Решения Загогулин

для начинающего программиста

Редакция 1 от 2013-03-29

Количество решений: 9

5.21 Отсев простых чисел решетом Эратосфена

```
{ Отсев простых чисел решетом эратосфена }
const CSize = 1000;
type TArr = array [1..CSize] of integer;
var Arr  : TArr;
    i, k : integer;
begin
  { Заполнение массива }
  for i:= 1 to CSize do Arr[i]:= i;
  { Вычёркивание части чисел }
  for i:= 2 to CSize div 2 do
    if Arr[i]<>0 then
      for k:=2*i to CSize do
            if (Arr[k] mod i)=0 then Arr[k]:=0;
  { Вывод оставшихся чисел }
  for i:= 1 to CSize do
    if Arr[i]<>0 then Write(i:4);
  Readln;
end.
```

5.23 Подсчёт несовпадающих чисел (2)

```
{ Подсчёт количества несовпадающих чисел в массиве }
const CSize = 1024;
type TArray = array [1..CSize] of integer;
function Unique(const arg: TArray): integer;
type TFlags = array [1..CSize] of boolean;
var i, k : integer;
    Flags: TFlags;
begin
  { Очищаем массив флагов }
  FillChar(Flags, SizeOf(Flags), false);
  { Устанавливаем флаги во всех последующих позициях,
    где числа совпадают с текущим }
  for i:=1 to CSize-1 do
    if not Flags[i]  { если нашли уникальное }
      then for k:=i+1 to CSize do
        Flags[k]:= Flags[k] or (arg[i]=arg[k]);
  { Подсчитываем уникальные числа (элементы, где флаг=false ) }
  k:=0;
  for i:=1 to CSize do
        if not Flags[i] then Inc(k);
  Unique:=k;
end;
var Arr : TArray;
      i : integer;
begin
  FillChar(Arr, SizeOf(Arr), 0);
  Writeln ('Zero= ', Unique(Arr));
  for i:= 1 to CSize do Arr[i]:=i;
  Writeln ('CSize= ', Unique(Arr));
  for i:= 1 to CSize do Arr[i]:=Random(50);
  Writeln ('Random(50)= ', Unique(Arr));
  for i:= 1 to CSize do Arr[i]:=Random(200);
  Writeln ('Random(200)= ', Unique(Arr));
  Readln;
end.
```

6.12 Сортировка чётных элементов массива

```
{ Сортировка чётных элементов массива }

const CSize = 20;
type TArray = array [1..CSize] of integer;
  { Сортировка чётных элементов }
procedure SortEven(var arg: TArray);
var ind : TArray;
    i, k, cnt : integer;
    tmp : integer;
begin
  cnt:= 0;
  { Формируем вспомогательный массив индексов и счётчик }
  for i:= 1 to CSize do
    if (arg[i] mod 2)=0 then begin
      Inc(cnt);
      ind[cnt]:=i;
    end;
  { Сортируем выбором }
  for k:= 1 to cnt-1 do
    for i:=cnt downto k+1 do
      if arg[ind[k]] > arg[ind[i]] then begin
        tmp:= arg[ind[i]];
        arg[ind[i]]:= arg[ind[k]];
        arg[ind[k]]:= tmp;
      end;
end;
var X : TArray;
    i : integer;
begin
  Randomize;
  for i := 1 to CSize do X[i]:= Random(100);
  for i := 1 to CSize do Write(X[i]:3);  Writeln;
  SortEven(X);
  for i := 1 to CSize do Write(X[i]:3);  Writeln;
  Readln;
end.
```

7.18 Датчик вращения (1)

```
function Direct(const arg: string): integer;
var i, n : integer;
begin
  Direct:=0;
  n:=0;
  { arg - строка из трёх символов-цифр }
  for i:=1 to 3 do n:=10*n + Ord(arg[i]) - Ord('0');
  case n of
    123, 231, 312: Direct:= 1;
    321, 213, 132: Direct:=-1;
  end;
end;

var S, T: string;
    i: integer;
begin
  S:= '231231231233213213213211';
  for i:= 1 to Length(S)-2 do begin
    T:= Copy(S,i,3);
    Writeln(Direct(T));
  end;
  Readln;
end.
```

7.19 Датчик вращения (2)

```
function Direct(t1,t2,t3 : integer): boolean;
var n1,n2,n3 : integer;
begin
  n1:= (t1-t2)-(t1-t3);
  n2:= (t3-t2)-(t1-t2);
  n3:= (t1-t3)-(t2-t3);
  if n1>0 then n1:=1 else n1:=-1;
  if n2>0 then n2:=1 else n2:=-1;
  if n3>0 then n3:=1 else n3:=-1;
  Direct:= n1*n2*n3 < 0;
end;

var F, R: Text;
    T1, T2, T3 : integer;
    cnt: integer;
begin
  cnt:=0;
  Assign(F,'Rotate.in');  Reset(F);
  Assign(R,'Rotate.out'); Rewrite(R);
  while not SeekEof(F) do begin
    if SeekEoln(F) then Readln(F);
    T1:=T2; T2:=T3;
    Read(F,T3);
    if cnt<2
      then Inc(cnt)
      else begin
        Writeln(R, T1:5, T2:5, T3:5, Direct(T1,T2,T3):7);
      end;
  end;
  Close(R);  Close(F);
  Write('OK'); Readln;
end.
```

12.14 Количество путей в клетку (арифметический квадрат)

```
{ Количество путей в клетку (арифметический квадрат) }

const CN=7;  CM=7;
type TMatrix = array [1..CN, 1..CM] of integer;
var M : TMatrix;
    i, j : integer;
begin
  { В первый столбец возможен лишь один путь }
  for i:=1 to CN do M[i, 1]:=1;
  { В первую строку возможен лишь один путь }
  for j:=1 to CM do M[1, j]:=1;
  { В остальные клетки пути слева и справа складываем }
  for i:=2 to CN do
    for j:=2 to CM do
      M[i,j] := M[i-1,j] + M[i,j-1];
  { Распечатка }
  Assign(Output,'Matrix_Path.txt'); Rewrite(Output);
  for i:=1 to CN do begin
    for j:=1 to CM do Write(M[i,j]:6);
    Writeln;
  end;
  Close(Output);
  Readln
end.
```

12.15 Путь с минимальной ценой

```
{ Путь с минимальной ценой }
const CInFile = 'MatrRand.txt';
      COutFile ='Matr_2.out';
const CN=7;  CM=7;   { кол-во строк и столбцов }
type TMatrix = array [1..CN, 1..CM] of integer;
var MatrIn, MatrOut : TMatrix;
{ Чтение матрицы из файла }
procedure Read_Matrix(var M : TMatrix);
var i, j: integer;
begin
  Assign(Input, CInFile); Reset(Input);
  FillChar(M, SizeOf(M), 0);
  for i:=1 to CN do begin
    for j:=1 to CM do begin
      if Eof then Break;
      Read(M[i,j]);
      if Eoln then Readln;
    end;
    if Eof then Break;
  end;
  Close(Input);
end;
{ Распечатка матрицы }
procedure Write_Matrix(const M : TMatrix);
var i, j: integer;
begin
  for i:=1 to CN do begin
    for j:=1 to CM do Write(M[i,j]:4);
    Writeln;
  end;
  Writeln;
end;
{ Формирование решения }
procedure Solve;
var i, j : integer;
begin
  MatrOut:= MatrIn; { копируем исходную матрицу }
  { формируем стоимости в первой строке }
  for j:=2 to CM do MatrOut[1,j]:= MatrOut[1,j] + MatrOut[1,j-1];
  { формируем стоимости в первом столбце }
  for i:=2 to CN do MatrOut[i,1]:= MatrOut[i,1] + MatrOut[i-1,1];
  { оптимальные стоимости в остальных клетках }
  for i:=2 to CN do
    for j:=2 to CM do
      if MatrOut[i-1,j] < MatrOut[i,j-1]
        then MatrOut[i,j]:= MatrOut[i,j] + MatrOut[i-1,j]
        else MatrOut[i,j]:= MatrOut[i,j] + MatrOut[i,j-1];
  Write_Matrix(MatrOut); {-- для отладки }
  { Оптимальный путь помечаем нулями,
    двигаясь от последней клетки к первой }
  i:= CN; j:= CM;  { для последней клетки }
  repeat
    MatrOut[i,j]:=0;
    if i=1 then Dec(j)
     else if j=1 then Dec(i)
             else
               if MatrOut[i-1,j] < MatrOut[i,j-1]
                 then Dec(i)
                 else Dec(j);
  until (i=1) and (j=1);
```

```
  MatrOut[i,j]:=0;     { для первой клетки }
end;
{ Печать оптимального пути, отмеченного нулями }
procedure Write_Result(const M : TMatrix);
var i, j: integer;
begin
  for i:=1 to CN do begin
    for j:=1 to CM do
      if M[i,j]<>0 then Write('.':2) else Write ('#':2);
    Writeln;
  end;
  Writeln;
end;
{ Главная программа }
begin
  Assign(Output, COutFile); Rewrite(Output);
  Read_Matrix(MatrIn);
  Write_Matrix(MatrIn);
  Solve;
  Write_Matrix(MatrOut);
  Write_Result(MatrOut);
  Close(Output);
  Readln;
end.
```

12.17 Числовой крест

```
{ Числовой крест }

var N, M : integer;
    i, j : integer;
begin
  Write('N= '); Readln(N);
  Write('M= '); Readln(M);
  Assign(Output,'Krest.out'); Rewrite(Output);
  for i:=1 to N do begin
    for j:=1  to M do
      Write(1+abs(2*i-1-N) div 2 + abs(2*j-1-M) div 2 : 3);
    Writeln;
  end;
  Close(Output);
  Readln;
end.
```

14.12 Числа Хэмминга

```
{
Вывести на экран первые N элементов последовательности Хэмминга
- числа, не имеющие других делителей кроме 2,3,5.
Это числа 2,3,5,6,8,9,10,12,15,16,18,20,24 ...
}

type  TNumber = Extended;     { Longint, Int64, Extended }

      PRec = ^TRec;          { Тип указатель на запись }
      TRec = record          { Запись для хранения чисел }
        mNumber : TNumber;   { Число }
        mNext : PRec;        { указатель на следующую запись }
      end;

var   Que : PRec;  { Указатель на начало очереди ("голова") }
      Count: integer = 0; { количество элементов в очереди }

{ Создаем динамическую переменную-запись и размещаем данные в ее полях }

function Create(aNumber: TNumber): PRec;
var p : PRec;
begin
  New(p);
  p^.mNumber:= aNumber;
  p^.mNext:=nil;
  Create:=p;
end;

{ Вставка числа в очередь при условии, что его там нет }

procedure PutInQue(aNumber: TNumber);
var p, q : PRec;
begin
  { Если список пуст... }
  if not Assigned(Que)
    then begin
      Que:= Create(aNumber);  { ...голова указывает на новую запись }
      Inc(Count);
    end else begin
     { Поиск места вставки начинаем с головы }
     q:= Que;
     { Двигаемся по списку, пока следующий элемент существует
       и его номер больше вставляемого }
     while Assigned(q^.mNext) and (q^.mNext^.mNumber > aNumber)
        do q:=q^.mNext;
     if q^.mNumber < aNumber then begin
        { вставка на первое место }
        Inc(Count);
        p:= Create(aNumber);
        p^.mNext:=Que;       { первый становится вторым }
        Que:=p;              { а текущий- первым }
     end else begin
        if q^.mNext^.mNumber <> aNumber then begin
          Inc(Count);
          p:= Create(aNumber);
          { вставка в середине списка, если такого числа еще нет }
          p^.mNext:=q^.mNext;  { связываем текущий со следующим }
          q^.mNext:=p;         { связываем предыдущий с текущим }
        end;
     end
```

```
    end;
end;

{ Извлечение числа из начала очереди (из конца списка) }

function GetFromQue: TNumber;
var p1, p2: PRec;
begin
  GetFromQue:=0;
  if Assigned(Que) then begin
    Dec(Count);
    { Переход к первому элементу очереди }
    p1:= Que;  p2:=p1;
    { если в очереди только один элемент, цикл не выполнится ни разу! }
    while Assigned(p1^.mNext) do begin
      p2:=p1;           { текущий }
      p1:=p1^.mNext;    { следующий }
    end;
    { теперь p1 указывает на первый элемент очереди,
      а p2 - на второй (или на тот-же самый,
      если в очереди всего один элемент)
    }
    GetFromQue:= p1^.mNumber; { извлекаем данные }
    if p1=p2                  { если в очереди был один элемент... }
      then Que:= nil          { очередь стала пустой }
      else p2^.mNext:= nil;   { а иначе "отцепляем" первый элемент }
    Dispose(p1);              { освобождаем память первого элемента }
  end;
end;

var N, L : integer;
    R: TNumber;
    F: Text;

begin
  Assign(F,'Hamming.out'); Rewrite(F);
  Que:= nil;
  PutInQue(2);  PutInQue(3);  PutInQue(5);
  Write('N= '); Readln(N);
  Writeln(F,'N=',N);
  L:=0;
  while N>0 do begin
    Dec(N);
    R:= GetFromQue;
    Write(F,R:15:0);
    Inc(L);  if (L mod 5 = 0) or (N=0) then Writeln(F);
    PutInQue(2*R); PutInQue(3*R);  PutInQue(5*R);
  end;
  Writeln(F,'Осталось в очереди: ', Count:7, ' элементов');
  Close(F);
  Write('OK'); Readln;
end.
```