Цитується за виданням:
Програмування. Практикум / Укл.: Семенюк А.Д., Сопронюк Ф.О. – Чернівці: Рута, 2001.– 143 с.

Лабораторна робота № 13

Рекурсивні процедури і функції

Мета роботи:
1. Засвоєння принципів організації рекурсивних обчислювальних процесів.
2. Отримання практичних навиків розробки і використання рекурсивних процедур і функцій.

Завдання:

Задано натуральне n. Розробити програму для обчислення заданих сум. При обчисленні сум використати рекурсивні процедури або функції.

1. $\sum_{k=1}^{n} \frac{a_k b_k}{k!}$, $a_1 = 1,\ \ a_k = 0.5\left(\sqrt{b_{k-1}} + 0.5\sqrt{a_{k-1}}\right)$, $b_1 = 1,\ \ b_k = 2a_{k-1}^2 + b_{k-1}$.

2. $\sum_{i=1}^{n} \frac{x_i}{1+|y_i|}$, $x_1 = 1,\ \ x_i = 0.3x_{i-1}$, $y_1 = 1,\ \ y_i = x_{i-1}^2 + y_{i-1}$.

3. $\sum_{k=1}^{n} \frac{2^k}{(1+a_k+b_k)}$, $a_1 = 1,\ \ a_k = 3b_{k-1} + 2a_{k-1}$, $b_1 = 1,\ \ b_k = a_{k-1}^2 + b_{k-1}$.

4. $\sum_{k=1}^{n} \frac{a_k b_k}{(k+1)!}$, $a_1 = 1,\ \ a_k = 0.3b_{k-1} + 0.2a_{k-1}$, $b_1 = 1,\ \ b_k = a_{k-1}^2 + b_{k-1}^2$.

5. $\sum_{i=1}^{n} \frac{x_i}{k!!}$, $x_1 = 1,\ x_2 = 2,\ x_3 = 3$, $x_i = x_{i-1} + x_{i-2} + x_{i-3}$.

6. $\sum_{k=1}^{n} \frac{a_k}{(k+1)!}$, $a_k = \left(1 - \frac{1}{2!}\right)\left(1 + \frac{1}{3!}\right)\cdots\left(1 + \frac{(-1)^k}{(k+1)!}\right)$.

7. $\sum_{k=1}^{n} \frac{a_k}{3^k}$, $a_k = \frac{1}{2}\sqrt{\frac{1}{1} + \frac{1}{2}\sqrt{\frac{1}{2} + \frac{1}{2}\sqrt{\frac{1}{3} + \cdots + \frac{1}{2}\sqrt{\frac{1}{k}}}}}$.

8. $\sum_{k=1}^{n} \frac{(-1)^{k+1}}{(2k+1)!} a_k$, $a_0 = 1$, $a_k = 5 + \frac{1}{3^k}\sin^2(a_{k-1} - 1)$.

9. $\sum_{k=1}^{n} \frac{(-1)^{k+1}}{k!!} a_k$, $a_0 = 2$, $a_k = \frac{\sin^2(a_{k-1} - 1)}{k^2}$.

10. $\sum_{k=m}^{n} \frac{(-1)^k}{k!} \left(\frac{a_k}{2}\right)^k$, $a_0 = 1$, $a_k = \sqrt{|4a_{k-1} + 2|}$.

Задано натуральні n і m. Розробити програму для обчислення значень заданих виразів. При обчисленні виразів використати рекурсивні процедури або функції.

11. $\frac{3^m}{\sqrt{1! + \sqrt{2! + \cdots + \sqrt{n!}}}}$,

12. $\frac{\sqrt{n + \sqrt{(n-1) + \cdots + \sqrt{1}}}}{m!!}$,

13. $\frac{2^m}{\frac{1}{3}\sqrt{\frac{1}{1!} + \frac{1}{3}\sqrt{\frac{1}{2!} + \frac{1}{3}\sqrt{\frac{1}{3!} + \cdots + \frac{1}{3}\sqrt{\frac{1}{n!}}}}}}$

14. $\dfrac{m!}{1+\dfrac{1}{3+\dfrac{1}{\cdots\cdots\cdots}}}$, $\dfrac{1}{2n-3+\dfrac{1}{2n-1}}$ **15.** $\dfrac{m!}{m^2+\dfrac{2}{m^2+\dfrac{4}{\cdots\cdots\cdots}}}$, $\dfrac{2n-2}{m^2+\dfrac{2n}{m^2}}$

16. $\dfrac{u_m}{\sqrt{n!+\sqrt{(n-1)!+\cdots+\sqrt{1!}}}}$, $u_0=0,\ u_1=1,$ $u_m=u_{m-1}+u_{m-2}$, **17**, $\dfrac{m!}{\sqrt{2^n+\sqrt{2^{n-1}}+\cdots+\sqrt{2^1}}}$

18. $\sum_{i=1}^{n} H(n,i)$, $H(n,i)=\begin{cases}1, & \text{при } n=0,\\ i, & \text{при } n=1,\\ 2\big(iH(n-1,i)-nH(n-2,i)\big), & \text{при } n>1.\end{cases}$

19. $\sum_{i=1}^{n} C_n^i$, $C_n^i=\begin{cases}0, & \text{при } i<n,\\ 1, & \text{при } i=0 \text{ або } n=i,\\ C_{n-1}^{i}+C_{n-1}^{i-1}, & \text{при } n>i.\end{cases}$

20. $\sum_{i=1}^{n} A(n,i)$, $A(n,i)=\begin{cases}i+1, & \text{при } n=0,\\ A(n-1,1), & \text{при } i=0,\ n>0,\\ A(n-1,A(n,i-1)), & \text{при } n>0,\ i>0.\end{cases}$

Задано масиви чисел $A(n), B(n), (n\le 100)$. Розробити програму для обчислення заданої суми, де a_i, b_i – відповідні елементи масивів A і B. При обчисленні сум використати рекурсивні процедури або функції.

21. $\sum_{i=1}^{n} \dfrac{a_1}{b_1+\dfrac{a_2}{b_2+\dfrac{a_3}{\cdots\cdots\cdots}}}$. $\dfrac{a_{i-1}}{b_{i-1}+\dfrac{a_i}{b_i}}$ **22.** $\sum_{i-1}^{n} \dfrac{a_i^2}{\sqrt{b_i}+\dfrac{a_{i-1}^2}{\sqrt{b_{i-1}}+\dfrac{a_{i-2}^2}{\cdots\cdots\cdots}}}$. $\dfrac{a_2^2}{\sqrt{b_2}+\dfrac{a_1^2}{\sqrt{b_1}}}$

23 $\sum_{i=1}^{n} \dfrac{a_i}{\sqrt{b_1+\sqrt{b_2+\cdots+\sqrt{b_i}}}}$ **24.** $\sum_{i=1}^{n} \dfrac{b_i}{\sqrt{\dfrac{1}{a_i}+\sqrt{\dfrac{1}{a_{i-1}}+\cdots+\sqrt{\dfrac{1}{a_1}}}}}$

25. $\sum_{i=1}^{n} \dfrac{|a_i-b_i|}{u_i}$, $u_0=0,\ u_1=1,\quad u_n=u_{n-1}+u_{n-2}.$

Задано дійсні x і масив $Y(n)$, $n\le 200$. Розробити програму обчислення значення заданих сум. При обчисленні сум використати рекурсивні процедури або функції.

26. $\sum_{k=0}^{n}(-y_k)^k T_k(x)$, $T_0(x)=1,\ T_1(x)=x,\quad k=2,3,\ldots,$ $T_k(x)=2xT_{k-1}(x)-T_{k-2}(x),$

27. $\sum_{k=0}^{n}(-y_k)^k H_k(x)$, $H_0(x)=1,\ H_1(x)=x,\quad k=2,3,\ldots,$ $H_k(x)=xH_{k-1}(x)-(k-1)H_{k-2}(x),$

28. $\sum_{k=0}^{n}(-y_k)^k G_k(x)$, $G_0(x)=1,\ G_1(x)=x-1,\qquad k=2,3,...,$
$G_k(x)=(x-2k+1)\,G_{k-1}(x)-(k-1)^2\,G_{k-2}(x).$

29. $\sum_{k=0}^{n}(-y_k)^k L_k(x)$, $L_0(x)=1,\ L_1(x)=x,\qquad k=2,3,...,$
$L_k(x)=x\,L_{k-1}(x)-\frac{(k-1)^2}{(2k-2)(2k-1)}L_{k-2}(x),$

30. $\sum_{k=0}^{n}(-y_k)^k R_k(x)$, $R_0(x)=1,\ R_1(x)=x, k=2,3,...,$
$R_k(x)=\frac{xR_{k-1}(x)+x^2R_{k-2}(x)}{(2k)!}.$

Теоретичні відомості
Рекурсивні процедури і функції

Рекурсія – це такий спосіб організації обчислювального процесу, при якому підпрограма в ході виконання звертається сама до себе.

Програми, в яких використовуються рекурсивні процедури і функції, відрізняються простотою і наглядністю, але вони вимагають більше пам'яті і виконуються, як правило, повільніше.

Виклик рекурсивної процедури або функції повинен здійснюватися за умовою, яка на деякому рівні рекурсії стає хибною і процес завершується, інакше процес зациклюється, що приводить до переповнення стеку.

Рекурсивний виклик може бути прямим і непрямим. При непрямому виклику підпрограма звертається до себе опосередковано шляхом виклику другої підпрограми, в якій міститься звернення до першої. У цьому випадку використовується директива FORWARD.

Розглянемо приклад прямого рекурсивного виклику.

Приклад. Обчислити значення виразів:

$s1=\sqrt{n+\sqrt{(n-1)+\cdots+\sqrt{1}}}$, $s2=\sqrt{1+\sqrt{2+\cdots+\sqrt{n}}}$.

Для розв'язку задачі командою File!New Application створимо новий проект. Встановимо формі заголовок Caption = Рекурсивне обчислення виразів та присвоїмо їй програмне ім'я Name = FROV. Командою File!Save All запишемо програмний модуль під іменем ULABR10_1.pas, а проект – LAB10_1.dpr.

Розробимо форму для введення початкових даних і виведення результату Для цього розмістимо на формі один компонент Edit для введення початкових даних і два для виведення результатів. Присвоїмо цим компонентам програмні імена Edit1, Edit2, Edit3, встановлені за замовчуванням (властивість Name), і очистимо їм значення властивості Text.

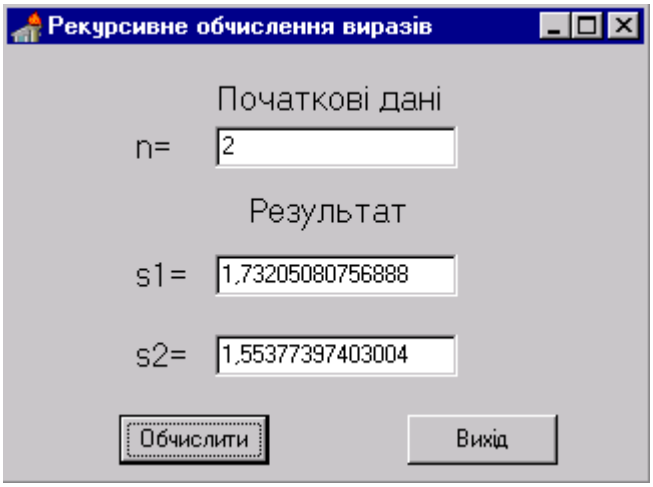

Рис 10.1. Форма Рекурсивне обчислення виразів.

Пояснення до цих компонентів зробимо за допомогою компонента Label (властивість Caption).

Крім цього, розмістимо на формі дві керуючих кнопки (компонент Button) з написами Обчислити та Вихід (властивість Caption) і програмними іменами Button1, Button2 (властивість Name за замовчуванням) (Рис !0.1).

Обробники кнопок Обчислити і Вихід містяться у програмному модулі ULABR10_1.

```
{ Обробник кнопки Обчислити }
procedure TFOVR.Button1Click(Sender: TObject);
VAR     n, i: integer;
        s1,s2: double;
{Рекурсивні функції обчислення коренів}
Function Kor1(n: integer ): double;
begin
  if n=1 then result:=1
  else result:=sqrt(n+Kor1(n-1));
end;
Function Kor2(i, n: integer ): double;
begin
  if i=n then result:=sqrt(n)
  else result:=sqrt(i+Kor2(i+1,n));
end;
begin
{Введення початкових даних}
 n:=StrToInt(Edit1.Text);
{Обчислення коренів}
  s1:=Kor1(n);
  s2:=Kor2(1,n);
{Виведення результатів}
   Edit2.Text:=FloatToStr(s1);
   Edit3.Text:=FloatToStr(s2);
end;
{ Обробник кнопки Вихід }
procedure TFOVR.Button2Click(Sender: TObject);
begin
Close;
end;
```

Тепер командою Run!Run проект можна запустити на виконання. По завершенню компіляції потрібно ввести початкові дані і натиснути кнопку Обчислити. Результати обчислення приведені на Рис 10.1. Для завершення роботи програми потрібно натиснути кнопку Вихід.

Розглянемо приклад непрямого рекурсивного виклику.

Приклад. Для заданого n обчислити значення суми

$$S=\sum_{k=1}^{n}\frac{a_k+b_k}{k!},\quad \text{де}\quad \begin{array}{l} a_1=1,\ \ a_k=\sqrt{b_{k-1}}+\sqrt{a_{k-1}},\\ b_1=1,\ \ b_k=a_{k-1}^2+b_{k-1}^2. \end{array}$$

Для розв'язку задачі командою File!New Application створимо новий проект. Встановимо формі заголовок Caption = Рекурсивні підпрограми та присвоїмо їй програмне ім'я Name = FR. Командою File!Save All запишемо програмний модуль під іменем ULAB10_2.pas, а проект – LAB10_2.dpr.

Аналогічно як у попередньому прикладі розробимо форму для введення початкових даних і виведення результату (Рис. 10.2).

Обробники кнопок Обчислити і Вихід містяться у програмному модулі ULABR10_2 і мають вигляд:

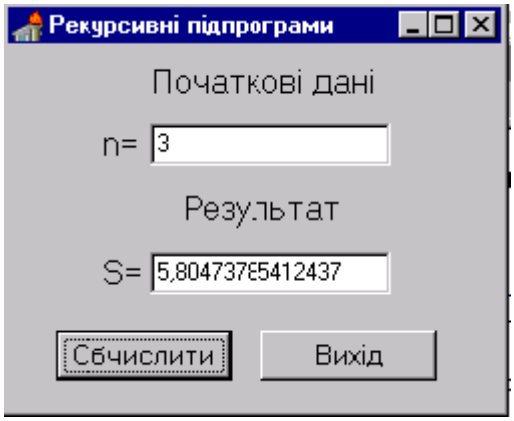

Рис. 10.2. Форма Рекурсивні підпрограми

```
{ Обробник кнопки Обчислити }
procedure TFR.Button1Click(Sender: TObject);
VAR     n: integer;
        s: double;
{Попередній опис функції Ak}
Function Ak(k: integer ): double; FORWARD;
{Функція обчислення  Bk}
Function Bk(k: integer ): double;
 begin
    if k=1 then result:=1
      else result:=sqr(Ak(k-1))+sqr(Bk(k-1));
 end;
{Функція обчислення Ak}
Function Ak(k: integer ): double;
 begin
    if k=1 then result:=1
    else result:=sqrt(Ak(k-1))+sqrt(Bk(k-1));
 end;
{Функція обчислення факторіалу}
Function Fact(k: integer ): longint;
 begin
   if (k=0) or (k=1) then result:=1
        else result:=Fact(k-1)*k;
 end;
{Функція обчислення суми}
Function Sum(k: integer ): double;
 begin
   if k=1 then result:=2
     else result:=Sum(k-1)+(Ak(k)+Bk(k))/Fact(k);
 end;
begin
{Введення початкових даних}
   n:=StrToInt(Edit1.Text);
{Обчислення суми}
   s:=Sum(n);
{Виведення результату}
   Edit2.Text:=FloatToStr(s);
end;
{ Обробник кнопки Вихід }
procedure TFR.Button2Click(Sender: TObject);
begin
  Close;
end;
```

Результат роботи програм наведений на Рис. 10.2.