Розв'язування задач, у яких використовуються обчислення за ітераційними формулами. Вкладені цикли. Оператори переривання циклів. Розв’язування задач, що потребують комбінування циклічних з операторами розгалуження

Розв'язування задач, у яких використовуються обчислення за ітераційними формулами. Вкладені цикли

Розв'язування задач, у яких використовуються обчислення за ітераційними формулами

Дуже часто в задачах програмування необхідно одержати доступ до цифр цілого числа. Це може знадобитися для перетворення чисел з однієї системи числення в іншу, для знайдення суми цифр числа, для відшукання найбільшої і найменшої цифр у записі числа й багато для чого ще. Розглянемо на прикладі подібні перетворення.

Приклад 9. Дано чотиризначне ціле додатне число. Необхідно одержати суму цифр, що складають запис цього числа. Наприклад, дано число 1234; відповідь: 1 + 2 + 3 + 4 = 10.

Розв’язання. Спробуємо написати алгоритм перетворення — розбиття числа на цифри за розрядами.

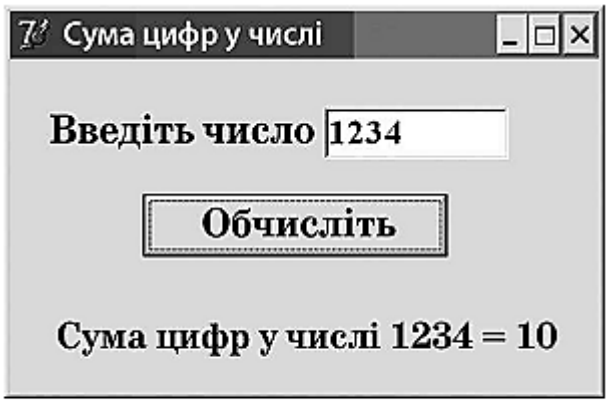

Рис. 9.2. Визначення суми цифр у записі числа

1234 mod 10 = 4. Ми виділили розряд одиниць.

1234 div 10 = 123. Число стало в 10 разів меншим, а розряд десятків опинився останнім. 123 mod 10 = 3. Виділили розряд десятків. 123 div 10 = 12. Наше число стало в 10 разів меншим, а розряд сотень опинився останнім. 12 mod 10 = 2. Виділили розряд сотень. 12 div 10 = 1. Число стало в 10 разів меншим, а розряд тисяч опинився останнім.

1 mod 10 = 1. Ми виділили розряд тисяч. Тепер якщо ми зможемо підсумувати остачі від ділення, то одержимо потрібний результат. При цьому простежується циклічна послідовність дій. На рис. 9.2 подано інтерфейс завдання. Програма матиме такий вигляд:

```
unit Unit1;
interface
uses
   Windows, Messages, SysUtils, Variants, Classes, Graphics,
    Controls, Forms, Dialogs, StdCtrls;
type
   TForml = class(TForm)
   Label1: TLabel;
   Editl: TEdit;
   Button1: TButton;
   Label2: TLabel;
   procedure Button1Click(Sender: TObject);
 private
  { Private declarations }
 public
  { Public declarations }
 end;
var
       Forml: TForml;
implementation
{$R *.dfm}
procedure TForm1.Button1Click(Sender: TObject);
var
    x,i,s,ost,y: integer;
begin
   x:=strtoint(Edit1.Text);
   s: =0;
  y:=x;
  for i:=1 to 4 do
  begin
     ost:=x mod 10; x:=x div 10; s:=s+ost;
  end;
  x:=y;
  label2.Caption:='Сума цифр у числі ' +inttostr(x) +'=' +inttostr(s);
end;
end.
```

Використання циклу **FOR** для розв'язування подібних завдань виправдане лише у випадку, якщо відома кількість цифр у записі числа. Якщо ж кількість цифр не відома, то зручно скористатися циклом

WHILE. Для розв'язання цього завдання в загальному вигляді перепишемо цикл:

```
var
   x,i,s,ost,y: integer;
begin
   x:=strtoint(Edit1.Text);
   s: =0;
   y:=x;
   while x>0 do
   begin
      ost:=x mod 10;
      x:=x div 10;
      s:=s+ost;
   end;
   x:=y;
   label2.Caption:='Сума цифр у числі ' +inttostr(x) +'=' +inttostr(s);
end;
```

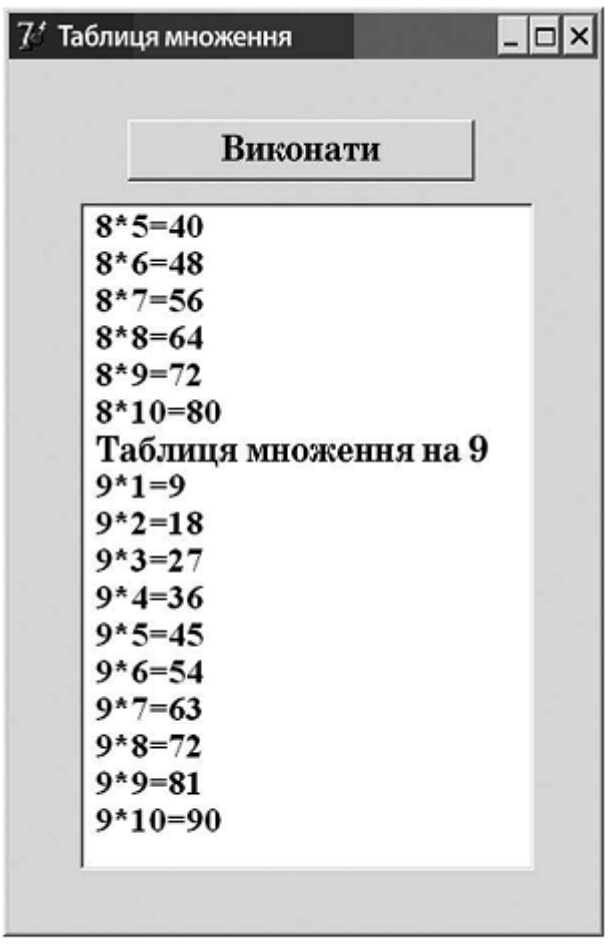

Рис. 9.3. Приклад використання вкладеного циклу

Вкладені цикли

Вкладений цикл — це частина алгоритму або програми, що неодноразово виконується і перебуває в тілі іншого, зовнішнього, циклу.

Для кожного значення змінної-лічильника зовнішнього циклу лічильник вкладеного циклу встигає пробігти всі свої значення.

Ніяких спеціальних конструкцій для вкладених циклів немає. Усе працює так, як і у випадку використання звичайних циклів. Застосовуються вони здебільшого для оброблення таблиць (двовимірних масивів).

Розглянемо найпростіший приклад застосування вкладеного циклу.

Приклад 10. Вивести таблицю множення.

Розв'язання. Необхідно створити цикл, що послідовно надає множнику значення від 1 до 9, формуючи стовпчик добутків. А щоб вивести таблицю (перемноживши числа від 1 до 10), потрібен ще один такий самий цикл. На рис. 9.3 наведено інтерфейс цього завдання. Програма матиме такий вигляд:

```
unit Unit1;
interface
uses
    Windows, Messages, SysUtils, Variants, Classes,
    Graphics, Controls, Forms, Dialogs, StdCtrls;
type
    TForm1 = class(TForm)
      Button1: TButton;
      Memo1: TMemo;
    procedure Button1Click(Sender: TObject);
 private
   { Private declarations }
public
   { Public declarations }
end;
var
    Forml: TForml;
implementation
    {$R *.dfm}
procedure TForm1.Button1Click(Sender: TObject);
var
    i, j: integer;
begin
    Memo1.Lines.C1ear;
    for i:=1 to 9 do
    begin
      Memo1.Lines.Add('Таблиця множення на '+inttostr(i));
      for j :=1 to 10 do
        Memo1.Lines.Add(inttostr( i ) +' * ' + inttostr(j)+' = ' +
        inttostr(i*j));
 end;
 end;
end.
```

Переривання й продовження циклу

Виконання фіксованого числа ітерацій не завжди приводить до потрібного результату. Іноді можуть виникнути ситуації, коли було б логічним перервати цикл, не виконуючи його до кінця, тобто просто виключити всі дальші ітерації. Така можливість існує — для цього необхідно скористатися командою **Break.** Ця команда завершує цикл, що виконується в цей момент, і продовжує дальше виконування програми. При цьому поточна ітерація до кінця не виконується — переривання відбувається саме в тому рядку, на якому записана команда **Break.**

Якщо цикл, виконання якого переривається командою **Break**, вкладений в інший цикл, то зовнішній цикл продовжить виконуватися, тобто команда **Break** зупиняє тільки один цикл, а не всі наявні.

Команда **Break** — це вихід із циклу. Іноді ж потрібно просто пропустити поточну ітерацію й перейти до наступної. Вручну це можна зробити, узявши всі команди в блок умовного оператора, однак такий спосіб не дуже зручний. Саме тому існує команда продовження циклу, вона називається **Continue.** Ця команда змушує цикл відразу перейти до наступної ітерації, не продовжуючи виконання поточної.

ТЕОРЕТИЧНІ ПИТАННЯ ДЛЯ САМОКОНТРОЛЮ

1. Поясніть на прикладах використання операцій div і mod.
2. Опишіть алгоритм переведення цілого додатного числа, заданого в десятковій системі числення, у двійкову систему.
3. У чому особливість використання циклів WHILE і FOR у переведенні чисел у двійкову систему числення?
4. Що таке вкладені цикли?
5. Яким оператором здійснюється переривання циклу? У яких випадках таке переривання необхідне?

ПРАКТИЧНІ ЗАВДАННЯ

1. Ввести на форму ціле чотирьохцифрове додатне число. Перевірити правильність твердження: сума перших двох цифр дорівнює сумі двох останніх. Наприклад: 4581 →4 + 5 = 8 + 1.
2. Ввести на форму ціле чотирьохцифрове додатне число. Перевірити правильність твердження: задане число симетричне (числа, запис яких читається однаково справа наліво і зліва направо, називаються паліндромами). Наприклад: 1551, 8778 і т. ін.
3. Ввести на форму ціле чотирьохцифрове додатне число. Перевірити, чи містить задане число три однакові цифри. Наприклад: 6676, 4544, 6000.
4. Ввести на форму ціле чотирьохцифрове додатне число. Перевірити, чи містить задане число чотири різні цифри. Наприклад: 1234, 8976, 3461.
5. Ввести на форму ціле чотирьохцифрове додатне число. Показати, на яких позиціях (розряди) у заданому числі стоять цифри, кратні числу 3.
6. Ввести на форму ціле чотирьохцифрове додатне число. Перевірити, чи всі цифри заданого числа парні. Наприклад: 6842 — «так», 6354 — «ні».