Перевод натуральных чисел в различные системы счисления и выполнение арифметических операций над ними. Программа на языке Pascal

Учебный проект

2011 г.

Цель

Написать программу на языке программирования Паскаль (Pascal), которая

1. принимает два числа в любой из четырех систем счисления (2-й, 8-й, 10-й или 16-й),
2. переводит их в остальные системы счисления,
3. выполняет над числами указанную арифметическую операцию (+, -, *, /),
4. выводит результат во всех четырех системах счисления.

Для упрощения программа будет рассчитана на работу с целыми положительными числами.

Ввод данных пользователем

1. Система счисления: 2, 8, 10, 16.
2. Первое число: от 0 до 32767.
3. Второе число: от 0 до 32767.
4. Операция: +, -, *, / (выполняется только деление нацело).

Вывод данных на экран

2	8	10	16
1-е число	1-е число	1-е число	1-е число
2-е число	2-е число	2-е число	2-е число
результат	результат	результат	результат

Этапы написания программы

1. Перевод числа, заданного в десятичной системе счисления, во все остальные системы счисления. Организация вывода результата на экран.
2. Перевод числа, заданного в любой системе счисления, во все остальные системы счисления.
3. Перевод двух чисел в десятичную систему счисления (если требуется). Выполнение заданной арифметической операции. Перевод исходных чисел и результата во все остальные системы счисления. Организация вывода результата на экран.
4. Обработка корректности ввода. Управляемый выход из программы.
5. *(дополнительно) Операции с отрицательными и дробными числами, оформление вывода с помощью модуля* ***Crt****.*

1 этап. Перевод числа, заданного в десятичной системе счисления, во все остальные системы счисления

Алгоритм основной ветки программы

1. Записать в переменную десятичное число.
2. Вызвать функцию перевода числа в двоичную систему счисления. Результат присвоить переменной.
3. Вызвать функцию перевода числа в восьмеричную систему счисления. Результат присвоить переменной.
4. Вызвать функцию перевода числа в шестнадцатеричную систему счисления. Результат присвоить переменной.
5. Вывод переменных на экран в табличной форме.

Алгоритмы перевода десятичного числа в иные системы счисления

См. перевод десятичного числа в двоичную и восьмеричную системы счисления, перевод десятичного числа в шестнадцатеричное число.

Код программы

```
var
   decimal: integer;
   binary, octal, hexa: string;

function binary_octal(bin_oct:byte;decimal:integer): string;
begin
   binary_octal := '';
   while decimal > 0 do begin
      binary_octal := chr(ord('0') + (decimal mod bin_oct)) + binary_octal;
      decimal := decimal div bin_oct
   end;
end;

function hexadecimal(decimal:integer):string;
var digit:byte; ch:char;
begin
   hexadecimal := '';
   while decimal > 0 do begin
      digit := decimal mod 16;
      if digit in [10..15] then
         case digit of
         10: ch := 'A';
         11: ch := 'B';
         12: ch := 'C';
         13: ch := 'D';
         14: ch := 'E';
         15: ch := 'F'
```

```
            end
        else
            ch := chr(ord('0') + digit);
        hexadecimal := ch + hexadecimal;
        decimal := decimal div 16
    end;
end;

begin
    write('Decimal: ');
    readln(decimal);

    binary := binary_octal(2,decimal);
    octal := binary_octal(8,decimal);
    hexa := hexadecimal(decimal);
    writeln('notation:':10,'2':10,'8':5,'10':5,'16':5);
    writeln('value:':10,binary:10,octal:5,decimal:5,hexa:5);

readln
end.
```

2 этап. Перевод числа, заданного в любой системе счисления, во все остальные системы счисления

Алгоритм основной ветки программы

1. В зависимости от указанной системы счисления записать в определенную переменную число.
2. Если число двоичное, то перевести его в десятичную систему счисления. Полученное десятичное число перевести в восьмеричную и шестнадцатеричную системы счисления.
3. Если число восьмеричное, то перевести его в десятичную систему счисления. Полученное десятичное число перевести в двоичную и шестнадцатеричную системы счисления.
4. Если число десятичное, то перевести его в двоичную, восьмеричную и шестнадцатеричную системы счисления.
5. Если число шестнадцатеричное, то перевести его в десятичную систему счисления. Полученное десятичное число перевести в восьмеричную и шестнадцатеричную системы счисления.
6. Вывести данные на экран.

Алгоритмы перевода числа в иные системы счисления

См. перевод десятичного числа в двоичную и восьмеричную системы счисления, перевод десятичного числа в шестнадцатеричное число, перевод двоичного числа в десятичное, перевод шестнадцатеричного числа в десятичное число.

Перевод восьмеричного числа в десятичное почти аналогичен переводу двоичного.

Код программы

```
var
   notation: byte;
   decimal: integer; binary, octal, hexa: string;

function decimal_binary(decimal:integer): string;
begin
   decimal_binary := '';
   if decimal = 0 then decimal_binary := '0';
   while decimal > 0 do begin
      decimal_binary := chr(ord('0') + (decimal mod 2)) + decimal_binary;
      decimal := decimal div 2
   end;
end;
```

```
function decimal_octal(decimal:integer): string;
begin
   decimal_octal := '';
   if decimal = 0 then decimal_octal := '0';
   while decimal > 0 do begin
      decimal_octal := chr(ord('0') + (decimal mod 8)) + decimal_octal;
      decimal := decimal div 8
   end;
end;

function decimal_hexa(decimal:integer):string;
var digit:byte; ch:char;
begin
   decimal_hexa := '';
   if decimal = 0 then decimal_hexa := '0';
   while decimal > 0 do begin
      digit := decimal mod 16;
      if digit in [10..15] then
         case digit of
         10: ch := 'A';
         11: ch := 'B';
         12: ch := 'C';
         13: ch := 'D';
         14: ch := 'E';
         15: ch := 'F'
         end
      else
         ch := chr(ord('0') + digit);
      decimal_hexa := ch + decimal_hexa;
      decimal := decimal div 16
   end;
end;

function binary_decimal(binary:string):integer;
var n,m,i,j,digit:byte; pow:integer;
begin
   n := length(binary);
   binary_decimal := 0;
   m := n;
   for i:=1 to n do begin
      digit := ord(binary[i]) - ord('0');
      m := m - 1;
      if digit = 1 then begin
         pow := 1;
         for j:=1 to m do
            pow := pow * 2;
         binary_decimal := binary_decimal + pow;
      end;
   end;
end;

function octal_decimal(octal:string):integer;
var n,m,i,j,digit:byte; pow:integer;
begin
   n := length(octal);
   octal_decimal := 0;
   m := n;
```

```
    for i:=1 to n do begin
        digit := ord(octal[i]) - ord('0');
        m := m - 1;
        pow := 1;
        for j:=1 to m do
            pow := pow * 8;
        octal_decimal := octal_decimal + digit * pow;
    end;
end;

function hexa_decimal(hexa:string):integer;
var n,m,i,j,digit:byte; pow:integer; ch: char;
begin
    n := length(hexa);
    hexa_decimal := 0;
    m := n;
    for i:=1 to n do begin
        ch := hexa[i];
        if ch in ['A'..'F'] then
            case ch of
            'A': digit := 10;
            'B': digit := 11;
            'C': digit := 12;
            'D': digit := 13;
            'E': digit := 14;
            'F': digit := 15
            end
        else
            digit := ord(ch) - ord('0');
        m := m - 1;
        pow := 1;
        for j:=1 to m do
            pow := pow * 16;
        hexa_decimal := hexa_decimal + digit * pow;
    end;
end;

begin
    write('Notation: '); readln(notation);
    write('Number: ');
    if notation = 2 then begin
        readln(binary);
        decimal := binary_decimal(binary);
        octal := decimal_octal(decimal);
        hexa := decimal_hexa(decimal)
    end
    else
        if notation = 8 then begin
            readln(octal);
            decimal := octal_decimal(octal);
            binary := decimal_binary(decimal);
            hexa := decimal_hexa(decimal)
        end
        else
            if notation = 10 then begin
                readln(decimal);
                binary := decimal_binary(decimal);
                octal := decimal_octal(decimal);
```

```
            hexa := decimal_hexa(decimal)
        end
        else
            if notation = 16 then begin
                readln(hexa);
                decimal := hexa_decimal(hexa);
                binary := decimal_binary(decimal);
                octal := decimal_octal(decimal)
            end
            else
                writeln('Must be goto end.');

    writeln('notation:':10,'2':10,'8':5,'10':5,'16':5);
    writeln('value:':10,binary:10,octal:5,decimal:5,hexa:5);

readln
end.
```

3 этап. Выполнение арифметической операции. Перевод исходных чисел и результата во все остальные системы счисления

Алгоритм основной ветки программы

1. Узнать систему счисления.
2. Дважды вызвать процедуру, которая заполняет переменные значениями. В процедуру передавать переменные, а не значения (использование **var**).
3. Узнать требуемую математическую операцию.
4. Выполнить операцию над числами в десятичной системе счисления.
5. Перевести результат в остальные системы счисления.
6. Организовать вывод данных на экран.

Переменные

decimal1, binary1, octal1, hexa1 – варианты представления первого числа;

decimal2, binary2, octal3, hexa4 – варианты представления второго числа;

decimal_res, binary_res, octal_res, hexa_res – варианты представления результата;

Код программы

```
var
   notation: byte;
   decimal1: integer; binary1, octal1, hexa1: string;
   decimal2: integer; binary2, octal2, hexa2: string;
   decimal_res: longint; binary_res, octal_res, hexa_res: string;
   operation: char;

function decimal_binary(decimal:integer): string;
begin
   см. предыдущий этап
end;

function decimal_octal(decimal:integer): string;
begin
   см. предыдущий этап
end;

function decimal_hexa(decimal:integer):string;
var digit:byte; ch:char;
begin
   см. предыдущий этап
end;

function binary_decimal(binary:string):integer;
var n,m,i,j,digit:byte; pow:integer;
begin
   см. предыдущий этап
```

```
end;

function octal_decimal(octal:string):integer;
var n,m,i,j,digit:byte; pow:integer;
begin
   см. предыдущий этап
end;

function hexa_decimal(hexa:string):integer;
var n,m,i,j,digit:byte; pow:integer; ch: char;
begin
   см. предыдущий этап
end;

procedure manager(var decimal:integer; var binary,octal,hexa:string;
notation:byte);
begin
   if notation = 2 then begin
      readln(binary);
      decimal := binary_decimal(binary);
      octal := decimal_octal(decimal);
      hexa := decimal_hexa(decimal)
   end
   else
      if notation = 8 then begin
         readln(octal);
         decimal := octal_decimal(octal);
         binary := decimal_binary(decimal);
         hexa := decimal_hexa(decimal)
      end
      else
         if notation = 10 then begin
            readln(decimal);
            binary := decimal_binary(decimal);
            octal := decimal_octal(decimal);
            hexa := decimal_hexa(decimal)
         end
         else
            if notation = 16 then begin
               readln(hexa);
               decimal := hexa_decimal(hexa);
               binary := decimal_binary(decimal);
               octal := decimal_octal(decimal)
            end
            else
               writeln('Must be goto end.');
end;

begin
   write('Notation: '); readln(notation);
   write('Number one: ');
   manager(decimal1,binary1,octal1,hexa1,notation);
   write('Number two: ');
   manager(decimal2,binary2,octal2,hexa2,notation);

   write('Operation (+,-,*,/): ');
   readln(operation);
   if operation = '+' then
```

```
        decimal_res := decimal1 + decimal2
    else
        if operation = '-' then
            decimal_res := decimal1 - decimal2
        else
            if operation = '*' then
                decimal_res := decimal1 * decimal2
            else
                if operation = '/' then
                    decimal_res := decimal1 div decimal2
                else
                    writeln('Error operation');
    binary_res := decimal_binary(decimal_res);
    octal_res := decimal_octal(decimal_res);
    hexa_res := decimal_hexa(decimal_res);

    writeln;
    writeln('notation:':10,'2':10,'8':5,'10':5,'16':5);
    writeln;
    writeln('value:':10,binary1:10,octal1:5,decimal1:5,hexa1:5);
    writeln('value:':10,binary2:10,octal2:5,decimal2:5,hexa2:5);
    writeln(' ':10,'------------------------');
    writeln('result:':10,binary_res:10,octal_res:5,decimal_res:5,hexa_res:5);


readln
end.
```

4 этап. Обработка корректности ввода. Управляемый выход из программы

Обычно в программировании при обработке корректности ввода используют синтаксические конструкции, обеспечивающие так называемую обработку исключений (т.е. исключительных, их ряда вон выходящих ситуаций). Если вы не знаете, как это делается и возможно ли это в языке программирования Паскаль, то можно пойти другим путем. Все, что вводит пользователь, следует «воспринимать» как строку. Это позволит избежать аварийного завершения программы, когда данным численного типа присваиваются нечисловые символы (пользователь осуществляет некорректный ввод). Однако следует помнить, что при этом объем кода существенно возрастает.

Приведем несколько примеров.

Пусть все, что вводит пользователь, сначала присваивается строчной переменной *userinput*. Также будем использовать метки.

Проверка корректности ввода системы счисления:

```
lab_notation:
 write('Notation: '); readln(userinput);
 if (length(userinput) = 1) and ((userinput[1] = '2') or
 (userinput[1] = '8')) then
    notation := ord(userinput[1]) - ord('0')
 else
    if (length(userinput) = 2) and (userinput[1] = '1') and
    (userinput[2] = '6') then
       notation := 16
    else
       if (length(userinput) = 2) and (userinput[1] = '1') and
       (userinput[2] = '0') then
          notation := 10
       else
          goto lab_notation;
```

Проверка корректности ввода чисел существенно сложнее, т.к. надо учитывать предел допустимых значений и используемые знаки. Опустим ее из-за сложности.

Проверка знака операции приводилась на третьем этапе. Остается лишь в последнюю ветку **else** добавить метку перехода на начало ввода операции при некорректном вводе.

Для управляемого выхода из программы можно организовать переход на ее начало при нажатии клавиши. Проще всего это сделать с помощью оператора **goto**. Следует иметь в виду, что от **goto** можно легко избавиться, например, с помощью цикла **while**.

```
write('Continue (y)? ');
readln(start);
if start = 'y' then
   goto lab_notation;
```