ПРОГРАММНЫЙ КОМПЛЕКС КОНСТАНТНОГО ОБЕСПЕЧЕНИЯ GROUND. БИБЛИОТЕКА ПРОГРАММ IRA ДОСТУПА К МНОГОГРУППОВЫМ КОНСТАНТАМ ИЗ ФОРТРАН-ПРИЛОЖЕНИЙ

С. С. Касаткин, А. В. Алексеев, Н. А. Крутько

ФГУП «РФЯЦ-ВНИИЭФ», г. Саров Нижегородской обл.

Введение

В настоящее время в ИТМФ РФЯЦ-ВНИИЭФ введена в действие новая система константного обеспечения нейтронно-физических расчетов, реализованная в программном комплексе константного обеспечения GROUND и основанная на использовании СУБД Microsoft SQL Server [1]. При ее разработке ставились следующие цели:

1. Повышение надежности хранения данных константного обеспечения.
2. Обеспечение современного сервиса визуализации данных.
3. Повышение достоверности рассчитываемых групповых констант.
4. Предоставление открытого документированного SQL-интерфейса к данным для всех возможных в будущем приложений.

Использование такой технологии хранения и доступа к данным обеспечивает повышенный уровень надежности, достоверности и целостности данных, дает гарантию защиты от несанкционированного доступа и изменения критически важных данных.

Соответственно, основными концепциями разработки комплекса GROUND являются:

1. Хранение всех типов данных константного обеспечения в единой БД на основе реляционной СУБД (централизованное хранение и администрирование, невозможность частных коллекций).
2. Интеграция всех функций пользовательского доступа и визуализации данных в единой программной оболочке – клиенте БД (единообразный аппарат навигации и визуализации данных, возможность перекрестного сравнения данных).
3. Интеграция технологии расчета групповых констант в составе единой оболочки – клиента БД (новый пользовательский интерфейс, хранение в БД полного протокола параметров расчета).
4. Перевод счетных комплексов (Фортран) на потребление групповых данных из БД.

Укажем классы данных, охватываемые комплексом константного обеспечения GROUND:

1. Оцененные (спектральные) данные (формат ENDF/B).
2. Экспериментальные данные (архив EXFOR).
3. Библиотека данных по уровням возбуждения и гамма-переходам ENSDF.
4. Справочные таблицы ядерных масс и энергий реакций.
5. Данные экспериментов с критическими сборками (архив BeAr).
6. Групповые данные (нейтронные, гамма-кванты, заряженные частицы).

Предыдущая система константного обеспечения

Ранее константное обеспечение было представлено разнородными библиотеками, построенными по принципу файлового архива. Любой пользователь мог иметь у себя локальную копию любой из библиотек, проводить в ней любые модификации и использовать в расчетах.

Общая схема комплекса константного обеспечения, существовавшего до проекта GROUND, приведена на рис. 1.

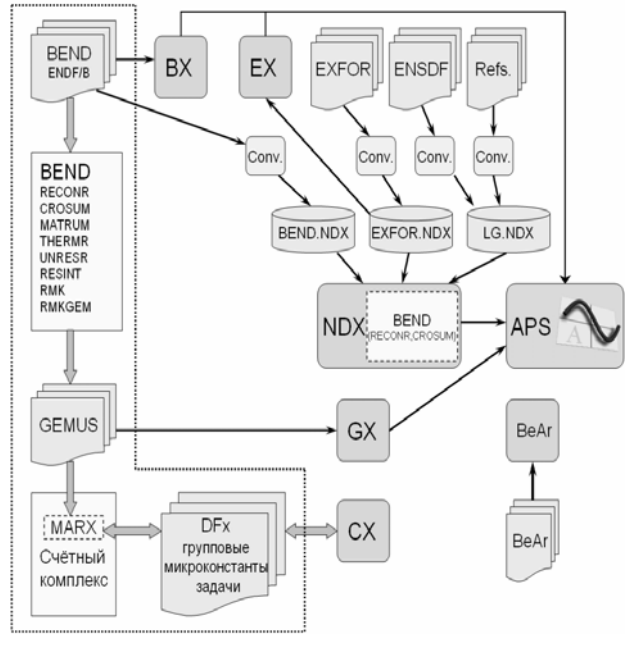

Рис. 1. Комплекс константного обеспечения до проекта GROUND

Пунктиром на этом рисунке выделен основной счетный цикл использования константного обеспечения в производственном счете. Как видно, каждый

тип данных хранится в соответствующем архиве (набор файлов на рис. 1), для каждого их которых разработана своя отдельная и изолированная программа визуализации и редактирования. Основным звеном здесь является информационно-справочная система NDX, в которой осуществляются просмотр и анализ нейтронно-физических констант.

GROUND − новая система константного обеспечения

Данная технология состоит из трех ключевых звеньев (рис. 2).

Первое звено – сама база данных GROUND, обеспечивает хранение, доступ и все виды контроля доступа и надежности хранения данных.

Вторым ключевым звеном данной технологии является графическая пользовательская оболочка GDE (**G**ROUND **D**atabase **E**nvironment), служащая для получения справочной информации, а также для проведения расчетов самих констант.

Третьим ключевым звеном является библиотека IRA (**I**nteroperability **R**ich **A**ssembly), обеспечивающая доступ производственным комплексам программ, в основном написанных на языке Fortran, к БД GROUND.

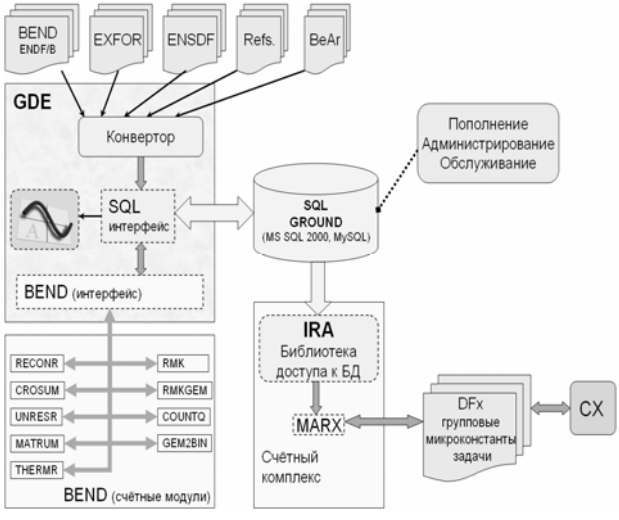

Рис. 2. Комплекс константного обеспечения GROUND

Библиотека программ IRA доступа к многогрупповым константам из Fortran-приложений

Основное внимание в данном докладе посвящено рассмотрению вопросов обеспечения взаимодействия Fortran-программ с серверами баз данных. Для лучшего представления о полноте и сложности решаемой задачи ее можно сформулировать следующим образом: обеспечить платформонезависимый механизм доступа Fortran-программы к СУБД, в частности к MS SQL Server 2000. Таким образом, искомое решение должно поддерживать как можно большее число СУБД, быть при этом реализованным на Fortran, кроссплатформенным и бесплатным.

Проведенное исследование готовых решений показало, что данная задача решена лишь частично, причем кроссплатформенное коммерческое решение поставляет фирма Oracle и только для одноименной СУБД. Также были найдены общие рекомендации по обеспечению доступа к СУБД MS SQL из ОС (Операционная Система) Windows, которые сводились к написанию Fortran интерфейса к практически не документированным функциям динамических библиотек, поставляемых с каждым отдельно взятым SQL сервером. Естественно, такие решения были просто неприемлемы, и в случае перехода на другую СУБД или ОС необходимо в лучшем случае заново переписывать интерфейсы и искать библиотеки, обеспечивающие доступ.

Таким образом, готовые решения данной задачи отсутствуют на Fortran, поэтому было принято решение расширить диапазон поиска на другие языки программирования, с которыми могут взаимодействовать Fortran-программы и которые являются платформонезависимыми и поддерживают возможности доступа к различным СУБД.

Наиболее подходящими с точки зрения наличия подробной документации и интегрированных сред разработки были выбраны Microsort. NetFramework (два и выше, работающая под Windows, Linux и MacOS) и язык программирования C# и Nokia Qt [2, 3] и C++. Для разработки была выбрана Nokia Qt 4.7.0, так как, начиная со стандарта Fortran 2003, вводится модуль ISO_C_BINDING [4, 5], служащий специально для целей облегчения взаимодействия между компонентами, реализованными на Fortran и C/C++.

После определения инструмента последовал выбор формы разработки посредника. Это либо статически подключаемая библиотека, внедряемая непосредственно в комплексы при сборке, либо динамическая библиотека, подключаемая в момент выполнения.

В качестве посредника выбрана именно динамическая библиотека. Такой выбор обусловлен обеспечением максимального качества работы производственных комплексов с библиотекой. Так, использование новой версии библиотеки не вызовет необходимости перекомпиляции комплекса, ее можно будет использовать «налету».

Библиотека IRA. Описание архитектуры и представления уровней данных

Основной целью библиотеки IRA является предоставление производственным комплексам многогрупповых микроконстант для их использования в дальнейшем счете. При этом сами константы хранятся в базе данных на MS SQL Server 2000 (рис. 3).

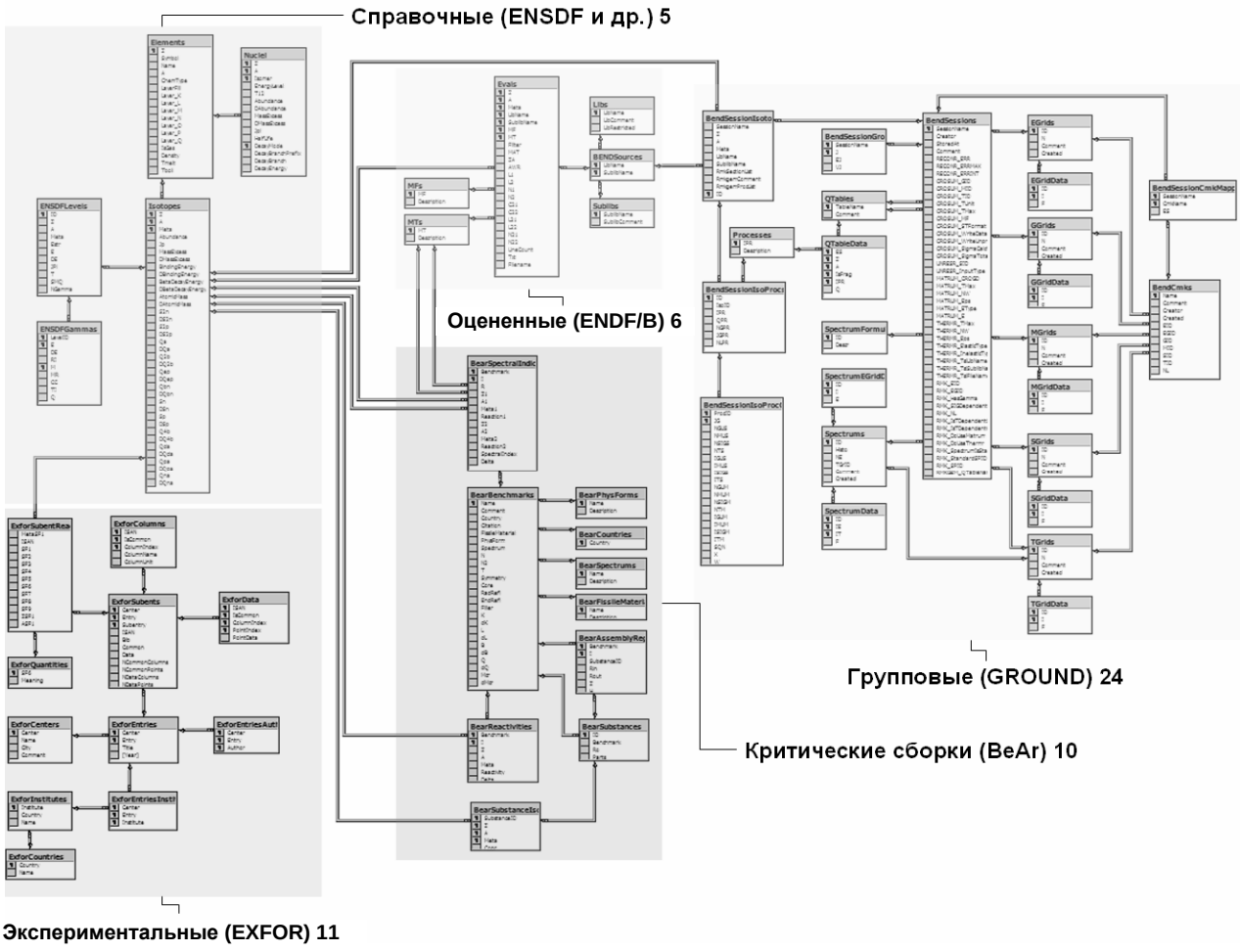

Рис. 3. Структура БД Ground (56 таблиц)

В настоящее время IRA работает только с групповыми данными. Логически все данные можно разделить на группы по уровню их детализации и удобства доступа. Эти уровни будем называть контекстами. Причем, контексты являются именно логическими, так как один логический контекст может формироваться из нескольких физических источников данных (таблиц). С данной точки зрения можно выделить следующие контексты:

1. DB (уровень базы данных).
2. CMK (система многогрупповых микроконстант).
3. ES (не более чем искусственное обозначение идентификатора расчета групповых микроконстант).
4. ISOTOPE (изотоп).
5. PROCESS (процесс).
6. GROUP (группа, на процессе).

Описанная иерархия приведена на рис. 4.

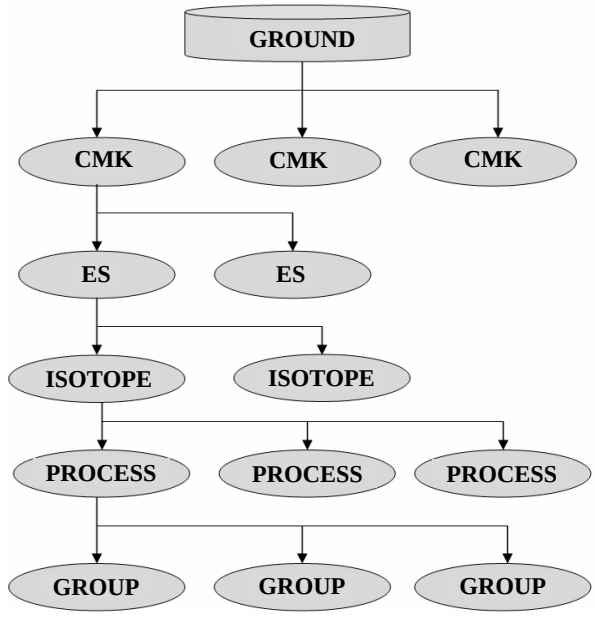

Рис. 4. Логическая структура контекстов групповых данных

Соответственно, в библиотеке IRA реализованы все необходимые функции для навигации по контекстам данных и получении самих данных. Так как библиотека разработана на C++, а программы-потребители − на Fortran, то для нее создан модуль, позволяющий клиентам взаимодействовать с ней. Однако одного интерфейса для обеспечения многоязыкового взаимодействия недостаточно, здесь начинается область вопросов совместимости языков программирования, форматов библиотек и их описаний для компиляции и сборки, модели памяти и форматы передачи аргументов в функции и т. д. Весь этот

спектр вопросов был рассмотрен при разработке IRA и будет рассмотрен далее.

Рассмотрим Fortran-интерфейс, предоставляемый пользователю для работы с IRA. Все функции и константы для работы с разработанной динамически подключаемой библиотекой сгруппированы, для удобства использования, в модуле IRA. Логически их можно разделить на два класса:

1. Функции установки нужного контекста.

2. Функции доступа к данным в текущем контексте.

Хотелось бы отметить, что контексты были организованны таким образом, чтобы максимально соответствовать характеристикам наборов групповых констант предшествующей системы константного обеспечения, реализованной в архиве GEMUS.

Для начала работы с библиотекой необходимо установить (или открыть) соединение с БД. Для этого предназначена функция IRA_CreateConnection. Устанавливать соединение обязательно, так как без него не будет доступа к БД, а соответственно и к данным. После вызова данной подпрограммы обязательно проверять код ошибки. Если он отличен от IRA_ERROR_SUCCESS (успешное выполнение операции), необходимо завершить работу с библиотекой и установить причину ошибки: отсутствие сетевого подключения, неправильно настроенный брандмауэр и т. д.

В случае успешного соединения с БД будет возвращен код IRA_ERROR_SUCCESS и с библиотекой можно работать дальше, используя функции установки необходимого контекста и далее выборки данных из него. После установления соединения текущим контекстом является DB (база данных). Из этого контекста можно получить общее количество СМК в базе данных и их список.

Для смены текущего контекста используются два вида функций:

1. IRA_Open<Контекст>, где <Контекст> один из: CMK, ES, ISOTOPE, PROCESS, GROUP. Применяется для установки контекста.

2. IRA_CloseContext − универсальная функция для закрытия контекстов.

Единственный контекст, который устанавливается и закрывается особым образом, − GROUND. Он устанавливается вызовом IRA_CreateConnection, а закрывается с помощью IRA_CloseConnection. После вызова IRA_CloseConnection соединение с БД разрывается и любые операции с ней становятся недоступны. Поэтому данный вызов должен быть последним, когда больше не будет обращений к БД.

Открыть закрытое соединение можно повторным вызовом IRA_CreateConnection.

Для доступа к данным определен родовой интерфейс IRA_Get со следующей сигнатурой (сигнатура записана на С-подобном псевдокоде):

IRA_Get(INTEGER(4) PARAMETER dataCode, – код запрашиваемых данных <DataType> value, – возвращаемое значение стандартного типа <DataType>, скаляр или массив INTEGER(4) errorCode – код завершения операции: успех или ошибка).

При передаче массивов необходимо следить, чтобы они были выделены, так как библиотека не проверяет соответствие типов данных и размерностей.

Таким образом, организован удобный интерфейс пользователя, который подробно задокументирован и использует стандарт Fortran 2003.

Общие вопросы взаимодействия Fortran и C/C++

Как было показано выше, для доступа к БД Ground, содержащей данные многогрупповых нейтронных микроконстант, на Qt была разработана библиотека IRA. Основными особенностями библиотеки являются:

1. Платформонезависимость.

2. Поддержка практически всех существующих СУБД.

За поддержание таких возможностей существует и «плата». Она заключается в использовании компилятора gcc. Это обеспечивает кроссплатформенность, так как компилятор является также кроссплатформенным и бесплатным, поэтому будет использоваться для компиляции на всех целевых ОС. Однако, формат самих выходных файлов (в Windows нотации .dll, .lib) несколько отличается от тех, что генерирует, скажем, cl (c/c++ компилятор от Microsoft).

Поэтому простого подключения данного .lib файла к Fortran-проекту для создания зависимости от целевой динамически подключаемой библиотеки недостаточно. Следовательно, необходимо создать такой .lib файл, который корректно будет потребляться Windows версией компилятора, на Linux данного этапа делать не требуется.

Итак, для обеспечения взаимодействия Fortran-приложения с динамической C/C++ библиотекой необходимы: корректный .lib файл, описания ее функций и Fortran-интерфейсы для функций, экспортируемых dll библиотекой.

Взаимодействие при помощи атрибутов

До принятия стандарта Fortran 2003 это можно было сделать всего одним способом – с помощью указания соответствующих атрибутов. Начиная со стандарта Fortran 2003, появляется возможность использовать встроенный модуль ISO_C_BINDING.

Для взаимодействия Fortran прогламм с C/C++ функциями необходимо в явном виде указать имя функции C/C++, которая будет использоваться с помощью атрибута ALIAS. При необходимости можно указать также атрибут DECORATE.

Хотелось бы также отметить, что для обнаружения экспортируемых DLL библиотекой функций при сборке необходимо указать lib файл с их описанием, иначе произойдет ошибка сборки и целевая программная единица не будет собрана.

После того, как имена внешних функций зафиксированы и найдены в библиотеке импорта, важным для рассмотрения вопросом является передача параметров. Существует два варианта передачи параметров во внешние процедуры: по значению и ссылке.

Здесь важную роль играет соглашение вызова внешних процедур, оно также зависит от компилятора и определяет порядок передачи аргументов в вызываемую процедуру/функцию.

Взаимодействие с использованием модуля ISO_C_BINDING

Для упрощения взаимодействия программ Fortran и C/C++ в стандарт Fortran 2003 был введен специальный модуль ISO_C_BINDING. В этом модуле содержатся:

1. Определения разновидностей типов Fortran, соответствующих встроенным типам C/C++.
2. Определения производных типов, позволяющих работать с указателями C/C++.
3. Определения символьных констант C/C++.
4. Встроенные функции для работы с указателями.

Ключевой особенностью данного модуля является полная стандартизация межъязыкового взаимодействия. С использованием данного модуля стало возможно выполнять различные операции с памятью. Например, передача из Fortran, ***неинициализированного указателя*** в C/C++ процедуру (через ее аргументы), выделение памяти в C/C++, работа с ней и передача уже инициализированного указателя в Fortran-программу, которая может работать непосредственно с данным указателем. Раньше такую операцию сделать было невозможно.

Кроме данного модуля, существует еще один важный оператор и атрибут: BIND, который позволяет определить, что объект взаимодействует с языком программирования C/C++. Особенностью применения данного атрибута является поддержка «области действия». То есть если атрибут применяется к имени функции, то все ее аргументы автоматически попадают в область действия данного атрибута. При этом если у каждого формального аргумента явно установить данный атрибут, то произойдет ошибка, так как данный атрибут уже установлен и его нельзя установить дважды.

Для всех процедур в интерфейсе IRA использованы атрибуты BIND, а для всех их аргументов использованы разновидности стандартных типов из модуля ISO_C_BINDING.

Модели памяти Fortran и C/C++

При программировании на нескольких языках важно знать, как обеспечивать взаимодействие компонентов друг с другом. Вопросы описания взаимодействия по вызовам и данным были рассмотрены в предыдущих двух пунктах. В данном пункте рассматриваются вопросы выполнения операций с памятью.

Важным различием языков программирования C/C++ и Fortran является модель памяти, которую они используют. В Fortran хранение данных в массивах производится «по столбцам», т. е. самый левый индекс меняется быстрее всего. В C/C++ хранение данных в массивах производится «по строкам», т. е. самый правый индекс меняется быстрее всего. Поэтому при работе с массивами, особенно для выходной информации, важно, чтобы C/C++ программы работали с моделью памяти Fortran или учитывали ее при формировании массивов выходных данных.

Рассмотрим хранение одно-, двух- и трехмерного массива в Fortran (рис. 5). Массивы такой же размерности в C/C++ хранятся следующим образом (рис. 6).

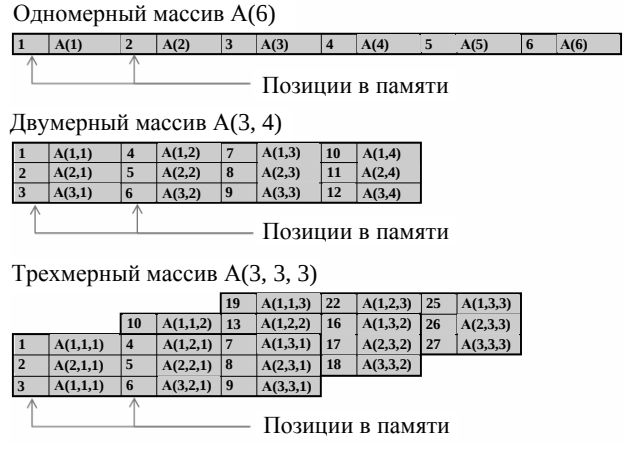

Рис. 5. Соответствие индексов элементов Fortran-массивов их линейным номерам в плоской модели памяти

Рис. 6. Соответствие индексов элементов C/C++ массивов их линейным номерам в плоской модели памяти

Еще одно различие в работе с массивами Fortran и C/C++ заключается в том, что в Fortran многомерные массивы представляют собой матрицы, все размерности которых по заданному направлению равны между собой. В C/C++ такого ограничения нет. То есть трехмерный массив может быть объявлен в самом общем виде: void*** pArray. Это объявление

трехмерного указателя, при этом необязательно, чтобы при выделении памяти все размерности массива по какому-либо выбранному направлению были равны между собой. То есть возможна ситуация выделения и работы со «Ступенчатым массивом». Двумерный случай показан на рис. 7.

Наконец, многомерные Fortran-массивы выделяются таким образом, что занимают непрерывный участок в памяти. Для C/C++ многомерных массивов такой гарантии нет. Более того, чтобы гарантировать непрерывность памяти, используемой многомерным C/C++ массивом, необходимо выделить ее специальным образом (как одномерный массив), а потом настроить многомерные указатели на соответствующие области памяти.

Двумерный массив mas

```
double **mas;
*mas = new double* [4];
for(int i = 0; i < 4; ++ i)
        mas [i] = new double [i + 1];
```

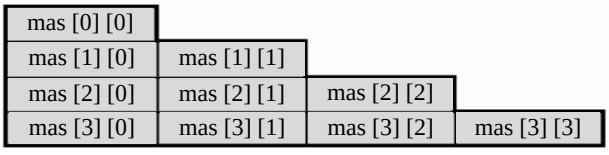

Рис. 7. Двумерный «ступенчатый» массив в C/C++

Таким образом, в C/C++ необходимо работать с многомерными Fortran-массивами как с занимающими непрерывную область памяти, при этом необходимо дополнительно вычислять смещение элемента в памяти по заданному индексу.

IRA учитывает все эти особенности при работе с Fortran-массивами в C/C++.

Заключение

Таким образом, была создана и аттестирована платформонезависимая динамически подключаемая библиотека IRA (Interoperability Rich Assembly) доступа к базе данных нейтронных констант Ground, основными характеристиками которой являются:

− Платформонезависимость.

− Поддержка практически всех существующих СУБД.

− Расширяемость.

− Параллельный доступ к данным.

− Самодокументированный программный интерфейс языка Фортран-90/2003.

− Включает примеры работы с библиотекой для C/C++ и Fortran, в т. ч. параллельную выборку с использованием стандарта OpenMP.

Литература

1. MSDN (Microsoft Software Development Network) http://msdn.com.

2. Шлее М. Qt 4,5. Профессиональное программирование на C++. СПб.: БХВ-Петербург, 2010.

3. Бланшет Ж, Саммерфилд М. Qt 4: программирование GUI на C++. М.: Кудиц-Пресс, 2007.

4. Intel® Visual Fortran Compiler Documentation http://www.intel.com/.

5. Немнюгин М. А., Стесик О. Л. Современный Фортран. Самоучитель. СПб.: БХВ-Петербург, 2004.