Требования к написанию программ практикума

Кафедра системного анализа

6 сентября 2016 г.

Аннотация

Этот текст предназначен для студентов третьего курса кафедры системного анализа и содержит ряд указаний, советов и требований по созданию и оформлению исходного кода программ в среде `Matlab`.

Введение

Обучение на кафедре системного анализа подразумевает написание студентом ряда вычислительных практикумов (до двух-трёх за семестр). Большинство практикумов выполняются в среде `Matlab`, основы которой излагаются в пятом семестре. Данный документ содержит ряд правил и указаний, следование которым должно помочь студентам в написании хорошо структурированного, легко понимаемого и читаемого (и, следовательно, легче проверяемого преподавателем) кода. Документ разбит на четыре раздела:

1. *Внешний вид кода.* В этом разделе описываются требования, предъявляемые к внешнему виду кода: форматированию и именованию переменных.
2. *Структура программы.* Этот раздел описывает структуру и организацию программ — в частности, разбиение исходного кода на файлы.
3. *Методы программирования.* Этот раздел содержит требования, предъявляемые к используемым приёмам создания алгоритмов, и требования, которым должно удовлетворять поведение программы на этапе выполнения.
4. *Оформление результатов.* В этом разделе содержатся требования, предъявляемые к результатам, выдаваемым в результате работы программы — например, к графикам.

1 Внешний вид кода

1. Запрещается размещение более одного оператора (объявления переменной, вызова функции, ... — далее, *команды*) на одной строке исходного текста программы.
2. Все команды, входящие в цикл или функцию (далее — в *блок*), должны начинаться с дополнительного отступа относительно команды, объявляющей начало блока. Например,

```
function result = DoSomethingUsefull(list)
        N = size(list, 1);
        result = zeros(N, 1);
        for k = 1:N
                if (mod(k, 2) == 0)      % k is even
                        result(k) = FunctionA(list(k, :));
                else                     % k is odd
```

```
                                result(k) = FunctionB(list(k, :));
                        end
                end
        end
```

   В приведенном примере в качестве отступа использован символ табуляции. Допускается использование в этом качестве фиксированного числа пробелов. Отступы должны быть оформлены одинаково во всём тексте программы.
3. Категорически не стоит использовать более одной пустой строки подряд.
4. Категорически не стоит использовать более одного пробельного символа подряд (Это не распространяется на пробелы, используемые для оформления отступов).
5. Пробельные символы обязаны стоять:
   (a) Между операндами и символами бинарных операций (`a + b * (c - d)`, но не `a+b*(c-d)`), в том числе — операции присваивания;
   (b) После запятых, в том числе — в перечислениях аргументов функций.
6. Пробельный символ не должен стоять перед запятой или точкой с запятой.
7. Имена переменных должны быть содержательными. Запрещается названия переменных вроде `aa, sdasd, tmpTest_old, myMatrix, iaChiselko` и т.д. Строчные однобуквенные переменные могут быть использованы только для счетчиков, заглавные — только для матриц. Исключение составляют однобуквенные переменные, имеющие общепринятую математическую семантику, например:

```
x = t - sin(t); % x and y are planar coordinates
y = 1 - cos(t); % t is time
f = @(x,y) (x + y) / 2;
plot3(x, y, f(x,y));
```

8. При использовании сложных выражений стоит всегда в явном виде расставлять скобки.

2 Структура программы

Каждое задание практикума, состоящее в реализации некоторого (численного) метода с последующим этого метода исследованием, должно быть оформлено следующим образом (не распространяется на задания, выполняемые в виде скриптов):

1. Сам метод должен быть оформлен в виде отдельной функции, возвращающей результат его применения.
2. Ввод данных, вычисления и отрисовка их результатов должны быть разнесены, т.е. выполняться тремя различными функциями (в скрипте — в трёх разных блоках).
3. После заголовка функции должен следовать краткий комментарий, описывающий семантику входных и выходных аргументов функции.
4. Каждая используемая функция должна быть размещена в отдельном файле. Исключение составляет случай, когда все функции, кроме заглавной (название которой совпадает с именем файла) не используются нигде, кроме как в заглавной функции (например, функции для передачи в команды вроде `ode45`). Кроме того, допускается размещение всех `callback`-функций графического интерфейса в одном файле. В файлах, содержащих несколько функций (кроме файлов, генерируемых Guide), каждая функция должна завершаться командой `end`.
5. В случае выполнения задания в виде вызова функции из интерфейса командной строки к самой функции должен прилагаться скрипт, разбитый на `cell`-блоки[1],

[1] Скрипт разбивается на блоки символами `%%`. Навигация между блоками осуществляется нажатием клавиш `Ctrl` + стрелки, запуск блока — `Ctrl` + `Enter`.

каждый из которых отвечает за выполнение отдельного пункта задания или демонстрацию особенности работы функции на определённом наборе параметров. Запуск каждого из блоков не должен требовать каких-либо изменений (например, комментирования) в других блоках.

6. Написание графических интерфейсов должно быть организовано таким образом, чтобы `callback`-функции и функции, занимающиеся вычислением (или отрисовкой) результатов были разнесены отдельно.

3 Методы программирования

1. Если какое-либо числовое значение встречается в коде более одного раза, то оно должно быть заменено на обращение к переменной. Исключение составляют константы 0 и 1. Например:

```
n = 10;
m = 2;
%   Right          %  Wrong
A = zeros(n, n); %  A = zeros(10, 10);
B = zeros(n, m); %  B = zeros(10, 2);
x = zeros(n, 1); %  x = zeros(10, 1);
```

2. Любая переменная на протяжении всего своего периода жизни должна хранить значения только одного типа. Не допускается, например, чтобы сначала в переменной хранилась строка, а потом — матрица.
3. Запрещается использование глобальных переменных для организации взаимодействия между функциями.
4. Если некоторая операция, занимающая более одной-двух команд, выполняется более одного раза, то она должна быть выделена в отдельную функцию.
5. Вывод в консоль должен осуществляться с помощью команды `disp`, а не отсутствием точки с запятой.
6. Программа не должна выводить никакой отладочной информации в процессе своей работы. Допускается вывод коротких сообщений о выполнении отдельных частей программы, занимающих значительное время. Например:

```
disp('Loading␣data...')
[origImage, data] = LoadData('input.mat');
disp('Bulding␣statistics...')
stat = BuildStatistics(data, origImage);
disp('Correcting␣image...')
newImage = CorrectImage(origImage, stat);
```

7. Память под любые данные, инициализуруемые в цикле, должна быть выделена до начала цикла. Например[2]:

```
optimalTrajectories = zeros(timeGrid, numTries);
for i = 1 : numTries
        optimalTrajectories(:, i) =
                = GetOptimalTrajectory(A, B, x0, l(:, i));
end
```

[2]В простых ситуациях, аналогичных приведенному примеру, `Matlab` подсветит соответствующую строчку в текстовом редакторе предупреждением.

4 Оформление результатов

1. На всех графиках должны быть подписаны все оси. Приветствуется наличие заголовка над графиком. Если на графике, изображающем зависимость одной величины от другой, изображено несколько линий, то на нём должна присутствовать легенда.
2. В заданиях, в которых требуется установить зависимость какой-либо характеристики (времени работы, точности, ...) изучаемого метода в зависимости от размера входных данных, значение характеристики на данных фиксированного размера должно устанавливаться через усреднение по нескольким запускам, число которых долно быть вынесено в параметры скрипта. Допускается подготовка графиков для больших размерностей и большого числа усреднений заранее (дома). В таком случае график должен быть представлен в виде сохранённой фигуры, а сами данные, по которым он был получен, представлены в виде `.mat` файла.

Список литературы

[1] P. Gagarinov. *MATLAB Programming Style Requirements*. 2012.

[2] В.Г. Баула, Д.К. Мещеряков. *Учебное пособие по написанию программ для выполнения работ практикума на ЭВМ*. 2007.