МІНІСТЕРСТВО ОСВІТИ І НАУКИ,
МОЛОДІ ТА СПОРТУ УКРАЇНИ

Національний технічний університет
«Харківський політехнічний інститут»

МЕТОДИЧНІ ВКАЗІВКИ
до лабораторної роботи
«Рекурсивні підпрограми у програмах мовою Delphi»

з курсу «Програмування»
для студентів напряму 6.040302 – Інформатика
(спеціалізація «Соціальна інформатика»)

Затверджено редакційно-видавничою
радою університету,
протокол № 2 від 01.12.10.

Харків НТУ «ХПІ» 2011

Методичні вказівки до лабораторної роботи «Рекурсивні підпрограми у програмах мовою Delphi» з курсу «Програмування» для студентів напряму 6.040302 – Інформатика (спеціалізація «Соціальна інформатика») / Уклад. М. І. Безменов. – Х. : НТУ «ХПІ», 2011. – 12 с.

Укладач М. І. Безменов

Рецензент Л. М. Любчик

Кафедра системного аналізу і управління

ВСТУП

Рекурсивні функції та процедури є ефективним засобом реалізації циклічних алгоритмів.

Мета роботи – освоєння методики визначення та практичного застосування рекурсивних функцій та процедур у програмах, написаних мовою Delphi.

1. ТЕОРЕТИЧНІ ОСНОВИ

1.1. Рекурсивні підпрограми

Кожна з підпрограм може викликати інші підпрограми, у тому числі звертатися до самої себе. Підпрограма, що звертається до самої себе, називається ***рекурсивною***. Рекурсія може бути і неявною, коли підпрограма `first` викликає підпрограму `second`, а та у свою чергу викликає підпрограму `first`. У зв'язку з наявністю взаємних посилань підпрограм однієї на іншу виникає проблема, що зумовлена вимогою Delphi до описів: спочатку описати, потім використовувати. При неявній рекурсії підпрограми описуються з використанням випереджального опису за допомогою описової директиви **`forward`**.

Обов'язковим елементом в описі будь-якого рекурсивного процесу є деяке твердження (оператор), що визначає умову завершення рекурсії; іноді вона називається опорною умовою. В ній може бути задане яке-небудь фіксоване значення, що обов'язково буде досягнуте в ході рекурсивного обчислення, дозволяючи тим самим організувати своєчасну зупинку процесу. Крім того, повинен бути другий елемент – спосіб вираження одного кроку розв'язання за допомогою іншого, простішого. Кількість рівнів вкладеності не обмежена і може бути досить великою.

Кожен виклик рекурсивної підпрограми повинен наближати випадок, що зупиняє рекурсивні виклики, інакше виконання підпрограми ніколи не припиниться через нескінченний ланцюжок рекурсивних викликів. У дійсності нескінченний ланцюжок рекурсивних викликів не може реалізуватися, оскільки відбувається аварійне завершення програми. Найпростішим способом гарантування виконання опорної умови є зменшення деякого додатної величини до досягнення деякого «малого» значення.

Особливістю реалізації рекурсивних підпрограм є те, що у програмі існує тільки один екземпляр коду цієї підпрограми. На кожному рівні рекурсії здійснюється нове звертання до цього коду. Рис. 1.1 ілюструє виконання рекурсивного процесу з трьома рівнями рекурсивного виклику, правда для простоти по-

яснення **вважається**, що на кожному рівні створюється новий екземпляр коду. Розглянемо послідовність дій що виконується у цьому випадку.

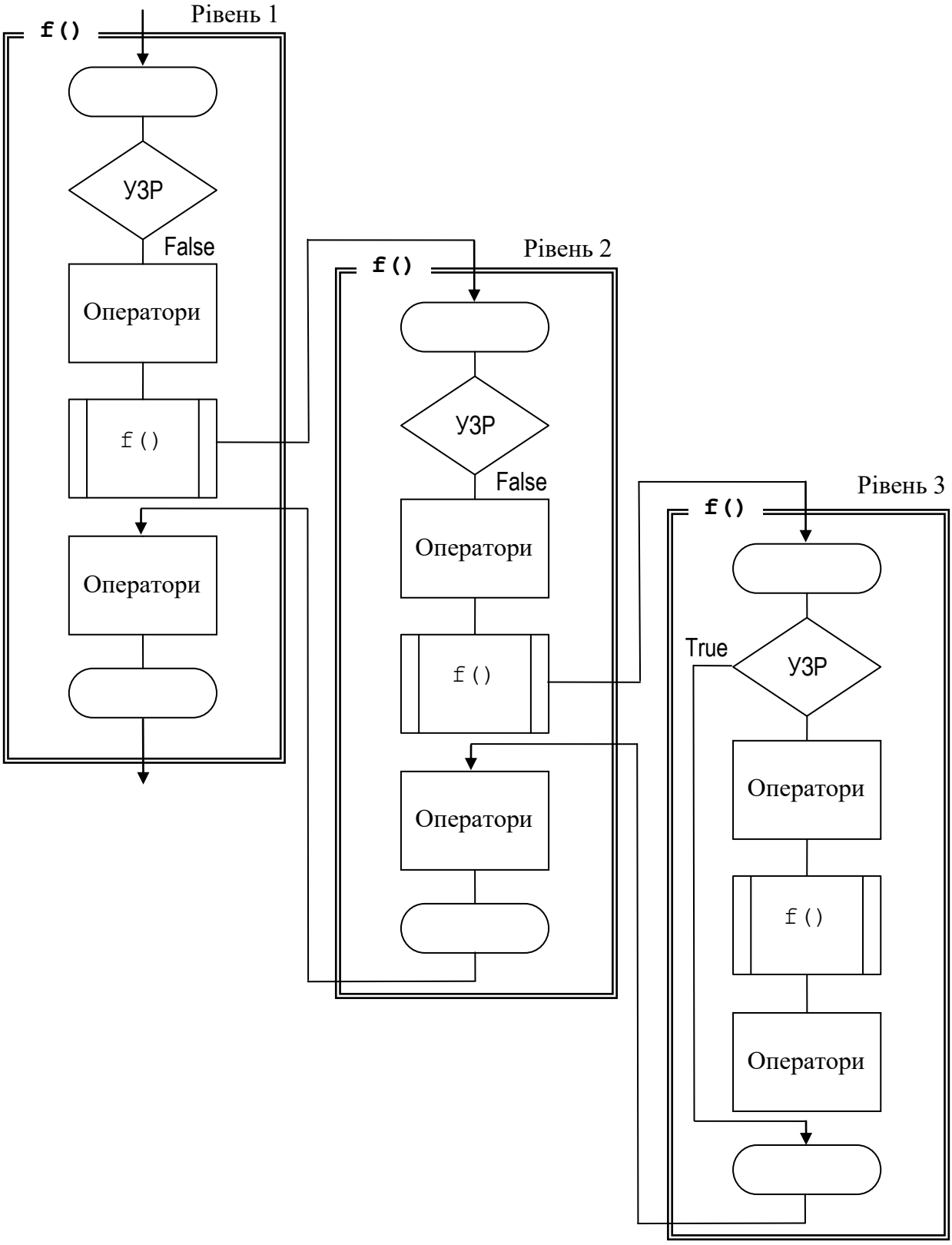

Рис. 1.1 – Схема виконання рекурсивного процесу
(УЗР – умова завершення рекурсії)

Виклик функції забезпечує початок виконання її коду. Вважаючи що на першому рівні рекурсії опорна умова не виконується, маємо виконання операторів, які йдуть за опорною умовою, і новий виклик підпрограми. На цьому виконання підпрограми на першому рівні тимчасово переривається, і починається другий рівень рекурсії. Зображений на рис. 1.1 другий рівень рекурсії аналогічний першому. Він також тимчасово переривається з переходом до третього рівня рекурсії.

Вважаючи, що на третьому рівні рекурсії опорна умова виконана, маємо завершення третього рівня з поверненням на другий рівень у точку, де здійснилося тимчасове переривання його виконання. Далі завершуються обчислення на другому рівні рекурсії з наступним поверненням на перший рівень. Після завершення обчислень на першому рівні, отримуємо остаточний результат. Звичайно, у випадку виконання опорної умови також можуть виконуватися відповідні оператори.

Незважаючи на простоту та наочність коду рекурсивних підпрограм, вони можуть вимагати дуже великих обсягів пам'яті для свого виконання. Це пояснюється тим, що, як і у випадку звичайних підпрограм, локальні змінні створюються в програмному стеку на кожному рівні рекурсивного виклику, що може привести до ситуації, коли обсяг стека просто вичерпається, оскільки знищення локальних змінних здійснюється тільки при виході з підпрограми.

2. ПРИКЛАДИ ПРОГРАМ

Приклад 1. Рекурсивна функція обчислення факторіала:

$$0! = 1,\ k! = k \cdot (k-1)!.$$

```
function factorialR(k: Integer): Integer;
begin
  if k = 0 then Result := 1        //Якщо опорна умова виконана
  else Result := k * factorialR (k - 1);              //Якщо ні
end;
```

Створимо форму з однорядковим редактором edInput1 (у властивість Text запишемо 0), багаторядковим редактором mmOutput1 (властивість Lines очистимо) та кнопкою Button1. Над редактором розмістимо мітку lbOutput1, записавши у її властивість Caption текст Уведіть ціле число від 0 до 12.

Наведений вище текст функції помістимо в секції **implementation** перед текстом опрацьовувача події OnClick кнопки Button1, що може мати такий вигляд:

```
procedure TForm1.Button1Click(Sender: TObject);
var
  N: Integer;
begin
  N := StrToInt(edInput1.Text);
  if (N < 0) or (N > 12) then begin
    mmOutput1.Lines.Add('Повторіть введення');
    edInput1.SetFocus;
  end
  else begin
    mmOutput1.Lines.Add(IntToStr(N) + '! = ' +
                        IntToStr(factorialR(N)));
    Button1.Enabled := False;
  end;
end;
```

У наведеному тексті в процедурі `TForm1.Button1Click` функція `factorial` після першого її виклику багаторазово звертається сама до себе зі значенням параметра `k`, яке зменшується від заданого при першому виклику початкового значення `N` до `0`. При досягненні параметром значення `0` виконується опорна умова і здійснюється вихід з функції. При цьому відбувається повернення в точку виклику функції на попередньому рівні з продовженням обчислення значення виразу `k*factorial(k-1)` і наступним виходом з функції з поверненням на попередній рівень.

Приклад 2. Визначити множення цілих чисел через додавання: $x \times 0 = 0$, $x \times y = x + x(y-1)$ при $y > 0$.

Скористаємося тією ж формою, що й у прикладі 1, записавши у властивість Caption мітки текст Уведіть перший співмножник. У секції **public** опису класу `TForm1` оголосимо поля `N1`, `N2` та `z` типу `Integer`, а перед описом процедури `TForm1.Button1Click` вставимо опис процедури `Multiplication`:

```
procedure Multiplication(x, y: Integer; var z: Integer);
begin
  if y < 0 then begin
    y := -y;
    x := -x;
  end;
  if y = 0 then z := 0                          //Опорна умова
  else begin                                  //Рекурсивний виклик
    Multiplication(x, y - 1, z);
```

```
    z := x + z;
  end;
end;
```

Опрацьовувач події OnClick для компонента Button1 у цій задачі може бути наступним:

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  if Tag = 0 then begin
    N1 := StrToInt(Trim(edInput1.Text));
    lbOutPut1.Caption := 'Уведіть другий співмножник';
  end
  else begin
    N2 := StrToInt(Trim(edInput1.Text));
    Multiplication(N1, N2, z);
    mmOutput1.Lines.Add(IntToStr(N1) + ' * ' + IntToStr(N2) +
                        ' = ' + IntToStr(z));
    Button1.Enabled := False;
  end;
  Tag := Tag + 1;
end;
```

У процедурі `Multiplication` для правильної організації рекурсії доводиться нарощувати змінну `z` після виходу з попереднього рівня рекурсії.

Приклад 3. Визначити і використати рекурсивну функцію підсумовування елементів масиву, що містить n дійсних чисел ($n \le 20$).

Розмістимо на формі однорядковий редактор Edit1 для введення чисел (у його властивість Text запишемо число 1), багаторядковий редактор Memo1 для виведення результату та кнопку Button1 (у її властивість Caption запишемо текст Ввести). Над редактором розмістимо мітку, записавши у її властивість Caption підказку N. Перед описом класу TForm1 опишемо тип

```
T20Real = array [1..20] of Real;
```

а у секції **public** опису класу форми впишемо два рядки

```
N: Integer;
a: T20Real;
```

визначивши відповідно змінні для позначення розміру масиву і самого масиву.

У верхній частині секції **implementation** модуля перед кодом опрацьовувача події OnClick кнопки Button1 опишемо функцію:

```
function SumRecurs(a: T20Dbl; Count: Integer): Real;
begin
                // Якщо в масиві один елемент, то він і є сумою
  if Count = 1 then Result := a[1]      // Завершення рекурсії
  else    // Інакше до останнього елемента додаємо суму решти
          //елементів, яка обчислюється
          // рекурсивним викликом функції
    Result := a[Count] + SumRecurs(a, Count - 1);
end;
```

Код оброблювача події OnClick кнопки Button1 може бути таким:

```
procedure TForm1.Button1Click(Sender: TObject);
var
  i: Integer;
begin
  Edit1.SetFocus;
  if Tag = 0 then begin
    DecimalSeparator := '.';
    N := StrToInt(Edit1.Text);
    Label1.Caption := 'a[' + IntToStr(Tag + 1) + ']';
  end
  else begin
    a[Tag] := StrToFloat(Edit1.Text);
    Label1.Caption := 'a[' + IntToStr(Tag + 1) + ']';
    if Tag = N then begin
      Memo1.Lines.Add('Початковий масив');
      for i := 1 to N do
        Memo1.Lines.Add('a[' + IntToStr(i) + '] = '
                        + FloatToStr(a[i]));
      Memo1.Lines.Add('Сума елементів масиву дорівнює '
                      + FloatToStr(SumRecurs(a, N)));
      Button1.Visible := False;
      Label1.Visible := False;
      Edit1.Visible := False;
    end;
  end;
  Tag := Tag + 1;
end;
```

3. ЗАВДАННЯ НА ЛАБОРАТОРНУ РОБОТУ

За час, відведений для виконання лабораторної роботи (2 академічні години), студент повинен:

1. Розробити алгоритм розв'язання задачі, запропонованої для програмування.
2. Здійснити проектування форми для функціонування розроблюваної програми.
3. Здійснити програмну реалізацію розробленого алгоритму.
4. Здійснити відлагодження програми, виправивши синтаксичні та логічні помилки.
5. Підібрати тестові дані для перевірки програми, включаючи виняткові випадки.
6. Оформити звіт до лабораторної роботи.
7. Відповісти на контрольні запитання.
8. Здати викладачу працездатну програму з демонстрацією її роботи на декількох варіантах вихідних даних.

4. ВАРІАНТИ ЗАДАЧ

1. Дано натуральні числа n, m. Знайти їх найбільший спільний дільник (НСД). Для розв'язання задачі визначити й використати рекурсивну підпрограму обчислення НСД, засновану на співвідношенні НСД(n, m) = = НСД(m, r), де r – остача від ділення n на m.
2. Числа Фібоначчі u_0, u_1, u_2, … визначаються за таким правилом: $u_0 = 0$, $u_1 = 1$, $u_n = u_{n-1} + u_{n-2}$ $(n = 2, 3, ...)$. Скласти рекурсивну підпрограму обчислення u_n для даного невід'ємного цілого n, що базується на безпосередньому використанні співвідношення $u_n = u_{n-1} + u_{n-2}$.
3. Дано невід'ємні цілі числа n та m. Написати рекурсивну підпрограму для обчислення функції Акермана:

$$A(n,m) = \begin{cases} m+1, & \text{якщо } n = 0, \\ A(n-1,1), & \text{якщо } n \neq 0,\ m = 0, \\ A(n-1, A(n, m-1)), & \text{якщо } n > 0,\ m > 0. \end{cases}$$

4. Скласти рекурсивну підпрограму, що обчислює суму двох цілих невід'ємних чисел шляхом багаторазового додавання числа 1. Наприклад, 6 + 10 = (6 + 1) + 9 = (7 + 1) + 8 = ...

5. Визначити функцію множення цілих чисел через додавання: $x \times 0 = 0$, $x \times y = x + x(y-1)$ при $y > 0$. Дано два цілих числа. Знайти їх добуток за допомогою розробленої функції.
6. Описати функцію для обчислення C_n^m з використанням рекурсії:

$$C(0,n) = C(n,n) = 1,$$
$$C(m,n) = C(m,n-1) + C(m-1,n-1) \text{ при } 0 < m < n.$$

Дано цілі числа n і m ($n > 0$, $0 \le m \le n$). За допомогою описаної функції знайти значення C_n^m.

7. Дано натуральні числа m, n_1, n_2, …, n_m ($m > 1$). Обчислити найбільший спільний дільник чисел n_1, n_2, …, n_m: НСД($n_1, n_2, \ldots, n_m$). Відомо, що для будь-якого натурального $k > 1$ виконується таке співвідношення:

$$\text{НСД}(n_1, n_2, \ldots, n_k) = \text{НСД}(\text{НСД}(n_1, n_2, \ldots, n_{k-1}), n_k).$$

Визначити рекурсивну функцію обчислення найбільшого спільного дільника елементів масиву натуральних чисел.

8. Описати функцію відшукання мінімального елемента масиву з n дійсних чисел, де n – натуральне число. У функції не повинно бути явної організації циклу. Скористатися цією функцією для пошуку мінімуму в заданому одновимірному масиві.
9. Описати рекурсивну підпрограму сортування за зростанням масиву з n цілих чисел. Ідея методу така: помістити найменший елемент на першу позицію, після чого відсортувати частину масиву, що залишилася, за допомогою рекурсивного виклику.
10. Описати рекурсивну функцію з одним аргументом, яка без використання операторів циклу перевіряє, чи є одновимірний цілочисловий масив симетричним.
11. Дано квадратну матрицю $\boldsymbol{A}$ розміру $n \times n$, де n – натуральне число. Обчислити її визначник. Використати рекурсивну підпрограму обчислення визначника розкладанням за першим рядком.
12. Описати рекурсивну підпрограму з трьома аргументами (одновимірним дійсним масивом і двома індексами-обмежниками), що не використовує оператори циклу. Підпрограма повинна розвертати на 180º частину одновимірного дійсного масиву від першого індексу-обмежника до другого включно. Протестувати цю підпрограму для таких випадків:
    а) перший індекс-обмежник менший за другий;
    б) перший індекс-обмежник більший від другого;
    в) індекси-обмежники рівні.

5. КОНТРОЛЬНІ ЗАПИТАННЯ

1. Що таке рекурсія?
2. Що повинно бути у підпрограмі для правильної організації рекурсії?
3. Що таке неявна рекурсія?
4. Як треба описувати підпрограми для неявної рекурсії?
5. Чи може бути нескінченним рекурсивний процес?
6. Що можна сказати про обсяги пам’яті, які необхідні для роботи рекурсивних підпрограм?
7. Перелічіть переваги та недоліки звичайних і рекурсивних підпрограм.

СПИСОК ЛІТЕРАТУРИ

1. Безменов, М. І. Основи програмування в середовищі Delphi : навч. посіб. / М. І. Безменов. – Х. : НТУ «ХПІ», 2010. – 608 с.
2. Кэнту, М. Delphi 7 : Для профессионалов / М. Кэнту – СПб. : Питер, 2004. – 1101 с.
3. Архангельский, А. Я. Программирование в Delphi 6 / А. Я. Архангельский. – М. : БИНОМ, 2002. – 1120 с.
4. Дарахвелидзе, П. Г. Программирование в Delphi 7 / П. Г. Дарахвелидзе, Е. П. Марков. – СПб. : БХВ-Петербург, 2003. – 784 с.
5. Культин, Н. Б. Основы программирования в Delphi 7 / Н. Б. Культин. – СПб. : БХВ-Петербург, 2003. – 608 с.
6. Пестриков, В. М. Delphi на примерах / В. М. Пестриков, А. Н. Маслобоев. – СПб. : БХВ-Петербург, 2005. – 496 с.
7. Ремкеев, А. А. Курс Delphi для начинающих. Полигон нестандартных задач / А. А. Ремкеев, С. В. Федотова. – М. : СОЛОН-Пресс, 2006. – 360 с.
8. Митчелл, К. Керман. Программирование и отладка в Delphi™ : учебный курс / Митчелл К. Керман. – М. : Вильямс, 2004. – 720 с.
9. Парижский, С. М. Delphi : Только практика / С. М. Парижский. – К. : МК-Пресс, 2005. – 208 с.
10. Культин, Н. Б. Основы программирования в Delphi 2007 / Н. Б. Культин. – СПб. : БХВ-Петербург, 2008. – 480 с.

Навчальне видання

Методичні вказівки
до лабораторної роботи
«Рекурсивні підпрограми у програмах мовою Delphi»
з курсу «Програмування» для студентів напряму 6.040302 – Інформатика
(спеціалізація «Соціальна інформатика»)

Укладач БЕЗМЕНОВ Микола Іванович

Відповідальний за випуск О. С. Куценко
Роботу до видання рекомендував О. В. Горелий

За авторською редакцією

План 2011 р., поз. 4 / 76-11

Підп. до друку 23.05.2011 р. Формат 60 × 84 $^{1}/_{16}$. Папір офісний.
Riso-друк. Гарнітура Таймс. Ум. друк. арк. 0,6. Наклад 50 прим.
Зам. № 160. Ціна договірна.

Видавничий центр НТУ «ХПІ».
Свідоцтво про державну реєстрацію ДК № 3657 від 24.12.2009 р.
61002, Харків, вул. Фрунзе, 21

Друкарня НТУ «ХПІ», 61002, Харків, вул. Фрунзе, 21