Разработка системной поддержки вызова программ, реализованных на языке Fortran, из среды Java.

С.С. Гайсарян, К.Н. Долгова

Аннотация. Статья посвящена исследованию возможности вызова программ, реализованных на языке Fortran 95, из среды Java. Для того чтобы среды могли обмениваться данными, должно быть отображение данных одной среды на данные другой. В статье представлено описание отображения данных языка Fortran на данные языка Java и обратно. Также описан способ эффективной передачи данных из среды Java в среду Fortran и обратно. Он заключается в том, что память, выделенная средой Fortran для размещения общих блоков и массивов, отождествляется с прямыми буферами среды Java. То есть прямые буферы среды Java размещаются по тем же адресам памяти, по которым размещены общие блоки и массивы языка Fortran. Помимо этого, в статье описан метод организации вызова подпрограмм, реализованных на языке Fortran из окружения Java, заключающийся в передаче параметров через прямые буферы окружения Java.

1. Введение

Имеется достаточно большое количество программ, реализованных на языке Fortan и не потерявших ценность.

В настоящее время широкую популярность получила среда программирования Java, обеспечивающая переносимость программ.

Следовательно, возникает потребность иметь возможность вызывать подпрограммы, реализованные на языках Fortan, из Java-программ.

Для вызова подпрограмм, реализованных на языке C из Java-программ есть JNI, который доступен начиная с версии JDK 1.2. Аналогичного интерфейса для вызова Fortran-подпрограмм нет. Предложенная работа повящена разработке методики вызова Fortran-подпрограмм из Java-среды.

В настоящей работе рассмотрены основные отличия языков C и Fortran, препятствующих использованию методике аналогичной JNI для вызова Fortran-подпрограмм из Java-программ. Построено отображение данных языка Fortran на данные Java и обратно. Предложена методика реализации общей области памяти для Java- и Fortran-сред через прямые буферы пакета java.nio. В последнем разделе описана прототипная реализация, выполненная с

использованием JNI, которая показала эффективность предложенной методики.

2. Отличия языков C и Fortran

У языков программирования C и Fortran, существует ряд различий, из-за которых нельзя перенести организацию JNI для языка C на организацию подобного интерфейса для языка Fortran.

(1) В стандарте языка C напрямую не указан размер примитивных типов [1]. Выбор наилучшего для данной архитектуры размера типов оставлен на рассмотрение разработчиков компилятора. В стандарте языка Fortran для каждого примитивного типа данных строго задан их размер. Это позволяет установить взаимно однозначное соответствие между типами языка Java и типами языка Fortran, не используя промежуточных типов, как это реализовано в JNI.

(2) Среда Fortran размещает данные в статической области памяти программы. К данным есть доступ только по ссылке, и нет возможности получить адрес памяти, где они расположены. Среда Fortran не поддерживает динамически создаваемых объектов данных. Среда C, во-первых, располагает данные программы в стеке, в куче и в статической области памяти программы, во-вторых, определена операция взятия адреса, позволяющие получить доступ не только к значениям данных, но и к адресам памяти, где они расположены. Соответственно, для передачи данных из Java-среды в C-среду JNI достаточно указать адрес области памяти, где данные хранятся. Передачу данных из Java-среды в Fortran-среду нельзя выполнить аналогично тому, как это сделано в JNI.

(3) Все параметры в языке Fortran передаются только по ссылке, потому что в нем не определено понятие адреса переменной. В языке C параметры передаются только по значению, однако есть возможность передавать в качестве параметра функции указатели на ту область памяти, где хранится переменная. Соответственно, передачу данных из среды Java в среду Fortran и обратно нельзя выполнить аналогично тому, как это сделано в JNI.

(4) В многомерных массивах языка C данные располагаются по строкам, тогда как в многомерных массивах языка Fortran данные располагаются по столбцам. В языке Fortran есть возможность непосредственно работать с частями массива – вырезками и сечениями. В языке C такой возможности нет. Следовательно, методика передачи массивов, реализованная в JNI, не может быть применена для среды Fortran.

(5) В языке Fortran есть общие блоки COMMON. Эти блоки можно размечать по-разному в каждой подпрограмме. Так, например, в одной подпрограмме может быть объявлен массив типа complex размера 100,

расположенный в общем блоке /A/, а в другой подпрограмме на этой же памяти, то есть в том же общем блоке /A/ может быть объявлен массив типа real размера 200. Оба массива будут размещаться в памяти, начиная с одного и того же виртуального. Данные, которые в нем расположены – одни и те же, однако тип данных разный. В языке C аналогичная возможность может быть реализована посредством использования объявления union. Однако передача данных, расположенных в COMMON блоках с целью эффективности должна выполняться по схеме, отличной от той, которая реализована в JNI. Однако передача данных, расположенных в COMMON блоках с целью эффективности должна выполняться по схеме, отличной от той, которая реализована в JNI для передачи данных, объявленных в union.

Учитывая то, что в реализации связывания подпрограмм, написанных на языке Fortran, с Java окружением должна быть сделана эффективная передача данных между Fortran-подпрограммами и основным Java-модулем, а так же принимая во внимание отличия языков C и Fortran, можно сделать вывод о том, что организация связывания между виртуальной машиной Java и подпрограммами, реализованными на языке Fortran, должна осуществляться по несколько иной схеме, нежели связывание C-методов и виртуальной машины Java.

3. Размещение данных в среде Fortran

Программа, написанная на языке Fortran, допускает использование следующих видов программных единиц: стандартных функций, подпрограмм FUNCTION, подпрограмм SUBROUTINE, операторов – функций, подпрограмм, написанных на других языках программирования, и подпрограмм BLOCK DATA [2].

Формальные параметры языка Fortran передаются обычно по ссылке, за исключением тех случаев, когда параметр не модифицируется в подпрограмме [3]. В языке Fortran имеются средства, позволяющие использовать одну и ту же область памяти для хранения данных, общих для двух или более программных модулей выполняемой программы. Таким средством является общий блок [3]. Значения объектов из общего блока доступны всем программным единицам, в которых этот блок описан [2]. Каждый общий блок обязательно занимает в памяти непрерывный участок. Если некоторый программный модуль содержит несколько объявлений COMMON с одним и тем же именем, то все они рассматриваются как одно объявление и располагаются в памяти непрерывно и последовательно.

Для объявления массивов в языке Fortran существуют специальные предложения спецификации: объявление размерности DIMENSION [3]. Так же массивы могут быть расположены в общих блоках. Массивы, полученные объявлением DIMENSION, представляют собой локальные данные той подпрограммы, внутри которой они описаны.

При вызове подпрограмм, реализованных на языке Fortran, из Java-окружения необходимо передавать данные из Java-среды в Fortran-среду. Также может возникнуть необходимость передавать данные из среды Fortran в среду Java, если вызываемая программная единица из среды Fortran – FUNCTION – возвращает значение. Java-среда передает все параметры только по значению, в среде Java нет методов работы с указателями, а все данные Java-программы расположены в куче.

Любая подпрограмма, реализованная на языке Fortran, может получать данные извне либо как параметры, либо через общие блоки, которые в ней описаны.

Если Fortran-подпрограмма получает данные для обработки через общие блоки, то вызывающая Java-программа должна иметь доступ на запись и чтение к той памяти, в которой эти общие блоки расположены. Такой доступ Java-программе возможно обеспечить, если на памяти, где располагается общий блок, разместить Java-объект. В качестве такого Java-объекта может быть взят прямой буфер класса Buffer, методы работы с которым доступны через пакет java.nio. Для того чтобы получить такой буфер, нужно из Fortran среды передать адрес начала общего блока и его размер. Дальше достаточно создать прямой буфер байтов, адрес начала которого будет совпадать с адресом начала общего блока, а размер будет такой же, как у соответствующего общего блока.

Если Fortran-подпрограмма получает данные для обработки через формальные параметры, то для передачи таких параметров из Java-окружения необходимо выделить прямой буфер в Java-окружении, на который будут помещены передаваемые параметры. После того, как передаваемые параметры будут расположены на буфере, Fortran-подпрограмме нужно передавать только адрес этого буфера и смещение в нем, по которому расположен соответствующий параметр.

Возврат данных из функций языка Fortran осуществляется по значению. Для передачи возвращаемого значения функцией языка Fortran в Java-окружения нужно это значение располагать в той области памяти, которая доступна и Java-окружению, и среде Fortran. Такой областью памяти с точки зрения среды Java может выступать прямой буфер. На нем необходимо выделить место для значения, возвращаемого функцией среды Fortran, и передать смещение в буфере Fortran-функции как параметр. А Fortran-функция запишет по полученному адресу возвращаемое значение.

4. Отображение типов данных языка Java в типы данных языка Fortran

Основные типы языка Java и соответствующие им типы языка Fortran представлены в таблице 1. Данные для таблиц взяты из литературы [4] и [2].

Тип данных языка Java	Требуемый объем памяти	Тип данных языка Fortran
Int	4 байт	INTEGER
Short	2 байт	INTEGER*2
Long	8 байт	INTEGER*8
Byte	1 байт	CHARACTER
Float	4 байт	REAL
Double	8 байт	DOUBLE PRECISION
Char	2 байт	CHARACTER
Boolean	1 байт	LOGICAL*1

Таблица 1. Отображение примитивных типов языка Java в типы языка Fortran.

Массив языка Java можно отобразить на такое представление данных языка Fortran как массив. Отображение массива языка Java на массив языка Fortran можно сделать через прямой буфер, средства работы с которым предоставлены в пакете «java.nio».

Тип данных языка Fortran	Требуемый объем памяти	Тип данных языка Java
INTEGER*2	2 байт	short
INTEGER INTEGER*4	4 байт 4 байт	int int
REAL REAL*4 DOUBLE PRECISION REAL*8 REAL*16	4 байт 4 байт 8 байт 8 байт 16 байт	float float double double double double
COMPLEX COMPLEX*8 COMPLEX*16 COMPLEX*32	8 байт 8 байт 16 байт 32 байт	float float float float double double double double double double
LOGICAL*1 LOGICAL LOGICAL*4	1 байт 4 байт 4 байт	byte int int
CHARACTER CHARACTER*L	1 байт L байт	byte string

Таблица 2. Отображение примитивных типов языка Fortran в типы языка Java.

Данные в Fortran-программах могут быть представлены в виде констант или имен переменных (или идентификаторов).

Основные типы языка Fortran и соответствующие им типы языка Java представлены в таблице 2. Данные для таблиц взяты из литературы [2] и [4]

Для отображения данных, определенных в общем блоке, в окружении Java следует использовать прямой байт буфер. Такое отображение легко организовать, потому что общий блок представляет собой некоторую область памяти, хранящую неоднородные данные. Прямой байт буфер, доступный в Java-окружении также представляет собой область памяти, которая может хранить неоднородные данные.

Для каждого как именованного, так и неименованного общего блока можно использовать по одному буферу.

В языке Fortran массивом называется упорядоченная последовательность данных, занимающая непрерывную область памяти, к которой можно обращаться по имени [2]. Массивы характеризуются типом значений их элементов и граничными парами – диапазоном индексов по каждому измерению.

Несмотря на то, что Fortran массивы могут быть как одномерными, так и многомерными, в памяти они располагаются как одномерный массив. Причем элементы многомерного массива располагаются в памяти таким образом, что значение первого индексного выражения возрастает быстрее второго, значение второго – быстрее третьего и т. д. [2].

Следовательно, приведенный индекс многомерного массива можно рассчитать по ниже приведенной формуле, а именно: пусть имеется многомерный массив arr[N, M, K], тогда приведенный индекс элемента arr[i, j, k] рассчитывается следующим образом: $(i-1)+(j-1)*N+(k-1)*N*M$.

Массив языка Fortran следует отображать на прямой байт буфер, доступный в Java среде. Такое представление выгодно, потому что многомерный Fortran-массив в памяти располагается как одномерный массив. Для получения данных из прямого буфера соответствующих элементу многомерного Fortran-массива в Java окружении реализуется специальный класс.

Так как многомерный массив не может иметь больше 7 измерений [5], то всегда можно автоматически получить данные из прямого буфера, соответствующие элементу многомерного Fortran-массива в Java окружении. При этом следует обойтись без транспонирования самого Fortran-массива.

Для ссылки на элемент массива задается индексированная переменная; на массив в целом ссылаются по его имени. Начиная со стандарта Fortran 90, в языке есть возможность непосредственно работать с частями массива – вырезками и сечениями. Вырезка из массива представляет собой подмассив вида

```
<имя_массива>(< нижняя граница – верхняя граница),
```

Элементы вырезки из массива могут быть взяты с шагом, отличным от единицы. В этом случае вырезка по соответствующему измерению задается уже не граничной парой, а триплетом.

```
<имя_массива>(< нижняя граница – верхняя граница, шаг >,…)
```

Если по какому-то измерению опущены обе границы, то говорят о сечении массива. Вырезку из массива можно также задать с помощью векторного индекса[5].

Для отображения вырезки или сечения массива на объекты Java среды также можно использовать байт буфер. Такое отображение можно выполнить следующим образом. Весь массив отображается на буфер, а дальше в Java-среде организуется специальный класс, содержащий методы получения и записи элементов вырезки и сечения массива посредством пересчета с учетом шага.

Таким образом, любой массив языка Fortran можно отобразить на прямой байтовый буфер языка Java. Если массив размещен на общем блоке, он автоматически отобразится в Java окружение при отображении общего блока. Если массив определен посредством использования оператора DIMENSION, то для него надо создать прямой буфер, расположенный на том участке памяти, который компилятор языка Fortran выделил для хранения данного массива. Что касается вырезки и сечения массивов, то это представление данных можно отобразить через указатели на соответствующие элементы.

5. Вызов Fortran-подпрограмм из Java-среды

При вызове Fortran-подпрограмм из Java-среды необходимо учитывать особенности чтения данных в Java- и Fortran-средах.

Java-машина читает байты, в которые записано одно число, слева направо (прямое чтение), а в C- и Fortran- – программах порядок байт в записи чисел зависит от архитектуры. То есть, на некоторых платформах используется чтение справа налево (так называемое, инвертированное чтение). Следовательно, на некоторых платформах для корректной работы, данные, записанные Fortran-подпрограммой, нужно подвергнуть дополнительному преобразованию в формат языка Java, чтобы Java-программа прочитала их корректно. И наоборот, данные, записанные Java-программой, тоже надо подвергать обратному преобразованию в формат языка Fortran, чтобы подпрограмма, реализованная на языке Fortran, смогла прочитать именно то, что было помещено в Java-коде. В выше упомянутом преобразование предполагается менять местами соответствующие записи в ячейках. Такое преобразование необходимо осуществлять каждый раз, когда обработка данных, расположенных в памяти, общей и для Java-окружения, и для среды Fortran, передается от Java-машины Fortran-среде и обратно.

Такое преобразование предполагается целесообразным выполнять при каждой записи виртуальной машиной Java данных в общую память и при каждом считывании данных из общей памяти Java-машиной. Следовательно, среда Fortran всегда будет обрабатывать данные, записанные в формате языка Fortran, а Java-машина всегда будет работать с данными, записанными в формате языка Java.

6. Описание практической части

Прототипная реализация выполнена посредством связывания вызова подпрограммы, реализованной на языке Fortran, из Java-программы через язык C (JNI). В настоящее время окружение Java не предоставляет возможности вызывать напрямую подпрограммы, реализованные на языке Fortran.

Реализация выполнена для GNU компилятора Fortran (g77), GNU компилятора C (gcc) версии 3.3.4 и JDK версии 1.4.2_03.

Компилятор g77 основан на стандарте ANSI Fortran 77, но он включает в себя многие особенности, определенные в стандартах Fotran 90 и Fortan 95 [6].

JDK версии 1.4.2_03 содержит пакет java.nio, который предоставляет возможность использования новых средств ввода-вывода таких как прямые буферы и JNI (Java Native Interface).

Как уже отмечалось в пункте 2, прежде чем выполнить вызов подпрограммы, реализованной на языке Fortran, из Java-среды, необходимо выделить область памяти, которая была бы доступна как из Java-окружения, так и из среды Fortran.

Для этого нужно:

1. На языке Fortran реализовать подпрограмму. В этой подпрограмме должны быть объявлены все общие блоки, которые будут использоваться для обмена данными Fortran-среды с Java-окружением.
2. На языке C должен быть реализован модуль, который через разделяемую библиотеку посредством JNI будет вызываться из Java-среды. Модуль должен содержать функцию, которая вызывается из среды Fortran. Данной функции, в качестве параметров по ссылке, из Fortran-среды передается адрес первого, адрес последнего элемента и размер в байтах последнего элемента общего блока. По полученным данным вычисляются и сохраняются начало и размер общего блока. Такая функция вызывается для каждого общего блока. Некоторая функция вычисляет и сохраняет размер общего блока, а так же сохраняет адрес начала общего блока. Теперь во встроенном модуле, реализованном на языке C, хранятся адреса и размеры всех общих блоков, которые определены в подпрограмме, реализованной на языке Fortran. Следовательно, запросив по указанному адресу прямой буфер нужного размера, будет получено размещение нового байт-буфера Java-среды на том же участке памяти, что и соответствующий ему общий блок.

3. На языке Java реализуется класс, который содержит метод инициализации и метод получения прямого байт-буфера. Метод инициализации вызывает встроенный метод инициализации, реализованный на языке C в описанном в пункте 2 модуле. Встроенный метод инициализации вызывает Fortran-подпрограмму, описанную в пункте 1. Метод получения прямого байт-буфера вызывает встроенный C-метод, который заказывает в оперативной памяти прямой буфер нужного размера, начиная с указанного адреса. Дальше полученный прямой байт-буфер уже сам пользователь может представлять как буфер тех данных, которые ему нужны.

Байт-буферы расположены непосредственно в том же участке памяти, что соответствующие им общие блоки, следовательно, все данные, которые записываются в прямой буфер в Java-коде, автоматически становятся доступными из общего блока в коде, реализованном на языке Fortran. И наоборот: все, что помещено в общий блок в Fortran-подпрограмме, автоматически становится доступно из прямого буфера в Java-программе. Такое расположение данных полностью решает поставленную в пункте 1 задачу о совместном размещении данных Java-окружения и среды Fortran на одном участке памяти.

Чтобы выполнить вызов Fortran-подпрограммы из Java-среды нужно сделать:

1. В Java-среде нужно расположить параметры для передачи в среду Fortran на прямом буфере. Этот прямой буфер передается в качестве параметра вспомогательным C-функциям, которые описаны в пункте 2. Так же, в качестве параметра передается смещение в буфере, по которому расположены передаваемые параметры.
2. На языке C реализовать встраиваемый через JNI в Java-окружение модуль. В этом модуле реализуются вспомогательные функции для каждой вызываемой Fortran-подпрограммы из Java окружения. Каждая такая вспомогательная функция вызывается из Java-программы. Одним из ее действий является непосредственный вызов Fortran-подпрограммы. Также вспомогательная функция выполняет передачу параметров из Java-окружения в среду Fortran, как это описано в пункте 2. То есть вспомогательная функция получает адрес буфера, вычисляет адреса параметров, зная смещения их расположения в буфере, и передает вычисленные адреса Fortran-подпрограмме.
3. На языке Java реализуется класс, который занимается записью и чтением данных из общей для Fortran-среды и Java-оболочки памяти. При этом при записи выполняется преобразование данных из формата языка Java в формат языка Fortran, а при чтении выполняется преобразование данных из формата языка Fortran в формат языка Java, как это описано в пункте 2.

7. Накладные расходы

В предложенной реализации накладные расходы возникают при вызове метода инициализации прямых Java-буферов, но эти накладные расходы возникают только один раз за все время работы программы, поэтому время, которое на них тратится, не существенно влияет на общую производительность программного продукта.

Накладные расходы возникают при преобразовании данных из формата языка Java в формат языка Fortran. Однако полное преобразование данных из одного формата в другой есть необходимость выполнять только дважды за работу всего приложения: в начале, после инициализации, и в конце, перед тем, как вывести окончательный результат работы приложения. Следовательно, эти накладные расходы тоже считаются разовыми и не существенно влияют на время выполнения программного продукта.

Однако возникают еще накладные расходы, когда данные обрабатываются не только в Fortran-подпрограммах, но и в основной программе, написанной на языке Java. В этом случае при каждом переключении есть необходимость преобразовать данные из одного формата в другой. Но, как правило, объем данных обрабатываемый сразу и в Java-коде и в коде, реализованном на языке Fortran, не очень велик. Следовательно, не следует преобразовать сразу все данные, которые рассчитываются в приложении, а нужно преобразовать только тот их фрагмент, который нужен для обработки. Такой подход позволит сократить накладные расходы на преобразование данных. Именно эти накладные расходы следует учитывать при оценке времени работы программного приложения.

8. Пример

Чтобы убедиться в корректности работы реализации была взята программа расчета динамики взрыва сверхновой звезды, реализованная на языке Fortran. [7]. Основная функция main, которая управляет расчетами, была переписана на язык Java. Остальные подпрограммы оставлены на языке Fortran.

Результаты работы исходной программы, реализованной только на языке Fortran, и программы, основная часть которой реализована на языке Java, а подпрограммы выполнены на языке Fortran, одинаковые.

Для сравнения времени работы полученного приложения, реализованного на языке Java с использованием Fortran-подпрограмм, было произведено сравнение с точно таким же приложением, но реализованным целиком на языке Fortran и на языке Java. Приложение можно представить в виде следующей схемы, представленной на Рис. 1.

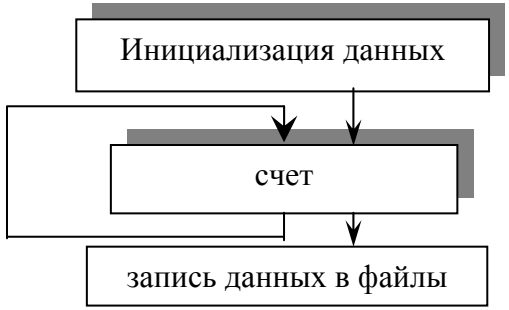

Рис.1. Схема приложения.

В приложении, реализованном на языках Java+Fortran, инициализация данных и запись данных в файлы выполняется в Java-окружении, а счет выполняется в Fortran-среде.

Для сравнения времени выполнения были выполнены замеры, как скорости работы всего программного приложения, так и отдельных его частей, в соответствии с Рис. 1. Замеры проводились на персональном компьютере. Размер оперативной памяти 512 MB, частота процессора 1700 MHz. Характеристики кэш-памяти процессора следующие:

CPU L1 Cache: 64K (64 byte/line), CPU L2 Cache: 526K (64 byte/line)

Сравнение времени работы представлено в таблице 3 и на Рис. 2.

	полное приложение (ms)	инициализация (ms)	счет (ms)	запись (ms)
Fortran	261559	42826	218450	283
Java + Fortran	266223	43540	221623	1060
Java	337225	69874	265727	1624

Таблица 3. Сравнительная производительность.

Как видно в таблице 4 и на Рис. 2 (а) реализация приложения на языках Java+Fortran не значительно проигрывает по времени выполнения приложению, реализованному только на языке Fortran. Это достигается за счет того, что в приложении, реализованном на Java+Fortran, вычисления полностью выполняют в Fortran-среде.

Однако приложение, реализованное на языках Java+Fortran, значительно быстрее работает, нежели приложение, реализованное на языке Java. Как видно на Рис. 2 (б), 2(в), 2(г) в приложении, реализованном только на языке Java, не только вычисление занимает больше времени, нежели в приложении, реализованном на языках Java+Fortran, но и инициализация и запись данных в файл. Потеря времени происходит за счет того, что большая часть инициируемых данных берется из файла. В приложении, реализованном

только на языке Java, данные из файла записываются в массивы языка Java, а в приложении, реализованном на языках Java+Fortran, - в прямые буферы.

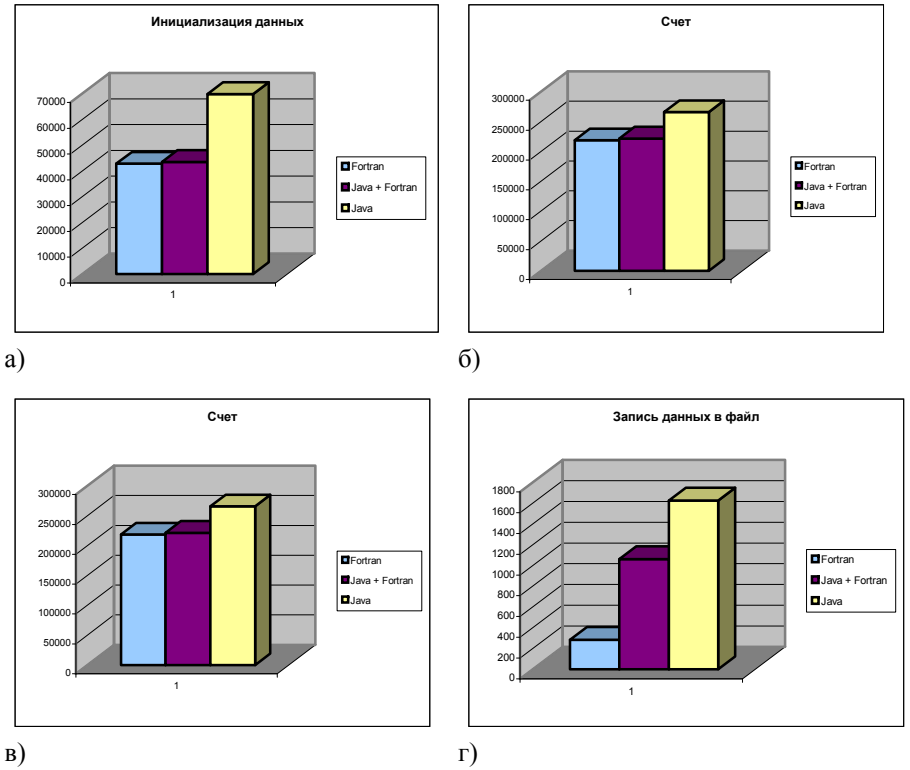

Рис. 2. Время выполнения.

9. Некоторые ограничения реализации приложения пользователя

Fortran-подпрограммы, которые вызываются из Java-среды, не должны содержать символа «подчеркивание» в своем имени. В противном случае разделяемая библиотека не сможет сопоставить реализованные в ней методы с теми, которые вызываются. Это вызовет падение работы всего приложения.

Компилятор GNU g77, разрешает использование переменных без их явного описания. Однако если явно не определить тип переменной в подпрограмме, которая вызывается из Java-окружения, вероятен случай, что виртуальная Java-машина получит внешний сигнал. Этот сигнал, номер которого 11, сообщает виртуальной машине о некорректном обращении к памяти за пределами ее работы. Аналогичная ситуация может возникнуть и с функциями. При

описании функций стандартом предусмотрено описывать явно тип возвращаемого значения. Однако если этого не сделать, то компилятор сам подберет соответствующий тип, исходя из типа возвращаемого выражения. Если такую функцию вызывать из программы, реализованной только на языке Fortran, то все стабильно будет работать. Но, как только объектный модуль с такой функцией участвует в формировании разделяемой библиотеки и подобного рода функция вызывается подпрограммой, которая в свою очередь вызывается виртуальной машиной Java, выполнение основной программы прекращается по причине получения виртуальной Java-машиной сигнала номер 11.

Данные ограничения в дальнейшем развитии работы будут сняты посредством автоматического добавления в код Fortran-программы недостающих описаний.

10. Заключение

Разработана организация взаимодействия среды Java и подпрограмм, реализованных на языке Fortran. Была выполнена прототипная реализация. Прототипная реализация показала, что описанная методика вызова подпрограмм, реализованных на языке Fortran, из окружения Java вызывается с минимальными накладными расходами, а, следовательно, эффективна.

Дальнейшее развитие предполагает разработку методики рефакторинга Fortran-программ с целью преобразования их в такой вид, какой было бы удобно автоматически транслировать на язык Java.

Литература

1. Б.Керниган, Д.Ритчи. Язык программирования Си. Санкт-Петербург, 2001
2. Фортран 77 ЕС ЭВМ. Справочное издание. Москва «Финансы и статистика», 1989
3. Фортран. Программированное учебное пособие. Киев «Вища школа», 1980
4. У.Савитч. Язык Java. Курс программирования. Москва – Санкт-Петербург – Киев «Вильямс», 2002
5. Ю.И. Рыжиков. Современный фортран. Санкт-Петербург «Корона принт», 2004
6. Артур Гриффитс. GCC. Полное руководство. Москва – Санкт-Петербург – Киев DiaSoft, 2004
7. С. Д. Устюгов, В. М. Чечеткин. Взрыв сверхновой при крупномасштабной конвективной неустойчивости вращающейся протонейтронной звезды. // Астрономический журнал, 1999, том 76, №11, с. 816-824.