

Capítulo 4

Manual de usuario

4.1. Introducción rápida

El fin de este manual de usuario es el de informar a la persona que utiliza el programa como sacarle el máximo partido posible a la aplicación del intérprete gráfico, además de detallar en profundidad todas las opciones que el intérprete gráfico ofrece. A continuación se explicará como usar la aplicación, sacando partido a todas las características que se han implementado y todas las funcionalidades que el programa permite realizar.

Se empezará explicando como lanzar la aplicación, cuales son los componentes de la ventana del programa y se explicará paso a paso cómo realizar una traza completa de un programa fuente de código tres direcciones, cómo poner y eliminar breakpoints y otras acciones de interés que se pueden realizar con la aplicación.

La mayor parte de las acciones pueden lanzarse de diferentes modos, ya bien sea desde botones directos, desde alguna opción del menú, con alguna combinación de teclas o con menús contextuales. Todas estas acciones se explicarán debidamente en el apartado donde corresponda.

Se detallará paso a paso cómo usar el programa, y se mostrarán ejemplos de todos los posibles errores que puedan darse, como por ejemplo errores léxicos o sintácticos en el programa fuente, errores semánticos en su carga o en su ejecución, errores al introducir un dato requerido, y el modo correcto de finalizar un programa.

El manual de usuario irá acompañado de ilustraciones que indicarán cuales son los elementos con los que hay que interactuar para llevar a cabo para realizar la acción que se esté explicando en ese momento.

4.2. Instalación

En un principio el programa consta solo de un ejecutable que no necesita instalación. Simplemente con copiar el fichero a un directorio donde se tenga

acceso y cambiar sus permisos deberíamos poder usarlo correctamente. Puede hacerlo con la siguiente orden.

```
cp /directorio_ejecutable/virtual ~
chmod +x virtual
```

La única dependencia que puede tener el programa con librerías no instaladas de forma predeterminada en un sistema Linux son con las librerías de ejecución Qt4. En el caso que el programa falle al iniciar indicando la falta de la librería debemos instalarla con una sencilla instrucción. Si nuestra distribución está basada en debian se instalará como `sudo apt-get install lib-qt4`. Si en cambio está basada en Red Hat será `sudo yum install libqt4`.

Para ahorrar todos los inconvenientes antes reseñados, también se distribuye el ejecutable en código fuente. Para compilar el programa sólo hay que ejecutar el script de compilación

```
chmod +x do
./do
```

Y tras el proceso de compilación el script dejará el ejecutable en la misma carpeta de entrada.¹

Finalmente, para ejecutar el intérprete gráfico deberemos escribir en la consola

```
./virtual
```

O bien haciendo doble clic en un entorno de ventanas. Recordamos que si está ejecutando el entorno de ventanas KDE 4, la librería Qt4 por definición estará seguro instalada en nuestro sistema. Recomendamos lanzar por primera vez la aplicación desde una consola, ya que así podremos leer cualquier mensaje de error que aparezca. Si lo ejecutamos haciendo doble clic los mensajes no aparecerán.

4.3. Primera vista

Al lanzar la aplicación se abre la ventana principal del programa, que será similar a la mostrada en la figura 4.1, dependiendo del tema que tenga su escritorio configurado:

Se puede observar que está dividida en el panel de instrucciones (1), un panel de memoria (2), panel de salida (3), panel de salida de errores (4), barra de menú, barra de herramientas y barra de estado. Antes de cargar ningún programa los tres paneles aparecen vacíos. Cada panel tiene multitud de información disponible.

- En el panel de instrucciones:

¹No hay que olvidar que para ello necesitaremos instalar algunas librerías y programas dependientes. Ver la sección 5.2.

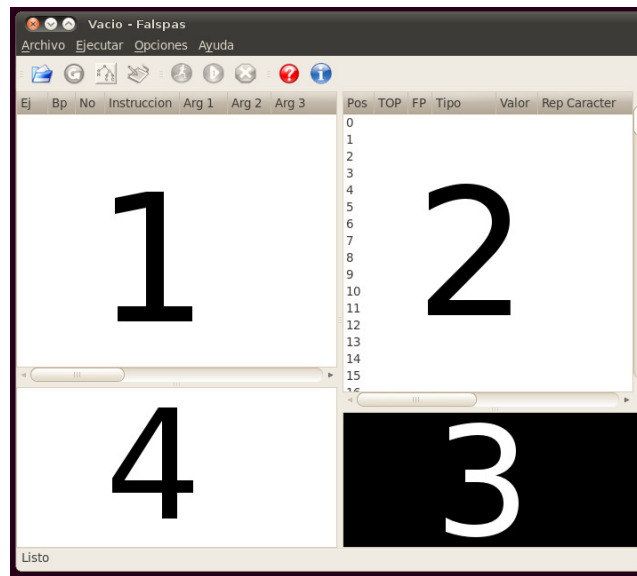



Figura 4.1: Intérprete gráfico descargado.

- En la columna Ej se marcará con una flecha la próxima instrucción a ser ejecutada.
 - En la columna Bp se indicará mediante un círculo rojo las instrucciones con un punto de ruptura asignado.
 - La columna No indicará el número de instrucción, empezando por cero.
 - La columna Instruccion mostrará el mnemotécnico de la instrucción.
 - En las tres columnas restantes se mostrarán cada uno de los argumentos de la instrucción.
 - Al pasar el puntero sobre un argumento de tipo posición, se indicará la dirección de memoria real.
- En el panel de memoria:
- En la columna Pos se muestra la posición que ocupa la celda en la memoria. El intérprete gráfico por defecto está implementado con mil posiciones de memoria en la que se puede almacenar tanto un entero como un real.
 - La columna TOP será marcada con un círculo verde en la posición de memoria apuntada por el puntero a cima.
 - En la columna FP veremos un círculo azul en la posición de memoria apuntada por el *frame pointer*.

- La columna **Tipo** indica el tipo de valor almacenado en la posición de memoria correspondiente. Puede ser:
 - Para valores reales se mostrará **R**.
 - Para valores enteros se mostrará **E**.
 - Si el valor no ha sido inicializado no se mostrará texto. En ese caso no hay que suponer que el valor de la memoria en dicha posición sea cero.
 - Para el valor de retorno almacenado tras una instrucción **CALL**, aparecerá **DIR**, y entre paréntesis el número de la instrucción **CALL** que ha dejado este valor.
 - Para el valor de un **frame pointer** apilado por la instrucción **PUSHFP**, aparecerá **FP** seguido del número de la instrucción que ha dejado el valor, entre paréntesis.
 - La columna **Valor** muestra el número, mostrando el signo en caso de ser un número negativo y evitando la notación científica siempre que el número pueda representarse con claridad en el espacio disponible.
 - Por último la columna **Rep Caracter** mostrará el carácter que en representación *ASCII* mostraría el número, para los enteros positivos entre 32 y 126.
- En el panel de salida aparecerá todo lo que nuestro programa escriba con las instrucciones **EWRITE**, **RWRITE** y **CWRITE**, dando la apariencia de ser un antiguo monitor de fósforo verde.
 - En el panel de salida de errores aparecerá cualquier error en la carga de programas, además de cualquier *warning* o excepción.

4.4. Carga de un programa

Para cargar un programa fuente hay que realizar una de las tres acciones siguientes:

1. Usando la ruta de menús **Fichero**▷**Abrir**.
2. Pulsar el botón **abrir**, indicado con la imagen de una carpeta abierta. 
3. Con la combinación de teclas **Ctrl+A**.


Tras lo cual se muestra un diálogo pidiendo un fichero con extensión *.c3d* en el directorio actual. Si nuestro fichero tiene otra extensión debemos cambiar el filtro de **Mostrar ficheros *.c3d** a **Mostrar todos los ficheros *.***. Además podemos navegar por todo el espacio de directorios hasta encontrar el fichero necesario.

Tras elegirlo, el intérprete gráfico realiza un barrido del código para verificar que no existan fallos léxicos, sintácticos o semánticos detectables en tiempo de carga (ver sección 4.4.2 para profundizar en los errores detectados). Solo si pasa

la prueba se modificará el estado de la aplicación (véase página 24), es decir, no se modificará la memoria ni el programa que esté depurándose en ese momento. En caso contrario la memoria se vaciará, el *Frame Pointer* y el puntero a cima pasarán a apuntar a cero, los posibles breakpoints se borrarán, se cargará el programa en el panel de instrucciones y la instrucción disponible a ser lanzada será la número cero. Si todo ha ido bien se mostrará un mensaje de éxito en la barra de estado. En caso contrario en el panel de salida de errores se mostrará una lista con los errores detectados y el número de línea donde se encuentran.

4.4.1. Recargar un programa

El requisito para poder recargar un programa es haber cargado un programa correcto con anterioridad. Así podemos cargar varias veces un mismo programa sin tener que hacer uso del diálogo de selección de ficheros. Esta opción es útil cuando compilamos varias veces un programa, ya sea al comportarse éste de un modo erróneo o al añadirle nuevas funcionalidades. Se puede llamar con alguna de las siguientes acciones:

1. Usando la ruta de menús Fichero▷Recargar.
2. Pulsar el botón recargar, indicado con la imagen de una flecha circular.

3. Con la combinación de teclas Ctrl+R.

Las consecuencias de recargar un programa son las mismas que abrirlo. Se borrará la memoria y se inicializará el puntero a cima, el *frame pointer* y el puntero a la instrucción próxima a ejecutarse. También se borrarán los posibles breakpoints activos.

4.4.2. Posibles errores al cargar un programa

Puede ocurrir que el fichero fuente de código de tres direcciones esté mal generado. Ocurrirá especialmente si abrimos el fichero con un editor y modificamos las líneas sin meditar las consecuencias. En ese caso la carga del fichero se interrumpirá y una ventana nos avisará del problema. Los posibles errores detectados, junto a un ejemplo y la detección son los siguientes:

- Error léxico:


```
4 EADDI p:(1,-2), i:5, p:(1,1)
```

 No existe ninguna instrucción con ese mnemotécnico. El intérprete gráfico mostrará un error léxico en la línea 4.


```
4 RMULT p:(-1,-2), r:5.6, p:(1,1)
```

 El primer parámetro es inválido dado que el primer valor de una posición de memoria debe ser obligatoriamente un cero o un uno.
- Error sintáctico:


```
6 EMULT i:8 i:5 p:(1,1)
```

Las instrucciones no están separadas por comas. El intérprete gráfico mostrará un error sintáctico en la línea 6.

■ Errores semánticos:

- 8 `RMULT r:3.2, e:8, p:(0,0)`
El segundo parámetro es incompatible con la instrucción, debiendo ser un real o una posición de memoria (que deberá apuntar a un real). El intérprete gráfico mostrará un error semántico en la línea 8.
- 12 `ERead , , p:(1, -1)`
14 `EWRITE , , p:(1, -1)`
No deben faltar instrucciones. Los números de línea deben ser correlativos, sin saltos y empezar desde la instrucción número 0. El intérprete gráfico mostrará un error semántico en la línea 13.
- 18 `GOTOS , , e:115 #Nuestro programa tiene 60 líneas`
No debe haber etiquetas que apunten a instrucciones que no existan. El intérprete gráfico marcará como errónea la instrucción, avisando de un error en la línea 18 y no cargará el programa.
- 153 `PUSHFP , ,`
Por definición solo se permiten programas de 150 líneas como máximo. El intérprete gráfico marcará como erróneo cualquier programa que lo supere, avisando de un error en la línea 150.

4.5. Cambiar el ancho de los paneles y las columnas de datos

En un principio el panel de instrucciones ocupa un cuarto de la ventana del intérprete gráfico, al igual que el panel de la memoria, de la salida de datos y de la salida de errores. Para cambiar las proporciones hay que situar el puntero del ratón encima de las líneas que lo dividen y pulsar sobre ellas. Moviendolo el ratón se conseguirá ensanchar o estrechar el panel.

Las columnas que muestran las instrucciones y la memoria se fijan a una anchura óptima al cargar cada programa. Sin embargo se puede modificar su anchura pulsando y arrastrando entre dos de las etiquetas que tienen las columnas en su parte superior. Haciendo doble clic se dejará automáticamente una anchura óptima, de modo que se verá todo el texto disponible sin dejar más espacio vacío del necesario.


4.6. Ejecutar paso a paso un programa

Tras la carga correcta de un programa, podemos ejecutarlo paso a paso. Este sistema tiene la ventaja de poder ver de forma continua los cambios que provocan las instrucciones en la memoria, los saltos que toma el camino de ejecución de nuestro programa y poder estar más atento a los mensajes de

Pos	TOP	FP	Tipo	Valor	Rep	Caracter
0			DIR (1):	2		
1			FP (24):	0		
2		→	E:	70	F	
3			E:	1		
4						
5	→					
6						
7						
8						




Figura 4.2: Memoria con la tercera posición de memoria sombreada.

error en tiempo de ejecución que pueda dar nuestro código. Para ejecutar una instrucción deberemos usar uno de los siguientes métodos:

1. Usando la ruta de menús **Ejecutar**▷**Ejecutar la siguiente instrucción**.
2. Pulsar el botón ejecutar la siguiente instrucción, indicado con la imagen verde de un triángulo con el dígito número uno inscrito. 
3. Con la combinación de teclas **Ctrl+J**.

4.6.1. Detalles en la ejecución de un programa

En un programa que está siendo depurado hay algunas marcas útiles para su depuración:

- La instrucción que será ejecutada en la siguiente iteración está marcada por una flecha². 
- La última posición de memoria donde se ha escrito un resultado aparecerá sombreada, como la tercera posición de memoria en en la figura 4.2. Si la última instrucción no ha escrito nada en memoria (por ejemplo, una instrucción de salto o entrada y salida de datos) la indicación desaparecerá hasta que no se vuelva a escribir en memoria.
- La posición del puntero a cima estará en cada momento señalada por una flecha azul en la columna TOP. Se debe tener en cuenta que la posición de memoria señalada es la primera libre, no la última escrita. 
- La posición del frame pointer está indicado en cada momento con una flecha verde en la columna FP. 
- Si pasamos el puntero sobre el panel de instrucciones y lo detenemos encima de un parámetro posición de memoria, aparecerá un pequeño cuadro

²El dibujo de la flecha puede ser ligeramente distinto dependiendo de la versión de la librería de ejecución **Qt4** instalada en su sistema.

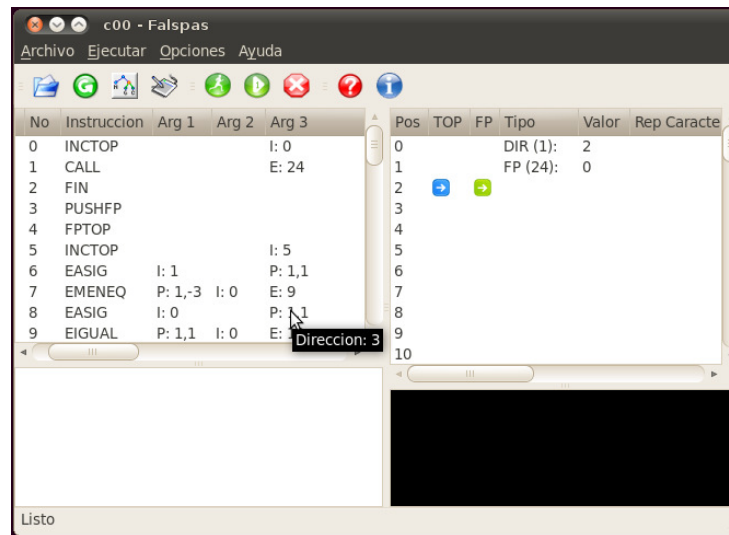


Figura 4.3: Pequeño cuadro de ayuda que indica la dirección física de memoria.

de ayuda indicando la dirección física de memoria en vez de la lógica. Evidentemente la información mostrada cambia cada vez varía la dirección del frame pointer. Un ejemplo se muestra en la figura 4.3. Como puede comprobarse la dirección física mostrada es la posición tercera, resultado de sumar al *frame pointer* el desplazamiento (2+1).

4.6.2. Uso estricto de la memoria

Un programa bien escrito en código de tres direcciones debería cumplir unas reglas semánticas en cuanto al uso de la memoria:

1. No debería poderse leer ni escribir en posiciones de memoria iguales o superiores al puntero a cima.
2. El entero que apila en memoria una instrucción CALL solo debería leerse (y a la vez desapilarse) por una instrucción RET.
3. Las antiguas posiciones del frame pointer apiladas en memoria por la instrucción PUSHFP solo deberían poder leerse (y desapilarse) por una instrucción FPPOP.

Así, al activar el uso estricto de la memoria

- Usando la opción de menú Opciones▷Memoria estricta.
- O bien la combinación de teclas Ctrl+I.

Se comprobará que se cumplan estas tres reglas y en caso contrario se mostrarán las advertencias necesarias en el panel de salida de errores.

4.6.3. Posibles errores en tiempo de ejecución

Todas las instrucciones pueden dar mensajes de advertencia o excepciones en tiempo de ejecución cuando se den condiciones que compromentan la integridad semántica del programa. Estas advertencias se dan:


1. Cuando una instrucción busca un dato en memoria y éste no es del tipo correcto. Por compromiso se leerá un dato truncado, pero en realidad nuestro compilador no está bien escrito.
2. Cuando una instrucción busca un dato en memoria y en la posición no se ha escrito aún. Por compromiso se leerá un cero, pero en realidad nuestro compilador no está bien escrito. Todas las posiciones de memoria deberían inicializarse antes de leerlas.
3. El segundo argumento de una instrucción de división entera, real o la instrucción de resto es un cero. Por compromiso el resultado de la operación será de cero, pero es evidente que los datos que le hemos proporcionado a nuestro programa no son correctos. Para evitarlo deberíamos comprobar en nuestro programa fuente que no se de esta condición antes de efectuar la división.
4. Un programa acaba en una instrucción distinta a la instrucción FIN. Si la última instrucción de un programa no es de salto o una instrucción FIN, al actualizar el contador de programa se intentará leer una instrucción inexistente, finalizando el programa. Aunque puede parecer que el programa funciona, en realidad nuestro compilador no lo ha generado correctamente. Esta excepción es irrecuperable, debiendo reiniciar el estado del intérprete gráfico (Ver sección 4.9).
5. Ejecutar la instrucción EPOP cuando el puntero a cima esté en su posición inicial. Por compromiso devolveremos un cero y el puntero a cima no se modificará.
6. Ejecutar la instrucción DECTOP con un argumento superior al valor actual del puntero a cima. Esto quiere decir que por ejemplo, si el puntero a cima apunta a la posición de memoria 5, ejecutar DECTOP con un argumento mayor que 5 fallará. Por compromiso dejaremos el puntero a pila en la posición inicial.
7. Análogamente, cualquier instrucción que modifique o lea valores por encima del puntero a cima.
8. Ejecutar la instrucción RET con el puntero a cima apuntando a cero, a un dato no inicializado o a un real.
9. Cuando una instrucción escribe a posiciones de memoria que no existen. En este caso el dato no se escribirá.

10. Aunque sean enteros, el intérprete analiza que los números que escribe la instrucción CALL en la memoria solo puedan ser usados por la instrucción RET y por ninguna otra.
11. Aunque sean enteros, el intérprete también hace lo mismo con las cifras escritas por la instrucción FPTOP y TOPFP.

Sin embargo hemos de remarcar que no se indicará ninguna condición de *overflow* o *underflow* en la ejecución de los cálculos.

4.7. Ejecutar el programa de forma continua

Aunque es conveniente ejecutar antes el programa paso a paso para cerciorarse de que funciona correctamente, también podemos ejecutarlo de forma continua. Un programa así ejecutado solo parará ante una instrucción FIN, ante la excepción señalada con el número 4 de la sección anterior o ante una instrucción marcada como breakpoint. Hay que tener en cuenta que un programa ejecutado de forma continua no modificará la representación del panel de memoria ni la indicación de instrucción próxima a ejecutarse hasta que la ejecución finalice. Para ejecutar el programa de este modo usaremos cualquiera de estos métodos:

1. Navegando por el menú, haremos clic en Ejecutar ▷ Ejecutar.
2. Pulsar el botón ejecutar, indicado con la imagen de una persona corriendo dentro de un círculo verde. 
3. Con la combinación de teclas Ctrl+E.

Añadir que se pueden mezclar los dos tipos de ejecución (continua y paso a paso) en una misma traza sin ningún problema. También debemos tener la precaución que el programa no entre un estado de bucle infinito. En ese caso deberemos detener el proceso de nuestro intérprete gráfico usando una instrucción `kill -9` y abrirla de nuevo.

4.8. Añadir y quitar breakpoints


Los breakpoints son muy útiles para, por ejemplo, ejecutar de forma continua un bucle y parar la ejecución a la salida, pudiendo ver entonces el estado de la memoria. Evidentemente los breakpoints no tienen sentido cuando se ejecuta el programa paso a paso ya que el intérprete gráfico detiene su ejecución del código tras cada instrucción automáticamente. Para insertar un breakpoint basta con hacer doble clic en la instrucción que queremos que el intérprete pare *antes* de ejecutarla. Para quitar el breakpoint haremos también doble clic sobre la instrucción. Las instrucciones marcadas con breakpoints se distinguen por un pequeño símbolo en la columna Bp (breakpoint) similar a una señal de tráfico.



Para quitar todos los breakpoints de un programa de golpe –en el caso de haber alguno– se puede ejecutar la opción borrar todos los breakpoints navegando por el menú **Opciones**▷**Borrar los breakpoints**, o bien usar la combinación de teclas **Ctrl+B**. Tras su uso en la barra de estado se indicarán cuales son las líneas donde se ha añadido o quitado el breakpoint, o cuántos se han quitado de golpe.

4.9. Detener y reiniciar el estado

Si hemos introducido datos erróneos o por algún motivo queremos reiniciar el estado de nuestro intérprete gráfico podemos hacerlo con alguno de los siguientes métodos:

1. Desde el menú, navegaremos por las opciones **Ejecutar**▷**Detener la ejecución y reiniciar el intérprete gráfico**.
2. Pulsar el botón detener, indicado con la imagen de un hexágono rojo con un aspa. .
3. Con la combinación de teclas **Ctrl+D**.

Esta orden borra e inicializa la memoria, pone a cero el puntero a cima y el *frame pointer* y marca la instrucción cero como la próxima a ejecutarse. Sin embargo no modifica los breakpoints ni borra los datos de salida de anteriores ejecuciones del panel.

4.10. Desplazar automáticamente la representación de la memoria y las instrucciones

Si la ventana en la que ejecutamos el intérprete gráfico es pequeña o nuestro programa usa muchas instrucciones o memoria llegará un momento en que se escriba en posiciones de memoria que queden fuera del área de memoria representada en el panel o se ejecutarán instrucciones posteriores a las representadas en el panel de instrucciones. En ese caso tendremos que hacer scroll con las barras desplazadoras de los paneles. Como hacer eso durante la ejecución paso a paso de un programa es realmente incómodo, se puede usar una opción que garantiza que siempre estará visible la próxima instrucción a ejecutarse y el último dato que se ha escrito en memoria. Para activar la opción hay que hacer clic en la opción de menú **Opciones**▷**Desplazar la memoria** o bien usar la combinación de teclas **Ctrl+Z**.

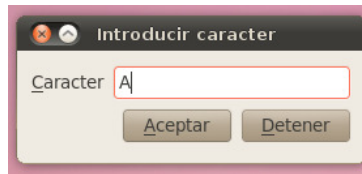




Figura 4.4: Petición de un dato por parte del intérprete gráfico

4.11. Mostrar la ayuda y la autoría del intérprete gráfico

El intérprete gráfico tiene tanto un pequeño cuadro de ayuda indicando el significado de las distintas instrucciones como un recuadro mostrando la autoría y el objeto de la aplicación. Podemos acceder a la ayuda del programa mediante cualquiera de los siguientes métodos:

1. Usando el menú, con la opción Ayuda▷Ayuda.
2. Pulsar el botón de ayuda, indicado con la imagen de un interrogante en un círculo rojo. .
3. Con la combinación de teclas Ctrl+Y.

Y para acceder a la autoría y descripción del programa podremos:

1. Navegando por el menú, usando la opción Ayuda▷Acerca de.
2. Pulsar el botón Acerca de, indicado con la imagen de una letra i en un círculo azul. .
3. Con la combinación de teclas Ctrl+C.

4.12. Peticiones de datos por parte del intérprete gráfico

Cuando el intérprete gráfico ejecute la instrucción EREAD, RREAD o CREAD mostrará un pequeño dialogo que consta de una caja de introducción de caracteres, un botón aceptar y otro botón detener como la mostrada en la figura 4.4.

La caja tiene un validador que hace imposible introducir una secuencia de caracteres que no corresponda al tipo de datos necesitado.


- En caso de un entero sólo acepta cadenas que cumplan la expresión regular `"-"?[0-9]+`
- El validador de un real acepta `"-"?[0-9]+"."[0-9]+`

- Y finalmente la petición de caracter acepta `[\x20-\x7E]{1}`, es decir, un y solo un caracter imprimible, lo que es equivalente a que su código ASCII esté comprendido entre el 32 y el 126.

Cuando se haya escrito un dato válido, el botón **Aceptar** del diálogo pasará a estar disponible. Pulsándolo, el intérprete gráfico recibirá el dato y la asignará a la posición de memoria indicada en el tercer argumento de la instrucción. Si en cambio se pulsa el botón **Detener** el intérprete no guardará dato alguno y el programa se detendrá si se estaba ejecutando en modo continuo.

4.13. Ayudas al desarrollo


He querido que mi herramienta tuviese opciones para facilitar la tarea de crear un compilador o programa adecuados. Para ello he añadido dos opciones que pueden resultar muy útiles:

- La opción de compilar puede iniciarse con alguno de estos dos métodos:
 - Con la opción de menú **Archivo▷Compilar**.
 - Pulsando el botón de compilar, donde se encuentra dibujado un árbol sintáctico. 

Esta herramienta compila de nuevo el programa que tengamos cargado en memoria y a continuación lo carga. Funciona de la siguiente manera:

- A través de la variable interna del intérprete `fichero_actual` (ver la sección 3.4.3 para más información) se obtiene la ruta absoluta y el nombre de fichero.
- Se elimina la posible extensión del fichero (casi con toda seguridad `.c3d`) y se cambia por la extensión `.c`.³
- Sin ninguna intervención del usuario el intérprete llama al compilador, recompila el archivo `.c` y recarga el archivo `.c3d` suponiendo que no contenga errores.

Tenemos que advertir que para usar esta opción el archivo de código fuente ha de tener la extensión `.c` y debe encontrarse en el mismo directorio que el archivo `.c3d`. De lo contrario no funcionará.

- La opción para editar puede iniciarse desde el menú o desde el botón de herramientas:
 - Mediante la orden del menú **Archivo▷Editar**.
 - Pulsando el botón de editar, donde se muestra el icono de una tableta de dibujo. 

³Ver la información de la clase `QFileInfo` en la documentación oficial de **Qt4** (Qt 4 Assistant) para obtener más información.

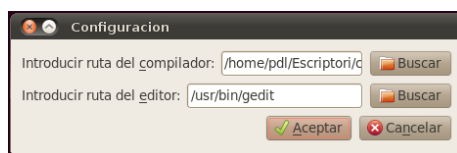


Figura 4.5: Ventana de configuración.

Usando este método se abrirá en un editor de texto el fichero `.c3d` que tengamos cargado actualmente. Podremos modificarlo y tras guardarlo y cerrar el editor el intérprete volverá a cargar el programa modificado en caso de no contener errores. Esta opción deberá usarse con el único propósito de hacer pruebas en el caso de que estemos programando un compilador.

- Tanto para usar la opción de compilar como la de editar debemos antes establecer cual es el editor y compilador que queremos usar. Para ello tenemos que especificarlo haciendo uso de la ventana de configuración mostrada en la figura 4.5. Para invocar la ventana de configuración debemos introducir la orden de menú **Opciones** ▸ **Configuración** que está marcado con el icono de una libreta y un engranaje. 📝⚙️

En esta ventana podemos hacer uso de las cajas de texto o de los botones que despliegan diálogos para buscar archivos. Finalmente al pulsar el botón aceptar se comprobará que las rutas sean correctas y los ficheros tengan permiso de ejecución. Pasado el test se guardarán las opciones. Para la elección de editor recomiendo el uso de `gedit`, en la ruta `/usr/bin/gedit`, mientras que la elección del compilador dependerá de cada usuario.

4.14. Persistencia de la configuración y opciones por defecto.

La primera vez que se ejecute el intérprete gráfico en nuestro equipo cargará las órdenes por defecto. Estas opciones incluyen el tamaño y posición de la ventana, el tamaño de los cuatro paneles con los que consta la ventana principal, la anchura de las columnas de los paneles de memoria e instrucciones, el uso estricto de la memoria, el desplazamiento automático de los paneles de instrucciones y memoria y finalmente la ruta del compilador y el editor, que por defecto son `./cmc` y `/usr/bin/gedit`. Al cerrar el intérprete si hemos cambiado alguno de estos aspectos durante su uso se guardará su nueva configuración para que la próxima vez que se ejecute se cargue de nuevo tal y como lo cerramos. Para ello el intérprete usa la herramienta de `QSettings` que es independiente del entorno de ventanas que estemos ejecutando⁴.

⁴Para más información sobre `QSettings`, lugar donde guarda los archivos de configuración y demás cuestiones mirar la sección `QSettings` en `Qt 4 Assistant`.

Pero evidentemente hemos de señalar que si se está ejecutando el intérprete desde una distribución de GNU/Linux *live* desde un CD o similares, dichos cambios no se mostrarán cuando se reinicie el ordenador.