

Programación en FORTRAN

Nivel Básico - Sesión 1

Martin Josemaría Vuelta Rojas

28 de enero de 2018

SoftButterfly

1. Preliminares
2. Introducción histórica
3. Entorno de desarrollo
4. Conceptos básicos
5. Formatos de escritura en Fortran
6. Estructura de un programa en Fortran
7. Expresividad: el algoritmo TPK

Preliminares

El curso básico de Fortran tiene por objetivo

El curso básico de Fortran tiene por objetivo

- Que obtengan un conocimiento sólido de las características de Fortran.

El curso básico de Fortran tiene por objetivo

- Que obtengan un conocimiento sólido de las características de Fortran.
- Que se familiarizen con el flujo de información en los programas desarrollados en Fortran.

El curso básico de Fortran tiene por objetivo

- Que obtengan un conocimiento sólido de las características de Fortran.
- Que se familiarizen con el flujo de información en los programas desarrollados en Fortran.
- Que tengan una introducción nuevas características de los estándares recientes Fortran 2003 y 2008.

El curso básico de Fortran tiene por objetivo

- Que obtengan un conocimiento sólido de las características de Fortran.
- Que se familiarizen con el flujo de información en los programas desarrollados en Fortran.
- Que tengan una introducción nuevas características de los estándares recientes Fortran 2003 y 2008.
- Que puedan contruir su entorno de darrollo para programar en Fortran cómodamente.

Metodología

Metodología

- Exposiciones dialogadas utilizando, apuntes, elementos de proyección fija.

Metodología

- Exposiciones dialogadas utilizando, apuntes, elementos de proyección fija.
- Talleres prácticos grupales en la elaboración de programas.

Metodología

- Exposiciones dialogadas utilizando, apuntes, elementos de proyección fija.
- Talleres prácticos grupales en la elaboración de programas.
- Las clases serán eminentemente prácticas en una relación 70% práctica - 30% teórica.

Referencias

Referencias

- I. Chivers, J. Sleightholme, *Introduction to Programming with Fortran. With Coverage of Fortran 90, 95, 2003, 2008 and 77*, Springer-Verlag London, 2012.

Referencias

- I. Chivers, J. Sleightholme, *Introduction to Programming with Fortran. With Coverage of Fortran 90, 95, 2003, 2008 and 77*, Springer-Verlag London, 2012.
- Michael Metcalf, John Reid, Malcolm Cohen, *Modern Fortran Explained*, Oxford University Press, USA 2011

Referencias

- I. Chivers, J. Sleightholme, *Introduction to Programming with Fortran. With Coverage of Fortran 90, 95, 2003, 2008 and 77*, Springer-Verlag London, 2012.
- Michael Metcalf, John Reid, Malcolm Cohen, *Modern Fortran Explained*, Oxford University Press, USA 2011
- Morten Hjorth-Jensen, *Computational Physics, Lecture Notes Fall 2015*, Department of Physics, University of Oslo. 2015.

Materiales

Materiales

Repositorio del curso

Materiales

Repositorio del curso

 <https://github.com/zodiacfireworks/course-fortran-basic>

Materiales

Repositorio del curso

- 🔗 <https://github.com/zodiacfireworks/course-fortran-basic>
- 🔗 <https://github.com/zodiacfireworks/course-fortran-intermediate>

Materiales

Repositorio del curso

- 🔗 <https://github.com/zodiacfireworks/course-fortran-basic>
- 🔗 <https://github.com/zodiacfireworks/course-fortran-intermediate>

¡Importante!

Materiales

Repositorio del curso

🔗 <https://github.com/zodiacfireworks/course-fortran-basic>

🔗 <https://github.com/zodiacfireworks/course-fortran-intermediate>

¡Importante!

Cada alumno debera tener una cuenta en GitHub

Materiales

Repositorio del curso

- 🔗 <https://github.com/zodiacfireworks/course-fortran-basic>
- 🔗 <https://github.com/zodiacfireworks/course-fortran-intermediate>

¡Importante!

Cada alumno debera tener una cuenta en GitHub

- 🔗 <https://github.com>

Martín Josemaría Vuelta Rojas

Martín Josemaría Vuelta Rojas

- Software Developer

Martín Josemaría Vuelta Rojas

- Software Developer
- Web Developer

Martín Josemaría Vuelta Rojas

- Software Developer
- Web Developer
- Investigador

Martín Josemaría Vuelta Rojas

- Software Developer
- Web Developer
- Investigador
- Programador

Martín Josemaría Vuelta Rojas

- Software Developer
- Web Developer
- Investigador
- Programador
 - En investigación:

Martín Josemaría Vuelta Rojas

- Software Developer
- Web Developer
- Investigador
- Programador
 - En investigación:
 - C, C++, Fortran, Python, R, Julia, Mathematica, Matlab, LaTeX

Martín Josemaría Vuelta Rojas

- Software Developer
- Web Developer
- Investigador
- Programador
 - En investigación:
 - C, C++, Fortran, Python, R, Julia, Mathematica, Matlab, LaTeX
 - En web:

Martín Josemaría Vuelta Rojas

- Software Developer
- Web Developer
- Investigador
- Programador
 - En investigación:
 - C, C++, Fortran, Python, R, Julia, Mathematica, Matlab, LaTeX
 - En web:
 - HTML, CSS, JavaScript, Python

Martín Josemaría Vuelta Rojas

- Software Developer
- Web Developer
- Investigador
- Programador
 - En investigación:
 - C, C++, Fortran, Python, R, Julia, Mathematica, Matlab, LaTeX
 - En web:
 - HTML, CSS, JavaScript, Python
 - En mobile:

Martín Josemaría Vuelta Rojas

- Software Developer
- Web Developer
- Investigador
- Programador
 - En investigación:
 - C, C++, Fortran, Python, R, Julia, Mathematica, Matlab, LaTeX
 - En web:
 - HTML, CSS, JavaScript, Python
 - En mobile:
 - Kotlin, Java, C++

Martín Josemaría Vuelta Rojas

- Software Developer
- Web Developer
- Investigador
- Programador
 - En investigación:
 - C, C++, Fortran, Python, R, Julia, Mathematica, Matlab, LaTeX
 - En web:
 - HTML, CSS, JavaScript, Python
 - En mobile:
 - Kotlin, Java, C++
 - Hobbie:

Martín Josemaría Vuelta Rojas

- Software Developer
- Web Developer
- Investigador
- Programador
 - En investigación:
 - C, C++, Fortran, Python, R, Julia, Mathematica, Matlab, LaTeX
 - En web:
 - HTML, CSS, JavaScript, Python
 - En mobile:
 - Kotlin, Java, C++
 - Hobbie:
 - Scala, Pixie, Clojure, Elixir, Haskell, Oz, Kotlin, ...

Martín Josemaría Vuelta Rojas

Martín Josemaría Vuelta Rojas

- SoftButterfly

Martín Josemaría Vuelta Rojas

- SoftButterfly
- HackSpace Perú

Martín Josemaría Vuelta Rojas

- SoftButterfly
- HackSpace Perú
- Jupyter Notebook

Martín Josemaría Vuelta Rojas

- SoftButterfly
- HackSpace Perú
- Jupyter Notebook
- Fedora

Martín Josemaría Vuelta Rojas

- SoftButterfly
- HackSpace Perú
- Jupyter Notebook
- Fedora
- GNOME

Martín Josemaría Vuelta Rojas

- SoftButterfly
- HackSpace Perú
- Jupyter Notebook
- Fedora
- GNOME
- UNMSM

Martín Josemaría Vuelta Rojas

Martín Josemaría Vuelta Rojas

📱 +51 982 042 088

Martín Josemaría Vuelta Rojas

 +51 982 042 088

 martin.vuelta@gmail.com

Martín Josemaría Vuelta Rojas

 +51 982 042 088

 martin.vuelta@gmail.com

 martinvuelta

Martín Josemaría Vuelta Rojas

 +51 982 042 088

 martin.vuelta@gmail.com

 martinvuelta

 zodiacfireworks

Introducción histórica

En el origen ...

En el origen ...

- Código máquina en notación octal

En el origen ...

- Código máquina en notación octal
- Conocimiento muy detallado del hardware

En el origen ...

- Código máquina en notación octal
- Conocimiento muy detallado del hardware

A inicios de los 50s

En el origen ...

- Código máquina en notación octal
- Conocimiento muy detallado del hardware

A inicios de los 50s

- Assembler

En el origen ...

- Código máquina en notación octal
- Conocimiento muy detallado del hardware

A inicios de los 50s

- Assembler
- Menos laborioso que el código máquina

En el origen ...

- Código máquina en notación octal
- Conocimiento muy detallado del hardware

A inicios de los 50s

- Assembler
- Menos laborioso que el código máquina
- Conocimiento detallado del hardware

En el origen ...

- Código máquina en notación octal
- Conocimiento muy detallado del hardware

A inicios de los 50s

- Assembler
- Menos laborioso que el código máquina
- Conocimiento detallado del hardware

El panorama general ...

En el origen ...

- Código máquina en notación octal
- Conocimiento muy detallado del hardware

A inicios de los 50s

- Assembler
- Menos laborioso que el código máquina
- Conocimiento detallado del hardware

El panorama general ...

- Conocimiento del hardware

En el origen ...

- Código máquina en notación octal
- Conocimiento muy detallado del hardware

A inicios de los 50s

- Assembler
- Menos laborioso que el código máquina
- Conocimiento detallado del hardware

El panorama general ...

- Conocimiento del hardware
- Facilidad de cometer errores

En el origen ...

- Código máquina en notación octal
- Conocimiento muy detallado del hardware

A inicios de los 50s

- Assembler
- Menos laborioso que el código máquina
- Conocimiento detallado del hardware

El panorama general ...

- Conocimiento del hardware
- Facilidad de cometer errores
- Encontrar los errores en los programas era bastante difícil

Génesis 1953 ...

Génesis 1953 ...

John Backus envía una carta a su jefe en IBM pidiendo permiso para investigar una *mejor forma* de programar las computadoras. La carta contenía un esboo de proyecto con un tiempo de desarrollo de 6 meses.

Génesis 1953 ...

John Backus envía una carta a su jefe en IBM pidiendo permiso para investigar una *mejor forma* de programar las computadoras. La carta contenía un esbozo de proyecto con un tiempo de desarrollo de 6 meses.

Así empezó el proyecto que daría origen a Fortran

"The project completion was always six months away!"
John Backus

1957

1957

En febrero FORTRAN, el primer lenguaje de programación de alto nivel, fue anunciado al mundo por John Backus y su equipo de IBM en la Western Joint Computer Conference celebrada en Los Ángeles.

1957

En febrero FORTRAN, el primer lenguaje de programación de alto nivel, fue anunciado al mundo por John Backus y su equipo de IBM en la Western Joint Computer Conference celebrada en Los Ángeles.

A mediados de abril de 1957 tuvo lugar la primera entrega del compilador de FORTRAN para IBM 704 a Westinghouse Bettis para su uso en el diseño de reactores nucleares.

Las versiones de FORTRAN

Las versiones de FORTRAN

1957 FORTRAN I

Las versiones de FORTRAN

1957 FORTRAN I

1958 FORTRAN II

Las versiones de FORTRAN

1957 FORTRAN I

1958 FORTRAN II

1958 FORTRAN III (No disponible al público)

Las versiones de FORTRAN

1957 FORTRAN I

1958 FORTRAN II

1958 FORTRAN III (No disponible al público)

1961 FORTRAN IV (Una versión mejorada de FORTRAN II)

Algoritmo TPK en FORTRAN 0

```
1      DIMENSION A(11)
2      READ A
3      2  DO 3,8,11 J=1,11
4      3  I=11-J
5          Y=SQRT(ABS(A(I+1)))+5*A(I+1)**3
6      IF (400>=Y) 8,4
7      4  PRINT I,999.
8      GOTO 2
9      8  PRINT I,Y
10     11 STOP
```

Algoritmo TPK en FORTRAN I

```
1  C      THE TPK ALGORITHM
2  C      FORTRAN I STYLE
3          FUNF(T)=SQRTF(ABSF(T))+5.0*T**3
4          DIMENSION A(11)
5  1      FORMAT(6F12.4)
6          READ 1,A
7          DO 10 J=1,11
8              I=11-J
9              Y=FUNF(A(I+1))
10         IF(400.0-Y)4,8,8
11     4      PRINT 5,I
12     5      FORMAT(I10,10H TOO LARGE)
13         GOTO 10
14     8      PRINT 9,I,Y
15     9      FORMAT(I10,F12.7)
16    10      CONTINUE
17         STOP 52525
```


1962 El primer comité de estandarización de la ASA (Ahora ANSI) se reúne.

La estandarización

- 1962 El primer comité de estandarización de la ASA (Ahora ANSI) se reúne.
- 1966 Publicación del ANSI X3.91966 (FORTRAN 66), el primer estándar.

- 1962 El primer comité de estandarización de la ASA (Ahora ANSI) se reune.
- 1966 Publicación del ANSI X3.91966 (FORTRAN 66), el primer estándar.
- 1978 Publicación del ANSI X3.91978 (FORTRAN 77), también publicado como ISO 1539:1980.

- 1962 El primer comité de estandarización de la ASA (Ahora ANSI) se reune.
- 1966 Publicación del ANSI X3.91966 (FORTRAN 66), el primer estándar.
- 1978 Publicación del ANSI X3.91978 (FORTRAN 77), también publicado como ISO 1539:1980.
- 1991 ISO/IEC 1539:1991 (Fortran 90)

- 1962 El primer comité de estandarización de la ASA (Ahora ANSI) se reúne.
- 1966 Publicación del ANSI X3.91966 (FORTRAN 66), el primer estándar.
- 1978 Publicación del ANSI X3.91978 (FORTRAN 77), también publicado como ISO 1539:1980.
- 1991 ISO/IEC 1539:1991 (Fortran 90)
- 1997 ISO/IEC 15391:1997 (Fortran 95)

- 1962 El primer comité de estandarización de la ASA (Ahora ANSI) se reúne.
- 1966 Publicación del ANSI X3.91966 (FORTRAN 66), el primer estándar.
- 1978 Publicación del ANSI X3.91978 (FORTRAN 77), también publicado como ISO 1539:1980.
- 1991 ISO/IEC 1539:1991 (Fortran 90)
- 1997 ISO/IEC 15391:1997 (Fortran 95)
- 2004 ISO/IEC 15391:2004 (Fortran 2003)

- 1962 El primer comité de estandarización de la ASA (Ahora ANSI) se reúne.
- 1966 Publicación del ANSI X3.91966 (FORTRAN 66), el primer estándar.
- 1978 Publicación del ANSI X3.91978 (FORTRAN 77), también publicado como ISO 1539:1980.
- 1991 ISO/IEC 1539:1991 (Fortran 90)
- 1997 ISO/IEC 15391:1997 (Fortran 95)
- 2004 ISO/IEC 15391:2004 (Fortran 2003)
- 2010 ISO/IEC 15391:2010 (Fortran 2008)

Algoritmo TPK en FORTRAN 77

```
1      PROGRAM TPK
2      C      THE TPK ALGORITHM
3      C      FORTRAN 77 STYLE
4      REAL A(0:10)
5      READ (5,*) A
6      DO 10 I = 10, 0, -1
7          Y = FUN(A(I))
8          IF ( Y .LT. 400) THEN
9              WRITE(6,9) I,Y
10             FORMAT(I10,F12.6)
11         ELSE
12             WRITE (6,5) I
13             FORMAT(I10,' TOO LARGE')
14         ENDIF
15     10 CONTINUE
16     END
17
18     REAL FUNCTION FUN(T)
19     REAL T
20     FUN = SQRT(ABS(T)) + 5.0*T**3
21     END
```

Algoritmo TPK en FORTRAN 90

```
1      PROGRAM TPK
2      !      The TPK Algorithm
3      !      Fortran 90 style
4      IMPLICIT NONE
5      INTEGER          :: I
6      REAL             :: Y
7      REAL, DIMENSION(0:10) :: A
8      READ (*,*) A
9      DO I = 10, 0, -1      ! Backwards
10         Y = FUN(A(I))
11         IF ( Y < 400.0 ) THEN
12             WRITE(*,*) I, Y
13         ELSE
14             WRITE(*,*) I, ' Too large'
15         END IF
16     END DO
17     CONTAINS      ! Local function
18     FUNCTION FUN(T)
19     REAL :: FUN
20     REAL, INTENT(IN) :: T
21     FUN = SQRT(ABS(T)) + 5.0*T**3
22     END FUNCTION FUN
23     END PROGRAM TPK
```

Algoritmo TPK en FORTRAN 95

```
1      module Functions
2      public :: fun
3      contains
4          function fun(t) result (r)
5              real, intent(in) :: t
6              real :: r
7              r = sqrt(abs(t)) + 5.0*t**3
8          end function fun
9      end module Functions
10
11     program TPK
12     !     The TPK Algorithm
13     !     F95 style
14     use Functions
15     integer :: i
16     real :: y
17     real, dimension(0:10) :: a
18     read *, a
19     !     Continua
```

Algoritmo TPK en FORTRAN 95

```
20  !      Continuacion
21  do i = 10, 0, -1      ! Backwards
22      y = fun(a(i))
23      if ( y < 400.0 ) then
24          print *, i, y
25      else
26          print *, i, " Too large"
27      end if
28  end do
29  end program TPKTPK
```

"I don't know what the programming language of the year 2000 will look like, but I know it will be called FORTRAN."

Charles Anthony Richard Hoare

Circa 1982

Aplicaciones

Aplicaciones

- Predicción del clima

Aplicaciones

- Predicción del clima
- Análisis de datos de sísmicos para la exploración de depósitos de gas y petróleo

Aplicaciones

- Predicción del clima
- Análisis de datos de sísmicos para la exploración de depósitos de gas y petróleo
- Análisis financiero

Aplicaciones

- Predicción del clima
- Análisis de datos de sísmicos para la exploración de depósitos de gas y petróleo
- Análisis financiero
- Simulación de choques vehiculares

Aplicaciones

- Predicción del clima
- Análisis de datos de sísmicos para la exploración de depósitos de gas y petróleo
- Análisis financiero
- Simulación de choques vehiculares
- Análisis de datos de sondas espaciales

Aplicaciones

- Predicción del clima
- Análisis de datos de sísmicos para la exploración de depósitos de gas y petróleo
- Análisis financiero
- Simulación de choques vehiculares
- Análisis de datos de sondas espaciales
- Modelación de armas nucleares

Aplicaciones

- Predicción del clima
- Análisis de datos de sísmicos para la exploración de depósitos de gas y petróleo
- Análisis financiero
- Simulación de choques vehiculares
- Análisis de datos de sondas espaciales
- Modelación de armas nucleares
- Dinámica de fluidos computacionales

Aplicaciones

- Predicción del clima
- Análisis de datos de sísmicos para la exploración de depósitos de gas y petróleo
- Análisis financiero
- Simulación de choques vehiculares
- Análisis de datos de sondas espaciales
- Modelación de armas nucleares
- Dinámica de fluidos computacionales
- “Numerical Wind Tunnel”

- Intel

En la actualidad ...

- Intel
- IBM

En la actualidad ...

- Intel
- IBM
- NVIDIA

- **WG5:** <https://wg5-fortran.org/>

- **WG5:** <https://wg5-fortran.org/>
- **Fortran Wiki:** <http://fortranwiki.org/>

- **WG5:** <https://wg5-fortran.org/>
- **Fortran Wiki:** <http://fortranwiki.org/>
- **J3:** <http://www.j3-fortran.org/>

Entorno de desarrollo

Entorno de desarrollo

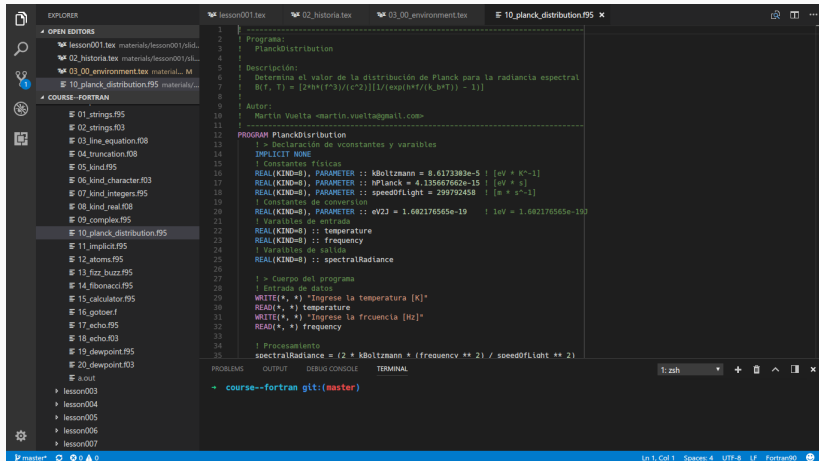


Figura 1: Entorno de desarrollo VS Code

Que tiene un entorno de desarrollo

Que tiene un entorno de desarrollo

- Editor de código

Que tiene un entorno de desarrollo

- Editor de código
- Compiladores o intérpretes

Que tiene un entorno de desarrollo

- Editor de código
- Compiladores o intérpretes
- Debugger

Que tiene un entorno de desarrollo

- Editor de código
- Compiladores o intérpretes
- Debugger
- Otras utilidades

Editor de código

Editor de código

- VS Code + Fortran Package [✓]

Editor de código

- VS Code + Fortran Package [✓]
- Sublime Text

Editor de código

- VS Code + Fortran Package [✓]
- Sublime Text
- Atom

Editor de código

- VS Code + Fortran Package [✓]
- Sublime Text
- Atom
- Vim

Editor de código

- VS Code + Fortran Package [✓]
- Sublime Text
- Atom
- Vim
- Emacs

Editor de código

- VS Code + Fortran Package [✓]
- Sublime Text
- Atom
- Vim
- Emacs
- ...

Compilador

Compilador

- GFortran [✓]

Compilador

- GFortran [✓]
- Intel

Compilador

- GFortran [✓]
- Intel
- IBM

Compilador

- GFortran [✓]
- Intel
- IBM
- Oracle

Compilador

- GFortran [✓]
- Intel
- IBM
- Oracle
- ...

Debuggers

Debuggers

- gdb [✓]

Debuggers

- gdb [✓]
- idb

Debuggers

- gdb [✓]
- idb
- ddd

Debuggers

- gdb [✓]
- idb
- ddd
- totalview

Debuggers

- gdb [✓]
- idb
- ddd
- totalview
- ...

VS Code en Ubuntu y derivados

VS Code en Ubuntu y derivados

- Descargar el paquete .deb desde
<https://code.visualstudio.com/download>

VS Code en Ubuntu y derivados

- Descargar el paquete .deb desde <https://code.visualstudio.com/download>
- Instalar el paquete desde la terminal

```
1 sudo dpkg -i <nombre del archivo>.deb
2 sudo apt-get install -f
```

Instalación de editor de código

VS Code en Ubuntu y derivados

- Descargar el paquete .deb desde <https://code.visualstudio.com/download>
- Instalar el paquete desde la terminal

```
1 sudo dpkg -i <nombre del archivo>.deb
2 sudo apt-get install -f
```

- Actualizar el paquete e instalar code

```
1 sudo apt-get update
2 sudo apt-get install code # o también "code-insiders"
```

VS Code en Fedora/CentOS

VS Code en Fedora/CentOS

- Descargar el paquete .rpm desde
<https://code.visualstudio.com/download>

VS Code en Fedora/CentOS

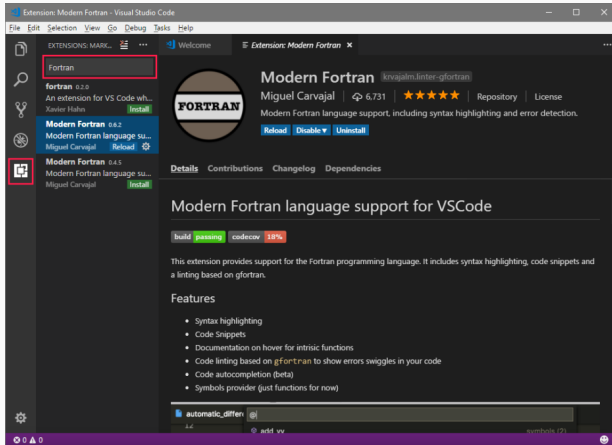
- Descargar el paquete .rpm desde <https://code.visualstudio.com/download>
- Instalar el paquete desde la terminal según la versión Fedora/CentOs (yum o dnf)

```
1 yum check-update
2 sudo yum install <nombre del archivo>.rpm
```

```
1 dnf check-update
2 sudo dnf install <nombre del archivo>.rpm
```

Fortran en VS Code

- En la pestaña 'Extensiones' (Ctrl+Shift+X), instalar 'Modern Fortran 0.6.2' encontrado con el buscador y recargar la extensión.



Extensión Modern Fortran 0.6.2 en VS Code

Compilador y debugger en Ubuntu y derivados

Compilador y debugger en Ubuntu y derivados

- Instalación de Gfortran

```
1 sudo apt-get install gfortran
```

Instalación de compilador y debugger

Compilador y debugger en Ubuntu y derivados

- **Instalación de Gfortran**

```
1 sudo apt-get install gfortran
```

- **Instalación del paquete binutils**

```
1 sudo apt-get update  
2 sudo apt-get install binutils
```

Instalación de compilador y debugger

Compilador y debugger en Ubuntu y derivados

- **Instalación de Gfortran**

```
1 sudo apt-get install gfortran
```

- **Instalación del paquete binutils**

```
1 sudo apt-get update  
2 sudo apt-get install binutils
```

- **Instalación del paquete build-essential**

```
1 sudo apt-get update  
2 sudo apt-get install build-essential
```

Compilador y debugger en Fedora/CentOS

Compilador y debugger en Fedora/CentOS

- Instalación de Gfortran

```
1 yum install gcc-gfortran
```


Compilador y debugger en Fedora/CentOS

- Instalación de Gfortran

```
1 yum install gcc-gfortran
```

- Instalación del paquete Development tools

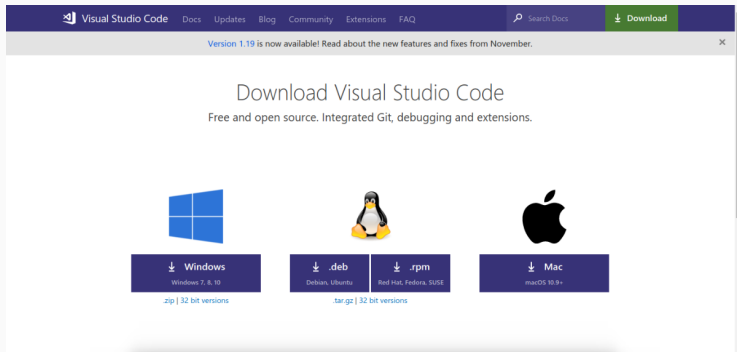
```
1 yum clean all  
2 yum groupinstall "Development tools"
```

Instalación de editor de código

VS Code en Windows

- Descargar el instalador desde

<https://code.visualstudio.com/download>



Página oficial de Visual Studio Code

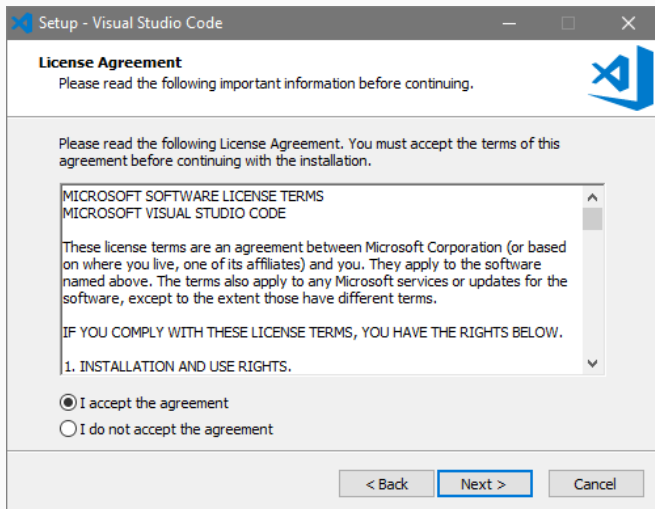
Instalación de editor de código

VS Code en Windows

- Iniciar el instalador y seguir los siguientes pasos:

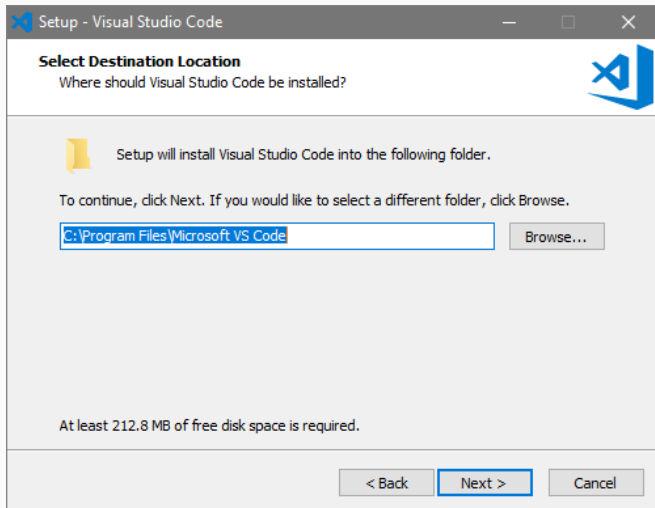


VS Code en Windows



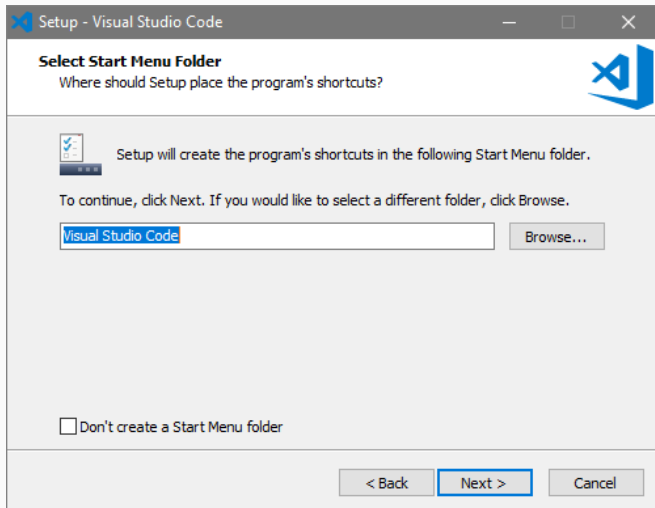
Instalación de editor de código

VS Code en Windows

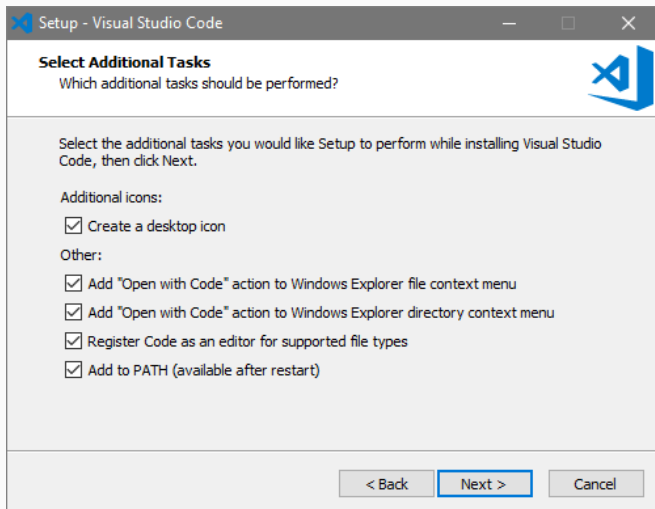


Instalación de editor de código

VS Code en Windows

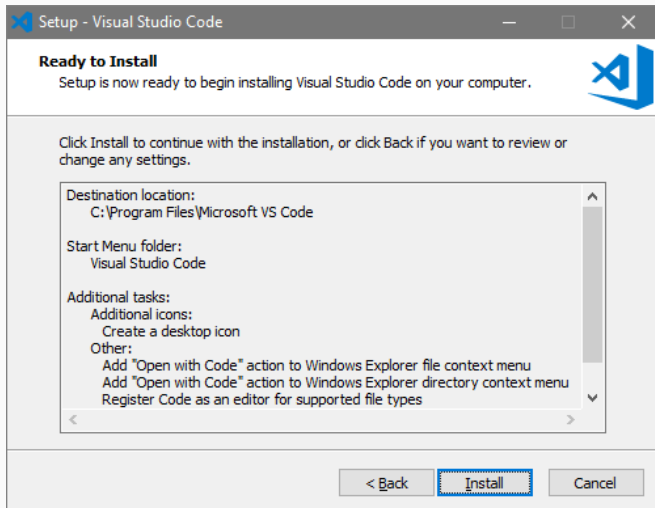


VS Code en Windows



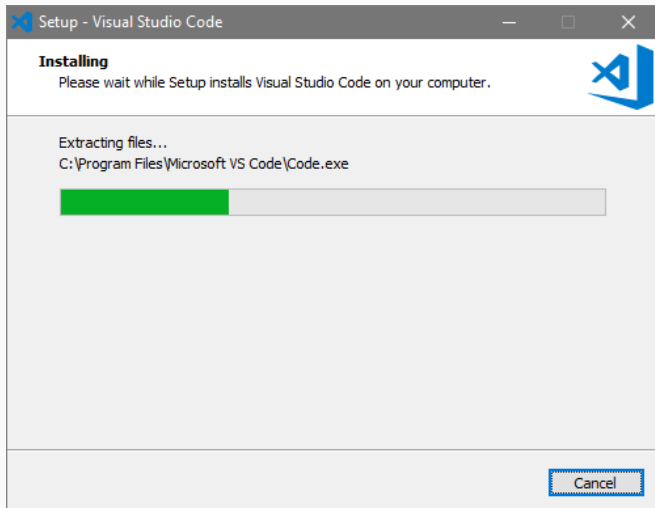
Instalación de editor de código

VS Code en Windows

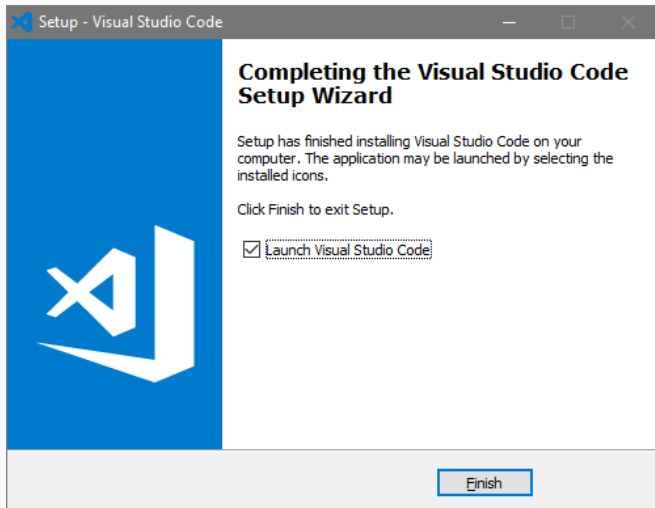


Instalación de editor de código

VS Code en Windows



VS Code en Windows



Instalación del compilador TDM-GCC

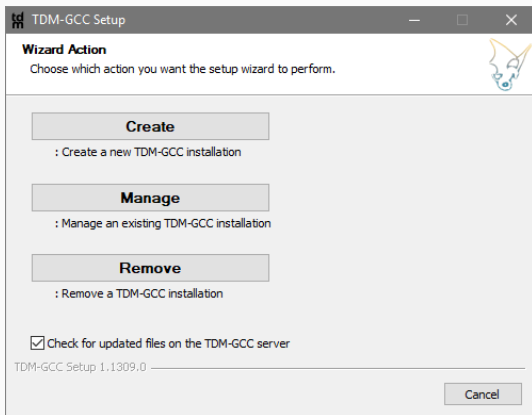
Instalación del compilador TDM-GCC

- Descargar el paquete TDM-GCC según la versión de windows (32-64 bits) en <http://tdm-gcc.tdragon.net/>

Instalación de compilador y debugger

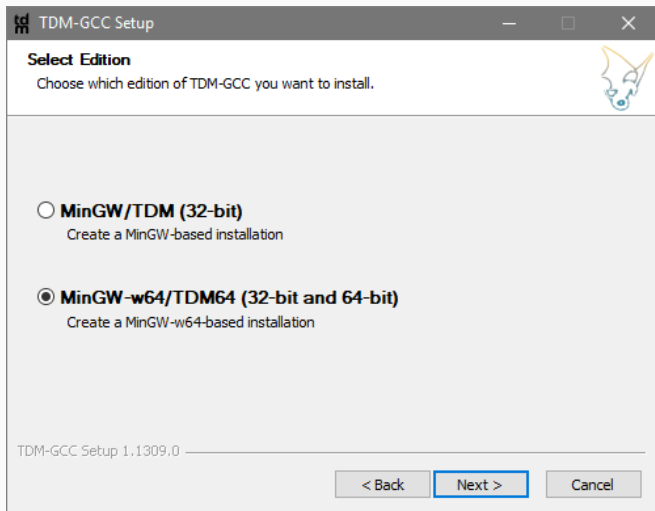
Instalación del compilador TDM-GCC

- Descargar el paquete TDM-GCC según la versión de windows (32-64 bits) en <http://tdm-gcc.tdragon.net/>
- Iniciar el instalador y seguir los siguientes pasos:



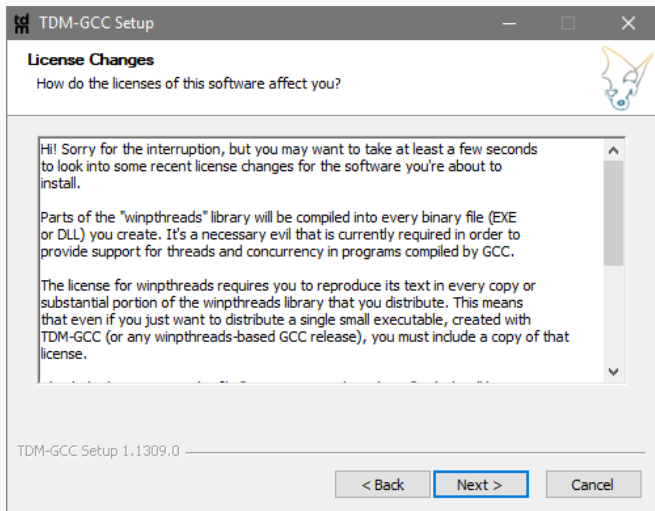
Instalación de compilador y debugger

Instalación del compilador TDM-GCC



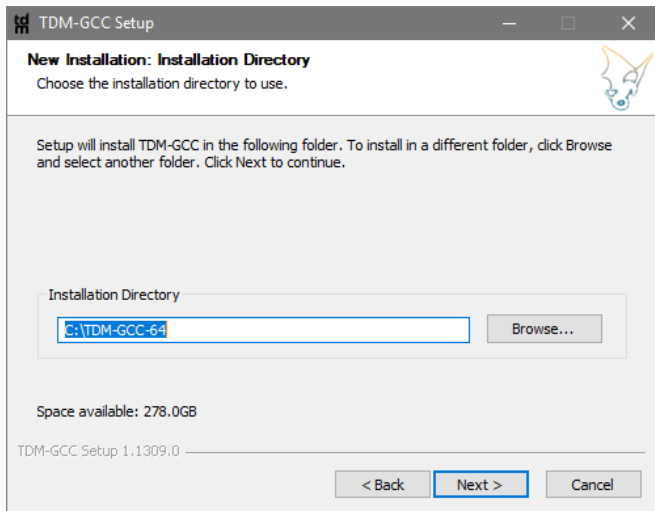
Instalación de compilador y debugger

Instalación del compilador TDM-GCC



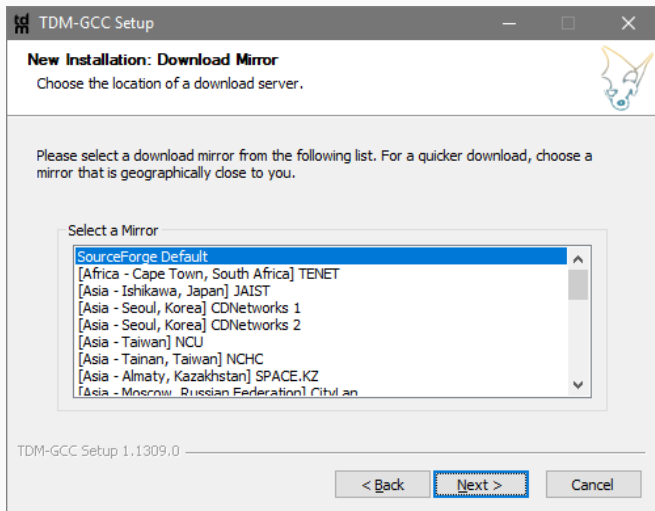
Instalación de compilador y debugger

Instalación del compilador TDM-GCC



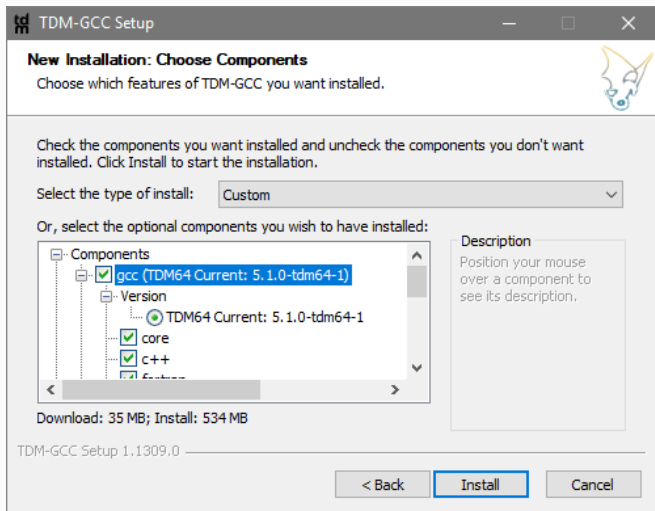
Instalación de compilador y debugger

Instalación del compilador TDM-GCC



Instalación de compilador y debugger

Instalación del compilador TDM-GCC



Download: 35 MB; Install: 534 MB

TDM-GCC Setup 1.1309.0

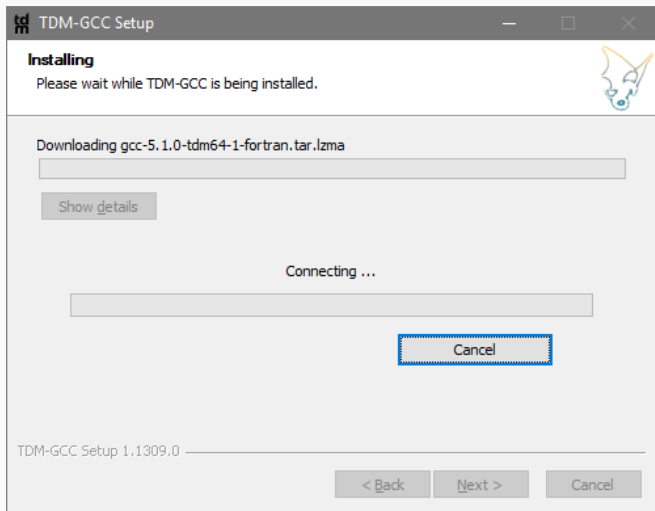
< Back

Install

Cancel

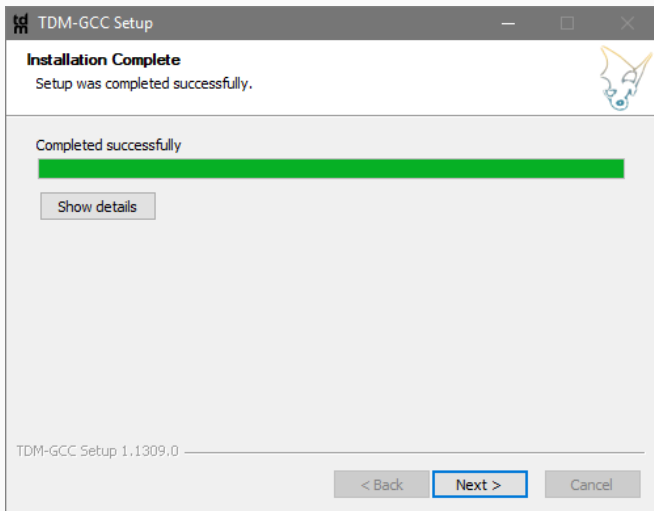
Instalación de compilador y debugger

Instalación del compilador TDM-GCC



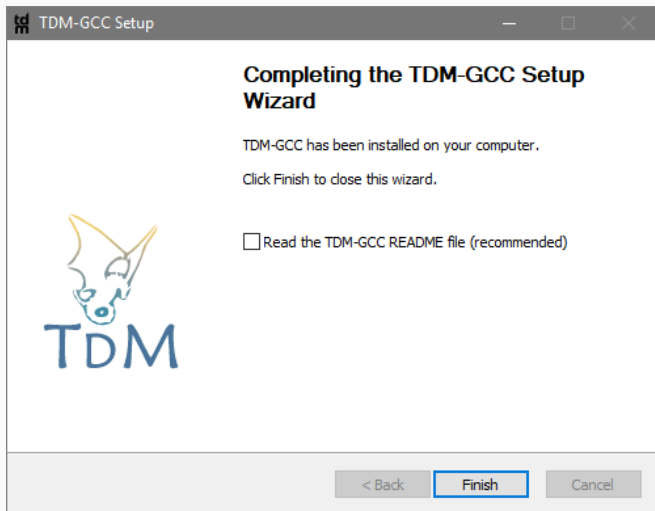
Instalación de compilador y debugger

Instalación del compilador TDM-GCC



Instalación de compilador y debugger

Instalación del compilador TDM-GCC



Conceptos básicos

Hello world! y el proceso de compilación

Hello world!

Hello world! y el proceso de compilación

Hello world!

- Se llama programa a un conjunto de instrucciones, realizadas computacionalmente en un tiempo determinado, aplicadas en la introducción, procesamiento o salida de datos.

Hello world! y el proceso de compilación

Hello world!

- Se llama programa a un conjunto de instrucciones, realizadas computacionalmente en un tiempo determinado, aplicadas en la introducción, procesamiento o salida de datos.
- Un programa en fortran tiene la siguiente forma:

Hello world! y el proceso de compilación

Hello world!

- Se llama programa a un conjunto de instrucciones, realizadas computacionalmente en un tiempo determinado, aplicadas en la introducción, procesamiento o salida de datos.
- Un programa en fortran tiene la siguiente forma:

```
1  PROGRAM hello_world
2      WRITE(*, *) Message
3  END PROGRAM hello_world
```

Hello world! y el proceso de compilación

Scientific Hello world!

Hello world! y el proceso de compilación

Scientific Hello world!

```
1  PROGRAM hello_world
2  IMPLICIT NONE
3
4  ! Angulo de entrada
5  REAL(KIND=4) :: theta
6
7  ! Resultado de aplicar la función seno
8  REAL(KIND=4) :: sin_of_theta
9
10 ! Mensaje
11 CHARACTER(len=*), PARAMETER :: Message = 'Hello World'
12
13 WRITE(*, *) 'Ingrese un ángulo [rad]: '
14 READ(*, *) theta
15 sin_of_theta = SIN(theta)
16
17 WRITE(*, *) Message
18 WRITE(*, *) "sin(", theta, ") = ", sin_of_theta
19 END PROGRAM hello_world
```


Proceso de compilación

- El proceso de compilación se puede resumir en dos pasos

Proceso de compilación

- El proceso de compilación se puede resumir en dos pasos
 - Compilación

Proceso de compilación

- El proceso de compilación se puede resumir en dos pasos
 - Compilación
 - Enlazado

Proceso de compilación

- El proceso de compilación se puede resumir en dos pasos
 - Compilación
 - Enlazado
- El proceso de compilación en fortran posee la siguiente sintaxis:

```
1  fcomp [options] file1 [file2] [...] [fileN]
```

Proceso de compilación

- El proceso de compilación se puede resumir en dos pasos
 - Compilación
 - Enlazado
- El proceso de compilación en fortran posee la siguiente sintaxis:

```
1  fcomp [options] file1 [file2] [...] [fileN]
```

- fcomp \Rightarrow denota el comando para llamar al compilador. (gfortran, ifort, ...)

Proceso de compilación

- El proceso de compilación se puede resumir en dos pasos
 - Compilación
 - Enlazado
- El proceso de compilación en fortran posee la siguiente sintaxis:

```
1  fcomp [options] file1 [file2] [...] [fileN]
```

- fcomp \Rightarrow denota el comando para llamar al compilador. (gfortran, ifort, ...)
- options \Rightarrow opciones que permite el compilador. (-o, -f, -c, ...)

Proceso de compilación

- El proceso de compilación se puede resumir en dos pasos
 - Compilación
 - Enlazado
- El proceso de compilación en fortran posee la siguiente sintaxis:

```
1  fcomp [options] file1 [file2] [...] [fileN]
```

- fcomp \Rightarrow denota el comando para llamar al compilador. (gfortran, ifort, ...)
- options \Rightarrow opciones que permite el compilador. (-o, -f, -c, ...)
- file \Rightarrow denota el archivo con su respectiva extensión (.f90, .f95, .o, ...)

Proceso de compilación

- El proceso de compilación se puede resumir en dos pasos
 - Compilación
 - Enlazado
- El proceso de compilación en fortran posee la siguiente sintaxis:

```
1  fcomp [options] file1 [file2] [...] [fileN]
```

- fcomp \Rightarrow denota el comando para llamar al compilador. (gfortran, ifort, ...)
- options \Rightarrow opciones que permite el compilador. (-o, -f, -c, ...)
- file \Rightarrow denota el archivo con su respectiva extensión (.f90, .f95, .o, ...)
- Finalmente se obtiene un producto final que comúnmente es un archivo ejecutable.

Compilador

Compilador

- Un compialdor es un programa que transforma el codigo escrito en un lenguaje de programación (lenguaje fuente) en otro archivo escrito en otro lenguaje de programacion (lenguaje objetivo.

Compilador

- Un compialdor es un programa que transforma el codigo escrito en un lenguaje de programación (lenguaje fuente) en otro archivo escrito en otro lenguaje de programacion (lenguaje objetivo.
- El lenguaje fuente es con frecuencia un lenguaje legible para los humanos, facil de leer e interpretar, como FORTRAN.

Compilador

- Un compialdor es un programa que transforma el codigo escrito en un lenguaje de programación (lenguaje fuente) en otro archivo escrito en otro lenguaje de programacion (lenguaje objetivo.
- El lenguaje fuente es con frecuencia un lenguaje legible para los humanos, facil de leer e interpretar, como FORTRAN.
- Por otra parte el lenguaje objetivo es el lengaje máquina, el que se emplea para darle instrucciones que la maquina pueda entender y ejeutar.

Compilación

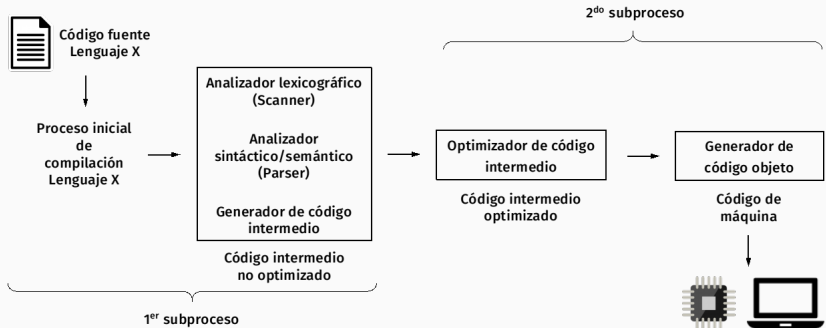


Figura 2: Diagrama de bloques del proceso de compilación

Formatos de escritura en Fortran

Formatos de escritura de un programa de Fortran

Formatos de escritura de un programa de Fortran

- Se crea el código fuente en un editor de textos y se guarda con su respectiva extensión (.f, .f90, .f95).

```
1 editor nombre.extensión
```

Formatos de escritura de un programa de Fortran

- Se crea el código fuente en un editor de textos y se guarda con su respectiva extensión (.f, .f90, .f95).

```
1 editor nombre.extensión
```

- Se emplean caracteres ASCII en el código fuente (alfanuméricos, símbolos y espacio)

Formatos de escritura de un programa de Fortran

- Se crea el código fuente en un editor de textos y se guarda con su respectiva extensión (.f, .f90, .f95).

```
1 editor nombre.extensión
```

- Se emplean caracteres ASCII en el código fuente (alfanuméricos, símbolos y espacio)
- No hay distinción entre mayúsculas y minúsculas.

Formatos de escritura de un programa de Fortran

- Se crea el código fuente en un editor de textos y se guarda con su respectiva extensión (.f, .f90, .f95).

```
1 editor nombre.extensión
```

- Se emplean caracteres ASCII en el código fuente (alfanuméricos, símbolos y espacio)
- No hay distinción entre mayúsculas y minúsculas.
- Los caracteres propios del español no pueden ser utilizadas en las instrucciones (á, é, ñ)

Formato libre

Formato libre

- Aplicable a versiones Fortran90 en adelante de extensión f90, f95...

Formato libre

- Aplicable a versiones Fortran90 en adelante de extensión f90, f95...
- Permite un máximo de 132 caracteres.

Formato libre

- Aplicable a versiones Fortran90 en adelante de extensión f90, f95...
- Permite un máximo de 132 caracteres.
- Se emplea el caracter (!) para empezar un comentario, sin que este se tome en cuenta durante la compilación.

Formato libre

- Aplicable a versiones Fortran90 en adelante de extensión f90, f95...
- Permite un máximo de 132 caracteres.
- Se emplea el caracter (!) para empezar un comentario, sin que este se tome en cuenta durante la compilación.

```
1  <instruccion> ! <comentario>
```

Formato libre

- Aplicable a versiones Fortran90 en adelante de extensión f90, f95...
- Permite un máximo de 132 caracteres.
- Se emplea el caracter (!) para empezar un comentario, sin que este se tome en cuenta durante la compilación.

1 <instruccion> ! <comentario>

- Se emplea el caracter (;) para separar dos instrucciones.

Formato libre

- Aplicable a versiones Fortran90 en adelante de extensión f90, f95...
- Permite un máximo de 132 caracteres.
- Se emplea el caracter (!) para empezar un comentario, sin que este se tome en cuenta durante la compilación.

```
1  <instruccion> ! <comentario>
```

- Se emplea el caracter (;) para separar dos instrucciones.

```
1  <instruccion>; <instruccion>
```

Formatos de escritura de un programa de Fortran

- En caso falte espacio en una línea, el caracter (&) permite continuar las instrucciones en la siguiente línea, colocándolo al final de la línea anterior y al inicio de la línea por comenzar.

Formatos de escritura de un programa de Fortran

- En caso falte espacio en una línea, el caracter (&) permite continuar las instrucciones en la siguiente línea, colocándolo al final de la línea anterior y al inicio de la línea por comenzar.

```
1      <instruccion 1> &  
2      & <instruccion 1> &  
3      & <instruccion 1>  
4      <instruccion 2>
```

Formatos de escritura de un programa de Fortran

- En caso falte espacio en una línea, el caracter (&) permite continuar las instrucciones en la siguiente línea, colocándolo al final de la línea anterior y al inicio de la línea por comenzar.

```
1      <instruccion 1> &  
2      & <instruccion 1> &  
3      & <instruccion 1>  
4      <instruccion 2>
```

- Las variables y unidades programaticas deben de ser nobradas empleando una combinacion de maximo 31 caracteres siendo el primer caracter una letra.

Formatos de escritura de un programa de Fortran

- En caso falte espacio en una línea, el caracter (&) permite continuar las instrucciones en la siguiente línea, colocándolo al final de la línea anterior y al inicio de la línea por comenzar.

```
1      <instruccion 1> &  
2      & <instruccion 1> &  
3      & <instruccion 1>  
4      <instruccion 2>
```

- Las variables y unidades programaticas deben de ser nobradas empleando una combinacion de maximo 31 caracteres siendo el primer caracter una letra.

```
1  PROGRAM TPK95
```

Formato fijo

Formato fijo

- Aplicable a versiones anteriores a Fortran90 de extensión f, for...

Formato fijo

- Aplicable a versiones anteriores a Fortran90 de extensión f, for...
- Permite un máximo de 72 caracteres.

Formato fijo

- Aplicable a versiones anteriores a Fortran90 de extensión f, for...
- Permite un máximo de 72 caracteres.
- Se emplea el caracter (C) para comenzar un comentario.

Formato fijo

- Aplicable a versiones anteriores a Fortran90 de extensión f, for...
- Permite un máximo de 72 caracteres.
- Se emplea el caracter (C) para comenzar un comentario.

```
1  c --- Comentario
```


Formato fijo

- Aplicable a versiones anteriores a Fortran90 de extensión f, for...
- Permite un máximo de 72 caracteres.
- Se emplea el caracter (C) para comenzar un comentario.

```
1  c --- Comentario
```

- Se permite una sola instrucción por línea, comenzando a partir del séptimo caracter.

Formato fijo

- Aplicable a versiones anteriores a Fortran90 de extensión f, for...
- Permite un máximo de 72 caracteres.
- Se emplea el caracter (C) para comenzar un comentario.

```
1  c --- Comentario
```

- Se permite una sola instrucción por línea, comenzando a partir del séptimo caracter.

```
1  c23456789
2      <Instruccion>
```

Formatos de escritura de un programa de Fortran

- La continuación de una instrucción en una siguiente línea está dada por algún carácter no alfanumérico en el sexto carácter.

Formatos de escritura de un programa de Fortran

- La continuación de una instrucción en una siguiente línea está dada por algún carácter no alfanumérico en el sexto carácter.

```
1  c23456789
2      <Instruccion muy larga>
3      &<Continuacion de instruccion muy larga>
```

Formatos de escritura de un programa de Fortran

- La continuación de una instrucción en una siguiente línea está dada por algún carácter no alfanumérico en el sexto carácter.

```
1  c23456789
2      <Instruccion muy larga>
3      &<Continuacion de instruccion muy larga>
```

- Los primeros cinco caracteres en una línea de instrucción se pueden emplear en caso se requiera una etiqueta (número positivo diferente de cero con un máximo de cinco dígitos).

Formatos de escritura de un programa de Fortran

- La continuación de una instrucción en una siguiente línea está dada por algún carácter no alfanumérico en el sexto carácter.

```
1  c23456789
2      <Instruccion muy larga>
3      &<Continuacion de instruccion muy larga>
```

- Los primeros cinco caracteres en una línea de instrucción se pueden emplear en caso se requiera una etiqueta (número positivo diferente de cero con un máximo de cinco dígitos).

```
1  c23456789
2  75289 <Instruccion muy larga>
3      &<Continuacion de instruccion muy larga>
```

Formatos de escritura de un programa de Fortran

- La continuación de una instrucción en una siguiente línea está dada por algún caracter no alfanumérico en el sexto caracter.

```
1  c23456789
2      <Instruccion muy larga>
3      &<Continuacion de instruccion muy larga>
```

- Los primeros cinco caracteres en una línea de instrucción se pueden emplear en caso se requiera una etiqueta (número positivo diferente de cero con un máximo de cinco dígitos).

```
1  c23456789
2  75289 <Instruccion muy larga>
3      &<Continuacion de instruccion muy larga>
```

- Las variables deben de ser nobradas empleando una combinacion de maximo 6 caracteres siendo el primer caracter una letra.

Formatos de escritura de un programa de Fortran

- La continuación de una instrucción en una siguiente línea está dada por algún caracter no alfanumérico en el sexto caracter.

```
1  c23456789
2      <Instruccion muy larga>
3      &<Continuacion de instruccion muy larga>
```

- Los primeros cinco caracteres en una línea de instrucción se pueden emplear en caso se requiera una etiqueta (número positivo diferente de cero con un máximo de cinco dígitos).

```
1  c23456789
2  75289 <Instruccion muy larga>
3      &<Continuacion de instruccion muy larga>
```

- Las variables deben de ser nobradas empleando una combinacion de maximo 6 caracteres siendo el primer caracter una letra.

```
1  c23456789
2  12 REAL*8 FUNCTION Fib
```


Estructura de un programa en Fortran

Estructura de un programa en Fortran

Estructura de un programa en Fortran

- Un programa en Fortran está ordenado lógicamente y jerárquicamente en unidades programáticas.

Estructura de un programa en Fortran

- Un programa en Fortran está ordenado lógicamente y jerárquicamente en unidades programáticas.
- Tiene una unidad principal, llamada programa principal, la cual contiene las instrucciones que definen el objetivo del programa.

Estructura de un programa en Fortran

- Un programa en Fortran está ordenado lógicamente y jerárquicamente en unidades programáticas.
- Tiene una unidad principal, llamada programa principal, la cual contiene las instrucciones que definen el objetivo del programa.
- Es posible recurrir a subprogramas, que agrupados constituyen una instrucción del programa principal.

Estructura de un programa en Fortran

- Un programa en Fortran está ordenado lógicamente y jerárquicamente en unidades programáticas.
- Tiene una unidad principal, llamada programa principal, la cual contiene las instrucciones que definen el objetivo del programa.
- Es posible recurrir a subprogramas, que agrupados constituyen una instrucción del programa principal.

```
1 PROGRAM          ejemplo      !nombre del programa          !sentencia no ejecutable
2 IMPLICIT NONE      !declara todas las variables          !sentencia no ejecutable
3 INTEGER          i          !declaración variables          !sentencia no ejecutable
4                  i = 3      !código del programa          !sentencia ejecutable
5 WRITE(*,*)        , i = ,i  !código del programa          !sentencia ejecutable
6 END PROGRAM ejemplo      !sentencia ejecutable
```

Estructura de un programa en Fortran

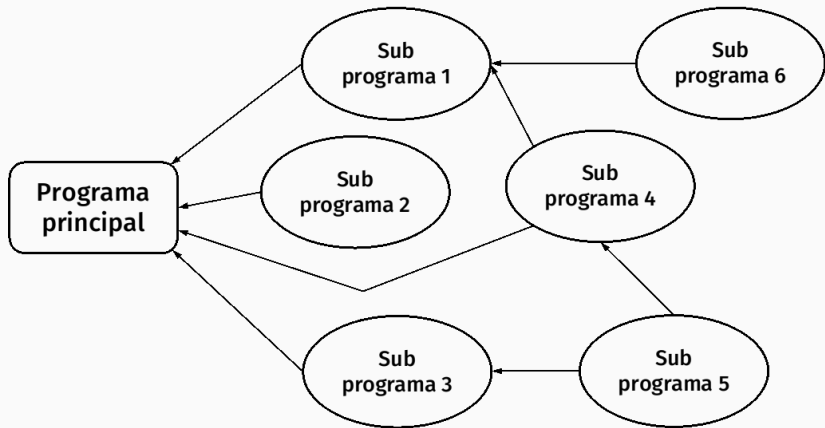


Figura 3: Esquema de la estructura de un programa en Fortran

Expresividad: el algoritmo TPK

- El algoritmo TPK es el "ejemplo inicial" para una secuencia de aprendizaje en programación. Fue empleado en el informe técnico *The Early Development of Programming Languages* por Donald Knuth y Luis Trabb del departamento de ciencia computacional de la Universidad de Stanford.

El algoritmo TPK

```
1  MODULE Functions
2  PUBLIC :: fun
3  CONTAINS
4      FUNCTION fun(t) result (r)
5          REAL, INTENT(IN) :: t
6          REAL :: r
7          r = SQRT(ABS(t)) + 5.0*t**3
8      END FUNCTION fun
9  END MODULE Functions
10
11 PROGRAM TPK95
12     ! The TPK Algorithm
13     ! F style
14     USE Functions
15     INTEGER :: i
16     REAL    :: y
17     REAL, DIMENSION(0:10) :: a
18     READ *, a
19     DO i = 10, 0, -1 ! Bucle DO con contador hacia atrás
20         y = fun(a(i))
21         IF ( y < 400.0 ) THEN
22             PRINT *, i, y
23         ELSE
24             PRINT *, i, " Too large"
25         END IF
26     END DO
```