

Ángela Ruiz Robles (1895-1975)



- Mestra e creadora da primeira enciclopedia mecánica
- Libro electrónico fabricado en 1949 no Parque de Artillería do Ferrol
- Usaba bobinas automáticas e un sistema de botóns a presión conectados a unha serie de abecedarios que construían textos de temas
- Recibiu a Cruz da Orde de Afonso X en 1947 en recoñecemento á súa traxectoria profesional

Depuración con *gdb* (I)

- O *gdb* é un depurador: un programa que permite executar o programa paso a paso para así atopar erros de execución e lóxicos
- Para usar o *gdb*, compila o programa coa opción -g: *gfortran -g programa.f90* (isto engade información de depuración no executábel *a.exe*)
- Entra no *gdb* cargando o executábel: *gdb a.exe*
- Establece un *punto de ruptura* no programa, é decir, unha liña na que se rompe a execución do programa
- O programa execútase normalmente ata esa liña na que hai o punto de ruptura
- Para establecer un punto de ruptura na liña 45: *break 45* ou tecleando *b 45* (*b*=abreviatura de *break*)

Depuración con *gdb* (II)

- Executa o programa no *gdb* co comando *run* ou *r*.
- O *gdb* executa o programa normalmente ata esa liña na que hai o punto de ruptura
- Cando a execución se para na liña 45, executas liña a liña co comando *next* ou *n*
- En todo momento podes ver o valor de calquera variábel co comando *print variable*
- Por exemplo, para ver a variábel *i*, executa *print i* ou *p i*
- Se a seguinte sentenza é unha chamada a un subprograma (que veremos máis tarde) e queres entrar nel, executa *step* ou *s*. Se non, executa *next*
- Sae do *gdb*: *quit* ou *q*

Corrección de erros de execución

- Compila coa opción -g: *gfortran -g programa.f90*
- Carga o executábel no *gdb*: *gdb a.exe*
- Executa o programa no *gdb*: *run*
- Mira en qué liña do programa se produce o erro de execución co comando *where* ou *w*
- Selecciona o nivel situado en *programa.f90*: *frame 3* (se o nivel #3 é o de *programa.f90*)
- Inspecciona os valores das variábeis: *print i* ou *p i*. Isto permite ver en que punto toman valores incorrectos, adiviñar por que, e correxir o fallo.
- Executa paso a paso: *step*
- Sair do *gdb*: *quit*

Depuración visual con VSCode (I)

- Debes tener instaladas las extensiones *Modern Fortran* e *cppdebug* (botón *Extensions*, último en barra vertical izquierda)
- Crea un directorio llamado *.vscode* en el directorio fortran
- Necesitas poner 3 archivos:

1) `extensions.json`



```
{  
  "recommendations": [  
    "fortran-lang.linter-gfortran"  
  ]  
}
```

Depuración en VSCode (II)

2) launch.json:

3) tasks.json:



```
{
  "version": "2.0.0",
  "tasks": [
    {
      "label": "compile",
      "type": "shell",
      "command": "gfortran",
      "args": ["-Wall", "-g", "programa.f90"],
      "group": "build",
    },
  ],
}
```

launch.json:

```
{
  "version": "0.2.0",
  "configurations": [
    {
      "name": "debug",
      "type": "cppdbg",
      "request": "launch",
      "program": "${workspaceFolder}/a.exe",
      "cwd": "${workspaceFolder}",
      "MIMode": "gdb",
      "preLaunchTask": "compile",
    }
  ]
}
```

Desafortunadamente, hai que cambiar o nome do programa no arquivo *tasks.json* para cada programa que fagas

Depuración en VSCode (III)

- No VSCode, abre a carpeta *fortran* e o *programa.f90*
- Vai ao meu *Run, Start debugging*, ou pulsa *F5*
- Así executa o programa e podes ver tódalas variábeis poñendo o rato sobre o seu nome (ver volcado de pantalla en páxina seguinte)
- Tamén podes engadir no apartado *Watch* (panel esquerdo medio) as variábeis que queres que te mostre
- Móstrase na parte superior unha barra de botóns. Pulsando no botón *Step Over (F10)* podes executar paso a paso
- Se queres entrar nun subprograma, pulsa no botón *Step Into (F11)*
- Para rematar a execución, botón *Stop* (pulsa *shift+F5*)

Depuración en VSCode (IV)

The image shows the VSCode interface with a Fortran program being debugged. The code is as follows:

```
1 program hello_no_input
2   implicit none
3   integer :: i
4
5   do i = 1, 2
6     print*, 'Hello', i
7   end do
8 end program
```

The current line of execution is line 6, which is highlighted in yellow. The left sidebar shows the 'VARIABLES' panel with 'i = 1' and the 'WATCH' panel with 'i = 1'. The bottom panel shows the 'DEBUG CONSOLE' with a list of loaded libraries and a breakpoint at line 5.

Annotations on the image:

- Step out**: Points to the 'Step Out' button (a blue arrow pointing right) in the top toolbar.
- Step into**: Points to the 'Step Into' button (a blue arrow pointing down) in the top toolbar.
- Stop**: Points to the 'Stop' button (a red square) in the top toolbar.

The bottom panel shows the 'DEBUG CONSOLE' with the following output:

```
8 end program
Loaded '/lib/x86_64-linux-gnu/libgfortran.so.5'. Symbols loaded.
Loaded '/lib/x86_64-linux-gnu/libc.so.6'. Symbols loaded.
Loaded '/lib/x86_64-linux-gnu/libquadmath.so.0'. Symbols loaded.
Loaded '/lib/x86_64-linux-gnu/libm.so.6'. Symbols loaded.
Loaded '/lib/x86_64-linux-gnu/libgcc_s.so.1'. Symbols loaded.

Breakpoint 2, hello_no_input () at hello_no_input.f90:5
5 do i = 1, 2
Execute debugger commands using "-exec <command>", for example "-exec info registers" will list registers in use (when GDB is the debugger)
```