С. І. Ганжела

С. О. Шлянчак

ОСНОВИ ІНФОРМАТИКИ З ЕЛЕМЕНТАМИ ПРОГРАМУВАННЯ ТА СУЧАСНІ ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ НАВЧАННЯ

Частина ІІ

Елементи програмування

Кропивницький – 2017

ББК 32.973я73
УДК 6Ф7(075)
Г19

Ганжела, С. І. Основи інформатики з елементами програмування та сучасні інформаційні технології навчання. Ч. ІІ. Елементи програмування / С. І. Ганжела, С. О. Шлянчак. – Кропивницький: РВВ ЦДПУ ім. В. Винниченка, 2017. – 61 с.

Рецензенти:

Кушнір В. А. - доктор педагогічних наук, професор кафедри математики, завідувач кафедри математики

Радченко Ю. Л. - кандидат педагогічних наук, старший науковий співробітник відділу андрагогіки ІПООД НАПН України

У посібнику представлено загальні відомості з програмування: основи алгоритмізації, мова програмування та засоби створення програм, середовище *PascalABC.Net*, програма *Сходинки до інформатики*, середовище *Scratch*. Посібник складається з двох частин: «Теоретична частина» і «Лабораторні роботи». Посібник містить приклади, рекомендації виконання завдань та примітки у вигляді порад для їхнього раціонального виконання.

Навчальний посібник призначений для супроводу курсу «Основи інформатики з елементами програмування та сучасні інформаційні технології навчання», який передбачено навчальним планом для студентів напрямів підготовки: «Початкова освіта», «Дошкільна освіта». Також посібник адресовано педагогічним працівникам і студентам інших напрямів підготовки.

Рекомендовано до друку методичною радою Кіровоградського державного педагогічного університету імені Володимира Винниченка (протокол № 4 від 22 березня 2017 року).

Передмова

На сьогоднішній день в Україні відбувається реформування освіти, що є частиною процесу оновлення освітніх систем, які проходять останні кілька років у європейських країнах. При цьому початкова освіта знаходиться на передових засадах оновлення національної освіти, коли зростає роль умінь добувати, переробляти інформацію, одержану з різних джерел, застосовувати її для індивідуального розвитку та самовдосконалення людини. Потрібно змінити застарілий зміст освіти, що дасть можливість перейти до високотехнологічного інформаційного суспільства, в якому якість людського потенціалу, рівень освіченості та культури всього населення набувають особливого значення для економічного й соціального поступу країни.

На жаль, підготовка майбутніх вчителів початкової школи з основ алгоритмізації і програмування знаходиться на дуже низькому рівні. На це є цілий ряд причин. Не розкриваючи всі ці причини потрібно зазначити, що однією з головних є недостатня кількість годин, що відводилися відповідним темам у шкільній програмі. Все це спонукало авторів до написання даного навчального посібника. Матеріал, який дібраний у книзі, пройшов випробування часом і використовувався, певним чином видозмінюючись, у навчальному процесі протягом трьох останніх років.

Навчальний посібник дозволить читачу самостійно, швидко й ефективно навчитися створювати алгоритми, записувати їх у вигляді блок-схем, навчальною алгоритмічною мовою і мовою програмування *Pascal*, а також використовувати програму *Сходинки до інформатики* +, середовище об'єктно-орієнтованого візуального програмування *Scratch*, що призначене для створення комп'ютерних анімацій, мультимедійних презентацій, анімаційних та інтерактивних історій, ігор, моделей.

Друга частина навчального посібника містить вісім лабораторних робіт з курсу «Елементи програмування». Для ефективного засвоєння даного курсу перед виконанням кожної лабораторної роботи потрібно прочитати теоретичну частину до даного розділу і виконати наведені приклади. Для того, щоб дати глибоку відповідь на деякі питання, потрібно скористатися додатковою літературою, список якої наведено у кінці навчального посібника.

У теоретичному матеріалі є основні поняття й означення з усіх тем другої частини навчального посібника і правила-орієнтири, які спрощують роботу студенів з програмним забезпеченням. Також у книзі можна знайти багато корисних порад, які допоможуть при виконанні лабораторних робіт і знаходженню відповідей на контрольні питання. Особливу увагу слід приділити пунктам, що позначені як *Примітка* з наведеними ідеями щодо виконання завдань.

Навчальний посібник орієнтований на майбутнього вчителя молодших класів, а також може бути використаний учнями школи, студентами інших спеціальностей для здійснення перших кроків у програмуванні.

ЗМІСТ

РОЗДІЛ ІІ. ЕЛЕМЕНТИ ПРОГРАМУВАННЯ
ТЕОРЕТИЧНА ЧАСТИНА ДО РОЗДІЛУ «ЕЛЕМЕНТИ ПРОГРАМУВАННЯ»

Основи алгоритмізації

Алгоритми та їх властивості. Форми подання алгоритмів

Досвід практичної діяльності людини дозволив алгоритмічним способом розв'язувати складні задачі. Так, процес розв'язування задачі поділяється на кілька етапів, кожному з яких відповідає певна дія. Якщо виконавець зазначеного процесу відомий, то всі поетапні дії можна подати у вигляді конкретних команд. Виконавцем алгоритму може бути людина, комп'ютер, робот інше. Для кожного виконавця є команди, які він може виконати, і команди, які він виконати не може. Тому виконавець обов'язково повинен правильно розуміти відповідні команди в прямому чи переносному значенні. У такому разі кажуть, що розв'язування задачі зводиться до виконання певного алгоритму.

Саме слово «алгоритм» походить від лат. «algorithmi», що є формою написання імені великого математика IX століття Аль-Хорезмі, який сформулював правила виконання арифметичних дій. Тому спочатку під алгоритмом розуміли тільки правила виконання чотирьох арифметичних дій над багатоцифровими числами в десятковій системі числення. В подальшому поняття «алгоритм» стали використовувати для позначення послідовності дій, що приводять до розв'язання задачі. Перед складанням алгоритму повинні бути чітко визначені початкові умови й те, що має бути одержано.

Алгоритм – це чітка, скінченна послідовність дій (команд) із строго визначеним порядком виконання, після реалізації яких досягається поставлена мета. Також *алгоритмом* називають зрозуміле і точне розпорядження виконавцю виконати послідовність дій, спрямованих на розв'язання поставленої задачі. Тобто, це певна інструкція для виконавця, що може бути задана різними способами (словами, формулами, послідовністю обчислювальних операцій, логічних дій тощо). Але не кожна інструкція може бути алгоритмом. Наприклад, дитина може виконати команди: подивитися на дошку, відкрити зошит, але не може – стрибнути 10 метрів.

Властивості алгоритмів

1. *Масовість (універсальність).* Алгоритм повинен передбачати можливість його застосування для розв'язання цілого класу однотипних задач, що відповідають загальній постановці задачі.
2. *Визначеність (детермінованість, однозначність).* Кожна команда алгоритму однозначно визначає дії виконавця і не допускає двоякого тлумачення. Порядок виконання команд є строго визначеним.
3. *Дискретність.* Процес виконання алгоритму повинен складатися з послідовності окремих кроків (команд або дій), кожен з яких виконується за скінченний інтервал часу. Тільки закінчивши виконання однієї команди, виконавець переходить до виконання іншої.

4. *Результативність.* Виконання алгоритму повинно приводити до цілком конкретного результату, що відповідає поставленій меті. Виконання алгоритму не може закінчуватися невизначеною ситуацією.
5. *Формальність.* Будь-який виконавець, здатний сприймати та виконувати вказівки алгоритму (навіть не розуміючи їх змісту). Тобто, може виконати алгоритм, не вникаючи в суть задачі.
6. *Скінченність.* Виконання алгоритму завершиться після скінченної кількості кроків і за скінченний час при будь-яких допустимих початкових даних. Також зустрічається визначення алгоритму через його властивості, як скінченна однозначно визначена послідовність операцій, формальне виконання якої приводить до розв'язання певної задачі за скінченну кількість кроків.

Досконалим виконавцем алгоритмів обробки інформації є комп'ютер, робота якого здійснюється під управлінням програми. *Програма* – це реалізація алгоритму мовою «зрозумілою» комп'ютеру.

Кожний тип комп'ютера має свій власний набір операцій, із яких можна складати програми. Це, так званий, набір машинних команд комп'ютера, що визначає обмеження на розробку програм. Тому процес створення програм є творчим, адже потрібно «навчити» комп'ютер розв'язувати різноманітні (як правило, складні) задачі, використовуючи обмежений набір простих операцій (команд). Разом з тим набір операцій кожного із сучасних комп'ютерів є алгоритмічно повним. Тому з простих комп'ютерних операцій (як із цеглинок) можна складати алгоритми для розв'язування задач будь-якої складності.

Алгоритм описується зрозумілою виконавцю мовою. Оскільки комп'ютер виконує операції з багаторозрядними числами у двійковій системі числення, то мовою комп'ютера є послідовність кодів із нулів і одиниць. Використання такої комп'ютерної мови для складання програм є дуже неефективним. Адже записи програм таким способом виявляються дуже складними і громіздкими, а процес їхньої розробки – достатньо трудомістким. Тому для створення програм використовуються спеціальні мови – *мови програмування*. Мова програмування дозволяє записувати команди в такій формі, щоб їх можна було замінити на машинні коди. Це перетворення автоматично здійснюється за допомогою спеціальних програм-перекладачів, які називають *компіляторами*. Компілятори є складовою частиною системного програмного забезпечення комп'ютера. Чим досконалішими стають комп'ютери, тим більше мови програмування за своїми властивостями наближаються до природної мови людини.

Кожна мова (алгоритмічна, програмування) має такі *компоненти*:

1) **алфавіт** – множина символів, за допомогою яких можна утворити слова та речення цієї мови;
2) **словник** – набір службових (зарезервованих) слів, за допомогою яких можна записати будь-яку команду;
3) **синтаксис** – сукупність правил запису алгоритмів або програм на алгоритмічній мові або мові програмування;

4) **семантика** – правила тлумачення конструкцій, записаних алгоритмічною мовою або мовою програмування.

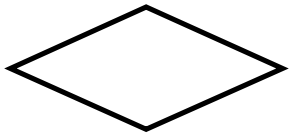
Існує кілька способів запису алгоритму, вибір яких залежить від виконавця та того, хто його задає.

Різні форми подання алгоритму

- *Словесна* форма (алгоритм записується як послідовність занумерованих словесних команд, найчастіше використовується в людському спілкуванні, в інструкціях користувачам програмних засобів або побутових приладів тощо).
- Подання у вигляді формул, таблиць, схем, малюнків тощо: запис алгоритму у вигляді *формул* (з формул випливає порядок здійснення обчислень для отримання числового результату; *таблична* форма (якщо виконується серія розрахунків за однаковими формулами); у вигляді *малюнків* (наприклад, малюнки з правилами поведінки на дорозі); у вигляді *схем* (умовні позначення на купленому товарі щодо його користування, наприклад, заварювання чаю, прання білизни інше).
- *Блок-схема* складається з окремих геометричних фігур (блоків), які з'єднуються напрямленими лініями, що показують послідовність переходу від одного блоку до іншого.
- *Навчальна алгоритмічна мова (псевдокод).*

Табл. 2. 1.

Основні елементи блок-схеми алгоритму

Назва блоку	Зображення	Опис дії
Термінатор		Позначає початок або кінець алгоритму
Процес		Позначає команду, яку треба виконати
Дані		Позначає введення вхідних даних і виведення вихідних даних (результатів)
Рішення		Позначає перевірку значення логічного виразу деякої умови

Спосіб запису алгоритмів у вигляді блок-схеми можна розглядати як певну алгоритмічну мову – систему позначень і правил для однотипного запису алгоритмів та їх виконання.

Навчальна алгоритмічна мова

Навчальною алгоритмічною мовою (НАМ) називають алгоритмічну мову, що створена для вивчення правил і прийомів складання алгоритмів.

Службовими словами навчальної алгоритмічної мови є слова або їхні скорочення українською мовою. Це спрощує як читання і розуміння алгоритмів, написаних НАМ, так і складання алгоритмів засобами НАМ. Важливою складовою НАМ є те, що її службові слова, конструкції і правила запису алгоритму подібні до тих, які є характерними для поширених мов програмування. Завдяки такій схожості оволодіння НАМ є кроком до реального програмування.

Структура алгоритму, описаного НАМ

АЛГ ім'я_алгоритму (список параметрів із вказанням їх типів)
АРГ список аргументів
РЕЗ список результатів
} Заголовок алгоритму

ПОЧ список допоміжних параметрів із вказанням їх типів
Дії
} Тіло алгоритму

КІН

Ідентифікатор – лексична одиниця, яка використовується як ім'я елементу мови програмування. Ідентифікатори використовують для позначення констант, змінних, процедур, функцій, файлів та програм.

Правила запису ідентифікаторів:

1) ідентифікатор може складатися з букв, цифр і знаку підкреслення;
2) ідентифікатор починається з букви або знаку підкреслення;
3) максимальна довжина ідентифікатора 127 символів, але комп'ютер розрізняє лише перші 63 символи;
4) між двома ідентифікаторами повинен бути хоча б один пропуск;
5) написання ідентифікатора не повинно співпадати з написанням службових слів;
6) великі і малі літери в ідентифікаторах не розрізняються (для мови Паскаль).

Для опису даних використовують різні типи, основними є: цілий, дійсний, символьний.

Оператори введення, виведення, присвоювання

Синтаксис вказівки введення: запит (список ідентифікаторів)
Синтаксис вказівки виведення: вивід (список ідентифікаторів)
Вказівка присвоювання: ідентифікатор:=вираз

Оператор присвоювання **:=** працює наступним чином: спочатку обчислюється значення виразу у правій частині, а потім це значення надається змінній, ідентифікатор якої записаний у лівій частині.

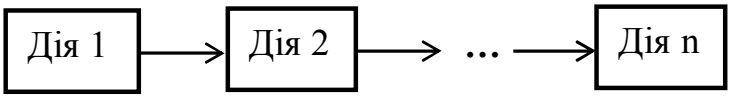
Керувати інформацією, що виводиться, можна таким способом: у список ідентифікаторів можуть входити текстові константи, які НАМ будемо позначати у прямокутній рамці. Текстові константи виводяться на екран без змін.

Базові алгоритмічні конструкції

Будь-який алгоритм і програму можна описати з використанням вказівок лише трьох типів:

1) вказівки про безумовне виконання деяких операцій;
2) вказівки про розгалуження;
3) вказівки про повторне виконання деяких операцій (цикли).

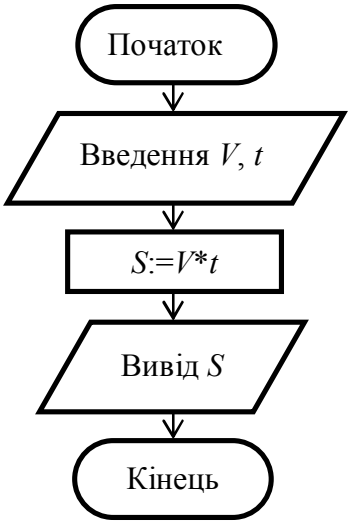
Вказівки про безумовне виконання деяких операцій називають *лінійними алгоритмами*.



Лінійний алгоритм – алгоритм, в якому всі дії (вказівки) виконуються одна за іншою у порядку їх слідування, при цьому ніякі дії не пропускаються і не повторюються. Дії в таких алгоритмах виконуються послідовно, одна за одною, лінійно.

Задача 1.

Скласти блок-схему і написати алгоритм НАМ для знаходження довжини шляху, пройденого пішоходом, якщо відома його швидкість і час, витрачений на дорогу.




```
АЛГ Шлях (V, t, S: дійсні)
АРГ V, t
РЕЗ S
ПОЧ
      Вивід(V=)
      Запит(V)
      Вивід(t=)
      Запит(t)
      S:=V*t
      Вивід(S=, S)
КІН
```

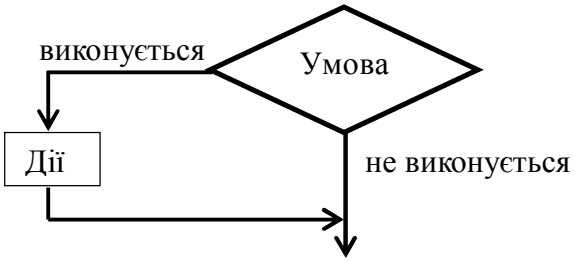
Розгалуження (вказівки про розгалуження)

Розгалуження – це форма організації дій, послідовність яких здійснюється у разі виконання або невиконання заданої умови.

Умова – це будь-яке твердження, яке виконується або не виконується, тобто можна отримати одну з двох відповідей: «так» або «ні».

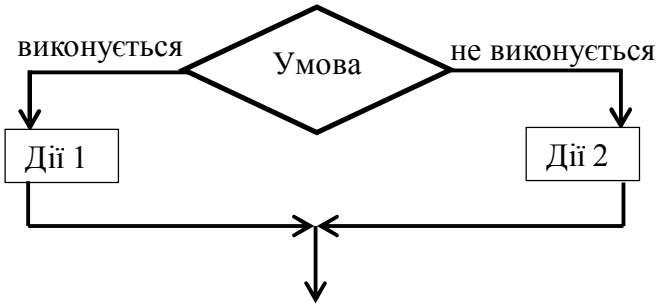
В алгоритмічній мові для таких випадків є умовний оператор, який має дві форми розгалуження: *повне* і *неповне*.

Неповне розгалуження. Якщо *Умова* виконується, то виконуються *Дії*, а якщо *Умова* не виконується, то *Дії* не виконуються. Після виконання або невиконання *Дій* виконавець переходить до наступної команди, що записана після вказівки розгалуження**.**



ЯКЩО <Умова>
ТО
Дії
ВСЕ

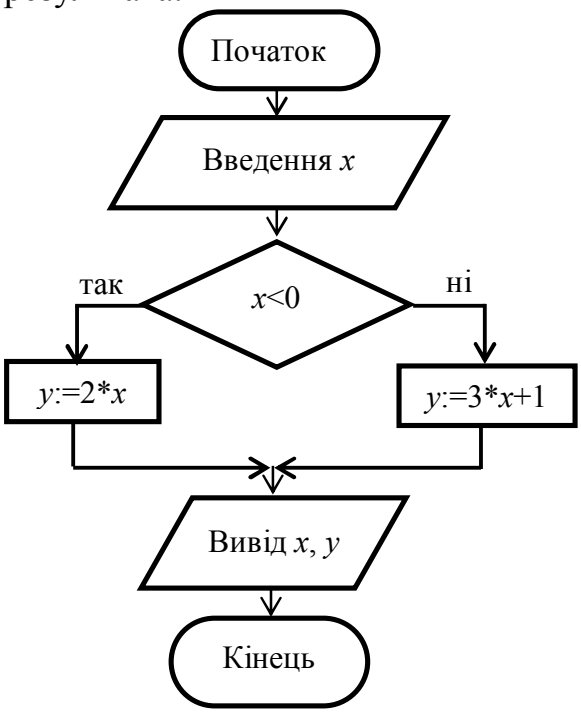
Повне розгалуження. Якщо *Умова* виконується, то виконуються *Дії 1*, а якщо *Умова* не виконується, то виконуються *Дії 2*. Після виконання серії *Дій* виконавець переходить до наступної команди, що записана після вказівки розгалуження**.**



ЯКЩО <Умова>
ТО
Дії 1
ІНАКШЕ
Дії 2
ВСЕ

Задача 2.

Скласти блок-схему і написати алгоритм НАМ для обчислення значення функції: $y = \begin{cases} 2x, & якщо \quad x < 0; \\ 3x + 1, & якщо\, x \geq 0. \end{cases}$ Вивести на екран значення аргумента і результата.



АЛГ функція_1 (*X*,*Y*: дійсні)
АРГ X
РЕЗ Y
ПОЧ
ВИВІД (X=)
ЗАПИТ (*X*)
ЯКЩО *X*<0
ТО
Y:=2*X*
ІНАКШЕ
Y:=3*X*+1
ВСЕ

ВИВІД (X=, *X*, Y=, *Y*)

КІН

Вказівки про повторне виконання деяких операцій (цикли)

Часто зустрічаються такі задачі, при виконанні яких потрібно виконувати одні і ті самі дії кілька разів. Тоді кажуть, що така структура команд називається циклічною або утворена структура «повторення».

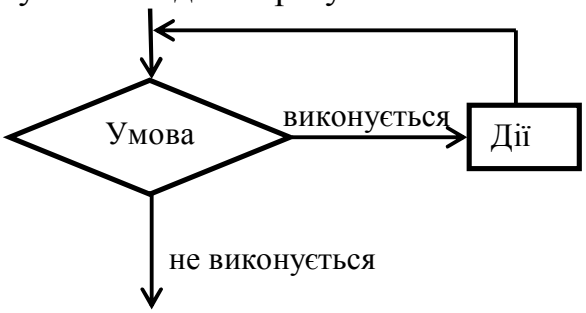
Цикл – це форма організації дій, за якою одна і та сама послідовність дій виконується кілька разів доти, поки виконується (або не виконується) певна умова. Серія команд, що виконується кілька разів без змін при кожному проході циклу, називається *тілом циклу*.

Є два типи повторень: *з передумовою* і *післяумовою (постумовою)*.

Цикл-ПОКИ (*оператор циклу з передумовою*). Виконання циклу починається з того, що перевіряється *Умова*. Якщо вона істина, то виконуються *Дії* і знову перевіряється *Умова*. Якщо вона істина, усе повторюється.

Якщо ж при черговій перевірці *Умова* стає хибною, *Дії* не виконуються, і взагалі виконання оператора циклу на цьому закінчується.

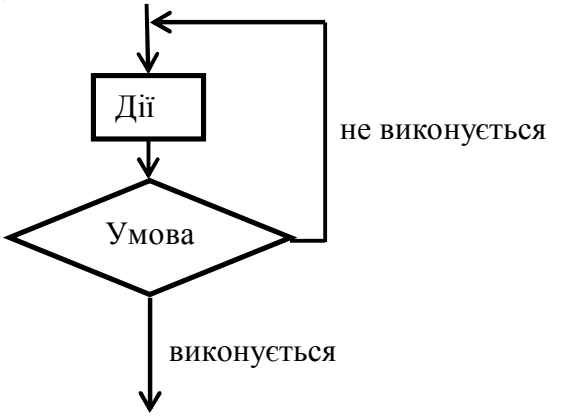
Зокрема, якщо при першій перевірці *Умова* хибна, то тіло циклу не виконується жодного разу.



ПОКИ <Умова>
ПЦ
    Дії
КЦ

Цикл-ДО (*оператор циклу з постумовою*). При використанні вказівки повторення з післяумовою спочатку виконуються *Дії*, потім перевіряється *Умова*, і якщо вона хибна, тобто не виконується, то знову виконуються *Дії* і т.д.

Виконання циклу завершується після того, як при перевірці *Умови* одержано значення «істина», тобто вона виконується. Таким чином, істинність умови означає завершення, а не продовження виконання циклу, на відміну від циклу-ПОКИ.



ПОВТОРЮВАТИ
ПЦ
    Дії
КЦ
ДО <Умова>

Ще одна відмінність полягає у тому, що *Дії* при використанні циклу-ДО обов'язково будуть виконуватися, незалежно від *Умови*, принаймні хоча б один раз.

Задача 3.

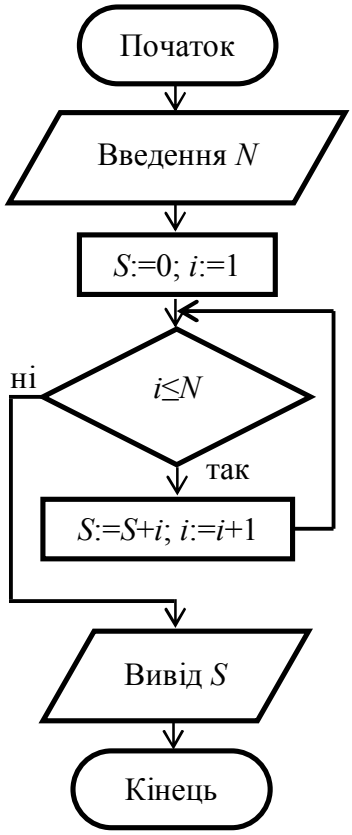
Скласти блок-схему і написати алгоритм НАМ для обчислення суми всіх натуральних чисел від 1 до *N*. Результат вивести на екран.

Суму позначимо через *S*, черговий доданок – через *i*. Спочатку *S:=0*, оскільки ще суми не знаходили, *i:=1* (перше натуральне число). Для знаходження суми потрібно до попередньої суми додати наступний доданок: *S:=S+i*. Для отримання наступного натурального числа потрібно попереднє збільшити на одиницю: *i:=i*+1.

Дану задачу розв'яжемо двома способами: із використанням циклу з передумовою і циклу з постумовою.

Цикл-ПОКИ

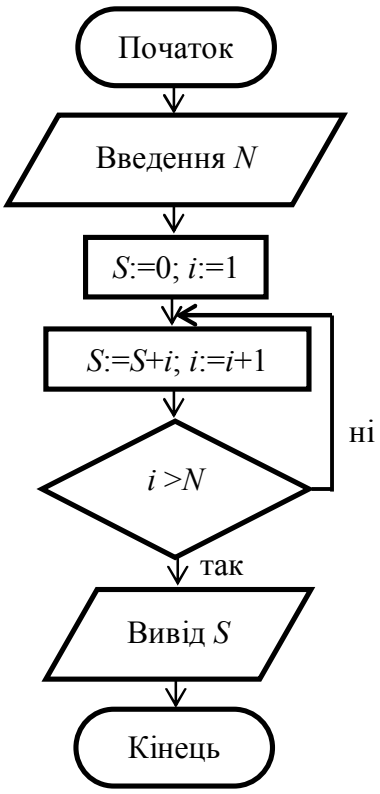
Виконання циклу продовжується поки i≤*N*.




```
АЛГ сума_1 (N,S: цілі)
АРГ N
РЕЗ S
ПОЧ ціле i
   ВИВІД (Введіть кількість чисел)
   ЗАПИТ (N)
S:=0
i:=1
ПОКИ i≤N
ПЦ
        S:=S+i
        i:=i+1
КЦ
ВИВІД (S=, S)
КІН
```

Цикл-ДО

Виконання циклу продовжується до тих пір, поки умова: *i>N* буде хибною.




```
АЛГ сума_2 (N,S: цілі)
АРГ N
РЕЗ S
ПОЧ ціле i
   ВИВІД (Введіть кількість чисел)
   ЗАПИТ (N)
   S:=0
   i:=1
   ПОВТОРЮВАТИ
   ПЦ
         S:=S+i
         i:=i+1
   КЦ
   ДО i>N
   ВИВІД (S=, S)
КІН
```

На практиці алгоритми розв'язування складних задач містять у собі всі три типи базових структур алгоритмів. Розглянуті принципи конструювання алгоритмів називають принципами структурного програмування.

Принципи структурного програмування

- Алгоритми, у яких використовується тільки структура «слідування», називають лінійними.
- Алгоритми, в основі яких лежить структура «розгалуження», називають алгоритмами з розгалуженнями.
- Алгоритми, в основі яких лежить структура «повторення», називають циклічними.

Мова програмування. Засоби створення програм. Середовище *PascalABC.Net*

Мова програмування Pascal

Одна з найпопулярніших мов програмування – це мова *Pascal*, яку створив у 1968-1969 роках швейцарський учений Ніклаус Вірт і назвав її Паскалем на честь великого французького математика, філософа і винахідника XVII століття Блеза Паскаля. Саме Паскаль винайшов обчислювальний пристрій, тому новій мові програмування було дано його ім'я. Вона дозволяє записувати команди, завдяки яким комп'ютер може розв'язувати математичні задачі, обробляти тексти, будувати зображення на екрані дисплея. Першим компілятором мови *Pascal* є *ETH Pascal*, створений у 1970 році.

Як і кожна мова, *Pascal* має свій алфавіт. До нього входять латинські літери, цифри від 0 до 9, спеціальні знаки (+, –, круглі, квадратні і фігурні дужки, крапка, кома та ін.), а також службові слова мови (*begin*, *end*, *for*, *while*

інші). Для зручності до текстів програм можна вносити пояснення (коментарі), які мовою *Pascal* записуються у фігурних дужках.

Одним із основних етапів розробки програм є присвоєння даним імен (ідентифікаторів), які використовуються для розв'язання задачі.

Структура алгоритму, описаного НАМ і програми мовою Pascal

АЛГ ім'я_алгоритму (список параметрів із зазначенням їх типів)	PROGRAM ім'я_програми;
АРГ список аргументів	розділи описів програми;
РЕЗ список результатів	
ПОЧ список допоміжних параметрів із зазначенням їх типів	BEGIN
Дії	Дії;
КІН	END.

Для опису даних на мові *Pascal* використовують такі основні типи: цілий, дійсний, символьний (табл. 2. 2).

Табл. 2. 2

Основні типи даних

НАМ	**Мовою *Pascal***	**Діапазон**
цілий	integer	-32768 … 32767
дійсний	real	-2,9·1039 … 1,7·1038
символьний	char	Множина символів кодової таблиці *ASCII*. Кожному символу ставиться у відповідність ціле число з діапазону 0…255

До розділу описів програми можуть входити розділи описів *констант* і *змінних*.

1. Розділ описів *констант*

const ідентифікатор1=значення константи1; ідентифікатор2=значення константи2; …

Приклад 1.

```
const a=5;
      b=10;
      Max=50.34;
      X1=68.5;
```

або

```
const a=5; b=10; Max=50.34; X1=68.5;
```

2. Розділ описів *змінних*

var список1 ідентифікаторів: тип1; список2 ідентифікаторів: тип2; …

Приклад 2.

var *sum*_1,*sum*_2,*L*:real;
*x*2,*min*:integer;

або

var *sum*_1,*sum*_2,*L*:real; *x*2,*min*:integer;

Розглянемо синтаксис операторів введення, виведення, присвоювання навчальною алгоритмічною мовою і мовою Pascal (табл. 2. 3).

Табл. 2. 3

Оператори введення, виведення, присвоювання

НАМ	***Мовою Pascal***
1. Вказівка введення	
запит (список ідентифікаторів)	read (список ідентифікаторів); readln (список ідентифікаторів);
2. Вказівка виведення	
вивід (список ідентифікаторів)	write (список ідентифікаторів); writeln (список ідентифікаторів);
3. Вказівка присвоювання	
ідентифікатор:=вираз	ідентифікатор:=вираз;

Оператор введення read (список ідентифікаторів), перехід на новий рядок не здійснює, після введення необхідної кількості даних, на відміну від оператора readln (список ідентифікаторів).

Приклад

read(*a*,*b*);
readln(*a*,*b*);

Оператор виведення writeln (список ідентифікаторів), автоматично здійснює перехід курсору на новий рядок, після виведення необхідної кількості даних, на відміну від оператора write (список ідентифікаторів).

Приклад

write(*a*,*b*);
writeln(*a*,*b*);

Оператор присвоювання **:=** працює аналогічно до НАМ.

Керувати інформацією, що виводиться, можна й іншим способом. У список ідентифікаторів можуть входити текстові константи, які НАМ позначають у прямокутній рамці, а мовою *Pascal* їх будемо виділяти одинарними лапками (**' '**). Текстові константи виводяться на екран без змін.

Задача 3.

Написати алгоритм НАМ і програму мовою *Pascal* для знаходження довжини шляху, пройденого пішоходом, якщо відома його швидкість і час, витрачений на дорогу.

Розв'язок. Блок-схема і алгоритм НАМ створений раніше (задача 1).

АЛГ Шлях (*V*, *t*, *S*: дійсні)
АРГ *V*, *t*
РЕЗ *S*
ПОЧ
    Вивід(*V*=)
    Запит(*V*)
    Вивід(*t*=)
    Запит(*t*)
    S:=*V***t*
    Вивід(*S*=, *S*)
КІН

```
Program shlyah;
Var V,t,S:real;

Begin
    write('V=');
    readln(V);
    write('t=');
    readln(t);
    S:=V*t;
    writeln('S=',S);
End.
```

Середовище PascalABC.Net

Завантаження *PascalABC.Net* відбувається за допомогою ярлика на робочому столі або з головного меню.

Після завантаження системи на екрані з'являється вікно програми *PascalABC.Net*, що може складатися із таких основних частин:

1) *вікно введення коду програми* (в це вікно, як у текстовий редактор, із клавіатури, або копіюванням із інших програм, уводиться текст програми. Редагування відбувається аналогічно до редагування у текстовому редакторі);
2) *вікно виведення результатів програми* (призначене для виведення значень ідентифікаторів, що описано у списках ідентифікаторів команд *write* і *writeln*);
3) *вікно введення даних* (призначене для введення значень ідентифікаторів, що описано у списках ідентифікаторів команд *read* і *readln)*. Це вікно відкривається лише після запуску програми на виконання і наявності в ній операторів введення. Введення даних у це вікно супроводжується їх автоматичним виведенням у вікні виведення результатів програми. Після натискання клавіші *Enter* відповідні ідентифікатори приймають уведені значення із вікна введення даних, а саме вікно після цього закривається.

На рисунку 2.1 представлено вікно *PascalABC.Net* після запуску на виконання програми (задача 3).

Перехід з основного вікна програми в головне меню і навпаки здійснюється за допомогою мишки або клавіші *F10*.

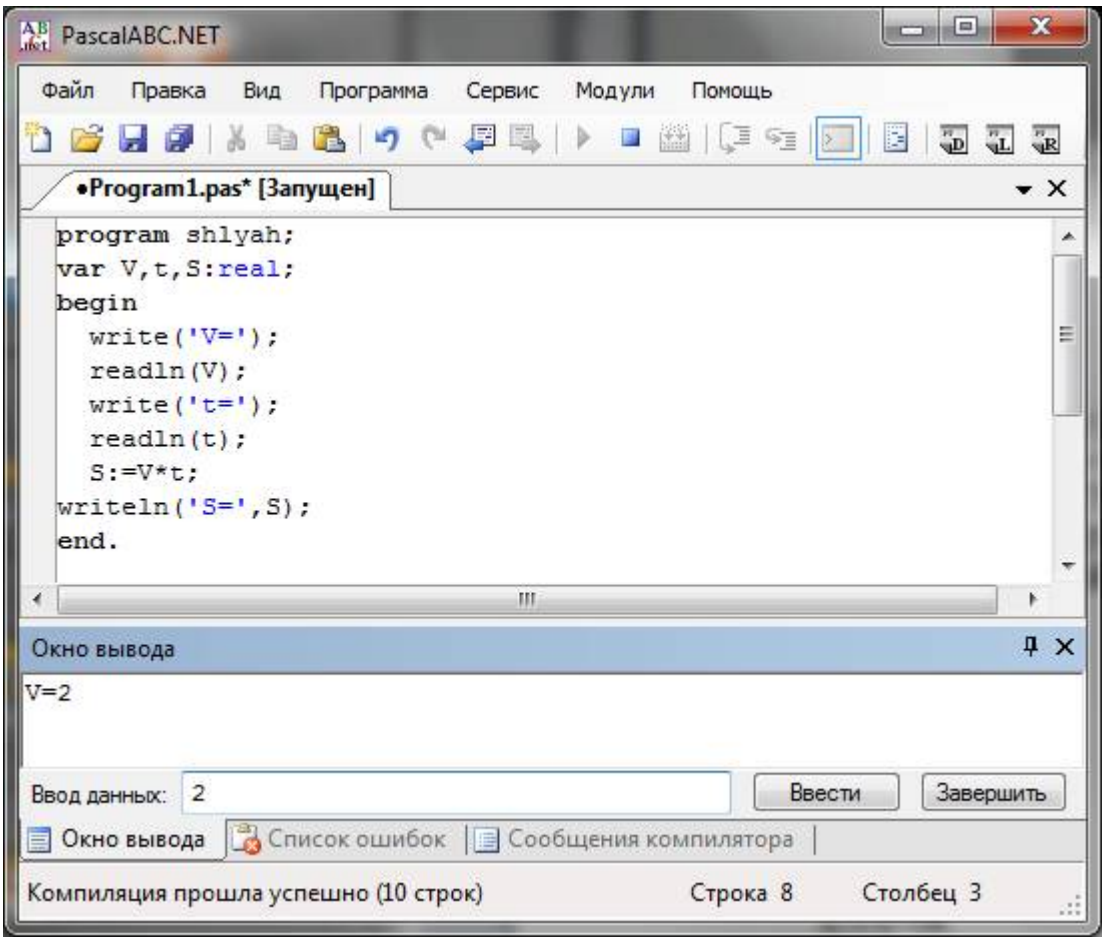


Рис. 2.1. Вікно PascalABC.Net

Умовні оператори мовою Pascal

Неповне розгалуження

ЯКЩО <Умова>	IF <Умова>
ТО	THEN
	BEGIN
Дії	Дії;
ВСЕ	END;

Повне розгалуження

У повному і неповному розгалуженні службові слова **BEGIN** і **END** можна не писати, якщо **Дії** складаються з однієї дії.

ЯКЩО <Умова>	IF <Умова>
ТО	THEN
	BEGIN
Дії 1	Дії 1;
	END
ІНАКШЕ	ELSE
	BEGIN
Дії 2	Дії 2;
ВСЕ	END;

Задача 4.

Написати алгоритм НАМ і програму мовою *Pascal* для обчислення значення функції: $y = \begin{cases} 2x, & якщо \quad x < 0; \\ 3x + 1, & якщо\ x \geq 0. \end{cases}$ Вивести на екран значення аргумента і результата.

Блок-схема і алгоритм НАМ створено раніше (задача 2).

```
АЛГ функція_1 (X,Y: дійсні)
АРГ X
РЕЗ Y
ПОЧ
    ВИВІД (X=)
    ЗАПИТ (X)
    ЯКЩО X<0
        ТО
            Y:=2X
        ІНАКШЕ
            Y:=3X+1
    ВСЕ
    ВИВІД (X=, X, Y=, Y)
КІН
```

```
Program function_1;

Var X,Y:real;
Begin
write('X=');
readln(X);
if X<0
        then
            Y:=2*X
        else
            Y:=3*X+1;

writeln('X=',X, ', Y=',Y)
End.
```

Цикли в алгоритмах і програмах мовою Pascal

Оператор циклу з передумовою (WHILE)

```
ПОКИ <Умова>                    WHILE <Умова> DO
ПЦ                              BEGIN
    Дії                             Дії;
КЦ                              END;
```

У написанні циклу з передумовою службові слова **BEGIN** і **END** можна не писати, якщо **Дії** складаються з однієї дії.

Оператор циклу з постумовою (REPEAT)

```
ПОВТОРЮВАТИ                     REPEAT
ПЦ
    Дії                             Дії;
КЦ
ДО <Умова>                      UNTIL <Умова>;
```

Задача 5.

Написати алгоритм НАМ і програму мовою *Pascal* для обчислення суми всіх натуральних чисел від 1 до *N*. Результат вивести на екран.

Дану задачу розв’яжемо із використанням циклу з передумовою. Виконання циклу продовжується до тих пір, поки i≤*N*.

```
АЛГ сума_1 (N,S: цілі)
АРГ N
РЕЗ S
ПОЧ ціле i
   ВИВІД (Введіть кількість чисел)
   ЗАПИТ (N)
   S:=0
   i:=1
   ПОКИ i≤N
   ПЦ
        S:=S+i
        i:=i+1
   КЦ
   ВИВІД (S=, S)
КІН
```

```
Program suma_1;

Var N,S,i:INTEGER;
Begin
   writeln('Введіть кількість чисел');
   readln(N);
   S:=0;
   i:=1;
   while i<=N do
      Begin
         S:=S+i;
         i:=i+1
      End;
   writeln('S=',S)
End.
```

Задача 6.

Написати алгоритм НАМ і програму мовою *Pascal* для побудови таблиці значень функції $y:=3x^2+x$ на проміжку $x\in[1; 3]$ з кроком 0,2, де x, y – дійсні числа.

```
АЛГ функція_1 (x,y: цілі)
АРГ x
РЕЗ y
ПОЧ
   x:=1
   ПОКИ x≤3
   ПЦ
      y:=3x^2+ x
      ВИВІД (y=, y)
      x:=x+0.2
   КЦ
КІН
```

```
program f_1;
var x,y:real;

begin
      x:=1;
      while x<=3 do
      begin
            y:=3*sqr(x)+x;
            writeln('y=',y);
            x:=x+0.2;
      end;
end.
```

Середовище Scratch, призначення, розробники, інтерфейс

Історичні відомості

Scratch є середовищем об'єктно-орієнтованого візуального програмування, що призначене для створення комп'ютерних анімацій, мультимедійних презентацій, анімаційних та інтерактивних історій, ігор, моделей. *Scratch* створено у дослідницькій групі Lifelong Kindergarten research group при Массачусетському технологічному інституті. *Scratch* створювали для учнів віком від 8 до 16 років, але й діти молодшого віку можуть працювати

в цьому середовищі над проектами разом з батьками або старшими товаришами. З іншого ж боку, студентам вищих навчальних закладів також рекомендується використовувати *Scratch* на заняттях.

Переваги використання середовища Scratch:

- *платформна незалежність* (можна встановлювати на комп'ютерах під керуванням операційних систем Microsoft Windows, Macintosh, Linux);
- *алгоритмічна повнота* (підтримка концепції об'єктно-орієнтованого програмування, а саме: лінійні процеси, циклічні процеси, розгалужені процеси, використання глобальних і локальних змінних, робота з даними різних типів (символьними, числовими, логічними, графічними, аудіо) та виразами (числовими, текстовими, логічними, порівняння), властивості об'єктів (спрайтів), методи, події);
- *наочність* створення алгоритму (перетягування вказівок на поле скриптів);
- *налаштування інтерфейсу* (наприклад, встановлення потрібної мови);
- *відкритість* (середовище програмування можна безкоштовно завантажити і вільно використовувати у навчальних цілях).

Посилання для завантаження програми можна знайти на офіційному сайті розробників за адресою: *http://scratch.mit.edu/scratch_1.4/*.

Основні поняття Scratch

У перекладі з англ. мови іменник «scratch» можна тлумачити по-різному: дряпання, насічка, мітка, стартова межа інші. Дієслово «to scratch» означає: дряпати, шкрябати, рити кігтями, надряпати малюнок інші. Можливо, тому символом програми служить веселий рудий кіт. Якщо розглядати переклад слова «scratch» як частини висловів, то в перекладі отримується: випадково, поспішно, на швидку руку. А вислів «start from scratch» перекладають як «почати з нуля». Тобто, вивчати елементи програмування можна з використанням середовища *Scratch*.

Scratch є мультимедійною системою, основну частину операторів якої спрямовано на роботу з графікою, звуком, анімацією та ін. Середовище передбачає колективну роботу над проектами й обмін результатами через сайт *Scratch*-товариства. *Scratch* є інтерактивним середовищем, що побудовано на принципах, які інтуїтивно зрозумілі дитині. А саме, складання програми мишкою з готових блоків-цеглин подібно до того, як діти будують будиночки і машинки з деталей конструктора. Такий спосіб складання програм унеможливлює виникнення синтаксичних проблем, що є важливим саме для молодших школярів.

Головне вікно програми поділено на декілька частин (рис. 2. 2).

У середовищі *Scratch* виконавцями є спрайт (-и) і сцена. *Спрайтом* називають об'єкт *Scratch*, що пов'язаний із зображенням, набором змінних і скриптів, які визначають його поведінку.

За замовчуванням використовують спеціального виконавця вказівок – *Рудого кота*. Він може рухатися, говорити, змінювати зовнішній вигляд, взаємодіяти з іншими виконавцями на сцені. Інших

виконавців можна додавати з бібліотеки *Scratch* або створювати у графічному редакторі (вбудованому в *Scratch* або інших графічних програмах).

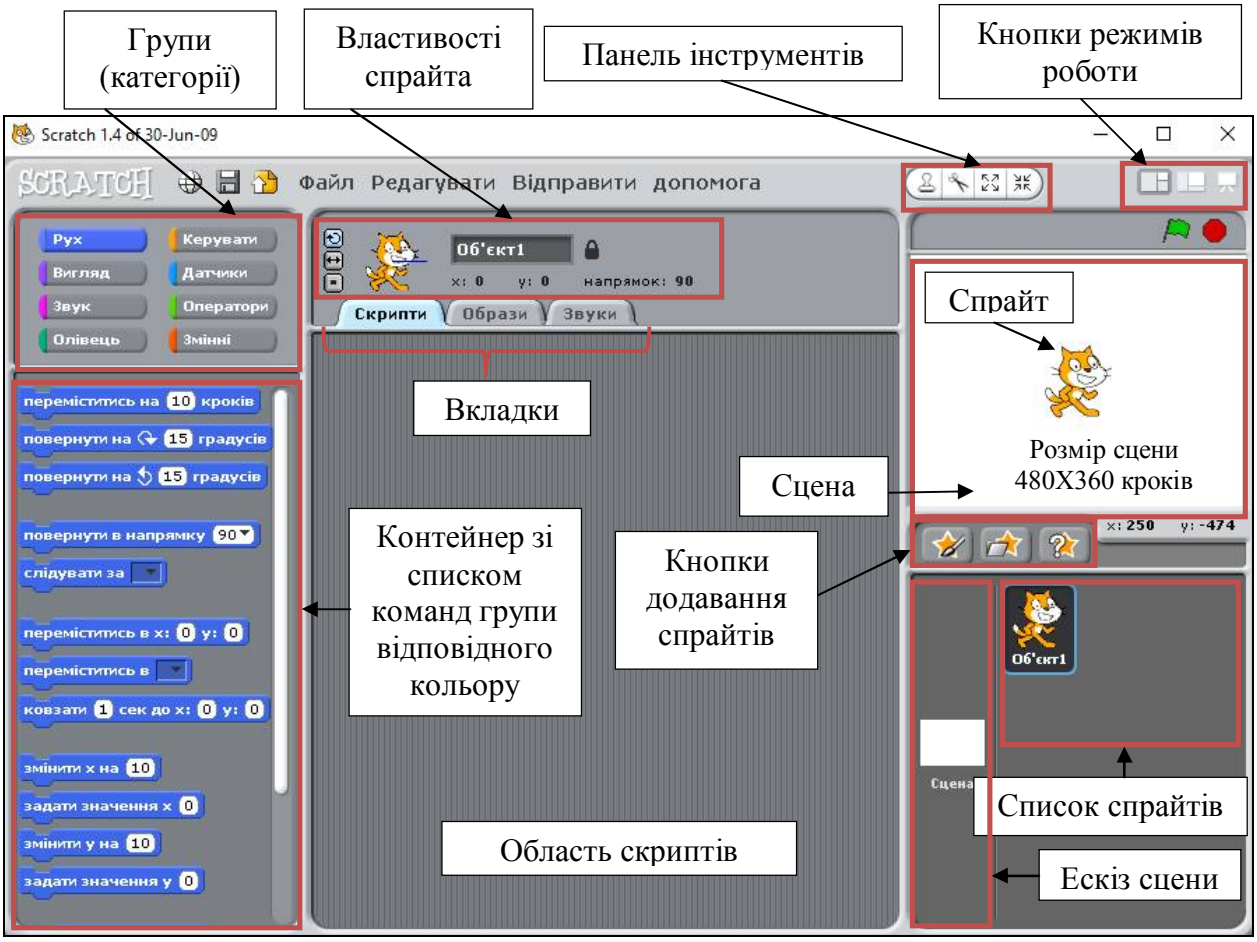


Рис. 2. 2. Елементи вікна середовища *Scratch*

Сценою називають область, в якій діє спрайт при виконанні програми. *Скрипт* (в перекладі означає сценарій, метод) – послідовність вказівок, що визначає дії та порядок їх виконання певним об'єктом (спрайтом). Скрипти збирають за допомогою сполучення блоків послідовно або вбудовуючи в інший блок. Один спрайт може мати декілька скриптів, що запускаються натисненням клавіші або кнопки миші, таймером або отриманням повідомлення від іншого спрайту. Скрипт складається зі стеків, що є набором послідовно сполучених різнокольорових графічних блоків у межах однієї події. Блок є мінімальним фрагментом програми у *Scratch* (змінна, оператор інше). *Образи* є сукупністю зображень одного й того ж об'єкту, кожне з яких відрізняється від попередніх. Звуки містять приєднані звукові ефекти та музику.

Команди виглядають як *блоки*, які користувач перетягує в область скриптів. Команди об'єднані в групи: «*Рух*» (табл. 2. 4), «*Вигляд*» (табл. 2. 5), «*Звук*» (табл. 2. 6), «*Олівець*» (табл. 2. 7), «*Керувати*» (табл. 2. 8), «*Датчики*» (табл. 2. 9), «*Оператори*» (рис. 2. 3), «*Змінні*».

Табл. 2. 4

Команди та їх призначення (група «*Рух*»)

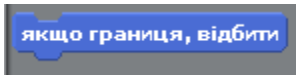
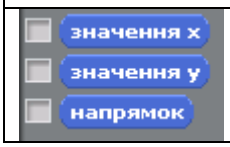
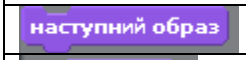
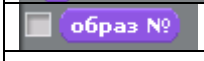
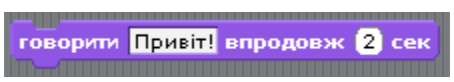
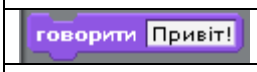
Команда	Призначення
	пройти вказану кількість кроків
	повернути (стрілочка вказує напрямок повороту – за годинниковою стрілкою чи проти)
	повернути у вказаному напрямку (90° – праворуч, –90° – ліворуч, 0° – вгору, 180° – вниз)
	повернутися у напрямку іншого об’єкта або вказівника миші
	у робочій області середовища перемістись в точку з вказаними координатами
	перейти в точку, де знаходиться вказівник миші або інший об’єкт
	плавно перемістись в точку з вказаними координатами за вказаний час (у секундах)
	змінити положення об’єкта по осі X на вказану кількість кроків
	розмістити об’єкт у вказане значення по осі X
	змінити положення об’єкта по осі Y на вказану кількість кроків
	розмістити об’єкт у вказане значення по осі Y
	якщо об’єкт доходить до краю робочої області, то відійти
Наступні команди використовуються в якості вхідних параметрів для інших команд	
	повертає значення по осі X повертає значення по осі Y повертає напрямок

Табл. 2. 5

Команди та їх призначення (група «*Вигляд*»)

Команда	Призначення
	вибрати із запропонованого списку новий образ героя
	перейти до наступного образу
	повертає значення образу спрайта
	над героєм з’являється репліка, яка видима задану кількість секунд
	над героєм з’являється репліка
	над героєм з’являється думка-репліка, яка видима задану кількість секунд

подумати Хм...	над героєм з’являється думка-репліка
змінити ефект колір на 25	задати нове значення та/або новий ефект зміни вигляду об’єкта (колір, здуття, обертання, пікселями, мозаїка, яскравість, привид)
встановити ефект колір в 0	задати значення ефекту зміни вигляду об’єкта (колір, здуття, обертання, пікселями, мозаїка, яскравість, привид)
очистити графічні ефекти	відмінити всі попередні команди щодо зміни зовнішнього вигляду об’єкта
змінити розмір на 10	змінити розмір об’єкта збільшення та зменшення)
задати розмір 100 %	встановити розмір об’єктів
розмір	повертає значення розміру образа спрайта
показати	об’єкт з’являється на сцені
сховати	об’єкт зникає зі сцени
стати попереду всіх	перемістити об’єкт на передній план
стати позаду 1 об'єктів	перемістити об’єкт на задній план

Табл. 2. 6

Команди та їх призначення (група «*Звук*»)

Команда	**Призначення**
грати звук няв	відтворити звук, виконання об’єктом інших команд відбувається одночасно зі звуковим супроводом
грати звук няв до завершення	відтворити звук, виконання об’єктом інших команд відбувається після завершення звукового супроводу
зупини всі звуки	зупинка звуків
програти на барабані 48 0.2 ударів	на ударному інструменті програти вказану кількість тактів
пауза 0.2 тактів	зупинити відтворення звуку на вказану долю тактів
грати ноту 60 0.5 тактів	грати вказану ноту задану кількість тактів (у секундах)
задати інструмент 1	вибрати музичний інструмент з запропонованого списку (музичні інструменти позначаються парою - число і назва)
змінити гучність на -10	змінити гучність звучання на вказану величину
встановити гучність 100 %	задати гучність звучання у відсотках
гучність	повертає значення гучності звучання
змінити темп на 20	змінити темп звучання на вказану величину

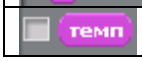
	задати темп звучання
	повертає значення темпу мелодії

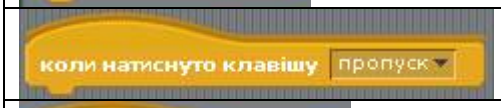
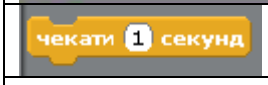
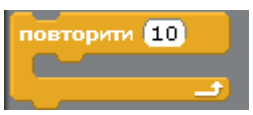
Табл. 2. 7

Команди та їх призначення (група «*Олівець*»)

Команда	Призначення
	очистити екран від усіх зображень
	опустити олівець (об'єкт рухається і залишає слід у вигляді лінії)
	підняти олівець (об'єкт рухається і не наносить зображення)
	вибрати колір лінії, що залишає об'єкт
	змінити колір лінії, що залишає об'єкт
	задати значення кольору лінії, що залишає об'єкт
	змінити градієнт кольору лінії, що залишає об'єкт (100 – білий колір, 0 – чорний)
	задати градієнт кольору лінії, що залишає об'єкт
	змінити товщину лінії, що залишає об'єкт
	задати товщину лінії, що залишає об'єкт
	копіювання зображення

Табл. 2. 8

Команди та їх призначення (група «*Керувати*»)

Команда	Призначення
	команди починають виконуватись при натисненні кнопки у вигляді зеленого прапорця
	запуск команд за допомогою клавіатури
	запуск команд при натисненні на обраному об'єкті
	команда паузи
	блок команд, які знаходяться всередині цієї команди будуть виконуватись постійно (поки не натиснути кнопку «Стоп»)
	повторюється цикл команд, які знаходяться всередині цієї команди (кількість повторень задається)
	повідомлення для умови виконання іншої команди

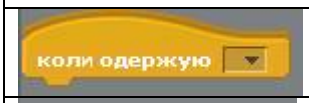
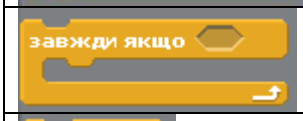
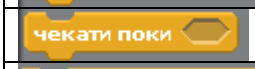
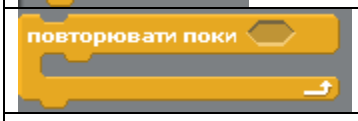
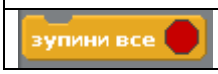
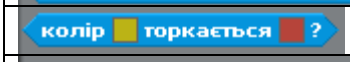
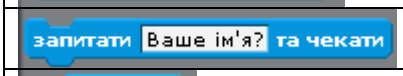
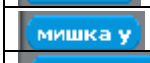
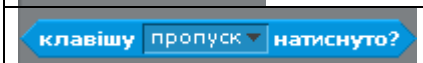
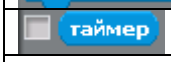
	повідомлення, за яким повинна слідувати інша команда
	повідомлення запускає активність іншого виконавця
	умова, під час виконання якої завжди виконуються команди всередині конструкції
	неповне розгалуження
	повне розгалуження
	цикл (чекати поки не виконається умова)
	цикл з передумовою
	зупинити виконання програми для даного виконавця
	зупинка виконання всіх програм

Табл. 2. 9

Команди та їх призначення (група «*Датчики*»)

Команда	**Призначення**
	перевіряє чи торкається об’єкт вказівника миші, границі або іншого об’єкта
	перевіряє чи торкається об’єкт певного кольору
	перевіряє чи торкається колір 1 кольору 2
	введення імені у відображене поле
	відображення відповіді в області сцени
	повертає значення показника миші по осі х
	повертає значення показника миші по осі у
	перевіряє чи натиснута керуюча клавіша мишки
	перевіряє чи натиснута одну з клавіш на клавіатурі
	відстань до вказівника миші або іншого об’єкта
	почати знову відлік часу за таймером
	встановлення відліку часу
	положення по осі Х, У чи параметри розміру, гучності, напрямку, образу зі сцени або іншого об’єкта
	встановлення кнопки гучності
	встановлення кнопки більшої гучності

	встановлення кнопки значень датчика: слайдер, легкий, звук, резистентність (A, B, C, D)
	встановлення кнопки датчика: кнопку натиснуто, під'єднано (A, B, C, D)

Рис. 2. 3. Вигляд блоків групи «Оператори»

В групі «*Змінні*» наявні два блоки, що дозволяють виконати команди: *створити змінну*, *створити список*.

ЛАБОРАТОРНІ РОБОТИ ДО РОЗДІЛУ «ЕЛЕМЕНТИ ПРОГРАМУВАННЯ»

Лабораторна робота 2. 1

Базові алгоритмічні конструкції. Лінійні алгоритми

Завдання

1. Човняру потрібно перевезти в човні через річку вовка, козу і капусту. У човні, крім човняра, вміщується або тільки вовк, або тільки коза, або тільки капуста. На березі не можна залишати козу з вовком або козу з капустою. Скласти алгоритм перевезення. Подати його у словесній формі (ця старовинна задача вперше трапляється в математичних рукописах VIII ст.).
2. Скласти блок-схему алгоритму знаходження x з рівняння $2x+a=c$. Виконати його при: *а*) $a=5$, с=7; *б*) $a=-15$, $c=105$; *в*) $a=5$, $c=5$.
3. Скласти алгоритми (НАМ) розв'язування задач: №№ 50, 51, 54 на стор.54; №№ 73, 74, 75 на стор.55 з посібника *Караванова Т.П. Інформатика: основи алгоритмізації та програмування: 777 задач з рек. та прикл.: Навч. посіб. / За заг. ред. М. З.Згуровського – К. : Генеза, 2006. – 286 с.*

Контрольні питання

1. Що таке алгоритм? Наведіть приклади.
2. Що таке система команд виконавця? Наведіть приклади виконавців і системи їх команд.
3. Назвіть властивості алгоритму. Поясніть кожну з них.

4. Поясніть відмінність між словесною та графічною формами подання алгоритму.
5. Назвіть елементи блок-схеми алгоритму та поясніть їх призначення.
6. Який алгоритм (фрагмент алгоритму) називається лінійним?
7. Що таке команда присвоювання? Як вона позначається?

Лабораторна робота 2. 2
Засоби створення програм. Система програмування Pascal ABC.NET

Завдання

1. Запустити програму *Pascal ABC.NET*.
2. Ознайомитися з призначенням команд меню середовища програмування *Pascal ABC.NET*.
3. Відкрити папку *C:\PABCWork.NET\Samples*. У даній папці знаходяться приклади готових до виконання програм. Відкрити не менше трьох програм, запустити їх на виконання і переглянути отримані результати.
4. Знайти у папці *Samples* хоча б дві програми, результатом виконання яких є фрактальний малюнок (*C:\PABCWork.NET\Samples\Graphics\GraphABC\Fractals*). Запустити їх на виконання і переглянути отримані результати.
5. Створити папку з іменем *Pascal* (з урахуванням правил розміщення папок і файлів на жорстких дисках університетських комп'ютерів). Результати виконання наступних завдань зберігати у цій папці.
6. У середовищі програмування *Pascal ABC.NET* набрати текст наступної програми.

```
Program Hello;
Var x:string;
Begin
    writeln('Привіт!!!');
    writeln('Як тебе звати?');
    read(x);
    writeln('Ласкаво просимо, ',x,', до програмування мовою Паскаль!');
    writeln('Бувай!');
End.
```

7. Зберегти програму у файлі з іменем *Hello*.
8. Компілювати програму. Для цього виконати команду меню *Программа => Компилировать* або натиснути кнопку  (*Компилировать*) на панелі інструментів, також можна скористатися комбінацією клавіш *Ctrl+F9*. Якщо є повідомлення про помилки, то виправити їх. Запустити програму на виконання. Для цього виконати команду *Программа => Выполнить* або натиснути кнопку 

(*Выполнить*) на панелі інструментів, також можна скористатися клавішею F9.

9. На запит програми відреагувати введенням власного імені українською мовою. Натиснути клавішу *Enter*.
10. Переглянути результат виконання програми у вікні виведення. Для цього (якщо потрібно) встановити нижню панель командою меню *Вид => Нижняя панель*; якщо вікно виведення не буде встановлено, то скористатися командою меню *Вид => Окно вывода*.
11. У зошиті дати письмово відповідь на питання, що означає запис рядка: **var** x:string.
12. Написати, зберегти, здійснити компіляцію та виконати програму, яка б за допомогою знаків ***** або інших символів, виводила деяку фігуру. Фігура у кожного виконавця повинна бути унікальною. Приклад виведення фігури представлено на рисунку 2. 4.

Рис. 2. 4. Вигляд фігури у вікні виведення

Контрольні питання

1. Назвіть команди меню середовища програмування *Pascal ABC.NET*?
2. Як створити новий файл?
3. Як виділити потрібний фрагмент тексту?
4. Як виконати копіювання і вставку блоку тексту програми?
5. Як виконати переміщення блоку тексту програми?
6. Як виконати видалення блоку тексту програми?
7. Як зберегти програму у файлі на диску?
8. Як компілювати програму?
9. Як запустити програму на виконання?
10. Як переглянути результат виконання програми у вікні виведення?
11. Як вийти з середовища *Pascal ABC.NET*?

Лабораторна робота 2. 3
Реалізація лінійних алгоритмів у середовищі програмування

Завдання

1. Написати алгоритм і програму для обчислення значення функції $y = a + \frac{2x-1}{7}$. Вивести на екран значення аргументів і результату.

АЛГ функція (*a*, *x*, *y*:дійсні)	PROGRAM funktsiya;
АРГ *a*, *x*	var *a*,*x*,*y*:real;
РЕЗ *y*	
ПОЧ	BEGIN
вивід (*Уведіть a, x*)	write('*Уведіть a,x*');
запит (*a,x*)	read(*a,x*);
$y = a + \frac{2x-1}{7}$	y:=*a*+(2**x*-1)/7;
вивід (*a*=,*a*, *x*=,*x*, *y*=,*y*)	write('*a*=',*a*,', *x*=',*x*,' ,*y*=',*y*);
КІН	END.

2. Зберегти, здійснити компілювання та тестування програми.
3. Відповідно до свого варіанту написати алгоритм розв'язку задачі НАМ, реалізувати її у середовищі *Pascal ABC.NET*. Знайти розв'язки при відповідних вхідних даних. Оформити звіт згідно вимог до виконання лабораторної роботи.

1.

a) $z = 2x^2 - 3x + 2$, для *x=0; x=-2,6;*

b) $z = \frac{c+1}{\sqrt{x-1}}$, для *x=1,12 і c=-3,14.*

2.

a) $y = 2x^3 + x - 1$, для *x=-1; x=1,3*;

b) $t = \frac{a+b}{|a-b|}$, для *a=-0,3; b=2,7.*

3.

a) $z = 3x^3 - 5x + 2$, для *x=-2,05; x=7*;

b) $k = \frac{a-2}{x^2+2}$, для *a=3,2; x=-3,6.*

4.

a) $t = 5x^2 + x$, для *x=0,8; x=-1,7*;

b) $y = \sqrt{a+d} - \frac{a-d}{d}$, для *a=-0,1; d=2,32.*

5.

a) $y = 2x^2 - 3x + 1$, для *x=0; x=-5,6*;

b) $t = b + 1 - \frac{b+1}{b-c}$, для *b=-0,2; c=2,6.*

6.

a) $y = 3x^3 - 2x - 3$, для *x=1,8; x=-3,6*;

b) $k = \frac{a-5}{x^2+2-a}$, для *a=0,12; x=-5,1.*

7.

a) $y = x^4 - x + 3$, для *x=1,12; x=-2,6*;

b) $z = \frac{c+2}{\sqrt{x-3}}$, для *x=-2,2 і c=3,14.*

8.

a) $y = 2x^3 + x - 3$, для *x=2,8; x=-1,6*;

b) $t = \frac{a+6}{|a-b|} + 8$, для *a=-1,8; b=3,17.*

9.

a) $y = 4x^3 + 2x - 1$, для *x=-1,91; x=1,22*;

b) $k = \frac{x^2}{x-2}$, для *a=2,12; x=-2,7.*

10.

a) $y = 3x^2 - x$, для *x=1,17; x=-4,3*;

b) $y = \sqrt{a-d} - \frac{a+1}{d+5}$, для *a=-1,1; d=5,2.*

11.

a) $y = 3x^3 - |x-3|$, для *x=-5,8; x=2,6*;

b) $k = \frac{a-5}{x-x^2}$, для *a=0,7; x=-3,21.*

12.

a) $y = x^2 - 2x - 3$, для *x=-5,8; x=2,12*;

b) $t = -\frac{b^2+b+1}{b+c}$, для *b=-0,52; c=3,87.*

13.

a) $y = 3x^2 + x - 1$, для *x=-3; x=1,7*;

b) $t = \frac{a^2-b}{|a-b|}$, для *a=3; b-=1,7.*

14.

a) $y = 5x^2 - 3x - 7$, для *x=0; x=-5,5*;

b) $z = 3b - \frac{b+1}{b+c}$, для *b=-0,2; c=3,8.*

15.

a) $y = 3x^3 - 4x + 8$, для *x=2,05; x=7,3*;

b) $z = \frac{a+2}{x^2+2}$, для *a=3; x=-3,16.*

Контрольні питання

1. Назвіть основні засоби створення програм.
2. Що таке комп'ютерна програма?
3. Які дані називають вхідними, вихідними, проміжними?
4. Що таке мова програмування?
5. Назвіть компоненти, з яких складається мова програмування.

6. Опишіть кожний компонент мови програмування.
7. Назвіть типи елементів, що використовуються при написанні програм.
8. Що таке синтаксична помилка?
9. Назвіть складові інтегрованого середовища розробки програм.
10. Що таке машинна мова програмування? Який вигляд мають команди в цій мові програмування?
11. Які програми називають компіляторами? Для чого вони призначені?
12. Назвіть команди введення даних у мові програмування *Pascal*.
13. Назвіть команди виведення даних у мові програмування *Pascal*.
14. Опишіть загальну структуру *Pascal*-програми.

Лабораторна робота 2. 4
Вказівки розгалуження. Умовні оператори

Завдання

1. Відповідно до свого варіанту написати алгоритм НАМ і програму для обчислення значення функції. Вивести на екран значення аргументу і відповідного результату.

1) $$Y=\begin{cases}\dfrac{2x-1}{3x-6}, \text{ якщо } x<1;\\ \sqrt{x-1}, \text{ якщо } x\ge 1.\end{cases}$$

2) $$Y=\begin{cases}\dfrac{2x-1}{3x+6}, \text{ якщо } x\ne -2;\\ 5x^2+x, \text{ якщо } x=-2.\end{cases}$$

3) $$Y=3x^2-4+\begin{cases}\dfrac{2x-3}{4x-8}, \text{ якщо } x\ne 2;\\ 2x-1, \text{ якщо } x=2.\end{cases}$$

4) $$Y=4x-3+\begin{cases}5x+1, \text{ якщо } x<-1;\\ \sqrt{x+1}, \text{ якщо } x\ge -1.\end{cases}$$

5) $$Y=\frac{3x-2}{4}+\begin{cases}2x^2-7, \text{ якщо } x<-1;\\ \sqrt{x+1}, \text{ якщо } x\ge -1.\end{cases}$$

6) $$Y=3x+\begin{cases}2x-1, \text{ якщо } x<-\dfrac{1}{3};\\ \sqrt{3x+1}, \text{ якщо } x\ge -\dfrac{1}{3}.\end{cases}$$

7) $$Y=6x-5+\begin{cases}x-1, \text{ якщо } x<\dfrac{3}{2};\\ \sqrt{2x+3}, \text{ якщо } x\ge \dfrac{3}{2}.\end{cases}$$

8) $$Y=\frac{3x-2}{4}+\begin{cases}2x^2-7, \text{ якщо } x<-1;\\ \sqrt{x+1}, \text{ якщо } x\ge -1.\end{cases}$$

9) $Y = 2x + \begin{cases} x+1, \text{ якщо } x > -1; \\ x^2 - 1, \text{ якщо } x \le -1. \end{cases}$

10) $Y = x + \begin{cases} x-1, \text{ якщо } x \ne 1; \\ 2x^2 - 1, \text{ якщо } x = 1. \end{cases}$

11) $Y = x + 2 + \begin{cases} 3x+1, \text{ якщо } x > -1; \\ x^2 - 1, \text{ якщо } x \le -1. \end{cases}$

12) $Y = 5x - 5 - \begin{cases} 2x - 1, \text{ якщо } x < \frac{4}{2}; \\ \sqrt{4x+3}, \text{ якщо } x \ge \frac{4}{2}. \end{cases}$

13) $Y = x - \begin{cases} \frac{5x-1}{3x-6}, \text{ якщо } x < 1; \\ \sqrt{2x-1}, \text{ якщо } x \ge 1. \end{cases}$

14) $Y = \frac{x-2}{6} - \begin{cases} 5x^2 - 8, \text{ якщо } x < -1; \\ \sqrt{x+1}, \text{ якщо } x \ge -1. \end{cases}$

15) $Y = \frac{x^2-2}{4} - \begin{cases} 4x^3 + 7, \text{ якщо } x < -1; \\ \sqrt{x+1}, \text{ якщо } x \ge -1. \end{cases}$

2. Написати алгоритм і програму розв'язання задач.
   1) Визначити, чи є серед трьох чисел *а, в, с* хоча б одна пара рівних між собою чисел.
   2) Дано трицифрове число. Визначити, чи можуть цифри даного числа бути довжинами сторін трикутника.

Контрольні питання

1. Що таке логічний вираз?
2. Які значення можуть бути результатом виконання команди перевірки умови (обчислення значення логічного виразу)?
3. Як позначається команда перевірки умови в блок-схемі алгоритму?
4. Назвіть характерні особливості розгалуження. Зобразіть блок-схему розгалуження. Опишіть особливості її виконання.
5. Наведіть приклади правил з української мови, що містять розгалуження.
6. Наведіть приклади життєвих ситуацій, які можна описати алгоритмом з розгалуженням.

Лабораторна робота 2. 5
Реалізація циклічних алгоритмів у середовищі програмування
Завдання

1. Виконати наступні вправи (1-6). Розв'язки і результати занотувати у зошит.

№ 1
Знайти значення *l* і *k*:
```
l:=1;
k:=1;
n:=3;
while k<n do
      begin
      l:=l+k;
      k:=k+1.5;
end;
```

№ 2
Знайти значення *p*:
```
p:=8;
k:=5;
n:=-1;
while k<n do
begin
      p:=p+1;
      k:=k-0.5;
end;
```

№ 3
Знайти функцію *f(x)*, якщо *n*>4
s=f(x) і:
```
k:=3;
s:=1/2;
while k<n do
begin
      s:=s+1/k;
      k:=k+1;
end;
```

№ 4
Знайти значення *a*:
```
a:=0;
k:=0;
n:=2;
while k<n do
begin
      a:=a+1;
      k:=k+0.5;
end;
```

№ 5
Знайти значення *p*:
```
p:=-1;
k:=3;
n:=1;
while k>n do
begin
      p:=p+1;
      k:=k-0.5;
end;
```

№ 6
Знайти значення *b*:
```
b:=1;
t:=1;
m:=3;
while t<m do
begin
      b:=b-3;
      t:=t+0.6;
end;
```

2. Відповідно до свого варіанту написати алгоритм НАМ розв'язку задачі, реалізувати її у середовищі *Pascal ABC.NET*. Оформити звіт згідно вимог до виконання лабораторної роботи.

Задача. Побудувати таблицю значень функції (функцію обрати згідно свого варіанту) на проміжку $x \in [2; 5]$ з кроком 0,5, де x, y – дійсні числа.

1.1. $y = \sqrt{1+x}$
1.2. $y = \dfrac{1}{x^2+1}$
1.3. $y = \sqrt{x+1}$
1.4. $y = x^2 - 5$
1.5. $y = \sqrt{x+4}$
1.6. $y = \sqrt{|x|}$
1.7. $y = \dfrac{x}{x-2}$
1.8. $y = \dfrac{1}{x^2+1}$
1.9. $y = x^2 - x + 1$
1.10. $y = 2x\sqrt{x+1}$
1.11. $y = \sqrt{x(x+1)}$
1.12. $y = \dfrac{x-1}{x^2+1}$
1.13. $y = \dfrac{3x}{x^2+1}$
1.14. $y = \dfrac{x-1}{x^2+1}$
1.15. $y = 2x^2 + x - 1$

Контрольні питання

1. Наведіть загальну форму циклу із передумовою в мові програмування *Pascal*.
2. Що таке заголовок та тіло циклу?
3. Наведіть блок-схему циклу з передумовою.
4. Поясніть відмінності у виконанні основних алгоритмічних структур (слідування, розгалуження, цикл).

Лабораторна робота 2. 6
Сходинки до інформатики. Алгоритми і виконавці

Завдання

Завдання 1. Лінійні алгоритми

Скориставшись програмою «*Сходинки до інформатики +*» (вкладка *3-4 класи*, *Виконавець* «*Навантажувач*»), написати програму за допомогою якої можна завантажити корабель відповідно до свого варіанту. Налаштувати кількість вантажів і їхню максимальну сумарну вагу за допомогою кнопки *Параметри…*). Створити документ у заголовку якого вказати тему лабораторної роботи і вставити у нього скріншот із розв'язком свого варіанту. На рисунку 2. 5 наведено приклад такої програми для кількості вантажів *3* і максимальної сумарної ваги вантажів *27*.

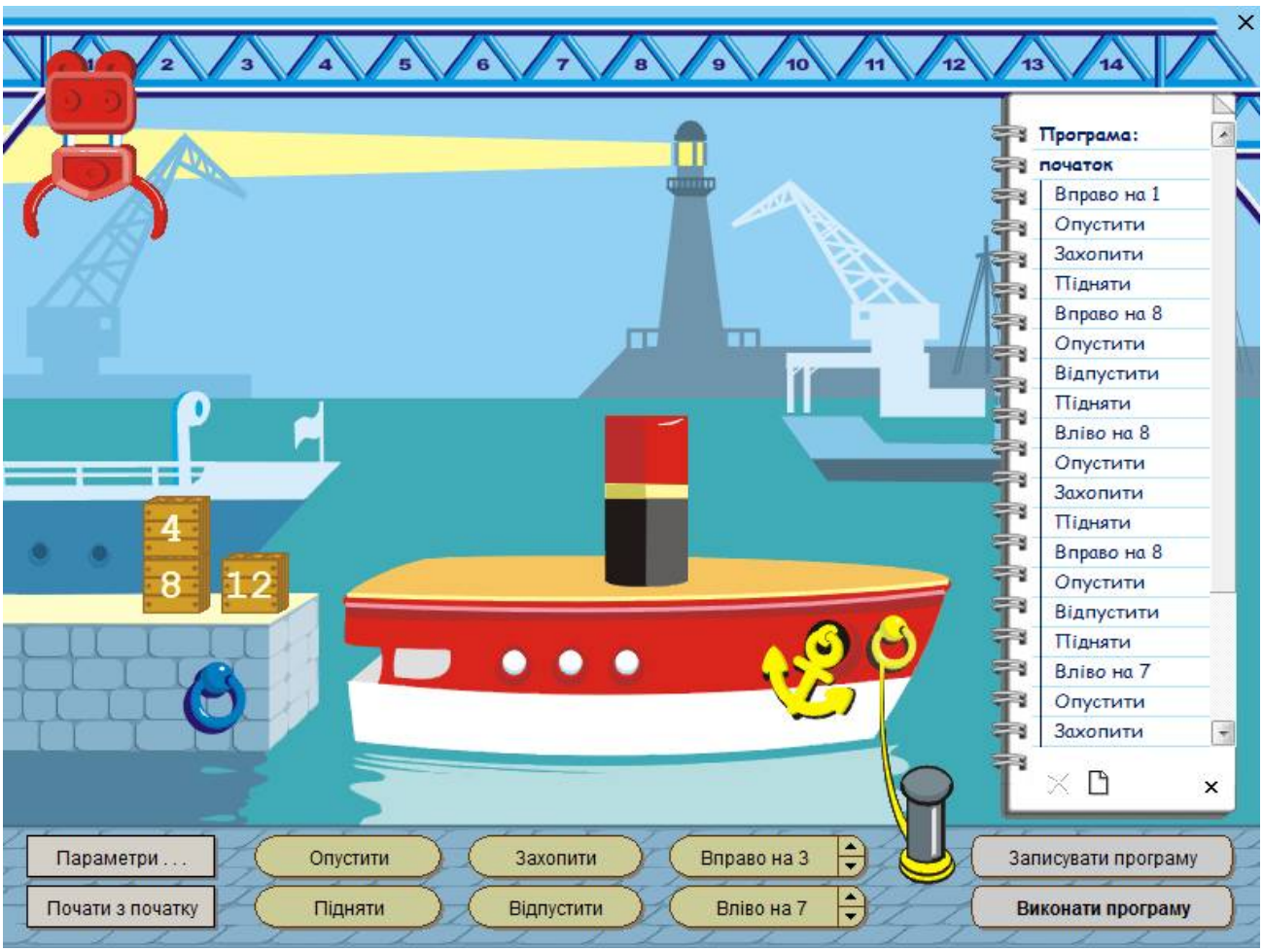


Рис. 2. 5. Приклад лінійної програми

Варіанти до завдання 1

1. Кількість вантажів 5, максимальна сумарна вага вантажів 35.
2. Кількість вантажів 4, максимальна сумарна вага вантажів 30.
3. Кількість вантажів 3, максимальна сумарна вага вантажів 55.
4. Кількість вантажів 5, максимальна сумарна вага вантажів 27.
5. Кількість вантажів 4, максимальна сумарна вага вантажів 88.
6. Кількість вантажів 3, максимальна сумарна вага вантажів 95.
7. Кількість вантажів 5, максимальна сумарна вага вантажів 45.
8. Кількість вантажів 4, максимальна сумарна вага вантажів 58.
9. Кількість вантажів 3, максимальна сумарна вага вантажів 100.
10. Кількість вантажів 5, максимальна сумарна вага вантажів 49.
11. Кількість вантажів 4, максимальна сумарна вага вантажів 45.
12. Кількість вантажів 4, максимальна сумарна вага вантажів 40.
13. Кількість вантажів 3, максимальна сумарна вага вантажів 50.
14. Кількість вантажів 5, максимальна сумарна вага вантажів 60.
15. Кількість вантажів 4, максимальна сумарна вага вантажів 70.

Завдання 2. Алгоритми з розгалуженням і повторенням

Скориставшись програмою «*Сходинки до інформатики* +» (вкладка 3-4 *класи*, *Виконавець «Восьминіжка»*), написати програму за допомогою якої

можна провести восьминіжку з клітинки де вона знаходиться до клітинки, що позначена цифрою 1 відповідно до свого варіанту. По дорозі потрібно перефарбувати у синій колір усі клітинки, зафарбовані жовтим кольором. Створити документ, у заголовку якого вказати тему лабораторної роботи і вставити у нього два скріншоти (до і після виконання програми). На рисунку 2. 6 наведено приклад до виконання програми, а на рисунку 2. 7 - після.

Варіанти до завдання 2

1. рис. 2. 8
2. рис. 2. 9
3. рис. 2. 10
4. рис. 2. 11
5. рис. 2. 12
6. рис. 2. 13
7. рис. 2. 14
8. рис. 2. 15
9. рис. 2. 16
10. рис. 2. 17
11. рис. 2. 18
12. рис. 2. 19
13. рис. 2. 20
14. рис. 2. 21
15. рис. 2. 22

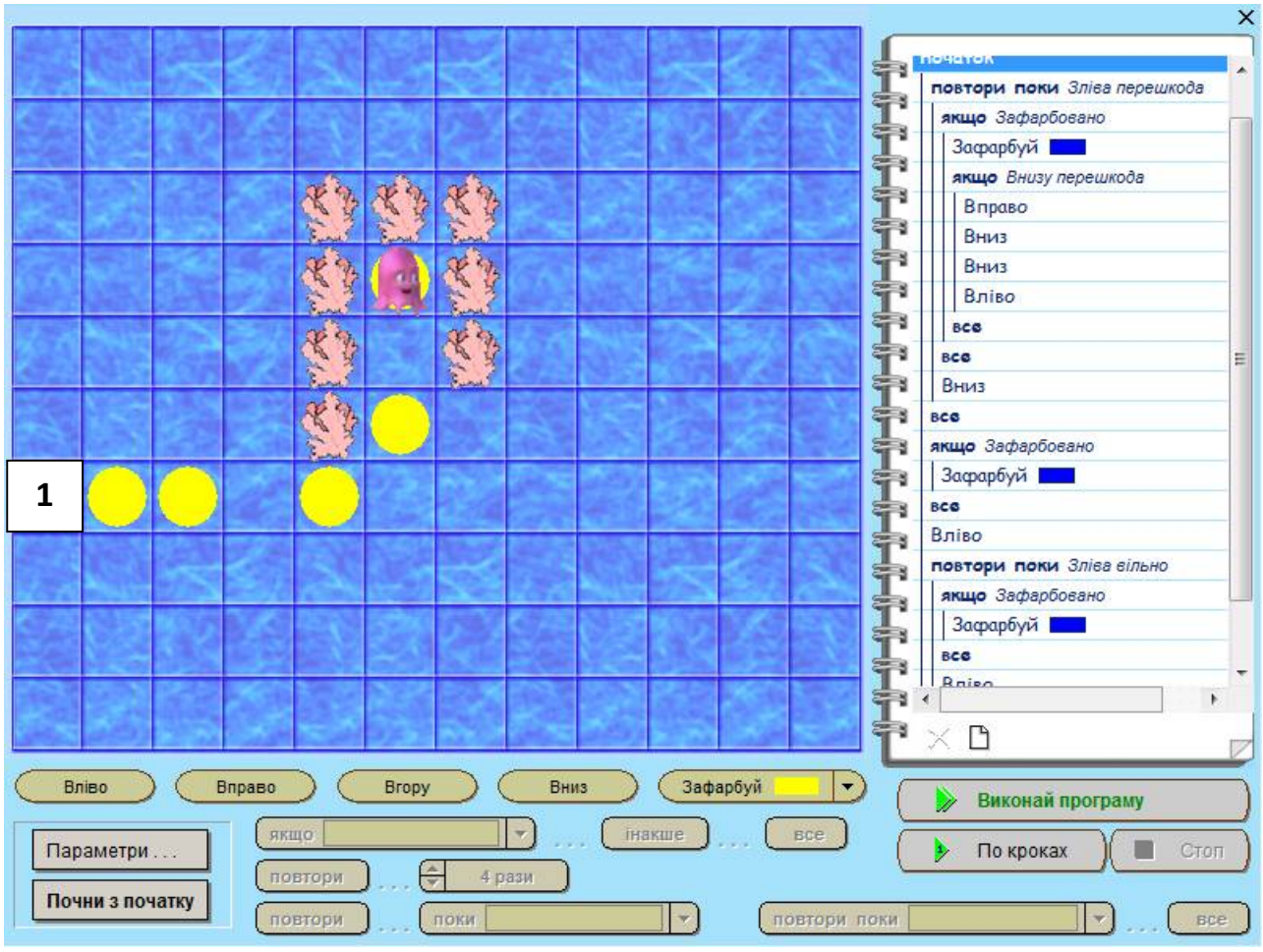


Рис. 2. 6. Приклад програми з розгалуженням і повторенням (до виконання)

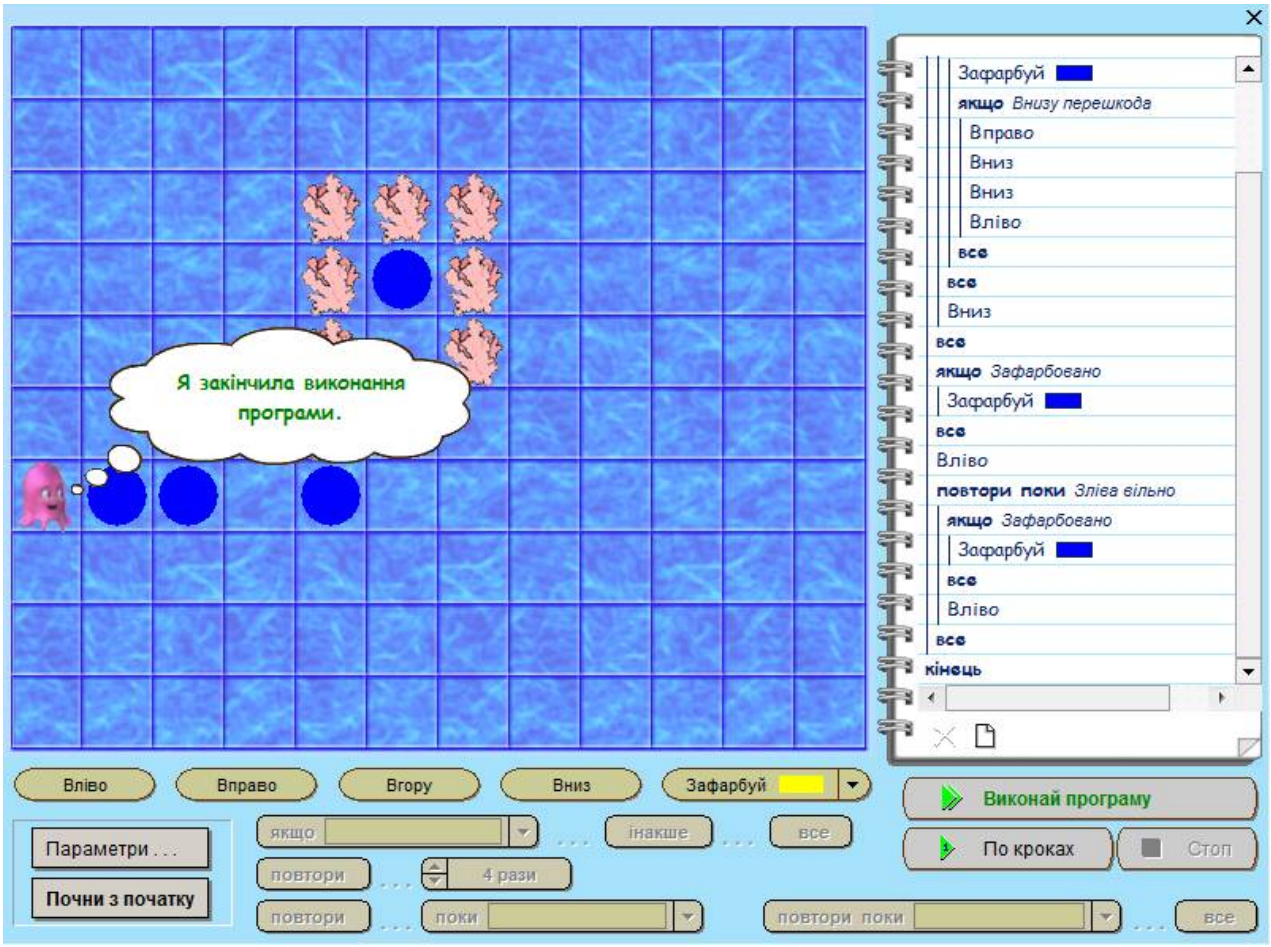


Рис. 2. 7. Приклад програми з розгалуженням і повторенням (після виконання)

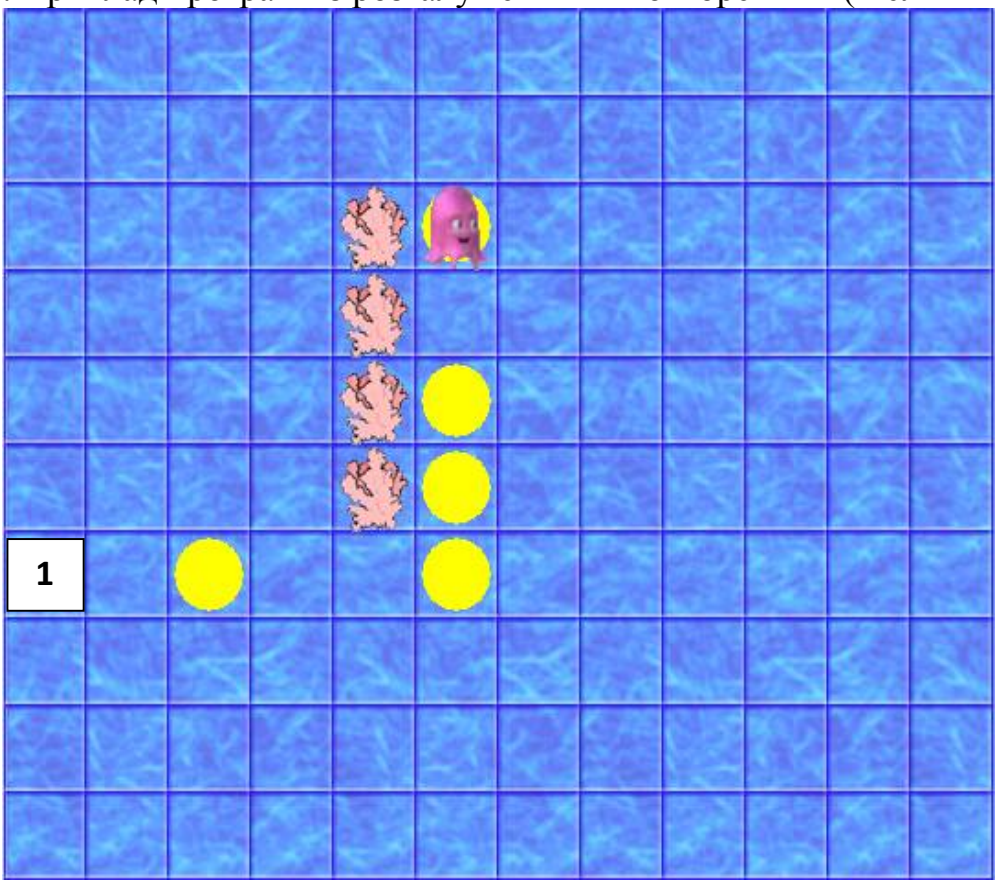


Рис. 2. 8. Вихідні дані (варіант 1)

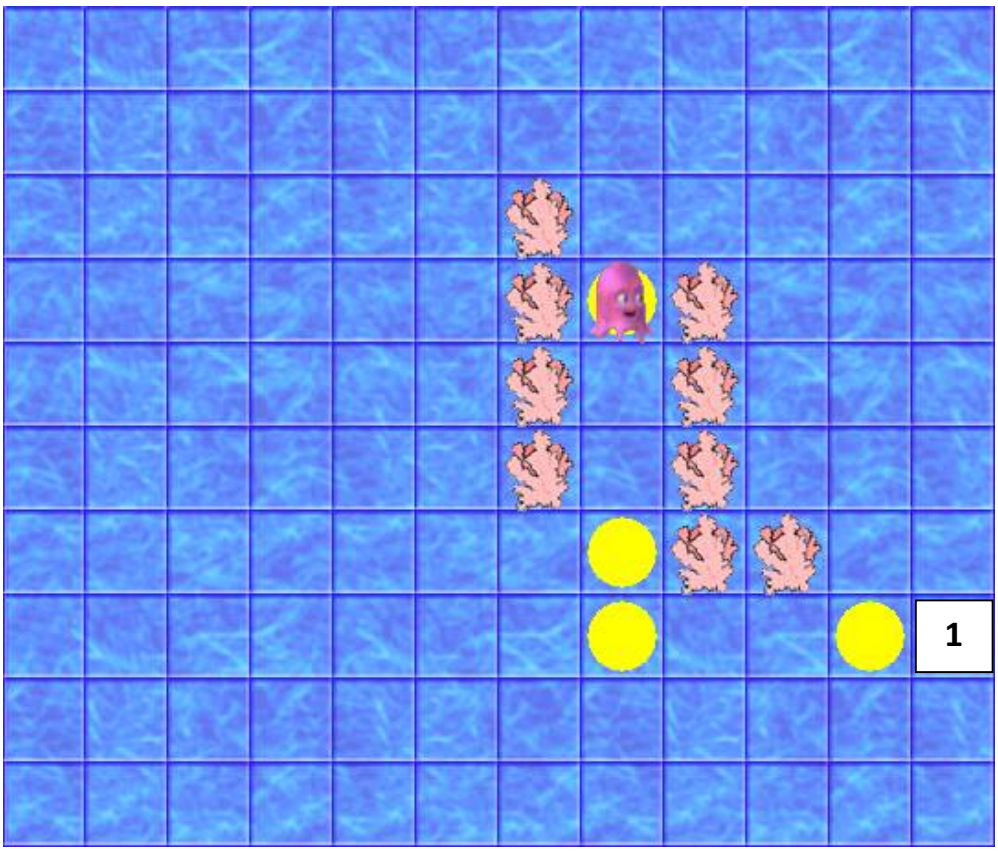


Рис 2. 9. Вихідні дані (варіант 2)

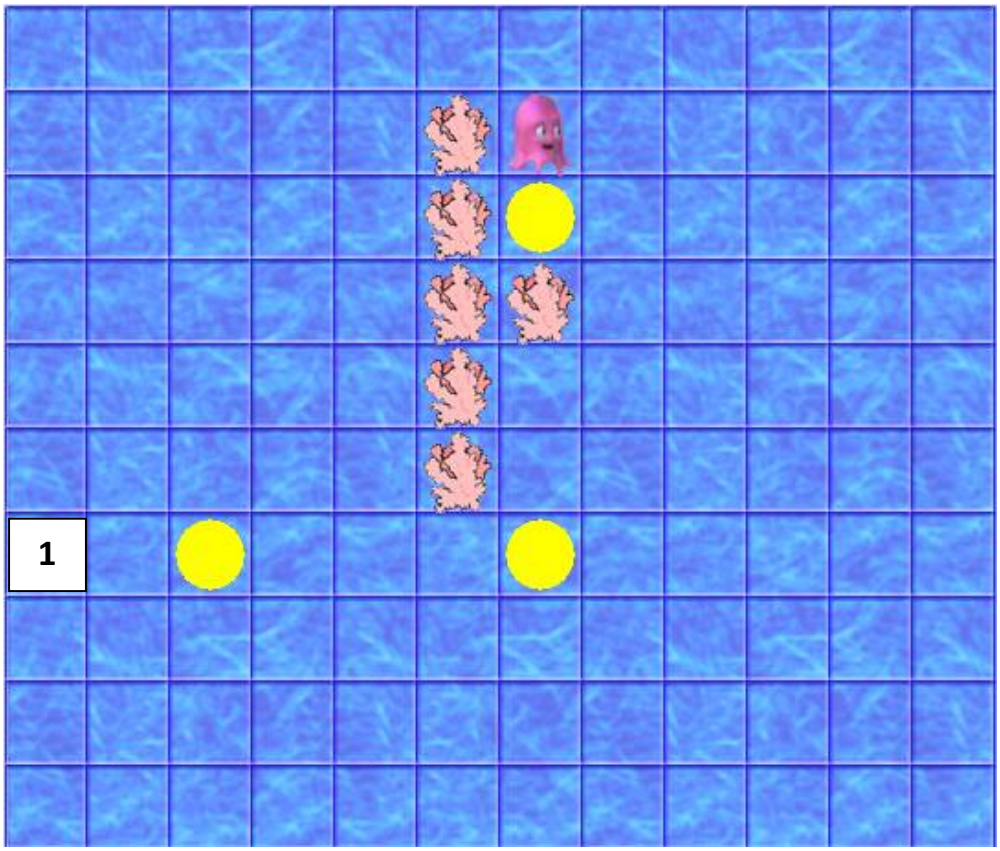


Рис. 2. 10. Вихідні дані (варіант 3)

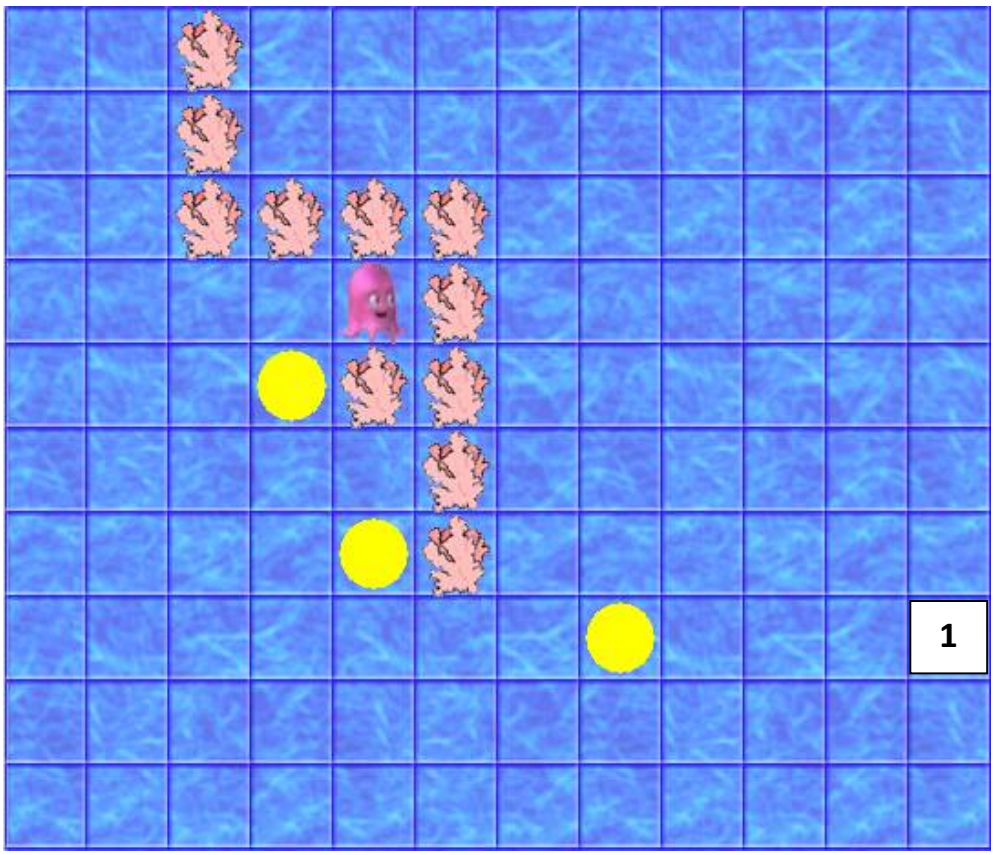


Рис. 2. 11. Вихідні дані (варіант 4)

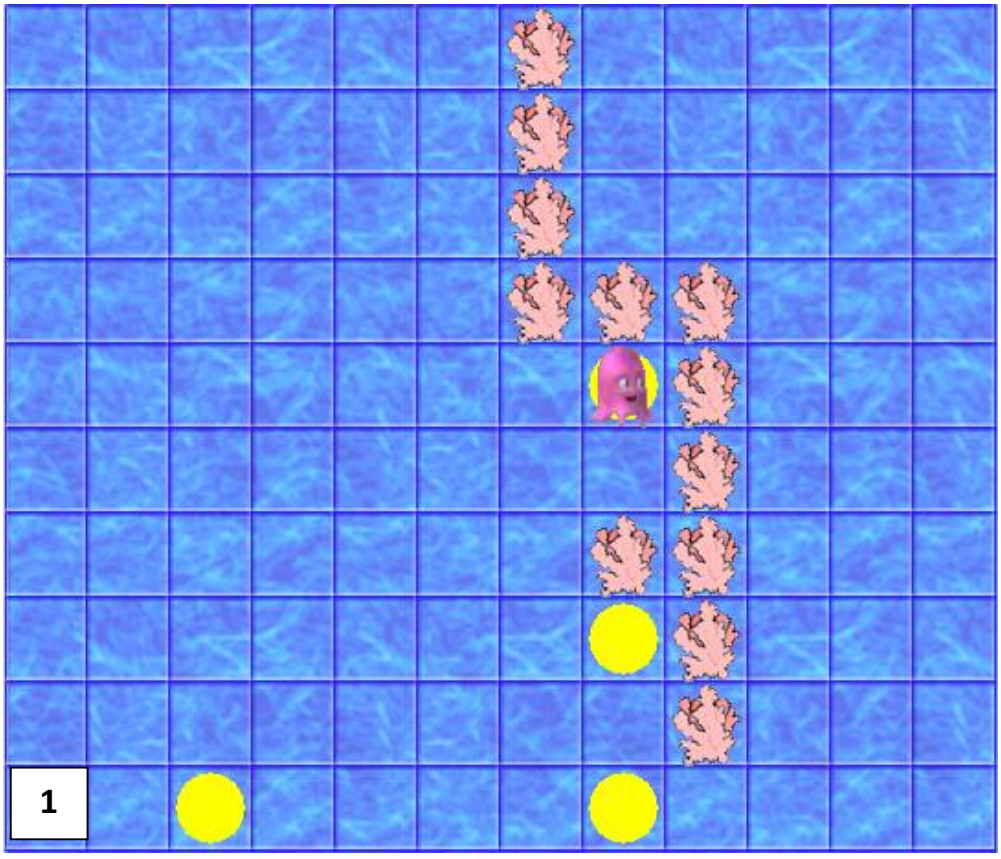


Рис. 2. 12. Вихідні дані (варіант 5)

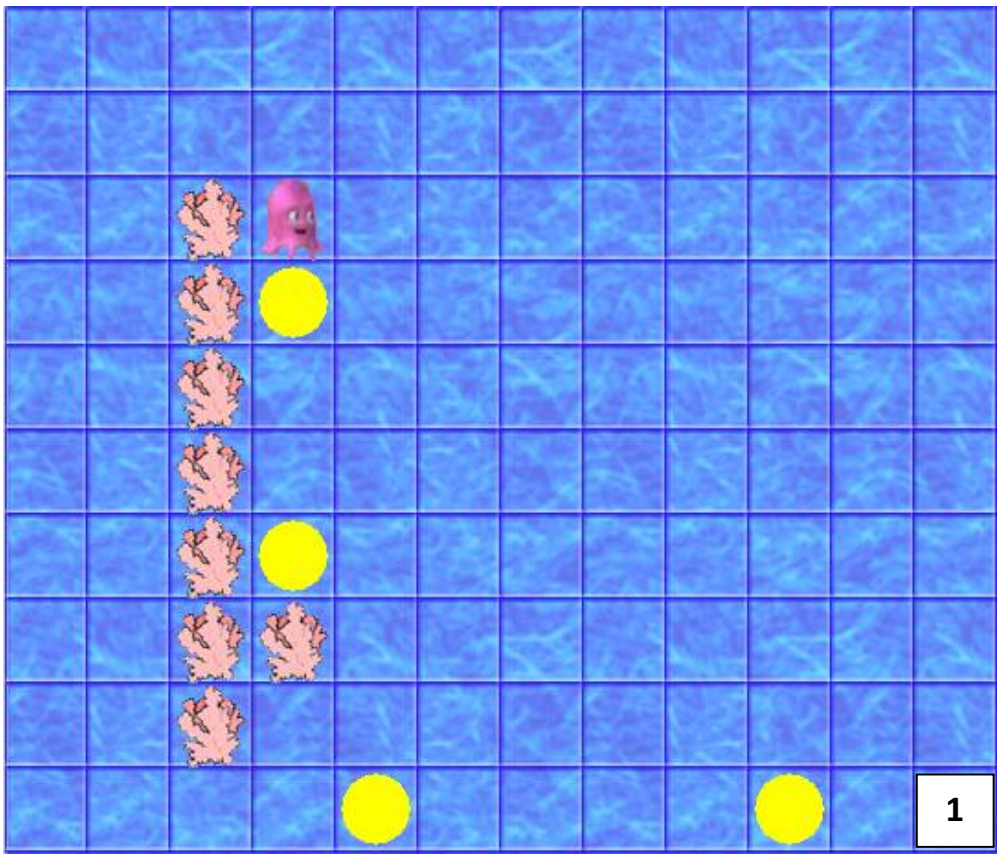


Рис. 2. 13. Вихідні дані (варіант 6)

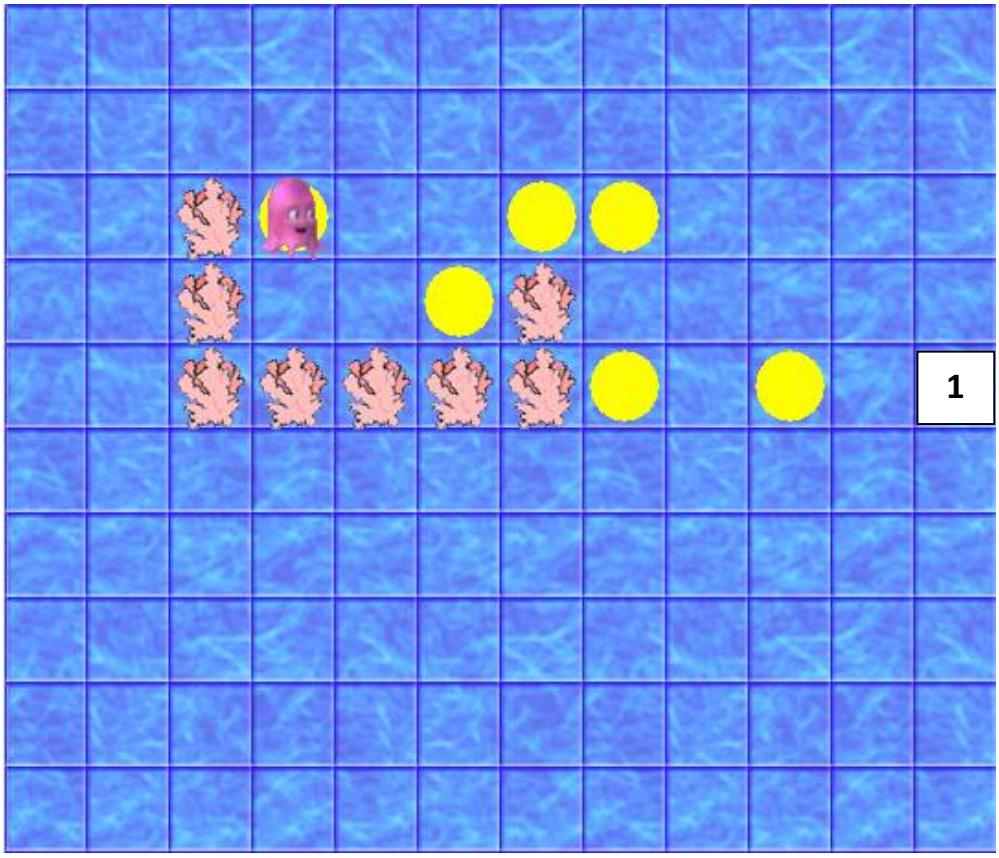


Рис. 2. 14. Вихідні дані (варіант 7)

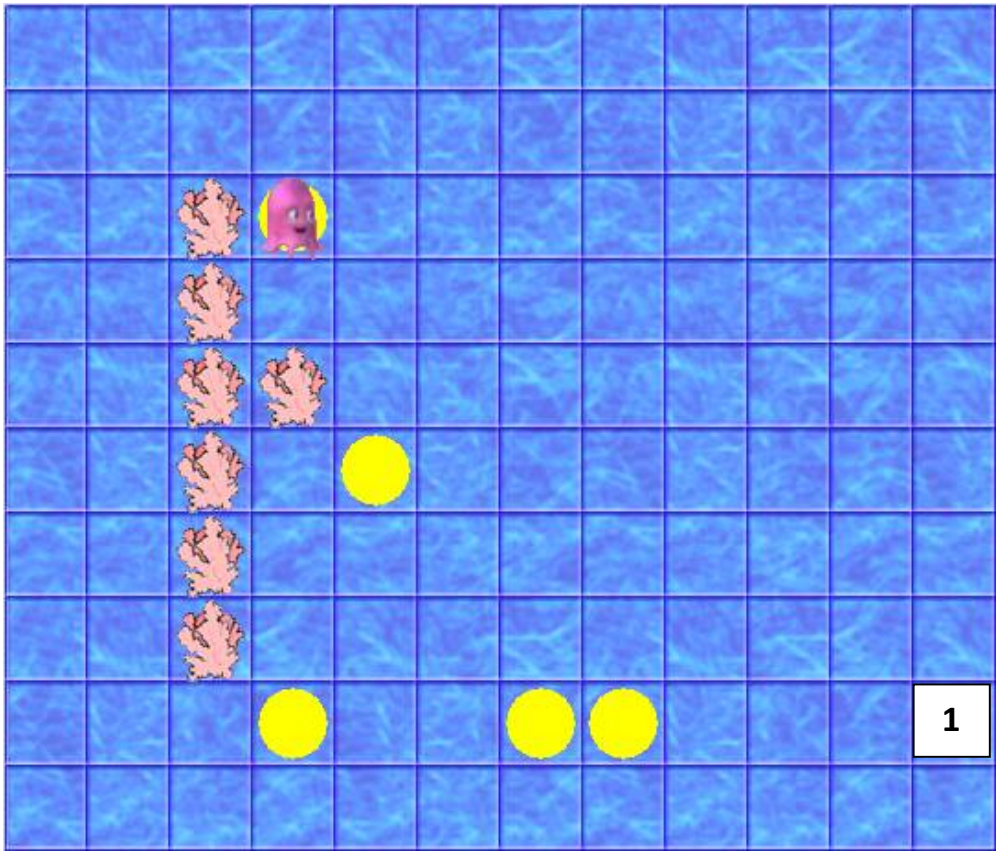


Рис. 2. 15. Вихідні дані (варіант 8)

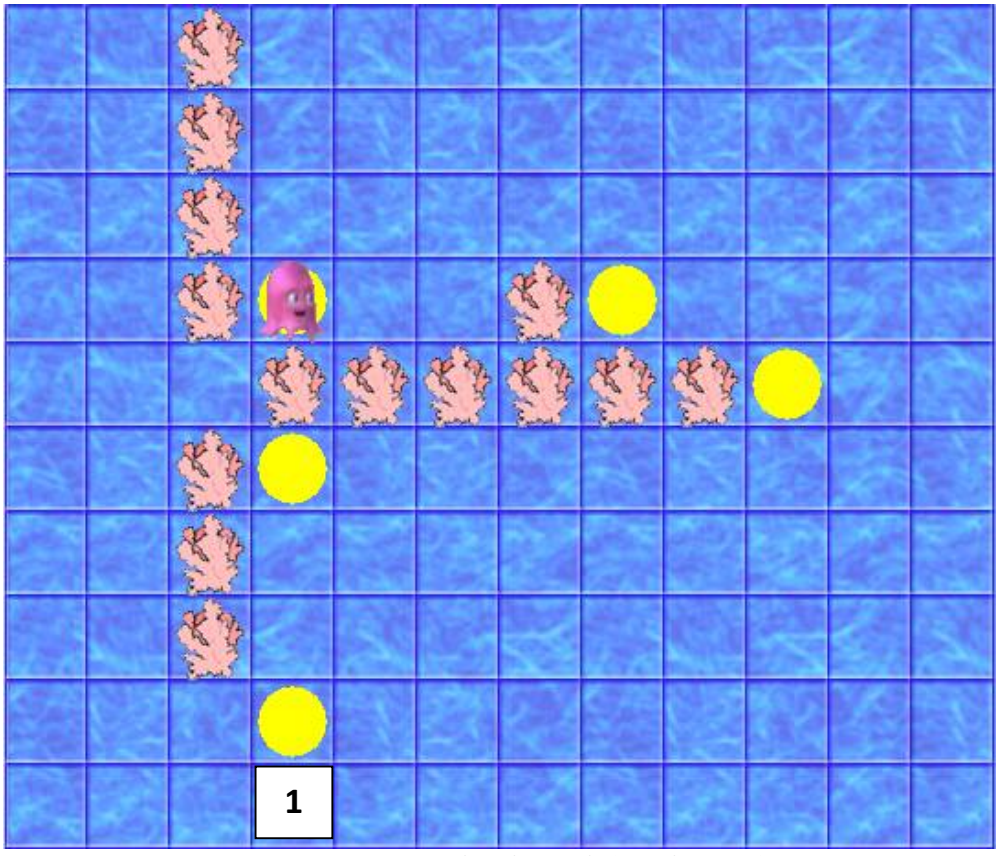


Рис. 2. 16. Вихідні дані (варіант 9)

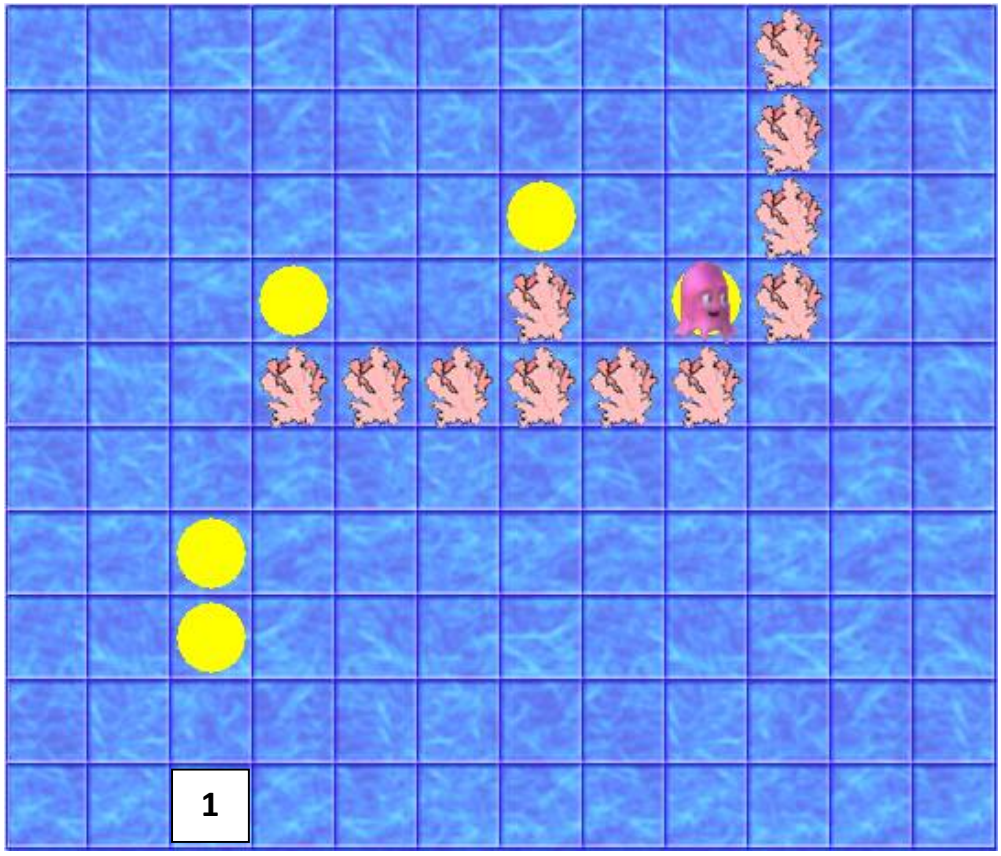


Рис. 2. 17. Вихідні дані (варіант 10)

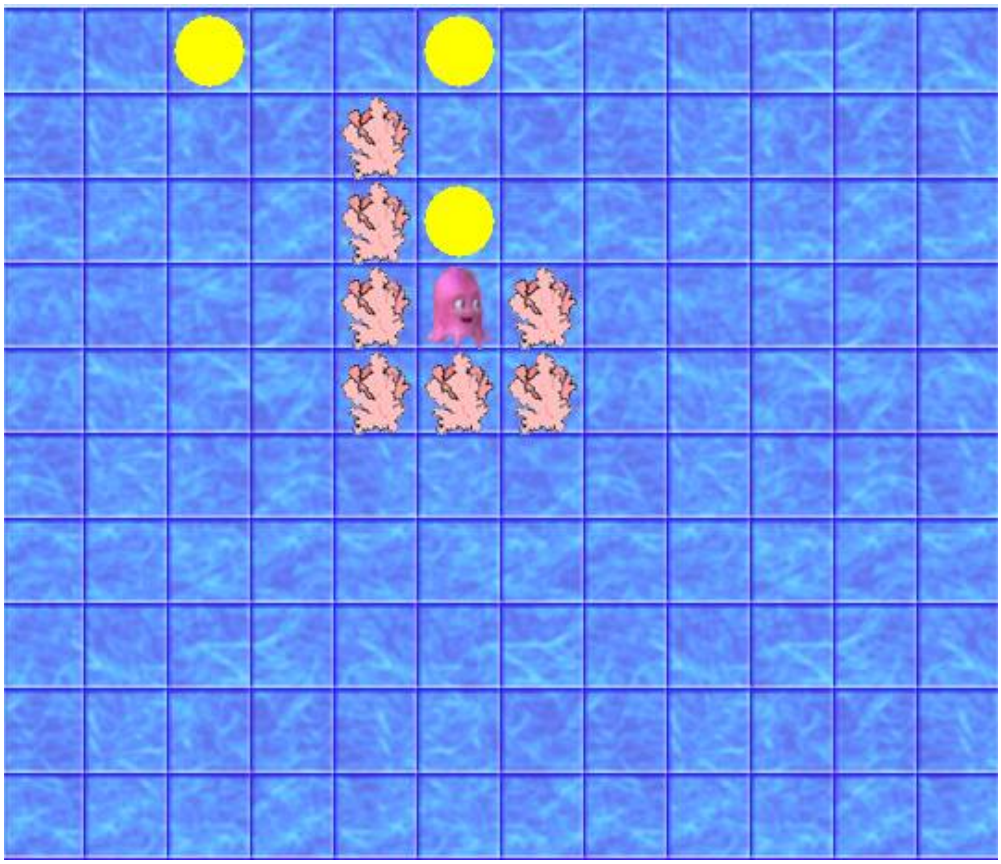


Рис. 2. 18. Вихідні дані (варіант 11)

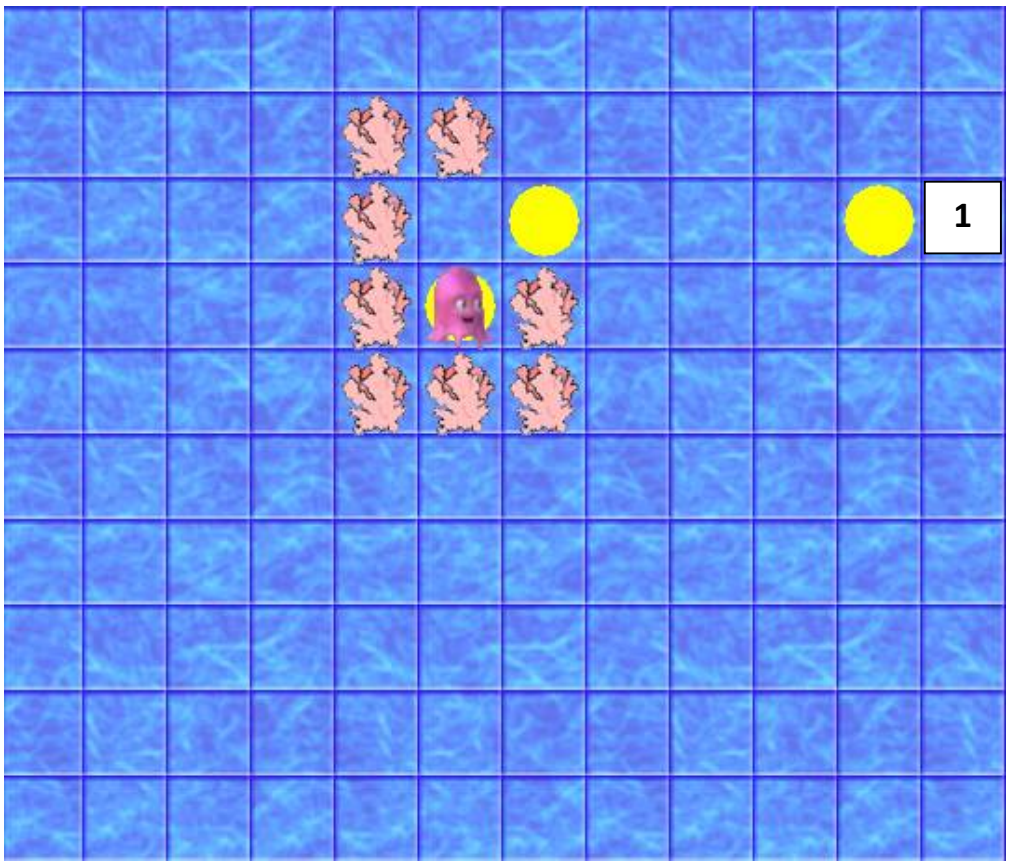


Рис. 2. 19. Вихідні дані (варіант 12)

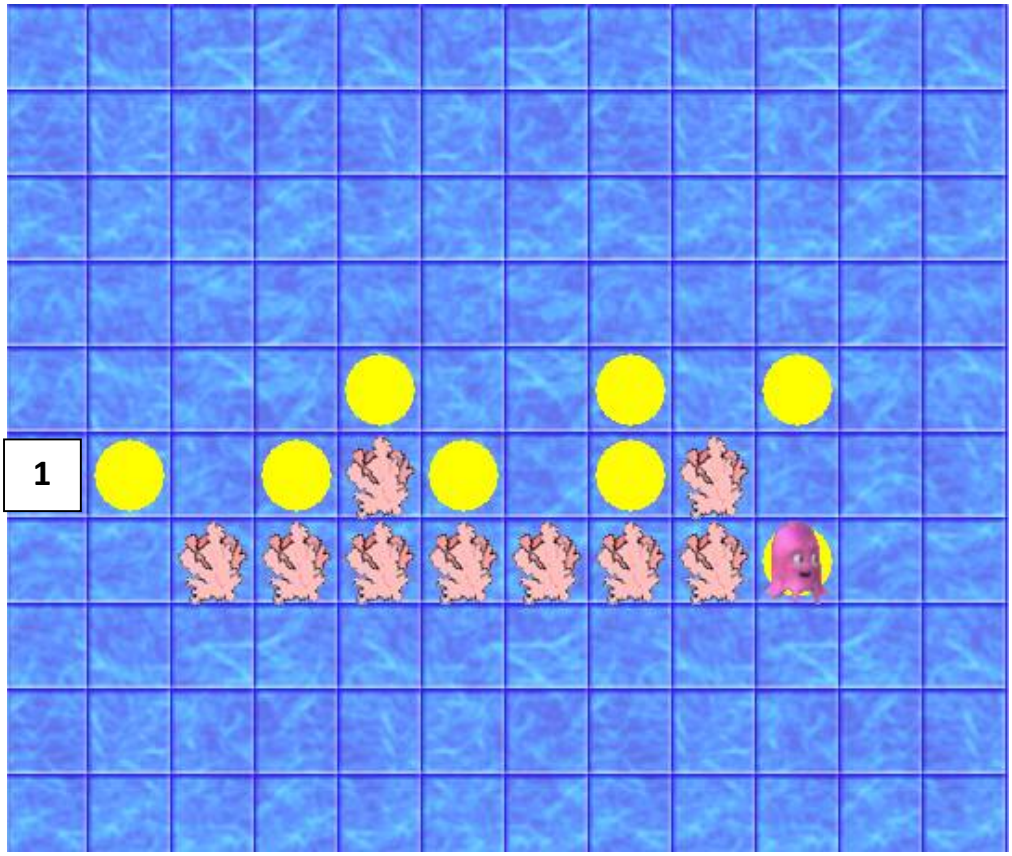


Рис. 2. 20. Вихідні дані (варіант 13)

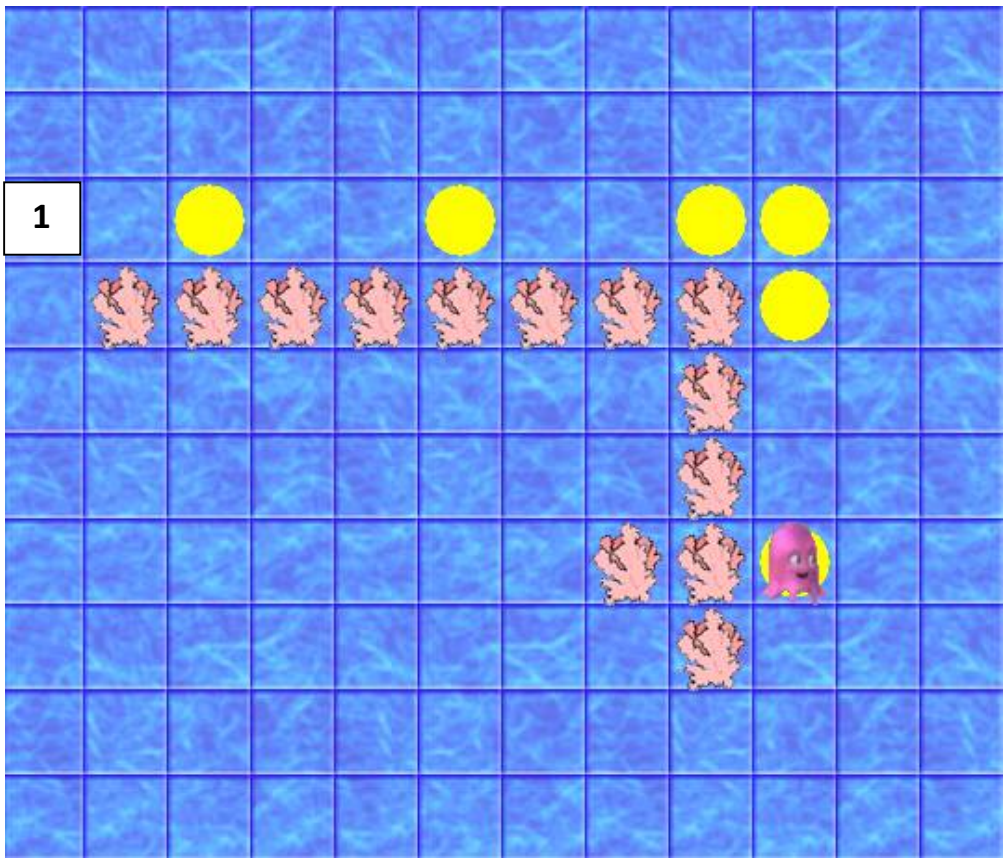


Рис. 2. 21. Вихідні дані (варіант 14)

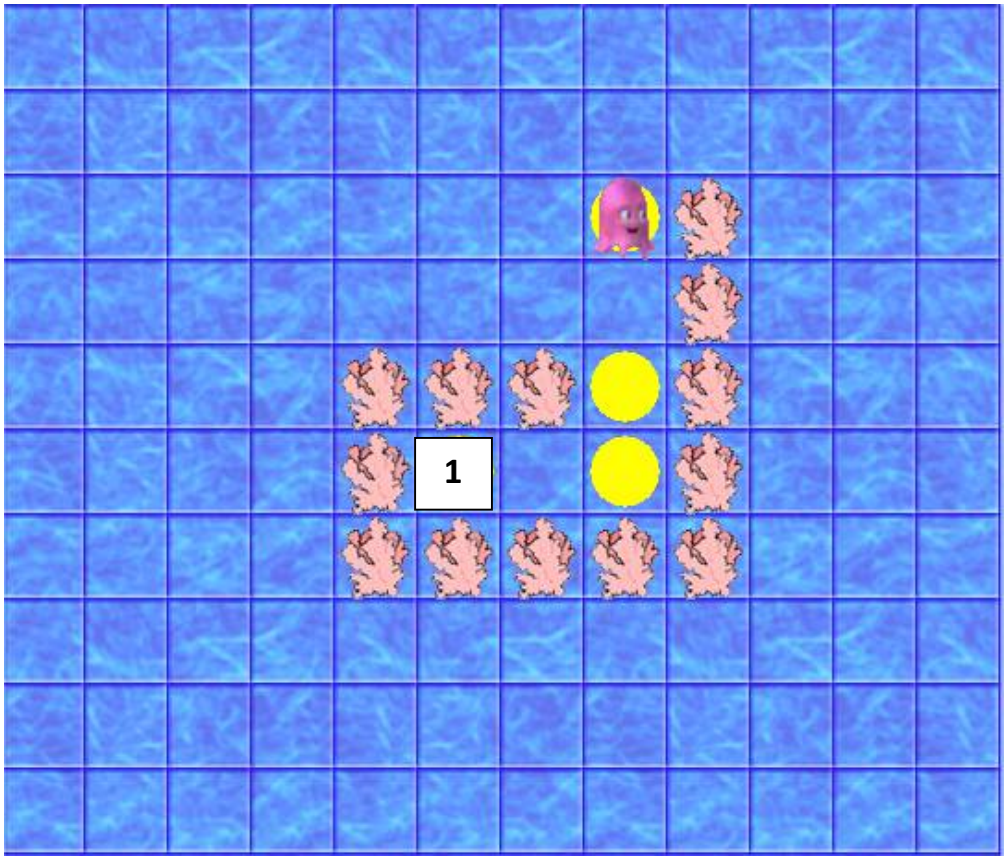


Рис. 2. 22. Вихідні дані (варіант 15)

Контрольні питання

1. Вкладки і виконавці програми «*Сходинки до інформатики+*».
2. Налаштування параметрів програми «*Сходинки до інформатики+*».
3. Лінійні програми.
4. Структури розгалуження.
5. Циклічні структури.

Лабораторна робота 2. 7
Середовище Scratch. Задачі на рух

Завдання

Завдання 1. Рух спрайта по діагоналі

1. Відкрити середовище *Scratch* та ознайомтесь із його об’єктами (головне меню, блоки, закладки, сцена, спрайт та інші).
2. Перемістити спрайт у лівий нижній кут екрана.
3. Встановити напрям руху спрайта, використовуючи кнопку *приймати тільки зліва направо*.
4. Відкрити область скриптів для спрайту та перетягнути блоки з категорій *Керувати* і *Рух* (рис. 2. 23).

Рис. 2. 23. Приклад скрипта до завдання 1

5. Повернути спрайт у вихідне положення та запустити програму на виконання знову. Властивості спрайта відображено на рисунку 2. 24.

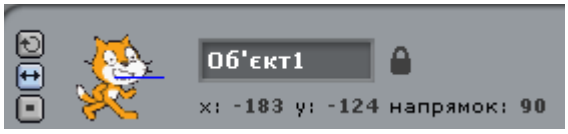


Рис. 2. 24. Властивості спрайта

6. Під час виконання програми перевести проект у режим перегляду (демонстрації), використовуючи кнопки режимів роботи  .
7. Зберегти програму.

Завдання 2. Створення анімованого об'єкту (ілюзії руху) шляхом додавання і зміни костюмів спрайта

1. Запустити програму *Scratch*.
2. Перетягнути спрайт в лівий нижній кут екрана.
3. Встановити напрям руху спрайта в положення зліва направо.
4. Перейти до вкладки *Образи*, переглянути два наявних костюми для кота.
5. Перейти до вкладки *Звуки* та відкрити вікно *Імпортувати звук*, використовуючи кнопку *Імпортувати*.
6. Вибрати в папці *Music Loops* довільну мелодію.
7. В область скриптів перетягнути блоки з командами із контейнерів відповідних кольору категорій (рис. 2. 25).

Примітка. Звернути увагу, що частина скрипту дублює блоки команд. Тому введіть частину команд, потім виберіть із контекстного меню команду «Дублювати» та перетягніть копію частини скрипту у необхідне положення (рис. 2. 26).

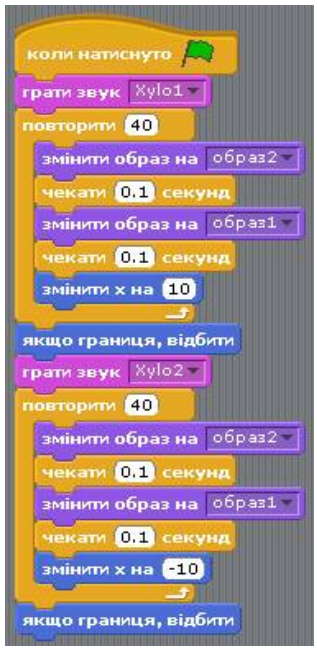
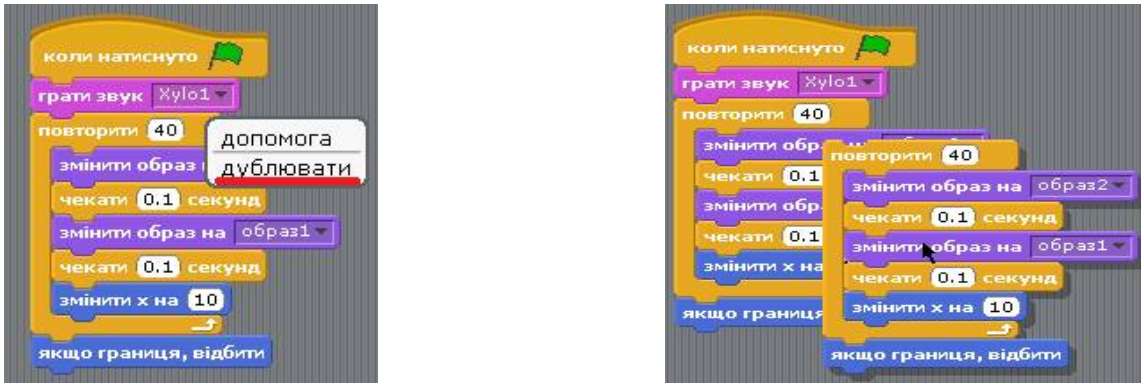


Рис. 2. 25. Приклад скрипта до завдання 2

8. Запустити скрипт на виконання.



а б

Рис. 2. 26. Дублювання блоків команд:
а) вибір команди з контекстного меню; б) перетягування блоків

9. Додати до скрипту кілька блоків команд з довільних категорій.
10. Зберегти програму.

Завдання 3. Швидкий танок на сцені

1. Запустити програму *Scratch*.
2. Вилучити спрайт в образі *Рудого кота*, використовуючи контекстне меню об’єкта (рис. 2. 27).

Рис. 2. 27. Вилучення об’єкта

3. Вибрати новий об’єкт із файлу, скористатись кнопками додавання нових виконавців  .
4. У вікні *Новий об’єкт* відкрити папку *People* та знайти рисунок з ім’ям *cassy-dancing-1*, що з’явиться на місці спрайта 1.
5. Відкрити вкладку *Образи* і додати костюми *cassy-dancing-2* і *cassy - dancing- 3*.
6. Імпортувати мелодію з папки *Music Loops*, обравши вкладку *Звуки*.
7. В області скриптів зібрати блоки з командами із контейнерів відповідних кольору категорій (рис. 2. 28).

Рис. 2. 28. Приклад скрипта до завдання 3

8. Здійснити запуск скрипта.

Примітка. Вдосконалити танок спрайту таким чином, щоб він відбувався на сцені.

9. Вибрати ескіз сцени, що розташований у вигляді білого прямокутника з підписом *Сцена*.

Примітка. Звернути увагу, що вкладки об'єкта: «Скрипти», «Образи», «Звуки» зміняться на вкладки сцени «Скрипти», «Фони», «Звуки».

10. Активізувати вкладку *Фони* та імпортувати рисунок *Spotlight-stage* з папки *Indoors*. Білий фон можна вилучити.
11. Вибрати команду циклу *Завжди* для продовження танку, який можна буде зупинити натисканням на кнопці *Зупинити все* (рис. 2. 29).
12. Зберегти програму.

Рис. 2. 29. Приклад вдосконаленого скрипта до завдання 3

Завдання 4. Керування рухом з клавіатури

Починається задача із заставки.

1. Запустити програму *Scratch*. Змінити образ спрайту з *Рудого кота* на людину та імпортувати другий костюм обраного образу.
2. Встановити довільний фон сцени.
3. В області скриптів зібрати блоки з командами із контейнерів відповідних кольору категорій (рис. 2. 30).

Рис. 2. 30. Приклад скрипта до завдання 4

4. Запустити скрипт та здійснити управління спрайтом за допомогою клавіш у вигляді стрілок вправо/ вліво.
5. Зберегти програму.

Завдання 5. Переміщення рудого кота

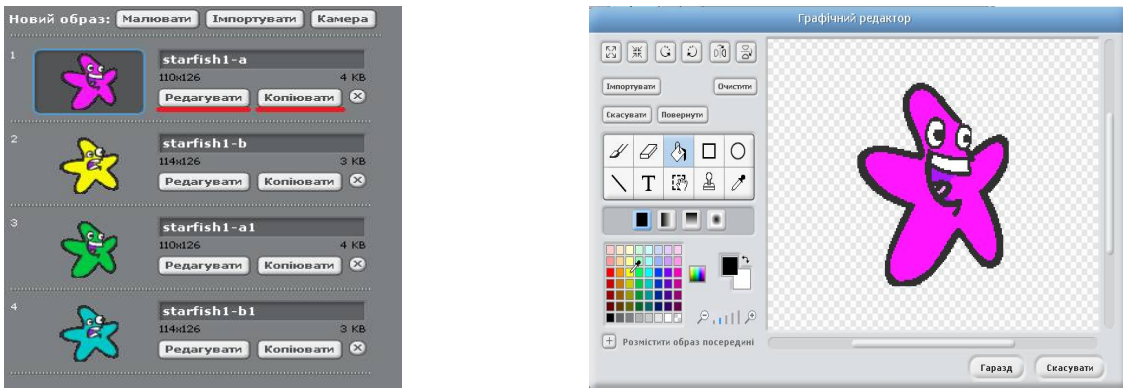
1. Запустити програму *Scratch*.
2. Перетягнути Рудого кота в лівий верхній кут сцени.
3. Подати Рудому коту команду перемістити на 300 кроків.
4. Відкрити групу команд *Вигляд* та подати Рудому коту команду говорити «Привіт» впродовж 4 сек.
5. Подати Рудому коту команду повернути за годинниковою стрілкою на 90 градусів.
6. Подати Рудому коту команду перемістити на 200 кроків.
7. Подати Рудому коту команду говорити «Привіт» впродовж 4 сек.
8. Подати Рудому коту команду повернути за годинниковою стрілкою на 90 градусів.
9. Подати Рудому коту команду перемістити на 300 кроків.
10. Подати Рудому коту команду говорити «Привіт!» впродовж 4 сек.
11. Подати Рудому коту команду повернути за годинниковою стрілкою на 90 градусів.
12. Подати Рудому коту команду перемістити на 200 кроків.
13. Подати Рудому коту команду говорити «Привіт!» впродовж 4 сек.
14. Подати Рудому коту команду повернути за годинниковою стрілкою на 90 градусів.
15. Зберегти програму.

Завдання 6. Рух, зміна кольору, розмова об'єкту

Створити проект, у якому об'єкт рухається, змінює колір і розмовляє. Спрайт обрати відмінний від того, що представлений у прикладі.

Приклад руху *морської зірки*, яка змінює свій колір і вітається

1. Запустити програму *Scratch*.
2. Вилучити з проекту *Рудого кота* та додати до нього новий спрайт *starfish1-a*.
3. Активізувати обраний спрайт та перейти на вкладку *Образи* для додавання нових образів.
4. Використати кнопку *Імпортувати* для додавання нового образу (*starfish1-b*).
5. Скористатися кнопкою *Копіювати* для створення копій образів (*starfish1-a1* та *starfish1-b1*).
6. Скористатись кнопкою *Редагувати* для встановлення кольору заливки відповідних образів спрайту (рис. 2. 31).



а б

Рис. 2. 31. Робота з образами:
а) вкладка «Образи»; б) редагування образу у вікні графічного редактора

7. В області скриптів зібрати блоки з командами із контейнерів відповідних кольору категорій (рис. 2. 32).

Рис. 2. 32. Приклад скрипта до завдання 6

8. Здійснити запуск скрипта (рис. 2. 33).

9. Зберегти програму.

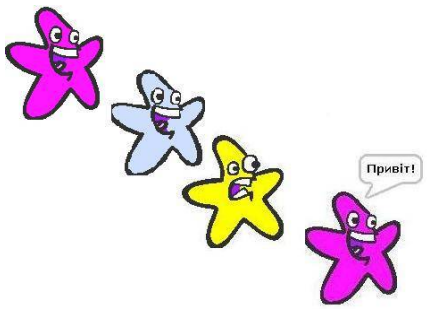


Рис. 2. 33. Покроковий вигляд спрайту на сцені

Контрольні питання

1. Об’єкти середовища *Scratch*.
2. Команди та їх призначення за групами.
3. Зміна спрайту.
4. Робота з образами.
5. Дублювання блоків команд та їх вилучення.

Лабораторна робота 2. 8
Анімування об’єктів у у середовищі Scratch. Створення та виконання алгоритмів малювання

Завдання

Завдання 1. Створення анімованих об’єктів

1. В середовищі *Scratch* створити анімований об’єкт, що відповідає варіанту завдання.
2. Додати до об’єкта не менше трьох образів, які відповідають різним стадіям анімування. Наприклад, різні варіанти посмішки або підморгування смайлика, зміна кольору образа об’єкта інше (рис. 2. 34).

Рис. 2. 34. Образи спрайту

3. Створити скрипт для анімації послідовно змінюючи образи об’єктів з невеликою затримкою часу. Для цього в тілі циклу задати перехід на наступний образ.
4. Запустити скрипт на виконання та зберегти файл проекту.

Варіанти завдання 1

1. 

2. 

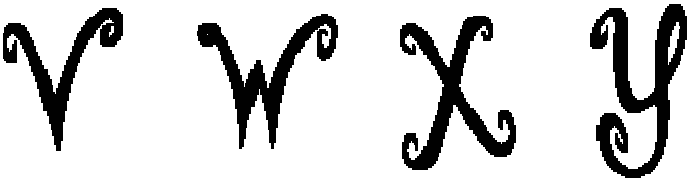
3. 

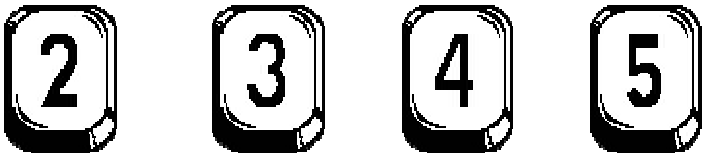
4. 

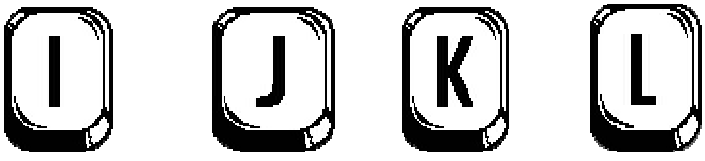
5. 

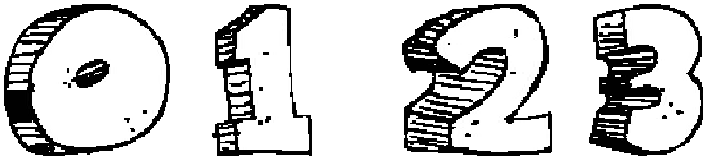
6. 

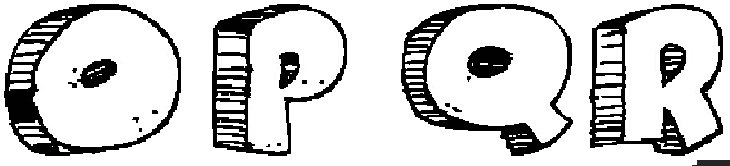
7. 

8. 

9. 

10. 

11. 

12. 

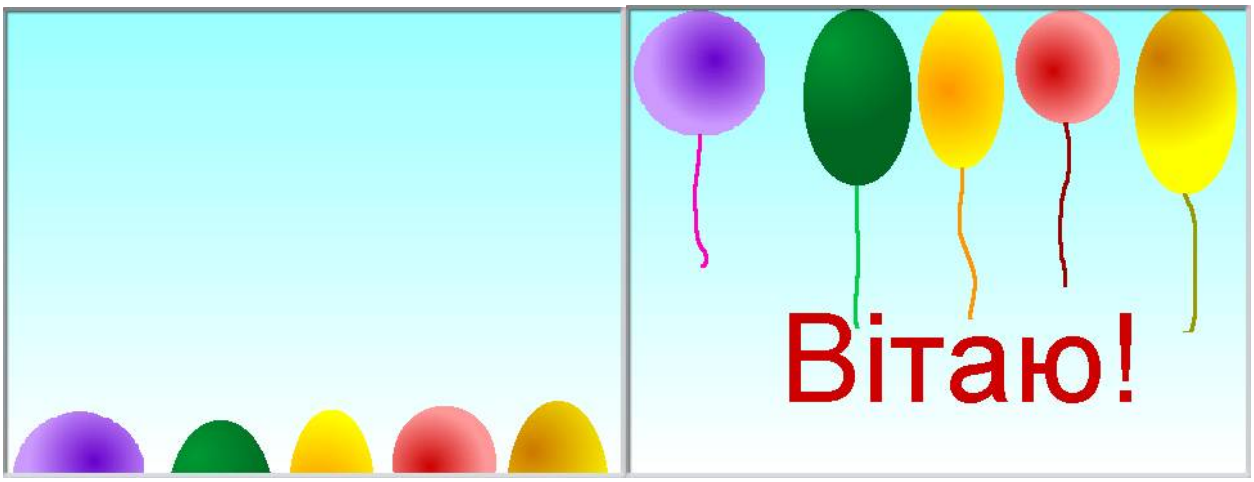
13. 

14.

15. 

Завдання 2. Виготовлення анімаційної вітальної листівки зі святом

Вигляд вітальної листівки на етапі відображення вихідних даних та результату виконання програми представлено на рисунку 2. 35. Кожна кулька рухається тільки після натиснення на неї мишею. Після того, як всі кульки піднімуться догори, повинен з'явитися надпис «Вітаю!» з ефектом зміни кольору тексту.



а б

Рис. 2. 35. Приклад анімаційної листівки:
а) початкове зображення; б) результат виконання

1. Запустити програму *Scratch* та створити анімаційну вітальну листівку, що відповідає варіанту завдання.
2. Створити п'ять спрайтів: чотири кульки, надпис «*Вітаю!*» та змінити сцену.
3. Для кожної кульки створити відповідний їй скрипт, що перемістить кульку у вихідну позицію (рис. 2. 36).

Рис. 2. 36. Переміщення спрайту в точку із заданими координатами

4. Для кожного спрайту створити відповідний йому скрипт, який при натисненні на кульці вказівником миші перемістить її вгору (в точку із заданими координатами) (рис. 2. 37).

Примітка. В умові зазначено, що слово «Вітаю!» повинно з'явитися після того, як вилетять усі кульки. Тому необхідно ввести лічильник кульок,

створивши змінну «Пуск» та додати до скрипта спрайту команду збільшення значення лічильника на 1.

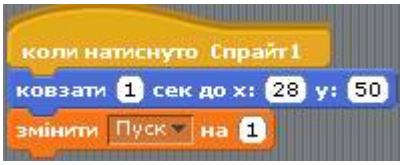


Рис. 2. 37. Переміщення спрайту вгору та зміна значення лічильника

5. Скопіювати необхідні скрипти для інших спрайтів.

Примітка. На початку роботи зі скриптом змінну «Пуск» необхідно обнулити і змінити її значення на 1.

6. В області скриптів додати команди для спрайту тексту «*Вітаю!*» (рис. 2. 38).

Примітка. Спрайт «Вітаю!» спочатку треба сховати і показати за умови піднятих усіх кульок догори (значення лічильника дорівнюватиме п'яти). Багаторазову зміну кольору тексту можна забезпечити командою «Завжди».

Рис. 2. 38. Скрипт спрайту тексту «Вітаю»

Запустити скрипт на виконання та зберегти файл проекту.
Створити вітальну листівку відповідно до варіанту (табл. 2. 10).

Табл. 2.10

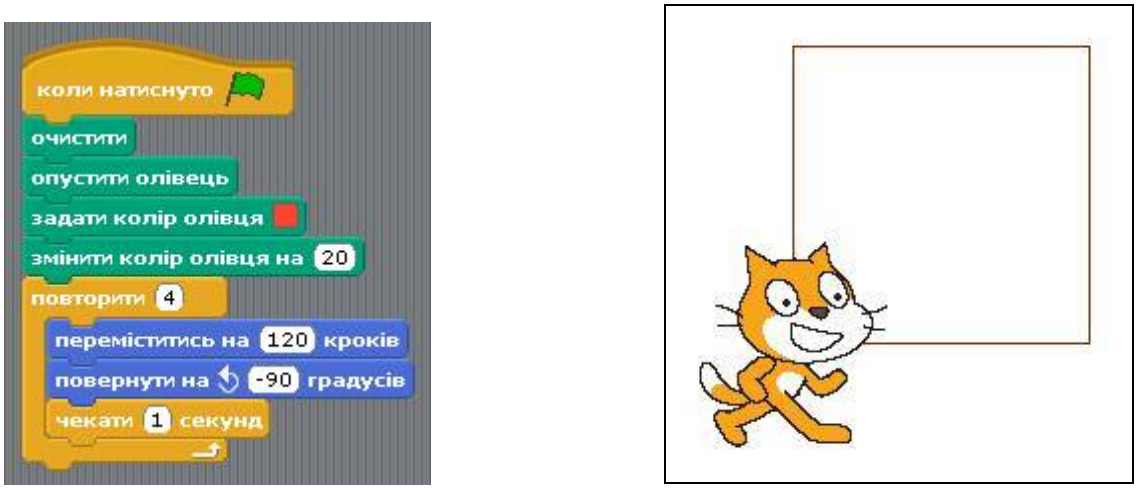
Варіанти до завдання 2

Варіант	**Свято**	**Ефект тексту**
1	Новий рік	колір
2	Святий Вечір	вздуття
3	Різдво	обертання
4	Старий Новий рік (Щедрий вечір)	пікселями
5	День святого Валентина (закоханих)	мозаїка
6	Міжнародний жіночий день	яскравість
7	День сміху	привид
8	Великдень або Пасха	колір
9	День Конституції України	вздуття
10	Свято Івана Купала	обертання
11	День Незалежності України	пікселями

12	День знань	мозаїка
13	День працівників лісу	яскравість
14	День працівників освіти	привид
15	День студента	колір

Завдання 3. Побудова квадрату

1. Запустити програму *Scratch* та перемістити спрайт *Рудого кота* у верхній лівий кут екрану.
2. Для спрайту створити скрипт побудови квадрату із довжиною сторін 120 кроків, використовуючи вказівки керування, руху, малювання та повторення (рис. 2. 39).



а б

Рис. 2. 39. Приклад побудови квадрату
а) скрипт побудови квадрату; б) результат запуску скрипта

Запустити скрипт на виконання та зберегти файл проекту.

Завдання 4. Створення орнаменту

1. Запустити програму Scratch та створити орнамент, що відповідає варіанту завдання (табл. 2. 11).
2. Визначити елемент малюнка, що повторюється, кількість повторень, довжини відрізків (рис. 2. 40).

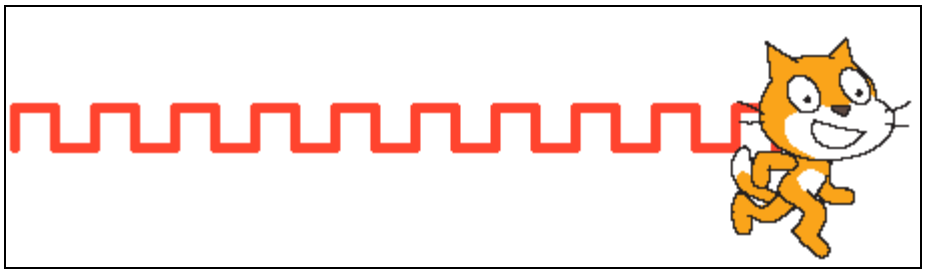


Рис. 2. 40. Вигляд орнаменту з довжиною відрізка 20 кроків

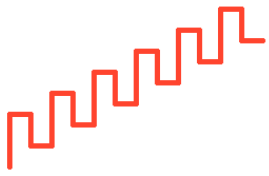
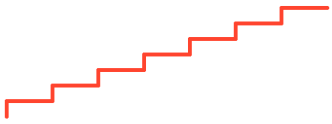
3. Зібрати скрипт для побудови відповідного малюнку, використати вказівки початку програми, переміщення спрайту *Рудого кота* у початкове положення (точка сцени з координатами (–230; 0)), очищення сцени та команди з категорії *Олівець*. Послідовність вказівок для створення елемента вкласти в цикл із лічильником (рис. 2. 41).

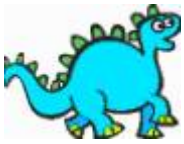
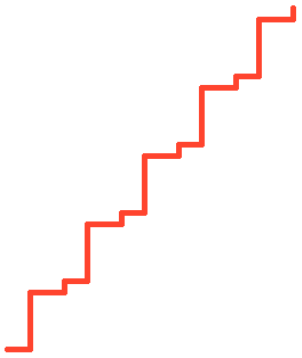
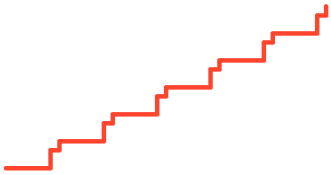
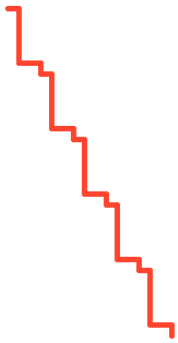
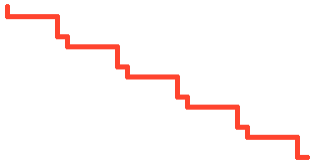
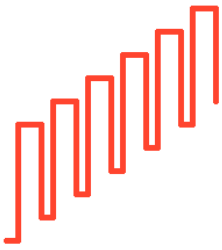
4. Запустити скрипт на виконання та зберегти файл проекту.

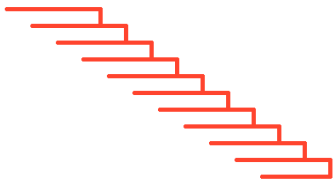
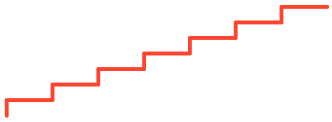
Рис. 2. 41. Приклад скрипта до завдання 4

Табл. 2. 11

Варіанти завдання 4

Варіант	**Спрайт**	**Сцена**	**Орнамент**	**Довжини відрізків**
1		hill		50 20 30 20
2		flowers		10 30
3		woods-and-bench		20 20 40
4		tree		20 20 40

Варіант	Спрайт	Сцена	Орнамент	Довжини відрізків
5		graffiti		20 50 30 10
6		room2		50 20 10 10
7		woods		10 50 20 10
8		desert		10 50 20 10
9		room1		10 50 20 50
10		flower-bed		10 100 20 80

Варіант	Спрайт	Сцена	Орнамент	Довжини відрізків
11		school1		100 20 80
12		spotlight-stage		10 40 10 20
13		canyon		15 50
14		school2		20 20 40
15		stars		20 20 40 20

Контрольні питання

1. Об’єкти середовища *Scratch*.
2. Команди та їх призначення за категоріями.
3. Створення анімованих об’єктів.
4. Організація розгалуження та циклічної структури.
5. Робота з лічильником.

Література

1. Верлань А.Ф., Апатова Н.В. Інформатика: Підруч. для учнів 10–11 кл. серед. загальноосвіт. шк. – К.: Форум, 2001. – 255 с.
2. Глинський Я.М. Практикум з інформатики. Навч. посібник. 5-е вид. – Львів: Деол, 2002.
3. Глинський Я.М. Інформатика: 8–11 класи. Навч. посібник для загальноосвітніх навчальних закладів: У 2-х кн. – Кн. 2. Інформаційні технології. 2-е вид. – Львів: “Деол”, 2002. – 256 с.
4. Жалдак М.І., Рамський Ю.С. Інформатика: Навч. посібник / За ред. М.І. Шкіля. – К.: Вища шк., 1991. – 319 с: іл.
5. Зарецька І.Т., Гуржій А.М., Соколов О.Ю. Інформатика: Підручник для 10 – 11 кл. загальноосвіт. навч. закладів. У 2-х част. – К.: Форум, 2004. – 392 с. іл. Ч. 1.
6. Зеленяк О.П. Программирование в среде Turbo Pascal. – Александрия, 1999. – 308 с.
7. Информатика. Базовий курс /Симонович С.В. и др. – СПб.: Издательство «Питер», 2003. – 640с.: ил.
8. Інформатика: підруч. для 6-го кл. загальноосвіт. навч. закл. / Й. Я. Ривкінд [та ін.]. – К. : Генеза, 2014. – 256 с. : іл.
9. Інформатика: підруч. для 7-го кл. загальноосвіт. навч. закл. / Й. Я. Ривкінд [та ін.]. – К. : Генеза, 2015. – 240 с. : іл.
10. Інформатика. Комп’ютерна техніка. Комп’ютерні технології: Підручник / Баженов В.А. та ін. – К.: Каравела, 2004. – 464 с.
11. Інформатика: 11 кл.: підруч. для загальноосвіт. навч. закл.: академічний рівень: профільний рівень / Й.Я.Ривкінд, Т.І.Лисенко, Л.А.Чернікова, В.В.Шакотько; за заг. ред. М.З. Згуровського. – К.: Генеза, 2011. – 304 с.: іл.
12. Інформатика: підруч. для 11 кл. загальноосвіт. навч. закл.: рівень стандарту / Н.В.Морзе, О.В.Барна, В.П.Вембер [та ін.]. – К.: Школяр, 2011. – 304 с.: іл.
13. Караванова Т.П. Інформатика: основи алгоритмізації та програмування: 777 задач з рек. та прикл.: Навч. Посіб. / За заг. ред. М.З.Згуровського – К. : Генеза, 2006. – 286 с.
14. Ковалюк Т.В. Основи програмування. – К. : BHV, 1997. – 384 с.
15. Коршунова О. В. Інформатика : підруч. для 4-го кл. загальноос-віт. навч. закл. / О.В. Коршунова. — Київ : Генеза, 2015. — 176 с.: іл.
16. Коршунова О. В. Сходинки до інформатики : підруч. для 3-го кл. загальноосвіт. навч. закл. / О.В. Коршунова. – К. : Генеза, 2014. – 176 с.
17. Коффман Э. Turbo Pascal. – М. : Вильямс, 2002. – 896 с.
18. Ломаковська Г.В. Інформатика: підруч. для 4 кл. загальноосвіт. навч. закладів / Г.В. Ломаковська, Г.О. Проценко, Й.Я.Ривкінд, Ф.М. Рівкінд. – К. : Видавничий дім «Освіта», 2015. – 160 с.

19. Ломаковська Г.В. Сходинки до інформатики: підруч. для 3 кл. загальноосвіт. навч. закладів / Г.В. Ломаковська, Г.О. Проценко, Й.Я.Ривкінд, Ф.М. Рівкінд . – К. : Видавничий дім «Освіта», 2013. – 160 с.
20. Лупан І.В., Присяжнюк О.В., Копотій В.В. Лабораторний практикум з дисципліни «Інформатика» для студентів спеціальності «Математика». Частина 3. – Кіровоград: КДПУ ім. В.Винниченка, 2005. – 88 с.
21. Марченко А.И., Марченко Л.А. Borland Pascal 7.0 / Учебное пособие. – К., «ЮНИОР», 1995. - 480 с
22. Морзе Н.В. Методика навчання інформатики: Навч. посіб.: У 4 ч. / За ред. акад. М.І. Жалдака. – К. : Навчальна книга, 2003. – Ч. IV: Методика навчання основ алгоритмізації та програмування. – 368 с.
23. Паращук С.Д., Троценко З.М. Програмування мовою Паскаль. Лабораторний практикум: Навчальний посібник. – Кіровоград: Авангард, 2008.- 159 с.
24. Пильщиков В.Н. Упражнения по языку Паскаль. – М. : Изд–во МГУ, 1986.
25. Рындак В. Г , Дженжер В.О., Денисова Л.В. Проектная деятельность школьника в среде программирования Scratch. Учебно-методическое пособие. - Оренбург, 2009, 116 с.

ОСНОВИ ІНФОРМАТИКИ З ЕЛЕМЕНТАМИ ПРОГРАМУВАННЯ ТА СУЧАСНІ ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ НАВЧАННЯ

Частина ІІ

Елементи програмування

Ганжела Сергій Іванович

Шлянчак Світлана Олександрівна

СВІДОЦТВО ПРО ВНЕСЕННЯ СУБ'ЄКТА ВИДАВНИЧОЇ СПРАВИ
ДО ДЕРЖАВНОГО РЕЄСТРУ ВИДАВЦІВ,
ВИГОТІВНИКІВ І РОЗПОВСЮДЖУВАЧІВ ВИДАВНИЧОЇ ПРОДУКЦІЇ
Серія **ДК** № **1537** від **22.10.2003 р.**

Підп. до друку 20.04.2017. Формат 60×90 $^{1}/_{16}$. Папір офсет. Друк різограф.
Ум. др. арк. 3,2. Тираж 100. Зам. № 8461.

РЕДАКЦІЙНО-ВИДАВНИЧИЙ ВІДДІЛ
Центральноукраїнського державного педагогічного університету імені Володимира Винниченка
25006, Кіровоград, вул. Шевченка, 1
Тел.: (0522) 24-59-84.
Fax.: (0522) 24-85-44.
E-Mail: mails@kspu.kr.ua