§ 62. Цикл із лічильником

Вивчивши цей параграф, ми:

познайомимося з циклами, які виконуються наперед задану кількість разів;

дізнаємося, як виконуються команда повторення для *і оператор циклу* **for**;

з'ясуємо, що таке датчик випадкових чисел і як його використовують;

довідаємося, як створюються програми-тренажери.

====62.1. Повторення задану кількість разів. Лічильник циклу ====================

В алгоритмах із циклами, які ми розглядали раніше, кількість повторень циклу до початку його виконання не була відомою. Цикл завершував свою роботу за результатом перевірки умови циклу. Проте існує багато задач, де повторення певних дій мають відбуватися задану кількість разів. Наприклад, для обчислення таблиці значень функції на проміжку від *a* до *b* із кроком *h* цикл має повторюватися $1+(b-a)/h$ разів.

У циклах із заданою кількістю повторень використовують спеціальну змінну цілого типу — ***лічильник циклу***. Для лічильника циклу встановлюється перелік його значень. Цикл виконується стільки разів, скільки значень міститься в переліку. Кожне наступне проходження циклу відбувається із черговим значенням лічильника. Так, для розрахунку таблиці кубів усіх цілих чисел від 1 до 30 цикл має повторитися 30 разів, лічильник циклу набуватиме значень: 1, 2, …, 30, і для кожного значення буде обчислюватися його куб.

Перелік значень лічильника задається його початковим значенням, кроком змінювання й кінцевим значенням, за межу якого виходити не можна. Наприклад, якщо задано початкове значення лічильника 2, крок 4, кінцеве 12, то лічильник набуватиме значень 2, 6, 10.

Цикл із лічильником можна реалізувати за допомогою команди повторення доки (або оператора циклу **while**):

лічильник := початкове_значення	лічильник :=початкове_значення;
доки лічильник <= кінцеве_значення	**while** лічильник <= кінцеве_значення **do**
пс	**begin**
<команди тіла циклу>	<оператори тіла циклу>
лічильник :=лічильник+крок	лічильник :=лічильник+крок;
кс	**end;**

====62.2. Команда повторення для і оператор циклу **for**==========================

Для утворення циклу з лічильником у НАМ існує спеціальна команда, яка називається ***командою повторення з лічильником***, або ***командою*** для.

Команда для утворюється з використанням службових слів для, від, до, крок:

для лічильник від початкове_значення до кінцеве_значення крок крок

<тіло циклу>

Якщо тіло циклу складається з більш ніж однієї команди, то вони об'єднуються в одну складену команду, тобто розміщуються між службовими словами пс і кс.

Лічильник є обов'язково змінною цілого типу, її початкове й кінцеве значення, а також крок задаються цілими значеннями або виразами, що приводять до цілих значень.

Якщо значення лічильника циклу *збільшуються* з кроком 1, то слова крок 1 можна пропустити.

Аналогом команди для в мові Паскаль є ***оператор циклу з лічильником***, або ***оператор*** **for**, який утворюється із використанням службових слів **for** (для), **to** (до) або **downto** (вниз до), **do** (робити) і знаку присвоєння (замість слова «від»).

Лічильником, як і в НАМ, може бути тільки змінна цілого типу, початкове і кінцеве значення якої задаються цілими значеннями або виразами, що приводять до цілих значень. Крок змінювання значень лічильника може дорівнювати тільки +1 або –1. Отже, лічильник набуває у*сіх підряд* цілих значень від початкового до кінцевого.

Для зростаючих значень лічильника (крок +1) в операторі циклу **for** використовується службове слово **to**:

```
for лічильник := початкове_значення to кінцеве_значення do
<тіло циклу>;
```

Для спадаючих значень лічильника (крок –1) в операторі циклу for використовується службове слово downto:

```
for лічильник := початкове_значення downto кінцеве_значення do
<тіло циклу>;
```

Якщо тіло циклу складається з більш ніж одного оператора, їх об'єднують в один складений оператор, тобто розміщують між службовими словами **begin** і **end**.

Розглянемо приклади застосування оператора for.

Оператор	**Значення, що виводяться на екран**
`for n := 1 to 4 do write(2*n: 3);`	2 4 6 8
`for n := 4 downto 1 do write(2*n: 3);`	8 6 4 2
`m:= 2; for n := m-1 to sqr(m) do write(2*n: 3);`	2 4 6 8

Тіло циклу з лічильником може містити інший цикл із лічильником. Внутрішній цикл має повністю розміщуватися у зовнішньому. Імена змінних-лічильників внутрішнього й зовнішнього циклів повинні бути *різними*.

====62.3. Програма «Таблиця значень функції» ==========

Розробимо програму для виведення на екран таблиці значень кубів цілих чисел від 1 до 20. Позначимо числа через x, а їх куби через y. Цикл має виконуватися 20 разів — для кожного значення x. Отже, змінну x використаємо як лічильник циклу.

Будемо виводити значення x і y в таблицю із заголовком «Таблиця значень кубів чисел», яка містить два стовпці, позначені як x і y. Запишемо програму.

```
program table_of_function;
 uses Crt;
 var x, y: integer;
begin
 clrscr;
 writeln('Таблиця значень кубів чисел');          {Заголовок таблиці}
 writeln ('');
```

```
 writeln('x':7, 'y':14);                    {Виведемо заголовки стовпців таблиці}
 writeln('----------------------------------------');
 for x:=1 to 20 do
  begin
   y := x*sqr(x);                                    {Знаходимо куб числа}
   writeln(x:7, y:14);                    {Виводимо на екран значення x і y}
  end;
 readln;
end.
```

Нашу програму легко модифікувати для будь-якої іншої функції з цілочисельними значеннями аргументу.

Для обчислення таблиці значень функції `y(x)` з кроком `h`, де `h≠1`, для `x` від `x0` до `xk` цикл можна організувати в такий спосіб:

```
n := 1+trunc(xk-x0)/h;
x := x0;
for i:=1 to n do
 begin
  y := …;
  writeln(x, y);
  x := x+h;
 end;
```

====62.4. Датчик випадкових чисел. Функція random і її використання==========

У Паскалі можна використовувати вбудований ***датчик випадкових чисел*** — програму, яка генерує одне за одним числа випадковим чином. Насправді послідовність чисел відповідає певній закономірності, але вона задана в такий спосіб, що майже не відрізняється від випадкової.

Щоб скористатися датчиком випадкових чисел, його потрібно запустити оператором **`randomize`** (від англ. random — випадковий, вибраний навмання). У такому разі далі в програмі можна діставати випадкові числа за допомогою ***функції random(n)***, де `n` є цілим додатним числом. Ця функція повертає ціле число, випадковим чином вибране з діапазону чисел від `0` до `n-1`.

Наприклад, у результаті виконання оператора `y:=random(5)`; змінна `y` може отримати одне із значень: 0, 1, 2, 3, 4. Яке саме — буде вибрано випадковим чином.

Для того щоб змінна `y` набула випадкового значення з діапазону від `n1` до `n2` включно, потрібно скористатися оператором:

```
y := n1+random(n2-n1+1);
```

Так, якщо ми хочемо виводити на екран різнокольорові символи, то аргументу `n` оператора **`textcolor`**`(n)` можна надавати значення, вибрані з діапазону від 1 до 15:

```
n := 1+random(15);
```

Наведена нижче програма демонструє роботу датчика випадкових чисел. Програма підраховує і виводить на екран кількість одиниць (`n1`), двійок (`n2`) і трійок (`n3`), отриманих за допомогою багаторазового використання функції **`random`**.

```
program one_two_three;
 uses Crt;
 var i, x, n1, n2, n3: integer;
```

```
begin
 clrscr;
 randomize;                                  {Запускаємо датчик випадкових чисел}
 n1 := 0; n2 := 0; n3 := 0;
                                             {До початку роботи датчика n1=n2=n3=0}
 for i:=1 to 30000 do                        {Випробуємо 30 000 випадкових чисел}
  begin
    x := 1+random(3);                        {Дістаємо число 1, 2 або 3}
    if x=1 then n1 := n1+1;                  {Якщо це 1, то збільшуємо кількість одиниць}
    if x=2 then n2 := n2+1;                  {Якщо це 2, то збільшуємо кількість двійок}
    if x=3 then n3 := n3+1;                  {Якщо це 3, то збільшуємо кількість трійок}
    gotoxy(1, 5);                            {Виходимо в першу позицію 5-го рядка}
    writeln('Кількість одиниць: ', n1:8);    {Виводимо дані}
    writeln('Кількість двійок: ', n2:8);     {в одне й те саме}
    writeln('Кількість трійок: ', n3:8);     {місце на екрані}
    delay(100);                              {Затримуємо виконання програми на 100 мілісекунд}
  end;
 readln;
end.
```

Ми скористалися оператором **delay** для затримки виконання програми на 100 мілісекунд, щоб можна було спостерігати за зміною значень, що виводяться на екран.

====62.5. Програма «Блукаюча зірочка» ==========

Розробимо програму виведення на екран блукаючої зірочки. Зірочка з'являється в довільному місці екрана, гасне, з'являється в іншому і т. д. Колір зірочки й позиція, де вона виникає, задаються датчиком випадкових чисел. Для того щоб «гасити» зірочку, будемо повторно виводити її в те саме місце екрана кольором фону.

Наведемо текст програми.

```
program roving_star;
 uses Crt;
 var i, x, y: integer;
begin
 textbackground(0);
 clrscr;                                     {Очищаємо екран}
 randomize;                                  {Запускаємо датчик випадкових чисел}
 for i:=1 to 100 do                          {Виводимо 100 зображень зірочки}
  begin
    textcolor(1+random(15));                 {Вибираємо колір зірочки}
    x := 1+random(80);                       {Визначаємо випадковим чином}
    y := 1+random(25);                       {позицію зірочки}
    gotoxy(x, y);                            {Ставимо курсор у задану позицію}
    write('*');                              {Виводимо зірочку}
    delay(1000);                             {Милуємося зірочкою 1 с}
```

```
    textcolor(0);                                   {Задаємо колір фону}
    gotoxy(x, y);                                   {Ставимо курсор у позицію зірочки}
    writeln('*');                                   {Гасимо зірочку}
   end;
 writeln('Добраніч!');
 readln;
 textcolor(7); textbackground(0);                   {Повертаємо початкові кольори}
 clrscr;
end.
```

Оператор **delay**(n) створює паузу у виконанні програми на задане ціле число мілісекунд. Адже комп'ютер так швидко виконує всі дії, що без уведення затримки ми побачимо на екрані лише кінцевий результат.

Зверніть увагу: у нашій програмі лічильник циклу в тілі циклу ніяк не використовується, він просто рахує кількість запусків циклу.

====62.6. Програма-тренажер «Таблиця множення» ==========

Функція **random**(n) дозволяє створювати програми-тренажери, які пропонують користувачеві одне за одним завдання певного типу, перевіряють правильність наданих ним відповідей і виводять на екран повідомлення про результати роботи користувача.

Розробимо програму-тренажер на прикладі таблиці множення. Позначимо через x і y числа, які потрібно перемножити, через z – результат множення, введений користувачем, через pv – кількість правильних відповідей користувача. Кількість завдань позначимо через k. Усі перелічені змінні є змінними цілого типу.

```
program multiplication_table;
 uses Crt;
 const k=20;
 var i, x, y, z, pv: integer;
begin
 clrscr;
 randomize;                                 {Запускаємо датчик випадкових чисел}
 writeln('Чи пам'ятаєте ви таблицю множення? ');
 pv := 0;                {Перед початком роботи надаємо змінній pv нульового значення}
 for i:=1 to k do
  begin
    x := 2+random(8);                               {Визначаємо множники}
    y := 2+random(8);
    write(x, ' x ', y, '=');                        {Виводимо на екран  приклад}
    readln(z);                                      {Зчитуємо відповідь користувача}
    if z =x*y                                       {і перевіряємо її правильність}
     then pv := pv+1;                               {Рахуємо правильні відповіді}
  end;
 write('Загальний рахунок ', pv, ' : ', k-pv);      {Виводимо на екран}
 if pv > k-pv      {результат роботи користувача з програмою-тренажером}
  then writeln(' на вашу користь.');
```

```
  if pv < k-pv
    then writeln(' на мою користь.');
  readln;
end.
```

Аналогічним способом можна розробити тренажер на розв'язання рівнянь певного типу.

ВИСНОВКИ

У багатьох задачах певна група дій має повторюватися заздалегідь відому кількість разів. Такі задачі приводять до циклів, де кількість повторень задається переліком значень деякої змінної. Цю змінну називають лічильником циклу. При кожному проходженні циклу лічильник набуває нового значення згідно із заданим переліком. Для зручності складання таких циклів у НАМ існує команда повторення з лічильником, або команда для, а в мові Паскаль — оператор циклу з лічильником, або оператор **for**. У програмах із циклами (та інших) іноді виникає потреба надавати величинам значення, які випадковим чином вибираються із заданого діапазону. Це здійснюється з використанням датчика випадкових чисел. Датчик запускається оператором **randomize**. Для надання змінній довільного значення застосовується функція **random**(n), яка повертає ціле число з діапазону від 0 до n−1. За допомогою функції **random**(n) можна створювати програми-тренажери, які «генерують» завдання заданого виду і перевіряють правильність їх виконання користувачем.

Контрольні питання та вправи

1. Циклом із лічильником називають такий цикл, який повторюється:

   а) наперед відому кількість разів;
   б) заздалегідь невідому, але скінченну кількість разів;
   в) заздалегідь відому кількість разів згідно з переліком значень деякої цілої змінної.

2. Змінна s має значення 1. Визначте значення змінної s після виконання циклів:

   1) `for n:=2 to 5 do s := s+n;`
   2) `for n:=1 downto -1 do s:= s+n;`
   3) `for n:=1 to 4 do s := s*n;`
   4) `for n:=3 downto 1 do s:= n-2*s;`

3. Визначте, з якого діапазону цілих чисел буде вибрано значення змінної k оператором:

```
k := 2+random(10);
```

   а) від 2 до 12;
   б) від 3 до 12;
   в) від 2 до 11;
   г) від 3 до 11;
   д) від –8 до 12.

4. У тіло циклу:

```
y:=1;
для i від 1 до n
 . . .
```

вставте команду, потрібну для обчислення таких функцій:

   1) $y = 1+x+2x+\ldots+nx$;
   2) $y = (x+1)\cdot(x+2)\cdot\ldots\cdot(x+n)$;
   3) $y = (\ldots(x+1)\cdot x+2)\cdot x+\ldots)\cdot x+n$;
   4) $y = (\ldots(1+x)\cdot 2+x)\cdot 3+\ldots+x)\cdot n$.

Виберіть відповідь із запропонованих:

а) `y := y*(x+i);`

б) `y := x*y+i;`

в) `y := i*x+y;`

г) `y := x+i*y;`

5. Із кубічних блоків побудовано чотирикутну піраміду. Її основою є квадрат з m^2 блоків, а на вершині знаходиться 1 блок. Складіть програму підрахування кількості блоків n, витрачених на побудову піраміди, за відомим значенням m.

6. Складіть програму «Ворожка», яка пропонує користувачеві загадати бажання і для перевірки, збудеться воно чи ні, ввести ціле додатне число n<1000. Програма підраховує, яких чисел більше — парних чи непарних — одержано від датчика випадкових чисел у результаті $2n+1$ звернення до нього, і виводить на екран відповідне повідомлення «збудеться» або «не збудеться». Налаштуйте датчик так, щоб він генерував цілі числа від 10 до 99.

7. Скористайтеся датчиком випадкових чисел так, щоб одержувати з його допомогою тільки –1 або +1. Складіть ігрову програму, яка пропонує користувачеві відгадати, яким буде чергове число: +1 чи –1. Якщо користувач відгадав, йому нараховується очко, інакше — комп'ютеру. Поточний рахунок гри виводиться на екран. Гра ведеться до заданої кількості n спроб. Дослідіть, як змінюється рахунок гри із зростанням n.

8. Число 481 має чудову властивість: якщо число 12 подвоїти, приписати справа нуль, до результату додати те саме число 12 й одержану суму помножити на 481, то в запису шестизначного результату тричі повториться число 12: 121212. Складіть програму перевірки, чи зберігає число 481 таку властивість для інших двозначних чисел, крім 12. Поясніть одержані результати. Зробіть висновок про знайдену властивість числа 481. Спробуйте довести її аналітично.

9. Із зоопарку втік крокодил. Опівночі він опинився на перехресті доріг. Там він навмання вибрав подальший напрям руху — на північ, південь, схід чи захід. За годину він подолав квартал і на перехресті зробив так само. У такий спосіб крокодил блукав до сьомої ранку. Складіть програму, яка моделює рух утікача й виводить на екран пораду, де його слід шукати.

10. Неважко переконатися, що $9\cdot1+2=11$; $9\cdot12+3=111$; $9\cdot123+4=1111$. Складіть програму перевірки, чи має місце така властивість числа 9 і для більших чисел. Доведіть аналітично правильність одержаного результату.

11. Складіть програму-тренажер для перевірки правильності розв'язання рівнянь виду:

1) $a+bx=c$. Вказівка: значення a, b, x визначте за допомогою датчика випадкових чисел, значення c обчисліть як $a+bx$;

2) $x^2+bx+c=0$. Вказівка: значення x_1, x_2 визначте за допомогою датчика випадкових чисел, значення b, c — за допомогою теореми Вієта.

12 Складіть програму створення рухомого об'єкта на екрані. Наприклад, умовної фігурки, яка виконує зарядку, або знімає та надіває капелюх, або переміщується по екрану тощо.

датчик випадкових чисел, команда повторення "для", команда повторення з лічильником, лічильник циклу, оператор циклу з лічильником, оператор *for*, цикл з лічильником, функція випадкових чисел