Московский Государственный Технический Университет имени Н. Э. Баумана
Факультет «Информатика и системы управления»
Кафедра «Компьютерные системы и сети»

Иванова Г.С., Пугачев Е.К.

УТВЕРЖДАЮ
Зав. кафедрой ИУ6

д.т.н., проф. _____ Сюзев В.В.
"____"_____2013 г.

Оценка эффективности и качества программы

Методические указания по выполнению лабораторной работы

по дисциплине "Технология разработки программных систем"

Москва 2013

Введение.

В настоящее время перед разработчиками программного обеспечения стоит задача создания эффективных, технологичных и качественных программ. Данная задача усложняется тем, что четких и универсальных рекомендаций для оценки указанных свойств не существует. Однако, на данный момент накоплен некоторый опыт, который может быть использован разработчиками.

Цель работы - изучить известные критерии оценки и способы повышения эффективности и качества программных продуктов.

Продолжительность работы - 4 часа.

Краткие теоретические сведения.

Технологичность.

Технологичность определяет качество проекта и, как правило, имеет важное значение при разработке сложных программных систем. От технологичности зависят трудовые и материальные затраты на реализацию и последующую модификацию программного продукта. Факторами, которые определяют технологичность являются: *проработанность модели; уровень независимости модулей; стиль программирования и степень повторного использования кодов*. В данной лабораторной работе из-за ограниченного объема времени оценка этого качества не выполняется.

Эффективность.

Эффективность программы можно оценить отдельно по каждому ресурсу вычислительной машины. Критериями оценки являются:

- **Время ответа или выполнения.** Оценивается для программ, работающих в интерактивном режиме или в режиме реального времени.
- **Объем оперативной памяти.** Является важным для вычислительных систем, где оперативная память является дефицитным ресурсом.
- **Объем внешней памяти.** Оценивается, если внешняя память дефицитный ресурс или при интенсивной работе с данными.
- **Количество обслуживаемых терминалов** и др.

Выбор критерия оценки эффективности программы сильно зависит от выполняемых ею функций. Например, если основной функцией является поиск данных, то уменьшить время выполнения можно за счет избыточного объема оперативной памяти. Другими словами, оценка памяти будет уже второстепенной задачей.

Ниже описаны способы повышения эффективности программ.

Способы уменьшения времени выполнения.

Время выполнения программы в первую очередь зависит от используемых в ней методов. Однако, важную роль играют циклические фрагменты с большим количеством повторений. Поэтому по возможности необходимо минимизировать тело цикла.

При написании циклов рекомендуется:

- выносить из тела цикла выражения, не зависящие от параметров цикла;
- заменять «длинные» операции умножения и деления вычитанием и сдвигами;
- уменьшить количество преобразования типов в выражениях;
- исключать лишние проверки;
- исключать многократные обращения к элементам массивов по индексам (особенно многомерных, так как при вычислении адреса элемента используются операции умножения на значение индексов).

Пример . Пусть имеется цикл следующей структуры (Pascal):

```
for y:=0 to 99
  for x:=0 to 99 do
    begin
       a[320*x+y]:=S[k,l];
    end;...
```

В этом цикле операции умножения и обращения к элементу *S[k,l]* выполняются 10000 раз. Оптимизируем цикл:

```
skl:=S[k,l];          {выносим обращение к элементу массива из цикла}
for x:=0 to 99 do   {меняем циклы местами}
  begin
     i:=x shl 8+x shl 6;    {умножение заменяем на сдвиги и выносим}
```

```
    for y:=0 to 99 do
        a[i+y]:=skl;
end; …
```

(Сдвиг: Пусть x=11 в двоичной системе, тогда 11 сдвигаем на 8 -- 1100000000 и 11 сдвигаем на 6 --11000000 получаем 1111000000, т.е. 512+256+128+64+0+0+0+0+0+0 =960)

В результате вместо 10000 операций умножения будут выполняться 200 операций сдвига, а их время приблизительно сравнимо со временем выполнения операции сложения. Обращение к элементу массива *S[k,l]* будет выполнено один раз.

Способы экономии памяти.

* Следует выбирать алгоритмы обработки, *не требующие дублирования исходных данных структурных типов*. Например, метод «вставки» - не требует дополнительных массивов.

* По возможности использовать динамическую память. При необходимости выделять память, а потом освобождать.

* При передаче структурных данных в подпрограмму по значению, копии этих данных размещаются в стеке. Избежать копирования можно, если передавать данные не по значению, а как неизменяемые (описанные const). В последнем случае в стеке размещается только адрес данных, например:

Type Massiv=array[1..100] of real; function Summa(Const a:Massiv; …)…

Критерии качества программных продуктов.

Качество программных продуктов может оцениваться следующими критериями:

- правильность – функционирование в соответствии с заданием;
- универсальность – обеспечение правильной работы при любых допустимых данных и содержание средств защиты от неправильных данных;
- надежность (помехозащищенность) – обеспечение полной повторяемости результатов, т.е. обеспечение их правильности при наличии различного рода сбоев;
- проверяемость – возможность проверки получаемых результатов;
- точность результатов – обеспечение погрешности результатов не выше заданной;
- защищенность – сохранение конфиденциальности информации;
- эффективность – использование минимально возможного объема ресурсов технических средств;
- адаптируемость – возможность быстрой модификации с целью приспособления к изменяющимся условиям функционирования;
- повторная входимость – возможность повторного выполнения без перезагрузки с диска (для резидентных программ);
- реентерабельность – возможность «параллельного» использования несколькими процессами.

Ниже рассмотрим пример программы, в которой попытаемся сделать улучшения.

Задача. Найти корни функции методом "хорд" с точностью 0.000001

для $f(x) = x^3/100 - 5$.

```
Program hrd; {исходный текст программы}
  var x,p,z,b,eps:real;
 function f(x:real):real;  {исследуемая функция}
    begin f:=x*x*x/100-5 end;
 Begin
x:=-10;  b:=10;  eps:=0.000001;{исследуемый интервал и  погрешность}
  Repeat {основной цикл}
         p:=f(x);  {сохранение предыдущего значения}
         x:=x-f(x)*(b-x)/(f(b)-f(x)); {определение корня}
  Until abs(f(x)-p)<=eps;
writeln(' x = ',x:10:5,'  y = ',f(x):10:5); {вывод результатов}
End.
```

В результате анализа данной программы получаем следующие выводы:

1. В теле цикла многократно (5 раз) повторяются одни и те же действия, а именно, вычисление значения функции. Если сократить количество вычислений значения функции, то можно уменьшить время выполнения программы, а следовательно повысить эффективность.
2. Не корректно задан формат вывода вычисленного значения, т.к. он не учитывает порядок погрешности.

Экспериментально подтвердить сделанные выводы можно следующим образом:

1. Ввести в программу контрольные точки, например, фиксировать время начала выполнения цикла и время окончания с помощью процедуры Gettime, а затем определить время выполнения.
2. Чтобы время выполнения было легче зарегистрировать или увидеть, можно воспользоваться, образно говоря, «увеличительной линзой». Другими словами, можно ввести дополнительный цикл, телом которого является исследуемый фрагмент. В результате с помощью процедуры Gettime можно определять полученный эффект.

*Примечание. При внесении в программу контрольных точек, необходимо учитывать их влияние.

В рассматриваемом примере тело цикла можно модифицировать так:

```
Repeat   p:=z;  x:=x-p*(b-x)/(f(b)-p);  z:=f(x);  Until abs(z-p)<=eps;
```

Ниже приведены две программы (программа на языке Turbo Pascal и программа консольного приложения Delphi) , в которых отражены контрольные точки и исследуемые фрагменты.

```
Program hrd; { программа на языке Turbo Pascal}
  Uses dos,crt;
  var x,p,z,b,eps:real;   {z - дополнительная переменная}
     i,t1,t2,t3,t4,t11,t22,t33,t44:word; {вспомогательные переменные}
 function f(x:real):real;
       begin f:=x*x*x/100-5 end;
 Begin
      clrscr;
      Gettime(t1,t2,t3,t4); {фиксация времени начала}

            x:=-10; b:=10; eps:=0.000001; z:=f(x);
{for i:=1 to 50000 do
 { Исходный цикл}
 Repeat     p:=f(x);  x:=x-f(x)*(b-x)/(f(b)-f(x));   Until abs(f(x)-p)<=eps;
}

 for i:=1 to 50000 do
 {Модифицированный цикл}
Repeat    p:=z;    x:=x-p*(b-x)/(f(b)-p);    z:=f(x);    Until abs(z-p)<=eps;

Gettime(t11,t22,t33,t44); {фиксация  времени окончания}
writeln(' x = ',x:10:5); {результаты }
writeln(t1,' ',t2,' ',t3,' ',t4); {время начала выполнения}
writeln(t11,' ',t22,' ',t33,' ',t44);  { время окончания}
End.
```

В результате получаем, что время выполнения исходного цикла на языке Turbo Pascal, например, на Pentium 120 при imax=50000 равно 77 ms, а время выполнения модифицированного цикла 44 ms, т.е. повысили эффективность в 1.75 раза.

```
Program hrd;  {программа консольного приложения  Delphi}
{$APPTYPE CONSOLE}
uses
  SysUtils;
var x,p,z,b,eps:real;  // z - дополнительная переменная
      P1,P2: TDateTime;  // вспомогательные переменные
      H1,M1,S1,Ms1,H2,M2,S2,Ms2:word;
      i:longint;
function f(x:real):real;
        begin f:=x*x*x/100-5 end;
  Begin
 P1:= Now; // фиксация времени начала
 DecodeTime(P1, H1, M1, S1, Ms1);

x:=-10; b:=10; eps:=0.000001; z:=f(x);

{ for i:=1 to 50000000 do
  Repeat // Исходный цикл
    p:=f(x); x:=x-f(x)*(b-x)/(f(b)-f(x));
```

```
 Until abs(f(x)-p)<=eps;
 }

for i:=1 to 50000000 do
Repeat  // Модифицированный цикл
   p:=z;    x:=x-p*(b-x)/(f(b)-p);    z:=f(x);
Until abs(z-p)<=eps;

 P2:= Now; // фиксация  времени окончания
DecodeTime(P2, H2, M2, S2, Ms2);

writeln(' x = ',x:10:5);// результаты
Writeln(H1,':',M1,':',S1,':',Ms1); //  время начала выполнения
Writeln(H2,':',M2,':',S2,':',Ms2); //время окончания
readln;
End.
```

В результате получаем, что время выполнения исходного цикла программы консольного приложения Delphi, например, на Intel(R) Core™ i3-2100 при imax=50000000 равно 6031 ms, а время выполнения модифицированного цикла 2938 ms, т.е. повысили эффективность в 2 раза.

Порядок выполнения работы:

1. Ознакомиться с теоретическими сведениями.
2. Исследовать заданный пример программы с целью определения ее эффективности и качества.
3. Определить основные критерии оценки и количественные характеристики для заданной программы.
4. Предложить варианты повышения эффективности и улучшения качества для заданного примера программы.
5. Составить отчет. В отчете должны быть приведены: задача и исходная программа, улучшенный вариант программы и оценочные таблицы (см. табл.1 и табл.2).

Таблица 1- Оценка эффективности.

	Исходная программа		**Улучшенная программа**	
Критерий оценки	**Недостатки**	**Колич-я оценка**	**Улучшения**	**Колич-я оценка**
Время выполнения				
Оперативная память				
Внешняя память				
. . . . .				

Таблица 2 - Оценка качества.

	Правильность	**Универсальность**	**Проверяемость**	**Точность результатов**
Недостатки				
Оценка				

Варианты заданий.

1. Написать программу, которая строит отсортированный список вещественных чисел в динамической памяти. Реализовать сортировку по убыванию (программа - v1.dpr).
2. Написать программу, которая позволяет строить графики функций y=| cos(x)*x| и y=| sin(x)| . Обеспечить возможность вывода на экране графиков одновременно для двух функций с наложением фиксированных осей координат. (в папке V2 программа - v2. dpr).
3. Написать программу, в которой создается динамический список неповторяющихся целых чисел в диапазоне от -50 до 50. Обеспечить прямой и обратный вывод элементов списка. Программа должна посчитать сумму: 1+n,2+n-1, …i+n-i+1, …n/2+n/2+1. (программа - v3. dpr).

4. Написать программу, в которой реализуется метод «хорд» для нахождения корней функции с точностью 0.001 (программа - v4. dpr).
5. Вычислить значение интеграла с точностью 0.0001 с помощью метода прямоугольников (программа - v5. dpr).
6. Вычислить значение интеграла с точностью 0.001 используя метод Симпсона, т.е. $\int_a^b f(x)\,dx \approx \frac{h}{3}(f_0 + f_n + 4 \bullet (f_1 + f_3 + \ldots + f_{n-1}) + 2 \bullet (f_2 + f_4 + \ldots + f_{n-2}))$ (программа - v6. dpr).
7. Программа должна генерировать массив вещественных чисел в диапазоне от –10 до 10 и определять все минимальные положительные элементы, если их много (программа - v7. dpr).
8. Создать программу, в которой создается массив целых чисел в диапазоне от -20 до 5 и сортируется по убыванию (программа - v8. dpr).
9. Создать программу, в которой генерируется массив целых чисел в диапазоне от –20 до –10, а затем сортируется по убыванию с исключением повторяющихся элементов (программа - v9. dpr).
10. Программа должна сортировать четные строки массива вещественных чисел (программа - v10. dpr).
11. Написать программу вычисления суммы ряда с заданной точностью. Например, дан ряд: $S = \frac{1}{4} - \frac{1}{16} + \frac{1}{96} - \ldots (-1)^{n+1} \frac{1}{4^n \cdot n!}$, ряд сходится и имеет множитель $m = -\frac{1}{4 \cdot n}$. Определить сумму ряда с точностью 0.0001 (программа - v11. dpr).
12. Создать массив целых чисел. Определить все максимальные элементы в каждом столбце (программа - v12. dpr).
13. Реализовать итерационный метод Зейделя для решения систем линейных уравнений вида $A \cdot x = b$ с точность 0.01, где

$A = \begin{bmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ a_{21} & a_{22} & \ldots & a_{2n} \\ . & . & . & . \\ a_{n1} & a_{n2} & \ldots & a_{nn} \end{bmatrix}$ - матрица коэффициентов при неизвестных переменных;

$b = \begin{bmatrix} b_1 \\ b_2 \\ \ldots \\ b_n \end{bmatrix}$ - вектор столбец свободных членов; $x = \begin{bmatrix} x_1 \\ x_2 \\ \ldots \\ x_n \end{bmatrix}$ - вектор столбец неизвестных.

В данном методе последовательно уточняются компоненты решения. Систему приводят к виду:

$$X_1' = (b_1 - a_{11} \cdot X_1^0 - a_{12} \cdot X_2^0 - \ldots - a_{1n} \cdot X_n^0) / a_{11} + X_1^0$$
$$X_2' = (b_2 - a_{12} \cdot X_1' - a_{22} \cdot X_2^0 - \ldots - a_{2n} \cdot X_n^0) / a_{22} + X_2^0$$
. . . . . .
$$X_n' = (b_n - a_{n1} \cdot X_1' - a_{12} \cdot X_2' - \ldots - a_{nn} \cdot X_n^0) / a_{nn} + X_n^0$$

Выражение в скобках в идеале должно быть равно нулю. На диагональные элементы делим, т.к. определяем соответствующие неизвестные. В результате, например, вновь уточняемое X_1' должно быть приблизительно равно предыдущему X_1^0 (как следствие большого количества итераций).

По шагам:

а) $X_1' = b_1$ (остальные = 0 на данном шаге);

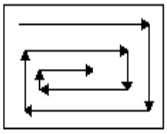
$X_2' = b_2 - a_{21} \cdot X_1'$ (остальные =0, а X_1' подставляем из 1-го уравнения);

и т.д. до X_n' .

б) На второй итерации все повторяется.

и т.д. до тех пор , пока не достигнем заданной точности. (программа - v13. dpr)

14. Написать программу, которая позволяет заполнять двумерный массив целыми двузначными числами по спирали следующим образом:



(программа - v14. dpr)

Список литературы

1. Костин А.Е., Шаньгин В.Ф. Организация и обработка структур данных в вычислительных системах. -М.: Высшая школа, 1987.- 248с.
2. Иванова Г.С. Технология программирования. -М.: Кнорус, 2011.- 333с.
3. Бахвалов Н.С. Численные методы. -М: Наука, 1975.-632с.