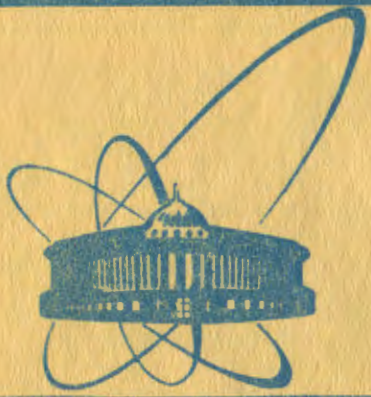

сообщения
объединенного
института
ядерных
исследований
дубна

1657
2-81

30/III-81

Р10-81-58

Г.Л.Бутцева, Н.Н.Воробьева, А.С.Завьялова,
Л.С.Нефедьева, А.А.Расторгуев, А.В.Романовский,
А.И.Салтыков, В.Н.Тарасова, Г.Ш.Успанова,
Д.Цэрэндулам, В.М.Ягафарова

МОДУЛЬНЫЙ ПРИНЦИП ПОСТРОЕНИЯ ПРОГРАММ ОБРАБОТКИ В СИСТЕМЕ ОБРАБОТКИ СПЕКТРОВ (СОС)

1981

Введение

Для обработки спектрометрической информации создано большое число различных комплексов математических программ, которые ориентированы на группу однотипных экспериментов и, как правило, имеют свою специфическую организацию ввода-вывода (см., например, /4-9/).

Многие программы из разных комплексов выполняют аналогичные функции (выделение пиков, калибровки и т.п.). Так как в комплексе все программы, как правило, связаны единым алгоритмом обработки информации, то это не дает возможности использовать такие программы для обработки данных, полученных в аналогичных экспериментах.

Указанные неудобства могут быть устранены путем разбиения единого алгоритма обработки на независимые единицы, обмен между которыми должен быть строго стандартным. Такой подход позволяет решить ряд проблем:

1) сократить параллелизм при разработке программ обработки;

2) освободить по возможности до минимума программы обработки от организации ввода-вывода информации;

3) объединить все программы обработки в библиотеку, построенную по единому принципу.

Проблемно-ориентированная система обработки спектрометрической информации (СОС)/1/, созданная на ЭВМ БЭСМ-6 в рамках операционной системы "Дубна", позволяет решить эти проблемы.

Система СОС включает в себя набор программ, организующих работу с файлами, транслятор с языка директив и библиотеку программ обработки спектрометрической информации/10/. В библиотеку СОС можно включать любые программы, удовлетворяющие определенным системным требованиям и оформленные в виде программных модулей/2/.

Программный модуль (или просто модуль) в системе СОС должен обладать следующими свойствами:

- массивы данных, используемые модулем в качестве исходных, должны быть организованы в виде файлов системы;
- массивы данных, полученные в результате работы модуля и предназначенные для последующей обработки (посредством других модулей), также должны быть файлами системы;
- место расположения файлов задается модулю извне.

Предполагается, что программы, составляющие модуль, пишутся на ФОРТРАНе, автокоде МАДЛЕН или любом другом языке системы "Дубна"[3], допускающем использование фортранной PROGRAM в качестве головной программы пакета задачи.

В дальнейшем, говоря о требованиях к модулям, мы будем иметь в виду головную программу модуля, то есть ту программу, к которой производится обращение из задания пользователя.

I. Требования и рекомендации, связанные с включением программного модуля в СОС

Требования, предъявляемые системой к модулю, обязательны. Без их выполнения программа не сможет работать в рамках системы СОС. Выполнение же рекомендаций не является обязательным, но повышает эффективность работы модуля.

I.I. Требования

Модуль вызывается посредством оператора CALL и, следовательно, должен быть оформлен по правилам фортранной программы SUBROUTINE. В качестве фактических параметров могут быть только имена файлов, числа целые и вещественные, а также холерические константы.

Информация между модулями не может быть передана через COMMON-блоки. Однако каждый модуль может вызывать любые другие подпрограммы и передавать им любую информацию как через фактические параметры, так и через COMMON -блоки. Допускается также вызов одних модулей из других.

I.2. Рекомендации

Рекомендации касаются использования памяти машины (оперативной и внешней), а также наименований модулей.

Во время работы системы СОС часть оперативной памяти отводится под рабочие поля системы, которые можно использовать и как рабочие

поля модулей. В частности, система использует COMMON -блок /COC1/, длина которого всегда не менее 4200 и может быть увеличена по желанию пользователя. Этот блок удобно использовать при считывании массивов данных из файлов системы и записи таких массивов в файлы (см. п.2).

Кроме COMMON -блока /COC1/ можно применять и другие блоки. При этом в целях экономии памяти удобно описывать эти блоки на минимальную длину 1 . Фактическая же длина блока (зависящая от конкретных условий работы модуля) будет определяться заданием.

В связи с тем, что COMMON -блоки позволяют более рационально использовать оперативную память, не рекомендуется задавать массивы в качестве формальных параметров модуля. В программных модулях допускаются любые операторы ввода-вывода. Однако удобнее использовать системный аппарат ввода -вывода, описанный ниже.

При выборе имени программного модуля (а также имен подпрограмм, к которым этот модуль обращается)желательно придерживаться такого правила: эти имена не должны начинаться с буквы "С". Дело в том, что многие системные программы начинаются с этой буквы. Поэтому выполнение указанного требования заведомо исключает совпадение имен подпрограмм системы и модуля.

2. Взаимодействие модуля с системой

Системный аппарат организации ввода-вывода, которым могут пользоваться программные модули, предусматривает хранение массивов данных во внешней памяти и организацию их в виде файлов. Путем обращения к системной программе CGET модуль может произвести считывание одной записи из файла в оперативную память. Если модуль отдает системе на хранение массив данных, то это делается путем обращения к системной программе CPUT . При этом массиву данных будет соответствовать одна запись в файле. Имена файлов, используемые модулем в программах CGET и CPUT , должны быть формальными параметрами данного модуля.

2.1. Считывание массивов данных из файла

Путем обращения

```
CALL CGET(IFILE,FIELD,L,IFOR)
```

производится считывание одной записи из файла IFILE на рабочее поле модуля FIELD. Здесь IFILE - массив из двух слов. В IFILE(1) хранится системное имя файла, а в IFILE(2) задается номер считыва-

емой записи. Значение IFILE(1) не может быть изменено в процессе работы модуля, а значение IFILE(2) может быть задано извне либо определено в процессе работы модуля. Если задано IFILE(2)=0 , считывается первая запись, а если значение IFILE(2) превосходит максимально допустимый номер записи, то выдается последняя запись. Если указано отрицательное значение IFILE(2) , выдается диагностика, и выполнение задания прекращается. Возможны еще два случая, когда считывание массива данных не может быть выполнено:

- система работает в режиме большого буфера (заданного пользователем) и длина запрашиваемой записи превосходит 4096 слов;
- файл еще пуст.

В качестве FIELD задается начало рабочего поля МОЗУ, куда выдается содержимое считываемой записи. Размер поля FIELD должен быть достаточным для размещения считываемой записи. Рекомендуется использовать в качестве рабочих полей модуля COMMON -блоки и в первую очередь COMMON/COC1/, который,как говорилось, всегда присутствует в оперативной памяти и имеет длину не менее 4200.

Значения параметров L и IFOR выдаются программой CGET. L содержит длину считанной записи, а IFOR - признак формата, характеризующий информацию, содержащуюся в этой записи. Признак формата может быть использован для целей контроля правильности считанной информации. Этот контроль должен осуществляться самим модулем. В системе имеются следующие стандартные признаки формата:

1 - спектр;
2 - набор физических параметров;
3 - любая числовая информация;
4 - текстовая информация.

В качестве признака формата допускаются любые целые числа от 0 до 8.

2.2. Запись массивов данных в файл

Если модулю необходимо записать массив данных в файл, то надо указать оператор

```
CALL CPUT(FIELD,IFILE,L,IFOR) .
```

Здесь параметры имеют тот же смысл, что и для CGET, однако значения параметров L и IFOR задаются на входе. Если на входе задано $L \leq 0$ или IFOR<0 , или IFOR > 8 , то выдается диагностика и выполнение задания прекращается.

Значение IFILE(2),то есть номер записи, куда заносится массив данных из FIELD, должно быть равно либо n+1 (для файлов любого ти-

па), либо от 1 до n+1 (только для файлов типа UPDATE). Здесь n – число записей, имеющихся в файле IFILE в данный момент. Если значение IFILE(2) равно -1, то массив данных заносится в (n+1)-ю запись, где n – номер последней записи. Если задано недопустимое значение номера записи, то выдается диагностика и задача снимается со счета.

3. Рекомендации по составлению программных модулей

Каждый программный модуль может использовать один или несколько файлов на входе и на выходе. Возможны два режима работы модуля: однотактный, когда из каждого файла берется только одна запись, и циклический, когда из файлов берется по нескольку записей.

В случае однотактного режима работы в качестве формальных параметров указываются исходные и результирующие файлы, причем модуль работает с текущими номерами записей, заданными на входе.

Пример:

```
      SUBROUTINE SUM(KFIL,LFIL,IRES)
      DIMENSION KFIL(2),LFIL(2),IRES(2)
      COMMON/COC1/F(4200)
      DO 1 I = 1,2000
    1 F(I)=0.
      CALL CGET(KFIL,F,LKF,IKF)
      CALL CGET(LFIL,F(1001),LLF,ILF)
      M=MAXO(LKF,LLF)
      DO 2 I=1,M
    2 F(I+2000)=F(I)+F(I+1000)
      CALL CPUT(F(2001),IRES,M,1)
      RETURN
      END .
```

В приведенном примере производится поэлементное суммирование содержимого текущих записей исходных файлов KFIL и LFIL , и образуется одна запись в результирующем файле IRES. Предполагается, что длина каждой записи не превосходит 1000.

Если модуль работает в циклическом режиме, то в качестве формальных параметров, кроме имен файлов, необходимо для каждого обрабатываемого файла указать две простые переменные целого типа. Одна из этих переменных предназначается для начального, а другая – для конечного номера записи соответствующего файла. Эти значения задаются на входе.

4. Использование модулей вне системы СОС

Структура модулей позволяет использовать их без существенных изменений и вне системы СОС. Для этого достаточно заменить программы CGET и CPUT другими программами (с теми же именами), выполняющими аналогичные функции. В частности, на машинах серии ЕС ЭВМ имеется удобный аппарат работы с файлами прямого доступа в рамках языка ФОРТРАН-IУ. Поэтому возможна сравнительно нетрудная адаптация задействованных на БЭСМ-6 модулей для ЕС ЭВМ.

Заключение

Описанный здесь модульный принцип организации программ обработки вполне оправдал себя на ЭВМ БЭСМ-6. В настоящее время в библиотеку СОС включено сорок программных модулей обработки спектров, сорок два – предварительной обработки и пятнадцать – ввода-вывода.

Литература

I. Нефедьева Л.С. и др. Автоматизированная система обработки спектров (СОС) на машине БЭСМ-6. "Зинатне", Рига, I975, с.4.

2. Йодан Э. Структурное проектирование и конструирование программ. "Мир", М., I979, с.II6.

3. Мазный Г.Л. Программирование на БЭСМ-6 в системе "Дубна" /под ред. Н.Н.Говоруна/. "Наука", М., I978.

4. Аврамов С.Р., Сосновская Е.В., Цупко-Ситников В.М. ОИЯИ, PIO-974I, Дубна, I976.

5. Rutti J.T., Prussin S.G. Nucl. Instr. & Meth., 1969, 72,125.

6. Волков Н.Г., Чубченко В.Г., Чураков А.К. Прикладная ядерная спектроскопия, вып.5, I975, с.24.

7. Кабина Л.П., Кондуров И.А. Препринт ФТИ им. А.Ф.Иоффе, № 399, Л., I972.

8. Гаджоков В. и др. ОИЯИ, PIO-I2724, Дубна, I979.

9. Винель Г.В., Цупко-Ситников В.М., Элер Г. ОИЯИ, IO-IO843, Дубна, I977.

IO. Аврамов С.Р. и др. ОИЯИ, БI-IO-80-345, Дубна, I980.

Рукопись поступила в издательский отдел
27 января I98I г.