# Тема №6 .

## ПІДПРОГРАМИ КОРИСТУВАЧА

За допомогою підпрограм (функцій та процедур), компонують групи операторів для виконання деякої єдиної дії. Підпрограми можна викликати з будь-яких місць програми багато разів, вони можуть повертати обраховане значення також їм можна передавати інформацію в як параметри, яку вони використовують для обчислення. Таким чином, бажано використання підпрограм, тому що код програми оптимізується. Якщо в різних місцях програми необхідно виконати однаковий набір операторів, завжди використовують функцію або процедуру.

Перед тим, як викликати підпрограму, її необхідно описати, тобто записати послідовність операторів, яку вона буде виконувати.

Оголошення процедури та функції мають такий загальний вигляд:

```
PROCEDURE <ім'я>(<список_параметрів>);
BEGIN
    <оператори>
END;

FUNCTION <ім'я>(<список_параметрів>):тип;
BEGIN
    <оператори>
END;
```

Після заголовку функції/процедури можна записати розділ оголошення констант, змінних і т.ін. (див. загальну структуру програми, розділ текстів і процедур в темі №2).

Всі змінні, константи, типи і т.ін., оголошені в функції чи процедурі називаються локальними, тобто такі, які діють «локально» – тільки в тілі даної функції чи процедури, за межами якої ці змінні вже не діють.

В дужках після імені підпрограми записується через крапку з комою список формальних параметрів із зазначенням їх типів. Для функції в кінці заголовку записується тип значення, яке повертає ця функція. Наприклад,

```
PROCEDURE MyProc(x:integer; y:real);
FUNCTION MyFunc(i,j:integer):char;
```

В тілі такої функції повинен бути рядок, який присвоює функції якесь значення, яке буде повернене в результаті виконання функції. Наприклад: `MyFunc := 'Z';`

Процедура не може повертати значення, чим вона і відрізняється від функції.

Якщо підпрограма передбачає параметри (в описі записано список формальних параметрів), то при виклику такої підпрограми в розділі основного тексту програми необхідно зазначити перелік фактичних параметрів – це можуть бути як значення, так і змінні відповідного типу. Наприклад, для записаних вище підпрограм правильним є такий їх виклик:

```
MyProc(5, 32.74);
symb:=MyFunc(2,7);
```

або

```
MyProc(k, veight);
symb:=MyFunc(n,m);
```

де *symb* – змінна типу *char*, оскільки функція повертає значення цього типу; отже, в результаті виконання цього рядка змінній *symb* буде присвоєний якийсь символ (наприклад – 'Z').

Якщо підпрограма не передбачає використання параметрів, список разом з дужками в заголовку підпрограми опускається, а в разі виклику такої підпрограми вказується лише її ім'я.

### *Завдання 1*

Скласти блок-схему та написати програму виконання арифметичних дій над двома змінними, зчитаними з клавіатури. Відповідні дії організувати у вигляді підпрограми користувача.

## Методичні рекомендації

1. Проаналізувати технічну постановку задачі.

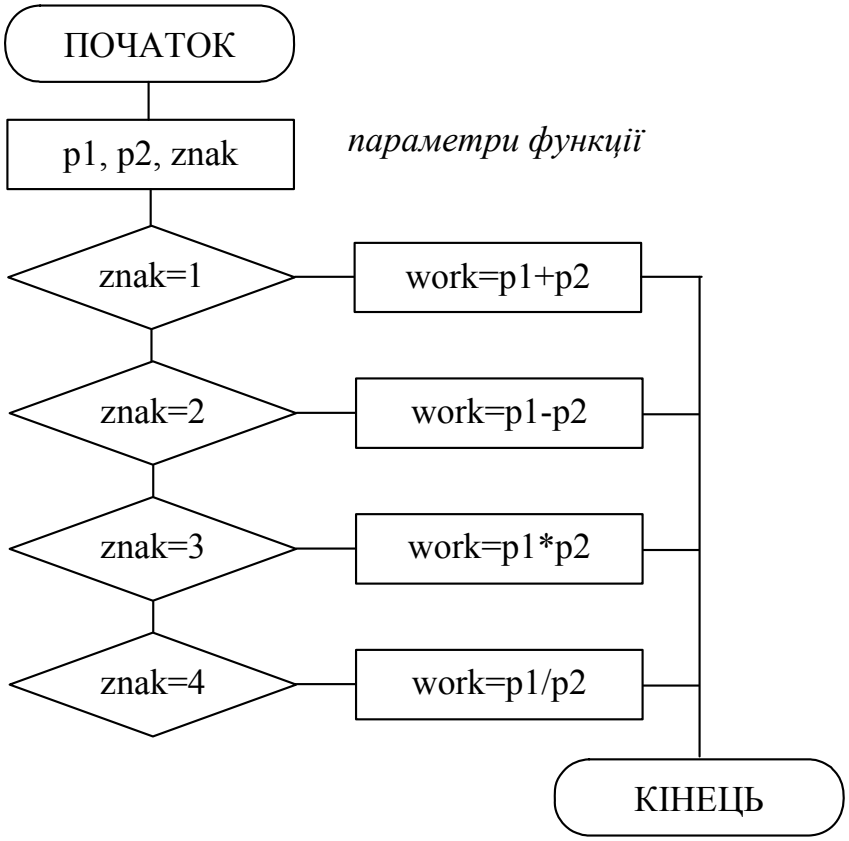
   Відповідно поставленої задачі потрібно зчитати з клавіатури два значення в окремі змінні. Потім необхідно дізнатися яку дію потрібно виконати над даними величинами (плюс, мінус, помножити, поділити). Для того, щоб користувач зробив вибір дії, яку потрібно виконати над введеними величинами, необхідно пояснити як зробити такий вибір. Наприклад, в нашій програмі для таких цілей можна застосувати цілу змінну, яка буде зберігати номер дії: 1 – додавання, 2 – віднімання, 3 – множення 4 – ділення. Щоб користувач зрозумів, що від нього вимагають введення цілого числа – номера дії, потрібно організувати відповідне „меню", тобто пояснення щодо можливості вибору і лише після цього запросити введення номеру дії.

   Після того, як величини і номер дії над ними введено, треба викликати підпрограму, і як параметри передати ці два значення та номер дії, яку необхідно виконати. Результат потрібно повернути назад в основний розділ програми та вивести його на екран. Отже, для того, щоб організувати повернення даних, потрібно використати функцію.

2. Скласти блок-схему функції виконання арифметичних дій.

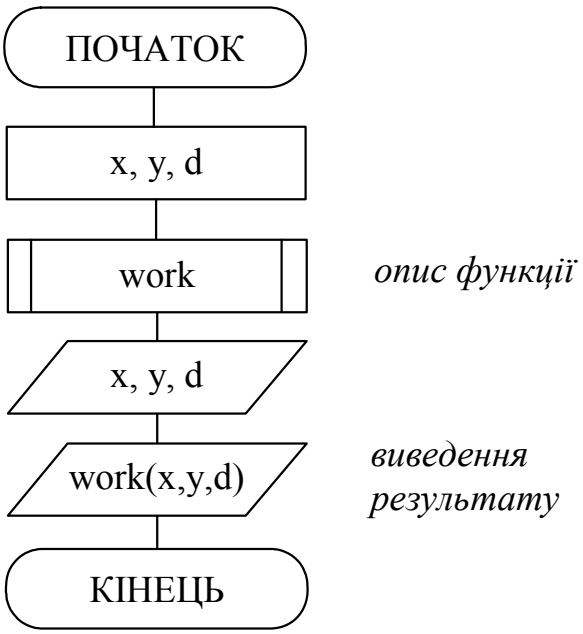
   Як параметри запишемо перших два дійсних числа, з якими потрібно виконати дію, третій, ціле значення – сама дія. Оскільки відомо, що третій параметр може приймати певний набір значень, а саме номер відповідної дії: 1, 2, 3, 4, тоді доцільно використати оператор CASE для аналізу цього значення. В кожному можливому варіанті будемо присвоювати значення функції (повертати) результат обраної арифметичної дії над першими двома параметрами функції:

*функція work(p1,p2,znak) – виконання арифметичної дії*




3. Скласти блок-схему для програми з використанням функції.

   Програма повинна зчитати два числа та знак дії, яку необхідно виконати. Потім викликати функцію `work`. Результат цієї функції вивести на екран:



4. Записати код програми для функції факторіала.

Відповідно до складених блок-схем програми та функції, код програми буде мати такий вигляд:

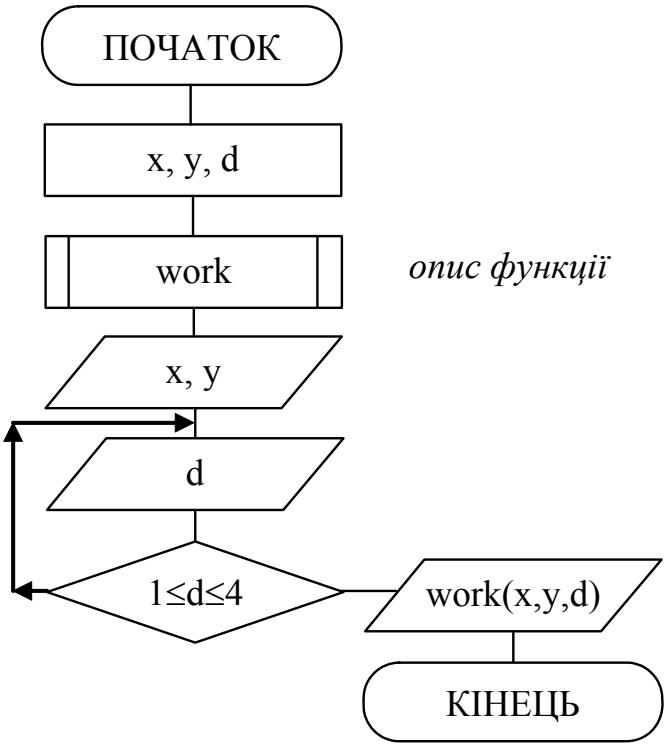
```
{*************************************************
 * SubPrg – програма виконання простих арифметичних *
 *           дій над двома числами                  *
 * Copyright (c) Шищук В.В. гр.ІС-04-1, 22/10/2004  *
 *************************************************}
PROGRAM SubPrg;
TYPE
    arif = 1..4;
VAR
    x,y: real;          {члени ряду}
    d: arif;            {номер дії}

FUNCTION work(p1,p2:Real;znak:arif):Real;
BEGIN
    case znak of
        1: work := p1 + p2;
        2: work := p1 - p2;
        3: work := p1 * p2;
        4: work := p1 / p2;
    end;
END;
BEGIN
    writeln('ПРОСТІ АРИФМЕТИЧНІ ДІЇ');
    write('Введіть перше число: '); readln(x);
    write('Введіть друге число: '); readln(y);
    writeln('--------------------');
    writeln('1. Додавання');
    writeln('2. Віднімання');
    writeln('3. Множення');
    writeln('4. Ділення');
    writeln('--------------------');
    writeln('Виберіть номер дії: '); readln(d);
    writeln('Результат: ', work(x,y,d):8:4);
    write('Для завершення натисніть <Enter>:'); readln
END.
```

В програмі використали власний тип даних *arif* (діапазон), оскільки не має сенсу оголошувати тип *integer* чи *byte*, так як відомо, що змінна *d* мо-

же приймати лише певний набір цілих значень (від 1 до 4).

9. Запустити програму на виконання.
10. Змінити програму таким чином, щоб введення номеру дії відбувалось доти, поки не буде введено коректне число, тобто число від 1 до 4, а ділення виконувалося лише в тому випадку, коли друге число не дорівнює 0.

    В розділі основного блоку програми потрібно додати перевірку на введений номер дії, і знов повторити введення, якщо число не в межах [1;4]. Таку перевірку найкраще реалізувати за допомогою `REPEAT`, адже попередньо невідомо скільки разів користувач буде вводити неправильне значення, за такої умови хоча б один раз, але введення повинно відбутися (з першого разу правильно). Потім потрібно перевірити, чи x не дорівнює 0 і одночасно `d=4` (ділення). Якщо так – вивести повідомлення, що на 0 ділити не можна, якщо ні – викликати функцію. Отже, блок схема програми прийме такий вигляд:



Враховуючи всі вищенаведені зміни в алгоритмі програми, перепишемо вихідний код:

```
{************************************************
 * SubPrg – програма виконання простих арифметичних *
 *           дій над двома числами                *
 * Copyright (c) Шищук В.В. гр.ІС-04-1, 22/10/2004  *
 ************************************************}
PROGRAM SubPrg;
TYPE
    arif = 1..4;
VAR
    x,y: real;          {члени ряду}
    d: arif;            {номер дії}

FUNCTION work(p1,p2:Real;znak:Integer):Real;
BEGIN
    case znak of
        1: work := p1 + p2;
        2: work := p1 - p2;
        3: work := p1 * p2;
        4: work := p1 / p2;
    end;
END;
BEGIN
    writeln('ПРОСТІ АРИФМЕТИЧНІ ДІЇ');
    write('Введіть перше число: '); readln(x);
    write('Введіть друге число: '); readln(y);
    writeln('--------------------');
    writeln('1. Додавання');
    writeln('2. Віднімання');
    writeln('3. Множення');
    writeln('4. Ділення');
    writeln('--------------------');
    repeat
       writeln('Виберіть номер дії: ');
       readln(d);
    until (d >= 1) and (d <= 4);
    if (y = 0) and (d = 4) then
        writeln('Помилка! На нуль ділити не можна!')
    else
        writeln('Результат: ', work(x,y,d):8:4);
    write('Для завершення натисніть <Enter>:'); readln
END.
```

11. Відкомпілювати програму та запустити її на виконання.
12. Протестувати програму за різними значеннями чисел та дій, які над ними виконуються.
13. Зберегти написану програму в персональну папку.

### *Завдання 2*

Змінити задачу попередньої теми (тема №5, завдання 4) таким чином, щоб визначення факторіала та піднесення до степеня відбувалося в окремих підпрограмах користувача.

### *Методичні рекомендації*

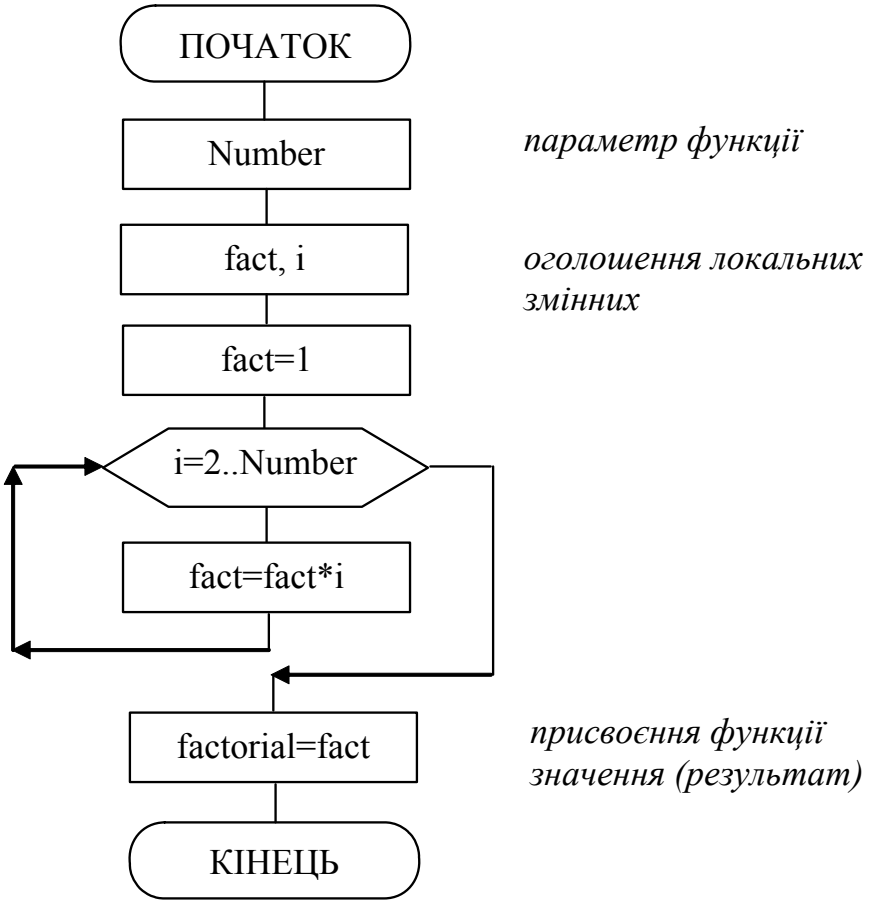
1. Проаналізувати технічну постановку задачі.

   Алгоритм розв'язку завдання 4 попередньої теми не оптимальний, оскільки в тесті програми повторюється частина коду з визначенням факторіала, що не є доцільним. Якщо в коді програми є фрагмент, який повторюється декілька разів, можливо з деякими незначними відмінностями (як в нашому випадку факторіал числа $n$ та числа $2n-1$), тоді потрібно використовувати підпрограми користувача. Для поставленої задачі можна створити функцію, яка буде обраховувати факторіал будь-якого числа, а саме число передавати в якості параметра в дану функцію. Крім того у виразі загального члена ряду є піднесення до степеня, і, хоча раніше записане перетворення є правильним, такий підхід не можна застосувати до від'ємної основи, оскільки використовується логарифм натуральний $\ln(x)$. Для цього моменту також доцільно створити окрему підпрограму з піднесенням будь-якого числа до будь-якої цілої степені. Отже, зміна програми полягає в записі двох підпрограм користувача (для знаходження факторіала та піднесення до степеня) та виклику їх в потрібних місцях з відповідними параметрами.

2. Скласти блок-схему для функції визначення факторіала.

   Для визначення факторіала використовуємо функцію, тому що після обчислення потрібно повернути визначене значення в розділ основного тексту програми. Як параметр треба передати число, факторіал якого необхідно знайти. Нехай цей параметр буде `Number`, який записується першим в послідовності блоків схеми функції:

*функція factorial(Number) – знаходження факторіалу числа*




3. Записати код програми для функції факторіала.

   При записі функції, необхідно пам'ятати, що вона повинна записуватися перед блоком основного коду програми (перед `begin`). Отже, залишаючи оголошення та коментарі програми, після блоку

оголошення змінних *VAR* записуємо код функції, притримуючись складеної блок-схеми:

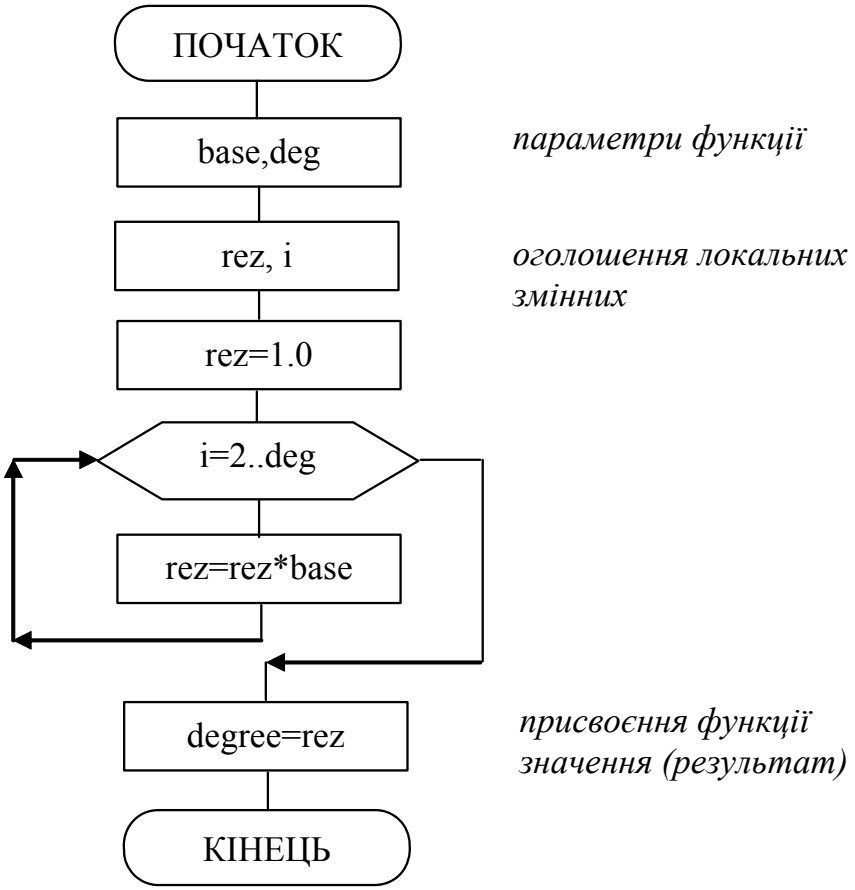
```
{*************************************************
 * RowElem – програма знаходження елементу нескінч. *
 *           ряду в заданій формулі заг. члена та  *
 *           точності різниці між сусідніми елемент.*
 *           з використанням підпрограм            *
 * Copyright (c) Шищук В.В. гр.ІС-04-1, 22/10/04    *
 *************************************************}
PROGRAM RowElem;
CONST
    line = '---------------------------';
VAR
    a,a_: real;         {члени ряду}
    i,n,znak,fact: integer;
FUNCTION factorial(Number:Integer):Integer;
VAR
    fact,i : integer;   //локальні змінні
BEGIN
    fact:=1;
    for i:=2 to Number do
        fact:=fact*i;
    factorial:=fact;    //присвоєння результату
END;

BEGIN
    ... ... ...
END.
```

В кінці рядка оголошення функції через двокрапку записаний тип даних, який буде повернутий на місце, звідки цю функцію викликали. А саме значення, яке буде повертатися, записане в кінці у вигляді присвоєння імені функції значення змінної *fact*. В дужках записаний параметр, який визначає факторіал якого саме числа буде обраховуватися. Для нього також обов'язково зазначати тип даних через двокрапку після самого імені параметра.

4. Скласти блок-схему для функції піднесення до степеня.

Тут також потрібно скористатися функцією для організації повернення знайденого значення. Щоб піднести будь-яке число до цілого степеня можна скористатися циклом, в якому будемо множити число саме на себе необхідну кількість разів, для чого потрібно знати основу, тобто число, яке потрібно піднести до степеня та саму степінь. Отже, цій функції потрібно передати два параметри: `base` – основа, `deg` – степінь. Блок-схема цієї функції буде мати вигляд:

*функція degree(base,deg) – піднесення числа до степеня*




5. Записати код програми функції `degree`.

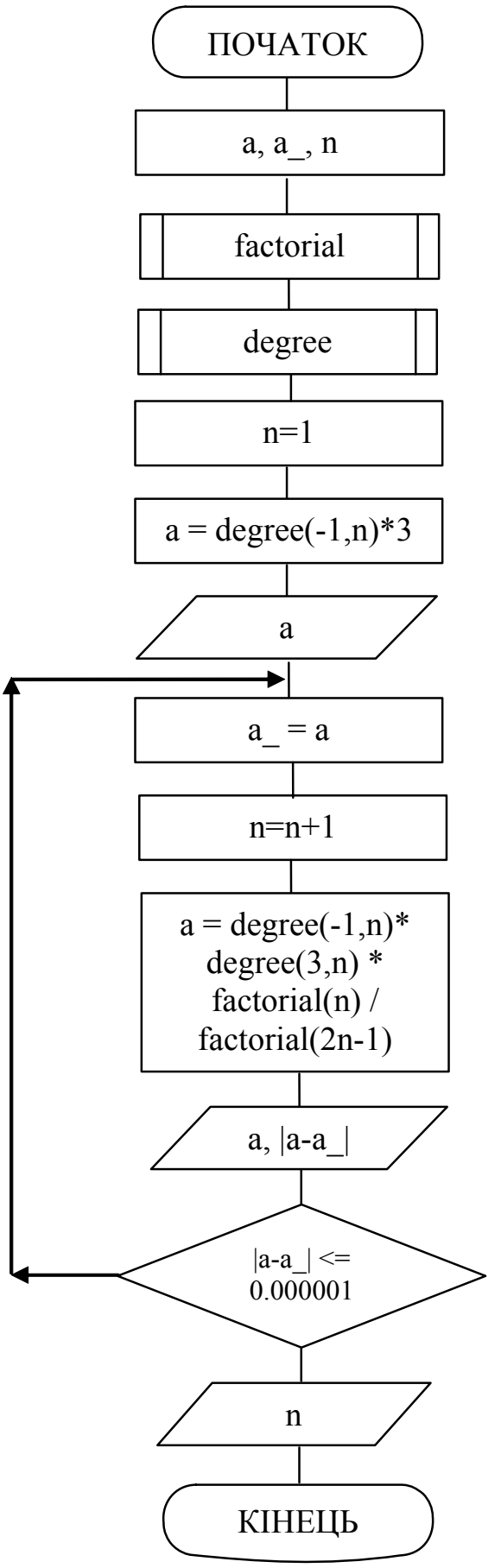
   Так само, як і для попередньої, дану функцію потрібно записати перед блоком основного коду програми, причому взаємне розташування функцій між собою не має значення:

```
{*************************************************
 * RowElem - програма знаходження елементу нескінч. *
 *           ряду в заданій формулі заг. члена та   *
 *           точності різниці між сусідніми елемент.*
 *           з використанням підпрограм             *
 * Copyright (c) Шищук В.В. гр.ІС-04-1, 22/10/04    *
 *************************************************}
PROGRAM RowElem;
CONST
    line = '---------------------------';
VAR
    a,a_: real;         {члени ряду}
    i,n,znak,fact: integer;
FUNCTION factorial(Number:Integer):Integer;
VAR
    fact,i : integer;   //локальні змінні
BEGIN
    ... ... ...
END;
FUNCTION degree(base:Real; deg:Integer):Real;
VAR
    i : integer;        //локальні змінні
    rez : real;
BEGIN
    rez:=1.0;
    for i:=2 to deg do
        rez:=rez*base;
    degree:=deg;        //присвоєння результату
END;
BEGIN
    ... ... ...
END.
```

Для даної функції записано два параметри, причому параметр *base* – дійсного типу, отже піднесення до степеня можна здійснити і для дійсних значень. Степінь – ціле число. Результат піднесення до степеня також буде дійсного типу, оскільки основа – дійсне число. Тип значення, яке повертається записане в кінці після двокрапки.

6. Скласти блок-схему для основного розділу програми з використанням створених функцій користувача.

   Блок-схема із функціями набуде вигляду:




*опис функції*

*опис функції*

*обчислення першого члена ряду*

*обчислення наступного члена ряду*

*виведення члена ряду та модуля різниці його з попереднім членом*

*умова циклу REPEAT*

*виведення номера елемента*

Як видно з блок-схеми, алгоритм став набагато простішим, крім того, вивільнилися дві глобальні змінні: *znak* та *fact*. Їх функції взяли на себе підпрограми та локальні змінні, які в них оголошені.

14. Написати програму реалізації задачі.

Код програми буде містити опис функцій відразу після оголошення глобальних змінних:

```
{*************************************************
 * RowElem – програма знаходження елементу нескінч. *
 *           ряду по заданій формулі заг. члена та  *
 *           точності різниці між сусідніми елемент.*
 *           з використанням підпрограм             *
 * Copyright (c) Шищук В.В. гр.ІС-04-1, 22/10/04    *
 *************************************************}
PROGRAM RowElem;
CONST
    line = '---------------------------';
VAR
    a,a_: real;          {члени ряду}
    i,n,znak,fact: integer;

FUNCTION factorial(Number:Integer):Integer;
BEGIN
    fact:=1;
    for i:=2 to Number do
        fact:=fact*i;
    factorial:=fact;    //присвоєння результату
END;

FUNCTION degree(base:Real; deg:Integer):Real;
VAR
    i : integer;        //локальні змінні
    rez : real;
BEGIN
    rez:=1.0;
    for i:=2 to deg do
        rez:=rez*base;
    degree:=deg;        //присвоєння результату
END;
```

```
BEGIN
    writeln('ЕЛЕМЕНТ НЕСКІНЧЕННОГО РЯДУ');
    n:=1;
    a:=degree(-1,n)*3;  {перший елемент ряду}
    wtieln(line)
    writeln('N = ',n, ' елемент = ',a:12:10);
    repeat
       a_:=a;           {попередній елемент}
       n:=n+1;
       a:=degree(-1,n)*degree(3,n)*factorial(n) /
          factorial(2*n-1);      {значення члена ряду}
       wtieln(line)
       write('N = ',n, ' елемент = ',a:12:10);
       writeln('Різниця = ',abs(a-a_):12:10);
    until abs(a-a_)<=0.000001;
    writeln;
    writeln('Шуканий елемент - ',n, '-й');
    write('Для завершення натисніть <Enter>:');
    readln
END.
```

15. Виконати компіляцію та запустити програму на виконання.
16. Зберегти написану програму в персональну папку.
17. Оформити звіт про виконану роботу (дві програми).

### *Індивідуальні завдання:*

Для даної теми брати індивідуальні завдання попередньої теми (тема №5). Виконання поставленої задачі обов'язково здійснювати за допомогою функцій та процедур користувача.

### *Контрольні запитання:*

1) Що називають підпрограмами і чому? Наведіть практичні приклади.
2) Що таке тіло підпрограми? Чим відрізняється тіло функції від процедури?
3) Які види підпрограм існують і яка між ними різниця в оголошеннях та виклику?

4) Що таке локальні змінні, константи і типи та чим вони відрізняються від глобальних? Наведіть приклади.
5) Що таке фактичні та формальні параметри?
6) Який оператор є необхідним для організації повернення значення функції?
7) В якій частині програми записується оголошення функцій та процедур?
8) Чим відрізняється оголошення підпрограм від їх виклику?
9) Запишіть приклад заголовку функції та процедури з параметрами.
10) Чи можна викликати з тіла підпрограми іншу підпрограму? Якщо так, то в яких випадках?