

# Программирование на языке FORTRAN для молекулярного моделирования

*Игнатов С.К.*

*Спецкурс для студентов 4 курса х/ф д/о*

*Нижний Новгород 2016*

# Лекция 1

Основные характеристики и выражения.

Операторы ввода-вывода

# Fortran, FORTRAN

*(Formula Translator, первая версия 1957 г.)*

## Преимущества:

- ориентирован на математические вычисления
- многие математические объекты (векторы, матрицы) являются частью языка
- высокая производительность
- много готовых приложений и библиотек

## Недостатки:

- слабая интеграция с ОС
- слабая поддержка оконных приложений (ориентация на консольные приложения)
- много устаревших вариантов выражений

# Fortran. Компиляторы и стандарты

**Компилятор** – программа для перевода программа на языке высокого уровня в машинные коды

Обычно компиляция состоит из двух фаз:

- Собственно компиляция (Prog.f90 → Prog.obj - т.н. объектный файл – программа в машинных кодах без системных подпрограмм и настроенных адресов переходов между подпрограммами)
- Сборка (построение, линковка) – добавление в объектный файл системных подпрограмм и настройка адресов. Выполняется программой-линкером (Prog.obj → Prog.exe, получается исполняемый файл)

Несколько известных компиляторов:

**Digital Visual Fortran (Windows)**

**Intel Fortran Composer v.19 for Windows + Developer Studio**

Intel Fortran for Linux (free)

gfortran (linux, free)

g95 (linux)

Portland Group Fortran (pgf, linux, commercial)

Несколько стандартов:

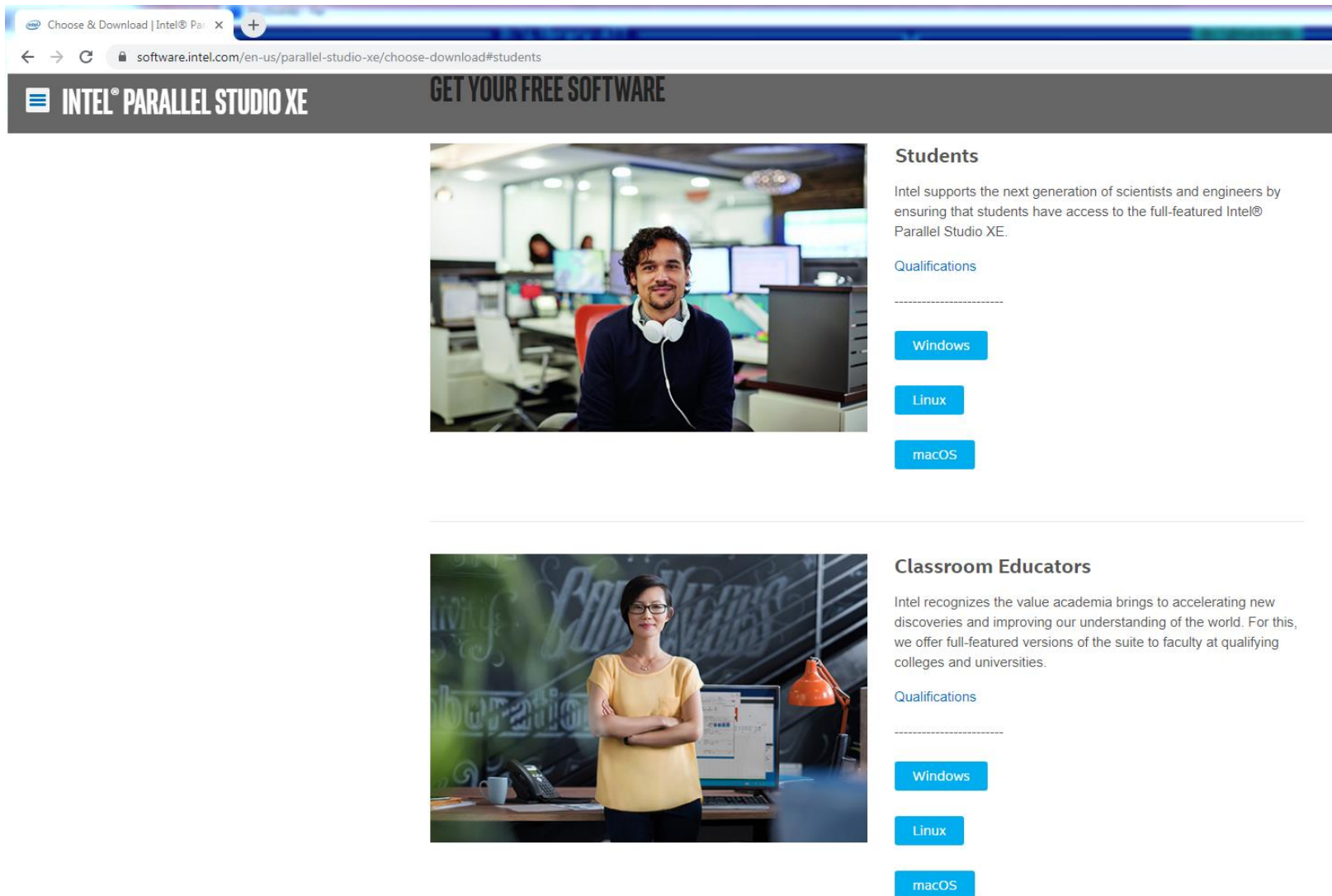
F66, F77 (устарели: fixed format, \*.f, \*.for)

**F90 (современный стиль, \*.f90)**

F95, F2008 (внедрены современные парадигмы программирования, поддержка параллельности)

...фортран скачать бесплатно без регистрации без смс...

## Сайт Интел:



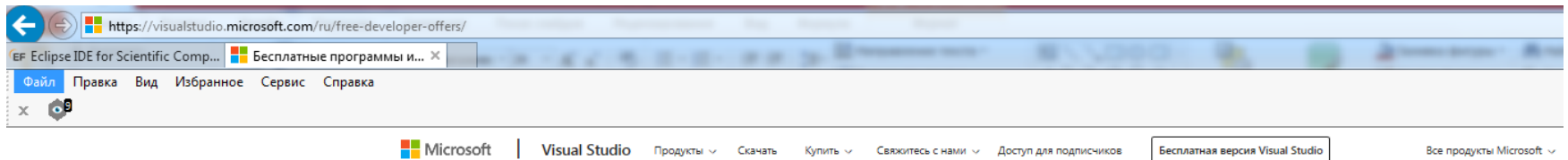
The screenshot shows the Intel Parallel Studio XE website. The browser address bar displays the URL: `software.intel.com/en-us/parallel-studio-xe/choose-download#students`. The page header includes the Intel logo, the text "Choose & Download | Intel® Parallel Studio XE", and a navigation menu with "INTEL® PARALLEL STUDIO XE" and "GET YOUR FREE SOFTWARE".

The main content area is divided into two sections:




- Students**  
Intel supports the next generation of scientists and engineers by ensuring that students have access to the full-featured Intel® Parallel Studio XE.  
Qualifications  
-----  
[Windows](#)  
[Linux](#)  
[macOS](#)
- Classroom Educators**  
Intel recognizes the value academia brings to accelerating new discoveries and improving our understanding of the world. For this, we offer full-featured versions of the suite to faculty at qualifying colleges and universities.  
Qualifications  
-----  
[Windows](#)  
[Linux](#)  
[macOS](#)

# IDE – Integrated Development Environment

## -- Microsoft Visual Studio (2010+)



Все, что потребуется для создания отличных приложений.  
бесплатно.

<p><b>Visual Studio Community</b></p> <p>Полнофункциональная интегрированная среда разработки для создания некорпоративных приложений для Windows, Android и iOS, а также современных веб-приложений и облачных служб.</p> 	<p><b>Azure DevOps (ранее VSTS)</b></p> <p>Инструменты Agile, Git и непрерывная интеграция для любого языка и любой операционной системы.</p> 	<p><b>Visual Studio Code</b></p> <p>Мощный бесплатный редактор на основе исходного кода, работающий на любой платформе.</p> 
<a href="#">Загрузка</a>	<a href="#">Бесплатная учетная запись</a>	<a href="#">Скачать</a>

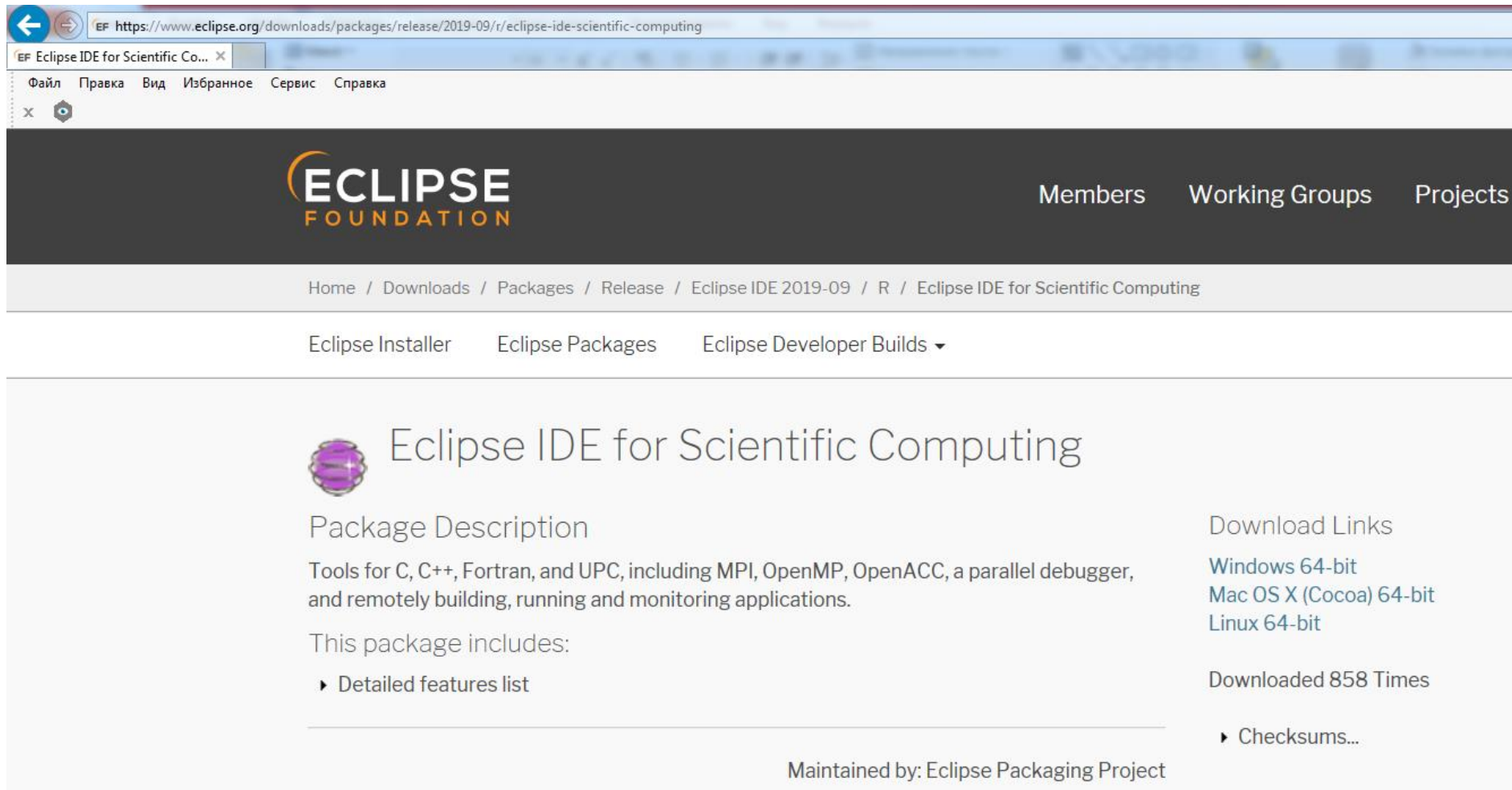
Visual Studio Dev Essentials — все вышеописанное и многое другое  
Получите все эти бесплатные средства, а также обучение Pluralsight, кредит Azure и файлы для загрузки — все абсолютно бесплатно.

[Присоединитесь сейчас](#)



# IDE – Integrated Development Environment

## -- ECLIPSE



The screenshot shows a web browser window with the URL <https://www.eclipse.org/downloads/packages/release/2019-09/r/eclipse-ide-scientific-computing>. The browser's address bar and tabs are visible. The page header features the Eclipse Foundation logo and navigation links for Members, Working Groups, and Projects. A breadcrumb trail indicates the current location: Home / Downloads / Packages / Release / Eclipse IDE 2019-09 / R / Eclipse IDE for Scientific Computing. Below this, there are links for Eclipse Installer, Eclipse Packages, and Eclipse Developer Builds. The main content area is titled "Eclipse IDE for Scientific Computing" and includes a package description, download links for Windows 64-bit, Mac OS X (Cocoa) 64-bit, and Linux 64-bit, and a section for detailed features list. The page also shows that the package has been downloaded 858 times and is maintained by the Eclipse Packaging Project.

EF <https://www.eclipse.org/downloads/packages/release/2019-09/r/eclipse-ide-scientific-computing>

Eclipse IDE for Scientific Co... x

Файл Правка Вид Избранное Сервис Справка

ECLIPSE FOUNDATION

Members Working Groups Projects

Home / Downloads / Packages / Release / Eclipse IDE 2019-09 / R / Eclipse IDE for Scientific Computing

Eclipse Installer Eclipse Packages Eclipse Developer Builds ▾

### Eclipse IDE for Scientific Computing

**Package Description**

Tools for C, C++, Fortran, and UPC, including MPI, OpenMP, OpenACC, a parallel debugger, and remotely building, running and monitoring applications.

This package includes:

- Detailed features list

**Download Links**

- [Windows 64-bit](#)
- [Mac OS X \(Cocoa\) 64-bit](#)
- [Linux 64-bit](#)

Downloaded 858 Times

- Checksums...

Maintained by: Eclipse Packaging Project

# Литература (находится в интернет)

Бартеньев О.В. Современный фортран.

Metcalf M., Reid J. Fortran 90, 95 explained. 1999

Numerical recipes in Fortran77

Numerical recipes in Fortran90

Using GNU fortran

Intel® Fortran Compiler User and Reference Guides

Using Intel® Visual Fortran to Create and Build Windows\*-Based Applications

Звягин В.Ф., Федоров С.В. Параллельные вычисления в оптике (Фортран)

Balint Aradi – Учебный курс  
Bremen University  
«Fortran2003 для научных  
расчетов»  
<https://www.bccms.uni-bremen.de/en/cms/people/b-aradi/>



Computational  
Materials Science Group



Bremen  
Center for  
Computational Materials Science

BCCMS / Groups / CMS / People / B. Aradi

English / Deutsch

- About CMS
- Research
- Teaching
- People
  - B. Aradi
    - Wissen. Progr.
    - Gruppentheorie
  - Open positions
  - Contact



- Prof. Fraunheim  
CMS
- Prof. Colombi Ciacchi  
HMI
- Prof. Ploshikhin  
ISEMP
- Prof. Wehling  
ECN



## Dr. Bálint Aradi



Phone:	+49 421 218 62331
Email:	<a href="mailto:aradi@uni-bremen.de">aradi@uni-bremen.de</a>
GPG-key:	<a href="#">aradi.asc</a> (@ pool.sks-keyservers.net)
Room:	3.14 (TAB-Building, Entry A)

### Teaching

**Group theory**  
Course [2014], Lecture notes

**Scientific programming (Fortran 2003 or Python)**  
Course description  
Teaching material (Fortan)  
Teaching material (Python)

**Computer & Software I (Mapple)**  
Teaching material

### Development



### Research

Investigation of point defects in bulk semiconductors.

Multiscale quantum mechanical modelling of semiconductor nanowires, combining *ab initio* and tight binding approaches.

For publication list, see [profile on Google Scholar](#).



# Формат файлов F77 (\*.f, \*.for) – устарел, но часто встречается в известных программах

```
123456789012345678901234567890123456789012345678901234567890123456789012345678
c      subroutine angfrc
c      x      (lsolva,lfree,lexcite,idnode,imcon,mxnode,ntangl,engang,virang)
c
c      subroutine for calculating bond angle energy and
c      force terms in molecular dynamics.
c
c      implicit none
c
c      logical safe,lsolva,lfree,lexcite,lselect
c      integer idnode,mxnode,imcon,ntangl,fail1,fail2
c      integer ii,iang1,iang2,i,ia,ib,ic,kk,keya
c      real(8) engang,virang,theta,fxc,fyc,fzc,rab,xab
c      real(8) strs(6)
c      define angular potential function and derivative
c      using the parameters in array prmang
c
c      safe=.true.
c
c      check size of work arrays
c
c      if((ntangl-mxnode+1)/mxnode.gt.msbad)call error(idnode,419)
c      iang1=(idnode*ntangl)/mxnode+1
c      iang2=((idnode+1)*ntangl)/mxnode
```

Текст начинается с 7  
позиции

Длина строки 72  
символа

Позиции 1 – комментарий (C),  
2-5 – поле метки  
6 – признак продолжения строки

# Современный формат файлов (\*.f90)

```
Program ReadH
Use Vars
Implicit Real(8) (A-H,O-Z)

Character(256) str

Open(3,File='ReadHistory.inp')
Open(6,File='ReadHistory.out')

! Open files

Call ReadInputParameters

Write(7, '('iChunk      N  dt, ps  Y      Z      Tot  DCx'', &
      SD(DCz)      SD(DCtot) ''')

iu=7
If (ModeRecalc==1) Then
  Goto 30
  iu=9
Endif

30  Read(5,*)      ! Skip comment
Do
  Read(5,FMT='(a256)',END=10,ERR=20) str
  If (INDEX(str,'timestep')==1) Then
    irec=irec+1
    timeold=time
    Read(5,'(20x,f20.10)')Box(2)
    Read(5,'(40x,f20.10)')Box(3)
    Natoms=0
```

Начало в любом  
месте

Длина строки не  
регламентирована  
(ограничена  
компилятором,  
обычно 128)

Комментарий

Знак продолжения

Комментарий

# Fortran. Основные программные единицы

!Основная (головная) программа – обязательна!

File1.f90

```
Program SuperProg  
...  
End
```

! Подпрограмма

```
Subroutine Subr1  
...  
End
```

! Блок данных (стиль устарел)

```
BLOCK DATA  
....  
END
```

! Функция, определяемая пользователем

```
Function Func1  
...  
End
```

File2.f90

(число файлов и  
программные единицы  
в них – любые)

! Модуль – содержит данные и/или подпрограммы, которые надо сделать доступными (или недоступными) другим программным единицам

```
Module ProgVars  
...  
End module
```

# Fortran. Структура программной единицы

! Комментарий: начинается с ! в любом месте программной единицы и строки

Program *SuperProg*

! Заголовок программной единицы

Use *Vars*

! Подключение данных или ПП из готовых модулей

! Неявное описание типов переменных для ускорения работы (не обязательно)

Implicit Real(8) (A-H,O-Z)

! Явное описание типов переменных (не обязательно, если не массив и есть Implicit)

Integer(4), parameter:: *MaxAt=100*

! Целочисленный параметр (не может изменяться)

Integer(4) *NA(MaxAt)*, *lmax/100/*

! Описание целой переменной (массива)

Real(8) *C(3,MaxAt)*

! Описание действительной переменной/массива

Character(10) *AtomName(MaxAt)*

! Описание строковой переменной/массива

Logical(4) *IsItYou/.TRUE./*

! Описание логической переменной

Complex(16) *Z,zz(10)*

! Комплексные переменные

! Исполняемые операторы

*NA(1)=6*

*C(1,1)=1.d0*

*Aname(1)='C'*

...

! Конец программной единицы

End

# Fortran. Основные операторы

**! Целые числа (Integer(4) от  $\sim 10^{10}$  до  $\sim 10^{10}$ )**      **! Действительные числа ( $\sim 7$  знач.цифр)**  
100 -1 -123      1.745 -5. -2.3e-18 1.  
**! Действительные числа двойной точности (Real(8) от  $\sim 10^{309}$  до  $\sim 10^{309}$  ( $\sim 12$  значащих цифр))**  
1.745d0 -5.d0 -2.3d-18 1.234d23 -1023.05d-26 1.d0  
**! Строковые константы**  
'Abcd' 'C23 ' ' N20 = 0.1'  
**! Логические константы**  
.TRUE. .FALSE.  
**! Комплексные числа**  
(-5.,2.) (-1.234e-5,0.546d0)  
**! Арифметические операторы**  
+ - \* / \*\* (возведение в степень) > < >= <= ==  
**! Операторы сравнения**  
> < >= <= == /=  
.GT. .LT. .GE. .LE. .EQ. .NE.    **! Устаревший вариант**  
**! Встроенные математические функции**  
SQRT(x) EXP(x) SIN(x) COS(x) TAN(x) ASIN(x) ACOS(x) ATAN(x)      **!(одинарная точность)**  
DSQRT(x) DEXP(x) DSIN(x) DCOS(x) DASIN(x) DACOS(x) DATAN(x) **!(двойная точность)**  
**! Встроенные строковые функции**  
INDEX(string, substring)      **! Ищет подстроку в строке (возвращает 0 или нач. номер)**  
REPEAT(n, symb)      **! Создает строку с повторяющимися n раз символами symb**  
CHAR(n)      **! Дает символ номер n из таблицы ASCII характеров**  
ICHAR(symb)      **! Дает номер для символа symb из таблицы ASCII характеров**

**! Преобразование типов**  
x=Dble(5)    **! I4->R8**  
i=INT(x)      **! R->I4**  
I2=INT2(x)    **! R->I2**  
**! Округление до ближайшего**  
i=IDNINT(x)  
**! Целая часть числа**  
i=INT(x)

# Fortran. Основные выражения

## ! Присвоение

```
lmax=100
X=1.
X2=1.d0
aa=1.d-5
C(1,5)=1.d0
B(l,j)=x
D1(l,j,k)=DSIN(45.d0)
FilOut='Output.txt'
```

## ! Открытие/закрытие файлов

```
Open(5,File='input.inp')
Open(6, File=FilOut)
...
Close(5)
```

## ! Условный оператор

```
If (x>10.d0) Then
    a=5.d0
Elseif (x==3.d0) Then
    a=1.d0
Else
    a=0.d0
Endif
```

## ! Условный оператор (кратк.вар.)

```
If (x<=0.d0) x=0.d0
```

## ! Безусловный переход

```
Goto 10
...
10 x=10.d0      ! метка
Goto A2         ! Буквенная метка
...
A2: Continue   ! Продолжение
```

## ! Цикл

```
Do i=1,100
    x=x+1.d0
    If (i/2*2==i) Cycle
    If (x>100.d0) Exit
Enddo
```

## ! Вызов подпрограмм и функций

```
Call Subr1(10,l,x)
X=Func1(10,i)
Y=DSQRT(3.d0)
```

## ! Чтение и запись

```
Read(*,*)x
Read(5,*) x

Write(*,10)l,x,y,z
10 Format(i2,3f10.5)

Write(5,'(i2,3f10.5)')i,x,y,z
```

## ! Позиционирование ! в файле

```
Write(6,*)
Backspace(5)
Rewind(5)
Do While (.not.EOF(5))
    Read(5,*)x
Enddo
! EOF() только Intel Fortran
```

## ! Аргументы командной ! строки

```
nrg=NARGS()
Call GetArg(Int2(1),arg1)
```

# Fortran. Чтение и запись

! Операторы чтения и записи

Read(unit, format) varlist

Write(unit, format) varlist

Print unit, varlist

!Unit – номер открытого файла

Open(5,file='input.txt')

Read(5,\*)a

! Unit \* означает стандартный канал ввода/вывода (клавиатура/экран)

! Format - метка оператора FORMAT

Write(5,10)x

10 Format(i2,3f10.5)

! Format внутри оператора

Read(5,'(i2,3f10.5)')i,x,y,z

Write(5,'(i2,3f10.5)')i,x,y,z

Самый часто используемый способ чтения/записи

! Спецификации формата:

in – n знакомест под целое число со знаком: i2, i4, i20

nx – n пробелов: 5x, 3x, 20x

Fn.m - n знакомест под действительное число с m знаками после запятой: f10.5 f12.6

En.m - n знакомест под число в экспоненциальном формате с m знаками после запятой (e18.6

позволяет напечатать \_\_\_\_-1.234567e-234 )

Gn.m - n знакомест под число в формате F (если оно небольшое и умещается в n знаков) или

E с m знаками после запятой (g18.6 напечатает 1234.567 или 1.234567e8 )

## Fortran. Чтение и запись 2

! Бесформатный ввод/вывод (более быстрый)

Read(unit) varlist

Write(unit) varlist

! Ввод/вывод под управлением списка

Read(5,\*)a,b,c

Write(6,\*)na,x,y,z

! Ввод/вывод в двоичный файл (быстрый, компактный, но нельзя прочитать как текст)

Open(6,file='out.bin', Format='BINARY')

Write(6)x,y,z

! Ввод/вывод из строки

Character(255) String

Read(5,'(a255)')String

Read(String,'(i2,3f10.5)')i,x,y,z

! Ввод/вывод массивов

Real(8) x(100),y(1000)

Write(6,'(10f12.4)')(x(j),j=1,50)

Write(6,'(10f12.4)')y

n=20

Write(6,'(<n+5>f12.4)')y

! Вывод элементов 1-50 массива X (напечатаются в 10

! колонок по 12 символов каждая с 4 знаками после точки)

! Вывод всех элементов массива в 10 колонок по 12 симв.

! Только для IntelFortran



# Fortran. Чтение и запись 3

! Обработка ошибок при чтении/записи

iu=5

Read(unit=5,fmt=10,err=20,end=30) i,x,y,z

10 Format(i2,f10.4)

...

20 Write(\*,'(//'' Reading error!'')')

Stop

...

30 Write(\*,'(//'' End of file during read from unit'',i2)')iu

Stop 'End of file'

...

! Часть данных можно не читать, а инициализировать в программе в момент компиляции

! (не путать с параметрами!). На их задание программа не будет тратить время:

Character(2) ELNAME(18)/ & ! (Знак & означает продолжение оператора на след. строке)

'H ', 'He', &

'Li', 'Be', 'B ', 'C ', 'N ', 'O ', 'F ', 'Ne', &

'Na', 'Mg', 'Al', 'Si', 'P ', 'S ', 'Cl', 'Ar' /

Real(8) AtomMass(18)/1.007825D+00,4.0026D+00,7.01600D+00,9.01218D+00,11.00931D+00, &

12.0D+00,14.00307D+00,15.99491D+00,18.99840D+00,19.99244D+00, &

22.9898D+00,23.98504D+00,26.98153D+00,27.97693D+00, &

30.97376D+00,31.97207D+00,34.96885D+00,39.948D+00/

Integer(4) NA(10)/1,2,3,4,5,6,7,8,9,10/,iOption/0/

# Fortran. Пример программы бесформатного чтения

Program ReadFreeForm

```
Implicit Real(8) (A-H,O-Z)
```

! Неявно объявляем переменные

```
Integer(4), parameter::MaxAt=100
```

! Задаем целочисленный параметр, если требуется

```
Integer(4) NA(MaxAt)
```

! Его можно использовать для объявления массивов

```
Real(8) C(3,MaxAt)
```

```
Character(255) String
```

```
Open(5,file='input.xyz')
```

! Открываем файлы...

```
Open(6,file='Student.out')
```

! ...ввода

! ....вывода

```
numat=0
```

! Счетчик атомов

```
Do While (.not.EOF(5))
```

! Основной цикл чтения до конца файла

```
    Read(5,'(a255)')String
```

! Сначала читаем строку длиной 255 символов целиком

```
    If (Len_Trim(String)==0) Exit
```

! Проверяем – пустая строка конец ввода: выход из цикла

```
    Read(String,*)i,x,y,z
```

! Если не пустая – читаем атомный номер и координаты

```
    numat=numat+1
```

! Увеличиваем счетчик атомов и заносим атом в массивы

```
    NA(numat)=i
```

```
    C(1,numat)=x
```

```
    C(2,numat)=y
```

```
    C(3,numat)=z
```

```
Enddo
```

```
Write(6,'(//'' Numat = '' ,i5)')Numat
```

! Печатаем в выходной файл число атомов

```
Do i=1,Numat
```

```
    Write(6,'(i2,3f10.4)')NA(i),C(1:3,i)
```

! Печатаем тип и координаты каждого атома

```
Enddo
```

```
Close(5)
```

! Закрываем файлы

```
Close(6)
```

```
End
```

# Fortran. Домашнее задание

1. Написать программу ReadMol1, которая:
  1. Открывает файл mol1.xyz
  2. Пропускает две строки
  3. Читает N атомов в виде <ат.номер> x y z по формату i2,3x,3f10.4 (Признак окончания пустая строка)
  
2. Написать программу ReadMol2, которая:
  1. Открывает файл mol2.xyz
  2. Пропускает строки, если они пустые или начинаются с ! (комментарии)
  3. Если строка начинается со слова GEO читает геометрию в формате <ат.номер> x y z (свободный формат!)
  4. Открывает файл mol3.xyz
  5. Пропускает две строки
  6. Записывает туда атомы в формате <Символ элемента+номер атома> x y z
  
3. Усовершенствовать программу ReadMol2 так, чтобы она имена файлов ввода и вывода брала из командной строки. Если они не заданы – выводила сообщение на экран об ошибке.
  
4. Написать программу, которая рассчитывает массу молекулы и межатомные расстояния

# Лекция 2

Работа с массивами.

Подпрограммы, функции и модули.

Обмен данными между программными  
единицами

# Fortran. Массивы. 1

! Массивы описываются оператором Dimension (устаревший вариант)

```
Real(8) A
```

```
Dimension A(10,10)
```

! или любым оператором типа (современный вариант)

```
Real(8) A(10,10)
```

! Границы массива и индексные переменные могут быть только целочисленными!

! Можно использовать целочисленные параметры или выражения:

```
Integer(4), parameter :: MaxAt=100
```

```
Real(8) C(3,MaxAt), A(2*MaxAt+1)
```

! Массивы могут быть одно- двух- и многомерными (макс.размерность определяется

! компилятором, обычно 7), могут быть любого типа

```
Integer(4) NA(50), IX1(10,20)      ! Целочисленные вектор и матрица
```

```
Real(8) A(10,20,100,5)             ! 4-х мерный массив
```

```
Character(50) Str(20)              ! Массив из 20 строк длиной 50 символов каждая
```

! Можно задать нижнее и верхнее значение индексов массива (отличие от других языков!)

```
Real(8) X(0:10), A(2:5,-10:10)
```

! Массив можно инициализировать при его описании (присвоение происходит в момент компиляции, не требует работы программы)

```
Integer(4) IA(10)/1,2,3,4,5,6,7,8,9,10/)
```

## Fortran. Массивы. 2

! Массиву могут быть присвоены значения с помощью обычного присвоения

```
Integer(4) NA(0:10),M(100)
```

```
NA(0)=0
```

! Простое присвоение значения одному элементу

```
Do i=1,10
```

```
    NA(i)=i*i+5
```

! Присвоение в цикле

```
Enddo
```

! ...или с помощью т.н. конструктора :

```
NA(1:10)=(/1,2,3,5,10,17,20,22,23/)
```

! Конструктор массива

! Присвоение можно произвести всему массиву сразу или отдельным «сечениям»:

```
Integer(4) NA(10,10)
```

```
NA=0
```

! Присвоение значения всему массиву

```
NA(1:3,1)=(/1,2,3/)
```

! Присвоение нескольким элементам

# Fortran. Массивы. 3

! Размер и форму массива можно задать в процессе работы программы:  
! Для этого надо объявить его размещаемым (allocatable) и использовать команду Allocate:

```
Integer(4), allocatable:: N(:), M(:, :)
```

! Объявление размещаемых массивов

```
...
```

```
Allocate(N(10),M(10,10))
```

! Размещение массива в памяти

```
N(1:10)=(/1,2,3,4,5,6,7,8,9,10/)
```

! Присвоение значений

```
...
```

```
Deallocate(N)
```

! Деаллокация (освобождение памяти)

```
Allocate(N(20))
```

! Переразмещение

```
N=100
```

! Присвоение всем элементам значения 100

```
...
```

## Fortran. Массивы. 4

! Можно работать с отдельными элементами массива, или работать с ним как с одной  
! переменной, при этом все операции производятся «поэлементно», т.е. все элементы  
! претерпевают одинаковые преобразования

```
Integer(4) II(10), JJ(10)
```

```
Real(8) A(10), B(10)
```

```
II=5
```

```
JJ=II+1      ! Все элементы JJ равны 6. Два массива обязаны иметь одинаковый размер и форму
```

```
JJ(3)=3      ! Элемент 1 задан отдельно
```

```
A=1.d0
```

```
B=DSIN(A)+5.d0*Dble(JJ)  ! Многие функции также действует на все элементы массива
```

! (т.н. элементная функция). Не все функции элементные, но многие, в т.ч. математические

! Надо помнить, что если массив имеет большой размер, то элементные операции могут быть

! невыгодны с точки зрения производительности

```
Integer(4), parameter:: MaxAt=1000
```

```
Real(8) A(MaxAt)
```

```
A=1.d0      ! Присвоение всем 1000 элементам
```

```
Numat=5
```

```
Do i=1,Numat
```

```
    A(i)=1.d0      ! Присвоение 5 элементам
```

```
Enddo
```

```
A(1:5)=1.d0      ! Присвоение 5 элементам (такая операция – сечение - самая быстрая!)
```



## Fortran. Массивы. 5

! Самые быстрые операции – элементные сечениями массивов:

```
Real(8) A(100),B(10)
```

```
B=4.d0
```

```
A(1:10)=(/1.d0,2.d0,3.d0,4.d0,5.d0/)
```

```
A(1:10)=Dsin(A(1:10)+B(1:10)*5.d0
```

! Аналогично можно использовать сечения многомерных массивов

```
Real(8) C(3,20),T(3,3)
```

```
C(1:3,1)=(/1.d0,2.d0,0.d0/)
```

```
Do i=1,3
```

```
    T(1:3,i)=(/i, i*i, i*i*i/)
```

```
    C(1:3,i+1)=T(1:3,i)
```

```
Enddo
```

! Со строками можно поступать как с массивами – рассматривать их сечения:

```
Character(255) String
```

```
String(10:30)='this is substring!..'
```

```
String(10:10)='T'
```

# Fortran. Подпрограммы и функции

! Если в программе имеются однотипные фрагменты и операции,  
! их можно и нужно оформлять в виде подпрограмм или функций

```
Program Prog1
Real(8) X(20),A(10,10)
...
A=2.d0
Do i=1,10
    X(i)=a(i,1)+2.d0*DSQRT(Dble(i))
Enddo
...
A=3.d0
Do i=1,20
    X(i)=a(i,2)+3.d0*DSQRT(Dble(i))
Enddo
```

```
Program Prog1
Real(8) X(20)
...
Call ProcX(2.d0,10,1,2.d0,X)
...
Call ProcX(3.d0,10,2,2.d0,X)
...
End
```

Актуальные  
параметры

! Подпрограмма

```
Subroutine ProcX(aa,N,ii,bb,X)
Real(8) X(20),A(10,10),aa,bb
Integer(4) N, ii, i
A=aa
Do i=1,N
    X(i)=a(i,ii)+bb*DSQRT(Dble(i))
Enddo
End
```

Формальные  
параметры

Актуальные и формальные  
параметры должны точно  
соответствовать друг другу по:  
(1) типу,  
(2) порядку в списке,  
(3) в случае массивов – размеру !

## Fortran. Подпрограммы и функции. 2

! Параметры подпрограмм и функций могут быть входными, выходными или обоих типов

! Входные могут быть константами, остальные только переменными

```
Program Prog1
```

```
Implicit Real(8) (A-H,O-Z)
```

```
b=1.d0
```

! Входные и выходные должны быть

! инициализированы перед вызовом подпрограммы

```
Call Subr1(10.d0,b,c)
```

! 10.d0 – входной, c – выходной, b – входной/выходной

```
End
```

```
!*****
```

```
Subroutine Subr1(aa,bb,cc)
```

! Имена параметров в П и ПП никак не связаны (могут

! совпадать или не совпадать)

```
Implicit Real(8) (A-H,O-Z)
```

! Все переменные д.б. описаны, включая параметры

```
cc=aa+Dsin(bb)
```

```
bb=cc-1.d0
```

```
If (cc>3.d0) Return
```

! Выход из ПП можно произвести в любой момент оператором Return

```
cc=cc+2.d0
```

```
bb=cc*2.d0
```

! aa=5.d0 – такую операцию произвести нельзя, т.к. aa – константа в вызывающей программе

```
End
```

! При достижении оператора End в ПП происходит выход из нее

## Fortran. Подпрограммы и функции. 2

! Единственное отличие функции от ПП – один из выходных параметров – сама функция

! Функция может быть и массивом или строкой

```
Program Prog1
```

```
Implicit Real(8) (A-H,O-Z)
```

```
b=1.d0
```

```
c=Func1(10.d0,b)           ! 10.d0 – входной, b – входной/выходной, Func1 – выходной
```

```
End
```

```
!*****
```

```
Function Func1(aa,bb)
```

! Имена параметров в П и Ф никак не связаны

```
Implicit Real(8) (A-H,O-Z)
```

! Описание переменных действует и на функцию Func1

```
cc=aa+Dsin(bb)
```

```
bb=cc-1.d0
```

```
If (cc>3.d0) Then
```

```
    Func1=0.d0
```

```
    Return
```

! Выход из Ф можно произвести в любой момент, если Func1 задано!

```
Endif
```

```
cc=cc+2.d0
```

```
bb=cc*2.d0
```

```
Func1=cc
```

! Функция обязана присвоить значение себе!

```
End
```

# Fortran. Подпрограммы и функции. 3

! При передаче параметров в ПП или Ф очень важно следить за типом и размерами параметров

Огромное число ошибок (наиболее неприятных!) вызывается  
несоответствием порядка, типов и размеров параметров

```
Program Prog1
Implicit Real(8) (A-H,O-Z)
Integer(4) NA(10)
Real(8) X(10),A(10,100),B(100)
LDA=10
m=100
Call Subr1(NA,X,c,LDA,A,m,B)
End
```

!\*\*\*\*\*

Subroutine Subr1(NA,XX,LDA,c,AA,m,B)	! Порядок C и LDA перепутан – вызовет ошибку!
! Implicit Real(8) (A-H,O-Z)	! Implicit закомментарен: типы не совпадут – будет ошибка!
Real(8) XX(20)	! Размер массива не совпадает – произойдет ошибка
Integer(4) NA(*)	! Удобный способ описать массив в ПП (не всегда возможен)
Real(8) A(LDA,*)	! Для двумерных массивов надо иметь LDA – ведущий размер ! (leading dimension)
Real(8) B(m)	! Другой способ передать размер массива
....	
End	

# Fortran. Обмен данными между подпрограммами

! Обмен общими данными в программе можно произвести тремя путями:

1. Через параметры ПП (уже рассмотрен)

2. С помощью модулей:

Module Vars

! Описание модуля

Implicit Real(8) (A-H,O-Z)

Integer(4) NA(10)

! Данные в модуле будут доступны всем ПП, где он подключен

Real(8) X(10),A(10,100),B(100)

End

!\*\*\*\*\*

Program Prog1

Use Vars

! Подключение модуля (должно производиться до описания переменных)

Implicit Real(8) (A-H,O-Z)

Call Subr1

End

!\*\*\*\*\*

Subroutine Subr1

Use Vars, Only: NA,X,AA=>A

! Подключение только нужных данных из модуля,

! причем массив A из модуля в ПП будет иметь имя AA

Implicit Real(8) (A-H,O-Z)

X=0.d0

! Изменение данных здесь вызовет изменение данных в модуле и его увидят все ПП

...

End

# Fortran. Обмен данными между подпрограммами

3. Еще один способ обмена - с помощью COMMON-блоков (устаревший способ).

COMMON – аналог модуля, но в нем надо следить за порядком и типом переменных (неудобно)

До сих пор можно встретить в старых программах:

```
Program Prog1
```

```
Implicit Real(8) (A-H,O-Z)
```

```
COMMON /COM1/ X(100),NA(10)
```

! Common-блок с именем COM1

```
Call Subr1
```

```
End
```

```
!*****
```

```
Subroutine Subr1
```

```
Implicit Real(8) (A-H,O-Z)
```

```
COMMON /COM1/ XX(100),NB(10)
```

! Подключение Common-блока. Порядок, тип и  
! размер переменных должны быть одинаковы  
! во всех ПП (за этим надо специально следить)

```
NB=0
```

```
XX=1.d0
```

```
End
```

# Усовершенствование программы бесформатного чтения

Program ReadFreeForm

```
Implicit Real(8) (A-H,O-Z)
```

```
Integer(4), parameter::MaxAt=100
```

```
Integer(4) NA(MaxAt)
```

```
Real(8) C(3,MaxAt)
```

```
Character String*255,Aname*10
```

```
Open(5,file='input.xyz')
```

```
Open(6,file='Student.out')
```

```
numat=0
```

```
Do While (.not.EOF(5))
```

```
    Read(5,'(a255)')String
```

```
    If (Len_Trim(String)==0) Exit
```

```
    Call ParseString(String,Aname,x,y,z)
```

```
    numat=numat+1
```

```
    Call ParseAname(Aname,nna)
```

```
    NA(numat)=nna
```

```
    C(1,numat)=x
```

```
    C(2,numat)=y
```

```
    C(3,numat)=z
```

```
Enddo
```

```
Write(6,'(//' Numat = ',i5)')Numat
```

```
Do i=1,Numat
```

```
    Call SetAname(NA(i),i,Aname)
```

```
    Write(6,'(a10,3f10.4)')Aname,C(1:3,i)
```

```
Enddo
```

```
Close(5)
```

```
Close(6)
```

```
End
```

! Читаемый файл может выглядеть так:

```
C1  0.00    1    2.0
O2    0    1.0    2.0
N3  0.0    1.0    2.0
F   0.0    1.0    2.0
```

! Aname хранит имя атома и номер (C1, O12, H25...)

! Открываем файлы...

! ...ввода

! ....вывода

! Счетчик атомов

! Основной цикл чтения до конца файла

! Сначала читаем строку длиной 255 символов целиком

! Проверяем – пустая строка конец ввода: выход из цикла

! Выделяем Aname и читаем x, y, z

! Увеличиваем счетчик атомов и заносим атом в массивы

! Определяем NA по Aname с пом. ПП

! Записываем атомный номер

! Печатаем в выходной файл число атомов

! Печатаем тип и координаты каждого атома

! Определяем Aname по NA и номеру атома

! Закрываем файлы



# Усовершенствование программы бесформатного чтения

!Входной файл может выглядеть так:

```
C1  0.00  1  2.0
      O2      0  1.0  2.0
N3  0.0  1.0  2.0
F  0.0  1.0  2.0
```

!Выходной файл должен выглядеть так:

```
Numat =      4
C1          0.0000  1.0000  2.0000
O2          0.0000  1.0000  2.0000
N3          0.0000  1.0000  2.0000
F4          0.0000  1.0000  2.0000
```

```
Subroutine ParseString(String,Aname,x,y,z)
```

! Input, output

! ПП выделяет из строки имя атома, а из остатка

! Читает x y z

```
Implicit Real(8) (A-H,O-Z)
```

```
Character String*255, Aname*10
```

```
String=AdjustL(String)
```

! Сдвигаем строку влево

```
Do i=1,255
```

! По очереди проверяем все символы в строке

```
    If (String(i:i)==' ') Exit
```

! Первый пробел (имя закончилось) - выход

```
    If (i<=10) Aname(i:i)=String(i:i)
```

! Если не пробел, переносим символ в Aname

```
    String(i:i)=' '
```

! Этот символ в строке заменяем на пробел

```
Enddo
```

```
Read(String,*)x,y,z
```

! Читаем координаты из строки

```
End
```

# Усовершенствование программы бесформатного чтения

```
Subroutine ParseAname(Aname,NA)

Implicit Real(8) (A-H,O-Z)
!
! Подпрограмма определяет атомный номер элемента по имени атома в виде C1, o12, H25...
!
Character(10) Aname
Character(2) ElName(10) & !<-- Продолжение оператора на другой строке
/'H ','HE','LI','BE','B ','C ','N ','O ','F ','NE'/      ! Имена хим. элементов (загл.буквы)
Character(1) symb,symb1      ! Вспомогательные переменные

Do i=1,10
    symb=Aname(i:i)
    ic=ICHAR(symb)           ! Переводим символ в ASCII-код
    If (ic>=48.and.ic<=57) Aname(i:i)=' '      ! Если символ-цифра (ASCII-коды от 48 до 57,
                                                ! то заменяем ее на пробел)
    If (ic>=97.and.ic<=122) Aname(i:i)=CHAR(ic-32) ! Заменяем строчные буквы на заглавные
Enddo

Do i=1,10
    If (INDEX(Aname,ElName(i))>0) Then      ! Есть ли в Aname подстрока ElName(i)?
        NA=i
        Return
    Endif
Enddo

End
```

# ASCII коды

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	<b>NUL</b> (null)	32	20	040	&#32;	Space	64	40	100	&#64;	@	96	60	140	&#96;	`
1	1	001	<b>SOH</b> (start of heading)	33	21	041	&#33;	!	65	41	101	&#65;	A	97	61	141	&#97;	a
2	2	002	<b>STX</b> (start of text)	34	22	042	&#34;	"	66	42	102	&#66;	B	98	62	142	&#98;	b
3	3	003	<b>ETX</b> (end of text)	35	23	043	&#35;	#	67	43	103	&#67;	C	99	63	143	&#99;	c
4	4	004	<b>EOT</b> (end of transmission)	36	24	044	&#36;	\$	68	44	104	&#68;	D	100	64	144	&#100;	d
5	5	005	<b>ENQ</b> (enquiry)	37	25	045	&#37;	%	69	45	105	&#69;	E	101	65	145	&#101;	e
6	6	006	<b>ACK</b> (acknowledge)	38	26	046	&#38;	&	70	46	106	&#70;	F	102	66	146	&#102;	f
7	7	007	<b>BEL</b> (bell)	39	27	047	&#39;	'	71	47	107	&#71;	G	103	67	147	&#103;	g
8	8	010	<b>BS</b> (backspace)	40	28	050	&#40;	(	72	48	110	&#72;	H	104	68	150	&#104;	h
9	9	011	<b>TAB</b> (horizontal tab)	41	29	051	&#41;	)	73	49	111	&#73;	I	105	69	151	&#105;	i
10	A	012	<b>LF</b> (NL line feed, new line)	42	2A	052	&#42;	*	74	4A	112	&#74;	J	106	6A	152	&#106;	j
11	B	013	<b>VT</b> (vertical tab)	43	2B	053	&#43;	+	75	4B	113	&#75;	K	107	6B	153	&#107;	k
12	C	014	<b>FF</b> (NP form feed, new page)	44	2C	054	&#44;	,	76	4C	114	&#76;	L	108	6C	154	&#108;	l
13	D	015	<b>CR</b> (carriage return)	45	2D	055	&#45;	-	77	4D	115	&#77;	M	109	6D	155	&#109;	m
14	E	016	<b>SO</b> (shift out)	46	2E	056	&#46;	.	78	4E	116	&#78;	N	110	6E	156	&#110;	n
15	F	017	<b>SI</b> (shift in)	47	2F	057	&#47;	/	79	4F	117	&#79;	O	111	6F	157	&#111;	o
16	10	020	<b>DLE</b> (data link escape)	48	30	060	&#48;	0	80	50	120	&#80;	P	112	70	160	&#112;	p
17	11	021	<b>DC1</b> (device control 1)	49	31	061	&#49;	1	81	51	121	&#81;	Q	113	71	161	&#113;	q
18	12	022	<b>DC2</b> (device control 2)	50	32	062	&#50;	2	82	52	122	&#82;	R	114	72	162	&#114;	r
19	13	023	<b>DC3</b> (device control 3)	51	33	063	&#51;	3	83	53	123	&#83;	S	115	73	163	&#115;	s
20	14	024	<b>DC4</b> (device control 4)	52	34	064	&#52;	4	84	54	124	&#84;	T	116	74	164	&#116;	t
21	15	025	<b>NAK</b> (negative acknowledge)	53	35	065	&#53;	5	85	55	125	&#85;	U	117	75	165	&#117;	u
22	16	026	<b>SYN</b> (synchronous idle)	54	36	066	&#54;	6	86	56	126	&#86;	V	118	76	166	&#118;	v
23	17	027	<b>ETB</b> (end of trans. block)	55	37	067	&#55;	7	87	57	127	&#87;	W	119	77	167	&#119;	w
24	18	030	<b>CAN</b> (cancel)	56	38	070	&#56;	8	88	58	130	&#88;	X	120	78	170	&#120;	x
25	19	031	<b>EM</b> (end of medium)	57	39	071	&#57;	9	89	59	131	&#89;	Y	121	79	171	&#121;	y
26	1A	032	<b>SUB</b> (substitute)	58	3A	072	&#58;	:	90	5A	132	&#90;	Z	122	7A	172	&#122;	z
27	1B	033	<b>ESC</b> (escape)	59	3B	073	&#59;	;	91	5B	133	&#91;	[	123	7B	173	&#123;	{
28	1C	034	<b>FS</b> (file separator)	60	3C	074	&#60;	<	92	5C	134	&#92;	\	124	7C	174	&#124;	
29	1D	035	<b>GS</b> (group separator)	61	3D	075	&#61;	=	93	5D	135	&#93;	]	125	7D	175	&#125;	}
30	1E	036	<b>RS</b> (record separator)	62	3E	076	&#62;	>	94	5E	136	&#94;	^	126	7E	176	&#126;	~
31	1F	037	<b>US</b> (unit separator)	63	3F	077	&#63;	?	95	5F	137	&#95;	_	127	7F	177	&#127;	DEL

ICHAR(symb)

CHAR(i)

# Усовершенствование программы бесформатного чтения

```
Subroutine SetAname(NA,n,Aname)

Implicit Real(8) (A-H,O-Z)

Character(10) Aname
Character(2) ElName(10)/'H ','He','Li','Be','B ','C ','N ','O ','F ','Ne'/
Character(10) buf

Write(buf,'(i10)')n
Aname=Trim(ElName(NA))//AdjustL(buf)

End
```

! Записываем число в строку buf  
! Формируем Aname из Elname и buf  
! // - конкатенация (слияние) строк  
! Trim удаляет пробелы в конце строки

# Домашнее задание

Усовершенствовать программу чтения так, чтобы:

1. Символы элементов задавались в отдельном модуле и подключались к разным программам
2. Символы элементов при этом хранились в верхнем и нижнем регистрах, как в обычном тексте: He, Ne (усовершенствовать ParseAname так, чтобы она конвертировала их в верхний регистр при распознавании)
3. Программа должна пропускать в начале файла пустые строки и комментарии и начинать чтение с команды GEO в начале строки.

# Лекция 3

Матричные операции.

Случайные числа.

Описание вращений

# Операции с матрицами – “old school”:

Элементное умножение  $C=A*B$  - не то же самое, что матричное умножение!

! Матричное умножение

Real(8) A(10,10), B(10,10), v(10) ! Максимальный размер системы

...

N=5

! «Реальный» размер системы

Do i=1,N

Do j=1,N

s=0.d0

Do k=1,N

s=s+A(i,k)\*B(k,j)

Enddo

C(i,j)=s

Enddo

Enddo

$$c(i, j) = \sum_{k=1}^n a(i, k) \cdot b(k, j)$$

! C = A B

! s=s+A(k,i)\*B(k,j) :: C=A<sup>T</sup>B

! Функция, вычисляющая произведение матрицы на вектор  $v=A*u$ :

Function MatVec(n,A,u,v)

Real(8) A(n,n), u(n), v(n)

v=0.d0

Do i=1,n

Do k=1,n

v(i)=v(i)+A(i,k)\*u(k)

Enddo

Enddo

End

```
Function DotProduct(N,X,Y)
Real(8) X(N), Y(N), DotProduct
DotProduct=0.d0
Do i=1,N
    DotProduct=DotProduct+X(i)*Y(i)
Enddo
End
```

! Вопрос: Что делает эта ПП?

# Fortran. Функции для работы с массивами. 1

Элементные операции и функции действуют на массивы ПОЭЛЕМЕНТНО. Элементное умножение – не то же самое, что матричное умножение! Матричное умножение можно выполнить с помощью встроенной функции `MatMul()`:

```
Real(8) A(10,10), B(10,10), C(10,10)
```

`C=A*B`                      ! Элементное умножение:      $c(i, j) = a(i, j) \cdot b(i, j)$

`C=matmul(A,B)`            ! Матричное умножение:      $c(i, j) = \sum_{k=1}^n a(i, k) \cdot b(k, j)$

! ВНИМАНИЕ: для использования `matmul` массивы A, B, C должны иметь согласованные  
! размеры, обеспечивающие правильное выполнение математической операции:

```
Real(8) C(N,M), A(N,K), B(K,M)    ! N, M, K – любые (в т.ч. 1), но их порядок – строго задан
```

! Можно производить умножение матрицы на вектор:

```
Real(8) A(10,10), b(10), c(10)
```

`c=matmul(A,b)`            ! Умножение матрицы на вектор дает вектор

! Скалярное произведение двух векторов (хотя бы один должен иметь форму матрицы!)

```
Real(8) r(5), v(5), x(1)
```

```
r(1:5)=(/1.d0,2.d0,3.d0,4.d0,5.d0/)
```

```
v(1:5)=(/2.d0,4.d0,1.d0,-4.d0,1.d0/)
```

```
x=matmul(Reshape(R,(/1,5/)),V)    ! Reshape() – изменение формы массива - один д.б. матрицей
```



## Fortran. Функции для работы с массивами.2

! Если массивы заданы статически (Например, с помощью параметров, «с запасом»), matmul  
! может привести к ошибкам или быть неэффективным:

Integer(4), parameter:: MaxAt=1000

Real(8) A(MaxAt, MaxAt), B(MaxAt, MaxAt), C(MaxAt, MaxAt)

Numat=10

! Зарезервированный размер массивов значительно больше, чем

C=matmul(A,B) ! реальный размер системы  $c(i, j) = \sum_{k=1}^{Numat} a(i, k) \cdot b(k, j)$

! В этих случаях матричное умножение удобно выполнить без matmul, с помощью цикла:

Do i=1, Numat

! Numat=10, в то время как размер матриц 1000

Do j=1, Numat

tmp=0.d0

Do k=1, Numat

tmp=tmp+a(i,k)\*b(k,j)

Enddo

c(i,j)=tmp

Enddo

Enddo

! Эту операцию можно оформить в виде ПП, если

! в нее передать размеры массивов и Numat:

Call Mulmat(MaxAt, MaxAt, MaxAt, Numat, A, B, C)

...

Subroutine MulMat(LDA, LDB, LDC, N, A, B, C)

Real(8) A(LDA,N), B(LDB,N), C(LDC, N)

...

! Еще одна функция для работы с матрицами Transpose() – транспонирует матрицу

Real(8) A(10,10), B(10,10), C(10,10), u(10), v(10), x

B=Transpose(A)

! Операция транспонирования:  $B = A^T : b(i, j) = a(j, i)$

A=Transpose(A)

C=matmul(Transpose(A), B)

! Matmul и Transpose можно комбинировать:  $C = A^T B$

## Fortran. Функции для работы с массивами.3

! Функция `Sum(array,[dim,mask])` суммирование и произведение элементов массива:

`Real(8) A(10),B(3,2)` (Аналогично - функция `Product()` – произведение эл.)

`A=2.d0`

`S=Sum(A)` ! `S=20.d0`

`S=Sum(A(1:3), mask=A>3.d0)` ! `S=0.d0`

**! ВНИМАНИЕ!** Суммируется весь массив, даже если реальный размер системы меньше!

`B(1:5,1)=(/1.d0,4.d0,2.d0,-5.d0,-1.d0/)`

`B(1:5,2)=(/2.d0,1.d0,-2.d0,-3.d0,0.d0/)`

`S=Sum(B)` ! `S=-1.d0`

`A(1:2)=Sum(B,1)` ! `A-> 1, -2` (суммирование по первому индексу в двух столбцах)

`A(1:2)=Sum(B,1,mask=B>0.d0)` ! `A-> 7, 3` (суммирование по первому индексу с доп.условием)

! Функции `MinVal(array, [dim,mask])`, `MaxVal(array, [dim,mask])` – мин. и макс. элемент массива:

`Integer(4) IA(10)`

`IA=(/1,2,5,4,7,8,3,2,-2,0/)`

`IAmin=MinVal(IA)` ! `Iamin=-2`

`IAmx=MaxVal(IA)` ! `Iamax=8`

! Функции `MinLoc(array, [dim,mask])`, `MaxLoc(array,[dim,mask])` – индексы минимального и максимального элемента массива:

`iimin=LocMin(IA,1)` ! `iimin=9` (если не указать `dim`, для результата потребуется массив)

`iimax=LocMax(IA,1)` ! `iimax=5`

**! ВНИМАНИЕ!** Просматривается весь массив, даже если реальный размер системы меньше!

# Операции с матрицами - ДЗ

! Вопрос: Что делают эти ПП и Ф?

```
Subroutine Normalize(n,X)
Implicit Real(8) (A-H,O-Z)
Real(8) X(n)
X=X/DSQRT(Sum(X**2))
End
```

```
Function VecNorm(n,X)
Implicit Real(8) (A-H,O-Z)
Real(8) X(n),VecNorm
VecNorm=DSQRT(Sum(X**2))
End
```

! Написать ПП, которая вычисляет расстояния между двумя заданными атомами в молекуле

# Матрицы вращений вокруг осей координат

$$U_{OX}(\varphi) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos \varphi & -\sin \varphi \\ 0 & \sin \varphi & \cos \varphi \end{pmatrix} U_{OY}(\varphi) = \begin{pmatrix} \cos \varphi & 0 & \sin \varphi \\ 0 & 1 & 0 \\ -\sin \varphi & 0 & \cos \varphi \end{pmatrix} U_{OZ}(\varphi) = \begin{pmatrix} \cos \varphi & -\sin \varphi & 0 \\ \sin \varphi & \cos \varphi & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Вращение вектора:

$$\mathbf{r}' = \mathbf{U}_{\xi}(\varphi)\mathbf{r}$$

Комбинация вращений:  $\mathbf{r}' = \mathbf{U}_{\xi_1}(\varphi)\mathbf{U}_{\xi_2}(\varphi)\dots\mathbf{U}_{\xi_n}(\varphi)\mathbf{r}$

**Теорема Эйлера:** Любая комбинация вращений в пространстве может быть представлена единственным вращением вокруг некоторой оси ON на некоторый угол  $\alpha$ , т.е парой (вектор, угол):  $(\mathbf{s}, \alpha)$