МЕТОДИКА РЕШЕНИЯ ВЫЧИСЛИТЕЛЬНЫХ ЗАДАЧ НА ЯЗЫКЕ ПРОГРАММИРОВАНИЯ FORTRAN

Газизова Алия Альбертовна, магистр
Гайнутдинова Татьяна Юрьевна, к.т.н., доцент
Казанский федеральный университет,
alijayandexmatur@mail.ru, tgainut@mail.ru

Аннотация. В данном исследовании рассматривается методика решения вычислительных задач на языке Fortran. Основное внимание уделяется научным расчетам по математике и описываются их алгоритмы.

Ключевые слова: алгоритм, вычислительные задачи, программирование.

SOLVING TECHNIQUE OF COMPUTING TASKS IN A PROGRAMMING LANGUAGE FORTRAN

Gazizova Aliya, master
Gainutdinova Tatayna, associate professor
Kazan Federal University
alijayandexmatur@mail.ru, tgainut@mail.ru

Abstract: Solving technique of computing tasks in a programming language Fortran is considered in this study. Main attention is focused on the scientific calculations in mathematics and description of their algorithms.

Key words: algorithm, computing task, programming

ВВЕДЕНИЕ

При современных темпах научно-технического прогресса, задачи, которые решает человек в своей образовательной, научно-исследовательской, профессиональной деятельности, делятся на две категории: вычислительные и функциональные. Функциональные задачи требуют решения при реализации функций управления, проектирования. Цель вычислительных задач – расчет параметров, характеристик, обработка данных. При их решении основные усилия направлены на то, чтобы найти ее решение.

FORTRAN - один из первых языков, который широко используется до настоящего времени для инженерных и научных вычислений. Он занимает лидирующее положение, ориентированных на решение научных и численных расчетов. Он является жёстко стандартизированным языком и поэтому легко переносится на различные платформы. Новые стандарты языка в значительной мере сохраняют преемственность с более старыми, что позволяет использовать коды ранее написанных программ и модифицировать их.

МАТЕМАТИЧЕСКОЕ МОДЕЛИРОВАНИЕ

С точки зрения целей моделирования можно выделить следующие типы математических моделей: *описательные, оптимизационные, игровые, имитационные*.

Описательные математические модели используются для описания объекта моделирования с помощью математических формул. Например, в решении экономических задач широко используются матричные математические модели, для исследования которых применяются методы линейной алгебры.

Оптимизационные модели. Возможны случаи, когда, моделируя те или иные процессы, можно воздействовать на них, пытаясь добиться какой-то цели. В этом случае в модель входит один или несколько параметров, значения которых можно изменять.

Игровые модели предназначены для обоснования решений в условиях неопределенности и связанного с этим риска. Рассматриваются ситуации, в которых сталкиваются противоборствующие стороны, каждая из которых преследует свою цель. Достижение цели каждой из сторон зависит от того, какие действия предпримет противник. Такие ситуации называются конфликтными.

Имитационные модели. Имитационное моделирование – это метод исследования, при котором изучаемый объект заменяется компьютерной математической моделью, с достаточной

точностью описывающей реальный объект. С полученной моделью проводятся эксперименты с целью получения информации об объекте [1, 5, 6].

МЕТОДИКА РЕШЕНИЯ ВЫЧИСЛИТЕЛЬНЫХ ЗАДАЧ

Составление алгоритмов решения задач - это работа творческая. Нет универсального способа, позволяющего без особого труда составлять любые алгоритмы. При решении вычислительных задач можно воспользоваться определенной схемой. Есть раздел математики, называемый вычислительной математикой, в котором представлен опыт решения разных вычислительных задач. Например, методы отыскания корней нелинейных уравнений, вычисления определенных интегралов, численного интегрирования дифференциальных уравнений, методы сортировки данных и многие другие.

В большинстве случаев та или иная задача может быть решена несколькими численными методами. Выбор конкретного численного метода решения задачи обычно производится по следующим критериям:

- обеспечение оптимального времени решения задачи;
- обеспечение оптимального использования имеющихся ресурсов (памяти);
- обеспечение требуемой точности вычислений;
- минимальные стоимостные затраты;
- возможность использования стандартных подпрограмм.

При дальнейшей постановке задачи на персональном компьютере отыскивается наиболее рациональный способ решения задачи.

Под вычислительной задачей будем понимать одну из трех задач, которые возникают при анализе математических моделей: прямую задачу, обратную задачу или задачу идентификации. Слово "вычислительная" подчеркивает, что основные усилия будут направлены на то, чтобы найти (вычислить) ее решение [2-4].

Первая задача

В основу методики заложен метод разбиения на участки.

Алгоритм расчета

Разбиваем график на участки (следует выбрать шаг табуляции каждого участка). Для прямолинейных участков шаг выбирается из расчета 2-5 точек на участок. На криволинейном участке с целью улучшения изображения в AGrapher шаг выбирается более мелкий, до 50 точек. Для прямой, параллельной оси X выводятся две точки (без цикла). (Приложение Advanced GRAPHER распространяется бесплатно, адрес сайта http://www.alentum.com/agrapher/.)

Для табуляции каждого участка использовать оператор цикла DO

```
x=xn,xk,step
...
enddo
```

Здесь x – переменная цикла; xn, xk – начальное и конечное значения x, step - шаг изменения x.

Количество повторений цикла рассчитывается по формуле:

$$M = Max\left(0, Int\left(\frac{xk - xn + step}{step}\right)\right)$$

или

$$M = Max\left(0, Int\left(\frac{xk - xn}{step} + 1\right)\right)$$

Из-за применения в расчетах функции Int (преобразование в целый тип) при вещественных значениях xn, xk, step можно потерять конечное значение переменной цикла. Возможные варианты предотвращения этой ошибки:

a) при вещественных значениях xn, xk или step конечное значение в операторе цикла всегда записывать в виде xk + step/2.0;

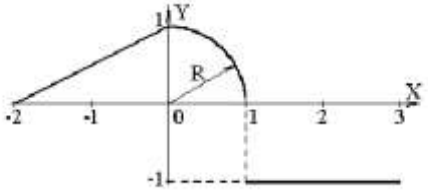
b) построить цикл не по вещественной переменной x, а по целой I (do I=1,M). Здесь M – количество повторений цикла, рассчитанное вручную при написании программы. При этом в цикле придется пересчитывать целочисленную переменную I в вещественную переменную x оператором x=xn+step*(I-1). В этом случае ошибка вычислений не накапливается;

c) выражать угол alpha в целых градусах, а не в радианах, и при этом использовать функции sind, cosd;

d) координаты последней точки участка вывести после цикла.

Пример

Функция задана графически на интервале [-2, 3]



Уравнения участков графика

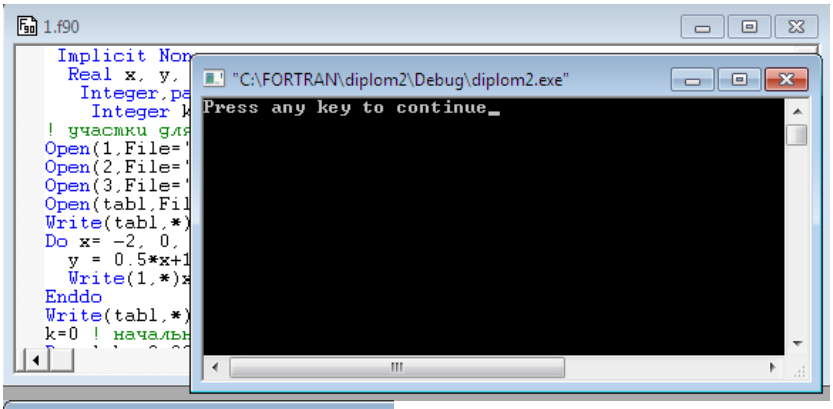
№	*Границы участка*	*Уравнение*	*Шаг цикла*	*Количество точек*	*Примечание*
1	[-2,0]	y =x/2 +1	1	3	Прямая пересекает оси в точках (-2,0) и (0,1). Уравнение прямой в отрезках.
2	[0, 1] α∈[0, 90°]	x = cosα y = sin α	3	31	Параметрические уравнения окружности радиуса R=1 с центром в начале координат.
3	[1,3]	y = -1	нет	2	Горизонтальная прямая (параллельна оси X)

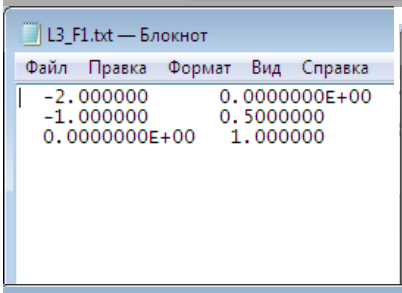
Программа

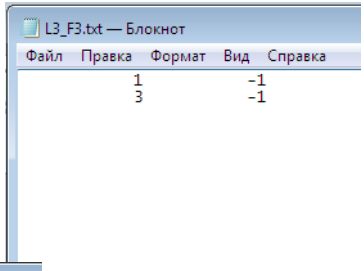
```
Program Function_graph! Табуляция функции
! студент (фамилия, имя)   группа №   работа №   вариант №
Implicit None
  Real x, y, alpha
Integer,parameter:: tabl=6 ! имени tabl присвоен номер устройства 6
Integerk !номер точки криволинейного участка
! участки для AGrapher
Open (1,File='L3_F1.txt')! 1 – координаты точек прямой
Open(2,File='L3_F2.txt')! 2 – координаты точек дуги окружности
Open(3,File='L3_F3.txt') ! 3 – горизонталь (координаты граничных точек)
Open(tabl,File='Out.txt')! контрольный текстовый файл
Write(tabl,*)'прямая'
Do x= -2, 0, 1
  y = 0.5*x+1
  Write (1,*)x,y;  Write (tabl,*)'(x,y)=',x,y
Enddo
Write(tabl,*)'дуга окружности'
k=0 ! начальная установка счетчика точек на дуге окружности
Do alpha=0,90,3 ! аргумент функций Sin d, Cos d -  в градусах, шаг цикла - целый
x=Cosd(alpha)
  y=Sind(alpha)
Write(2,*)x,y
k=k+1
  ! в tabl печатается каждая 5-я точка, Mod(k,5) – остаток от деления k на 5
if (Mod(k,5)==1) Write(tabl,*)'(x,y)=', x,y
Enddo
Write(tabl,*)'Горизонталь - две точки без цикла'
Write(tabl,*) '(1,-1) ==> (3,-1,)'
```

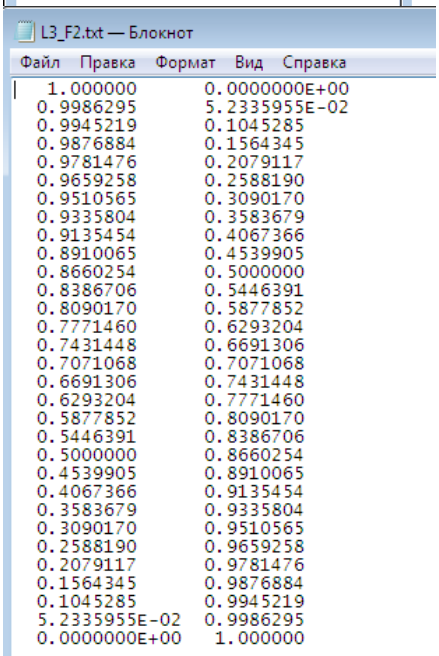
Write(3,*) 1,-1; Write(3,*) 3,-1
EndProgramFunction_graph

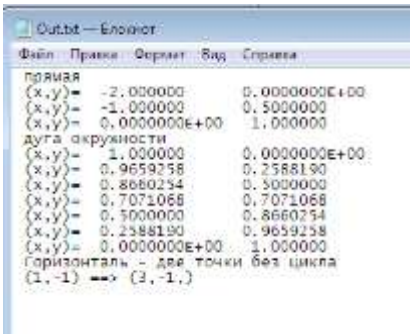
Анализ результатов исследования

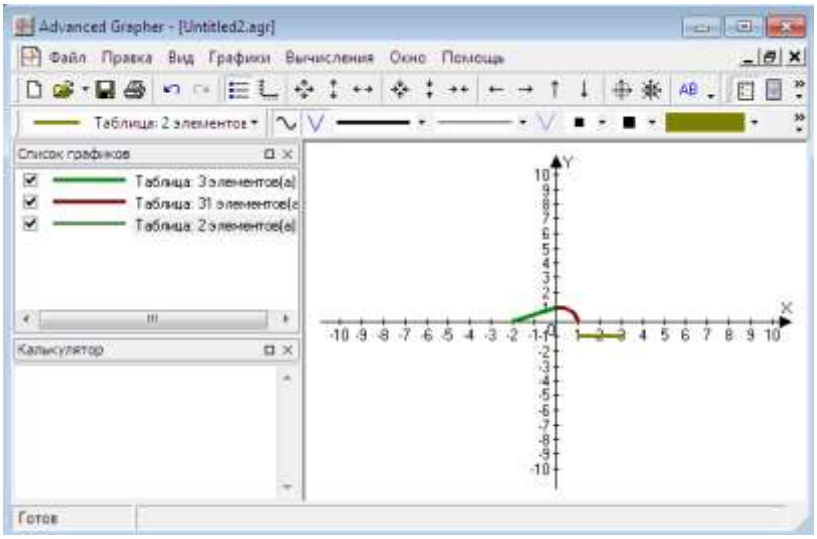












Вторая задача

В основу методики заложен метод простого обмена.

В создаваемом проекте могут использоваться:

1.встроенные процедуры;

2.подключаемые процедуры;

3.создаваемые при разработке проекта процедуры.

Встроенные процедуры входят в состав Фортрана и автоматически включаются в исполняемый код при обращении к ним в тексте программы (например, процедуры *sin, mod, tan* и т.д.).

Подключаемые процедуры находятся в ранее созданных библиотеках. Для их использования в программной единице следует подключить к ней модуль, содержащий используемые при работе с процедурами глобальные данные и интерфейсы. Такое подключение выполняется оператором USE. Пользователь может создать собственные прикладные библиотеки и использовать хранимые в них процедуры и модули в любом из своих проектов. Для доступа к содержащим объектный код библиотекам пользователю следует указать ее имя в опции компилятора /link.

Любая программа имеет одну головную программу, которая находится в начале программы. В общем виде головную программу можно представить:

[**PROGRAM** *имя программы*]
[*операторы описания*]
[*исполняемые операторы*]
END[PROGRAM[*имя программы*]]

В Фортране могут быть определены два типа процедур: ***подпрограммы*** *и* ***функции***. *Функция* отличается от *подпрограммы* тем, что вызывается непосредственно из выражения и возвращает результат, который затем используется в этом выражении. Процедуру следует оформлять в виде функции, если ее результат можно записать в одну переменную, в противном случае следует применять *подпрограмму.*

Имя процедуры является *глобальным*, т.е. к процедуре можно обратиться из головной программы и любой другой процедуры.

Процедура может *компилироваться отдельно* от использующих ее программных единиц.

Структура процедуры-подпрограммы :

SUBROUTINE *имя подпрограммы*[([список формальных параметров])]
[операторы описания]
[исполняемые операторы]
END[SUBROUTINE[*имя подпрограммы*]]

Структура процедуры-функции:

[type]**FUNCTION***имя функции*([список формальных параметров]) &
[**RESULT**(*имя результата*)]
[операторы описания]
[исполняемые операторы]
END[FUNCTION[*имя функции*]]

Фактические параметры	*Формальные параметры*
Простая переменная	Простая переменная
Строка	Строка
Подстрока	Строка
Массив, сечение массива или элемент Массива	Массив или простая переменная
Процедура	Процедура
Выражение, константа	Переменная

Пример

Создать процедуру обмена содержимого двух массивов.

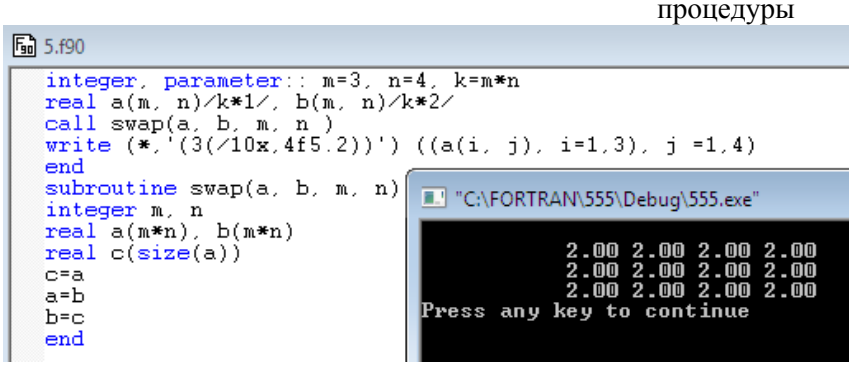
Программа:

```
integer, parameter:: m=3, n=4, k=m*n
real a(m, n)/k*1/, b(m, n)/k*2/
call swap(a, b, m, n )
write (*,'(3(/10x,4f5.2))') ((a(i, j), i=1,3), j =1,4)
end
subroutine swap(a, b, m, n)
integer m,n
real a(m*n), b(m*n)
real c(size(a))
c=a
a=b
b=c
end
```

Между вызывающей программой и процедурой *swap* устанавливается интерфейс во время вызова процедуры по фактическим массивам *a, b* и фактическим переменным*m, n*.

a,b– массивы заданной формы, нижняя граница их измерений равна 1, верхняя граница =*m*n*, где*m, n*– передаваемые в процедуру значения из вызывающей программы; формальные массивы *a,b* и фактические массивы *a,b* имеют разные формы;

c – автоматический массив; автоматический массив – это локальный массив в процедуре, размер которого меняется при разных вызовах процедуры



ЗАКЛЮЧЕНИЕ

В статье рассматривается алгоритмы решения вычислительных задач программирования на языке Fortran, как одного из основных составляющих формирования и развития профессиональных компетенций – способностей действовать на основе имеющихся умений, знаний и практического опыта в определенной области профессиональной деятельности.

Решение вычислительных задач способствует формированию практических навыков составления программ с использованием основных конструкций языка программирования Fortran.

Список литературы:

1. Горелик, А.М. Фортран сегодня и завтра / А.М. Горелик, В.Л. Ушкова.–М.: Наука. – 1990. – 340с.
2. Abrams, S.L, Efficientand Reliable Methods for Rounded-Interval Arithimetic / S.L. Abrams, W. Chot, C.-Y.Hu, T. Maekawa, N. M. Patrikalakis, E. C. Sherbrooke, and X. Ye.Computer. –Aided Design 30, no. 8. – 1998. –P.657-665.
3. Akin, Ed. Object-Oriented Programming via Fortran 90/95 / Ed. Akin. - New York: Cambridge University Press, 2003. – 368p.

4. Beckand, K. C. Andres. Extreme Programming Explained: Embrace Change. / K. Beckand, C. Andres. – Boston: Addison-Wesley Professional. – 2004.– 658p.

5. Beizer, B. Software Testing Techniques / B. Beizer. – Boston: Embrace Change. Boston: Addison-Wesley Professional. – 2004. – 452p.

6. Reimann, R. About Face 3: The Essentials of Interaction Design. Indianopolis, / R. Reimann, A. Cooper, D. Cronin. –Indiana: Wiley Publishing Inc., 2007. – 468s.

Интернет-ресурсы

1. Что такое «Fortran»? Язык программирования «Фортран»» –URL: ftp://ftp.nag.co.uk/sc22wg5/N1801-N1850/N1824.pdf
2. Интернет - ресурс: «Фортран, язык программирования» - URL: http://valgrind.org
3. CMake, accessed March 2012, http://www.cmake.org.
4. Coding Standard, accessed March 2012, http://c2.com/cgi/wiki?/CondingStandart.
5. Test-Driven Development. – Wikipedia, last modified March 2012, http;//en.wikipedia.org/wiki/Test-Driven_development, 2010.