МИР программирования

М.Ф. Гарифуллин

# Обработка текстовой и графической информации

ТЕХНОСФЕРА
Москва
2019

**УДК 519.682**
**ББК 32.97**
**Г20**

**Г20 Гарифуллин М.Ф.**
**Обработка текстовой и графической информации**
**М.: ТЕХНОСФЕРА, 2019. – 174 с. Табл., илл. 21. Библиогр.: 28 назв.**
**ISBN 978-5-94836-540-4**

Рассмотрены способы обработки текстовой и графической информации. Уделено внимание вопросам сортировки текстовых данных, построения и сканирования графиков, поддержки диалогового режима работы, создания и систематизации графических файлов. Приведены примеры текстов программ на языке FORTRAN с подробными комментариями.

Издание предназначено для научных работников и инженеров, занятых расчетами и обработкой экспериментальных данных, а также преподавателей, студентов и аспирантов технических вузов.

**УДК 519.682**
**ББК 32.97**



**ISBN 978-5-94836-540-4**

# Содержание

# Введение

Развитие вычислительной техники и программного обеспечения привело к широкому распространению коммерческих программ. Количество потребителей многократно выросло, а количество независимых производителей сократилось. Многие перестали развивать собственные разработки и превратились в обычных пользователей, тратя время на поиски готовых программ и средств на их закупку, хотя в большинстве случаев поставленные задачи могли бы быть решены с меньшими затратами собственными силами. В этой связи в книге приведены примеры, показывающие, что простыми средствами FORTRANa можно самостоятельно получить требуемое решение, не прибегая к услугам сторонних производителей. Аналогичным образом могут быть написаны программы и на других языках программирования.

Язык FORTRAN (FORmula TRANslation) многие годы используется при создании вычислительных программ, в том числе комплексов программ, предназначенных для решения научных и технических задач (прочности, аэрогидродинамики, аэроупругости и др.). Эти программы развиваются, поддерживаются, регулярно обновляются и широко используются в инженерной практике.

FORTRAN имеет богатую историю. Он стал первым языком программирования высокого уровня. Его первая версия была разработана еще в середине прошлого века, когда мир электронно-вычислительных машин (ЭВМ) ограничивался единичными экземплярами ламповых цифровых вычислительных машин. Существовал весьма узкий круг специалистов, способных писать программы к этим машинам и проводить расчеты. Создание FORTRANa стало большим шагом вперед с точки зрения программирования и внедрения цифровых ЭВМ в инженерную практику, так как этот язык существенно упростил и ускорил процедуры написания и отладки программ по сравнению с существующими в те годы вариантами, использующими машинные коды. Заметим, что в те времена, помимо цифровых машин, развивались и аналоговые

ЭВМ, которые обеспечивали высокую скорость вычислений, но были ограничены в количестве преобразований, так как с каждой операцией происходило быстрое накопление ошибок вычислений, возникали и другие трудности при их использовании. С дальнейшим развитием и совершенствованием цифровых ЭВМ, их распространением и расширением области применения создавались новые версии FORTRANa, которые позволили адаптировать язык к новым условиям. Были разработаны и другие языки программирования, ориентированные на решение различных прикладных задач.

С появлением новых языков высокого уровня делались заявления об их «неоспоримых» преимуществах перед другими языками программирования. Предлагалось прекратить использование «устаревших» языков, в том числе FORTRANa. Далее возникала типичная ситуация: следуя обещаниям, некоторые любознательные программисты предпринимали попытки переписывания уже существующих программ с языка FORTRAN на новые языки, что само по себе связано с весьма значительными трудозатратами, особенно если речь идет о больших программах. В этой связи заметим, что крупные комплексы программ создаются группами специалистов в течение многих лет и десятилетий. После написания и отладки новых версий программ проводились сравнения. Как правило, существенные преимущества у нового языка не обнаруживались, обещания оставались обещаниями. Нередко старые программы на FORTRANe оказывались более эффективными в вычислительном плане, чем новые. Интерес к новинке пропадал. За прошедшие десятилетия подобные ситуации повторялись неоднократно. Поэтому, несмотря на некоторый прогресс в развитии языков программирования, FORTRAN до сих пор используется при создании вычислительных программ.

Справедливости ради отметим, что, сохраняя основу языка, FORTRAN регулярно обновлялся и обновляется. Он подвергался многочисленным изменениям, особенно в последние годы. Это связано с развитием ЭВМ, изменением их архитектуры, изменением баланса между объемами памяти и производительностью, с внедрением многопроцессорных ЭВМ, с появлением новых периферийных устройств и адаптацией к новым операционным системам, программному обеспечению.

Одной из причин появления новых версий языка является программирование графических приложений, без которых сегодня трудно представить работу ЭВМ.

Развитие методов построения и обработки графиков, обработки фотографий, создания фильмов и компьютерных игр привело к разработке специализированных программ и языков программирования. Они позволяют эффективно решать подобные специфические задачи, в полной мере используя ресурсы графических процессоров, многочисленные приемы упрощения геометрических построений и ускорений вычислений. Поэтому вряд ли имеет смысл пытаться использовать для этих целей программы, написанные на языке FORTRAN. Вместе с тем накопленный положительный опыт решения научных и технических задач свидетельствует о целесообразности дальнейшего развития существующих программ, их поддержки и адаптации к новым условиям.

Проблема визуализации данных, полученных с помощью программ, написанных на языке FORTRAN, может быть решена следующим образом:

— подключением модулей, написанных на других языках программирования;

— написанием графических приложений на языке FORTRAN;

— передачей данных в специализированные программы обработки графической информации.

Перечисленные способы имеют свои преимущества и недостатки. Основными доводами в пользу применения графических модулей, написанных на языке FORTRAN, являются соответствие форматов данных, передаваемых из расчетных модулей в графические, возможность подключения графических модулей в любом месте вычислительной программы, отображения данных в графическом виде в ходе выполнения программы и при ее отладке, отсутствие необходимости углубленного изучения одновременно нескольких языков программирования и поддержки их многочисленных версий и обновлений, отсутствие необходимости согласования исполняемых модулей, написанных на разных языках, поддержки нескольких библиотек одновременно, устранения конфликтных ситуаций, связанных с совпадающими именами под-

ключаемых модулей различных библиотек, меньшие затраты при переходе с одних ЭВМ на другие и на другие операционные системы.

Естественно, все эти трудности могут быть преодолены. Программист сам может выбирать способ решения проблемы: написать свою полностью «независимую» программу или просто подключить свою вычислительную программу к набору «черных ящиков» сторонних производителей. Второй способ не требует больших усилий, но он ограничивает свободу действий, затрудняет модернизацию и перенос программ, так как это должно быть согласовано с производителями используемого программного обеспечения. Обновления подобных программ нередко связаны с покупками новых версий «черных ящиков». По существу, программист превращается в потребителя услуг. Логическим продолжением такого подхода являются отказ от собственных разработок и окончательный переход на коммерческие продукты сторонних производителей.

Основными недостатками применения графических модулей, написанных на языке FORTRAN, являются некоторые ограничения языка и менее высокая скорость выполнения графических приложений. Следует отметить, что графические приложения FORTRANa позволяют управлять цветом и яркостью каждой отдельной точки (пикселя) экрана. Это дает возможность при определенном опыте программирования создавать и обрабатывать сколь угодно сложные рисунки. Таким образом, вопрос о применении языка FORTRAN заключается лишь в удобстве написания графических программ и времени их выполнения. Заметим, что пользователей не интересует, на каком именно языке написана программа графической обработки, когда они рассматривают изображение на экране. Не имеет значения, с помощью каких программ нарисовали график или вывели текст на экран, важен сам график и сам текст.

Для предварительного ознакомления с особенностями языка FORTRAN можно рекомендовать публикации, приведенные в списке литературы. В данной книге вопросы программирования графических приложений рассмотрены на примерах решения частных задач. Также в книге приведены несколько простых примеров обработки текстовой информации с целью демонстрации приемов применения языка

FORTRAN при создании такого рода приложений. Тексты программ сопровождаются подробными комментариями. Для удобства изложения тексты не оптимизированы. Они служат для демонстрации способов решения задач и могут быть улучшены. На основе типовых приемов, представленных в этих примерах, достаточно просто могут быть реализованы свои версии программ обработки текстовых и графических файлов. Комментарии облегчают перевод текстов на другие языки. В случае необходимости может быть повышена производительность программ. В частности, могут быть применены более быстрые процедуры сортировки. При обработке большого количества файлов на многопроцессорных ЭВМ каждому процессору может быть выделен свой файл; при обработке графиков медленная процедура последовательного рисования линий от точки к точке может быть заменена более быстрой процедурой рисования многоугольника; при обработке сложных рисунков каждый процессор может обрабатывать свою часть рисунка. При создании многочисленных кадров фильма повторяющиеся элементы могут быть занесены в буфер и извлекаться из него без повторного формирования. Это касается и применения «медленных» процедур установки шрифтов. Предложения, слова и отдельные символы при многократном вызове иногда целесообразно запоминать в буфере и извлекать из него при необходимости. Могут быть использованы и другие средства повышения эффективности программ. Приемы и примеры оптимизации программ изложены в специализированной литературе. В технических приложениях в большинстве случаев используются графики сравнительно простого вида, не требующие больших вычислительных затрат. Эти графики легко могут быть реализованы собственными средствами, без использования дополнительных коммерческих программ.

Заметим, что в отличие от вычислительных программ обработка графической информации не всегда требует высокой производительности, так как обновление экрана происходит со сравнительно низкой частотой (порядка 60 Гц), время реакции человека составляет от несколько десятых долей секунды до нескольких секунд. В диалоговом режиме работы с ЭВМ чаще всего «слабым звеном» оказывается человек. При оформлении результатов уже после проведения исследований

основное время тратится на поиск нужных файлов с данными, а не на построение графиков. В таких случаях длительность формирования выводимого на экран кадра не столь важна. Быстродействие важно в случаях включения графических модулей непосредственно в вычислительную процедуру с целью обеспечения контроля сходимости решения в итерационном процессе. При наличии многих миллионов узлов расчетной сетки и сотен тысяч итераций трудоемкость графической обработки может оказаться существенной. В этих случаях имеет смысл использовать графические построения с некоторым достаточно большим шагом по итерациям, передачу данных в параллельно работающую независимую графическую программу через общие буферные файлы. С развитием ЭВМ, с распространением версий FORTRANa, поддерживающих многопроцессорную обработку, в том числе использующих ресурсы графических процессоров, скорость обработки графиков растет, и можно ожидать, что задача ускорения вычислений станет менее актуальной.

Так как версии FORTRANa работают с файлами рисунков формата *.BMP, которые занимают значительные объемы памяти, то при наличии многочисленных файлов с рисунками целесообразно использовать архиваторы, преобразующие их в другой, сжатый формат, например *.jpg. Как правило, это позволяет многократно сократить требуемые объемы памяти без существенной потери качества изображения. В случае необходимости повторной обработки таких файлов выполняется обратное преобразование в формат *.BMP. Это замечание касается и подключения различных «заставок» в виде фотографий, рисунков, графиков, полученных другими средствами и хранящихся в других форматах.

Алгоритмы обработки изображений, описания версий языка FORTRAN и приемы программирования графических процедур достаточно подробно изложены в литературе [1—28]. Поэтому многие общие вопросы, связанные с правилами написания программ на FORTRANe и геометрическими построениями, в книге не затрагиваются. Внимание уделено лишь некоторым особенностям программирования, которые могут представлять интерес для разработчиков программного обеспечения и студентов технических вузов.

В примерах приведены тексты программ в традиционном фиксированном формате. Файлы имеют вид (расширение) *.f или *.for. Комментарии отмечены латинской буквой «c» в первой колонке. Они не влияют на работу программы и могут быть удалены. Метки операторов располагаются в первых пяти колонках. Строки продолжения отмечены символом «*» в шестой колонке. Тексты программ (операторы FORTRANa) записаны в колонках 7—72. Примеры демонстрируют приемы обработки графической и текстовой информации простыми средствами языка FORTRAN. Эти приемы могут быть применены и при использовании других языков программирования.

# ГЛАВА 1

# ОБРАБОТКА ТЕКСТОВОЙ ИНФОРМАЦИИ

Обработка текстовой информации стала актуальной в связи с развитием сетевых технологий. Программы поиска файлов, текстов, ключевых слов нашли широкое применение. Обработка, сравнение и сортировка текстовых данных не являются типичными задачами, решаемыми средствами языка FORTRAN. Однако это не означает, что он не позволяет решать подобные задачи. При желании программиста перечисленные процедуры могут быть реализованы и на FORTRANe. Это можно продемонстрировать на следующих примерах.

## Пример 1. Сортировка строк текста (в данном случае списка литературы)

```
      character (40) namein

c Открыли выходной файл 2 с именем "SpisokLiteratuti.res"
      OPEN (UNIT=2,FILE='SpisokLiteratuti.res')

c Открыли входной файл 5 (список литературы) с именем "5.txt"
      namein='5.txt'
      open (unit=5,file=namein,status='old',iostat=iche)
      if(iche.eq.0) write (2,*) 'Найден файл  ',namein
      if(iche.ne.0) write (2,*) ' Stop! Не найден файл  ',namein
c Если файл не найден, то идем на выход
      if(iche.ne.0)  go to 1000

c Открыли выходной файл 6 с именем "6.txt"
      OPEN (UNIT=6,FILE='6.txt')
```

```
c  Вызвали подпрограмму сортировки текста
      call SortText

c Выход из программы
 1000 continue
      stop
      end

c   Подпрограмма сортировки текста

      subroutine SortText
c Выделили массивы для обработки текста
c (до 2000 строк по 1000 символов)
      character  (1)  text(2000,1000)
      character  (1)  stroka(1000)
      character  (1)  probel,bukwa1,bukwa2
c Задали значение символьной переменной probel
      data probel/" "/
c Ограничили число строк текста
      nstrok=2000
c Считывание текста из файла (в примере из файла №5)
c Перебор строк данных
      do 50 i=1,nstrok
c Считывание одной строки из файла данных (5).
c Поочередно считываем символы строки.
      read
     *(5,'(1000a1)',advance='no',end=5557,err=101,iostat=iche)
     *      (stroka(j),j=1,1000)
  101 continue

c В случае, если количество символов в строке меньше заданной
c величины (в данном случае 1000), выходим из цикла
c считывания символов и идем  к оператору (101 continue).
c В случае, если в файле число строк меньше заданного значения,
c выходим из цикла считывания строк и идем к оператору
c (5557 continue).
```

```
c Определили длину считанной строки jstr
      jstr=j

c Записали считанную строку в массив text(i,j)
      do 2 j=1,jstr
    2 text(i,j)= stroka(j)
c Очистили "хвост" строки от посторонних символов
c (заполнили пробелами)
      jstrp1=jstr+1
      do 22 j=jstrp1,1000
   22 text(i,j)= probel

c Распечатали считанную строку текста в файле 2 (для контроля)
      write (2,*) (text(i,j),j=1,jstr)
   50 continue
c Конец считывания текста из файла (5)

 5557 continue
      nst=i-1
      write (2,*)
      write (2,*) 'Число строк текста =',nst
      write (2,*)

c Сортировка текста
      nstrokm1= nst-1
c Первая строка  (stroka1). Перебор первых строк
      do 3 i=1,nstrokm1
      ip1=i+1

c Вторая строка (stroka2). Перебор вторых строк,
c расположенных ниже первой строки
      do 103 i1=ip1,nst

c Перебор первых 10 столбцов строки №1 (пример).
c Пропускаем цифры (старый номер строки №1
```

```
с в списке литературы).
с Ищем номер позиции первой буквы в текстовой строке
      j1=0
      do 14 j=1,10
      bukwa1= text(i,j)
      call kodd(bukwa1,kod1)
      if(kod1.ge.1) j1=j
      if(kod1.ge.1) go to 114
   14 continue
  114 continue
      if(j1.eq.0) go to 3

      j2=0
с Перебор первых 20 столбцов строки №2.
с Пропускаем старый номер строки №2 в списке литературы.
с Ищем первую букву
      do 15 j=1,20
      bukwa2= text(i1,j)
      call kodd(bukwa2,kod2)
      if(kod2.ge.1) j2=j
      if(kod2.ge.1) go to 115
   15 continue
  115 continue
      if(j2.eq.0) go to 103

с Задаем метку повторения строк
      metkapowt=0
с Сравниваем содержания строк №1 и №2.
с Последовательно перебираем буквы в строках
с (в примере проверяем первые 500 символов)
      do 4 j=1,500
      bukwa1= text(i, j+j1-1)
      bukwa2= text(i1,j+j2-1)
      call kodd(bukwa1,kod1)
      call kodd(bukwa2,kod2)
```

```
c Если буквы строк разные, меняем метку повторения строк
      if(kod1.ne.kod2) metkapowt=metkapowt+1

c Сравниваем буквы (по номерам их кодов).
c Идем на перестановку строк местами,
c если сравниваемые буквы расположены не в заданном порядке
      if(kod1.gt.kod2) go to 5
c Обнаружены одинаковые буквы, переходим на сравнение
c следующих букв строк
      if(kod1.eq.kod2) go to 4
c Строки расположены в правильном порядке,
c переходим к сравнению следующей строки
c (берем следующую строку №2)
      go to 13
    4 continue
c Переходим к сравнению следующей строки
c (берем следующую строку №2)
      go to 13

c перестановка строк местами (переставляем символы)
    5 continue
c         write (2,*) 'Переставляем строки'
c         write (2,*) (text(i,jj) ,jj=1,100)
c         write (2,*) (text(i1,jj),jj=1,100)
c         write (2,*)
      do 6 j=1,1000
      bukwa1  =text(i,j)
      text(i,j) =text(i1,j)
      text(i1,j)=bukwa1
 6    continue
c         write (2,*) 'Строки после перестановки'
c         write (2,*) (text(i,jj) ,jj=1,100)
c         write (2,*) (text(i1,jj),jj=1,100)
c         write (2,*)
c Конец перестановки строк
```

```
      go to 103

   13 continue
c Заполняем пробелами повторяющуюся копию строки
      if(metkapowt.ne.0) go to 103
      do 16 j=1,1000
      text(i1,j)=probel
   16 continue

  103 continue
c Конец перебора строк №2
    3 continue
c Конец перебора строк №1

      write (2,*)
      write (2,*)
      write (2,*) '       Результат сортировки'
      write (2,*)

c Перебор строк данных (в примере запись результата в файл №6)
      nomer=0
      do 60 i=1,nst
c Пропускаем старый номер строки
      j1=0
c Проверяем первые 10 символов строки
      do 1240 j=1,10
      bukwa1= text(i,j)
      call kodd(bukwa1,kod1)
      if(kod1.ge.1) j1=j
      if(kod1.ge.1) go to 1214
c Стираем старый номер строки текста (заполняем пробелами)
      text(i,j)=probel
 1240 continue
 1214 continue
```

```
      if(j1.le.0) j1=1
      jk=0
      kod1sum=0
c Перебираем символы строки
      do 61 j=j1,1000
      bukwa1= text(i, j)
c Определяем код символа
      call kodd(bukwa1,kod1)
      if(kod1.ne.0) jk=j
      if(kod1.ne.0) kod1sum=kod1sum+1
   61 continue
c Количество символов в строке без учета пробелов (kod1sum)
      if(kod1sum.le.0) go to 60
c Новый номер строки в списке литературы (nomer)
      nomer=nomer+1
c Длина строки jk  (увеличили длину строки на 10
c  с целью учета возможного наличия других
c  неучтенных символов в тексте)
      jk=jk+10
c Печатаем полученный список (без номеров строк) в файл 2
      write (2,*) (text(i,j),j=1,jk)
c Записываем строку (с новым номером строки) в файл 6
      write (6,'(i3,a2,1000a1)') nomer,'. ',(text(i,j),j=j1,jk)
   60 continue
c Конец записи текста в  файл 6.txt

 5000 continue
      stop
      end
```

Заметим, что в современных версиях FORTRANa имеются процедуры, с помощью которых можно создать другие варианты программы, решающей поставленную задачу. В представленных примерах реализованы наиболее простые средства, которые используют традиционные