1. **Изучить из Справка PascalABC.NET материал Директива parallel for**
2. **Выполнить прогамму и составить блок-схему**
3. **Написать программу на Python**

**Директива parallel for**

**Директива parallel for обеспечивает распараллеливание следующего за ней цикла.**

**{$omp parallel for}  
for var i: integer:=1 to 10 do  
  тело цикла**

**Здесь будет создано несколько потоков и разные итерации цикла будут распределены по этим потокам. Количество потоков, как правило, совпадает с количеством ядер процессора, но в некоторых случаях могут быть отличия, например, если поток ожидает ввод данных от пользователя, могут создаваться дополнительные потоки, чтобы по возможности задействовать все доступные ядра.**

**Все переменные, описанные вне параллельного цикла, будут разделяемыми, то есть, если в теле цикла есть обращение к таким переменным, все потоки будут обращаться к одной и той же ячейке памяти. Все переменные, объявленные внутри цикла, будут частными, то есть у каждого потока будет своя копия этой переменной.**

**Опция private позволяет переменные, описанные вне цикла, сделать частными. Опция записывается так:**

**{$omp parallel for private(список переменных)}**

**Список переменных – одна или несколько переменных через запятую.**

**var a,b: integer;  
{$omp parallel for private(a, b)}  
for var i: integer:=1 to 10 do  
  a := ...**

**В этом случае переменные a и b будут частными, и присваивание этим переменным в одном потоке не будет влиять на другие потоки.**

**Ограничение: счетчики распараллеливаемого цикла и вложенных циклов должны быть объявлены в заголовке цикла.**

**Не все циклы можно распараллеливать. Если на разных итерациях происходит обращение к одной и той же переменной и при этом ее значение меняется – распараллеливание такого цикла приведет к ошибкам, при разных запусках могут получаться разные результаты в зависимости от того, в каком порядке происходили обращения к этой переменной.**

**{$omp parallel for}  
for var i:=1 to 2 do  
  a[i] := a[i+1];**

**Здесь на первой итерации происходит чтение второго элемента массива, а на второй итерации – запись этого же элемента. Если первая итерация выполнится раньше второй – в первый элемент массива запишется значение из второго, а если позже – то из третьего элемента массива.**

**var a:integer;  
{$omp parallel for}  
for var i:=1 to 10 do  
begin  
  a := i;  
  ... := a;  //к этому моменту a может быть изменено другим потоком  
end;**

**Значение переменной a после этого цикла может быть любым в диапазоне от 1 до 10.**

**Наиболее эффективно распараллеливаются циклы, каждая итерация которых выполняется достаточно долго. Если тело цикла состоит из небольшого количества простых операторов, затраты на создание потоков и распределение нагрузки между ними могут превысить выигрыш от параллельного выполнения цикла.**

**Пример параллельного перемножения матриц**

**Перемножение матриц - классический пример иллюстрации параллельности. Вычисление различных элементов матрицы происходит независимо, поэтому не надо предусматривать никаких средств синхронизации.**

**uses Arrays;  
  
procedure ParallelMult(a,b,c: array [,] of real; n: integer);  
begin  
  {$omp parallel for }  
  for var i:=0 to n-1 do  
  for var j:=0 to n-1 do  
  begin  
    c[i,j]:=0;  
    for var l:=0 to n-1 do  
      c[i,j]:=c[i,j]+a[i,l]\*b[l,j];  
  end;  
end;  
  
procedure Mult(a,b,c: array [,] of real; n: integer);  
begin  
  for var i:=0 to n-1 do  
  for var j:=0 to n-1 do  
  begin  
    c[i,j]:=0;  
    for var l:=0 to n-1 do  
      c[i,j]:=c[i,j]+a[i,l]\*b[l,j];  
  end;  
end;  
  
const n = 400;  
  
begin  
  var a := Arrays.CreateRandomRealMatrix(n,n);  
  var b := Arrays.CreateRandomRealMatrix(n,n);  
  var c := new real[n,n];  
  ParallelMult(a,b,c,n);  
  writeln('Параллельное перемножение матриц: ',Milliseconds,' миллисекунд');  
  var d := Milliseconds;  
  Mult(a,b,c,n);  
  writeln('Непараллельное перемножение матриц: ',Milliseconds-d,' миллисекунд');  
end.**