDLE. Це орієнтоване на початківців середовище розробки, в якому є текстовий редактор для написання і налагодження Python-програми.

Алгоритм установки:

- Зайди на сайт http://www. python.org.
- Клікні Downloads, щоб відкрити сторінку завантаження.
- Клікні по кнопці з версією Python 3.6.1:
- Після завантаження інсталяційного файлу зроби по ньому подвійний клік, щоб встановити Python.
- Запусти IDLE. Для цього виконай команди ПУСК/Всі програми/ Python/ IDLE.

Відкриється вікно IDLE. Ми можемо починати програмувати, записуючи команди після позначки «>>>». Рядок >>> називається запрошенням, і його наявність означає, що комп'ютер готовий прийняти вашу першу команду.



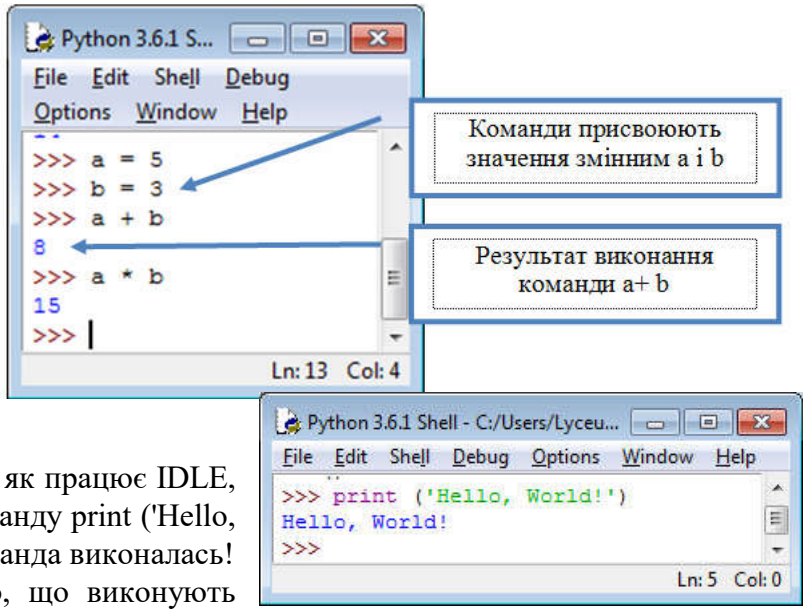
В Python існують два види вікон: вікно консолі (IDLE) і вікно програми.

У вікні програми можна писати і зберігати програмний код, а у вікні консолі – одразу виконувати команди Python.

Знайомство з IDLE

Вікно IDLE є вікном консолі. У цьому вікні відображаються результати виконання програми і повідомлення про помилки у програмному коді.

Python може виконувати команди, що введені у вікні консолі. Ці команди виконуються після натискання клавіші Enter, і результат одразу виводиться у вікні IDLE.



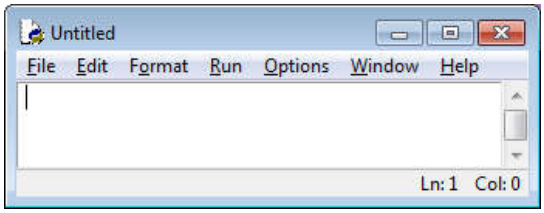
Приклад 23.1. Щоб зрозуміти, як працює IDLE, після позначки «>>>» запишіть команду print ('Hello, World!') і натисніть Enter. Наша команда виконалась!

У вікні консолі одразу видно, що виконують різні команди Python.

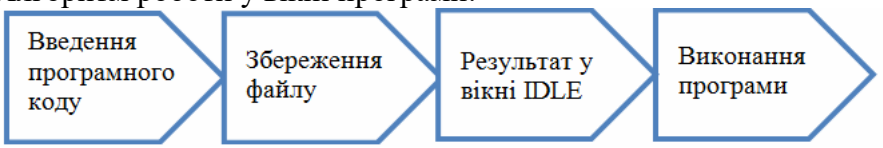
Вікно IDLE зручно використовувати для того, щоб зрозуміти, що виконує та чи інша команда. Крім того, вікно консолі можна використовувати як калькулятор. Але для великих програм, які потрібно зберігати і редагувати, потрібне вікно програми.

Вікно програми

Вікно програми призначене для написання і редагування тексту програми. Щоб відкрити вікно програми, в IDLE виконайте команду File/New File. Відкривається окреме вікно програми, яка до першого збереження має ім'я «Untitled».



Програму можна ввести, зберегти, запустити на виконання і переглянути результат у вікні консолі. Алгоритм роботи у вікні програми:

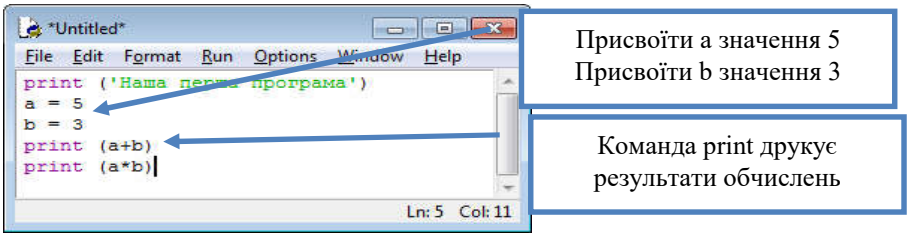


Програму не можна запустити на виконання, поки вона не збережена.
При спробі виконати програму без збереження з'явиться попередження.

Приклад 23.2. Виконаємо алгоритм роботи у вікні програми:

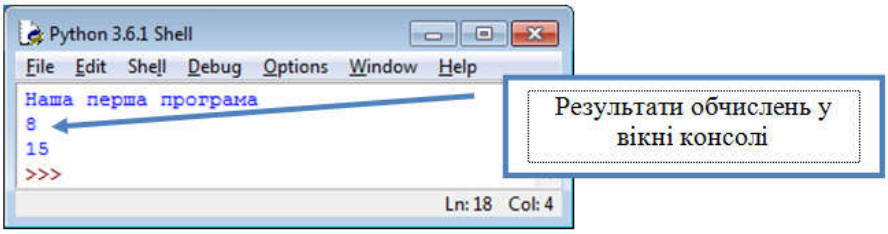
① Введіть код у вікні програми:

② виконайте команду File/ Save As. Введіть ім'я файлу Перша програма і натисніть Save. Файл збережено з розширенням .py.



③ виконайте команду Run/Run Module (або натисніть F5).

④ Перегляньте результат роботи програми у вікні консолі.



Кольори в тексті програми

IDLE розфарбовує текст програми різними кольорами.

Команди Python, наприклад, print, – фіолетового кольору

Рядки в лапках виділені зеленим кольором. Якщо дужки навколо текстового рядка теж виділені зеленим, десь не вистачає лапки

Службові слова мови Python (if, while ті інші) – помаранчевого кольору

Помилки у вікні програми і повідомлення про помилки у вікні консолі виділяються червоним

Ці кольори є для нас підказками, як Python сприймає різні складові програмного коду. Кольорові підказки допоможуть нам уникнути помилок при наборі тексту програми.

Питання для самоперевірки:

1. *Які види вікон існують в Python?*
2. *Які повідомлення відображаються у вікні консолі?*
3. *Як відкрити вікно програми?*
4. *Для чого призначене вікно програми?*
5. *Опишіть алгоритм роботи у вікні програми.*
6. *Чому складові тексту програми виділяються різними кольорами?*

Урок 18. Основні поняття мови Python

Згадайте, з чого ви починали вивчення української та іноземної мови – з алфавіту. Потім ви вивчали правила запису слів – синтаксис мови, і значення нових слів – семантику мови.

Основними складовими будь-якої мови програмування є алфавіт, синтаксис і семантика.

Отже, і вивчення мови Python ми теж розпочнемо з алфавіту.

Алфавіт мови Python

У мові Python при створенні програм можуть використовуватися такі символи:

- літери латинського алфавіту A..Z, a..z; цифри 0..9;
- знаки арифметичних операцій, спеціальні символи: + – * / \^ = < > ( ) . , : ; ‘ # _; комбінації символів: <=, >=, <>, = =; !=,**;
- службові слова, що мають фіксований для Python зміст, наприклад: **and, elif, if, print, as, else, import** тощо.

Синтаксис мови – сукупність правил побудови команд мови програмування.

Семантика мови – сукупність правил виконання комп’ютером команд, записаних мовою програмування.

З синтаксисом та семантикою команд Python ви будете ознайомлюватися по мірі вивчення мови програмування.

Величини в мові Python

Окремий інформаційний об’єкт (число, символ, рядок та ін.) називають ***величиною***. Основними характеристиками величин є назва, вид, тип і значення

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* — величини, значення яких можуть змінюватися в ході виконання програми.

Змінна – це просто іменована частина пам’яті твого комп’ютера, де ти тримаєш певну інформацію. На відміну від констант, потрібен якийсь спосіб для отримання доступу до змінних, і саме тому змінним дають імена (ідентифікатори).

Існують певні правила, яким потрібно слідувати при іменуванні змінних:

1) Першим символом імені має бути літера чи знак нижнього підкреслювання '_'.

2) Решта імені може складатися з літер, чисел або знаків нижнього підкреслювання. Не можна використовувати спеціальні символи, такі, як /, # або @.

3) Не можна використовувати пробіли, замість пробілу можна застосувати нижнє підкреслення.

3) Імена змінних чутливі до регістру символів. Наприклад, myname і myName – це різні змінні.

4) Не можна називати змінні іменами команд, наприклад, print.

Приклад 24.1. Правильними ідентифікаторами є i, __my_name, name_23, a1, b2. Приклад неправильних імен: 2things, this is spaced out, my-name.

Коментарі в програмі

Коментар — це текст, призначений для читання людиною, а не комп’ютером. Коментар – це підказка для нас, яку дію виконує програма. Вставляючи коментарі в текст програми, ми спрощуємо собі та іншим її читання і розуміння.

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 Python проігнорує ці фрагменти коду.

Приклад 24.2. Коментар пояснює призначення наступної команди:

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

Типи величин в мові Python

Змінні зручно представити у вигляді "поштових скриньок" (комірок пам'яті комп'ютера), на які навішені ярлики з їх іменами. Для різних величин створюються «скриньки» різного розміру, який залежить від типу величини.

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 і рядки.

В Python є два типи числових даних: цілі числа (int), тобто числа без дробової частини, і дійсні (float) – дробові числа з десятковою крапкою.

Цілі числа потрібні для рахунку (перший, другий, третій…). Кількість учнів у класі, вік людини, кількість предметів ми зазвичай указуємо за допомогою цілих чисел. Числа з плаваючою крапкою, або десяткові дроби, потрібні, коли ми хочемо указати частину чого-небудь, наприклад, 3.5 м, 1.25 грн. Звісно, у програмі ми не будемо указувати одиниці вимірювання (метри, гривні), лише число з дробовою частиною.

В якості роздільника між цілою і дробовою частиною числа використовуйте крапку.

Рядок – це взята в одинарні лапки послідовність будь-яких символів – цифр, літер, розділових знаків. У змінних рядкового типу ми зберігатимемо фрагменти тексту.

Приклад 24.3. Типи величин в мові Python (рис.24.1).

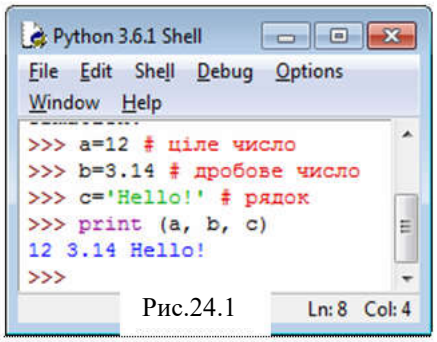


Рис.24.1

Приклад 24.4. Щоб дізнатися тип величини, можна у вікні консолі виконати команду type.

Дізнаємось тип величин a=24, b=3.14, c='a book' (рис.24.2).

```
>>> type (24)
<class 'int'>
>>> type (3.14)
<class 'float'>
>>> type ('a book')
<class 'str'>
>>>
```

Рис.24.2

Оператори Python

Математичні символи, такі як + (плюс) і – (мінус), називаються операторами, так як вони оперують (або виконують обчислення) числами в рівнянні. У мові Python використовується більшість операторів, якими ви користуєтеся на уроках математики, в тому числі +, -, дужки (). Однак деякі оператори відрізняються від використовуваних в школі: так, оператор множення представлений зірочкою (*), а оператор ділення – косою рискою /.

Математичні оператори Python			
Операція	**Символ оператора**	**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 = 5
Піднесення до степеня	**	A = 4**2	A = 16

Приклад 24.5. Виконаємо у вікні консолі обчислення виразів:

```
>>> 5/2
2.5
>>> 5//2
2
>>> 5%2
1
>>> 5**2
25
```

Приклад 24.6. Дужки указують Python, яку частину виразу обчислювати в першу чергу, тобто регулюють порядок дій у виразі:

```
>>> 3*(6+2)
24
>>> 3*6+2
20
>>> 6/3+2
4.0
>>> 6/(3+2)
1.2
```

Випадкові числа

Випадкові числа часто застосовують у програмуванні при створенні ігрових або тестових програм тощо. Щоб отримати випадкове число, необхідно за допомогою команди import завантажити в Python функцію randint. Функція randint(x1,x2) вибирає ціле випадкове число в діапазоні від x1 до x2.

Приклад 24.7. Отримання випадкового числа в діапазоні від 1 до 10.

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

Синтаксичні помилки

Якщо Python не може зрозуміти введену вами команду, то він може вивести у відповідь повідомлення про помилку з текстом “SyntaxError”. Це означає, що виникла проблема з тим, яким чином ви попросили комп'ютер виконати якесь завдання, тобто з синтаксисом.

Місце помилки у вікні консолі помічається червоним кольором. Після команди, що містить помилку, виводиться повідомлення про помилку. Уважно читайте це повідомлення – це допоможе зрозуміти, в чому помилка, і виправити її.

Приклад 24.8. У виразі допущено помилку: надруковано зайву дужку.

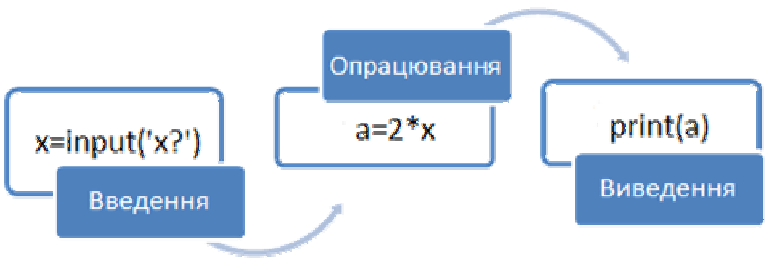
```
>>> (x+5))
SyntaxError: invalid syntax
```

Питання для самоперевірки:

1. *Назвіть основні характеристики величини.*
2. *Назвіть і охарактеризуйте види величин.*
3. *Які імена недопустимі в якості ідентифікаторів і чому?*
   а) suma; б) w1; в) primer 1; г) (sum); д) a–4; е) if; ж) 8a.
4. *Дайте визначення типу величини.*
5. *Обчисліть значення виразів:*
   а) 7/2 б) 7 // 2 в) 7 % 2
   г) 123 //100 д) 123 % 10 є) (123 // 10)% 10
6. *Як отримати випадкове число в діапазоні від 1 до 100?*

Урок 19. Лінійні алгоритми

Алгоритми, у яких використовується тільки структура «Слідування», називаються ***лінійними***. В програмах, що реалізують лінійні алгоритми, використовуються команди введення даних, присвоєння і виведення.



Будь-яка програма повинні прийняти вхідні дані (введення), опрацювати їх і повернути результат (виведення).

Введення даних

Команда input() призначена для введення даних з клавіатури. Коли програма зустрічає команду input, вона призупиняє роботу і очікує, поки користувач введе дані і натисне Enter. В дужках записується підказка користувачеві, що саме потрібно ввести.

Приклад 25.1. Випробуємо у вікні консолі, як працює input():

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

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

Приклад 25.2. При спробі додати до значення змінної a числа 3 виникне помилка, тому що 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 в число за допомогою функції int().

Функція int(s) перетворює рядок s в ціле число.

Приклад 25.3. Тепер помилки немає:

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

Функція float(s) перетворює рядок s в дробове число.

Вказівка присвоєння

В Python, як і в більшості сучасних мов програмування, ми присвоюємо значення змінної за допомогою знаку «=». Вираз присвоювання, такий як x=7, наказує комп'ютеру запам'ятати число 7 у змінній x.

Загальний вигляд команди присвоєння:

A = B,

де A – ім'я змінної, B – константа, змінна або вираз.

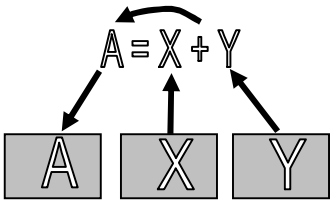


Схема виконання вказівки присвоєння: спочатку обчислюється значення виразу в правій частині вказівки присвоєння, потім це значення надається змінній, ім'я якої записане в лівій частині.

Присвоєння заповнює ділянку пам'яті, відведену для змінної, новим значенням, одночасно знищуючи старе.

Приклад 25.4. Наступні команди присвоюють змінній rabbits значення 5, потім те ж 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 (рис.24.1).

Приклад 25.5. Нехай a=10, x=2, y=3. Тоді після виконання вказівки присвоєння a = x + y змінна a отримає значення 5.

Приклад 25.6. Нехай a=10. Тоді після виконання вказівки присвоєння a = a+5 змінна a отримає значення 15.

Виведення значень змінних

В попередніх прикладах ми вже зустрічали команду print, яка потрібна, щоб вивести текст у вікно консолі.

Приклад 25.7. За допомогою команди print можна дізнатися значення змінної.

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

Приклад 25.8. Якщо потрібно вивести значення декількох змінних або виразів, їх потрібно перелічити через кому:

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

Приклад 25.9. 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
```

Питання для самоперевірки:

1. *Для чого призначена команда input()?*
2. *Для чого призначена команда print()?*
3. *Назвіть константи і змінні в списку виведення:*
4. *print ('a=', a, 5, '3 * b', 3*b)*
5. *Чому дорівнює значення X після виконання послідовності присвоювань: а) y = 2; x = y; б) x = 8; x = x+2; в) x = 5; x = -x; г) x=10; x=x +3?*
6. *Після виконання оператора x = y+x нові значення змінних x=10, y=3. Чому дорівнювали x і y до виконання оператора присвоєння?*
7. *Після виконання оператора x = y+x нові значення змінних x=3, y=10. Чому дорівнювали x і y до виконання оператора присвоєння ?*

Урок 20. Черепашача графіка

У світі Python Черепашкою зветься уявний робот – пристрій, який переміщається по екрану і повертається в заданих напрямках, при цьому залишаючи (або, за вибором, не залишаючи) за собою намальований слід заданого кольору і ширини. Положення і напрямок руху Черепашки відображає невелика чорна стрілочка, яка повільно пересувається по екрану. Це дозволяє відстежити рух Черепашки і зрозуміти, яким чином кожен рядок коду впливає на траєкторію руху Черепашки.

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

Система координат

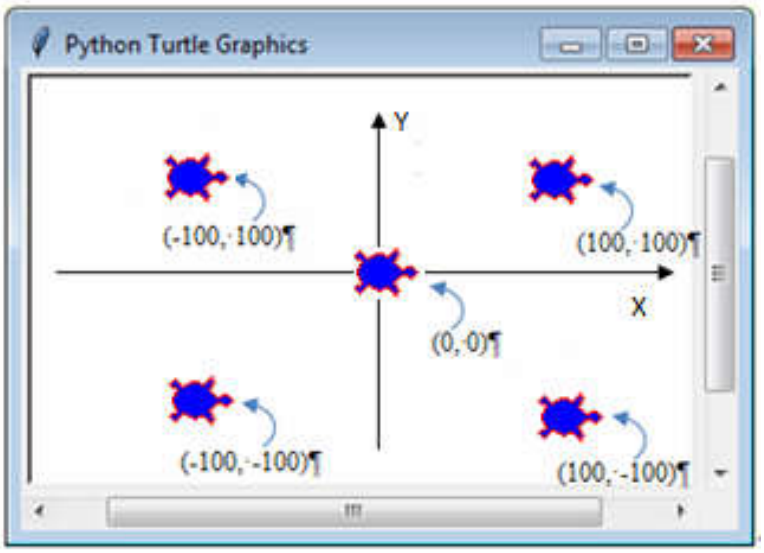
Результат виконання Черепашкою команд відображається у вікні Python Turtle Grafics (*див. малюнок*).

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 до верху, а горизонтальна *X* – зліва направо.

На математиці ви працювали з числами, розташованими на координатній прямій праворуч від 0. Але горизонтальну числову пряму можна продовжити вліво, а вертикальну – вниз від 0, а на променях ліворуч і знизу від 0 розташовані від'ємні числа, тобто числа зі знаком мінус (-).

На рисунку зображені 5 черепашок і указані координати їхнього місцеположення.



Команди Черепашки

Для завантаження команд роботи з Черепашкою потрібна команда:

from turtle import *

Після введення цієї команди ви можете давати Черепашці команди малювання.

Якщо у вашій програмі є і команди введення-виведення, і команди малювання, перед початком малювання запишіть команду

pendown() – почати малювати;

а завершивши створення малюнку – команду

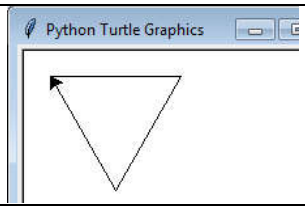
penup() – закінчити малювати.

Команда	Призначення
forward (n)	Проповзти вперед n кроків (пікселів)
left (angle)	Повернутися наліво на angle градусів
right (angle)	Повернутися направо на angle градусів
circle (r)	Намалювати коло радіуса r, центр якого знаходиться зліва від черепашки
circle (r,angle)	Намалювати дугу радіуса | r | і градусною мірою angle. Дуга малюється проти годинникової стрілки, якщо r> 0 і за годинниковою стрілкою, якщо r <0
goto (x,y)	Перемістити Черепашку в точку з координатами (x, y)
down ()	Опустити перо. Після цієї команди Черепашка почне залишати слід при будь-якому своєму пересуванні
up ()	Підняти перо
width (n)	Встановити ширину сліду Черепашки в n пікселів
write (s)	Вивести текстовий рядок s в точці знаходження Черепашки
clear ()	Очищення області малювання

Приклад 26.1. Виконання команд малювання

Введіть ці команди у вікні консолі. >>>from turtle import * >>> forward(100)	Черепашка рухається, залишаючи за собою лінію.

```
>>> right(120)
>>> forward(100)
>>> right(120)
>>> forward(100)
```



Приклад 26.2. Програма малювання кораблика.

```
from turtle import *
# Малюємо човника
 goto(-20,20)
goto(100,20)
goto(80,0)
goto(0,0)
up()
# Малюємо парус
goto(40,20)
down()
goto(40,120)
goto(60,60)
goto(40,20)
```

Створюємо кольоровий малюнок

Черепашка може залишати не тільки чорний, але й кольоровий слід.

З бібліотекою Turtle 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.

Команда	Призначення
color(s)	Встановити колір сліду Черепашки в s
color(s1,s2)	Встановити колір сліду Черепашки в s1, а колір заливки замкненої фігури в s2

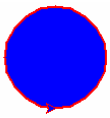
Для зафарбування замкнених фігур потрібні команди:

begin_fill() – почати стежити за черепашкою для заповнення області

end_fill() – заповнити кольором s2 область, пройдену Черепашкою, починаючи з begin_fill ().

Приклад 26.3. У вікні консолі введіть команди для малювання червоного кола, зафарбованого синім кольором.

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



Приклад 26.4. Програма малювання сонечка.

```
from turtle import *
width(3)
begin_fill()
color ('dark orange')
up()
goto(0,0)
down()
circle(50)
end_fill()
# малювання 1 променя
```

up()
goto(-100, 150)
down()
goto(100, -50)
up() {…}

Змінюємо зовнішній вигляд Черепашки

Ви помітили, що Черепашка у вікні Python Turtle Grafics зображується у вигляді стрілки. Напрямок стрілки вказує нам напрямок руху Черепашки. Але ми можемо змінити вигляд Черепашки за допомогою команди

shape(*Name*),

де *Name* – назва форми Черепашки з переліку ‘arrow’, ‘turtle’, ‘circle’, ‘square’, ‘triangle’, ‘classic’.

Приклад 26.5. Задаємо для Черепашки форму turtle:

>>> shape(‘turtle’)
>>> forward(100)

Зверніть увагу: якщо кольори малювання були змінені, черепашка теж стає кольоровою:

Приклад 26.6. Черепашка наслідує кольори малювання:

>>> color('red', 'blue')
>>> shape(‘turtle’)
>>> forward(100)

Після завершення алгоритму малювання зображення Черепашки на малюнку може бути зайвим. Сховати Черепашку можна за допомогою команди **hideturtle()**, а показати – за допомогою команди **showturtle()**.

Черепашача графіка надає нам можливості створювати яскраві складні малюнки.

Питання для самоперевірки:

1. *Яка команда потрібна для завантаження команд роботи з Черепашкою?*
2. *Запишіть команди для Черепашки:*
3. *Проповзти вперед 20 кроків.*
4. *Намалювати коло радіуса 30 пікселів.*
5. *Підняти перо.*
6. *Встановити зелений колір сліду Черепашки.*
7. *Встановити червоний колір сліду Черепашки, синій колір заливки замкненої фігури.*

Урок 21. Практична робота № 4. «Складання та виконання лінійних алгоритмів»

Див. робочий зошит «Інформатика. 5 клас» / Бондаренко О.О., Ластовецький В.В., Пилипчук О.П., Шестопалов Є.А. – Шепетівка: «Аспект», 2017.

Урок 22. Алгоритми з розгалуженнями

У попередніх уроках ми розглядали алгоритмічну конструкцію розгалуження.

У цьому параграфі ви дізнаєтеся, як виконати на комп’ютері алгоритм з розгалуженням, тобто як запрограмувати комп'ютер на прийняття рішень в залежності від того, чи є певна умова істинною або хибною.

Логічні операції

Крім вже відомих нам числового і рядкового типів даних в Python є логічний тип bool. Цей тип отримав назву на честь англійського математика Джорджа Буля, який створив один розділів математики – математичну логіку. Змінна типу bool може приймати одно з двох значень – True (Істина) або False (Хибність).

Приклад 28.1. Якщо змінній надати значення True, це буде змінна типу bool.

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

Умови в програмі записуються у вигляді логічних виразів. Логічними є вирази, результатом яких є True або False. Простий логічний вираз (проста умова) утворюється за допомогою логічних операцій:

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

Бувають ситуації, коли одночасно треба перевірити виконання декількох умов.

Складена умова – кілька простих умов, з'єднаних логічними операціями *AND (*логічне І, інакше – логічний добуток), *OR (*логічне АБО, інакше – логічна сума), *NOT* (логічне заперечення).

Приклад 28.2. Приклади складених умов:

not a <=3 – *рівнозначне виразу a > 3.*

age >=10 and age <= 18 – *істинне тоді і тільки тоді, коли значення age знаходиться в проміжку від 10 до 18.*

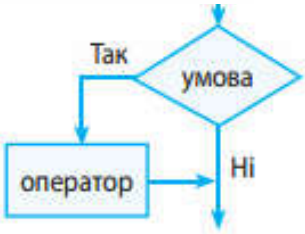
age < 10 or age > 18 – *істинне для всіх значень age, які не належать проміжку від 10 до 18.*

Приклад 28.3. Логічні операції працюють і у вікні консолі. Виконайте цей приклад, щоб ознайомитися з результатами обчислення логічних виразів:

```
>>> books = 10
>>> books == 5 # Перевірка, чи дорівнює books 5
False
>>> books > 1 # Перевірка, чи є books більшим за 1
True
>>> books >=5
True
>>> books != 10 # Перевірка, чи не дорівнює books 10
False
>>> not books ==10 #not змінює значення виразу books==10 на протилежне
False
>>> books ==10 or books == 5 #or перевіряє, чи дорівнює books 10 або 5
True
>>> books ==10 and books == 5 # and перевіряє, чи дорівнює books
                              одночасно 10 і 5
False
```

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».



Синтаксис оператора:
if <умова> :
<оператор
– обов'язковий відступ від лівого краю.

Оператор if перевіряє істинність зазначеної умови. Якщо умова істинна (True), програма виконає дію, зазначену в частині <оператор>. Якщо ж умова хибна (False), то керування передається оператору, що міститься після оператора if.

Приклад 28.4. Якщо виконати цю програму, у вікні консолі отримаємо результат «Вірно».

```
books = 10
if books == 10:
    print('Вірно')
```

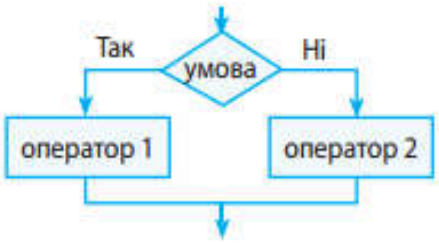
Приклад 28.5. Вітання «Happy Birthday!» виводиться, тільки якщо користувач вводить «y».

```
answer = input('Is it your birthday? (y/n)  ')
if answer =='y':
    print ('Happy Birthday!')
```

```
==== RESTART: C:/Users/Lyceum/Ap
Is it your birthday? (y/n)  y
Happy Birthday!
```

Умовний оператор if...else

Алгоритмічній конструкції «повне розгалуження» відповідає умовний оператор if...else.



Синтаксис оператора:
if <умова> :
<оператор 1>
else
<оператор 2>

Якщо результатом перевірки умови є значення True, то виконується блок дій <оператор 1>. Якщо перевірка умови дала результат False, виконується блок дій <оператор 2>, який міститься після службового слова else.

Приклад 28.6. Якщо виконати цю програму, у вікні консолі отримаємо результат «Невірно».

```
a = 13
if a = = 12:
   print("Вірно")
else:  # Інакше
   print("Невірно")
```

Приклад 28.7. Якщо користувач вводить «y», друкується вітання «Happy Birthday!», інакше текст «Good luck!».

```
answer = input('Is it your birthday? (y/n)  ')
if answer =='y':
    print ('Happy Birthday!')
else:
    print ('Good luck!')
```

Результат виконання програми у вікні консолі:

```
Is it your birthday? (y/n)  n
Good luck!
```

Приклад 28.8. Оператор if реалізує таку дію: якщо a>b, змінні міняються значеннями так, щоб виявилося a≤b.

```
File  Edit  Format  Run  Options  Window  Help
a = 5
b = 2
if a>b :
    c = a
    a = b
    b = c
    print ('a i b помінялися значеннями')
else :
    print ('обміну значеннями не потрібно')
print ('a=', a, ' b=', b)
```

У вікні консолі:

```
a i b помінялися значеннями
a= 2  b= 5
```

Зверніть увагу в прикладі 28.8 на відступи від лівого краю: команди, вкладені в гілки оператора if, були записані на одній вертикалі.

Отже, оператор if є важливим інструментом програмування, який стане вам в нагоді при створенні тестових та ігрових програм.

Питання для самоперевірки:

1. *Як записується і виконується умовний оператор у неповній формі?*
2. *Як записується і виконується умовний оператор у повній формі?*
3. *Як виконуються логічні операції and, or, not?*
4. *Яких значень набудуть змінні a і в після виконання умовних операторів, наведених у таблиці, для указаних початкових значень a і b?*

	a = 3, b = 5	**a**	**b**

1	if a>b :
	  a=0
	else :
	  b=0		

2	if a!= b : a=b		
3	if a % 3 ==0 : a=a // 3		

	a = 3, b = 5	**a**	**b**

4	if a>b :
	  a=a+10
	else :
	  b=b+10		

5	if a<b :
	  a=2*a
	else :
	  b=b*a		

5. *Запишіть у вигляді простої умови:*
   а) x більше 10;
   б) число x не більше числа y;
   в) x – парне число.
6. *Запишіть у вигляді складеної умови:*
   а) $2 < x < 10$;
   б) x не належить проміжку (2,10);

Урок 23. Вкладені розгалуження

У попередньому параграфі ми розглянули алгоритми і програми, що містять одну структуру розгалуження. Але часто в алгоритмі потрібно декілька разів вибирати, які дії виконати. Важливо відзначити, що в алгоритмах і програмах розгалуження можуть бути послідовними (розташованими одне за іншим) і вкладеними (одне всередині іншого). Вкладені розгалуження застосовуються, якщо при виконанні (або невиконанні) деякої умови знову необхідно робити вибір. У такій ситуації в умовному операторі по гілці True (після двокрапки) або по гілці False (після службового слова else) використовують оператор if.

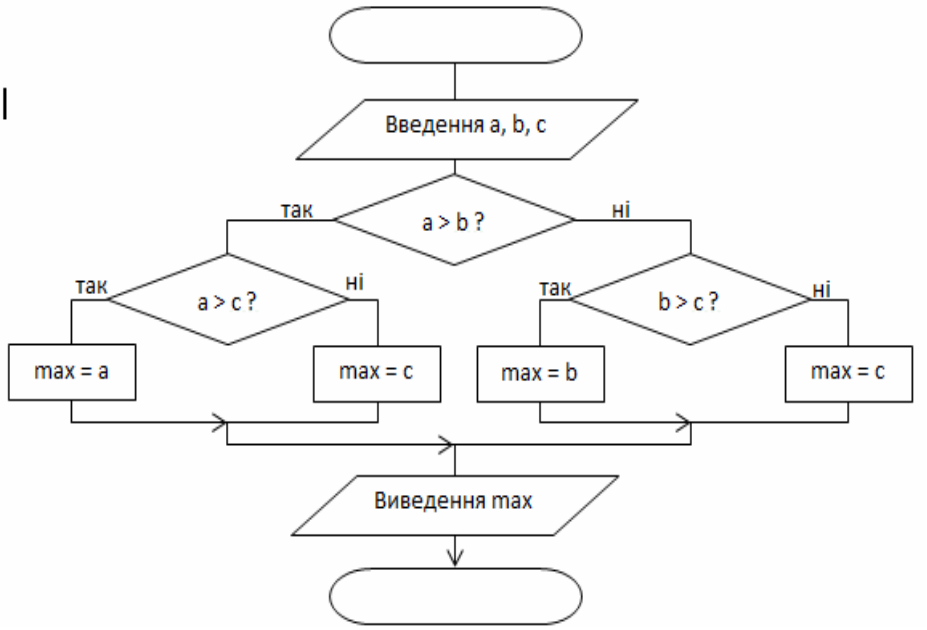
Вкладені розгалуження

Розглянемо задачу: знайти найбільше значення серед трьох чисел: a, b, c. Результат присвоїти змінній max.

Алгоритмічна структура розгалуження має тільки дві гілки, а у нас три варіанти (найбільшим може бути будь-яке з чисел: a, b, c), тому використаємо вкладення розгалужень.

Поміркуємо: якщо a>b, то найбільшим є більше з чисел a і c; інакше найбільшим є більше з чисел b і c. Таким чином, після вибору більшого з чисел a і b знов потрібно вибрати більше з чисел a і c, а по гілці False – вибрати більше з чисел b і c

Блок-схема алгоритму представлена на малюнку:



Запишемо програмний код даного алгоритму:

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

```
=== RESTART: C:/l
a=? 5
b=? 2
c=? 9
max= 9
```

Якщо a=5, b=2, вираз a>b отримує значення True. По гілці True перевіряється умова a>c, яка при c=9 отримує значення False, тому змінна max набуває значення 9.

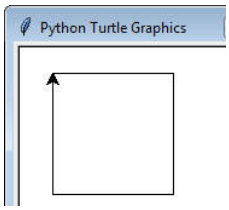
Відступи

В мові Python велике значення мають відступи команди від лівого краю вікна програми. Команди, вкладені в гілки оператора if, об'єднуються в блоки по величині відступів. Відступ може бути будь-яким, головне, щоб в межах одного вкладеного блоку відступ був однаковий.

Приклад 29.1. Програма запитує у користувача кількість кутів у багатокутника. Якщо кількість кутів не дорівнює 3 або 4, видається повідомлення «Помилка в кількості кутів», інакше якщо кількість кутів дорівнює 3, то малюється трикутник, інакше малюється квадрат.

```
from turtle import *
print('Накреслимо багатокутник')
a_str = input('Кількість кутів? (3 або 4)  ')
a = int(a_str)
if not (a == 3 or a == 4):
   print('Помилка в кількості кутів')
else:
   if a == 3:
```

У вікні Python Turtle Grafics:



```
    forward(100)
    right(120)
    forward(100)
    right(120)
    forward(100)
else:
    forward(100)
    right(90)
    forward(100)
    right(90)
    forward(100)
    right(90)
    forward(100)
```

У вікні консолі:

```
Накреслимо багатокутник
Кількість кутів? (3 або 4)  4
```

Якщо введено значення 5, видається повідомлення «Помилка в кількості кутів»:

```
Накреслимо багатокутник
Кількість кутів? (3 або 4)  5
помилка в кількості кутів
```

Отже, вкладені розгалуження використовуються в тих випадках, коли потрібно перевірити послідовно дві або більше умови. Під час запису програмного коду уважно слідкуйте за відступами, щоб команди, вкладені в гілки оператора if, були записані на одній вертикалі. По-перше, цього вимагає синтаксис мови Python, а по-друге, це робить текст програми більш наочним і зрозумілим.

Питання для самоперевірки

1. *Поясніть схему виконання оператора if, в якому застосовані вкладені оператори розгалуження.*
2. *Яке значення мають відступи команди від лівого краю вікна програми?*
3. *Яких значень набудуть змінні a і b після виконання умовного оператора if для наведених наборів початкових значень?*

Фрагмент програмного коду	*Початкові значення a, b*	*A*	*b*
if a<3: *if a>b :* *a =b* *else :* *b =a* *else :* *if a > b :* *b = a* *else :* *a = b*	*a= 1, b = 5*		
	a = 3, b = 5		
	a = 8, b = 5		
	a= 1, b = 2		

4. *Напишіть програмний код, який за допомогою циклу while 10 разів виводить на екран слово 'Hello'.*
5. *Напишіть програму, яка для двох чисел a і b виводить відповідь «a>b», «b>a» або «a=b».*
6. *Напишіть програму, яка запитує вік школяра і визначає, в якій ланці він навчається: початкова школа (1-4 класи), основна (5-9 класи) або старша (10-11 класи).*

Урок 24. Практична робота №6 «Алгоритми з розгалуженнями»

Див. робочий зошит «Інформатика. 5 клас» / Бондаренко О.О., Ластовецький В.В., Пилипчук О.П., Шестопалов Є.А. – Шепетівка: «Аспект», 2017.

Урок 25. Алгоритми з повтореннями. Цикл for

Повторення (цикл) — це алгоритмічна структура, за допомогою якої та сама послідовність дій виконується кілька разів. Для запису алгоритмів із повторенням (циклів) мовою Python використовують 2 види операторів циклу: з параметром та з умовою. Серію команд, що повторюється під час виконання циклу, називають **тілом циклу**. Кожне виконання тіла циклу називають **ітерацією**.

Повторення команд

Цикл **for** повторює блок команд (тіло циклу) задану кількість разів, позбавляючи необхідності декілька разів писати одні і ті самі команди.

Синтаксис, або правила запису циклу for, виглядає наступним чином:

for x in range(n) :
    <тіло циклу>

В першу чергу ми вказуємо ключове слово **for**, після чого указуємо змінну **x**, яка буде лічильником циклу. Ключове слово **in** наказує Python по черзі надати змінній x всі значення в діапазоні від 0 до n-1. Не забувайте, що комп'ютер зазвичай починає рахувати з 0, а не з 1, як люди. Вбудована функція **range** повертає безперервну зростаючу послідовність цілих чисел, які можна використовувати в якості індексів всередині циклу.

Щоб дати зрозуміти комп'ютеру, які команди слід повторити, використовуються відступи. Можна зробити відступ для кожної повторюваної в тілі циклу команди, натискаючи для цього клавішу Tab у вікні нового файлу.

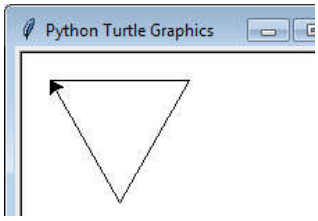
Приклад 31.1. Проаналізуйте зміну значень лічильника x в циклі for. Функція range(4) задає список значень [0, 1, 2, 3].

```
for x in range(4) :
        print (x)
```

```
=== RESTART:
0
1
2
3
```

Приклад 31.2. В прикладі 26.1 для малювання трикутника ми тричі, для кожної сторони трикутника, давали Черепашці команди forward(100) і right(120). Замість цього можна повторити ці команди в циклі.

```
from turtle import *
for x in range(3) :
    forward(100)
    right(120)
```



<u>Лічильник циклу for</u>

Змінна-лічильник циклу веде відлік, скільки разів повторився цикл, почергово набуваючи всіх значень з указаного діапазону.

Приклад 31.3. Діапазон значень може бути заданий у вигляді текстового рядка. Лічильник почергово приймає значення літер, з яких складається указаний рядок:

```
a = 'Рядок'
for x in a:
    print('->', x)
```

```
=== RESTART
-> Р
-> я
-> д
-> о
-> к
```

Крім рядків або числових значень змінна може містити списки. **Список** – це набір значень, розділених комами, взятий у квадратні дужки. У списках можна зберігати значення будь-якого типу, будь то числа або рядки.

Приклад 31.4. Діапазон значень може бути заданий у вигляді списку значень, взятого в квадратні дужки. Лічильник почергово приймає значення зі списку:

```
b = [2, 4, 6, 8, 10]
for x in b:
    print('->', x)
```

```
=== RESTART:
-> 2
-> 4
-> 6
-> 8
-> 10
```

Якщо список значень лічильника достатньо великий, і числа в списку змінюються з певним кроком, можна застосувати функцію range з такими параметрами:

range (від, до, крок)

Відлік починається зі значення «від» і завершується на 1 раніше за значення «до».

Приклад 31.5. Функція range (2,11,3) надає змінній x значення від 2 до (11-1) з кроком 3, тому, хоча 8+3=11, значення 11 вже до діапазону значень не входить.

```
for x in range (2,11,3):
    print('->', x)
```

```
=== RESTART:
-> 2
-> 5
-> 8
```

Приклад 31.6. Якщо потрібно вести зворотній відлік (в бік зменшення значень), укажіть значення «від» більшим, ніж значення «до», а «крок» повинний бути від'ємним:

```
for x in range (5,1,-1):
    print('->', x)
```

```
=== RESTART:
-> 5
-> 4
-> 3
-> 2
```

<u>Використання циклу for для створення малюнків</u>

Проаналізуємо виконання програми:

```
from turtle import *
for x in range(1,100,2) :
   forward(x)
   left(90)
```

Рис.31.1

При кожному проході циклу Черепашка малює лінію довжиною x пікселів і повертається вліво на 90° (рис. 31.1). Ви знаєте з курсу математики, що навколо точки можна виконати повний поворот на 360°. Таким чином, за 4 ітерації циклу Черепашка повертається до початкового напрямку (4*90°=360°). Програма малює спіраль, тому що кожний відрізок на 2 пікселі довший за попередній: перший відрізок має довжину 1 піксель, другий – 3, останній (50-й) – 99.

Рис. 31.2

Поворот на 90° створює квадратну спіраль. Змінимо кут повороту на 2°: left(92) . Зміна всього лише 1 числа призводить до значних змін у вигляді спіралі (рис.31.2).

Змінимо команду forward(x) на circle(x), яка малює коло радіусу x з поточної позиції.

```
for x in range(1,100,2) :
   circle(x)
   left(90)
```

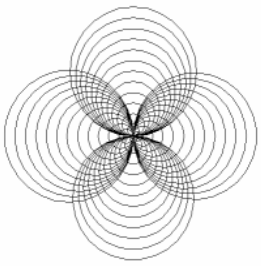


Рис. 31.3

Отримаємо 4 набори кіл, тому що після малювання кожного кола Черепашка повертається вліво на 90° (рис.31.3). Щоб намалювати n кіл, потрібно повертати Черепашку на 360°/n градусів. Внесемо такі зміни до програми, щоб Черепашка малювала 6 наборів кіл (360°/6=60°, рис. 31.4):

```
for x in range(1,100,2) :
   circle(x)
   left(60)
```

Як зробити наші спіралі різнокольоровими?

Рис. 31.4

Для цього нам потрібен список кольорів, а не один колір, тому ми створимо змінну-список з ім'ям colors і помістимо в цей список чотири кольори:

colors = ['red','yellow','green','blue']

Зверніть увагу, що ми помістили список кольорів у квадратні дужки. Звернутися до значення зі списку з номером x можна як до colors[x]. Наприклад, colors[0] ='red', colors[3] ='blue'.

Складне питання: як змусити черепашку при кожній ітерації циклу, коли x змінюється від 1 до 99, обирати одне з 4-х значень списку? Використаємо операцію % (остача від ділення). При обчисленні виразу x % 4 ми можемо отримати 4 значення остачі (0, 1, 2, 3), що відповідає номерам кольорів в списку colors. Повернемось до програми малювання квадратної спіралі, але крок циклу зробимо рівним 1 (рис.31.5):

```
from turtle import *
colors = ['red','yellow','green','blue']
for x in range(100) :
    color(colors[x% 4])
    forward(x)
    left(90)
```

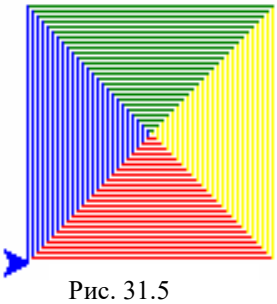
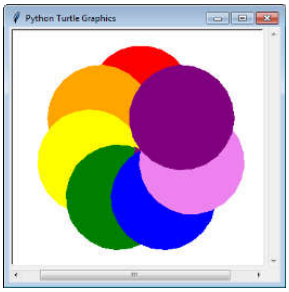


Рис. 31.5

Таким чином, вносячи нескладні зміни до операторів малювання в тілі циклу for, ми можемо отримувати складні цікаві зображення.

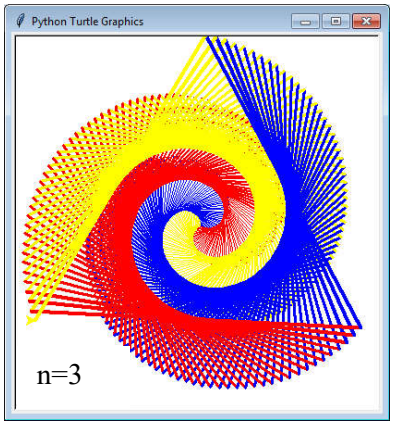
Приклад 31.7. Намалюємо квітку – семибарвицю. Змінна colors містить список кольорів.

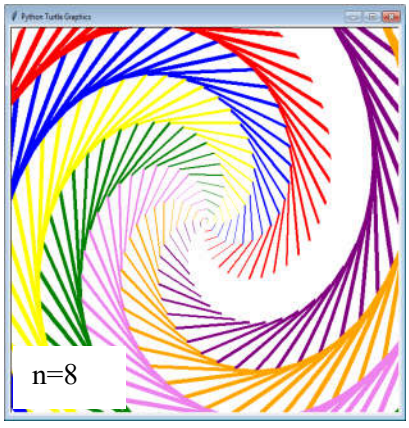
```
from turtle import *
colors = ['red','orange','yellow','green','blue','violet','purple']
n=7
for x in range(n) :
   color(colors[x],colors[x])
   begin_fill()
   circle (70)
   left(360/n)
  end_fill()
```



Приклад 31.8. Програма малювання кольорової спіралі. Кількість сторін у спіралі вводиться користувачем.

```
from turtle import *
n=int(input('n=? (1-8) ')
colors=['red', 'blue', 'yellow', 'green', 'violet', 'orange', 'purple', 'white']
for x in range(360) :
    color(colors[x % n] )
    forward(x) # довжина відрізка дорівнює значенню лічильника
    left (360/n+1)
    width(x*n/300) # ширина ліній збільшується по мірі розширення спіралі
```





Питання для самоперевірки

1. *Поясніть правила виконання циклу for.*
2. *Яких значень набуває змінна x в ході виконання циклу*
   for x in range(5) ?
3. *Яких значень набуває змінна x в ході виконання циклу*
   for x in range(0, 10, 2) ?
4. *Що буде надруковано в ході виконання циклу*
   s = 'цикл'
   for x in s :
   print (x)
5. *Що буде надруковано в ході виконання циклу*
   b = [1,3,5,7,9]
   for x in b :
   print (x)
6. *Чому дорівнює s після виконання циклу:*

1	2	3
s = 0 for a in range(5, 7, 1) s = s + 1	s = 0 for a in range(5, 7, 1) s = s + a	s = 0 for a in range(10, 5, -1) s = s + 1

Урок 26. Алгоритми з повтореннями. Цикл while

Цикл for зручно використовувати, якщо відома кількість повторень. Але часто цикл потрібно повторювати, поки виконується деяка умова. Цикл while буде повторюватися до тих пір, поки це потрібно.

Як працює цикл while

Цикл While (*«Поки»*) буде повторюватися, поки виконується задана умова. Ця умова називається умовою циклу і повертає True або False. Синтаксис оператора:

while <умова> :
<тіло циклу>

Тут <умова> — логічний вираз, що є умовою виконання циклу; <оператор> — простий або складений оператор, який виконується при кожній ітерації. Виконання оператора циклу while починається з обчислення значення логічного виразу — умови циклу (див. блок-схему). Якщо умова істинна, то виконуються оператори тіла циклу і керування повертається на перевірку умови. Якщо ж умова хибна, то виконується оператор, який є наступним після оператора while. Якщо при першій перевірці умова виявиться хибною, тіло циклу не виконається жодного разу.

Приклад 32.1. Для початкового значення x=7 цикл while x<10 виконається 3 рази:

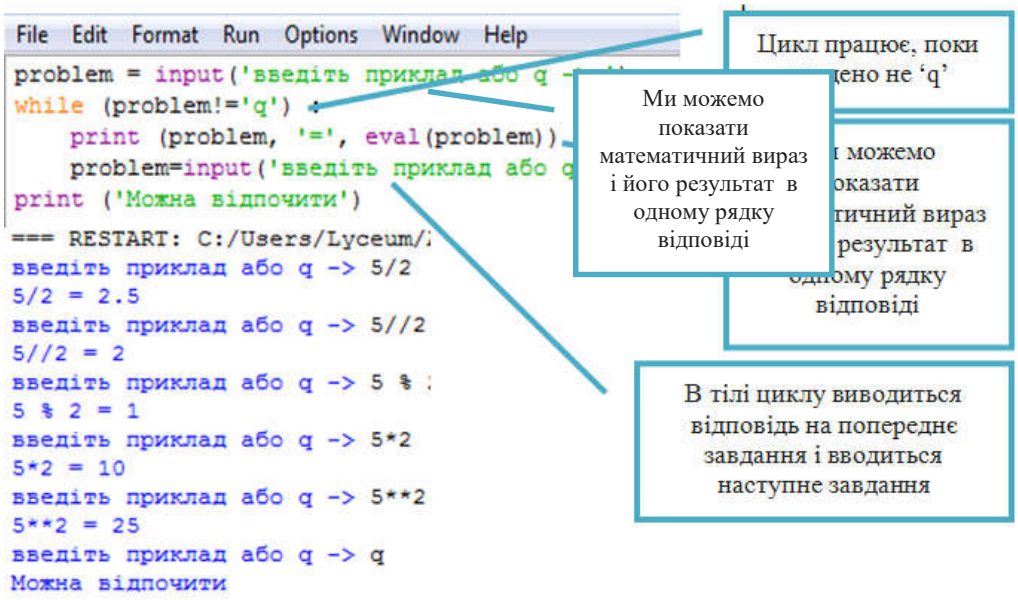
```
File  Edit  Format  Run
x=7
while x<10 :
    x=x+1
    print (x)
```

```
=== RESTART:
8
9
10
```

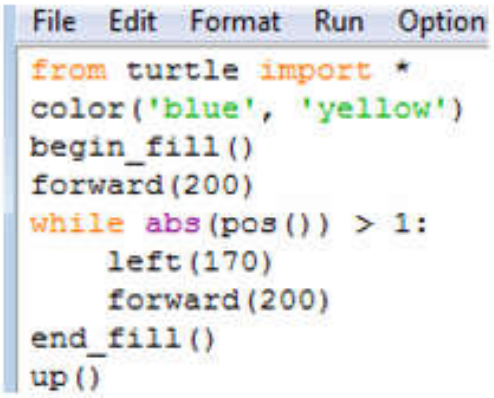
Приклад 32.2. Напишемо програму, в якій користувач вводить з клавіатури математичний вираз і отримує відповідь. Для обробки математичного виразу і обчислення результату використовується функція eval (), яка обробляє рядок клавіатурних символів точно так же, як і оболонка Python IDLE.

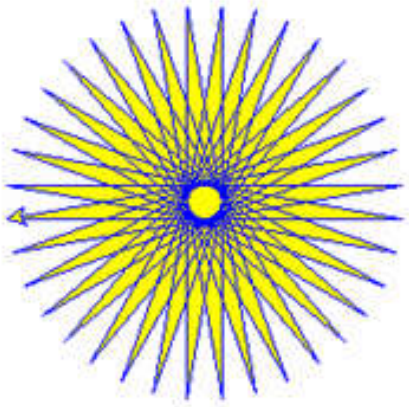
Таким чином, коли ми вводимо приклад в якості вхідних даних, функція eval () може дати нам відповідь на завдання.

Цикл while працює, поки користувач не введе ‘q’



Приклад 32.3. В циклі продовжується побудова відрізків і поворот Черепашки на 170° поки Черепашка не повернеться в точку (0,0) (abs(pos()) < 1).





Нескінченний цикл

Приклад 32.4. Наведений фрагмент програми ілюструє нескінченний цикл.

```
num = 0
while num < 20:
print (num)
```

У тілі циклу значення num не змінюється, тому умова num<20 завжди правильна і поданий цикл є нескінченним.

Якщо в якості умови циклу while задати True, умова ніколи не стане хибною і цикл не завершиться.

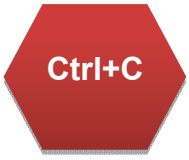
Приклад 32.5. Програма випробує витримку користувача – у нескінченному циклі пропонує користувачеві розв'язати задачу:

```
File  Edit  Format  Run  Options
from random import randint
while True :
    a = randint(1,10)
    b = randint(1,10)
    print (a,'+',b)
    x = int(input('=?'))
    if a+b==x :
        print ('Так')
    else :
        print ('Hi')
```

Змінні a і b набувають випадкових значень в межах від 1 до 10

```
=== RESTART:
5 + 4
=?9
Так
4 + 5
=?3
Hi
9 + 9
=?
```

Можна зупинити виконання нескінченного циклу у вікні консолі IDLE. Для цього у вікні консолі натисніть сполучення клавіш Ctrl+C. Після цього IDLE буде відправлений запит на зупинку програми. Можливо, доведеться натиснути Ctrl+C декілька разів, щоб IDLE зреагувала на запит.



Питання для самоперевірки:

7. Поясніть структуру і правила виконання циклу з умовою.

8. У чому відмінність у використанні циклу з параметром і циклу з умовою?

9. У якому випадку цикл While не виконається жодного разу?

10. У якому випадку виникає «нескінченний цикл»?

11. Назвіть початкові значення змінних; умову продовження циклу; оператори тіла циклу.

а) a=10
```
while a!=0
a = a – 1
```

б)
```
x = 1
s = 0
while x<=5 :
    x = x + 2
    s = s + x
```

12. Дано фрагмент програмного коду. Заповніть таблицю:

Програмний код	Початкове значення X	Після виконання циклу	Скільки повторень відбулося
while x<=10 : x = x + 1	7		
	10		
	11		

Урок 27. Практична робота №7 «Алгоритми з повтореннями»

Див. робочий зошит «Інформатика. 5 клас» / Бондаренко О.О., Ластовецький В.В., Пилипчук О.П., Шестопалов Є.А. – Шепетівка: «Аспект», 2017.