Циклы

Различается и время выполнения циклов разного типа. Время выполнения цикла со счетчиком и цикла с постусловием при всех прочих равных условиях , цикл с предусловием выполняется несколько дольше (примерно на 20-30 %).

При использовании вложенных циклов следует иметь в виду, что затраты процессорного времени на обработку такой конструкции могут зависеть от порядка следования вложенных циклов. Например, вложенный цикл со счетчиком на языке Turbo Pascal:

```
for j := 1 to 100000 do
   for k := 1 to 1000 do a := 1;
```

```
for j := 1 to 1000 do
   for k := 1 to 100000 do a := 1;
```

Цикл в левой колонке выполняется примерно на 10 % дольше, чем в правой.

На первый взгляд, и в первом, и во втором случае 100 000 000 раз выполняется оператор присваивания и затраты времени на это должны быть одинаковы в обоих случаях. Но это не так. Объясняется это тем, что инициализации цикла (обработка процессором его заголовка с целью определения начального и конечного значений счётчика, а также шага приращения счётчика) требует времени. В первом случае 1 раз инициализируется внешний цикл и 100 000 раз — внутренний, то есть всего выполняется 100 001 инициализация. Во втором случае таких инициализаций оказывается всего лишь 1001.

Аналогично ведут себя вложенные циклы с предусловием и с постусловием. Можно сделать вывод, что при программировании вложенных циклов по возможности следует делать цикл с наименьшим числом повторений самым внешним, а цикл с наибольшим числом повторений — самым внутренним.

Если в циклах содержатся обращения к памяти (обычно при обработке массивов), для максимально эффективного использования кэша и механизма аппаратной предвыборки данных из памяти порядок обхода адресов памяти должен быть по возможности последовательным. Классическим примером подобной оптимизации является смена порядка следования вложенных циклов при выполнении умножения матриц.

Инвариантные фрагменты кода [править | править код]

Оптимизация инвариантных фрагментов кода тесно связана с проблемой оптимального программирования циклов. Внутри цикла могут встречаться выражения, фрагменты которых никак не зависят от управляющей переменной цикла. Их называют *инвариантными фрагментами* кода. Современные компиляторы часто определяют наличие таких фрагментов и выполняют их автоматическую оптимизацию. Такое возможно не всегда, и иногда производительность программы зависит целиком от того, как запрограммирован цикл. В качестве примера рассмотрим следующий фрагмент программы (язык Turbo Pascal):

```
for i := 1 to n do
begin
---
    for k := 1 to p do
          for m := 1 to q do
          begin
                a[k, m] := Sqrt(x * k * m - i) + Abs(u * i - x * m + k);
                b[k, m] := Sin(x * k * i) + Abs(u * i * m + k);
          end;
---
am := 0;
bm := 0;
for k := 1 to p do
          for m := 1 to q do
          begin
                am := am + a[k, m] / c[k];
                bm := bm + b[k, m] / c[k];
          end;
end;
```

Здесь инвариантными фрагментами кода являются слагаемое `Sin(x * k * i)` в первом цикле по переменной `m` и операция деления на элемент массива `c[k]` во втором цикле по `m`. Значения синуса и элемента массива не изменяются в цикле по переменной `m`, следовательно, в первом случае можно вычислить значение синуса и присвоить его вспомогательной переменной, которая будет использоваться в выражении, находящемся внутри цикла. Во втором случае можно выполнить деление после завершения цикла по `m`. Таким образом, можно существенно сократить количество трудоёмких арифметических операций.