FORTRAN

Author: John Backus

History

Fortran programming language designed by John Backus and his team in IBM. Name is an abbreviation for FORmula TRANslation. The language was designed to program IBM 704. Before Fortran IBM computers were programmed with assembly language. Instead, John Backus proposed such a language that is powerful and easy to code compared to assembly to his supervisors in 1950. Some of the Fortran developers were chess players. Here are the names of all developers; Richard Goldberg, Sheldon F. Best, Harlan Herrick, Peter Sheridan, Roy Nutt, Robert Nelson, Irving Ziller, Harold Stern, Lois Haibt, and David Sayre. After six years of development phase, in 1956 first user manual is published.

At the beginnings community of programmers did not accept this new high-level language because of compiler performance worries. But in Fortran there were less code lines than assembly language by factor of 20. This gave a freedom to compiler designer to design more efficient and faster compilers.

Fortran became widely used in engineering applications because the power of handling complex equations. Computer producers began to produce Fortran appendaged computers. There were over 40 compilers in 1963 due to vogue of Fortran. And Fortran became a cross-platform programming language.

Why this language was invented?

Before Fortran computers (IBM) were programmed in assembly language. And it was a very hard and time-consuming way to programming devices. Programmer needs to write thousands of code-lines. To have ease of coding and faster coding compared to assembly language John Backus came up with Fortran. With this new programming language, complicated mathematic formulas and operation on engineering applications are became easier to code and faster compilation by computers.

When/why shall we use it?

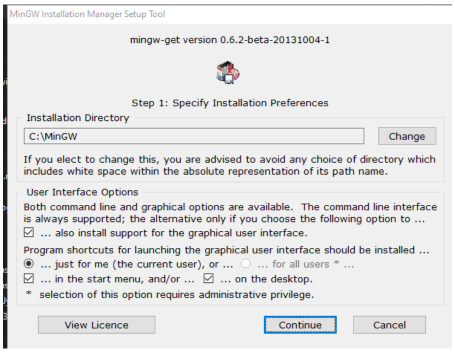
Fortran is still in use of scientific numerical programming. Because of its array handling and performance is better compared to matlab or python’s numpy library. And it is very hard to inadvertently code slow. Another benefit is there are tons of repositories about Fortran so it’s easy to get a solution if there is a problem or bug etc. And Fortran also supports object-oriented programming and parallel programing. In conclusion if we are dealing with scientific programming, Fortran can be used.

How to setup Fortran; *an environment to use it in different platforms (windows, mac, linux)*

In Windows:

Step 1: Go to https://sourceforge.net/projects/mingw/ to download MinGW.

Step 2: Double click to downloaded file. You can see



mingw-get version 0.6.2-beta-20131004-1

Step 1: Specify Installation Preferences

Installation Directory

C:\MinGW

If you elect to change this, you are advised to avoid any choice of directory which includes white space within the absolute representation of its path name.

User Interface Options

Both command line and graphical options are available. The command line interface is always supported; the alternative only if you choose the following option to …

☑ … also install support for the graphical user interface.

Program shortcuts for launching the graphical user interface should be installed …

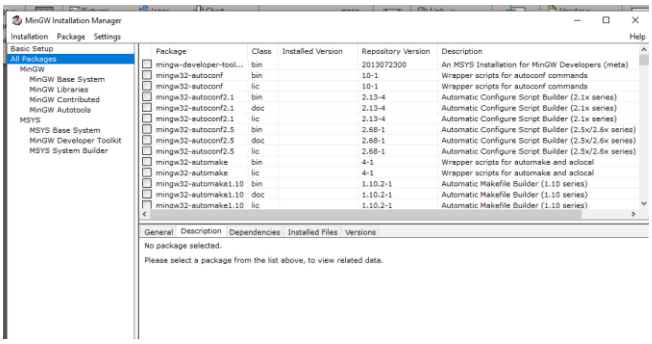
● … just for me (the current user), or … ○ … for all users * …

☑ … in the start menu, and/or … ☑ … on the desktop.

* selection of this option requires administrative privilege.

It is better to setup on recommended directory to not encounter further errors. Then click the continue button, wait for a while.

Step 3: When it is done a window is going to pop-up as below.



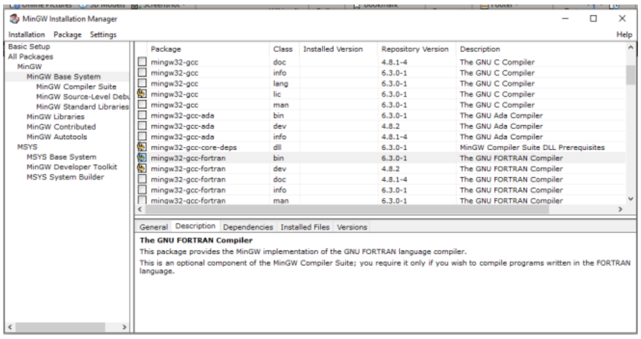
Package	Class	Installed Version	Repository Version	Description
☐ mingw-developer-tool...	bin		2013072300	An MSYS Installation for MinGW Developers (meta)
☐ mingw32-autoconf	bin		10-1	Wrapper scripts for autoconf commands
☐ mingw32-autoconf	lic		10-1	Wrapper scripts for autoconf commands
☐ mingw32-autoconf2.1	bin		2.13-4	Automatic Configure Script Builder (2.1x series)
☐ mingw32-autoconf2.1	doc		2.13-4	Automatic Configure Script Builder (2.1x series)
☐ mingw32-autoconf2.1	lic		2.13-4	Automatic Configure Script Builder (2.1x series)
☐ mingw32-autoconf2.5	bin		2.68-1	Automatic Configure Script Builder (2.5x/2.6x series)
☐ mingw32-autoconf2.5	doc		2.68-1	Automatic Configure Script Builder (2.5x/2.6x series)
☐ mingw32-autoconf2.5	lic		2.68-1	Automatic Configure Script Builder (2.5x/2.6x series)
☐ mingw32-automake	bin		4-1	Wrapper scripts for automake and aclocal
☐ mingw32-automake	lic		4-1	Wrapper scripts for automake and aclocal
☐ mingw32-automake1.10	bin		1.10.2-1	Automatic Makefile Builder (1.10 series)
☐ mingw32-automake1.10	doc		1.10.2-1	Automatic Makefile Builder (1.10 series)
☐ mingw32-automake1.10	lic		1.10.2-1	Automatic Makefile Builder (1.10 series)

No package selected.

Please select a package from the list above, to view related data.

Select mingw-developer-toolkit by left-click then it is automatically selected all the needed ones. Select mingw-base.

Step 4: Go to left panel and open MinGW Base System. Select mingw-gcc-fortran



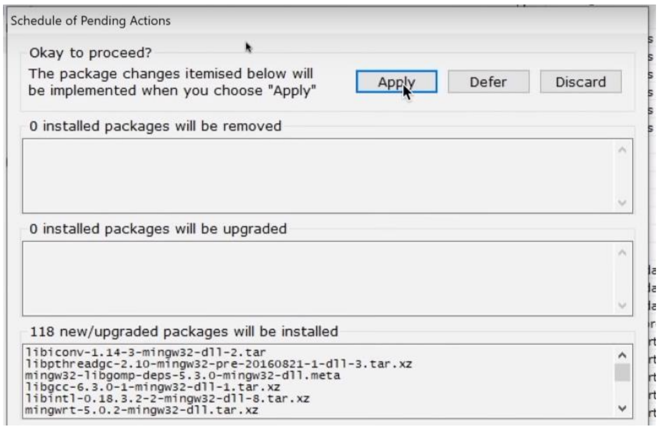
Package	Class	Installed Version	Repository Version	Description
☐ mingw32-gcc	doc		4.8.1-4	The GNU C Compiler
☐ mingw32-gcc	info		6.3.0-1	The GNU C Compiler
☐ mingw32-gcc	lang		6.3.0-1	The GNU C Compiler
☑ mingw32-gcc	lic		6.3.0-1	The GNU C Compiler
☐ mingw32-gcc	man		6.3.0-1	The GNU C Compiler
☐ mingw32-gcc-ada	bin		6.3.0-1	The GNU Ada Compiler
☐ mingw32-gcc-ada	dev		4.8.2	The GNU Ada Compiler
☐ mingw32-gcc-ada	info		4.8.1-4	The GNU Ada Compiler
☑ mingw32-gcc-core-deps	dll		6.3.0-1	MinGW Compiler Suite DLL Prerequisites
☑ mingw32-gcc-fortran	bin		6.3.0-1	The GNU FORTRAN Compiler
☑ mingw32-gcc-fortran	dev		4.8.2	The GNU FORTRAN Compiler
☐ mingw32-gcc-fortran	doc		4.8.1-4	The GNU FORTRAN Compiler
☐ mingw32-gcc-fortran	info		6.3.0-1	The GNU FORTRAN Compiler
☐ mingw32-gcc-fortran	man		6.3.0-1	The GNU FORTRAN Compiler

The GNU FORTRAN Compiler

This package provides the MinGW implementation of the GNU FORTRAN language compiler.

This is an optional component of the MinGW Compiler Suite; you require it only if you wish to compile programs written in the FORTRAN language.

Step 5: On the top of the window click on install and press apply changes. Then click on “Appl” on the pop- up



Schedule of Pending Actions

Okay to proceed?

The package changes itemised below will be implemented when you choose "Apply"

0 installed packages will be removed

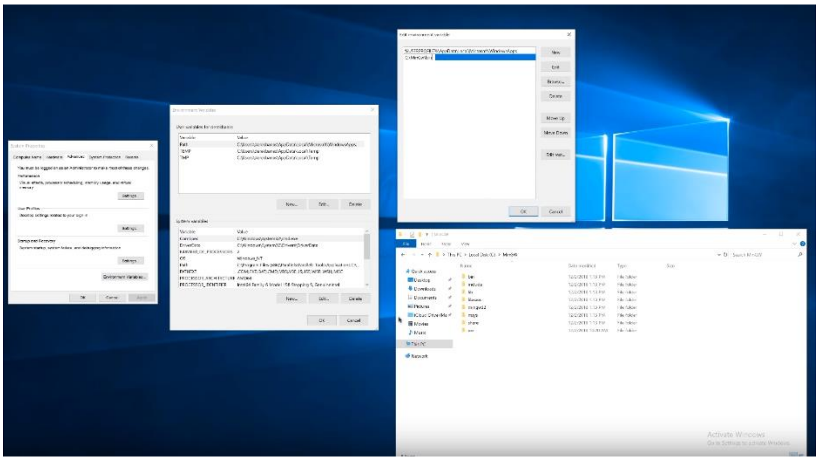
0 installed packages will be upgraded

118 new/upgraded packages will be installed

```
libiconv-1.14-3-mingw32-dll-2.tar
libpthreadgc-2.10-mingw32-pre-20160821-1-dll-3.tar.xz
mingw32-libgomp-deps-5.3.0-mingw32-dll.meta
libgcc-6.3.0-1-mingw32-dll-1.tar.xz
libintl-0.18.3.2-2-mingw32-dll-8.tar.xz
mingwrt-5.0.2-mingw32-dll.tar.xz
```

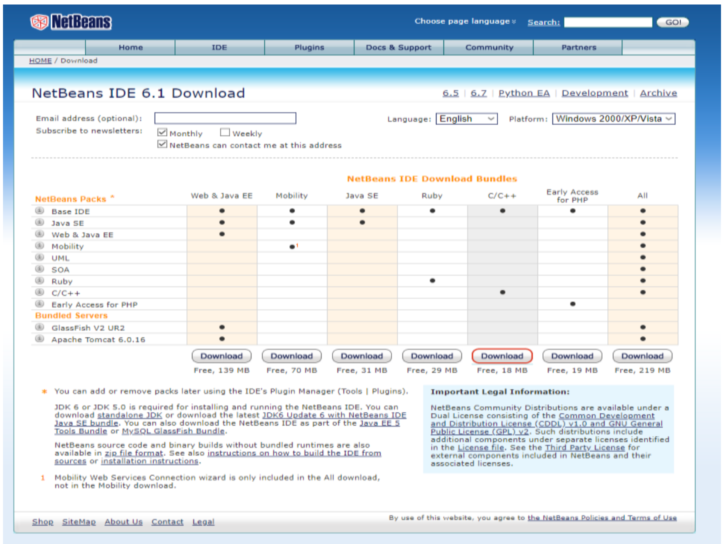
Step 6: Go to environment variables. On the user variables, click “Path” and click “edit” and click “New”.

Type “C:\MinGW\bin” click on “OK”



%USERPROFILE%\AppData\Local\Microsoft\WindowsApps
C:\MinGW\bin

Step 7: Download NetBeans IDE for C/C++ and install it.



NetBeans IDE 6.1 Download

Email address (optional):

Subscribe to newsletters: ☑ Monthly ☐ Weekly

☑ NetBeans can contact me at this address

Language: English

Platform: Windows 2000/XP/Vista

NetBeans IDE Download Bundles

NetBeans Packs *	Web & Java EE	Mobility	Java SE	Ruby	C/C++	Early Access for PHP	All
Base IDE	●	●	●	●	●	●	●
Java SE	●	●	●				●
Web & Java EE	●						●
Mobility		●[1]					●
UML							●
SOA							●
Ruby				●			●
C/C++					●		●
Early Access for PHP						●	
Bundled Servers							
GlassFish V2 UR2	●						●
Apache Tomcat 6.0.16	●						●
	Download	Download	Download	Download	Download	Download	Download
	Free, 139 MB	Free, 70 MB	Free, 31 MB	Free, 29 MB	Free, 18 MB	Free, 19 MB	Free, 219 MB

* You can add or remove packs later using the IDE's Plugin Manager (Tools | Plugins).

JDK 6 or JDK 5.0 is required for installing and running the NetBeans IDE. You can download standalone JDK or download the latest JDK6 Update 6 with NetBeans IDE Java SE bundle. You can also download the NetBeans IDE as part of the Java EE 5 Tools Bundle or MySQL GlassFish Bundle.

NetBeans source code and binary builds without bundled runtimes are also available in zip file format. See also instructions on how to build the IDE from sources or installation instructions.

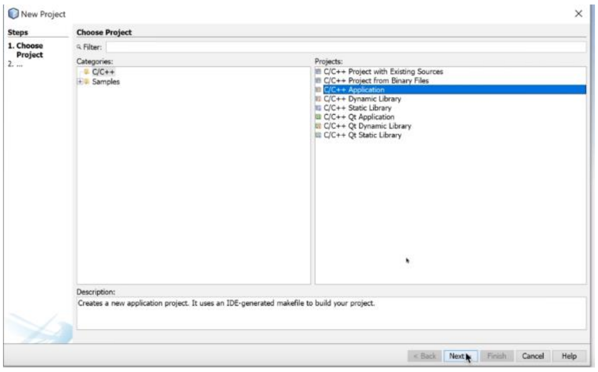
[1] Mobility Web Services Connection wizard is only included in the All download, not in the Mobility download.

Important Legal Information:

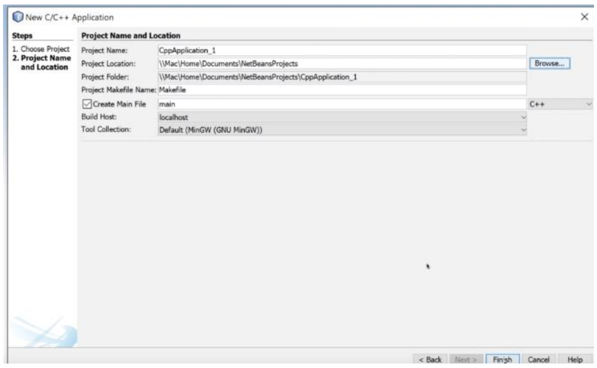
NetBeans Community Distributions are available under a Dual License consisting of the Common Development and Distribution License (CDDL) v1.0 and GNU General Public License (GPL) v2. Such distributions include additional components under separate licenses identified in the License file. See the Third Party License for external components included in NetBeans and their associated licenses.

By use of this website, you agree to the NetBeans Policies and Terms of Use

Step 8: Start a new project on NetBeans IDE. Select C/C++ Application. Click next, on the next page leave everything as suggested and click Finish.

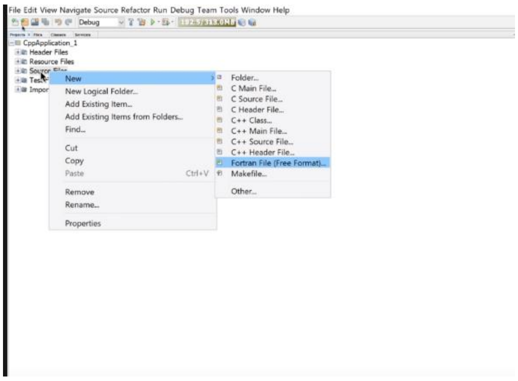


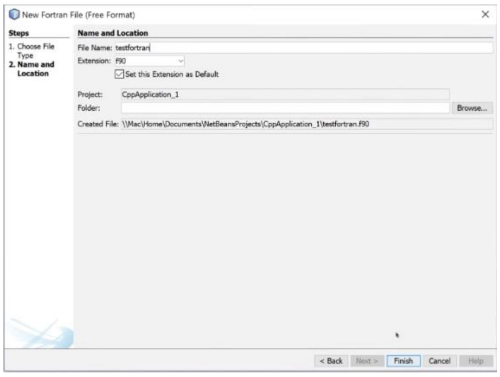
Creates a new application project. It uses an IDE-generated makefile to build your project.



Project Name: CppApplication_1
Project Location: \\Mac\Home\Documents\NetBeansProjects
Project Folder: \\Mac\Home\Documents\NetBeansProjects\CppApplication_1
Project Makefile Name: Makefile
☑ Create Main File: main C++
Build Host: localhost
Tool Collection: Default (MinGW (GNU MinGW))

Step 9: Right click on Source files, select Fortran File under New. Give a name as whatever you want, click on finish. Then you are ready to code in Fortran.

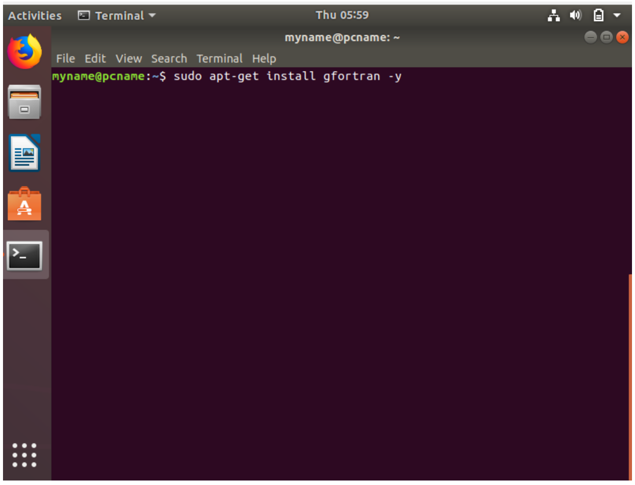




File Name: testfortran
Extension: f90
☑ Set this Extension as Default
Project: CppApplication_1
Folder:
Created File: \\Mac\Home\Documents\NetBeansProjects\CppApplication_1\testfortran.f90

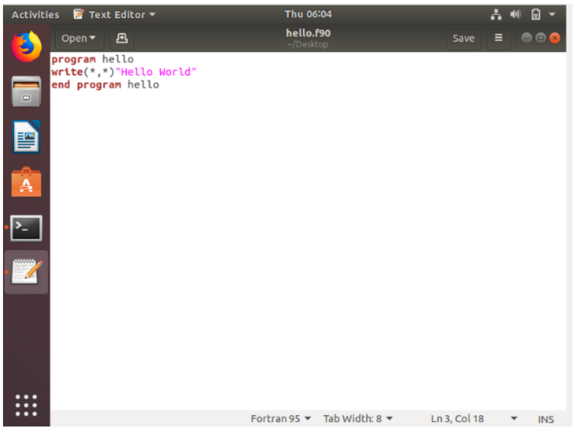
In Linux:

Step 1: Open terminal and write “sudo apt-get install gfortran -y” then enter your password.



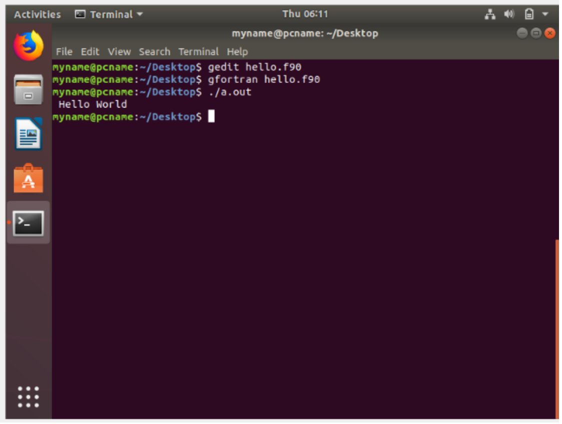
```
myname@pcname:~$ sudo apt-get install gfortran -y
```

Step 2: Open a text editor named “hello.f90 ” (before dot you can give any name) and write as below.



```
program hello
write(*,*)"Hello World"
end program hello
```

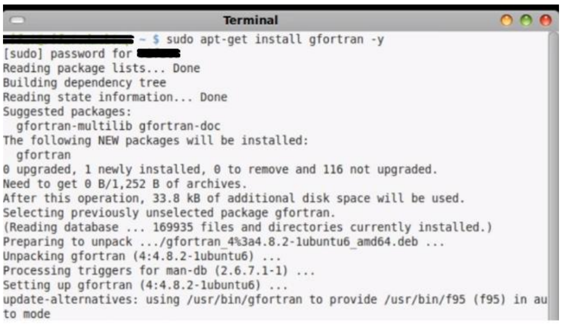
Step 3: To compile your program type “gfortran hello.f90 ” in terminal. And type “./a.out” to run the program.



```
myname@pcname:~/Desktop$ gedit hello.f90
myname@pcname:~/Desktop$ gfortran hello.f90
myname@pcname:~/Desktop$ ./a.out
 Hello World
myname@pcname:~/Desktop$ 
```

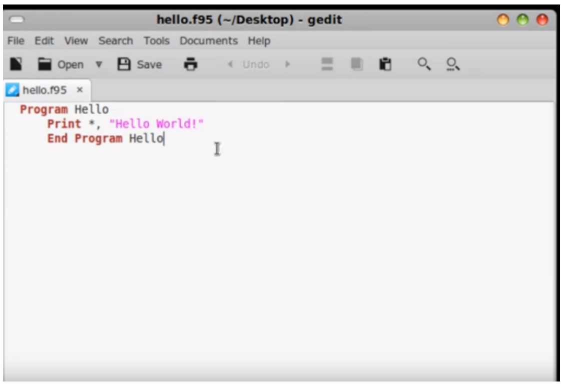
On Mac:

Step 1: Open terminal and write “sudo apt-get install gfortran -y” then enter your password.



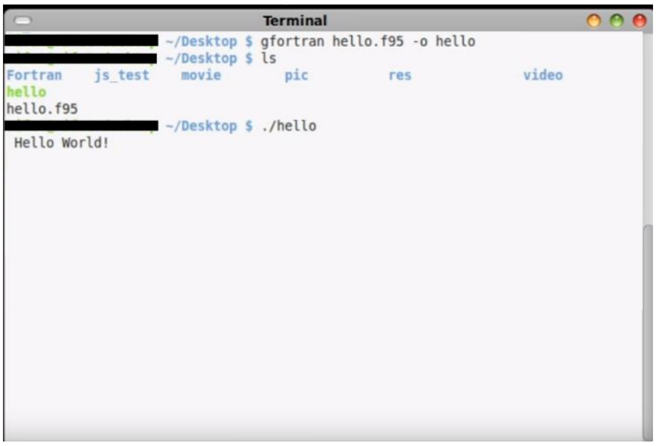
```
[redacted] ~ $ sudo apt-get install gfortran -y
[sudo] password for [redacted]
Reading package lists... Done
Building dependency tree
Reading state information... Done
Suggested packages:
  gfortran-multilib gfortran-doc
The following NEW packages will be installed:
  gfortran
0 upgraded, 1 newly installed, 0 to remove and 116 not upgraded.
Need to get 0 B/1,252 B of archives.
After this operation, 33.8 kB of additional disk space will be used.
Selecting previously unselected package gfortran.
(Reading database ... 169935 files and directories currently installed.)
Preparing to unpack .../gfortran_4%3a4.8.2-1ubuntu6_amd64.deb ...
Unpacking gfortran (4:4.8.2-1ubuntu6) ...
Processing triggers for man-db (2.6.7.1-1) ...
Setting up gfortran (4:4.8.2-1ubuntu6) ...
update-alternatives: using /usr/bin/gfortran to provide /usr/bin/f95 (f95) in au
to mode
```

Step 2: Open a text editor named “hello.f95 ” (before dot you can give any name) and write as below.


hello.f95 (~/Desktop) - gedit
File Edit View Search Tools Documents Help
Open
Save
Undo
hello.f95

```
  Program Hello
      Print *, "Hello World!"
      End Program Hello
```

Step 3: To compile your program type “gfortran hello.f95 -o hello ” in terminal. And type “./hello” to run the program.



```
[redacted] ~/Desktop $ gfortran hello.f95 -o hello
[redacted] ~/Desktop $ ls
Fortran     js_test    movie        pic          res              video
hello
hello.f95
[redacted] ~/Desktop $ ./hello
 Hello World!
```

Example codes

Example 1:

```
program hello
        print *, "Hello World!"
     end program hello
```

Example 2: Greatest Common Divisor in Fortran 77

```
PROGRAM EUCLID
        PRINT *, 'A?'
        READ *, NA
        IF (NA.LE.0) THEN
          PRINT *, 'A must be a positive integer.'
          STOP
        END IF
        PRINT *, 'B?'
        READ *, NB
        IF (NB.LE.0) THEN
          PRINT *, 'B must be a positive integer.'
          STOP
        END IF
        PRINT *, 'The GCD of', NA, ' and', NB, ' is', NGCD(NA, NB), '.'
        STOP
      END

      FUNCTION NGCD(NA, NB)
        IA = NA
        IB = NB
    1   IF (IB.NE.0) THEN
          ITEMP = IA
          IA = IB
          IB = MOD(ITEMP, IB)
          GOTO 1
        END IF
        NGCD = IA
        RETURN
      END
```

Example 3: Object Oriented Programming

```
module test_module
  implicit none
  private
  integer, public :: a=1
  integer, public, protected :: b=1
  integer, private :: c=1
end module test_module

!> import all public data of test_module
program main
  use test_module

  print *, a, b
end program main

!> import all data, and rename
program main
  use test_module, better_name => a

  ! new name use available
  print *, better_name

  ! old name is not available anymore
  !print *, a  <- ERROR
end program main
```

```
!> import only a subset of the public data
program main
  use test_module, only : a

  ! only a is loaded
  print *, a

  ! b is not loaded
  !print *, b  <- ERROR
end program main
```

Example 4: Parallel Programming Example in Fortran90

```
use omp_lib

  implicit none

  integer ( kind = 4 ), parameter :: nv = 6

  integer ( kind = 4 ) i
  integer ( kind = 4 ) :: i4_huge = 2147483647
  integer ( kind = 4 ) j
  integer ( kind = 4 ) mind(nv)
  integer ( kind = 4 ) ohd(nv,nv)

  call timestamp ( )
  write ( *, '(a)' )  ' '
  write ( *, '(a)' )  'DIJKSTRA_OPENMP:'
  write ( *, '(a)' )  '  FORTRAN90 version'
  write ( *, '(a)' )  '  Use Dijkstra''s algorithm to determine the minimum'
  write ( *, '(a)' )  '  distance from node 1 to each node in a graph,'
  write ( *, '(a)' )  '  given the distances between each pair of nodes.'
  write ( *, '(a)' )  ' '
  write ( *, '(a)' )  '  Although a very small example is considered, we'
  write ( *, '(a)' )  '  demonstrate the use of OpenMP directives for'
  write ( *, '(a)' )  '  parallel execution.'
!
!  Initialize the problem data.
!
  call init ( nv, ohd )
!
!  Print the distance matrix.
!
  write ( *, '(a)' )  ' '
  write ( *, '(a)' )  '  Distance matrix:'
  write ( *, '(a)' )  ' '
  do i = 1, nv
    do j = 1, nv
      if ( ohd(i,j) == i4_huge ) then
        write ( *, '(2x,a)', advance = 'NO' ) 'Inf'
      else
        write ( *, '(2x,i3)', advance = 'NO' ) ohd(i,j)
      end if
    end do
    write ( *, '(a)', advance = 'yes' )
  end do
!
!  Carry out the algorithm.
!
  call dijkstra_distance ( nv, ohd, mind )
```

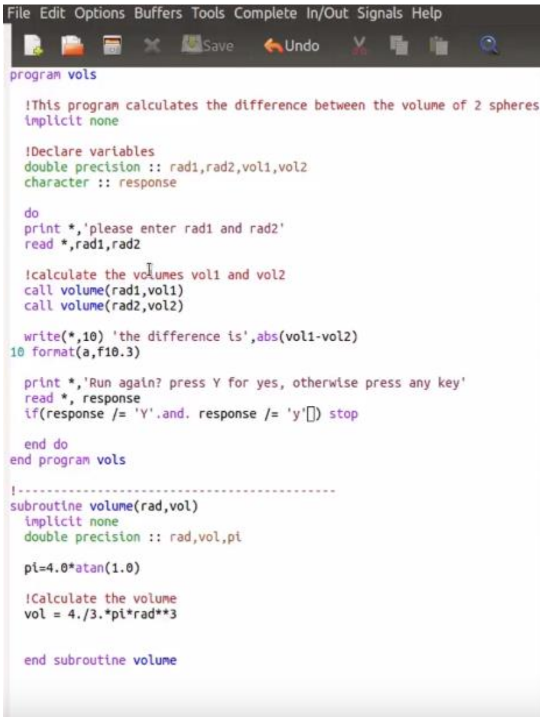
```
!
!  Print the results.
!
  write ( *, '(a)' )  ' '
  write ( *, '(a)' )  '  Minimum distances from node 1:'
  write ( *, '(a)' )  ' '
  do i = 1, nv
    write ( *, '(2x,i2,2x,i2)' ) i, mind(i)
  end do
!
!  Terminate.
!
  write ( *, '(a)' )  ' '
  write ( *, '(a)' )  'DIJKSTRA_OPENMP:'
  write ( *, '(a)' )  '  Normal end of execution.'

  write ( *, '(a)' )  ' '
  call timestamp ( )

  stop
end
```

Example 5: Scientific Programming

```
! ProjectileAir . f90 : Projectile Program using Dislin
 ! --------------------------------
 Subroutine buildPlot ( Npts , x0 , y0 , v0 , theta , k)
 Use d i s l i n
 Implicit None
 Integer , intent ( in ) :: Npts
 Real *8 , i n t e n t ( in ) : : x0 , y0 , v0 , theta , k
 Real *8 : : dt , g , vx , vy , xmax , ymax , t
 Real *8, dimension ( Npts ) :: aXValues , aYValues , nXValues ,
nYValues
 Integer :: i
 t = 0.
 g = 9.81
 dt = 2*v0* sin ( theta ) /( Npts *g )
 vx = v0* cos ( theta )
 vy = v0* sin ( theta )
 nXValues (1) = x0
 nYValues (1) = y0
 do i =1,Npts ,1
 t = (i -1)* d t
 ! Analytic
 aXValues ( i ) = x0+v0* cos ( theta ) * t
 aYValues ( i ) = y0+v0* sin ( theta ) * t-g* t * t /2.
 i f ( i >=2) then
 vx = vx-k*vx* d t
 vy = vy-g* dt-k*vy* d t
 ! Numberic
 nXValues ( i ) = nXValues ( i -1) + vx* d t
 nYValues ( i ) = nYValues ( i -1) + vy* d t
 end i f
 end do
 ! Dislin plotting routines
 call metafl (’XWIN’)
 call disini
 call name (’x-axis’ ,’X’)
 call name ('y-axis' ,'Y')
 call labdig(-1 ,'X')
 call ticks (10 ,'XY')
 call titlin (' Analytic (green) vs Numerical' ,1)
 xmax = 2. * v0 **2* cos ( theta ) * sin ( theta ) /g
 ymax = ( v0* sin ( theta )) * * 2/(2. * g )
 call graf ( x0 , x0+xmax , x0 , xmax / 10. , y0 , y0+ymax , y0 , ymax /10.)
 call title ()
 call color ('RED')
 call curve ( aXValues , aYValues , Npts )
 call color ('GREEN')
 call curve ( nXValues , nYValues , Npts )
 call color ('FORE')
 call dash
 call disfin
 End Subroutine buildPlot
 !
 Program projectileAir
 Implicit None
 ! Set-up all needed parameters
 Integer : : Npts = 200
 Real *8 : : x0 =0. , y0 =0. , v0 =20. , theta =3.14159265358979/4. , k =0.25
 call buildPlot ( Npts , x0 , y0 , v0 , theta , k)
 End Program projectileAir
```

Specific Things About Fortran

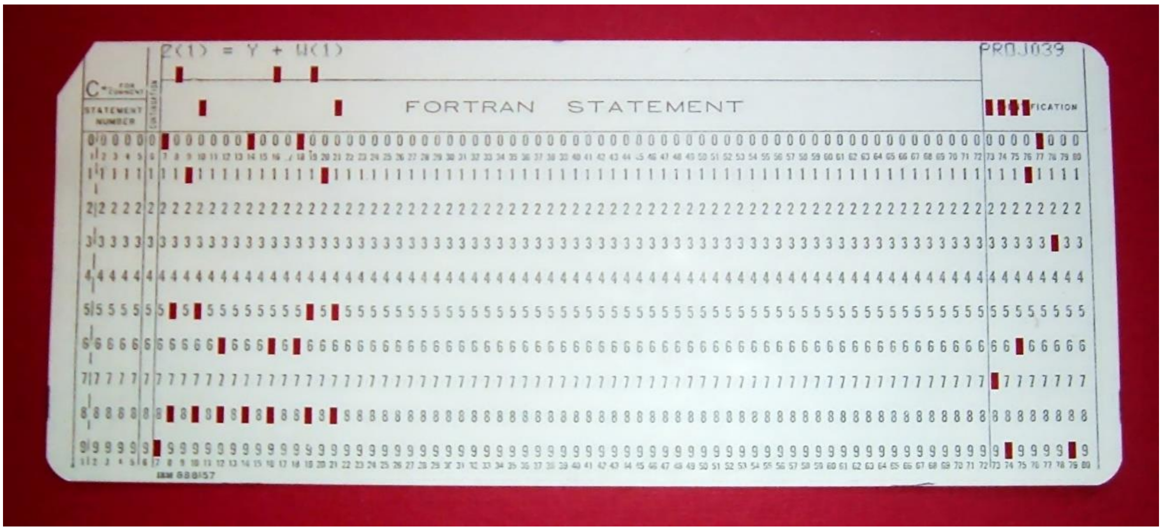
1.Subroutines: Subroutines are very useful to compute repetitive things. Subroutines are defined outside of the program. Unlike functions they return more than one thing. Or they do not return a value at all, just do an operation like swapping. In the example below, there is a subroutine that calculates volume difference between two spheres until user press any other key than 'Y' or 'y'.



File Edit Options Buffers Tools Complete In/Out Signals Help
Save
Undo

```
program vols

  !This program calculates the difference between the volume of 2 spheres
  implicit none

  !Declare variables
  double precision :: rad1,rad2,vol1,vol2
  character :: response

  do
  print *,'please enter rad1 and rad2'
  read *,rad1,rad2

  !calculate the volumes vol1 and vol2
  call volume(rad1,vol1)
  call volume(rad2,vol2)

  write(*,10) 'the difference is',abs(vol1-vol2)
10 format(a,f10.3)

  print *,'Run again? press Y for yes, otherwise press any key'
  read *, response
  if(response /= 'Y'.and. response /= 'y') stop

  end do
end program vols

!-------------------------------------------
subroutine volume(rad,vol)
  implicit none
  double precision :: rad,vol,pi

  pi=4.0*atan(1.0)

  !Calculate the volume
  vol = 4./3.*pi*rad**3


  end subroutine volume
```

2. Punch cards: Fortran was uses with punch-cards also, in its early times. Here is a punch card picture below.



3. Types: In Fortran, floating point numbers stored by “Real Type”. Means that they are represented with “real” keyword. Also, complex numbers can be stored with “complex” keyword. In older version of Fortran, there was a feature called “implicit typing”. you do not have to declare the variables before use. If a variable is not declared, then the first letter of its name will determine its type. Variable names starting with i, j, k, l, m, or n, are considered to be for integer variable and others are real variables. However, you must declare all the variables as it is good programming practice. For that you start your program with the “implicit none” statement.