# § 61. Алгоритми і програми з циклами

***Вивчивши цей параграф, ми:***

*познайомимося із засобами реалізації базової алгоритмічної структури повторення в НАМ і мові програмування Паскаль;*

*дізнаємося, як виконуються команди повторення* доки *і* повторювати – до *та їх аналоги — оператори циклу* **while** *і* **repeat – until**;

*з'ясуємо на прикладах особливості застосування операторів циклу.*

## ====61.1. Засоби реалізації базової алгоритмічної структури повторення з передумовою ==

Базова алгоритмічна структура повторення (цикл) має дві форми — цикл із передумовою і цикл із післяумовою. У навчальній алгоритмічній мові і в мові програмування Паскаль є відповідні засоби реалізації циклів: *команда повторення* й *оператор циклу*, які також мають дві форми — з передумовою і з післяумовою. Розглянемо їх по черзі.

Схема реалізації повторення з передумовою в НАМ і мові програмування Паскаль подана на рис. 61.1.

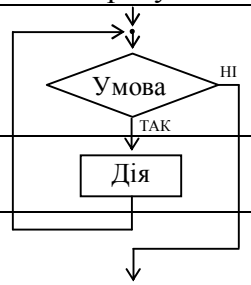
| Блок-схема повторення з передумовою | НАМ<br>Команда доки | Мова Паскаль<br>Оператор *while* |
|---|---|---|
|  | доки <умова> | **while** <умова> **do** |
| | <команда тіла циклу> | <оператор тіла циклу> |
| | | **;** |

Рис. 61.1. Схема реалізації циклу з передумовою в НАМ і мові програмування Паскаль

***Команда повторення з передумовою*** записується за допомогою службового слова доки. Після нього наводиться умова циклу. Далі розміщується тіло циклу. Тіло циклу, як показано на схемі, містить лише *одну* команду. Якщо тіло циклу повинно містити дві або більше команд, їх об'єднують в одну серію, тобто розміщують між службовими словами пс і кс.

Перед кожним виконанням тіла циклу перевіряється умова циклу. Якщо результатом перевірки є «так», виконується тіло циклу; якщо «ні» — здійснюється вихід із циклу, тобто перехід до виконання наступної за циклом команди.

Аналогом команди повторення з передумовою (або ***команди*** доки) є ***оператор циклу з передумовою*** в мові програмування Паскаль. Замість службового слова доки використовується його англійський еквівалент **while**. Крім того, на відміну від команди доки, в ***операторі*** while запис умови завершується службовим словом **do** (робити). Якщо тіло циклу містить декілька операторів, їх об'єднують в один складений оператор, тобто розміщують між службовими словами **begin** і **end**.

Розглянемо приклади покрокового виконання команди доки та оператора **while**. Значення змінних до виконання оператора: $x = 3$; $y = 7$.

| Команда "доки" | Оператор циклу *while* | Виконувана дія | | Виконувана дія | | Виконувана дія | |
|---|---|---|---|---|---|---|---|
| | | № | Результат | № | Результат | № | Результат |
| доки $y >= x$ | *while* $y >= x$ *do* | **1.** | 7≥3 так | **3.** | 4≥3 так | **5.** | 1≥3 ні |
| $y := y - x$ | $y := y - x$; | **2.** | $y := 4$; | **4.** | $y := 1$ | | |
| вивести('$y$=',$y$) | *writeln*('$y$=',$y$); | | | | | **6.** | ***y*=1** |

## ====61.2. Програма «Сума цифр числа» ==========

Продемонструємо застосування оператора циклу з передумовою на прикладі розробки програми обчислення суми цифр цілого числа.

•*Постановка задачі.* Для заданого цілого числа знайти суму його цифр.

•*Розробка інформаційної моделі задачі.* Позначимо задане число через n (вхідне дане), суму його цифр — через s (вихідне дане). Для обчислення s скористаємося таким прийомом: будемо знаходити останню цифру числа — z, додавати її до суми s і «відрізати» від початкового числа. Поступово скорочуючи у такий спосіб число, ми знайдемо всі його цифри, а число зведемо до нуля. Щоб вхідне значення n зберігалося, введемо змінну p, з якою й будемо виконувати описані дії. Початкове значення p дорівнює n.

Остання цифра числа знаходиться як остача від його ділення на 10; «скорочення» числа («відрізання» останньої цифри) здійснюється шляхом його ділення націло на 10.

•*Розробка комп'ютерної моделі задачі.* Усі змінні задачі є цілими. Це дозволяє скористатися для знаходження останньої цифри числа оператором:

```
z := p mod 10;
```

а для «відрізання» від числа p останньої цифри — оператором:

```
p := p div 10;
```

Для поступового винайдення всіх цифр числа і їх суми організуємо цикл із передумовою.

Звернемо увагу на важливу обставину. Якщо змінна n буде оголошена як змінна цілого типу, то її значення не може виходити за межі діапазону [–32 768; 32 767], у той час як наш алгоритм дозволяє опрацьовувати числа будь-якої довжини. Щоб розширити діапазон припустимих значень n, оголосимо її як змінну ***типу*** **longint** (від англ. long integer — довге ціле). Для зберігання значень змінних цього типу виділяються 4 байти (а не 2, як для **integer**), що дозволяє поширити їх діапазон від –2 147 483 648 до +2 147 483 647.

Запишемо розроблену програму.

```
program sum_ of_ numbers;
 uses Crt;
 var p, n: longint; s, z: integer;
begin
 clrscr;
 write('Уведіть ціле число, яке не більше 2-х мільярдів за модулем: ');
 readln(n);
 p := abs(n);                          {Замінюємо від'ємне значення його модулем}
 s := 0;                               {Надаємо змінній початкового нульового значення}
 while p>0 do
  begin
    z := p mod 10;                     {Знаходимо останню цифру числа}
    s := s + z;                        {Збільшуємо значення суми на знайдену цифру}
    p := p div 10;                     {«Скорочуємо» число на останню цифру}
  end;
 writeln('Сума цифр числа дорівнює ', s);          {Виводимо результат}
 readln;
end.
```

## ==== 61.3. Засоби реалізації базової алгоритмічної структури повторення з післяумовою==

Схема реалізації повторення з післяумовою в НАМ і мові програмування Паскаль подана на рис. 61.2.

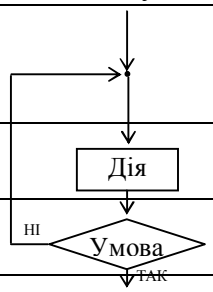
| Блок-схема повторення з післяумовою | НАМ<br>Команда повторювати – до | Мова Паскаль<br>Оператор **repeat-until** |
|---|---|---|
|  | повторювати | **repeat** |
| | <команди тіла циклу> | <оператори тіла циклу> |
| | до <умова> | **until** <умова>**;** |

Рис. 61.2. Схема реалізації циклу з післяумовою в НАМ і мові програмування Паскаль.

***Команда повторення з післяумовою*** записується за допомогою службових слів повторювати і до. Між ними розміщується тіло циклу. Після слова до записується умова циклу. Тіло циклу, як показано на схемі, може містити довільну кількість команд. Об'єднувати їх у серію немає потреби, тому що слова повторювати і до відіграють роль обмежників тіла циклу.

Цикл із післяумовою починається з виконання тіла циклу. Після кожного такого виконання перевіряється умова циклу. Якщо результатом перевірки є «ні», тіло циклу виконується ще раз; якщо «так» — здійснюється вихід із циклу, тобто перехід до виконання наступної за циклом команди.

Аналогом ***команди*** повторювати – до в мові програмування Паскаль виступає ***оператор циклу з післяумовою***, або ***оператор*** **repeat – until.** Слова **repeat** і **until** є англійськими еквівалентами службових слів повторювати і до.

Наведемо приклад покрокового виконання команди повторювати – до і оператора **repeat – until**. Значення змінних до виконання оператора: $x = 3$; $y = 7$.

| Команда НАМ<br>"повторювати – до" | Оператор циклу<br>*repeat-until* | Виконувана дія | | Виконувана дія | |
|---|---|---|---|---|---|
| | | № | Результат | № | Результат |
| повторювати | **repeat** | | 7≥3 так | | |
| $y := y - x$ | `y := y - x;` | **1.** | $y := 4$; | **3.** | $y := 1$ |
| до $y < x$ | **until** `y` $< x$ | **2.** | $4 < 3$ ні | **4.** | $1 < 3$ так |
| вивести('$y$=',$y$) | **writeln**`('y=',y);` | | | **5.** | ***y*=1** |

Приклад, який ми навели, є тим самим, що розглядався в п. 61.1. Ви можете порівняти, як виконуються цикли з перед- і післяумовою. Зверніть увагу на запис умов циклу: вони є протилежними, тому що після **while** (доки) записується умова повторного виконання тіла циклу, а після **until** (до) — умова виходу із циклу.

## ====61.4. Програма «Подвоєння внеску» ==========

Продемонструємо застосування оператора циклу з післяумовою на прикладі розробки алгоритму про подвоєння внеску, що зберігається в банку.

• *Постановка задачі.* Грошова сума, покладена у банк, щороку зростає на $p$ відсотків. Визначити, через скільки років сума на рахунку подвоїться.

• *Аналіз постановки задачі.* За рік сума на рахунку збільшується на *p* відсотків, тобто зростає у (1+*p*/100) разів. За даним *p* потрібно встановити, через скільки років сума на рахунку зрівняється або вперше перевищить подвоєний розмір початкового внеску. Задача має розв'язання для будь-якого значення *p*>0 і додатної суми початкового внеску. Це є обмеження на вхідні дані.

• *Розробка інформаційної моделі задачі.* Введемо позначення: v — розмір початкового внеску, vp — поточна сума на рахунку через t років.

Для t=0 маємо vp=v; далі щороку t зростає на 1, а vp збільшується у (1+p/100) разів. Якщо в результаті чергового збільшення виявиться, що vp≥2v, то значення t і є шуканою величиною. Отже, vp≥2v є *умовою виходу із циклу*.

• *Розробка комп'ютерної моделі задачі.* За смислом задачі змінні p, v і vp відносимо до дійсного типу, t — до цілого типу. Оскільки хоча б рік гроші зберігатимуться в банку, для побудови алгоритму розв'язання задачі зручно застосувати цикл з післяумовою.

Наш цикл може виявитися нескінченним, якщо вхідні значення будуть уведені користувачем неправильно. Щоб виключити можливість «зациклювання» програми, перевіримо, чи задовольняють вхідні дані умові: p>0 і v>0.

Запишемо програму.

```
program double_deposit;
 uses Crt;
 var t: integer; p, v, vp: real;
begin
 clrscr;
 write('Скільки відсотків за рік дає банк? ');          {Запрошуємо}
 readln(p);                                             {користувача ввести}
 write('Яка сума вашого внеску? ');                     {вхідні дані}
 readln(v);
 if (p>0) and (v>0)                                     {Перевіряємо їх правильність}
  then begin
   t := 0;                                              {Надаємо змінним початкових значень}
   vp := v;
   repeat                                               {Обчислюємо щорічний прибуток}
    t := t+1;
    vp := vp*(1+p/100);
   until vp>=2*v;                                       {Перевіряємо, чи не подвоївся внесок}
   writeln('Ваш внесок подвоїться за ', t, ' років');
   end
  else writeln('Ви ввели неправильні дані.');           {Виводимо повідомлення}
 readln
end.
```

Для тестування програми можна скористатися такими даними:

для p=100, v=1000 значення t має дорівнювати 1;

для p=25, v=1000 значення t має дорівнювати 4.

На завершення зазначимо, що оператором тіла циклу може виступати знову ж таки оператор циклу. Утворювана конструкція називається *вкладенням циклів*, або ***вкладеними циклами***.

Виконання вкладених циклів здійснюється у такий спосіб: для кожного проходження зовнішнього циклу здійснюється повне прокручування внутрішнього циклу.

**ВИСНОВКИ**

Для подання базової алгоритмічної структури повторення з передумовою в НАМ існує команда доки, а в мові Паскаль — оператор циклу **while**. Тіло циклу з передумовою може містити один оператор. Щоб розмістити декілька, їх об'єднують в один складений оператор. Для подання базової структури повторення з післяумовою в НАМ існує команда повторювати – до, а в мові Паскаль — оператор **repeat – until**. Між службовими словами повторювати і до, так само як і між **repeat** і **until**, можна розміщувати довільну кількість команд (операторів). Тіло оператора циклу може містити інший оператор циклу. У такому випадку маємо вкладені цикли.

---

**Контрольні питання та вправи**

1. Вставте пропущені слова. Базова алгоритмічна структура повторення з передумовою подається:

1) у НАМ — командою повторення з передумовою:
… <умова> <команда тіла циклу>

2) у мові Паскаль — оператором повторення з передумовою:

… <умова> … <оператор тіла циклу>

2 Вставте пропущені слова. Базова алгоритмічна структура повторення з післяумовою подається:

1) у НАМ командою повторення з післяумовою:
… <команди тіла циклу> … <умова>

2) у мові Паскаль оператором повторення з післяумовою:

… <оператори тіла циклу> … <умова>

3. Знайдіть, яка кількість зірочок буде виведена на екран у результаті виконання наведених фрагментів програми.

| | | | |
|---|---|---|---|
| `i:=0;`<br>`repeat`<br>`  i := i+1;`<br>`  write('*');`<br>`until i=10;` | `i:=0;`<br>`repeat`<br>`  write('*');`<br>`  i := i+1;`<br>`until i<10;` | `i:=0;`<br>`while i<>10 do`<br>`  begin`<br>`  i :=i+1;`<br>`  write('*');`<br>`  end;` | `i:=10;`<br>`while i>0 do`<br>`  begin`<br>`  write('*');`<br>`  i :=i-1;`<br>`  end;` |
| 1) | 2) | 3) | 4) |

4. Вхідні значення змінних `x=-5; n=1`. Які значення буде мати змінна `x` після виконання наведених фрагментів програми?

| | | |
|---|---|---|
| `while x<n do`<br>`  x := x+2*n;` | `repeat`<br>`  x := x+2*n;`<br>`until x>=n;` | `while x<n do`<br>`  x := x+n;`<br>`x := x+n;` |
| 1) | 2) | 3) |

5. Вхідні значення змінних: `x=5; n=30`. Для кожного з наведених циклів визначте, чи завершить цикл свою роботу; якщо так, то знайдіть, які значення будуть мати змінні `x` і `n` після виходу з циклу.

| 1) | 2) | 3) |
|---|---|---|
| `while x<n do`<br>` if n>10`<br>`  then n := n-10`<br>`  else n := n-4;` | `repeat`<br>`  n := n-10;`<br>`  x := x+n;`<br>`until x<0;` | `while x<n do`<br>`  begin`<br>`  n := n mod 8;`<br>`  x := 25 div x`<br>`  end;` |

6. Складіть програму, яка пропонує користувачеві відгадати, в якому з 12 стільців заховано брильянти мадам Пєтухової. Номер стільця з брильянтами задається в програмі як константа. Комп'ютер веде облік кількості спроб, які знадобилися користувачеві для відгадування, і після вдалої спроби виводить на екран відповідний коментар.

7. Дріжджові бактерії за сприятливих умов подвоюються кожні 5 хвилин. Складіть програму обчислення, за який термін покладені у літрову банку бактерії, що займають об'єм 1 $см^3$, заповнять її доверху.

8. Складіть програму обчислення суми й середнього арифметичного цілих чисел, які одне за одним користувач уводить з клавіатури. Якщо користувач уводить число 55 555, то це означає завершення роботи. На екран виводиться сума чисел, до якої число 55 555 не додається, кількість уведених чисел (число 55 555 не враховується) і відповідне середнє арифметичне.

9. За допомогою програми *double_deposit* виконайте такі дослідження:

1) перевірте, чи залежить термін подвоєння внеску від розміру внеску;
2) встановіть, у якому банку вигідніше зберігати гроші — в банку «Базис», який щомісячно нараховує 1,5%, чи в банку «Промінь», який дає річний відсоток 18%.

10. Складіть програму, яка виводить на екран орнамент, утворений повторенням певного елемента зображення. Кількість елементів в орнаменті обмежується шириною екрана. Елемент має цілком розміститися на екрані. Розробіть варіанти програми для виведення орнаментів, елементами яких є:

1) трикутник, складений із зірочок;
2) хрестик, складений із зірочок;
3) стилізована квітка;
4) різнокольорове зображення на ваш вибір.

Виведення на екран елемента орнаменту доцільно оформити як процедуру.

11. Розробіть програму обчислення планетного числа й характерних рис людини за заданою датою її народження. Скористайтеся програмою *sum_of_numbers* (див. п. 61.2).

Відомий старогрецький філософ та вчений Піфагор (580–497 рр. до н.е.) запевняв, що мудрість усього сущого у світі можна зашифрувати цифрами. Він вважав, що в житті людини значну роль відіграє так зване «планетне» число, яке визначається за датою народження людини та містить таємний код її долі.

Планетне число вказує на планету або зірку — покровительку людини, яка впливає на її особистісні якості, долю тощо. Наприклад, планетне число 1 означає зірку Сонце і такі типові риси людини, як лідерство, імпульсивність; число 2 — Місяць (емоційність, художня обдарованість); 3 — Юпітер (оптимістичність, відповідальність); 4 — Уран (упертість, внутрішня незалежність); 5 — Меркурій (ініціативність, жвавість); 6 — Венера (чуттєвість,

любов до життя); 7 — Нептун (філософічність, поступливість); 8 — Сатурн (сприйнятливість, вірність); 9 — Марс (мужність, сильна воля).

Для знаходження «планетного» числа треба виконати ланцюжок таких дій: записати підряд числа, які відповідають даті народження, — день, номер місяця, рік, утворивши таким чином багатозначне число; обчислити суму цифр цього числа; знайдену суму розглядати як чергове число; обчислити суму його цифр і т. д., доки не буде отримане однозначне число, яке і є шуканим «планетним» числом.

12. Складіть програму для комп'ютерного варіанта гри Баше. У грі беруть участь двоє. Ходять по черзі. Із заданої кількості *n* предметів кожний гравець за свій хід може взяти від 1 до *p* предметів. Значення *n* і *p* є параметрами гри, які встановлюються гравцями до початку гри за домовленістю. Переможеним вважається той, кому доводиться взяти останній предмет.

Програма має забезпечити:

а) виведення на екран умови гри;

б) вибір гравцем значень параметрів гри (*n*, *p*) і перевірку їх правильності (*n* і *p* — цілі додатні числа, *n*>*p*);

в) реалізацію необхідного діалогу з користувачем у процесі гри і контроль за дотримання ним правил гри;

г) визначення переможця гри.

Складіть програму для таких випадків:

1) право першого ходу надається користувачеві;

2) право першого ходу надається комп'ютеру;

3) користувач за його бажанням може зробити перший хід:

4) комп'ютер реалізує стратегію безпрограшної гри і може здійнити вибір – зробити перший хід або передати його користувачеві..

вкладені цикли, команда повторення "доки", команда "повторювати-до", оператор циклу з передумовою, оператор циклу з післяумовою, оператор "*repeat-until*", оператор "*while*"