6. Артёмов, А. А. Теоретические основы информационного управления [Текст] / А. А. Артёмов // Информационные войны. – 2015. – № 3. – С. 83–97.
7. Цыганов, В. В. Глобальное информационное противоборство [Текст] / В. В. Цыганов // Информационные войны. – 2015. – № 2. – С. 7–13.
8. Малков, С. Ю. Модель устойчивости/дестабилизации политических систем [Текст] / С. Ю. Малков, С. Э. Билюга // Информационные войны. – 2015. – № 1. – С. 7–18.
9. Lundberg, J. The resilience analysis matrix (RAM): visualizing functional dependencies in complex socio-technical systems [Text] / J. Lundberg, W. Rogier // 5th symposium on resilience engineering managing trade-offs, 2013.
10. Oosthuizen, R. An analysis methodology for impact of new technology in complex sociotechnical systems [Text] / R. Oosthuizen, L. Pretorius // 2013 International Conference on Adaptive Science and Technology, 2013. doi: 10.1109/icastech.2013.6707508
11. Simmons, M. P. Memes Online: Extracted, Subtracted, Injected, and Recollected [Text] / M. P. Simmons, L. A. Adamic, E. Adar // ICWSM. – 2011. – Vol. 11. – P. 17–21.
12. IBM Security Services 2014. Cyber Security Intelligence Index [Electronic resource]. – Available at: http://media.scmagazine.com/documents/82/ibm_cyber_security_intelligenc_20450.pdf
13. Андрєєва, О. М. Кіберзброя та аналіз її деструктивної діяльності на прикладі впливу вірусу нового покоління STUXNET на іранську ядерну програму [Текст] / О. М. Андрєєва, К. Мусієнко // Actual problems of international relations. – 2014. – Т. 1, № 103.
14. Остапенко, Г. А. Информационные операции и атаки в социотехнических системах [Текст] / Г. А. Остапенко – М.: Горячая линия – Телеком, 2007. – 134 с.
15. Дудатьєв, А. В. Моделі для організації протидії інформаційним атакам [Текст] / А. В. Дудатьєв // Захист інформації. – 2015. – № 2. – С. 157–162.
16. Чистяков, В. П. Курс теории вероятностей [Текст] / В. П. Чистяков – М.: Наука,1987. – 240 с.

Розглянуті способи передачі даних між програмними одиницями при виконанні інженерних розрахунків технологічного обладнання засобами Фортрану 90 і більш високого рівня. Наведено приклади, які зустрічаються в інженерній практиці машинобудівельних та споріднених їй спеціальностей. Можливість застосування різних способів передачі даних між програмними одиницями дає можливість виконувати ефективні обчислення в наукових і інженерних розрахунках

Ключові слова: втрата точності, програмна одиниця, передача даних, Фортран, інженер-механік, інженерні розрахунки

Рассмотрены способы передачи данных между программными единицами при выполнении инженерных расчетов технологического оборудования средствами Фортрана 90 и более высокого уровня. Приведены примеры, которые встречаются в инженерной практике машиностроительных и родственных ей специальностей. Возможность применения различных способов передачи данных между программными единицами дает возможность выполнять эффективные вычисления в научных и инженерных расчетах

Ключевые слова: потеря точности, программная единица, передача данных, Фортран, инженер-механик, инженерные расчеты

УДК 004.2.007.2
DOI: 10.15587/1729-4061.2015.65475

АНАЛИЗ ПРИМЕНЕНИЯ СПОСОБОВ ПЕРЕДАЧИ ДАННЫХ МЕЖДУ ПРОГРАММНЫМИ ЕДИНИЦАМИ В ИНЖЕНЕРНЫХ РАСЧЕТАХ

Д. Э. Сидоров
Кандидат технических наук, доцент*
E-mail: dsts1@ukr.net
И. А. Казак
Кандидат педагогических наук, доцент*
E-mail: AsistentIA@meta.ua
*Кафедра химического, полимерного и силикатного машиностроения
Национальный технический университет Украины «Киевский политехнический институт»
пр. Победы, 37, г. Киев, Украина, 03056

1. Введение

В современное время актуальным вопросом для инженеров-механиков является выполнение инженерных расчетов разнообразного технологического оборудования. Такая потребность постоянно существует для оценки эффективности работы оборудования на основе расчетов параметрических, кинематических, прочностных, характеристик производительности, нагрузок и т. п. Ресурсоемкие инже-

нерные расчеты целесообразно выполнять с применением эффективных средств современного Фортрана, ориентированых именно на решение подобных задач. Безупречная работоспособность прикладных инженерных программных продуктов и точность вычислений напрямую зависят от выбора и правильного программного оформления способов передачи данных между программными единицами Фортран-программы.

2. Анализ литературных данных и постановка проблемы

Среди множества языков программирования, Фортрану следует отдать предпочтение в научно-технических и инженерных приложениях, как языку программирования, который предназначен для решения сложных вычислительных задач. Фортран – исторически первый язык программирования высокого уровня [1]. Несмотря на этот факт, он остается популярным и востребованным среди инженеров и научных работников. Разработчики программ на Фортране имеют не только современные средства программирования, но и получают доступ к огромной коллекции программного обеспечения, уже написанного на Фортране.

Созданные на Фортране математические, статистические, графические и другие библиотеки интенсивно используются в различных областях науки и техники. Если речь идет о моделировании сложных процессов или обработке больших объемов информации, где скорость вычислений является решающим фактором, то применение современого Фортрана более, чем оправдано [2].

Строгая стандартизация и постоянное обновление прикладного программного обеспечения Фортрана позволяют сделать его мощным и универсальным вычислительным средством. Компиляторы Фортрана коммерчески выпускаются ведущими производителями программного обеспечения. Среди них Compaq, Intel, Genias Software Gmh., Pacific-Sierra Research и др.

Темп обновления Фортрана может показаться небыстрым, однако это освобождает инженера от необходимости постоянно знакомиться с новыми версиями языка и дает возможность сосредоточиться на решении задач по своей предметной области [1]. В стандарты Фортрана включаются те достижения компьютерной науки, которые действительно необходимы и полезны при программировании вычислений, в том числе сложных инженерных расчетов, и это никоим образом не сказывается на их скорости.

Среди многих других современных языков программирования современный Фортран хорошо приспособлен для обучения студентов и начинающих инженеров [3]. Основы программирования на Фортране очень доступны для восприятия и достаточно широко представлены в научно-технической литературе [1–8]. Терминология по Фортрану изложена в работах [8–10]. Особенности способов передачи данных между программными единицами, в том числе и на современном Фортране, рассматриваются в научно-технических источниках достаточно широко [1, 2, 6–8, 11, 12].

Например, одни исследователи рассматривают прием передачи сообщений между отдельными процессами на примерах MPI-программ на языках С и различные варианты двунаправленных пересылок данных [11], другие – описывают коллективные операции передачи данных для высокопроизводительных вычислений на Фортране [12].

Кроме того, Фортран 90 в объектно-ориентированной концепции для расчетов химических процессов представлен в исследованиях [13]. В работе [14] рассмотрены вопросы параллельных вычислений в Фортране и Фортране MPI. В статье [15] представлен модульный подход к адаптации существующих научных кодов. В работе [16] описывается интерфейс между HEREZH ++, который вычисляет механическое поведение материала, и соединительным кодом, выполяющим процедуру обмена данными между ABAQUS и HEREZH ++. Применение методов расчета теплопередачи строительных конструкций рассматривается в статье [17]. Показано, что ограничения методологии могут приводить к неточным или ошибочным вычислениям, описаны источники, которые вносят ошибки. В работе [18] описывается система конечных элементов и обмен данными для анализа методом конечных элементов между различными решателями и сетками. Система поддерживает шесть коммерческих кодов, включая ABAQUS, ANSYS, MSC. Marc, Морфео, VULCAN. В статье [19] представлены идея и реализация объектно-ориентированного подхода к программированию вместе с эффективным кодированием в С++ для метода граничных элементов в двумерном анализе теплопередачи.

При разработке программного обеспечения прикладных отраслевых задач перед инженером-механиком всегда стоит проблема выбора способа передачи данных между программными единицами. Как видно из литературных данных, достачно широко освещены разнообразные вопросы реализации программного обеспечения для инженерных расчетов. Однако анализу применения базовых методов передачи данных в Фортран-программах до сих пор не уделено достаточного внимания.

3. Цель и задачи исследования

Целью данной работы является анализ применения способов передачи данных между программными единицами при выполнении инженерных расчетов технологического оборудования средствами современного Фортрана для расширения возможностей использования их инженерами-механиками в профессиональной деятельности.

Для достижения поставленной цели решались следующие задачи:

– рассмотреть особенности различных способов передачи данных между программными единицами при выполнении инженерных расчетов технологического оборудования средствами современного Фортрана;

– выполнить анализ и обосновать применение различных способов передачи данных между программными единицами в Фортран-программах.

4. Материалы исследований применения способов передачи данных между программными единицами в инженерных расчетах

4. 1. Способы передачи данных между программными единицами

Извесно, что сложная, объемная программа может быть разбита на отдельные фрагменты кода, которые называют программными единицами (процедурами). Процедуры можно располагать в отдельных файлах или собирать несколько процедур в одном файле. Программная единица может быть оттранслирована отдельно от других фрагментов программного кода.

В Фортране различают следующие виды программных единиц: основная программа, подпрограмма, функция или подпрограмма-функция, модуль, подпрограмма данных [1–5].

В Фортране также нашли широкое применение внутренние функции и подпрограммы, которые по сути нельзя в полной мере считать программными единицами, однако они имеют унифицированные формальные признаки с соответствующими программными единицами. Внутренние процедуры идентифицируются оператором *Contains* и имеют ограниченную область видимости.

Передавать данные между программными единицами можно несколькими способами: с помощью списков параметров, через общие блоки памяти, через файлы с общим доступом. Кроме того, доступ к данным можно обеспечивать путем расширения зоны видимости объектов программных единиц.

Передача данных через списки параметров – самый очевидный и простой способ по реализации и наиболее подходит для передачи данных небольшого количества объектов.

К примеру:

Subroutine Namesub (x1, x2, x3)
! Любое количество операторов
End Subroutine Namesub

Тут *NameSub* – имя подпрограммы, подчиняется правилам именования, но с ним не связано никакого фактического значения; *x1*, *x2*, *x3* – необязательные параметры, которые передаются в подпрограмму из вызывающей программной единицы, могут быть в ней изменены и возвращены в вызывающую программную единицу.

Подобная формальная запись программных единиц является алгоритмической абстракцией, а параметры *x1, x2, x3*, в этом контексте, являются формальными и не несут фактических значений.

Вызов подпрограммы из вызывающей программной единицы осуществляется оператором *Call*.

К примеру:

Call NameSub (A1, A2, A3)

В данном случае, параметры *A1, A2, A3* – фактические и несут конкретные значения (данные).

В этом случае, данные передаются не по имени, а по значению. При этом разработчик обязан соблюдать порядок следования, количество и соответствие типов для объектов передачи.

Передача данных через общие блоки памяти, как правило, реализуется при значительном количестве данных для взаимодействующих объектов. При этом, с помощью оператора *Common* обеспечивается доступ к общим блокам памяти из разных программных единиц. Особенностями таких блоков является то, что они не имеют внутренних границ, и каждая программная единица сама определяет количество, структуру и типы данных в них.

Таким образом, за организацию взаимодействия объектов программных единиц через общие блоки памяти несет ответственность разработчик программного кода. Компилятор не в состоянии проверить логику размещения данных в таких блоках.

К примеру:

Block Data CommDat; Dimension T(400), index(220)
*Common T; Data T /400*1.13/, index /220*4/*
End Block Data CommDat
Function Gamma(), Dimension G(160),S(240)
Common G,S
! Операторы...
End Function Gamma

В этом примере видно, что данные массива *T* в программной единице *CommDat* доступны двум объектам *G* и *S* в функции *Gamma*. Данные массива *index* программной единицы *CommDat* вообще остаются недоступны для функции *Gamma*.

Передача данных между программными единицами через файловый интерфейс в Фортран-программе может быть организована с помощью восьми операторов и одной функции: *Backspase*, *Rewind*, *EndFile*, *Open*, *Close*, *Inquire*, *Write*, *Read*, функции *EOF()*.

К примеру:

Subroutine Sub1()
! Тут идут вычисления C, D...
Write (12,*) C, D
End Subroutine Sub1
Function F(); Rewind (12); Read (12,) W, T*
! Использование полученных значений W, T ...
End Function F

Таким образом, подпрограмма *Sub1* помещает на устройство 12 (в файл) два значения переменных *C, D*. Функция *F*, будучи вызванной, считывает эти значения в переменные *W, T*. Оператор *Rewind* заблаговременно перемещает файловый указатель в начало первай строки устройства 12.

Три рассмотренных способа взаимодействия программных единиц при передаче данных содержат одну особенность: данные абсолютно анонимны и могут быть интерпретированы в программных единицах по-разному, в том числе и ошибочно. При этом, нет гарантий, что ошибочные действия будут обнаружены на этапе компиляции, или даже при построении выполнимого модуля.

Подобные ошибки исключены при использовании метода расширения зоны видимости объектов. Зона видимости объектов (переменных, массивов, др.) может быть расширена до глобальной, либо лишь на избранные программные единицы. В первом случае достаточно применение атрибута *Public*, а во втором – необходимо воспользоваться особенностями модулей: область видимости объектов, которые описаны в явном виде в модуле, можно расширить на программные единицы, в которых есть ссылки на соответствующий модуль через оператор *Use*.

К примеру:

Module Mod1 !Модуль для расширения области видимости переменных

```
Real (4) :: A, B, C; Integer (4) :: M / 132 /
End Modul Mod1
Program P16 ! Основная программа
Use Mod1 ! Доступ к переменным A, B, C, M по их именам
A = 16.2;
Do C=0.12, 2.33, .001
B = A * M*Fun()
! Другие операторы...
EndDo
End Program P16
Function Fun() ! Функция
Use Mod1 ! Доступ к переменным A, B, C, M по их именам
Fun=Sin(C)*Cos(C)
! Другие операторы...
End Function Fun
```

Оператор *Use* ассоциирует модуль *Mod1* в программных единицах, где он используется. Таким образом, для программных единиц *P16 и Fun* реализовано общее подпространство имен объектов {*A, B, C, M*}.

Модуль может содержать ссылки на другие модули, в этом случае поддерживается вся вложенность подпространств имен.

4. 2. Практическая реализация способов передачи данных между программными единицами при выполнении инженерных расчетов

Рассмотрим применение различных способов передачи данных между программными единицами при выполнении инженерных расчетов средствами современного Фортрана на примерах разработки программных единиц для технологического объекта.

Цилиндрический переходной элемент стального трубопровода для транспортировки метана длиной l= =0,1 м, диаметром a=0,1 м, толщиной стенки h=0,005 м находится под действием внутреннего избыточного давления метана $P = 2{,}2 \cdot 10^5$ Па. Определить перемещения W и нагрузки Q в зависимости от координаты x по длине трубы l. Расчет выполнить с шагом $\Delta x = 0{,}001$ м по формулам:

$$W = -\frac{Pl^4}{64D\alpha^4}\left(1 - 2c_1 \cdot sh - 2c_2 \cdot ch\right);$$

$$Q = -\frac{Pl}{2\alpha}\left[c_1\left(cs - cm\right) + c_2\left(cs + cm\right)\right];$$

$$sh = \sin\beta x \cdot sh\beta x; \quad ch = \cos\beta x \cdot ch\beta x;$$

$$cs = \cos\beta x \cdot sh\beta x; \quad cm = ch\beta x \cdot \sin\beta x;$$

$$c_1 = \frac{\cos\alpha sh\alpha + ch\alpha \sin\alpha}{\sin 2\alpha + sh2\alpha};$$

$$c_2 = \frac{\cos\alpha ch\alpha - ch\alpha \sin\alpha}{\sin 2\alpha + sh2\alpha};$$

$$\beta = \left[\frac{3\left(1-\mu^2\right)}{A^2h^2}\right]^{1/4}; \quad D = \frac{Eh^3}{12\left(1-\mu^2\right)}; \quad \alpha = \frac{\beta l}{2}.$$

Величины sh, ch, cs, cm – определить как функции. Для определения c_1, c_2, β, α – составить подпрограмму.

Коэффициент Пуассона $\mu = 0{,}3$. Модуль упругости стали $E = 2 \cdot 10^{11}$ Па.

Программный код реализации способа передачи данных между программными единицами через списки параметров представлен ниже.

```
Program Osnov !Основная программа
Implicit None
Real(4):: P, l, h, x ,Mu, E, D, A, dX, W, Q, c1, c2, B, ALF, CH, SH, CS, CM
Data P/2.2E5/, A/0.1/, l/0.1/, h/0.005/, Mu/0.3/, E/2.E11/, dX/.001/
D=E*h**3/12./(1.-Mu**2)
! Передача данных через списки параметров: вызов подпрограммы
CALL Sub1 (Mu, h, l, A, B, ALF, c1, c2) ! Список параметров – 8 объектов
Do x=0., l, dX
W=- P*l**4/64./D/ALF**4*(1.-2*c1*SH(B, x)-2*c2*CH(B, x))
Q=- P*l/2/ALF*(c1*(CS(B, x)-CM(B, x))+c2*(CS(B, x)+CM(B, x)))
Write (*,*)x, W, Q; EndDo; Stop; End Program Osnov

Subroutine Sub1(Mu, A, h,l, B, ALF, c1, c2) !Подпрограмма
Implicit None; Real(4)::Mu, A, h, B, l, ALF, c1, c2
B=(3*(1-Mu**2)/A**2/h**2)**(1./4.); ALF=B*l/2.
c1=(cos(ALF)*sinh(ALF)+cosh(ALF)*sin(ALF))/(sin(2*ALF)+sinh(2*ALF))
c2=(cos(ALF)*cosh(ALF)-cosh(ALF)*sin(ALF))/(sin(2*ALF)+sinh(2*ALF))
Return; End Subroutine Sub1

Real Function SH(B, x) ! Функция
Implicit None; Real(4):: B, x; SH=sin(B*x)*sinh(B*x)
Return; End Function SH

Real Function CH(B, x) ! Функция
Implicit None; Real(4):: B, x; CH=cos(B*x)*cosh(B*x)
Return; End Function CH

Real Function CS(B, x) ! Функция
Implicit None; Real(4):: B, x; CS=cos(B*x)*sinh(B*x)
Return; End Function CS

Real Function CM(B, x) ! Функция
Implicit None; real(4):: B, x; CM=cosh(B*x)*sin(B*x)
Return; End Function CM
```

Программный код реализации способа передачи данных между программными единицами через общие блоки памяти представлен ниже.

```
Program Osnov !Основная программа
Implicit None
Real(4):: P, l, h, x, Mu, E, D, A, dX, W, Q, c1, c2, B, ALF, CH, SH, CS, CM
Common B, Mu, A, h, l, ALF, c1, c2 ! Неименованый общий блок
Common /xx/ x ! Именованый общий блок
P=2.2E5; l=.1; h=.005; Mu=.3; E=2.E11; dX=.001; A=0.1
D=E*h**3/12./(1.-Mu**2)
```

```
! Передача данных через общие блоки: вызов подпрограммы
Call Sub1() ! Список параметров пустой
Do x=0., l, dX
W=- P*l**4/64./D/ALF**4*(1.-2*c1*SH()-2*c2*CH())
Q=- P*l/2./ALF*(c1*(CS()-CM())+c2*(CS()+CM()))
Write (*,*)x, W, Q; EndDo; Stop; End Program Osnov

Subroutine Sub1() !Подпрограмма
Implicit None; Real(4)::Mu, A, h, B, l, ALF, c1, c2
Common B, Mu, A, h, l, ALF, c1, c2 ! Доступ к общему блоку памяти
B=(3*(1-Mu**2)/A**2/h**2)**(1./4.); ALF=B*l/2.
c1=(cos(ALF)*  sinh(ALF)+cosh(ALF)*sin(ALF))/(sin(2*ALF)+sinh(2*ALF))
c2=(cos(ALF)*cosh(ALF)-cosh(ALF)*sin(ALF))/(sin(2*ALF)+sinh(2*ALF))
End Subroutine Sub1

Real Function SH() ! Функция. Список параметров пустой
Implicit None; Real(4):: B, x
Common /xx/x; Common B ! Доступ к общим блокам памяти
SH=sin(B*x)*sinh(B*x); End Function SH

Real Function CH() ! Функция. Список параметров пустой
Implicit None; Real(4):: B, x
Common /xx/x; Common B ! Доступ к общим блокам памяти
CH=cos(B*x)*cosh(B*x); End Function CH

Real Function CS() ! Функция. Список параметров пустой
Implicit None; Real(4):: B, x
Common /xx/x; Common B ! Доступ к общим блокам памяти
CS=cos(B*x)*sinh(B*x); End Function CS

Real Function CM()! Функция. Список параметров пустой
Implicit None; Real(4):: B, x
Common /xx/x; Common B ! Доступ к общим блокам памяти
CM=cosh(B*x)*sin(B*x); End Function CM
```

Следующий исходный код иллюстрирует применение способа расширения области видимости объектов.

```
Module Mod1 ! Модуль для переменных Mu, h, l, ALF, c1, c2
Real(4):: Mu/0.3/ ,h/0.005/, l/0.1/, ALF, c1, c2; End Module Mod1

Module Mod2 ! Модуль для переменной B
Real(4):: A/0.1/, B; End Module Mod2

Module Mod3 ! Модуль для переменной x
real(4):: x; End Module Mod3

Program Osnov !Основная программа
Use Mod1; Use Mod2; Use Mod3 ! Доступ до объектов модулей по именам
Implicit None; Real(4):: P, E, D, dX, W, Q, CH, SH, CS, CM
Data P/2.2E5/, E/2.E11/, dX/.001/
D=E*h**3/12./(1.-Mu**2)
CALL sub1()
Do x=0., l, dX
W=- P*l**4/64./D/ALF**4*(1.-2.*c1*SH()-2*c2*CH())
Q=- P*l/2./ALF*(c1*(CS()-CM())+c2*(CS()+CM()))
Write (*,*)x, W, Q; EndDo; Stop; End Program Osnov

Subroutine Sub1() ! Подпрограмма
Use Mod1 ! Доступ к переменным Mu, h,l, ALF,c1,c2
Use Mod2 ! Доступ к переменным A, B
B=(3*(1-Mu**2)/A**2/h**2)**(1./4.); ALF=B*l/2.
c1=(cos(ALF)*sinh(ALF)+cosh(ALF)*sin(ALF))/(sin(2*ALF)+sinh(2*ALF))
c2=(cos(ALF)*cosh(ALF)-cosh(ALF)*sin(ALF))/(sin(2*ALF)+sinh(2*ALF))
End Subroutine Sub1

Real Function SH()! Функция. Список параметров пустой
Use mod2; Use mod3 ! Доступ к переменным модулей по их именам
SH=sin(B*x)*sinh(B*x); End Function SH

Real Function CH() ! Функция. Список параметров пустой
Use mod2; Use mod3 ! Доступ к переменным модулей по их именам
CH=cos(B*x)*cosh(B*x); End Function CH

Real Function CS() ! Функция. Список параметров пустой
Use mod2; Use mod3 ! Доступ к переменным модулей по их именам
CS=cos(B*x)*sinh(B*x); End Function CS

Real Function CM() ! Функция. Список параметров пустой
Use mod2; Use mod3 ! Доступ к переменным модулей по их именам
CM=cosh(B*x)*sin(B*x); End Function CM
```

Применение способа передачи данных между программными единицами с помощью файлового интерфейса иллюстрирует приведенный ниже код.

```
Program Osnov !Основная программа
Implicit None
Real(4):: P, l, h, x ,Mu, E, D, A, dX, W, Q, c1, c2, ALF, CM, CS, SH, CH
Open(1, File='data.txt', Status='Old') ! Исходные данные
Open(2, File='B.txt') ! Значение B
Open(3, File='x.txt') ! Значение x
Open(4, File='out.txt') ! Результаты счета B, ALF, c1, c2
Open(5, File='res.txt') ! Результаты счета x, W, Q
Read(1,*) ! Первая строка в файле data.txt не содержит данных
Read(1,*) A, Mu, l ,h, P, E, dX ! Читаем из файла заданные значения
D=E*h**3./12./(1.-Mu**2.); CALL Sub1(); Close(1)
Rewind (4); Read (4,*) ALF, c1, c2; Close (4) ! Получим значения из файла
```

```
Do x=0., l, dX
Rewind (3); Write (3,*) x !Пишем текущее значение x
W=- P*l**4/64./D/ALF**4*(1.-2*c1*SH()-2*c2*CH())
Q=- P*l/2./ALF*(c1*(CS()-CM())+c2*(CS()+CM()))
Write (*,*)x, W, Q ! Результаты на екран
Write (5,*)x, W, Q ! Результаты в файл
EndDo
Close(2); Close(3); Close(5); Stop; End Program Osnov

Subroutine Sub1()! Подпрограмма
Implicit None; Real(4) :: Mu, A, h, B, l, ALF,c1,c2
Rewind (1); Read(1,*;) Read(1,*)A, Mu, l ,h ! Читаем данные для расчета
B=(3*(1-Mu**2)/A**2/h**2)**(1./4.)
Rewind (2); Write (2,*) B ! Записываем значение в файл
ALF=B*l/2.
c1=(cos(ALF)*  sinh(ALF)+cosh(ALF)*sin(ALF))/(sin(2*ALF)+sinh(2*ALF))
c2=(cos(ALF)*cosh(ALF)-cosh(ALF)*sin(ALF))/(sin(2*ALF)+sinh(2*ALF))
Rewind (4); Write(4,*) ALF, c1, c2 ! Записываем результат в файл
End Subroutine Sub1

Real Function SH()! Функция. Список параметров пустой
Implicit None; Real(4):: B, x
Rewind (2); Read(2,*) B ! Читаем B
Rewind (3); Read(3,*) x ! Читаем x
SH=sin(B*x)*sinh(B*x); End Function SH

Real Function CH()! Функция. Список параметров пустой
Implicit None; Real(4):: B, x
Rewind (2); Read(2,*) B ! Читаем B
Rewind (3); Read(3,*) x ! Читаем x
CH=cos(B*x)*cosh(B*x); End Function CH

Real Function CS()! Функция. Список параметров пустой
Implicit None; Real(4):: B, x
Rewind (2); Read(2,*) B ! Читаем B
Rewind (3); Read(3,*) x ! Читаем x
CS=cos(B*x)*sinh(B*x); End Function CS

Real Function CM() ! Функция. Список параметров пустой
Implicit None; Real(4):: B, x
Rewind (2); Read(2,*) B ! Читаем B
Rewind (3); Read(3,*) x ! Читаем x
CM=cosh(B*x)*sin(B*x); End Function CM
```

На рис. 1 приведена графическая интерпретация результатов решения задачи.

Как видно из приведенного выше программного кода, вычислительная часть всех реализаций абсолютно одинаковая. Модификации подвергается только интерфейсная часть программного кода, ответственная за передачу данных между программными единицами.

Представленный код компилировался средствами Microsoft Fortran PowerStation V 4.0, Digital Visual Fortran V 6.0, Silverfrost FTN95 для некоммерческого тестирования.

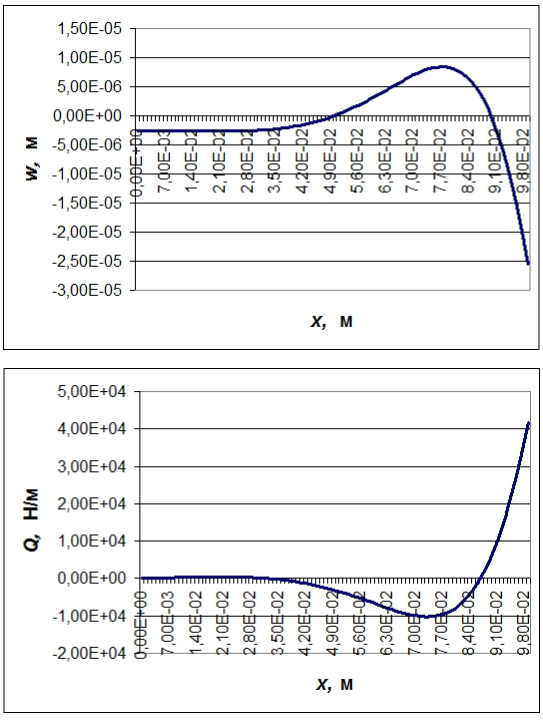

Рис. 1. Результаты счета к задаче об определении перемещения *W* и нагрузки Q по длине l переходного элемента стального трубопровода

5. Обсуждение результатов реализации способов передачи данных между программными единицами при выполнении инженерных расчетов

Несмотря на одинаковый вычислительный алгоритм, реализованный в приведенных выше примерах, можно отметить ряд отличий между ними. Например, медленнее всего компилируется модульная программа. Это обусловлено работой препроцессора, который запускается при наличии модулей. А медленнее всего выполняется программа с реализацией файлового интерфейса для передачи данных между программными единицами. На основании анализа приведенных примеров, а также практики работы с компиляторами Фортрана, относительное сравнение способов передачи данных между программными единицами выполнено в виде таблицы (табл. 1).

Применение файлового интерфейса ведет к потере точности, которая обусловлена усечением вывода на устройство до 7–8 значащих цифр (для типов данных обычной точности) и зависит от аппаратно-программного обеспечения ПЭВМ. Такой эффект иллюстрирует скриншот фрагмента результатов счета для модульной программы и варианта передачи данных через файловый интерфейс (рис. 2). Зато, если при использовании файлового интерфейса устройством вывода является физический носитель, то сохранность данных гарантирована даже в случае аварийного завершения работы программы или ПЭВМ.

Таблица1

Сравнение способов передачи данных

Обозначение	Списки параметров	Общие блоки	Модули	Файловый интерфейс
Скорость компиляции	Высокая	Средняя	Низкая	Средняя
Быстрота работы программного кода	Высокая	Высокая	Средняя	Низкая
Удобство программирования	Высокое	Среднее	Высокое	Низкое
Количество объектов передачи	Среднее	Среднее	Большое	Малое
Обработка массивов данных	Малых	Больших	Любых	Средних
Ограничение точности результата	по разрядной сетке	по разрядной сетке	по разрядной сетке	ограничение количества значащих цифр

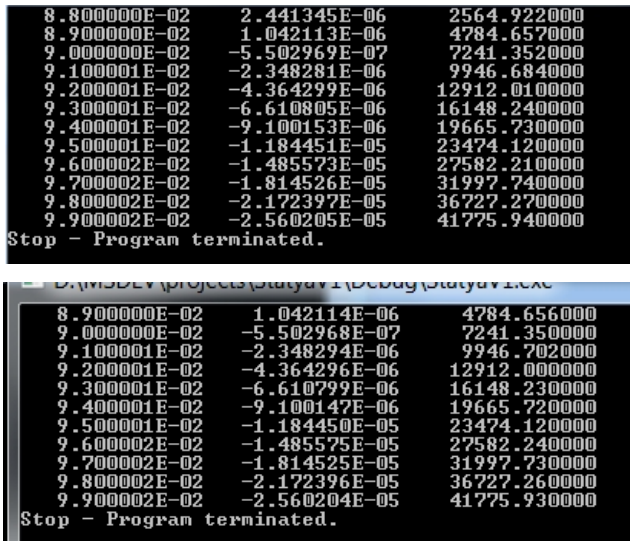

Рис. 2. Фрагменты результатов счета модульной программы (сверху) и программы реализации файлового интерфейса (снизу)

Программы с общими блоками памяти целесообразно разрабатывать при необходимости обработки большого количества данных в массивах. При этом следует учитывать, что поток данных, вероятно, будет направлен в современную программу все же не через оператор *Block Data*, а через файловый интерфейс.

Классическое использование списков параметров для передачи данных между программными единицами все еще широко применяется для программных продуктов «средней тяжести», где также удобно использовать и более современный инструмент – модули.

Следует заметить, что потеря точности для полученных результатов возможна лишь при использовании файлового интерфейса. Точность результатов вычислений при передаче данных между программными единицами и через списки параметров, и с помощью общих блоков памяти, и при расширении зоны видимости объектов определяется лишь разрядной сеткой назначенных типов данных для объектов программы. Как показывают многократные результаты тестирования подобных программ и приведенных выше примеров, результаты счета абсолютно совпадают при переходе от одного из этих способов передачи данных к другому, а также при их комбинации.

7. Выводы

В результате проведенных исследований:

1. Рассмотрены вопросы типизации программных единиц, особенности их синтаксиса применительно к современному Фортрану, а также существующие базовые способы передачи данных между программными единицами. Выполнено сравнение способов передачи данных между программными единицами по удобству программной реализации, скорости работы и ограничению точности результатов. Сравнение показало, что метод расширения зоны видимости объектов является наиболее приемлемым при программировании инженерных задач. Он обладает единственным недостатком – низкой скоростью компилирования, но этот недостаток ложится на плечи разработчика, а не пользователя программных продуктов.

2. Базовые способы передачи данных между программными единицами реализованы на примерах расчета технологического оборудования химического машиностроения. Проведен анализ применения различных способов передачи данных между программными единицами в Фортран-программах (через списки параметров, через общие блоки, через модули и с помощью файлового интерфейса). Выявлено, что при использовании файлового интерфейса для передачи данных между программными единицами возможна потеря точности для результатов счета. Иные способы передачи данных гарантируют получение результатов в пределах точности разрядной сетки для используемых типов данных.

Современное прикладное программное обеспечение, которое разрабатывает инженер-механик химического машиностроения, зачастую требует применения одновременно нескольких способов передачи данных между программными единицами одного проекта. Детальное рассмотрение различных способов и анализ их применения, а также прикладные примеры, приведенные в статье, дают возможность оценить трудности и преимущества реализации расчетного алгоритма при наложении на него различных условий передачи данных между программными единицами.

Практические задачи, которые приходится решать инженеру-механику в области химического машиностроения путем непосредственного программирования, как правило, несколько более сложные и ресурсоемкие, чем пример, приведенный в статье. Однако, если исключить очень специфические программные разработки расчетных систем и комплексов, то эти задачи можно отнести к программным продуктам не более, чем «средней тяжести». Таким образом, применение модульных принципов построения интерфейса между программными единицами представляется оптимальным компромиссом между удобством программирования и эффективностью выполнения прикладной программы, при гарантии получения вполне точных результатов.

Литература

1. Немнюгин, М. А. Современный Фортран. Самоучитель [Текст]: учебик / М. А. Немнюгин, О. Л. Стесик. – Санкт-Петербург: БХВ-Петербург, 2004. – 496 с.
2. Бартеньев, О. В. Современный Фортран [Текст] / О. В. Бартеньев. – Санкт-Петербург: БХВ-Петербург, 2004. – 390 с.
3. Рыжков, Ю. Программирование на Фортране PowerStation для инженеров [Текст] / Ю. Рыжков. – Санкт-Петербург: Корона принт, 1999 – 159 с.
4. Брич, З. С. Фортран 77 для ПЭВМ ЕС [Текст]: справ. пос. / З. С. Брич, Д. В. Капилевич, Н. А. Клевцова. – М.: Финансы и статистика, 1991 – 285 с.
5. Уорд, Т. Фортран и искусство программирования персональных ЭВМ [Текст] / Т. Уорд, Э. Бромхед; пер. с англ. – М.: Радио и связь, 1993. – 352 с.
6. Андреева, Е. Н. Энциклопедия учителя информатики. Выпуск 5 [Электронный ресурс] / Е. Н. Андреева, И. Н. Фалина // Журнал «Информатика». – 2007. – № 15. – Режим доступа: http://inf.1september.ru/article.php?ID=200701504/ – Загл. с экрана.
7. Акимова, Е. Н. Основы программирования на языке Фортран [Текст]: учеб. пос. / Е. Н. Акимова. – Екатеринбург: УВФУ, 2015. – 90 с.
8. Справочное руководство по языку Fortran 95 [Электронный ресурс]. – Режим доступа: http://www.math.spbu.ru/user/rus/cluster/Doc/Library/fortran95/langref/langr_oglav.shtml – Загл. с экрана.
9. Горелик, А. М. Словарь терминов языка Фортран 95 [Электронный ресурс] / А. М. Горелик. – Режим доступа: http://www.parallel.ru/tech/tech_dev/terms.html/ – Загл. с экрана.
10. Энциклопедия по машиностроению XXL. Оборудование, материаловедение, механика и ... [Электронный ресурс]. – Режим доступа: http://mash-xxl.info/info/106660/ – Загл. с экрана.
11. Антонов, А. С. Введение в параллельные вычисления [Текст]: метод. пос. / А. С. Антонов. – М: НИВЦ МГУ, 2002. – 69 с.
12. Баденко, В. Л. Высокопроизводительные вычисления [Текст]: учеб. пос. / В. Л. Баденко. – СПб: Изд-во Политехн. ун-та, 2010. – 180 с.
13. Bea, S. A. CHEPROO: A Fortran 90 object-oriented module to solve chemical processes in Earth Science models [Text] / S. A. Bea, J. Carrera, C. Ayora, F. Batlle, M. W. Saaltink // Computers & Geosciences. – 2009. – Vol. 35, Issue 6. – P. 1098-1112. doi: 10.1016/j.cageo.2008.08.010
14. Berg, B. A. Fortran code for SU(3) lattice gauge theory with and without MPI checkerboard parallelization [Text] / B. A. Berg, H. Wu // Computer Physics Communications. – 2012. – Vol. 183, Issue 10. – P. 2145–2157. doi: 10.1016/j.cpc.2012.03.021
15. Sewell, P. Implementing modular adaptation of scientific software [Text] / P. Sewell, S. Noroozi, J. Vinney, R. Amali, S. Andrews // Engineering Applications of Artificial Intelligence. – 2010. – Vol. 23, Issue 6. – P. 1000–1011.
16. Rio, G. Asynchronous interface between a finite element commercial software ABAQUS and an academic research code HEREZH++ [Text] / G. Rio, H. Laurent, G. Bl s Andrews // Advances in Engineering Software. – 2008. – Vol. 39, Issue 12. – P. 1010–1022. doi: 10.1016/j.advengsoft.2008.01.004
17. Li, X. Q. Applicability of calculation methods for conduction transfer function of building constructions[Text] / X. Q. Li, Y. Chen, J. D. Spitler, D. Fisher // International Journal of Thermal Sciences. – 2009. – Vol. 48, Issue 7. – P. 1441–1451. doi: 10.1016/j.ijthermalsci.2008.11.006
18. Afazov, S. M. Development of a Finite Element Data Exchange System for chain simulation of manufacturing processes [Text] / S. M. Afazov, A. A. Becker, T. H. Hyde // Advances in Engineering Software. – 2012. – Vol. 47, Issue 1. – P. 104–113. doi: 10.1016/j.advengsoft.2011.12.011
19. Qiao, H. Object-oriented programming for the boundary element method in two-dimensional heat transfer analysis [Text] / H. Qiao // Advances in Engineering Software. – 2006. – Vol. 37, Issue 4. – P. 248–259. doi: 10.1016/j.advengsoft.2011.12.011