

Guia de clase

Informática I

Ing. Jerónimo F. Atencio

Versión 16

Índice

| | |
|--|-----------|
| Índice | 1 |
| Acerca de esta guía | 6 |
| Modalidad de evaluación | 7 |
| Condiciones para la firma de la materia. | 7 |
| Condiciones para la promoción de la materia. | 7 |
| Realización de los parciales y recuperatorios | 7 |
| Entrega de trabajos prácticos | 8 |
| 1. Sistemas operativos | 9 |
| Instalación de Lubuntu en una máquina virtual | 9 |
| 2. Comandos básicos de la terminal | 21 |
| a. date: Fecha y hora | 21 |
| b. man: Manuales | 21 |
| c. pwd: Directorio actual | 21 |
| d. ls: Listar | 22 |
| e. mkdir | 22 |
| f. cd | 22 |
| g. touch | 22 |
| h. rm | 22 |
| i. rmdir | 22 |
| j. clear | 23 |
| 3. Instalando paquetes | 24 |
| 4. Instalando herramientas de desarrollo y man pages | 25 |
| Instalando el editor de texto visual studio code | 25 |
| Instalando extensiones en el editor de texto visual studio code | 26 |
| Instalando guest addition en virtualbox | 26 |
| 5. Mi primer programa en C | 28 |
| 6. Accediendo al repositorio | 30 |
| Verificando los repositorios desde el navegador. | 30 |
| Configurando Git por primera vez | 31 |
| Clonando el repositorio de material. | 32 |
| Actualizando el la versión local del repositorio de material desde uno remoto (pull) | 32 |
| Clonando su repositorio personal. | 33 |
| Agregando o modificando un archivo a su repositorio. (commit, push) | 34 |
| Resumen | 35 |
| 7. Sistemas de numeración | 36 |
| Clasificación de sistemas de numeración | 36 |
| Sistema de numeración decimal | 37 |
| Sistema de numeración binario | 37 |
| Sistema de numeración octal | 37 |
| Sistema de numeración hexadecimal | 37 |
| Símbolos para los sistemas de numeración base 2, 8, 10 y 16 | 38 |
| Cambio de base decimal a binario, octal o hexadecimal | 38 |
| Cambio de base de octal a binario y binario a octal | 40 |

| | |
|---|-----------|
| Cambio de base hexadecimal a binario o de binario a hexadecimal | 40 |
| Cambio de base octal a hexadecimal o hexadecimal a octal | 40 |
| Unidad de información | 41 |
| Ejercicios | 41 |
| 8. Ingreso de datos e impresión en pantalla | 42 |
| Tipos de datos | 42 |
| Secuencias de escape | 42 |
| Especificadores de formato | 42 |
| Tabla ASCII | 43 |
| Funciones utilizadas | 43 |
| Cómo obtener el manual de una función de biblioteca | 43 |
| Ejemplos | 43 |
| Ejercicios | 44 |
| 9. Operaciones aritmética - casteo. | 46 |
| Operadores aritméticos | 46 |
| Funciones utilizadas | 46 |
| Ejemplos | 46 |
| Ejercicios | 47 |
| 10. Sentencias condicionales if y switch-case | 49 |
| Operadores relacionales | 49 |
| Operadores lógicos | 49 |
| Sentencias utilizadas | 49 |
| Ejemplos | 49 |
| Ejercicios | 55 |
| 11. Sentencias de repetición for; while; do-while | 57 |
| Operadores asignación, incremento y decremento | 57 |
| Sentencias utilizadas | 57 |
| Ejemplos | 57 |
| Ejercicios | 59 |
| 12. Directiva de precompilador define | 61 |
| Directivas de precompilación | 61 |
| Ejemplos | 61 |
| Ejercicios | 62 |
| 13. Funciones | 63 |
| Constantes simbólicas | 63 |
| Ejemplos | 63 |
| Ejercicios | 66 |
| 14. Vectores y strings | 67 |
| Ejemplos | 68 |
| Ejercicios | 71 |
| 15. Punteros | 73 |
| Forma básica de usar punteros | 73 |
| Ejemplos | 74 |
| Ejercicios | 77 |

| | |
|---|------------|
| 16. Asignación dinámica de memoria | 79 |
| Funciones utilizadas de stdlib.h | 79 |
| Ejemplos | 79 |
| Ejercicios | 81 |
| 17. String | 82 |
| Funciones utilizadas de stdio.h | 82 |
| Funciones utilizadas de string.h | 82 |
| Ejemplos | 82 |
| Ejercicios | 88 |
| 18. Algoritmos integradores | 92 |
| Ejemplos | 92 |
| Ejercicios | 93 |
| 19. Modelo de memoria | 97 |
| Tipos de datos (para 64 bits) | 97 |
| Ámbito de uso de una variable. (scope de una variable) | 97 |
| Mapa de memoria | 99 |
| Modificadores de variables | 99 |
| Direcciones de memoria | 101 |
| Standard streams | 104 |
| 20. Estructuras | 105 |
| Operador | 105 |
| Ejemplos | 105 |
| Ejercicios | 116 |
| 21. Archivos I | 119 |
| Funciones utilizadas de stdio.h | 119 |
| Modos de apertura de un archivo | 119 |
| Programas necesarios | 119 |
| Comandos | 120 |
| Ejemplo de uso de comandos. | 120 |
| Ejemplos | 123 |
| Ejercicios | 130 |
| 22. Archivos II | 131 |
| Funciones utilizadas de stdio.h | 131 |
| Funciones utilizadas de sys/stat.h | 131 |
| Comandos | 131 |
| Ejemplo de uso de comandos. | 131 |
| Cada columna de lo devuelto por ls -l es: | 132 |
| Ejemplo de uso de uso de chmod, le damos al archivo permisos de lectura, escritura y ejecución para el propietario, el grupo y otros. | 132 |
| Ejemplos | 133 |
| Ejercicios | 138 |
| 23. Punteros, el regreso! | 140 |
| Ejemplos | 141 |
| Ejercicios | 148 |

| | |
|---|------------|
| 24. Matrices. | 150 |
| Ejemplos | 150 |
| Ejercicios | 152 |
| 25. Operaciones a nivel de bits, campos de bits. Uniones. Enum | 154 |
| Operadores a nivel de bits | 154 |
| Ejemplos | 154 |
| Ejercicios | 161 |
| 26. Recursividad. | 162 |
| Ejercicios | 163 |
| 27. Uso de makefile: compilación y linkeo. | 164 |
| Programas necesarios | 164 |
| Pasos de la compilación | 164 |
| Compilación de varios .c y .h | 165 |
| Makefile | 167 |
| Como instalar debugger en Atom | 168 |
| Cómo debuggear un programa en C | 169 |
| Library | 171 |
| Static library | 171 |
| Dynamic library | 172 |
| 28. Documentación del código: Doxygen e indentación | 174 |
| Programas necesarios | 174 |
| Doxygen | 174 |
| Documentando con doxygen | 174 |
| Ejemplo de documentación de archivo con función main. | 175 |
| Ejemplo de documentación de archivo con funciones varias. | 176 |
| Ejemplo de documentación de archivo .h | 177 |
| 29. Lista simple enlazadas I | 178 |
| Insertar un nodo al inicio (pila) | 178 |
| Imprime todos los nodos | 179 |
| Libera todos los nodos | 179 |
| Cuenta la cantidad de nodos de la lista simple enlazada | 180 |
| Inserta un nodo al final | 180 |
| Busca un nodo | 180 |
| Remueve un nodo | 181 |
| Inserta un nodo de forma ordenada | 182 |
| Función main que demuestra el uso de las funciones anteriormente descritas. | 183 |
| Ejercicios | 184 |
| 30. Signals | 186 |
| Funciones utilizadas | 186 |
| Comandos | 186 |
| Programa, proceso | 186 |
| Ejemplos | 187 |
| Ejercicios | 193 |
| 31. Threads | 194 |



| | |
|------------------------------------|------------|
| Funciones utilizadas | 194 |
| Ejemplos | 194 |
| 32. Sockets | 198 |
| Funciones utilizadas | 198 |
| Ejemplos | 198 |
| Ejercicios | 199 |
| 33. Interfaces visuales: Qt | 199 |
| Programas necesarios | 200 |
| Ejecución | 200 |
| Funciones utilizadas de stdlib.h | 201 |



Acerca de esta guía

Este documento intenta guiarte en el aprendizaje de lenguaje C sobre la plataforma Linux, abarcando la instalación del sistema operativo y las herramientas de desarrollo, para luego comenzar a realizar los primeros programas en C.

Cada apartado de programación comienza mostrando información relevante para llevar adelante el apartado, para luego continuar con algunos ejemplos y terminar con ejercicios propuestos para realizar. Es recomendable seguir el orden de la guía y los ejercicios ya que la dificultad de los mismos va en incremento. Los ejemplos son muy básicos, pero sirven para mostrar algún aspecto del lenguaje y/o como base para la resolución de los ejercicios, por lo cual es recomendable transcribirlos y probarlos.

Se recomienda que los programas de ejemplo los transcribas evitando copiar y pegar para ir tomando práctica en la sintaxis del lenguaje. Trate de leer los manuales (use el comando `man`) de las funciones, sentencias y comandos utilizados para interiorizarse en su uso más allá de la aplicación particular que se haga en esta guía.

Modalidad de evaluación

La evaluación de la materia se realizará con dos parciales y cada uno tendrá dos instancias de recuperación.

Condiciones para la firma de la materia.

1. Obtener 6 o más puntos en el primer parcial. Se puede recuperar en diciembre y/o en febrero.
2. Obtener 6 o más puntos en el segundo parcial. Se puede recuperar en diciembre y/o en febrero.
3. Tener aprobado el 75% de los trabajos prácticos.

Condiciones para la promoción de la materia.

1. Obtener 7 o más puntos en el primer parcial.
2. Obtener 8 o más puntos en el primer parcial.
3. Tener aprobado el 75% de los trabajos prácticos.

En este caso si alguno de los parciales no alcanza la condición para promoción, se podrá recuperar sólo uno en la fecha de diciembre. La nota que queda es la de la última evaluación realizada.

Ejemplos:

| Parciales | | Recuperatorio diciembre | | Recuperatorio febrero | | Trabajo práctico aprobado | Resultado |
|----------------|-----------------|-------------------------|-----------------|-----------------------|-----------------|---------------------------|------------|
| Primer parcial | Segundo parcial | Primer parcial | Segundo parcial | Primer parcial | Segundo parcial | | |
| 7 | 8 | - | - | - | - | Si | Promociona |
| 6 | 8 | 7 | - | - | - | Si | Promociona |
| 6 | 8 | 6 | - | - | - | Si | Firma |
| 7 | 7 | - | 5 | - | 6 | Si | Firma |
| 10 | 7 | - | - | - | - | Si | Firma |
| 10 | 7 | - | 7 | - | 8 | Si | Firma |
| 7 | 7 | - | 4 | - | 2 | Si | Recurso |
| 10 | 10 | - | - | - | - | No | Recurso |

Realización de los parciales y recuperatorios

Las fechas de los parciales y los recuperatorios las podrá encontrar en la planificación. En cada parcial entran todos los temas vistos hasta la clase anterior dicho parcial. Ambos parciales serán presenciales así como también los recuperatorios.

Entrega de trabajos prácticos

En la planificación encontrará las fechas límites para la entrega de los ejercicios de cada capítulo.

Todos los ejercicios deberán ser subidos al repositorio en una carpeta con la siguiente convención de nombres ejercicios_AA donde

- AA: Es el número de capítulo

Por ejemplo: para el capítulo 8 la carpeta se llamara se llamará ejercicios_08

Al realizar los ejemplos salvo que indique lo contrario utilizaremos la siguiente convención de nombres ejemplo_AA_BB.c donde

- AA: Es el número de capítulo
- BB: Es el número de ejemplo

Por ejemplo: el ejemplo 1 del capítulo 8 se llamará ejemplo08_01.c

Al realizar los ejemplos salvo que indique lo contrario utilizaremos la siguiente convención de nombres ejercicio_AA_BB.c donde

- AA: Es el número de capítulo
- BB: Es el número de ejercicio

Por ejemplo: el ejercicio 10 del capítulo 9 se llamará ejercicios 09_10.c



1.Sistemas operativos

El sistema operativo es el software que gestiona los recursos de hardware y software además de proveer una interfaz para los programas que se ejecutarán en él. Existen numerosos sistema operativos entre los cuales podemos mencionar:

- Linux
- Windows
- MacOS
- Unix

En esta guía utilizaremos Linux como sistema operativo, el cual posee varias distribuciones (distro), como por ejemplo:

- Debian (<https://www.debian.org>)
- Ubuntu (<https://ubuntu.com>)
- Lubuntu (<https://lubuntu.net>)
- Mint (<https://linuxmint.com>)
- Fedora (<https://getfedora.org>)
- Arch (<https://archlinux.org/>)

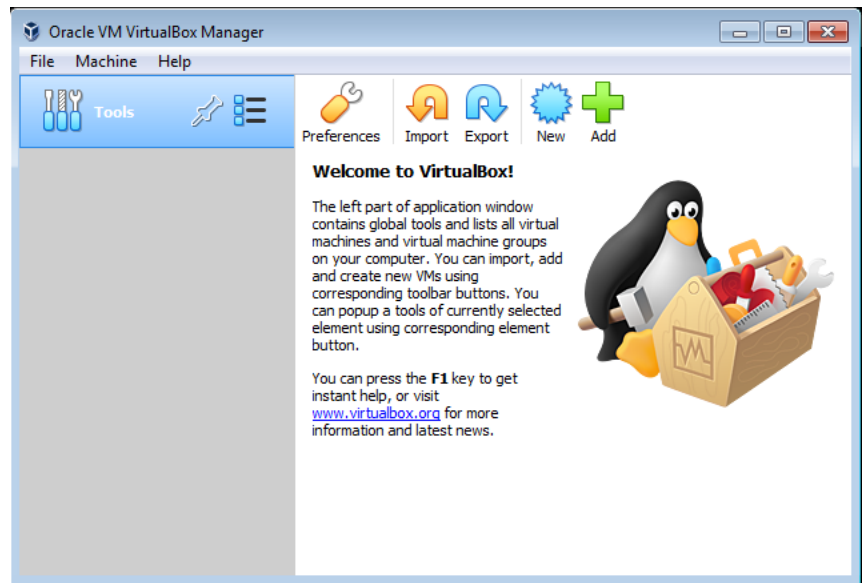
Una distribución utiliza el Kernel de linux (<https://www.kernel.org/>) y está compuesta por software adicional como por ejemplo un editor de texto, software para reproducir música o video, browser, un manejador de archivos, etc. Además de documentación, las herramientas GNU, un gestor de paquetes y sus bibliotecas junto con alguna interfaz de usuario que puede ser una manejador de ventanas o una terminal. La elección de la distribución suele depender del uso que se dará y de los gustos del usuario, en el caso de esta guía utilizaremos Lubuntu.

Instalación de Lubuntu en una máquina virtual

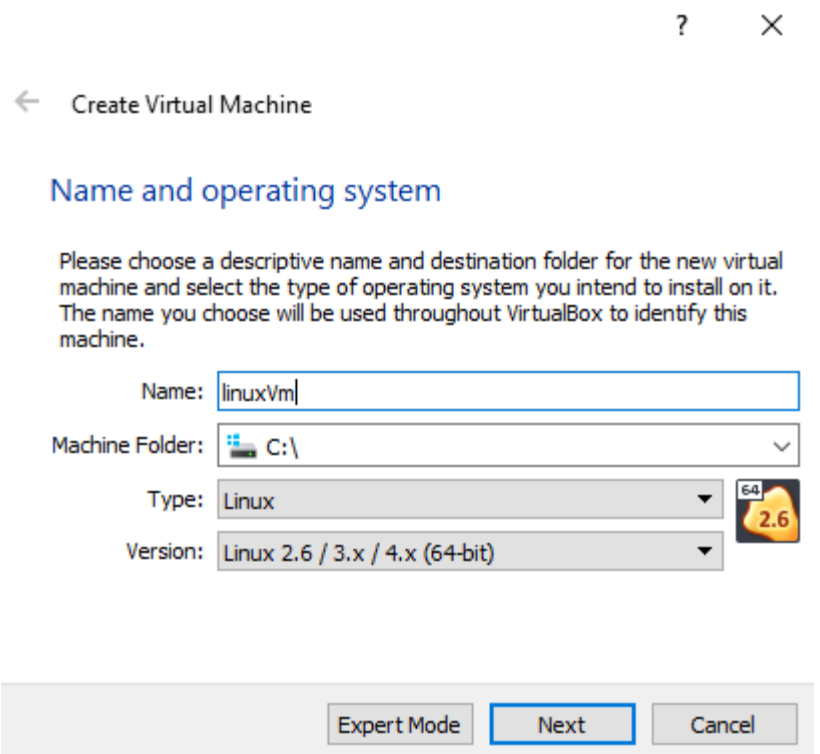
Si no tiene experiencia en instalación de sistemas operativos, instalarlo en una máquina virtual es una forma sencilla de aprender a hacerlo y segura, dado que ante cualquier error podría borrar la imagen y volver a intentarlo sin afectar sus sistema operativo nativo. Cuando adquiera alguna experiencia en Linux es conveniente que lo instale de forma nativa en su computadora. Los siguientes pasos asumen que su sistema nativo es Windows.

- a. Descargar una versión LTS (Long Term Support) de la distribución Lubuntu (<https://lubuntu.net>)
- b. Descargar el software VirtualBox (<https://www.virtualbox.org/>) que es el que usaremos para hacer la máquina virtual. Debe descargar el que dice Windows Host.
- c. Instale el VirtualBox, es recomendable reiniciar la computadora al finalizar la instalación.

- d. Una vez abierto el VirtualBox, cree una nueva máquina virtual presionando en el botón **New**



- e. En la siguiente ventana se configura lo siguiente
- El nombre de la máquina virtual
 - La carpeta donde estará el disco virtual (Dejamos la que está configurada)
 - El tipo de sistema operativo, seleccionamos linux
 - La versión (distribución) del sistema operativo



- f. Se selecciona la memoria que se reserva para nuestra máquina virtual, inicialmente se puede dejar el valor propuesto para la instalación, en el caso de que funcione lento se puede luego asignarle más. Se debe tener en cuenta que la memoria que reservamos para la máquina virtual es memoria que la máquina física no usará.

← Create Virtual Machine

Memory size

Select the amount of memory (RAM) in megabytes to be allocated to the virtual machine.

The recommended memory size is **1024 MB**.

4 MB 8192 MB

1024 MB

Next Cancel

- g. Se le secciona la opción para usar un disco rígido virtual nuevo

← Create Virtual Machine

Hard disk

If you wish you can add a virtual hard disk to the new machine. You can either create a new hard disk file or select one from the list or from another location using the folder icon.

If you need a more complex storage set-up you can skip this step and make the changes to the machine settings once the machine is created.

The recommended size of the hard disk is **8.00 GB**.

☐ Do not add a virtual hard disk

☒ Create a virtual hard disk now

☐ Use an existing virtual hard disk file

Folder icon

Create Cancel

- h. Selección del tipo de disco rígido virtual.

? ×

← Create Virtual Hard Disk

Hard disk file type

Please choose the type of file that you would like to use for the new virtual hard disk. If you do not need to use it with other virtualization software you can leave this setting unchanged.

☒ VDI (VirtualBox Disk Image)

☐ VHD (Virtual Hard Disk)

☐ VMDK (Virtual Machine Disk)

Expert Mode

Next

Cancel

- i. Se selecciona la opción para que cree el disco rígido virtual de tamaño fijo.

? ×

← Create Virtual Hard Disk

Storage on physical hard disk

Please choose whether the new virtual hard disk file should grow as it is used (dynamically allocated) or if it should be created at its maximum size (fixed size).

A **dynamically allocated** hard disk file will only use space on your physical hard disk as it fills up (up to a maximum **fixed size**), although it will not shrink again automatically when space on it is freed.

A **fixed size** hard disk file may take longer to create on some systems but is often faster to use.

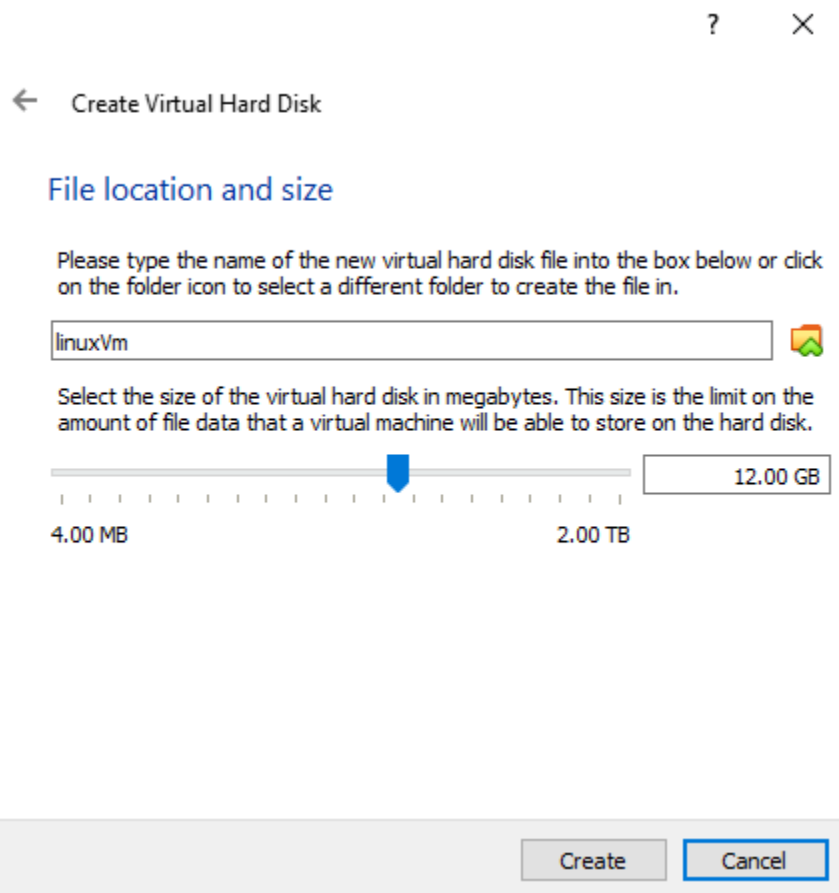
☐ Dynamically allocated

☒ Fixed size

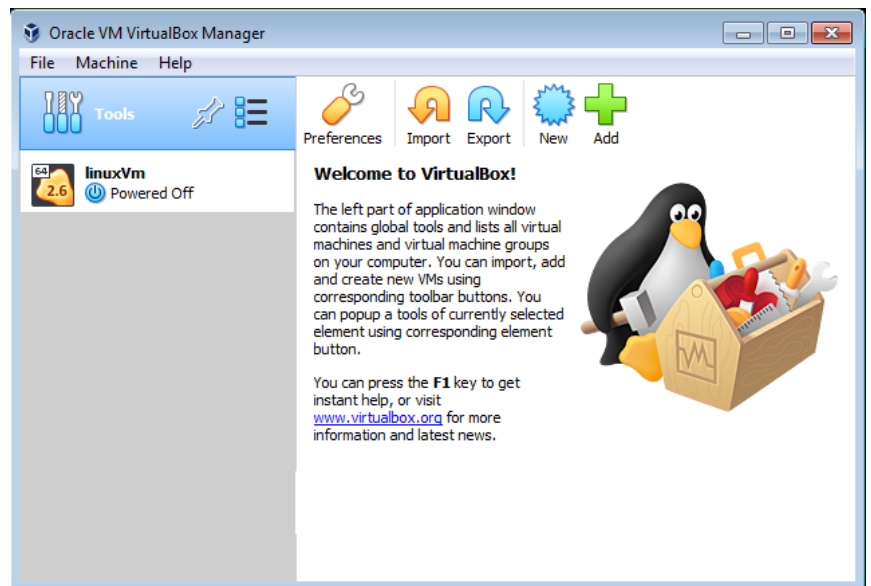
Next

Cancel

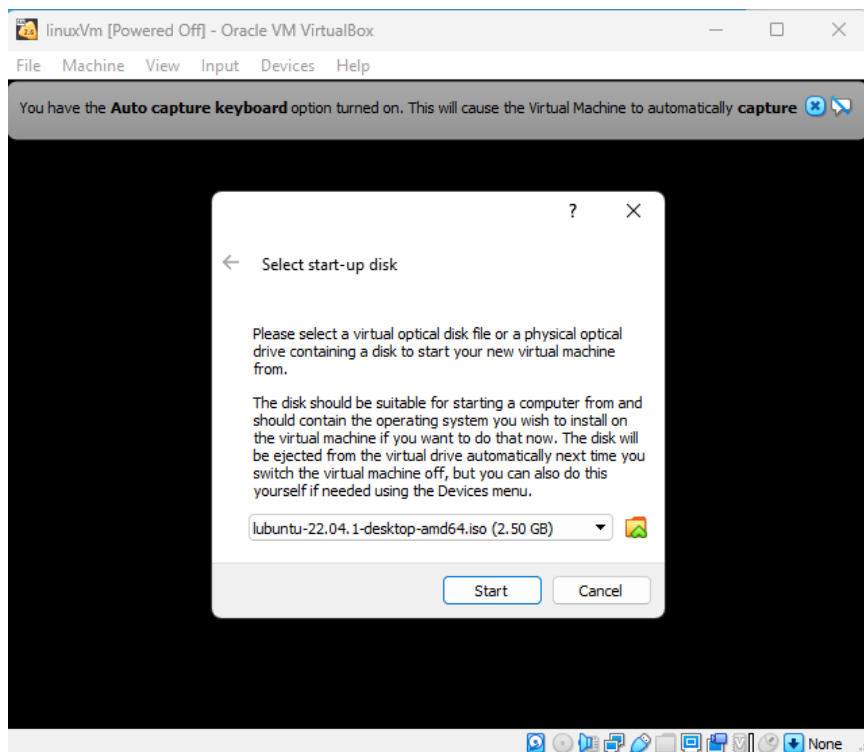
- j. Se selecciona el tamaño del disco rígido virtual, con al menos 12GB debería alcanzar para una instalación básica.



- k. Al finalizar la creación del disco rígido virtual se llega a una pantalla como esta. Se selecciona la máquina virtual creada y se le da start.

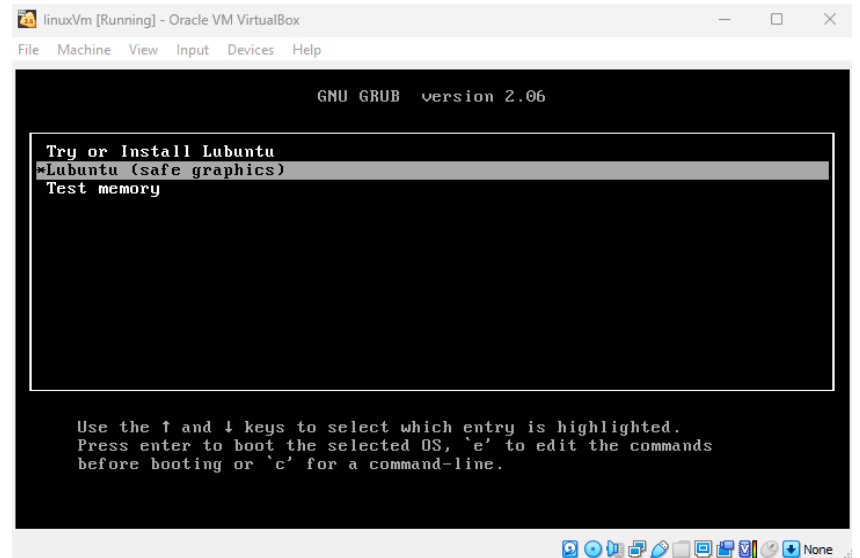


- I. Se selecciona la imagen de Linux que se desea instalar. La imagen a instalar suele tener extensión .iso

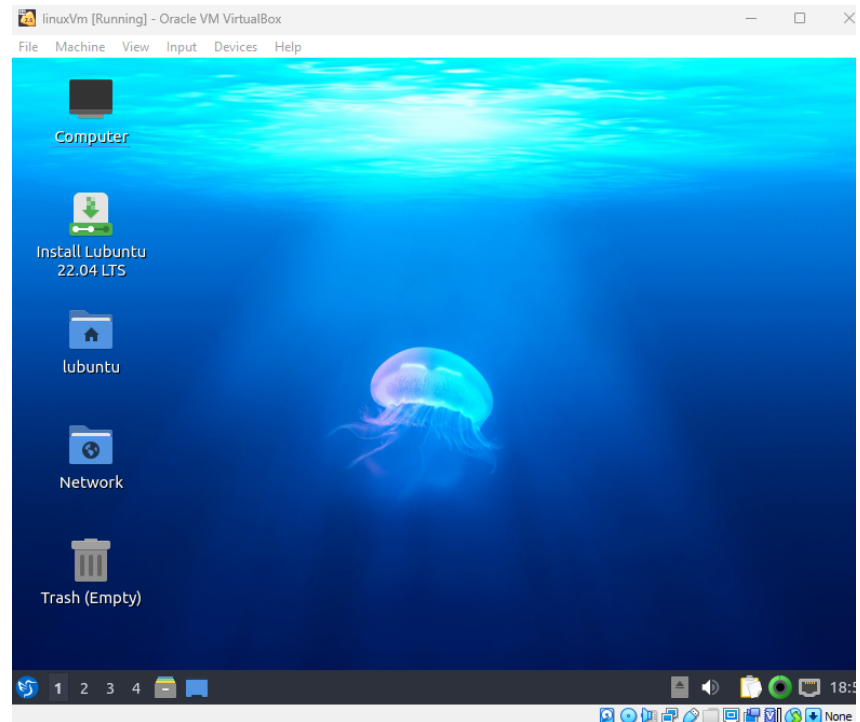


A partir de este punto se realiza la instalación del sistema operativo

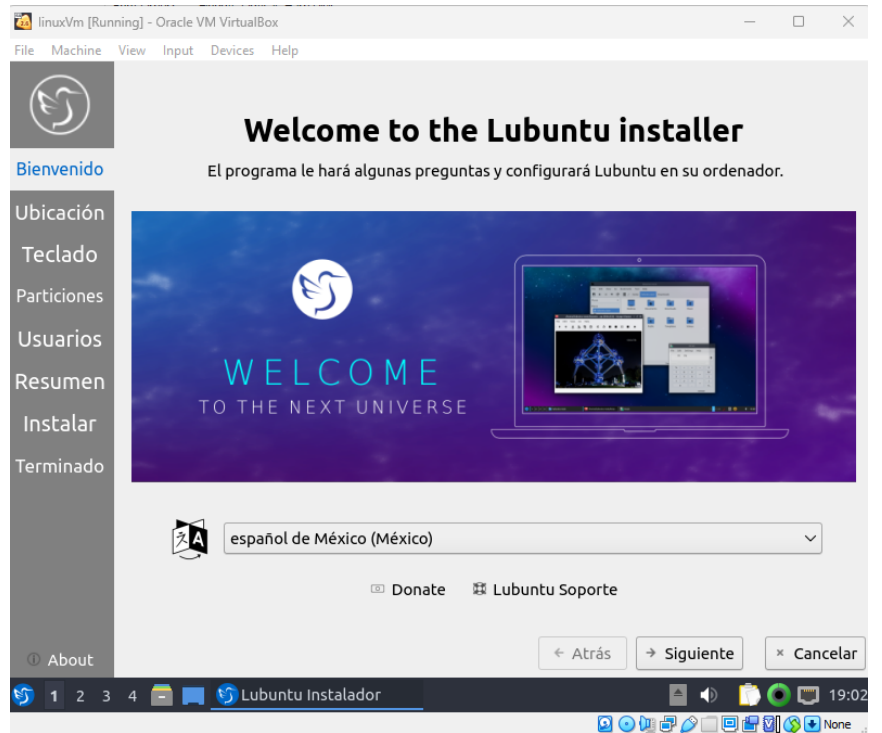
- m. Seleccione la opción Lubuntu (safe graphics)



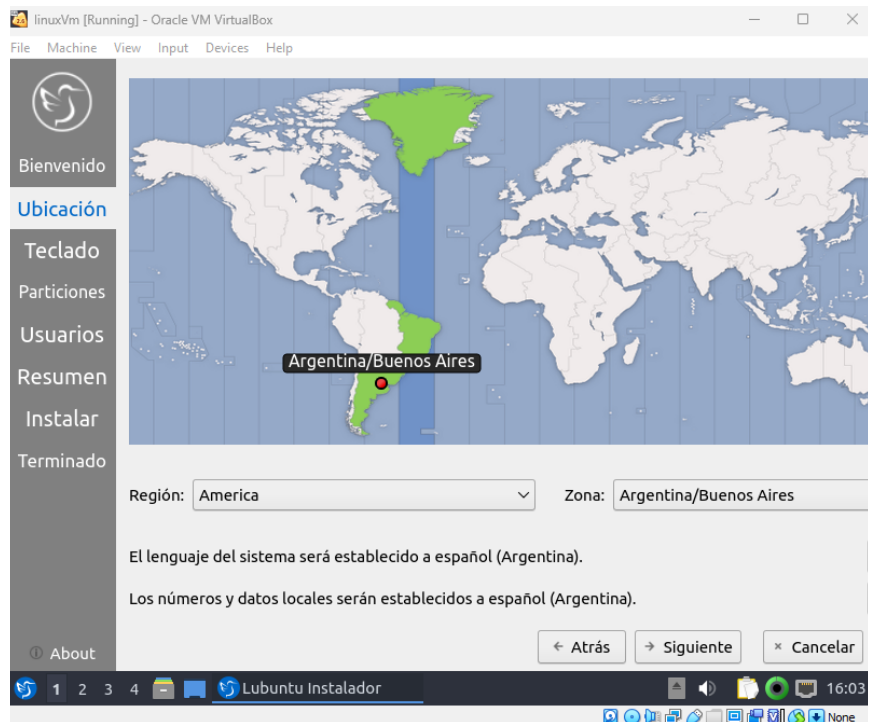
- n. Una vez que inicio el instalador, se selecciona la opción **Install Lubuntu**



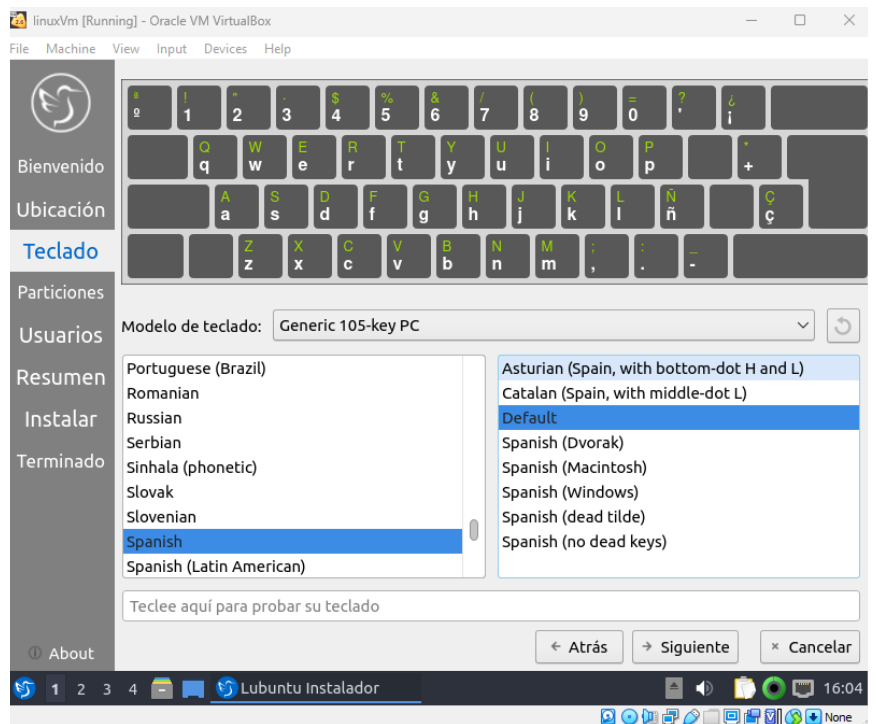
- o. Se selecciona el idioma para la instalación.



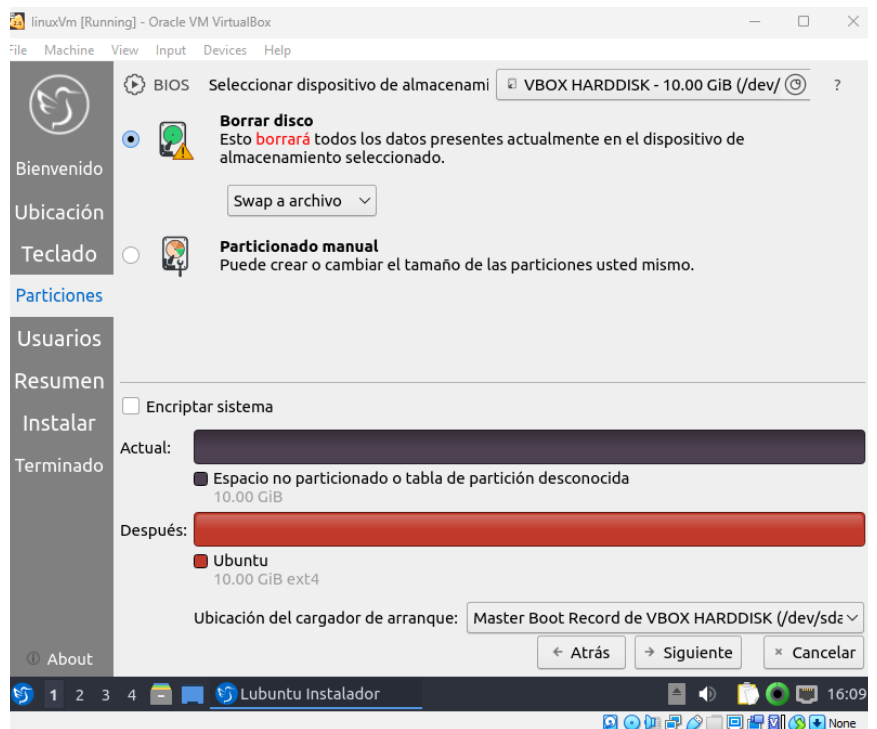
- p. Seleccione su ubicación. Para la configuración regional.



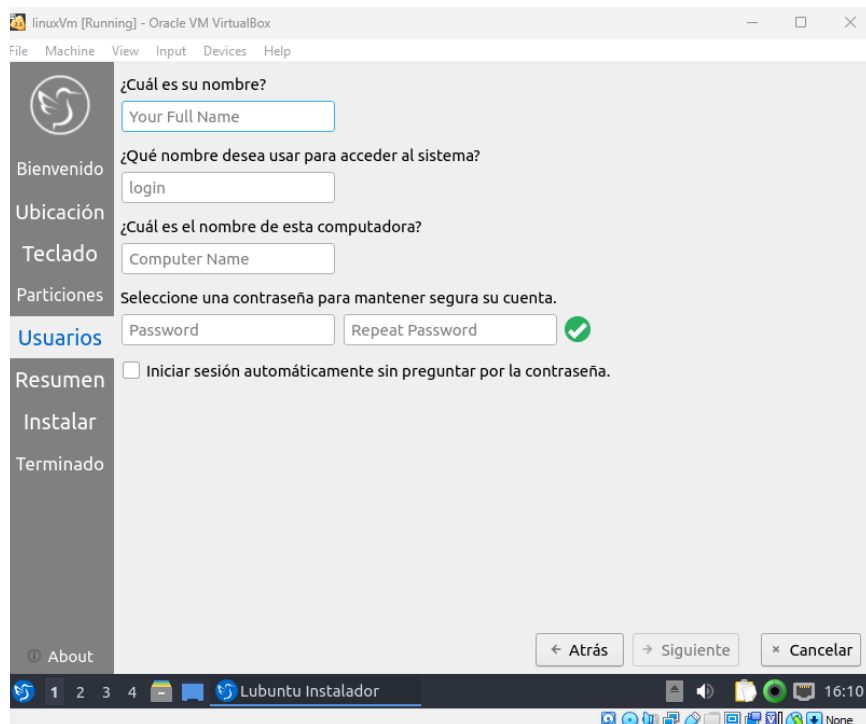
- q. Seleccione la distribución de su teclado.



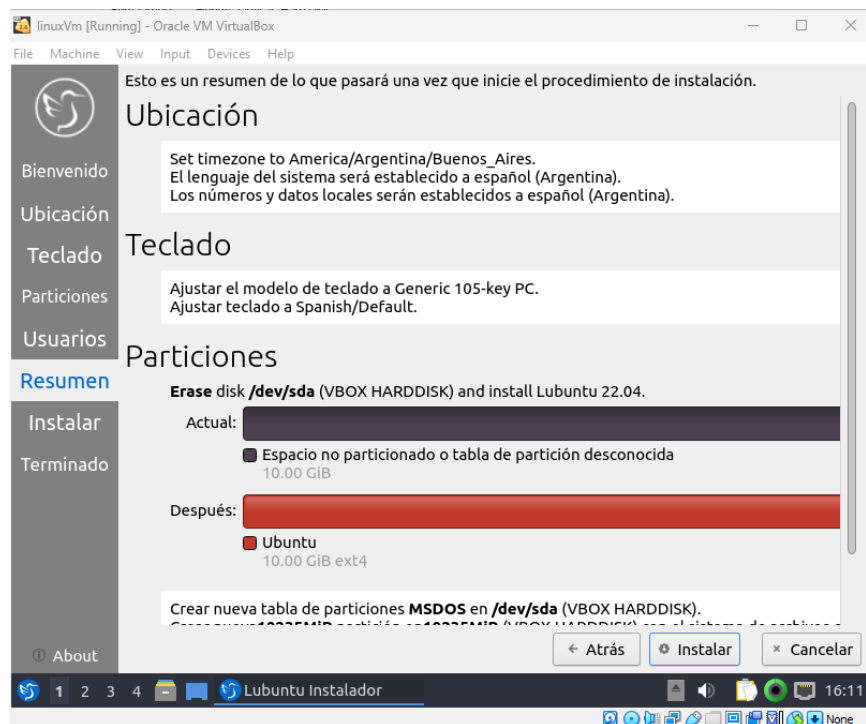
- r. Solicita información de cómo se particionara el disco para la instalación. En el caso de instalarlo en una máquina virtual seleccione Borrar disco.



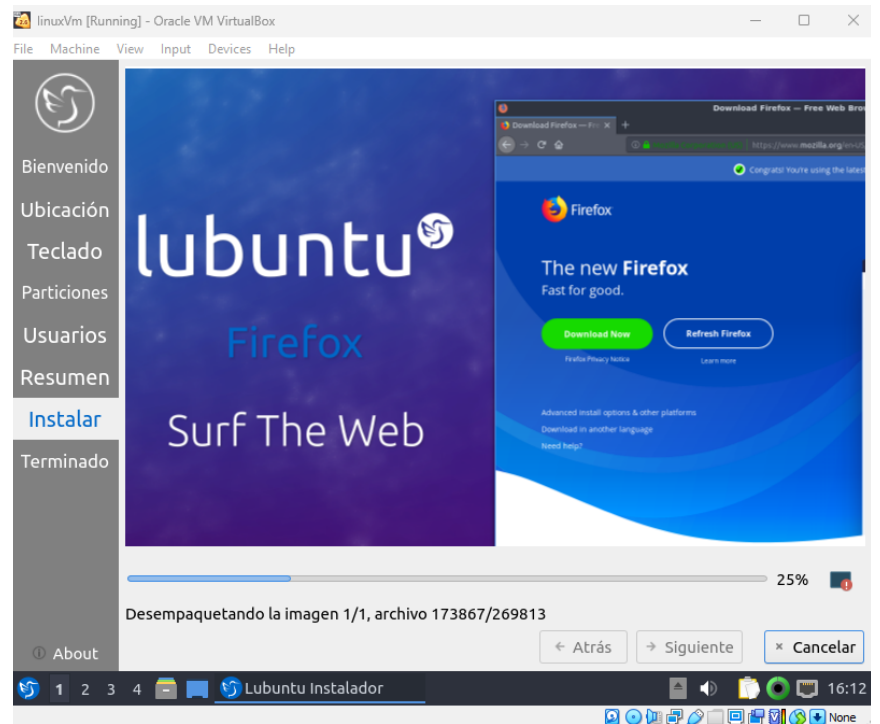
- s. Configurar el nombre de usuario, nombre de la computadora, passwords (Tratar de no olvidar los passwords).



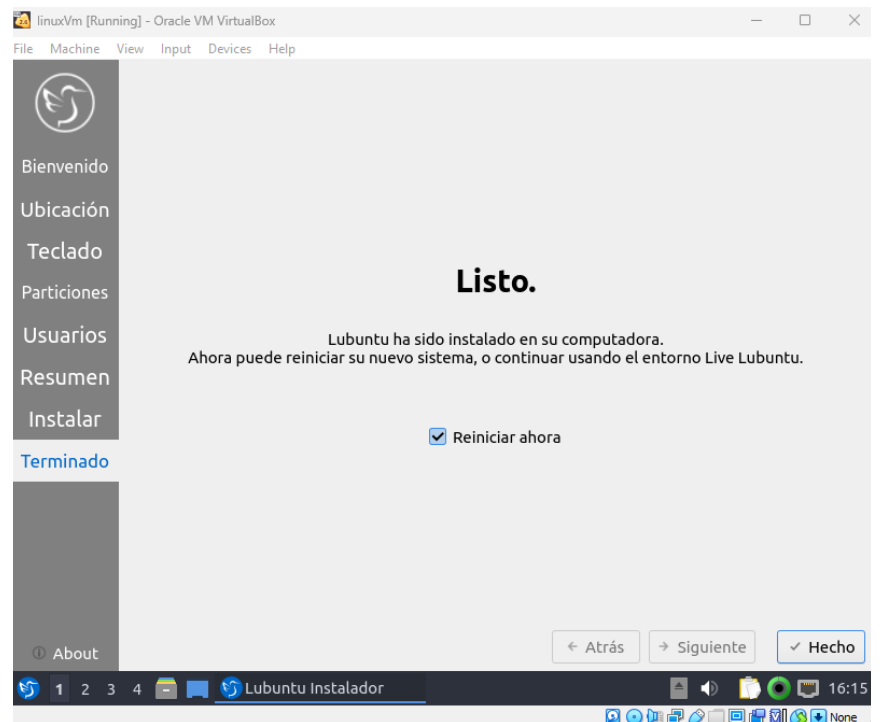
- t. Verifique en el resumen lo seleccionado previamente.



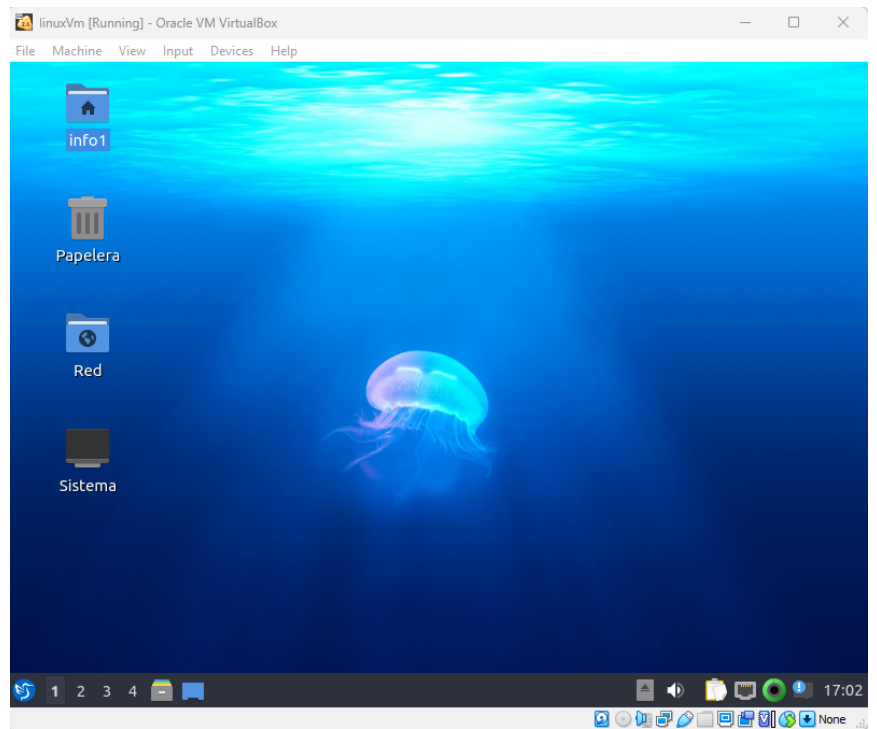
- u. Esperamos a que transcurra la instalación.



- v. Una vez finalizada la instalación, reiniciamos la computadora.



- w. Al reiniciar el sistema operativo estará instalado.



Advertencia:
Recuerde el password y usuario configurados.



2. Comandos básicos de la terminal

En esta sección se exploran algunos comandos útiles de la terminal. Para abrir la terminal puede usar el shortcut CTRL+ALT+T

a. date: Fecha y hora

Con la opción -a Muestra todos los manuales que contienen intro

```
jerome@linuxVm:~$ date  
dom mar 21 21:45:40 -03 2021
```

b. man: Manuales

Muestra el manual de un comando o función que es pasado como parámetro. El manual dispone de siete secciones que se describen a continuación.

1. Programas ejecutables o comandos del shell
2. Llamadas del sistema (Funciones provistas por el kernel)
3. Llamadas a biblioteca.
4. Archivos especiales. (Los encontrados en /dev)
5. Formatos y convenciones de archivos.
6. Juegos
7. Misceláneos
8. Comandos de administración de sistema
9. Rutinas de kernel

Para obtener el manual del manual, una vez dentro del manual si desea salir deberá presionar la letra **q**

```
jerome@linuxVm:~$ man man
```

Para obtener el manual date

```
jerome@linuxVm:~$ man date
```

Para buscar en el manual 1 date

```
jerome@linuxVm:~$ man 1 date
```

Con la opción -a Muestra todos los manuales que contienen intro

```
jerome@linuxVm:~$ man -a intro
```

c. pwd: Directorio actual

Imprime el nombre del directorio actual

```
jerome@linuxVm:~$ pwd  
/home/jerome
```

d. ls: Listar

Lista el contenido de un directorio

```
jerome@linuxVm:~$ ls
Desktop  Documents  Music  Pictures  Public  Templates  Videos
```

Lista el contenido de un directorio incluyendo la fecha y hora de última modificación entre otros datos.

```
jerome@linuxVm:~$ ls -l
total 32
drwxr-xr-x 2 info1 info1 4096 abr  6  2020 Desktop
drwxr-xr-x 4 info1 info1 4096 mar 24 00:18 Documents
drwxr-xr-x 2 info1 info1 4096 abr  6  2020 Music
drwxr-xr-x 2 info1 info1 4096 abr  6  2020 Pictures
drwxr-xr-x 2 info1 info1 4096 abr  6  2020 Public
drwxr-xr-x 2 info1 info1 4096 abr  6  2020 Templates
drwxr-xr-x 2 info1 info1 4096 abr  6  2020 Videos
```

e. mkdir

Crea un nuevo directorio. Crea un directorio llamado pruebaDir

```
jerome@linuxVm:~$ mkdir pruebaDir
jerome@linuxVm:~$ ls
Desktop  Documents  Music  Pictures  pruebaDir  Public  Templates
Videos
```

f. cd

Me permite moverse entre directorios. Me muevo al directorio con el nombre pruebaDir

```
jerome@linuxVm:~$ cd pruebaDir
jerome@linuxVm:~/pruebaDir$
```

g. touch

Crea un archivo vacío o actualiza la fecha de último acceso o última modificación.

```
jerome@linuxVm:~/pruebaDir$ touch pruebaArchivo.txt
jerome@linuxVm:~/pruebaDir$ ls
pruebaArchivo.txt
```

h. rm

Elimina un archivo

```
jerome@linuxVm:~/pruebaDir$ rm pruebaArchivo.txt
```

i. rmdir

Elimina un directorio.

Antes de borrar un directorio debo salir de él, para ello uso el comando **cd**

```
jerome@linuxVm:~/pruebaDir$ cd ..
jerome@linuxVm:~$ rmdir pruebaDir
```

j. clear

Limpia la pantalla

```
jerome@linuxVm:~/pruebaDir$ clear
```

■ ■ ■

3. Instalando paquetes

Cada distribución de linux posee un gestor de paquetes en el caso de Lubuntu es **apt** o **apt-get**. Un gestor de paquetes es un software automatiza el proceso de instalación, actualización, configuración y desinstalación de software en su computadora.

Como ejemplo vamos a instalar VLC, que es un reproductor de audio y video.

```
jerome@linuxVm:~$ apt install vlc
E: Could not open lock file /var/lib/dpkg/lock-frontend - open (13:
Permission denied)
E: Unable to acquire the dpkg frontend lock
(/var/lib/dpkg/lock-frontend), are you root?
```

Luego de ejecutar el comando **apt install vlc** nos devuelve un error indicándonos que no tenemos los permisos suficientes, por ello debemos usar **sudo** antes del comando. Luego de escribir **sudo apt install vlc** y ejecutarlo nos pedirá nuestro password. Finalmente nos indicará el espacio necesario para la instalación, luego nos preguntará y deseamos continuar entonces presionamos **Y** Comenzando con la instalación

```
jerome@linuxVm:~$ sudo apt install vlc
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  fonts-freefont-ttf vlc-bin vlc-l10n vlc-plugin-notify vlc-plugin-qt
vlc-plugin-samba
  vlc-plugin-skins2 vlc-plugin-video-splitter vlc-plugin-visualization
The following NEW packages will be installed:
  fonts-freefont-ttf vlc vlc-bin vlc-l10n vlc-plugin-notify
vlc-plugin-qt vlc-plugin-samba
  vlc-plugin-skins2 vlc-plugin-video-splitter vlc-plugin-visualization
0 upgraded, 10 newly installed, 0 to remove and 410 not upgraded.
Need to get 11,7 MB of archives.
After this operation, 57,0 MB of additional disk space will be used.
Do you want to continue? [Y/n]
```

Para verificar que la instalación se realizó correctamente

```
jerome@linuxVm:~$ vlc
```

■ ■ ■

4. Instalando herramientas de desarrollo y man pages

Para comenzar instalaremos las herramientas de desarrollo básicas.

- El compilador gcc
- El software de control de versiones git
- El editor de texto visual studio code

Para realizar la instalación ejecutaremos el siguiente comando, en el incluimos todos los paquetes a instalar y la opción -Y para que no nos pregunte si estamos de acuerdo con la instalación.

```
jerome@linuxVm:~$ sudo apt install gcc git -y
```

Para instalar los manuales de desarrollo y los manuales en español deberá ejecutar el siguiente comando

```
jerome@linuxVm:~$ sudo apt install manpages-es manpages-es-dev  
manpages-posix manpages-posix-dev -y
```

Realizar el update

```
jerome@linuxVm:~$ sudo apt update
```

Instalando el editor de texto visual studio code

Para el caso de la instalación del editor de texto deberemos ejecutar los siguientes comandos

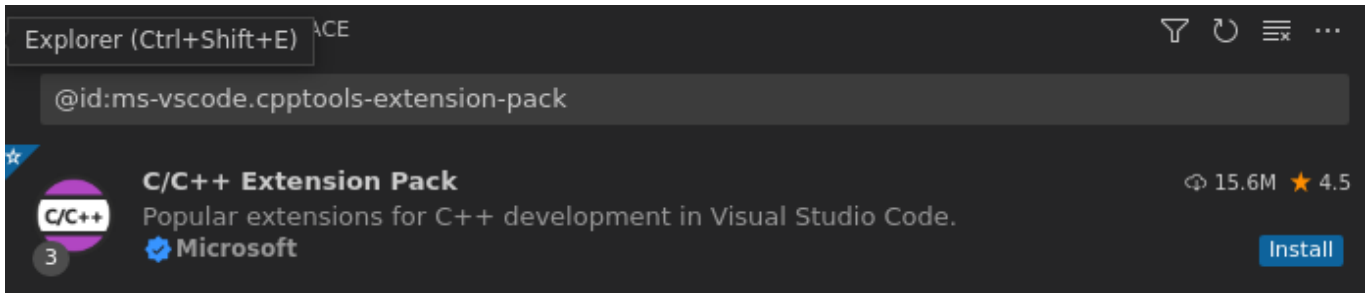
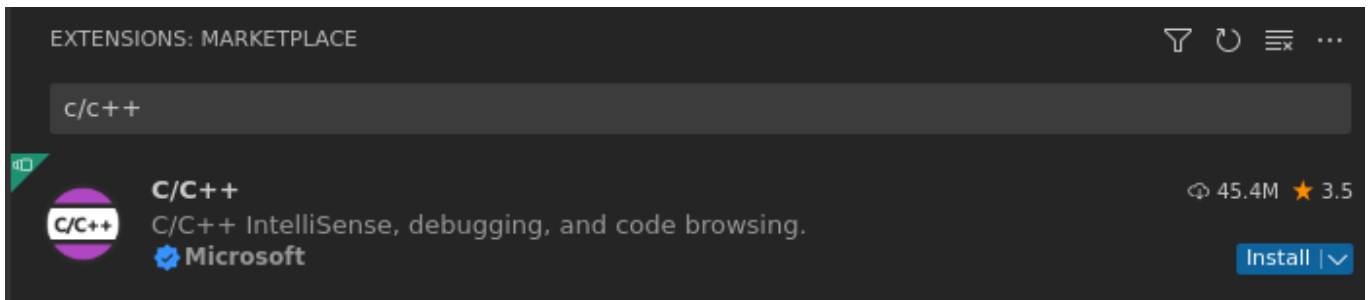
```
jerome@linuxVm:~$ sudo apt update  
  
jerome@linuxVm:~$ sudo apt install software-properties-common  
apt-transport-https wget -y  
  
jerome@linuxVm:~$ wget -q  
https://packages.microsoft.com/keys/microsoft.asc -O- | sudo apt-key  
add -  
  
jerome@linuxVm:~$ sudo add-apt-repository "deb [arch=amd64]  
https://packages.microsoft.com/repos/vscode stable main"  
  
jerome@linuxVm:~$ sudo apt install code
```

Abra el visual studio code desde la terminal ejecutando el comando code

```
jerome@linuxVm:~$ code
```

Instalando extensiones en el editor de texto visual studio code

Para instalar las extensiones presione Ctrl + Shift +X, esto abrirá el buscador de extensiones y deberá buscar las dos que se muestran a continuación

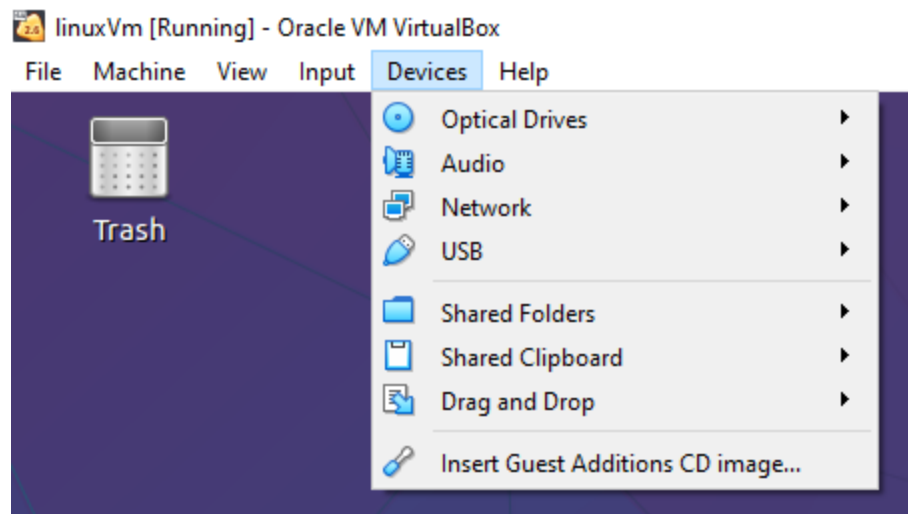


Instalando guest addition en virtualbox

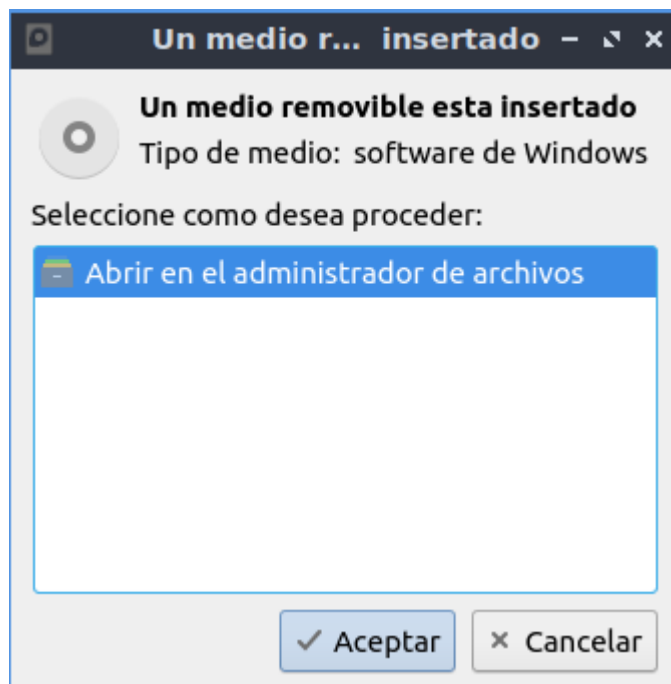
Los guest additions es un paquete de software para agregar algunas funciones a la máquina virtual, como por ejemplo poder usarla en pantalla completa o compartir el portapapeles.

Para instalarlos tenemos que realizar el siguiente procedimiento

- Inicio la máquina virtual y loguearnos en el sistema operativo.
- Instalar sudo update y luego build essentials
- Ir al siguiente menú y seleccionar **Insert Guest Additions CD image...**



- d. En el escritorio del sistema operativo dentro de la máquina virtual deberá aparecer el siguiente icono.



- e. Una vez en el administrador de archivos presione la tecla F4. Esto abrirá una terminal nueva.
f. En la terminal ejecute los siguientes comandos

```
jerome@linuxVm:/media/info1/VBox_Gas_6.0.4$ sudo apt update  
jerome@linuxVm:/media/info1/VBox_Gas_6.0.4$ sudo apt install  
build-essential -y  
jerome@linuxVm:/media/jerome/VBox_Gas_6.0.4$  
sudo ./VBoxLinuxAdditions.run
```

- g. Reinicie la máquina virtual.



5. Mi primer programa en C

1. Una vez que inicio Linux abra una terminal (presione CTRL+ALT+T para abrirla)
2. Desde la terminal se crea un directorio llamado **primerPrograma** utilizando el comando mkdir.

```
jerome@linuxVm:~$ mkdir primerPrograma
```

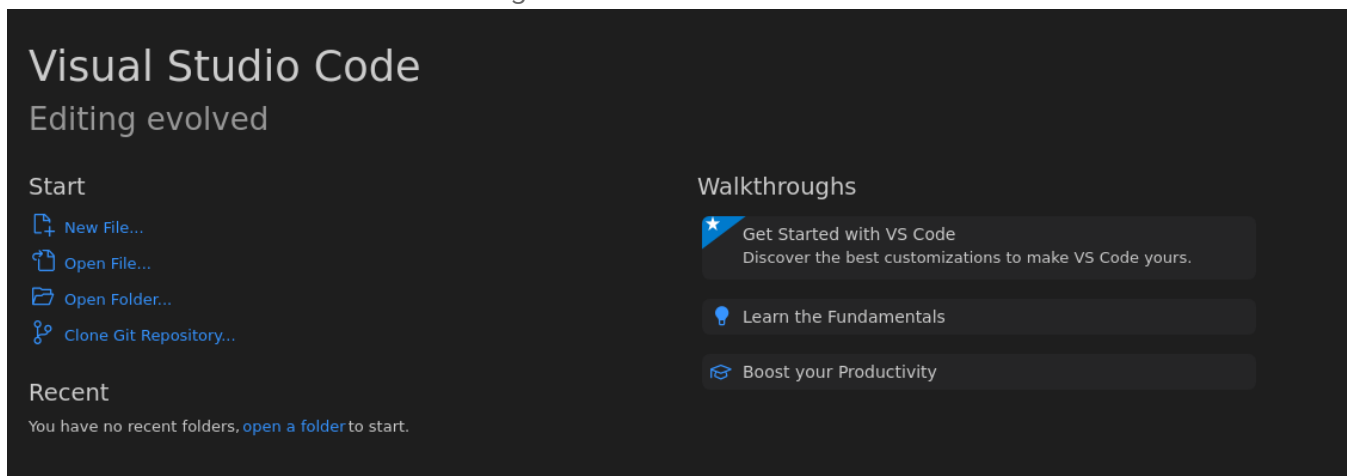
3. Muévase al directorio creado usando el comando cd.

```
jerome@linuxVm:~$ cd primerPrograma
```

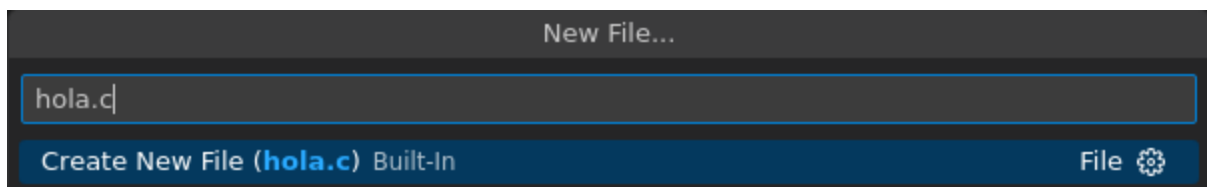
4. Abra el editor de texto visual studio code escribiendo el siguiente comando en la consola code

```
jerome@linuxVm:~$ code
```

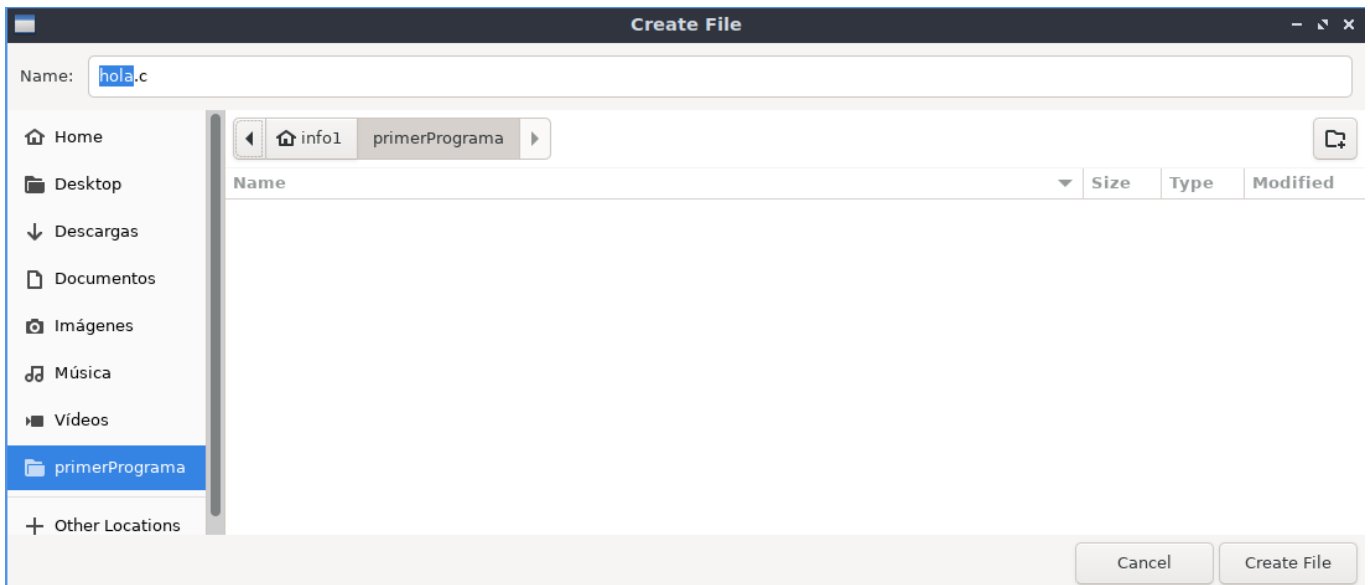
5. Con el visual studio code abierto haga click en el link New File



6. Ingrese el nombre del archivo a crear. En este caso el nombre del archivo es **hola.c** (respete las mayúsculas y minúsculas del nombre) Finalmente presione la tecla Enter



7. Busque la carpeta creada previamente (primerPrograma) y cree el archivo ahí.



8. En el archivo creado copie el siguiente programa y guárdelo. (Ctrl + S)

```
#include <stdio.h>

int main (void)
{
    printf ("Hola!!!\r\n");
    return (0);
}
```

Advertencia:
Verifique que salvo el código.

9. Vuelva a la consola, compile el programa con el siguiente comando.

```
jerome@linuxVm:~/primerPrograma$ gcc hola.c -Wall -ohola.out
```

Eso generará un archivo de salida con el nombre hola.out Verifique su creación listando los archivos del directorio con el comando ls -las

```
jerome@linuxVm:~/primerPrograma$ ls
hola.c    hola.out
```

10. Ejecute el programa compilado, escribiendo en la consola **./hola.out**

```
jerome@linuxVm:~/primerPrograma$ ./hola.out
Hola!!!
```

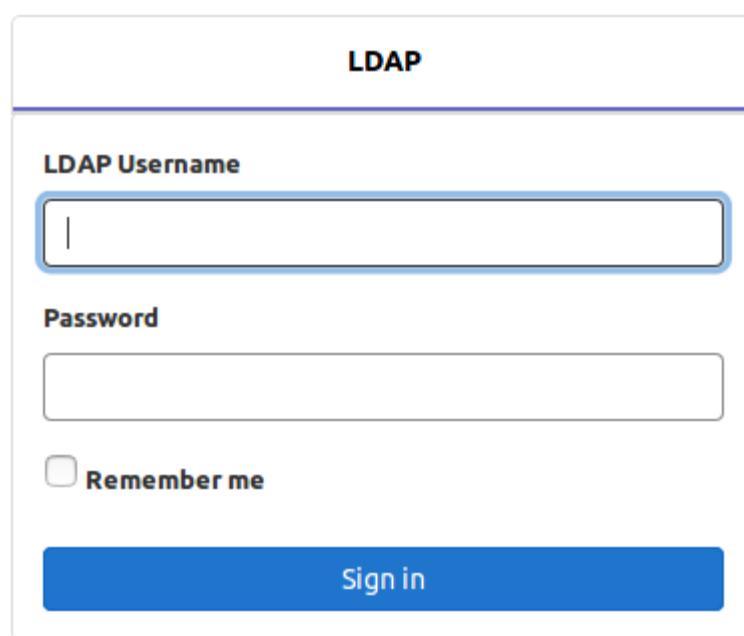


6. Accediendo al repositorio

Se mostrará en sencillos pasos cómo usar git para realizar tareas simples, verificando primero los repositorios desde un navegador y luego usando algunas funciones de git desde la consola.

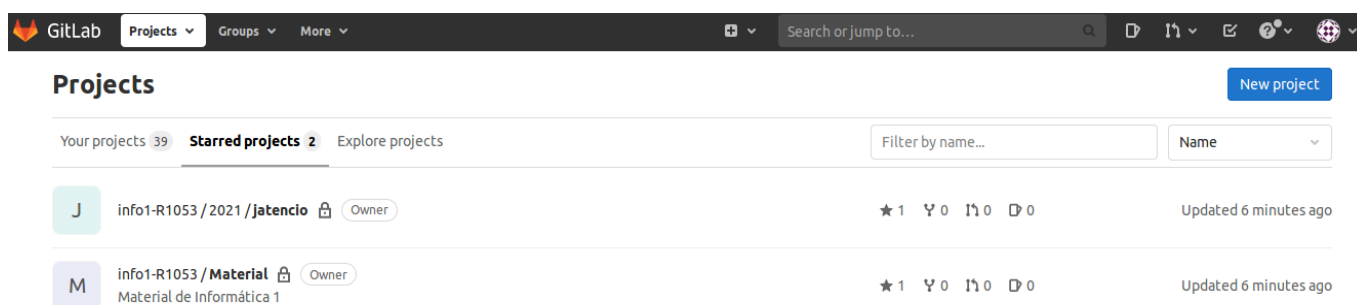
Verificando los repositorios desde el navegador.

1. Abra la ventana del navegador (Firefox, Chrome, chromium, etc) y acceda a la dirección <https://gitlab.frba.utn.edu.ar> y loguese con su usuario sinap



The image shows a login form titled "LDAP". It contains two input fields: "LDAP Username" and "Password". Below the password field is a checkbox labeled "Remember me". At the bottom of the form is a blue button labeled "Sign in".

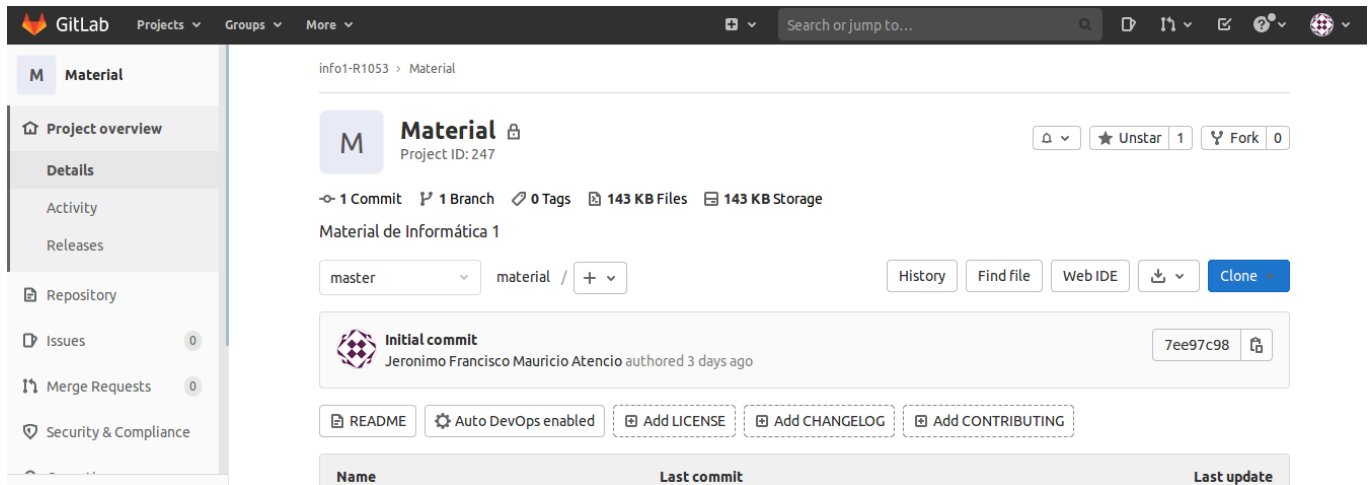
2. Accedera a una página similar a esta donde están al menos dos repositorios
 - a. material: Es el repositorio para descargar el material de la materia
 - b. Un repositorio con su usuario SinAp: Este repositorio lo usarán para subir los trabajos prácticos.Seleccione el repositorio de material



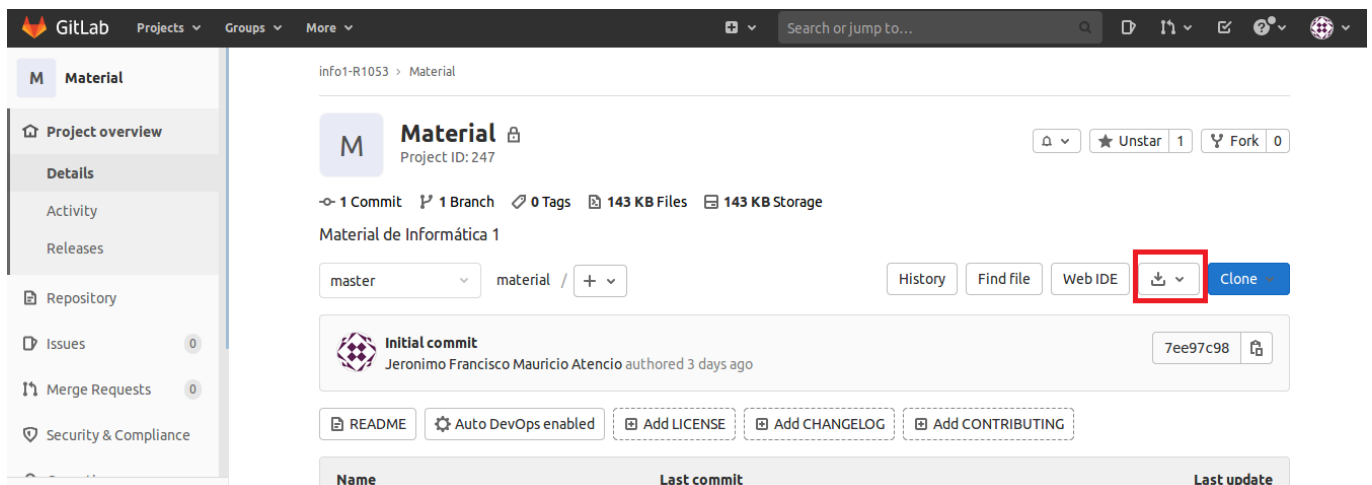
The image shows the GitLab "Projects" page. The header includes the GitLab logo, navigation tabs (Projects, Groups, More), a search bar, and a "New project" button. The main content area shows a list of projects under the "Starred projects" tab. The first project is "info1-R1053 / 2021 / jatencio" by user "J", and the second is "info1-R1053 / Material" by user "M". Both projects are marked as "Owner" and show statistics (1 star, 0 forks, 1 issue, 0 discussions). The last update for both is "6 minutes ago".

| Project Name | Owner | Stars | Forks | Issues | Discussions | Last Updated |
|-------------------------------|-------|-------|-------|--------|-------------|---------------|
| info1-R1053 / 2021 / jatencio | J | 1 | 0 | 1 | 0 | 6 minutes ago |
| info1-R1053 / Material | M | 1 | 0 | 1 | 0 | 6 minutes ago |

3. Accedera a una página como la siguiente



4. Seleccionando el icono indicado en rojo podrá descargar el contenido del repositorio.



Configurando Git por primera vez

Antes de comenzar a usar el git deberá ejecutar los siguientes comandos para configurar su usuario e email. Reemplace lo indicado en azul por sus datos

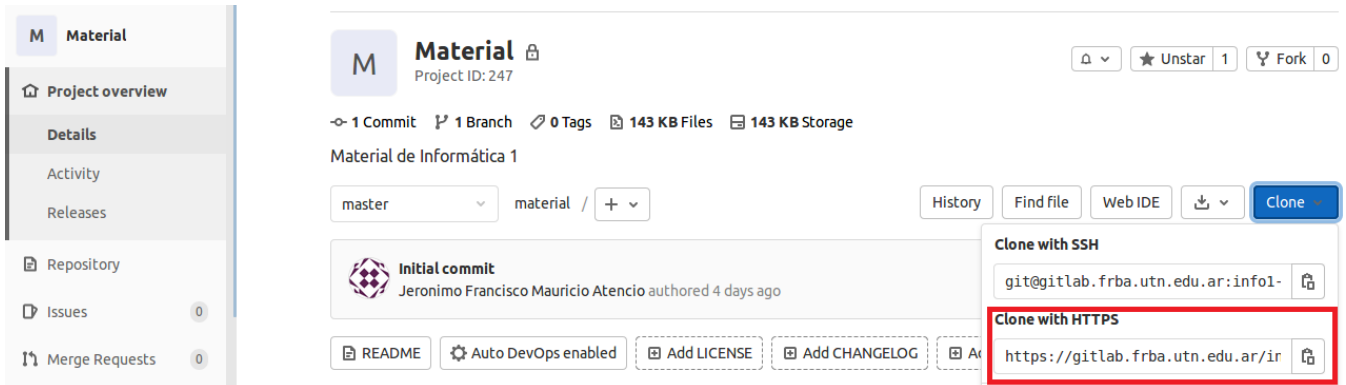
```
jerome@linuxVm:~$ git config --global user.name "usuario"
jerome@linuxVm:~$ git config --global user.email "email"
```

Utilizando el siguiente comando confirme que los datos se configuraron correctamente.

```
jerome@linuxVm:~$ git config --global --list
```


Clonando el repositorio de material.

- Obtener la dirección del repositorio de material. Para ello entre en el repositorio a clonar, presione el botón **Clone** y copie la dirección marcada con el recuadro rojo.



- La dirección obtenida en el punto anterior se coloca luego del comando git clone. Reemplazan lo indicado en azul por la dirección obtenida en el punto anterior.

```
jerome@linuxVm:~$ git clone direccionRepositorio
```

Luego de ejecutar el comando para realizar la clonación, se le solicitará el usuario y el password para acceder al repositorio. Si los datos son correctos se creará un directorio con el nombre del repositorio al cual puede acceder con el comando **cd**.

Por ejemplo para el repositorio de material quedaría

```
jerome@linuxVm:~/jatencio$ git clone
https://gitlab.frba.utn.edu.ar/info1-r1053/material.git
Cloning into 'material'...
Username for 'https://gitlab.frba.utn.edu.ar':
Password for 'https://jatencio@gitlab.frba.utn.edu.ar':
remote: Enumerating objects: 3, done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 3
Unpacking objects: 100% (3/3), done.
Checking connectivity... done.
```

Actualizando el la versión local del repositorio de material desde uno remoto (pull)

- Ingresar a la carpeta del repositorio de material utilizando el comando **cd**.

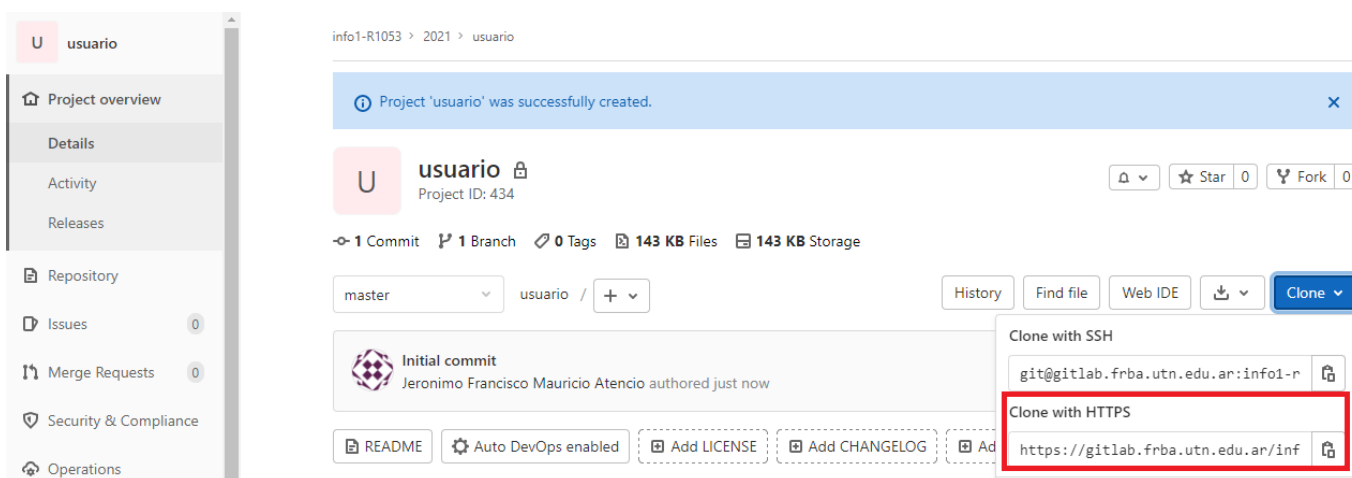
```
jerome@linuxVm:~$ cd material
jerome@linuxVm:~/material$
```

- Realizamos un pull del repositorio remoto. (Lo que está en azul debe reemplazarlo por su nombre de usuario)

```
jerome@linuxVm:~/material$ git pull
Username for 'https://gitlab.frba.utn.edu.ar': usuario
Password for 'https://usuario@gitlab.frba.utn.edu.ar':
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.
From https://gitlab.frba.utn.edu.ar/info1-r1053/material
   c6d8a8a..97a0a6a  master       -> origin/master
Updating c6d8a8a..97a0a6a
Fast-forward
 README.md | 5 ++---
 1 file changed, 2 insertions(+), 3 deletions(-)
```

Clonando su repositorio personal.

- Obtener la dirección del repositorio de su repositorio personal. Para ello entre en el repositorio a clonar, presione el botón **Clone** y copie la dirección marcada con el recuadro rojo.



- La dirección obtenida en el punto anterior se coloca luego del comando **git clone**. Reemplazan lo indicado en azul por la dirección obtenida en el punto anterior.

```
jerome@linuxVm:~$ git clone direccionRepositorio
```

Luego de ejecutar el comando para realizar la clonación, se le solicitará el usuario y el password para acceder al repositorio. Si los datos son correctos se creará un directorio con el nombre del repositorio al cual puede acceder con el comando **cd**.

Por ejemplo para el repositorio de material quedaría

```
jerome@linuxVm:~/jatencio$ git clone
https://gitlab.frba.utn.edu.ar/info1-r1053/2021/usuario.git
Cloning into 'usuario'...
Username for 'https://gitlab.frba.utn.edu.ar':
Password for 'https://usuario@gitlab.frba.utn.edu.ar':
remote: Enumerating objects: 3, done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 3
Unpacking objects: 100% (3/3), done.
Checking connectivity... done.
```

Agregando o modificando un archivo a su repositorio. (commit, push)

1. Vamos a agregar a nuestro repositorio remoto el primer programa en C que hicimos. Para ello copiamos el archivo **hola.c** desde el directorio **primerPrograma** hasta el directorio del repositorio. Reemplazan lo indicado en azul por su nombre de usuario, que coincide en este caso con el nombre del repositorio.

```
jerome@linuxVm:~$ cp ../primerPrograma/hola.c ../usuario
```

2. Ingresar a la carpeta del repositorio de material utilizando el comando **cd**. (Lo que está en azul debe reemplazarlo por su nombre de usuario)

```
jerome@linuxVm:~$ cd ../usuario
```

3. Para iniciar el seguimiento de un archivo tenemos que ejecutar la siguiente línea de comandos, esto lo deberá solo una vez por cada archivo que desee agregar al repositorio.

```
jerome@linuxVm:~/usuario$ git add ./hola.c
```

4. Para hacer el commit ejecutamos el siguiente comando. Luego del -m podemos colocar una leyenda que identifique el commit. El commit almacena nuestro cambio en el repositorio locals

```
jerome@linuxVm:~/usuario$ git commit -m "Primer Commit"
```

5. Finalmente subimos el cambio al repositorio remoto

```
jerome@linuxVm:~/usuario$ git push
```

6. Cada vez que usted modifique un archivo que ya esta agregado para su seguimiento deberá efectuar solamente las operaciones de commit y push para subir los cambios al repositorio remoto



Resumen

- git clone
- git add
- git commit
- git push
- git pull
- git config
- git status



7. Sistemas de numeración

Un sistema de numeración es un conjunto de reglas que permite representar números utilizando símbolos. Los números que construimos en cada sistema de numeración son entidades abstractas que nos permiten representar cantidades enteras o fracciones de cantidades enteras. El sistema de numeración que utilizamos en la vida cotidiana es el Decimal (o base 10) El cual consiste de diez símbolos (los números del cero al nueve)

Clasificación de sistemas de numeración

Los sistemas de numeración los podemos clasificar en dos:

1. No posicionales

Son los más primitivos, representando cantidades haciendo marcas en un palo o nudos en una cuerda por ejemplo. Su característica principal es que las marcas tienen el mismo valor sin importar la posición que ocupen. Esto resulta en que las operaciones de suma y resta consistan en contar, pero sea más dificultoso realizar otras operaciones.

2. Posicionales

En estos sistemas de numeración cualquier número puede ser representado con un conjunto limitado de símbolos, siendo la cantidad de símbolos lo que define la base del sistema de numeración. Su característica principal es que un determinado símbolo toma un valor distinto dependiendo de la posición que ocupa y cada posición corresponde a sucesivas potencias de la base del sistema de numeración. Si bien es posible definir cualquier número entero como base de un sistema de numeración, sólo analizaremos cuatro las siguientes:

- Binario, base dos
- Octal, base ocho
- Decimal, base diez
- Hexadecimal, base dieciséis

Genéricamente podemos construir un número de cualquier base como

$$A = (a_n; a_{n-1}; \dots; a_0, a_{-1}; a_{-2}; \dots; a_{-k})$$

$$A = a_n b^n + a_{n-1} b^{n-1} + \dots + a_0 b^0 + a_{-1} b^{-1} + \dots + a_{-k} b^{-k}$$

$$A = \sum_{j=n}^k a_j b^j$$

Donde:

- b: Base del sistema de numeración
- A: Número válido del sistema de numeración
- a_j : Símbolos válidos (b-1 símbolos)
- $n + 1$: Cantidad de símbolos de la parte entera
- k: Cantidad de dígitos de la parte fraccionaria

Sistema de numeración decimal

Su base es diez y los símbolos válidos son: 0; 1; 2 ;3; 4; 5; 6; 7; 8; 9

$$A = \sum_{j=-k}^n a_j 10^j$$

Ejemplo:

$$1022,74_{10} = 1 * 10^3 + 0 * 10^2 + 2 * 10^1 + 2 * 10^0 + 7 * 10^{-1} + 4 * 10^{-2}$$

Sistema de numeración binario

Su base es dos y los símbolos válidos son: 0; 1

$$A = \sum_{j=-k}^n a_j 2^j$$

Ejemplo:

$$00100100_2 = 0 * 2^7 + 0 * 2^6 + 1 * 2^5 + 0 * 2^4 + 0 * 2^3 + 1 * 2^2 + 0 * 2^1 + 0 * 2^0$$

Sistema de numeración octal

Su base es ocho y los símbolos válidos son: 0; 1; 2 ;3; 4; 5; 6; 7

$$A = \sum_{j=-k}^n a_j 8^j$$

Ejemplo:

$$117_8 = 1 * 8^2 + 1 * 8^1 + 7 * 8^0 = 79_{10}$$

Sistema de numeración hexadecimal

Su base es dieciséis y los símbolos válidos son: 0; 1; 2 ;3; 4; 5; 6; 7; 8; 9; A; B; C; D; E; F

$$A = \sum_{j=-k}^n a_j 16^j$$

Ejemplo:

$$EA80_{16} = E * 16^3 + A * 16^2 + 8 * 16^1 + 0 * 16^0 = 60032_{10}$$

Símbolos para los sistemas de numeración base 2, 8, 10 y 16

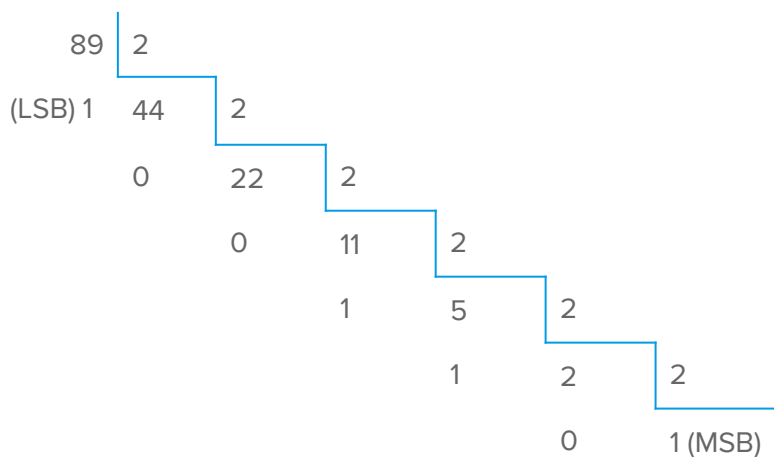
| Decimal | Binario | Octal | Hexadecimal |
|---------|---------|-------|-------------|
| 0 | 0000 | 0 | 0 |
| 1 | 0001 | 1 | 1 |
| 2 | 0010 | 2 | 2 |
| 3 | 0011 | 3 | 3 |
| 4 | 0100 | 4 | 4 |
| 5 | 0101 | 5 | 5 |
| 6 | 0110 | 6 | 6 |
| 7 | 0111 | 7 | 7 |
| 8 | 1000 | 10 | 8 |
| 9 | 1001 | 11 | 9 |
| 10 | 1010 | 12 | A |
| 11 | 1011 | 13 | B |
| 12 | 1100 | 14 | C |
| 13 | 1101 | 15 | D |
| 14 | 1110 | 16 | E |
| 15 | 1111 | 17 | F |

Cambio de base decimal a binario, octal o hexadecimal

El procedimiento para convertir un número decimal en binario, octal o hexadecimal consiste

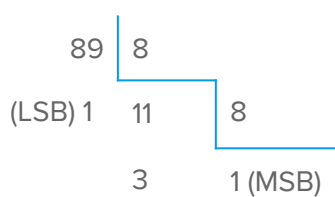
- En tomar el número decimal A y dividirlo por la base que se quiere convertir.
- Luego se repite la operación dividiendo el cociente obtenido por la base destino, hasta que el cociente sea menor que la base destino.
- El resultado se obtiene al concatenar los restos obtenidos de las sucesivas divisiones tomando el primer resto como el dígito menos significativo (LSB: Least Significant Bit) del número en la base destino y el último cociente corresponde al dígito más significativo (MSB: Most Significant Bit) del número en la base destino.

Ejemplo: Convertir el número 89 en base decimal a binario



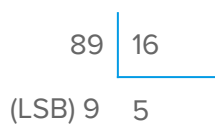
$$89_{10} = 1011001_2$$

Ejemplo: Convertir el número 89 en base decimal a octal



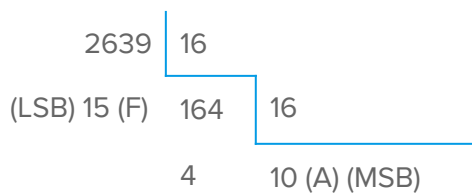
$$89_{10} = 131_8$$

Ejemplo: Convertir el número 89 en base decimal a hexadecimal



$$89_{10} = 59_{16}$$

Ejemplo: Convertir el número 2639 en base decimal a hexadecimal



$$2639_{10} = A4F_{16}$$

Cambio de base de octal a binario y binario a octal

1. Octal a binario

El método para convertir un número octal a binario, consiste en tomar cada dígito octal y convertirlo en binario de forma independiente. Por ejemplo para convertir el número 4126_{10} a binario

| | | | | |
|---------|-----|-----|-----|-----|
| Octal | 4 | 1 | 2 | 6 |
| Binario | 100 | 001 | 011 | 110 |

$$4126_8 = 100001011110_2$$

2. Binario a octal

El método es similar al anterior, pero en este caso se agrupan de a tres dígitos binarios comenzando del bit menos significativo para luego convertir cada grupo de forma individual. Por ejemplo para convertir el número 11101111000_2

| | | | | |
|---------|----|-----|-----|-----|
| Binario | 11 | 101 | 111 | 000 |
| Octal | 3 | 5 | 7 | 0 |

$$11101111000_2 = 3570_8$$

Cambio de base hexadecimal a binario o de binario a hexadecimal

El método para realizar estas conversiones es igual al explicado en el apartado anterior con la salvedad que de los dígitos binarios se agrupan de a 4 dígitos. Ejemplos

1. Hexadecimal a binario

| | | | | |
|-------------|------|------|------|------|
| Hexadecimal | F | 1 | C | 5 |
| Binario | 1111 | 0001 | 1100 | 0101 |

$$F1C5_{16} = 1111000111000101_2$$

2. Binario a hexadecimal

| | | | | |
|-------------|------|------|------|------|
| Binario | 0011 | 1010 | 1011 | 0000 |
| Hexadecimal | 3 | A | B | 0 |

$$0011101010110000_2 = 3AB0_{16}$$

Cambio de base octal a hexadecimal o hexadecimal a octal

Esta conversión de hexadecimal a octal se logra transformando de forma intermedia el número hexadecimal a binario, para luego transformar ese número binario a octal. Para el caso octal a hexadecimal se realiza el mismo proceso de pasar de forma intermedia a binario.

Unidad de información

Un bit (binary-digit) es la mínima unidad de información disponible en sistemas digitales, puede tomar solo dos valores. Los bits pueden ser agrupados en distintas cantidades como por ejemplo:

- Nibble: Es un conjunto de 4 bits, en el puede representarse un dígito hexadecimal.
- Bytes: Es un conjunto de 8 bits.
- Word: Genéricamente es la agrupación de varios bytes por ejemplo:
 - Word16: Agrupa 2 bytes, que son 16 bits
 - Word32: Agrupa 4 bytes, que son 32 bits
 - Word64: Agrupa 8 bytes, que son 64 bits

Ejercicios

Completa la siguiente tabla con los números en las bases correspondientes

| Binario | Octal | Decimal | Hexadecimal |
|----------|--------|-----------|-------------|
| 10100101 | | | |
| | 765 | | |
| | | 2019 | |
| | | | A5A5 |
| | | | FD07ACD367E |
| | | 291338389 | |
| | 205037 | | |
| 11101001 | | | |

Verifique los resultados utilizando la calculadora de Lubuntu



8. Ingreso de datos e impresión en pantalla

Tipos de datos

| Tipo de dato | Tamaño en bytes | Rango de representación | |
|--------------|-----------------|-----------------------------------|---------------------------------------|
| char | 1 (8 bits) | $[(-1) 2^7 \sim (2^7 - 1)]$ | $[-128 \sim 127]$ |
| int | 4 (32 bits) | $[(-1) 2^{31} \sim (2^{31} - 1)]$ | $[-2.147.483.648 \sim 2.147.483.647]$ |
| float | 4 (32 bits) | Ver IEEE 754 | |
| double | 8 (64 bits) | Ver IEEE 754 | |

Secuencias de escape

| Secuencia de escape | Descripción |
|---------------------|---|
| \n | Salto de línea. Avanza el cursor a la línea siguiente. |
| \r | Retorno de carro. Coloca el cursor al inicio de la línea. |
| \t | Tabulador horizontal. Mueve el cursor en un tabulador. |
| \\ | Imprime la contrabarra. |
| \" | Imprime las comillas dobles |

Especificadores de formato

| Especificador de formato | Tipo de dato | Descripción |
|--------------------------|--------------|--|
| %d | int | Imprime/convierte el dato en un entero decimal |
| %3d | int | Imprime/convierte el dato en un entero decimal con 3 dígitos, si hay menos deja los espacios adelante al imprimir. |
| %03d | int | Imprime/convierte el dato en un entero decimal con 3 dígitos, si hay menos dígitos coloca ceros adelante al imprimir. |
| %c | char | Imprime/convierte el dato en un carácter ASCII |
| %o | int | Imprime/convierte el dato en un entero octal |
| %x | int | Imprime/convierte el dato en un entero hexadecimal |
| %f | float | Imprime/convierte el dato al siguiente formato (-) dd.dddddd |
| %lf | double | Imprime/convierte el dato al siguiente formato (-) dd.dddddd |
| %2.3f | float | Imprime/convierte el dato al siguiente formato (-) dd.ddd 2 dígitos antes del punto decimal y 3 después del punto decimal |
| %% | ---- | Imprime % |

Tabla ASCII

Puede obtener la tabla ascii desde la terminal

```
jerome@linuxVm:~$ man ascii
```

Funciones utilizadas

| Función | Archivo de cabecera |
|---------|---------------------|
| printf | stdio.h |
| scanf | stdio.h |

Cómo obtener el manual de una función de biblioteca

Si pido el man de printf como se muestra a continuación, me dará el manual del comando printf y no de la función printf. Podrá notar que en la parte superior al costado de printf aparece un número entre paréntesis indicando el número de manual, en este caso el 1.

```
jerome@linuxVm:~$ man printf
PRINTF(1)                                User Commands                                PRINTF(1)

NAME
    printf - format and print data

SYNOPSIS
    printf FORMAT [ARGUMENT]...
    printf OPTION
... *
```

Para poder ver el manual de la función de librería podemos utilizar la opción -a y recorrerlos hasta encontrar lo que buscamos o buscar en el manual 3 como se muestra a continuación.

```
jerome@linuxVm:~$ man 3 printf
```

Ejemplos

1. Programa que imprime en pantalla una leyenda solicitando un número entero y después lo imprime

```
#include <stdio.h>

int main (void)
{
    int var;    //-- Declaracion de variable entera --

    /* Imprimo en pantalla una leyenda */
    printf ("Ingrese un numero:\r\n");
    /* Espero al que el usuario ingrese un numero entero */
    scanf ("%d", &var);

    //-- Imprimo una leyenda y el valor almacenado en la variable --
```

```
printf ("El numero ingresado es: %d\r\n", var);

//-- Imprimo el número en hexa --
printf ("El numero ingresado es(en hexa): %x\r\n", var);

return (0);
}
```

Escriba y guarde el código anterior en un archivo llamado ejemplo08_00.c

```
jerome@linuxVm:~$ gcc ejemplo08_00.c -Wall -oejemplo08_00.out
jerome@linuxVm:~$ ./ejemplo08_00.out
```

Ejercicios

1. Implemente un programa que utilizando la función printf imprima en pantalla la leyenda Hola Mundo.
La leyenda en pantalla debe verse de la siguiente forma:

```
jerome@linuxVm:~$ ./ejercicio08_01.out
Hola Mundo
```

Recuerde colocar al final de la leyenda \r\n para que baje una línea

2. Implemente un programa en el cual se define una variable de tipo int llamada varInt inicializada con el valor 376 y luego la imprime en pantalla utilizando la función printf. La salida del programa se muestra a continuación.

```
jerome@linuxVm:~$ ./ejercicio08_02.out
La variable varInt contiene el valor 376
```

3. Repita el programa anterior con los siguientes tipos de datos y valores de inicialización
 - a. char: varChar = 'c';
 - b. int: varInt = 0x55AA;
 - c. int: varInt = 017;
 - d. float: varFloat = 1.27;
 - e. double: varDouble = 2.7172;

Advertencia:

Se utiliza el punto y no la coma para separar la parte entera de la parte decimal de un número decimal.

Advertencia:

Los números enteros que comienzan con 0x representan números en hexadecimal.
Los números enteros que comienzan con 0 representan números en octal.

4. Implemente un programa que utilizando la función scanf le solicite al usuario que ingrese un número entero. Luego imprima este número como se muestra a continuación

```
jerome@linuxVm:~$ ./ejercicio08_04.out
Ingrese número: 33
El número ingresado es: 33
```

- 5. Repita el programa anterior utilizando los siguientes tipos de datos.
 - a. char (para este tipo de dato el usuario ingresará una letra)
 - b. float
- 6. Realice un programa que convierta un número entero ingresado por teclado a hexadecimal y octal.

```
jerome@linuxVm:~$ ./ejercicio08_06.out
Ingrese número: 16
El número ingresado fue: 16 (decimal); 0x10 (hexadecimal); 020 (octal)
```

- 7. Modifique el programa anterior para que le permita al usuario ingresar un número en hexadecimal y el programa lo convierta a octal y decimal. Utilice para ello `scanf` con los especificadores de formato `%o` y `%x`
- 8. Realice un programa que le pida al usuario que ingrese una letra e imprima su correspondiente código ASCII en decimal y hexadecimal.

```
jerome@linuxVm:~$ ./ejercicio08_08.out
Ingrese letra: A
El código ASCII para la letra A es: 65; 0x41
```

- 9. Realice un programa que le pida al usuario un número tipo float y lo imprima en pantalla con solo dos dígitos luego del punto decimal.

```
jerome@linuxVm:~$ ./ejercicio08_09.out
Ingrese numero: 1.2785
El ingresados es: 1.27
```



9. Operaciones aritmética - casteo.

Operadores aritméticos

| Operador | Descripción |
|----------|----------------------|
| + | Suma |
| - | Resta |
| * | Multiplicación |
| / | División |
| % | Resto de la división |

Funciones utilizadas

| Función | Archivo de cabecera |
|---------|---------------------|
| sin | math.h |
| cos | math.h |
| tan | math.h |
| sqrt | math.h |
| log | math.h |
| exp | math.h |
| pow | math.h |

Ejemplos

1. Programa que le solicita al usuario 2 números, los almacena en las variables a y b. Luego calcula a^b imprime el resultado.

```
#include <stdio.h>
#include <math.h>
int main (void)
{
    double a, b, r;
    /* Ingreso de datos */
    printf ("Ingrese un numero:\r\n");
    scanf ("%lf", &a);
    printf ("Ingrese un numero:\r\n");
    scanf ("%lf", &b);
    //-- Imprimo una leyenda y el valor almacenado en la variable --
    r = pow (a, b);
    printf ("El resultado es: %lf\r\n", r);
    return (0);
}
```

Escriba y guarde el código anterior en un archivo llamado ejemplo09_01.c

Compilando y ejecutando el programa.

```
jerome@linuxVm:~$ gcc ejemplo09_01.c -Wall -lm -o ejemplo09_01.out
jerome@linuxVm:~$ ./ejemplo09_01.out
```

Se agrega la directiva de linker -l junto con el nombre de la librería a incluir en este caso m. Por ello se observa -lm que debe ser agregado para linkear las funciones de la librería matemática.

Advertencia:

No olvide agregar las library necesarias para linkear su código. Use la directiva -l

2. Programa que calcula el promedio de dos números enteros ingresados por teclado.

```
#include <stdio.h>
#include <math.h>
int main (void)
{
    int a, b;
    float promedio;

    /* Ingreso de datos */
    printf ("Ingrese un numero:\r\n");
    scanf ("%d", &a);
    printf ("Ingrese un numero:\r\n");
    scanf ("%d", &b);
    //-- Imprimo una leyenda y el valor almacenado en la variable --
    p = (float)(a + b) / (float)2.0;
    printf ("El resultado es: %f\r\n", p);
    return (0);
}
```

Ejercicios

1. Realice un programa que sume dos números "reales"(use como tipo de dato float) y muestre el resultado en pantalla.

```
jerome@linuxVm:~$ ./ejercicio09_01.out
Ingrese número: 1.27
Ingrese número: 1
La suma de 1.27 + 1 es igual a 2.27
```

2. Implemente un programa que permita el ingreso de un número real (float) e imprima por separado la parte entera y la decimal

```
jerome@linuxVm:~$ ./ejercicio09_02.out
Ingrese número: 1.27
La parte entera es: 1
La parte decimal es: 0.27
```


3. Realice un programa que le pida al usuario el ingreso de dos números enteros, calcule la división e informe el cociente (entero) y el resto (use el operador %)

```
jerome@linuxVm:~$ ./ejercicio09_03.out
Ingrese número: 101
Ingrese número: 2
El cociente es: 50
El resto es: 1
```

4. Realice un programa que calcule e imprima (con 4 decimales) la raíz cuadrada de un número ingresado por teclado. ¿Qué ocurre si el número ingresado es negativo? Indique su respuesta en un comentario en el código.

```
jerome@linuxVm:~$ ./ejercicio09_04.out
Ingrese número: 2
La raíz cuadrada de 2 es 1.4142
```

5. Realice un programa que calcule e imprima la coseno de un ángulo expresado en grados ingresado por teclado. Se debe tener en cuenta que el parámetro de las funciones trigonométricas debe estar expresado en radianes.

```
jerome@linuxVm:~$ ./ejercicio09_05.out
Ingrese número: 45
El coseno de 45 es 0.707
```

6. Realice un programa que calcule la hipotenusa de un triángulo rectángulo cuyos valores de sus catetos son ingresados por teclado.

```
jerome@linuxVm:~$ ./ejercicio09_06.out
Ingrese número: 1
Ingrese número: 1
La hipotenusa es 1.4142
```

7. Implemente un programa que solicite un número y calcule el logaritmo en base 2 del mismo.

```
jerome@linuxVm:~$ ./ejercicio09_07.out
Ingrese número: 64
El logaritmo en base 2 de 64 es 6
```



10. Sentencias condicionales if y switch-case

Operadores relacionales

| Operador | Descripción |
|----------|-------------------|
| > | Mayor que |
| < | Menor que |
| == | Igual |
| != | Distinto |
| >= | Mayor e igual que |
| <= | Menor e igual que |

Operadores lógicos

| Operador | Descripción |
|----------|------------------------------|
| && | and lógica. (Y lógica) |
| | or lógica (O lógica) |
| ! | not lógico (Negación lógica) |

Sentencias utilizadas

| Sentencias |
|---------------|
| if |
| switch - case |

Ejemplos

1. Programa que le solicita al usuario un número y nos indique si vale cero

```
#include <stdio.h>
int main (void)
{
    int a;
    /* Ingreso de datos */
    printf ("Ingrese un numero:\r\n");
    scanf ("%d", &a);
    //-- Verifico si vale cero --
    if (a == 0) {
        printf ("El usuario ingreso cero\r\n");
    }
    return (0);
}
```

2. Programa que le solicita al usuario un número y nos indica si es mayor o igual a 6 o es menor.

```
#include <stdio.h>
int main (void)
{
    int a;

    /* Ingreso de datos */
    printf ("Ingrese un numero:\r\n");
    scanf ("%d", &a);

    //-- Verifico si es mayor o igual a seis --
    if (a >= 6) {
        printf ("El numero ingresado es mayor o igual a 6\r\n");
    } else {
        printf ("El numero ingresado es menor a 6\r\n");
    }
    return (0);
}
```

3. Programa que le solicita al usuario un número y nos indica si es positivo, negativo o cero.

```
#include <stdio.h>

int main (void)
{
    int a;

    /* Ingreso de datos */
    printf ("Ingrese un numero:\r\n");
    scanf ("%d", &a);

    //-- Verifica si es cero, positivo o negativo --
    if (a == 0) {
        printf ("El usuario ingreso cero\r\n");
    } else {
        if (a > 0) {
            printf ("El usuario ingreso un numero positivo\r\n");
        } else {
            printf ("El usuario ingreso un numero negativo\r\n");
        }
    }
    return (0);
}
```

4. Programa que le solicita al usuario dos números y nos indica cual es mayor o si son iguales

```
#include <stdio.h>

int main (void)
{
    int a, b;

    /* Ingreso de datos */
    printf ("Ingrese un numero:\r\n");
    scanf ("%d", &a);
    printf ("Ingrese un numero:\r\n");
    scanf ("%d", &b);

    //-- Compara dos numeros ingresados --
    if (a == b) {
        printf ("Son iguales\r\n");
    } else {
        if (a > b) {
            printf ("El primero es mayor que el segundo\r\n");
        } else {
            printf ("El segundo es mayor que el primero\r\n");
        }
    }
    return (0);
}
```

5. Programa que le solicita al usuario un número e indica si el mismo está entre 1 y 10 (incluyendo ambos)

```
#include <stdio.h>

int main (void)
{
    int a;

    /* Ingreso de datos */
    printf ("Ingrese un numero:\r\n");
    scanf ("%d", &a);

    //-- Verifico si el numero esta entre 1 y 10 --
    if ((a >= 1) && (a <= 10)) {
        printf ("El numero ingresado esta entre 1 y 10\r\n");
    }
    return (0);
}
```

6. Programa que le solicita al usuario un número e indica si el mismo es mayor a 10 o menor que 1

```
#include <stdio.h>

int main (void)
{
    int a;

    /* Ingreso de datos */
    printf ("Ingrese un numero:\r\n");
    scanf ("%d", &a);

    //-- Verifico si el numero es mayor a 10 o menor a 1 --
    if ((a < 1) || (a > 10)) {
        printf ("El numero es mayor que 10 o menor a 1 \r\n");
    }
    return (0);
}
```

7. Programa que le solicita al usuario un carácter e indica si el mismo es una letra mayúscula

```
#include <stdio.h>

int main (void)
{
    char a;

    /* Ingreso de datos */
    printf ("Ingrese un caracter:\r\n");
    scanf ("%c", &a);

    //-- Verifico si es una letra mayúscula --
    if ((a >= 'A') && (a <= 'Z')) {
        printf ("El caracter ingresado es una letra mayuscula\r\n");
    }
    return (0);
}
```

Consulte el el man del ascii para ver el orden de los caracteres en la tabla.

8. Programa que le solicita al usuario un carácter e indica si el mismo es una vocal minúscula

```
#include <stdio.h>

int main (void)
{
    char a;

    /* Ingreso de datos */
    printf ("Ingrese un caracter:\r\n");
    scanf ("%c", &a);

    //-- Verifico si es vocal --
    switch (a) {
        case 'a':
            printf ("Es vocal\r\n");
            break;

        case 'e':
            printf ("Es vocal\r\n");
            break;

        case 'i':
            printf ("Es vocal\r\n");
            break;

        case 'o':
            printf ("Es vocal\r\n");
            break;

        case 'u':
            printf ("Es vocal\r\n");
            break;

        default:
            printf ("No es vocal\r\n");
            break;
    }
    return (0);
}
```

9. Programa que pide el ingreso de un número entero de 1 dígito e imprime con letras el valor. En caso de que el dígito ingresado sea negativo o tenga más de un dígito escribe una leyenda indicándolo

```
#include <stdio.h>

int main (void)
{
    int a;

    /* Ingreso de datos */
    printf ("Ingrese un numero:\r\n");
    scanf ("%d", &a);

    //-- Verifico si es vocal --
    switch (a) {
        case 0: printf ("cero\r\n");    break;
        case 1: printf ("uno\r\n");    break;
        case 2: printf ("dos\r\n");    break;
        case 3: printf ("tres\r\n");    break;
        case 4: printf ("cuatro\r\n"); break;
        case 5: printf ("cinco\r\n");  break;
        case 6: printf ("seis\r\n");   break;
        case 7: printf ("siete\r\n");  break;
        case 8: printf ("ocho\r\n");   break;
        case 9: printf ("nueve\r\n");  break;

        default:
            printf ("Ingreso invalido\r\n");
            break;
    }
    return (0);
}
```

10. Programa que le solicita al usuario un número entero y una letra e imprime por pantalla un mensaje indicando el

```
#include <stdio.h>

int main (void)
{
    int n;
    char l;

    printf ("Ingrese numero\r\n");
    scanf ("%d", &n);
    printf ("Ingrese letra\r\n");
    scanf ("%*c%c", &l);

    if ((n >= 0) && (n <= 9)) {
        if ((n >= 'a') && (n <= 'z')) {
            printf ("Ingreso correcto\r\n");
        }
    }
}
```

```

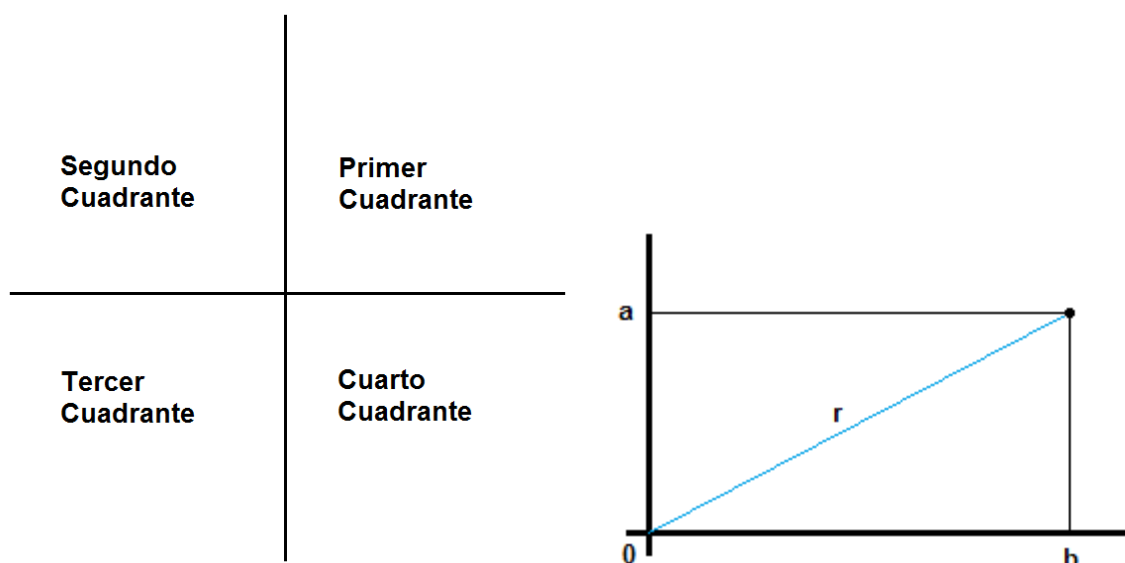
    } else {
        printf ("Letra fuera de rango\r\n");
    }
} else {
    printf ("Numero fuera de rango\r\n");
}

return (0);
}

```



Ejercicios

1. Escriba un programa que permita el ingreso de un número entero e indique si el mismo es par o impar. (use `if` y el operador `%`)
2. Escriba un programa donde ingrese por teclado un par de valores enteros, que representan las coordenadas rectangulares de distintos puntos en el plano. Se pide determinar e informar por pantalla:
 - Si ambos valores son cero.
 - A cuál cuadrante pertenece el punto.
 - La distancia al origen de coordenadas. que se calcula como: $r = \sqrt{a^2 + b^2}$



3. Realice un programa que indique si la letra ingresada es mayúscula, minúscula, un número u otro carácter. Verifique el funcionamiento con los siguientes casos

| Valores de entrada | stdout (pantalla) |
|---------------------|-------------------------|
| desde 'a' hasta 'z' | Es una letra minúscula. |
| desde 'A' hasta 'Z' | Es una letra mayúscula. |
| desde '0' hasta '9' | Es un número |
| Otro carácter | Es otro carácter |

- 
4. Elabore un programa donde se ingresan dos valores reales y el símbolo de la operación ('+', '-', '*', '/'). Se deberá presentar por pantalla, los datos ingresados, la operación y el resultado. Si el símbolo utilizado no correspondiera a ninguna de las cuatro operaciones deberá presentar un mensaje de "Operación no válida". Para leer el símbolo de la operación desde el teclado use `scanf ("%*c%c", &op)`. (El programa deberá resolverse mediante el uso de la estructura switch)
5. Implemente un programa que pida el ingreso de las notas de dos parciales para determinar si el estudiante de informática I:
- Firmó la materia.
 - Promociono.
 - Debe recuperar algún parcial.
- La nota válida estará en el rango de 1 a 10, en caso de error sale del programa.
- 

11. Sentencias de repetición for; while; do-while

Operadores asignación, incremento y decremento

| Operador | Descripción |
|----------|---|
| = | Asignación (igual) |
| += | Incremento. Ejemplo <code>x += 2;</code> es equivalente a <code>x = x + 2;</code> |
| -= | Decremento. Ejemplo <code>x -= 2;</code> es equivalente a <code>x = x - 2;</code> |
| ++ | Pre o post incremento. |
| -- | Pre o post decremento. |

Sentencias utilizadas

| Sentencias |
|------------|
| for |
| while |
| do - while |

Ejemplos

1. Programa de ejemplo de operadores de pre y post incremento.

```
#include <stdio.h>

int main (void)
{
    int w, x, y, z;

    w = 0; x = 0;    y = 0; z = 0;
    //-- Imprimo los valores originales --
    printf ("w = %d\tx = %d\ty = %d\tz = %d\r\n", w, x, y, z);

    //-- Incremento en uno e imprimo --
    w = w + 1;    x++; ++y; z+=1;
    printf ("w = %d\tx = %d\ty = %d\tz = %d\r\n", w, x, y, z);

    //-- Incremento en uno e imprimo --
    printf ("w = %d\tx = %d\ty = %d\tz = %d\r\n", w = w + 1, x++, ++y, z+=1);

    //-- Imprimo los valores --
    printf ("w = %d\tx = %d\ty = %d\tz = %d\r\n", w, x, y, z);

    return (0);
}
```

2. Programa que imprime la leyenda Hola Mundo 10 veces.

```
#include <stdio.h>

int main (void) {
    int i;

    for (i = 0; i < 10; i++) {
        printf ("Hola Mundo\r\n");
    }

    return (0);
}
```

3. Programa que imprime los números enteros del cero al nueve

```
#include <stdio.h>
int main (void)
{int i;
    for (i = 0; i < 10; i++) {
        printf ("%d\r\n", i);
    }

    return (0);
}
```

4. Programa que imprime le pide números al usuario sucesivamente hasta que este ingrese uno mayor que 10

```
#include <stdio.h>
int main (void)
{
    int a;

    do {
        printf ("Ingrese un numero:\r\n");
        scanf ("%d", &a);
    } while (a < 10);

    printf ("El numero ingresado fue mayor que 10\r\n");

    return (0);
}
```

5. Programa que acumula los números ingresados por teclado mientras esta acumulacion no supere el valor 100. Al superar el valor de 100 informa la suma total en pantalla.

```
#include <stdio.h>
int main (void)
{
    int a;
    int suma = 0;


    while (suma <= 100) {
        printf ("Ingrese un numero:\r\n");
        scanf ("%d", &a);
        suma += a;
    }

    printf ("La suma es %d\r\n", suma);

    return (0);
}
```

Ejercicios

1. Implemente un programa que imprima los números enteros pares entre el cero y el cien.
2. Implemente un programa que le pida al usuario dos números enteros e imprima todos los números enteros entre ellos, incluyendo los límites. Si los dos números son iguales deberá imprimir ese número solamente. Ejemplos:
 - El usuario ingresa -1 y 2. El programa debe imprimir: -1; 0; 1; 2
 - El usuario ingresa 2 y -1. El programa debe imprimir: 2; 1; 0; -1
 - El usuario ingresa 2 y 2. El programa debe imprimir: 2
3. Implemente un programa utilizando la sentencia **for** que calcule el promedio de 10 números enteros ingresados por teclado.
4. Realice un programa utilizando la sentencia **do-while** que imprima los números del 0 al 9
5. Realice un programa utilizando la sentencia **while** que imprima los números del 0 al 9
6. Implemente un programa que acumule los números ingresados por teclado mientras esta acumulacion no supere el valor 100. Informe este número en pantalla. Si el usuario ingresa: 10, 80, 20 el programa debe imprimir: 90
7. Realice un programa que calcule el promedio de todas las notas ingresadas por teclado. Las notas válidas están en el intervalo [1; 10] y el ingreso de datos terminará cuando el usuario coloque como nota el valor -1 el cual no deberá tenerse en cuenta para el promedio. Si el usuario ingresa un número fuera del rango deberá informar con una leyenda en pantalla y continuar el ingreso de datos. Finalmente muestre el promedio con 2 decimales.
8. Realice un programa que calcule el promedio de todas las notas ingresadas por teclado. Las notas válidas están en el intervalo [1; 10] y el ingreso de datos terminará cuando el usuario coloque como



nota el valor -1 el cual no deberá tenerse en cuenta para el promedio. Si el usuario ingresa un número fuera del rango deberá informar con una leyenda en pantalla y continuar el ingreso de datos. Además deberá controlar que la cantidad de notas válidas ingresadas sea mayor que tres en caso contrario deberá indicarle al usuario que continúe con el ingreso de datos. Finalmente muestre el promedio con 2 decimales.

9. Escriba un programa que le permita al usuario jugar a adivinar un número secreto entre 0 y 9. Para ello le pedirá que ingrese un número (el secreto) y luego le pedirá que ingrese números hasta adivinarlo. Si el usuario ingresa un número fuera del rango válido [0; 9] deberá indicarlo y en el caso del número secreto deberá continuar pidiendo hasta que ingrese el valor en el rango válido. El programa continuará pidiendo números hasta que el usuario lo adivine indicando "GANASTE" con lo cual debe de comenzar nuevamente el juego o se equivoque 3 veces indicando "NO GANASTE, INTENTALO OTRA VEZ" .
10. Implemente un programa que le pida al usuario que ingrese un número entero y luego informe la cantidad de dígitos del mismo. Por ejemplo: Si el usuario ingresa el número 1234 el programa deberá indicar que tiene 4 dígitos.



12. Directiva de precompilador define

Directivas de precompilación

| Directiva | Descripción |
|-----------|--------------------------------|
| define | Define una constante simbólica |

Ejemplos

1. Programa que imprime la leyenda Hola Mundo 10 veces. Coloque este umbral en un define.

```
#include <stdio.h>
#define CANT ((int)10)

int main (void)
{
    int i;

    for (i = 0; i < CANT; i++) {
        printf ("Hola Mundo\r\n");
    }

    return (0);
}
```

2. Programa que calcula el promedio de CANT datos ingresados.

```
#include <stdio.h>
#define CANT ((int)10)

int main (void)
{
    int i;
    int acc = 0, n;
    float promedio;

    for (i = 0; i < CANT; i++) {
        printf ("Ingrese numero\r\n");
        scanf ("%d", &n);
        acc += n;
    }

    promedio = acc / (float)CANT;

    printf ("Promedio %f\r\n", promedio);

    return (0);
}
```

Advertencia:
No se puede modificar el valor de un `define` en ejecución, por ejemplo NO puede hacer `CANT = 0`

Ejercicios

1. Implemente un programa que le pida un número al usuario e indique si el número es mayor a 100. Coloque este umbral en un `define`.
2. Implemente un programa que calcule el factorial de un número entero positivo ingresado por teclado. El factorial de n es el producto de todos los enteros positivos hasta n inclusive.
 - Para $n > 0 \Rightarrow n! = 1 \times 2 \times 3 \times \dots \times n$
 - Para $n = 0 \Rightarrow n! = 1$

Verifique todas las condiciones que considere pertinentes para que el resultado obtenido sea correcto. Determine el número máximo al cual le puede calcular el factorial. Verifique el funcionamiento del programa con los siguientes valores.

| x | x! (salida de Stdout) |
|----|--|
| -1 | No puedo calcular el factorial de un número negativo |
| 0 | 1 |
| 1 | 1 |
| 2 | 2 |
| 3 | 6 |
| 12 | 479001600 |

3. Ingresar un número entero por teclado y determinar si es primo. Para calcular si un número N es primo, hay que dividir el número N por todos los números enteros desde $N-1$ hasta 2 y si alguna de las divisiones da que el resto es cero, entonces el número N no es primo. Por ejemplo los primeros diez números primos son 2, 3, 5, 7, 11, 13, 17, 19, 23, 29
4. Implemente un programa que le pida al usuario que ingrese 10 (use `define`) números enteros e informe el mayor y el menor de todos los ingresados.
5. Implemente un programa que le pida al usuario que ingrese 10 (use `define`) número enteros positivo y cuente la cantidad de números pares e impares ingresados.



13. Funciones

Constantes simbólicas

| Constante | Descripción |
|-----------|-------------------------------|
| M_PI | Número Pi, definido en math.h |

Ejemplos

1. Función que imprime la leyenda "Hola Mundo".

```
#include <stdio.h>

void imprimir (void)
{
    printf ("Hola Mundo\r\n");
}

int main (void)
{
    imprimir ();
    return (0);
}
```

2. Función que imprime la leyenda "Hola Mundo", la cantidad de veces que se paso como parametro.

```
#include <stdio.h>

void imprimir (int cant)
{
    int i;

    for (i = 0; i < cant; i++) {
        printf ("Hola Mundo\r\n");
    }
}

int main (void)
{
    int c;

    printf ("Ingrese cantidad\r\n");
    scanf ("%d", &c);

    imprimir (c);

    return (0);
}
```


3. Función que devuelve el número pasado como parámetro sumándole 1.

```
#include <stdio.h>
#define CANT ((int)10)

int suma1 (int a)
{
    int r;

    r = a + 1;

    return (r);
}

int main (void)
{
    int c, r;
    printf ("Ingrese cantidad\r\n");
    scanf ("%d", &c);

    r = suma1 (c);
    printf ("Resultado = %d\r\n", r);

    return (0);
}
```

4. Función que convierta de mayúsculas a minúsculas la letra pasada como parámetro

char pasaAminusculas(char a)

```
#include <stdio.h>
#define CANT ((int)10)

char pasaAminusculas (char a)
{
    char r;

    if ((a >= 'A') && (a <= 'Z')) {
        r = (a - 'A') + 'a';
    } else {
        r = a;
    }

    return (r);
}
```

```

int main (void)
{
char c, r;

    printf ("Ingrese letra\r\n");
    scanf ("%c", &c);

    r = pasaAminusculas (c);
    printf ("Resultado = %c\r\n", r);

    return (0);
}

```

5. Función que devuelve la suma de dos números enteros pasados como parámetros

```

#include <stdio.h>

int sumaDosNumeros (int a, int b)
{
int r;

    r = a + b;

    return (r);
}

int main (void)
{
int x, y, r;
    printf ("Ingrese numero\r\n");
    scanf ("%d", &x);
    printf ("Ingrese numero\r\n");
    scanf ("%d", &y);

    r = sumaDosNumeros (x, y);
    printf ("Resultado = %d\r\n", r);

    return (0);
}

```

6. Ejemplo de printf con constantes útiles para realizar debugging.

```

#include <stdio.h>

void func (void)
{
    /* Esta linea imprime el nombre del archivo, la funcion y el numero de
       linea donde se encuentra. */
    printf ("Debug : %s, %s, %d\r\n", __FILE__, __FUNCTION__, __LINE__);
}

```

```

int main (void)
{
    // Esta linea imprime la fecha y hora de compilacion
    printf ("Debug : %s, %s\r\n", __DATE__, __TIME__);

    func ();

    /* Esta linea imprime el nombre del archivo, la funcion y el numero de
       linea donde se encuentra. */
    printf ("Debug : %s, %s, %d\r\n", __FILE__, __FUNCTION__, __LINE__);

    return (0);
}

```

Advertencia:
 Recuerde definir la función o colocar el prototipo antes de la llamada a la función.

Ejercicios

1. Implemente una función que calcule el área de un círculo. Utilice la constante de `M_PI` de `math.h`. El prototipo es

```
float areaCirculo (float radio);
```

2. Implemente una función que calcule el perímetro de un círculo. Utilice la constante de `M_PI` de `math.h`. El prototipo es

```
float perimetroCirculo (float radio);
```

3. Implemente una función a la cual le pase un carácter como parámetro y me devuelva
 - 0: si el carácter es una letra mayúscula.
 - 1: si el carácter es una letra minúscula.
 - 2: si el carácter es un número.
 - 3: en caso que no sea ninguno de los anteriores.

El prototipo es

```
int filtroASCII (char caracter);
```

4. Implemente una función que realice las cuatro operaciones básicas entre dos números de tipo `float` y retorne el resultado. El prototipo de la función es el siguiente

```
float calculo (float opA, float opB, char op)
```

Donde:

- `opA` y `OpB` son los números con los cuales se debe realizar la operación
- `op`: La operación a realizar
 - '+': Realiza la suma.
 - '-': Realiza la resta.
 - '*': Realiza el producto
 - '/': Realiza la división
- La función devuelve cero si la operación es inválida.

5. Implemente una función que le pase como parámetro dos números que representan los catetos de un triángulo rectángulo y me devuelva la hipotenusa. El prototipo es

```
float calcHipo (float catetoA, float catetoB)
```

6. Implemente una función que dado un número pasado como parámetro determine si ese número es primo o no. El prototipo es

```
int esPrimo (int n)
```

Donde:

- n: es el número que debe determinarse si es primo o no.
 - La función devuelve
 - 0 si n no es primo
 - 1 si n es primo
 - -1 si n es menor o igual a cero.
7. Implemente una función que calcule el factorial de un número pasado como parámetro. Si el factorial no puede ser calculado la función debe devolver cero.

```
int factorial (int n);
```



14. Vectores y strings

Ejemplos

1. Programa en el cual se define un vector de diez números enteros y se inicializa en tiempo de ejecución con los números del 0 al 9. Luego lo imprime en orden ascendente y descendente.

```
#include <stdio.h>

#define CANT ((int)10)

int main (void)
{
    int v[CANT];

    //-- Inicializo el vector --
    for (i = 0; i < CANT; i++) {
        v[i] = i;
    }

    //-- Imprimo en orden ascendente --
    for (i = 0; i < CANT; i++) {
        printf ("%d\r\n", v[i]);
    }

    //-- Imprimo en orden descendente --
    for (i = CANT - 1; i >= 0; i--) {
        printf ("%d\r\n", v[i]);
    }

    return (0);
}
```

2. Programa en el cual se define un vector de diez números enteros y se inicializa en tiempo de compilación con la tabla de multiplicar del 5. Pida al usuario que ingrese un número entre cero y nueve, para luego calcular la multiplicación por cinco de dicho número indexando el vector.

```
#include <stdio.h>

#define CANT ((int)10)

int main (void)
{
    int m5[CANT] = {0, 5, 10, 15, 20, 25, 30, 35, 40, 45};
    int num;

    //-- Ingreso numero --
    printf ("Ingrese numero\r\n");
    scanf ("%d", &num);
}
```

```

    if ((num >= 0) && (num <= 9)) {
        printf ("El resultado es: %d\r\n", m5[num]);
    } else {
        printf ("Imposible calcular\r\n");
    }

    return (0);
}

```

3. Programa en el cual se le pide al usuario que ingrese 10 números, para luego imprimir de forma separada los números pares e impares ingresados. El vector se inicializa en cero en tiempo de compilación.

```

#include <stdio.h>
#define CANT ((int)10)

int main (void)
{
    int v[CANT] = {0};
    int i;

    //-- Ingreso numero --
    for (i = 0; i < CANT; i++) {
        printf ("Ingrese numero\r\n");
        scanf ("%d", &v[i]);
    }

    printf ("Los pares son: \r\n");
    for (i = 0; i < CANT; i++) {
        if ((v[i] % 2) == 0) {
            printf ("%d.%d\r\n", i, v[i]);
        }
    }

    printf ("Los impares son: \r\n");
    for (i = 0; i < CANT; i++) {
        if ((v[i] % 2) != 0) {
            printf ("%d.%d\r\n", i, v[i]);
        }
    }

    return (0);
}

```

Advertencia:

Verifique siempre que el índice del vector esté dentro del rango definido.

4. Programa que muestra cómo generar 10 números pseudoaleatorios.

```
#include <stdio.h>
#include <time.h>
#include <stdlib.h>

#define CANT ((int)10)

int main (void)
{
    int i;

    srand(time(NULL));

    for (i = 0; i < CANT; i++) {
        printf ("%d\r\n", rand());
    }

    return (0);
}
```

5. Programa que le pide al usuario que ingrese una palabra para luego indicar la cantidad de vocales que tiene la misma.

```
#include <stdio.h>
#define CANT ((int)32)
#define CANT_VOCALES ((int)10)
int main (void)
{
    char v[CANT];
    char vocales[CANT_VOCALES] = {'a', 'e', 'i', 'o', 'u', 'A', 'E', 'I', 'O', 'U'};
    int vocalesCont = 0;
    int i, j;
    //-- Ingreso palabra --
    printf ("Ingrese palabra\r\n");
    scanf ("%s", &v[0]);

    //-- Cuento las vocales --
    i = 0;
    while (v[i] != '\0') {
        for (j = 0; j < sizeof (vocales); j++) {
            if (v[i] == vocales[j]) {
                vocalesCont++;
            }
        }
        i++;
    }
    printf ("La cantidad de vocales es: %d\r\n", vocalesCont);
    return (0);
}
```

6. Programa que inicializa un string en tiempo de compilación y lo muestra en pantalla.


```
#include <stdio.h>
#define CANT ((int)32)
#define CANT_VOCALES ((int)10)
int main (void)
{
char str0[] = "Hola";
char str1[] = {'h', 'o', 'l', 'a', '\0'};

    //-- Ingreso palabra --
    printf ("Palabra %s\r\n", str0);
    printf ("Palabra %s\r\n", str1);

    return (0);
}
```

Ejercicios

1. Escriba un programa en el cual defina un vector que almacene todas las letras del alfabeto, salvo la eñe (use como tipo de dato char). Inicialicelo en tiempo de ejecución. Imprímalo en orden alfabético
2. Implemente un código que genere un número pseudo aleatorio entre [0; 9] , luego el programa debe pedirle al usuario números e indicarle si acertó o no el número generado pseudo aleatoriamente. Si el usuario repite un número el programa deberá indicarlo, lo mismo si el usuario ingresa uno fuera de rango.
3. Realice un programa que permita al usuario ingresar las alturas de un grupo de como máximo cien personas. El fin del ingreso de datos ocurre cuando la altura ingresada sea menor que cero. Luego se le pedirá al usuario que ingrese dos valores de altura y el programa debe indicar la cantidad de personas con alturas en ese rango. Si el intervalo ingresado por el usuario es inválido indíquelo por pantalla.
4. Implemente un programa que le pida al usuario números enteros y los almacene en 4 vectores diferentes según su tipos.
 - Positivos y cero.
 - Negativos.
 - Pares.
 - Impares.El usuario ingresará diez valores y luego el programa deberá imprimir por pantalla la cantidad de números almacenados en cada vector y posteriormente los datos almacenados en cada uno de ellos.
5. Implemente un programa que le pida al usuario que ingrese una palabra por teclado y la imprima en mayúscula por pantalla. Puede suponer que la palabra no tendrá más de 16 caracteres.
6. Implemente un programa que le pida al usuario que ingrese una palabra por teclado e informe la cantidad de caracteres que esta posee sin contar el '\0'. Puede suponer que la palabra no tendrá más de 16 caracteres.

- 
7. Implemente un programa que le pida al usuario que ingrese dos palabras por teclado e indique si son iguales o cual aparece primero en el diccionario. Puede suponer que las palabras no tendrán más de 16 caracteres cada una.
 8. Implemente un programa que le pida al usuario que ingrese una palabra y un carácter por teclado. A continuación reemplace este carácter en la palabra por asterisco. Finalmente debe indicar la cantidad de veces que reemplazó el carácter. Puede suponer que la palabra no tendrá más de 16 caracteres.



15. Punteros

Forma básica de usar punteros

Con el fin de mantener controlados a los punteros evitando de esta forma violaciones de segmento se recomienda seguir las siguientes recomendación en su uso

- Declarar puntero.
- Inicializar el puntero siempre antes de usarlo. Puede inicializarlo a:
 - A la variable que corresponda
 - A NULL
- Hacer que el tipo de dato coincida con el del tipo de dato del puntero.
- Si la variable a donde inicializa el puntero es un vector inicialicelo al elemento cero.
- Cuando el puntero se inicializó con el elemento cero de un vector utilice el puntero de esta forma `*(p + i)` donde `i` es el índice entero

Estas recomendaciones evitan el uso de punteros modificando el valor al que apuntan, por ejemplo haciendo `p++`;

Ejemplos de uso de punteros

| | Variable int | Vector de int |
|--|------------------------------------|---|
| Declaración de variable | <code>int a;</code> | <code>int v[3];</code> |
| Declaración de puntero | <code>int *p;</code> | <code>int *p;</code> |
| Inicialización de puntero | <code>p = &a;</code> | <code>p = &v[0];</code> |
| Asigno un valor la variable a | <code>a = 10;</code> | <code>v[0] = 1;</code> <code>v[1] = 2;</code> <code>v[2] = 3;</code> |
| Asigno un valor a la variable usando un puntero | <code>*p = 10; // a = 10;</code> | <code>*(p + 0) = 1; // v[0] =1;</code> <code>*(p + 1) = 2; // v[1] =2;</code> <code>*(p + 2) = 3; // v[2] =3;</code> |
| Imprimo el valor de la variable | <code>printf ("%d\r\n", a);</code> | <code>printf ("%d\r\n", v[0]);</code> <code>printf ("%d\r\n", v[1]);</code> <code>printf ("%d\r\n", v[2]);</code> |
| Imprimo el valor de la variable usando el puntero | <code>printf ("%d\r\n", a);</code> | <code>printf ("%d\r\n", *(p + 0));</code> <code>printf ("%d\r\n", *(p + 1));</code> <code>printf ("%d\r\n", *(p + 2));</code> |

Ejemplos

1. Programa en el que se instancia una variable tipo `int` y un puntero del mismo tipo. Se apunta el puntero a esta variable y la inicializa usando el puntero con el valor `0x55`. Luego imprime el valor almacenado en la variable utilizando el puntero.

```
#include <stdio.h>

int main (void)
{
    int a;
    int *p;          //-- Declaro el puntero --

    p = &a;          //-- Inicializo el puntero --

    *p = 0x55; //-- Inicializo la variable a usando el puntero --

    printf ("%d\r\n", *p); //-- Imprimo usando el puntero --

    return (0);
}
```

2. Programa que define un vector de 10 elementos de tipo `char`, luego lo inicializa con los números del '0' al '9' utilizando un puntero. Finalmente imprime dicho vector usando un puntero.

```
#include <stdio.h>

#define CANT ((int)CANT)

int main (void)
{
    char c[CANT];
    char *p;

    p = &c[0];      //-- Inicializo el puntero --

    //-- Inicializo el vector usando el puntero --
    for (i = 0; i < CANT; i++) {
        *(p + i) = '0' + i;
    }

    //-- Imprimo el vector usando el puntero --
    for (i = 0; i < CANT; i++) {
        printf ("%d\r\n", *(p + i));
    }

    return (0);
}
```

3. Se realiza una función que suma dos números enteros que son pasado como parámetro y retorna el resultado usando el return de la función.

```
#include <stdio.h>

int suma (int *a, int *b)
{
    int r;

    r = *a + *b;

    return (r);
}

int main (void)
{
    int x, y, r;

    //-- Ingreso de los dos numeros a sumar --
    printf ("Ingrese numero\r\n");
    scanf ("%d", &x);
    printf ("Ingrese numero\r\n");
    scanf ("%d", &y);

    r = suma (&x, &y);

    printf ("El resultado %d\r\n", r);

    return (0);
}
```

4. Se realiza una función que suma dos números enteros que son pasado como parámetro y retorna el resultado usando un puntero.

```
#include <stdio.h>

void suma (int *a, int *b, int *r)
{
    *r = *a + *b;

    return;
}

int main (void)
{
    int x, y, r;

    //-- Ingreso de los dos numeros a sumar --
    printf ("Ingrese numero\r\n");
    scanf ("%d", &x);
    printf ("Ingrese numero\r\n");
    scanf ("%d", &y);
}
```

```

    suma (&x, &y, &r);

    printf ("El resultado %d\r\n", *r);

    return (0);
}

```

5. Se realiza una función que devuelve un puntero al elemento central de un vector de enteros.

Finalmente se imprime el elemento central del vector usando el puntero obtenido con el

```

#include <stdio.h>

#define CANT ((int)3)

int* medio (int *dataPtr, int dataCant)
{
    int *p;

    p = dataPtr + (dataCant / 2);

    return (p);
}

int main (void)
{
    int v[CANT] = {1, 2, 3};
    int *q;

    q = medio (&v[0], CANT);

    //-- Imprimo el elemento central --
    printf ("El elemento central vale %d\r\n", *q);

    return (0);
}

```

Advertencia:

No olvide inicializar el puntero antes de utilizarlo.

Advertencia:

Evite realizar operaciones de incremento o decremento sobre punteros, por ejemplo: p++ ó p--;

Ejercicios

1. Implemente una función que calcule el promedio de un vector de tipo float. El prototipo de la función es

```
float promedio (float *dataPtr, int dataCant);
```

Donde:

dataPtr: Es el puntero a los datos.

dataCant: Es la cantidad de elementos del vector apuntado.

2. Implemente una función que invierta el contenido de dos variables cuyo prototipo es:

```
void swap (int *a, int *b);
```

Donde: a y b son los punteros a las variables que se les debe invertir el contenido.

3. Implemente una función que verifique si un vector de int está ordenado de manera creciente o decreciente. El prototipo es

```
int orden (int *dataPtr, int dataCant);
```

Donde:

dataPtr: Es el puntero a los datos

dataCant: Es la cantidad de elementos del vector apuntado

Devuelve: 1 si el orden es creciente; 0 si no está ordenado; -1 si el orden es decreciente.

4. Implemente una función que devuelva un puntero al elemento que contiene el valor máximo de un vector.

```
int * myMax (int *dataPtr, int dataCant);
```

Donde:

dataPtr: Es el puntero a los datos

dataCant: Es la cantidad de elementos del vector apuntado

Devuelve: El puntero al elemento que contiene el máximo

5. Implemente una función que devuelva un puntero al elemento que contiene el valor mínimo de un vector.

```
int * myMin (int *dataPtr, int dataCant);
```

Donde:

dataPtr: Es el puntero a los datos

dataCant: Es la cantidad de elementos del vector apuntado

Devuelve: El puntero al elemento que contiene el mínimo


6. Implemente una función que imprima en hexadecimal todos los caracteres de un string. El prototipo de la función es el siguiente.

```
int myHexa (char *dataPtr)
```

Donde:

dataPtr: Es el puntero al string a pasar a hexadecimal.

Devuelve la cantidad de caracteres sin contar el '\0'

- 
7. Implemente una función que se le pase como parámetro un puntero a un string e indique si este contiene solo los dígitos del '0' al '9'. El prototipo de la función es el siguiente.

```
int esNumero (char *dataPtr);
```

Donde:

dataPtr: Es el puntero al string

Devuelve: Cero si algún carácter del string no corresponde a un dígito del '0' al '9', devuelve uno

8. Implemente una función que recibe un puntero a un string que contiene un número y devuelve ese número en un `int`. El prototipo de la función es el siguiente.

```
int convertirA_Int(char *dataPtr)
```

Donde:

dataPtr: Es el puntero al string a pasar a convertir.

Devuelve: El número entero si pudo convertirlo, si el string contiene un carácter distinto a los dígitos del '0' al '9' y menos uno en caso de error.



16. Asignación dinámica de memoria

Funciones utilizadas de stdlib.h

| Función | Descripción |
|---------|--|
| malloc | Asigna dinámicamente memoria. |
| free | Libera memoria asignada dinámicamente. |
| realloc | Asigna dinámicamente memoria. |

Ejemplos

1. Programa en el que se instancia dinámicamente un vector de 10 elementos de tipo int. Estos se inicializan con los números del 0 al 9. Finalmente se imprime este vector en pantalla y se libera la memoria asignada dinámicamente.

```
#include <stdio.h>
#include <stdlib.h>

#define CANT ((int)10)

int main (void)
{
    int *p;
    int i;

    //-- Pido memoria dinamicamente. --
    p = (int *)malloc (sizeof (*p) * CANT);
    if (p == NULL) {
        return (-1);
    }
    //-- Inicializo el vector --
    for (i = 0; i < CANT ; i++) {
        *(p + i) = i;
    }

    //-- Imprimo el vector --
    for (i = 0; i < CANT ; i++) {
        printf ("%d\r\n", *(p + i));
    }

    //-- Libero memoria --
    free (p);

    return (0);
}
```


2. Programa en el que se instancia dinámicamente un vector de 10 elementos de tipo int. Estos se inicializan con números del 0 al 9. Luego utilizando realloc se piden 10 elementos más y se los inicializa con los números del 10 al 19. Finalmente se imprime este vector en pantalla y se libera la memoria asignada dinámicamente.

```
#include <stdio.h>
#include <stdlib.h>
#define CANT ((int) 10)
int main (void)
{
    int *p, *pBack;
    int cantTotal, i;
    int cantUsada;

    //-- Pido memoria para CANT elementos --
    cantTotal = CANT;
    cantUsada = 0;
    p = (int*)malloc (sizeof (*p) * cantTotal);
    if (p == NULL) {
        return (-1);
    }

    //-- Inicializo la zona de memoria asignada con cero--
    for (i = cantUsada; i < cantTotal; i++) {
        *(p + i) = i;
        cantUsada++;
    }

    //-- Pido memoria para CANT elementos mas--
    pBack = (int*)realloc (p, sizeof (*p) * (cantTotal + CANT));
    if (pBack != NULL) {
        //-- Realloc ok --
        //-- Hay cantTotal + CANT elementos --
        cantTotal += CANT;
        p = pBack;
    }

    //-- Inicializo la zona de memoria asignada con uno--
    for (i = cantUsada; i < cantTotal; i++) {
        *(p + i) = i;
        cantUsada++;
    }

    //-- Imprimo el vector utilizado, en este caso es todo --
    for (i = 0; i < cantUsada; i++) {
        printf ("%d\r\n", *(p + i));
    }

    //-- Libero memoria --
    free (p);
    return (0);
}
```

Ejercicios

1. Implemente un programa que utilizando `malloc` reserve memoria para almacenar las letras mayúsculas del alfabeto (salvo la Ñ). Luego imprima la zona reservada y la libere antes de finalizar el programa.
2. Implemente un programa que le pregunte al usuario cuantas letras desea ingresar, reserve utilizando `malloc` la memoria necesaria y luego le pida al usuario caracteres hasta llenar el vector reservado dinámicamente. Finalmente debe imprimir el vector en orden inverso al ingresado y liberar la memoria reservada.
3. Implemente un programa que le pida al usuario que ingrese letras y las almacene en memoria, el fin del ingreso de datos ocurre cuando el usuario ingresa el signo de admiración. Luego se deberán imprimir todas las letras ingresadas por el usuario. (Use `realloc`)
4. Escribe una función que calcule todos los números primos hasta el número indicado. Estos números deben ser almacenados en un vector creado dinámicamente. El prototipo de la función es el siguiente

```
int * calcularPrimos (int n, int *cant);
```

Donde:

n: Es el número hasta el cual tengo que calcular los números primos

cant: Es el puntero a la variable donde se almacena el tamaño del vector creado dinámicamente con los números primos almacenados.

Devuelve: NULL en caso de error o un puntero al vector con los números primos..



17. String

Funciones utilizadas de stdio.h

| Función | Descripción |
|---------|---|
| sprintf | Formatea una cadena de caracteres y la almacena en un string. |
| fgets | Lee una cadena de caracteres desde el teclado. |

Funciones utilizadas de string.h

| Función | Descripción |
|---------|---|
| strcat | Concatenar un string con otro. |
| strchr | Busca un carácter en un string. |
| strcmp | Compara dos strings alfabéticamente. |
| strcpy | Copia un string en otro. |
| strlen | Obtiene el tamaño de un string. |
| strstr | Busca un string en otro string. |
| memcpy | Copia una cantidad de bytes de una zona de memoria en otra. |
| memset | Escribe una zona de memoria con un carácter determinado. |
| memcmp | Compara dos zonas de memoria. |

Ejemplos

1. Implemente una función que cuente la cantidad de caracteres de un string sin contar el '\0'. El prototipo de la función es

```
int myStrLen (char *s);
```

Además implemente un main que verifique automáticamente el funcionamiento básico de la función implementada

```
#include <stdio.h>
#include <stdlib.h>

int myStrLen (char *s)
{
    int c = 0;

    if (s != NULL) {
        while (*(s + c) != '\0') {
            c++;
        }
    }
    return (c);
}
```

```

int main (void)
{
    //-- Vectores de prueba --
    char v0[]="Hola";
    char v1[]="";
    char v2[]="Hola como te va?";
    int r;

    //-- Prueba de las funciones --
    r = myStrLen (v0);      printf ("strlen > %s: %d\r\n", v0, r);
    r = myStrLen (v1);      printf ("strlen > %s: %d\r\n", v1, r);
    r = myStrLen (v2);      printf ("strlen > %s: %d\r\n", v2, r);
    r = myStrLen (NULL);    printf ("strlen > NULL: %d\r\n", r);

    return (0);
}

```

Escriba y guarde el código anterior en un archivo llamado ejemplo01.c

Compilando y ejecutando el programa.

```

jerome@linuxVm:~$ gcc ejemplo01.c -Wall -oejemplo01.out
jerome@linuxVm:~$ ./ejemplo01.out
strlen > Hola: 4
strlen > : 0
strlen > Hola como te va? : 16
strlen > NULL : 0

```

2. Programa que demuestra el uso de sprintf. Se le pide al usuario que ingrese un número entero y se lo incluya en un string con formato.

```

#include <stdio.h>

int main(void)
{
    int n;
    char str[64];

    printf ("Ingrese numero\r\n");
    scanf ("%d", &n);

    sprintf (str, "El numero ingresado es: %d\r\n", n);

    printf ("%s", str);

    return 0;
}

```

3. El siguiente programa demuestra el uso de `fgets` que a diferencia del `scanf` permite ingresar textos con espacios y limita la cantidad de caracteres a ingresar.

```
#include <stdio.h>

int main(void)
{
    char str[64];

    printf ("Ingrese texto\r\n");
    fgets (str, sizeof (str), stdin);

    printf ("%s", str);

    return 0;
}
```

4. El siguiente programa agrega al final de un texto ingresado por teclado la palabra hola.

```
#include <stdio.h>
#include <string.h>

int main(void)
{
    char str[64];

    printf ("Ingrese texto\r\n");
    scanf ("%s", &str[0]);

    strcat (str, "hola");

    printf ("%s\r\n", str);

    return 0;
}
```

5. Este programa demuestra el uso de la función strchr, se utiliza para contar la cantidad de letras 'a' que hay en una palabra ingresada por teclado.

```
#include <stdio.h>
#include <string.h>

int main(void)
{
    char str[64], *p;
    int i, c = 0;
    printf ("Ingrese palabra\r\n");
    scanf ("%s", &str[0]);

    p = &str[0];
    while (p != NULL) {
        p = strchr (p, 'a');
        if (p != NULL) {
            c++;
            p++;
        }
    }
    printf ("Hay %d letras \'a\' en el texto %s\r\n", c, str);
    return 0;
}
```

6. Programa que compara 2 palabras y las imprime en orden alfabético

```
#include <stdio.h>
#include <string.h>

int main(void)
{
    char str0[64], str1[64];
    int r;
    printf ("Ingrese palabra\r\n");
    scanf ("%s", &str0[0]);
    printf ("Ingrese palabra\r\n");
    scanf ("%s", &str1[0]);

    r = strcmp (str0, str1);
    if ( r == 0) {
        printf ("Son iguales: %s, %s\r\n", str0, str1);
    } else {
        if ( r < 0) {
            printf ("%s, %s\r\n", str0, str1);
        } else {
            printf ("%s, %s\r\n", str1, str0);
        }
    }
    return 0;
}
```

7. El siguiente programa demuestra el uso de la función strcpy, copiando un string ingresado por teclado en otro.

```
#include <stdio.h>
#include <string.h>

int main(void)
{
    char str0[64], str1[64];

    printf ("Ingrese palabra\r\n");
    scanf ("%s", &str0[0]);

    strcpy (&str1[0], &str0[0]);

    printf ("%s, %s\r\n", str0, str1);

    return 0;
}
```

8. El siguiente programa demuestra el uso de la función strlen, imprime en pantalla la cantidad de caracteres de una palabra ingresada por teclado.

```
#include <stdio.h>
#include <string.h>

int main(void)
{
    char str[64];
    int r;

    printf ("Ingrese palabra\r\n");
    scanf ("%s", &str[0]);

    r = strlen (str);

    printf ("%s tiene %d caracteres\r\n", str, r);

    return 0;
}
```

9. El siguiente programa indica cuantas veces esta la palabra “hola” en el texto ingresado por teclado

```
#include <stdio.h>
#include <string.h>

int main(void)
{
    char str[64], *p;
    int c = 0;
    printf ("Ingrese texto\r\n");
    fgets (str, sizeof (str), stdin);

    p = &str[0];
    while (p != NULL) {
        p = strstr (p, "hola");
        if (p != NULL) {
            c++;
            p++;
        }
    }

    printf ("Hay %d \"hola\" en el texto: %s\r\n", c, str);

    return 0;
}
```


Ejercicios

En los ejercicios que solicite implementar una función genera además un main que demuestre de forma automática el funcionamiento de la función.

1. Implemente una función que reciba dos strings como parámetros: str y endtStr. La función debe analizar el string str para verificar si termina con el string endtStr. El prototipo de la función es el siguiente

```
int endsWith (char *str, char *endStr);
```

Donde:

- str: es el string que se desea analizar
- endStr: es el string que se busca verificar si esta al final de str

Retorna: 1 si endStr está al final de str y 0 en caso contrario

Ejemplos

| str | endStr | Retorna |
|-------------------|---------|---------|
| "hola como te va" | "como" | 0 |
| "hola como te va" | "adios" | 0 |
| "hola como te va" | "te" | 0 |
| "hola como te va" | "te va" | 1 |
| "hola como te va" | "va" | 1 |

2. Implemente una función que reciba dos strings como parámetros: str y startStr. La función debe analizar el string str para verificar si comienza con el string startStr.. El prototipo de la función es el siguiente

```
int startsWith (char *str, char *startStr);
```

Donde:

- str: es el string que se desea analizar
- startStr: es el string que se busca verificar si está al comienzo de str

Retorna: 1 si startStr está al comienzo de str y 0 en caso contrario

Ejemplos

| str | startStr | Retorna |
|-------------------|----------|---------|
| "hola como te va" | "como" | 0 |
| "hola como te va" | "adios" | 0 |
| "hola como te va" | "va" | 0 |
| "hola como te va" | "hola" | 1 |
| "hola como te va" | "ho" | 1 |

3. Implemente una función que extraiga una porción del string pasado como parámetro. El prototipo de la función es el siguiente

```
int subStr(char *str, int pos, int len, char *strRes);
```

Donde:

- str: es el string que se desea analizar
- pos: Es el índice desde donde se comienza la extracción del string.
- len: Es el largo del string a extraer
- strRes: Es el string extraído

Retorna: 1 si pudo extraer el string, cero en caso contrario

Ejemplos

| str | pos | len | srtRes | Retorna |
|-------------------|-----|-----|--------|---------|
| "hola como te va" | 0 | 4 | "hola" | 1 |
| "hola como te va" | 14 | 2 | "va" | 1 |
| "hola como te va" | 14 | 3 | "va" | 1 |
| "hola como te va" | 0 | 0 | "\0" | 0 |
| "hola como te va" | 16 | 10 | "\0" | 0 |

4. Implemente una función que elimina todas las ocurrencias de un carácter pasado como parámetro de un string.

```
int trim(char *str, char chr, char *strRes);
```

Donde:

- str: es el string que se desea analizar.
- chr: El carácter que se desea eliminar.
- strRes: Es el string resultante al cual se le extrajeron todas las ocurrencias del carácter pasado como parámetro.

Retorna la cantidad de caracteres extraídos o un número negativo en caso de error.

Ejemplos

| str | chr | strRes | Retorna |
|-------------------|-----|-------------------|---------|
| "hola como te va" | ' ' | "hola como te va" | 0 |
| "hola como te va" | ' ' | "holacomoteva" | 3 |
| "hola como te va" | 'a' | "hol como te v" | 2 |
| "hola como te va" | 'o' | "hla cm te va" | 3 |

5. Implemente una función agregue al final del string el carácter pasado como parámetro hasta que la longitud del string sea la deseada.

```
int padRigth(char *str, int len, char chr);
```

Donde:

- str: es el string que se desea rellenar.
- len: Largo final del string
- chr: El carácter que se desea usar de relleno.

Retorna la cantidad de caracteres agregados al string.

Ejemplos

| str | len | chr | str (luego de ejecutar la función) | Retorna |
|--------|-----|-----|------------------------------------|---------|
| "hola" | 4 | 'X' | "hola" | 0 |
| "hola" | 3 | 'X' | "hola" | 0 |
| "hola" | 5 | 'X' | "holaX" | 1 |
| "" | 3 | 'X' | "XXX" | 3 |

6. Implemente una función agregue al comienzo del string el carácter pasado como parámetro hasta que la longitud del string sea la deseada.

```
int padLeft(char *str, int len, char chr);
```

Donde:

- str: es el string que se desea rellenar.
- len: Largo final del string
- chr: El carácter que se desea usar de relleno.

Retorna la cantidad de caracteres agregados al string.

Ejemplos

| str | len | chr | str (luego de ejecutar la función) | Retorna |
|--------|-----|-----|------------------------------------|---------|
| "hola" | 4 | 'X' | "hola" | 0 |
| "hola" | 3 | 'X' | "hola" | 0 |
| "hola" | 5 | 'X' | "Xhola" | 1 |
| "" | 3 | 'X' | "XXX" | 3 |

7. Implemente una función agregue un string en la posición indicada en otro string.

```
int insertStr(char *strDest, int pos, char *strOrg);
```

Donde:

- strDest: es el string que se desea rellenar.
- pos: Es la posición donde se inserta el string.
- strOrg: es el string que se desea rellenar.

Retorna la cantidad de caracteres insertados en el string destino.

Ejemplos

| strDest | pos | strOrg | strDest (Luego de ejecutar la función) | Retorna |
|---------|-----|--------|--|---------|
| "hola" | 0 | "XX" | "XXhola" | 2 |
| "hola" | 4 | "XX" | "holaXX" | 2 |
| "hola" | 2 | "XX" | "hoXXla" | 2 |
| "hola" | 100 | "XX" | "hola" | 0 |
| "hola" | 0 | " " | "hola" | 0 |

8. Implemente una función que busque dentro de un string todas las ocurrencias de otro string pasado como parámetro y las reemplace por otro string. El prototipo de la función es el siguiente:

```
int reemplazaStr (char *str, char *strBuscar,  
                 char *strNueva);
```

Donde:

- str: Es un puntero al string donde se realizan los reemplazos.
- strBuscar: Es un puntero al string a buscar.
- strNueva: Es un puntero al string a reemplazar.

Retorna un número positivo indicando la cantidad de string reemplazados.



18. Algoritmos integradores

Ejemplos

1. Programa en el que se instancia dinámicamente un vector de 10 elementos de tipo `int`. Estos se inicializan con los números del 0 al 9. Se utiliza el algoritmo del burbujeo para ordenar el vector de mayor a menor. Se imprime el vector antes y después de ordenar. Finalmente se libera memoria.

```
#include <stdio.h>
#include <stdlib.h>
#define CANT ((int) 10)
int main (void)
{
    int *p;
    int i, j;
    int aux;

    //-- Pido memoria para CANT elementos --
    p = (int*)malloc (sizeof (*p) * CANT);
    if (p == NULL) {
        printf ("Error\r\n");
        return (-1);
    }

    //-- Inicializo la zona de memoria asignada con cero --
    for (i = 0; i < CANT; i++) {
        *(p + i) = i;
    }

    //-- Imprimo el vector --
    printf ("Antes de ordenar\r\n");
    for (i = 0; i < CANT; i++) {
        printf ("%d\r\n", *(p + i));
    }

    //-- Ordeno --
    for (i = 0; i < CANT - 1; i++) {
        for (j = i + 1; j < CANT; j++) {
            if (*(p + i) < *(p + j)) {
                aux = *(p + i);
                *(p + i) = *(p + j);
                *(p + j) = aux;
            }
        }
    }

    //-- Imprimo el vector --
    printf ("\r\nDespues de ordenar\r\n");
    for (i = 0; i < CANT; i++) {
        printf ("%d\r\n", *(p + i));
    }
}
```

```
//-- Libero memoria --
free (p);
return (0);
}
```

Ejercicios

En los ejercicios que solicite implementar una función genera además un main que demuestre de forma automática el funcionamiento de la función.

1. Implemente una función que reciba un vector con letras y lo ordene alfabéticamente. El prototipo de la función es el siguiente

```
void ordenarChar (char *dataPtr, int dataCant);
```

Donde

- dataPtr: Es el puntero al vector a ordenar
- dataCant: La cantidad de elementos del vector a ordenar

2. Implemente una función que me indique si el string ingresado contiene solo letras o solo números. El prototipo de la función es el siguiente:

```
int validaString (char *dataPtr);
```

Devuelve

- 1 si el string contiene solo letras .
- 2 si el string contiene sólo números.
- 0 Si no es ninguna de las anteriores.

3. Implemente una función que cuente la ocurrencia de cada carácter (histograma) de un string pasado como parámetro.

```
void contarCaracteres (char *dataPtr, int *dataCntPtr);
```

Donde:

dataPtr: Es el puntero al string en el que hay que contar la ocurrencia de cada carácter.

dataCntPtr: Es el puntero un vector de 256 enteros en el que se lleva la cuenta de los caracteres del string

Ejemplo:

El string apuntado por dataPtr es "11AB1B1ZZZZ1"

- La posición 65 (65 es el ASCII de la 'A') del vector apuntado por dataCntPtr debe tener el número 1 (Cantidad de 'A' en el string)
- La posición 66 (66 es el ASCII de la 'B') del vector apuntado por dataCntPtr debe tener el número 2 (Cantidad de 'B' en el string)
- La posición 90 (90 es el ASCII de la 'Z') del vector apuntado por dataCntPtr debe tener el número 4 (Cantidad de 'Z' en el string)
- La posición 49 (49 es el ASCII de la '1') del vector apuntado por dataCntPtr debe tener el número 5 (Cantidad de '1' en el string)
- El resto de los elementos del vector deberán estar en cero.

4. Implemente una función que analice un párrafo de texto y devuelva la cantidad de caracteres y palabras que contiene. El prototipo de la función es el siguiente:

```
int analizaString (char *dataPtr, int *palabrasCant,  
                  int *caracteresCant);
```

Donde:

- dataPtr es el puntero al string a analizar
- palabrasCant es un puntero a una variable donde se almacenará la cantidad de palabras del texto
- caracteresCant es un puntero a una variable donde se almacenará la cantidad de caracteres del texto

Devuelve

- 0 si el string contiene solo letras y signos de puntuación.
- -1 en caso contrario.

Notas de implementación:

- Los caracteres a contar son todas las letras del alfabeto, números, signos de puntuación y espacios.
- Una palabra se separa de otra por un espacio o signo de puntuación.
- No cuente espacios para obtener la cantidad de palabras, ya que el texto puede contener dos espacios seguidos y esto dará un conteo incorrecto.

5. Implemente una función que determine si una palabra pasada como parámetro es palindromo

Prototipo de la función

```
int detectorPalindromo(char *palabraPtr)
```

Donde:

- palabraPtr: es el puntero a la palabra a analizar

Devuelve:

- 0 si la palabra es palindromo
- -1 si la palabra no es palindromo
- -2 si la palabra tiene menos de 2 caracteres

6. Implemente una función que valide números de tarjeta de crédito utilizando el algoritmo de Luhn. El prototipo de la función es el siguiente:

```
int luhnAlg (char *tarjeta);
```

Parámetros

- tarjeta: puntero al vector que contiene el número de la tarjeta de crédito terminado en '\0'

Devuelve:

- Cero o un número positivo indicando que la tarjeta es válida.
- -1: Cuando la cantidad de dígitos de la tarjeta es distinto que 16.
- -2: Indica que el número de tarjeta es inválido.

Algoritmo de Luhn

- a. Multiplique los dígitos que se encuentran en la posición par del vector por dos, si el resultado es mayor o igual que diez se suman cada uno de los dígitos.
- b. Multiplique los dígitos que se encuentran en la posición impar del vector por uno.
- c. Suma todos los resultados obtenidos en los puntos a y b, obteniendo la suma llamada S. Si el módulo 10 de la suma obtenida S es igual a cero, el número de tarjeta es válido.

Ejemplo:

| Dígitos Tarjeta | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0 | 1 | 2 | 3 | 4 | 7 | |
|--------------------------------|----|----|----|----|----|----|-----|----|-----|----|----|----|----|----|----|----|------|
| Valor a multiplicar Por dígito | X2 | X1 | X2 | X1 | X2 | X1 | X2 | X1 | X2 | X1 | X2 | X1 | X2 | X1 | X2 | X1 | |
| Resultado de la multiplicación | 0 | 1 | 4 | 3 | 8 | 5 | 12 | 7 | 16 | 9 | 0 | 1 | 4 | 3 | 8 | 7 | |
| Dígitos a sumar | 0 | 1 | 4 | 3 | 8 | 5 | 1+2 | 7 | 1+6 | 9 | 0 | 1 | 4 | 3 | 8 | 7 | = 70 |

$70 \% 10 = 0 \Rightarrow$ La tarjeta es válida

7. Implemente una función que convierta un número positivo hexadecimal de 4 dígitos almacenado en un string y devuelva el correspondiente número decimal. Las letras del número hexadecimal están en mayúscula.

El prototipo de la función es:

```
int hexaToDec (char *dataPtr);
```

Devuelve:

- El valor hexadecimal en decimal
- -1 Si hay un símbolo que no corresponda a un número hexadecimal.
- -2 Si la cantidad de dígitos es distinta a 4

Ejemplos

```
hexaToDec ("0001"); // Devuelve 1
hexaToDec ("CAFE"); // Devuelve 51966
hexaToDec ("JJJJ"); // Devuelve -1
hexaToDec ("1");    // Devuelve -2
```

8. Implemente una función que convierta en binario un número entero positivo pasado como parámetro. El prototipo de la función es

```
char* imprimirBinario (int n);
```

Donde:

n número a imprimir en binario.

Devuelve NULL si el número es menor que cero, en caso contrario devuelve el puntero a un string con el número binario.

9. Implemente una función que calcule el promedio de las notas pasadas por un string separadas por espacios. La última nota tiene un espacio también. El prototipo de la función es el siguiente

```
float calcularPromedio (char *dataPtr)
```

Si el string pasado como parámetro está mal formado devuelva NAN

Ejemplos:

```
calcularPromedio("Hola");           // Devuelve NAN
calcularPromedio("");               // Devuelve NAN
calcularPromedio("1 2 3 12");       // Devuelve NAN
calcularPromedio("0 1 2 3 4 5 6 7 8 9 "); // Devuelve 4.5
```

Notas de implementación:

- Esta función solamente calcula el promedio de números de un dígito separados por espacio.

10. Implemente una función que determine si un password cumple con todas las recomendaciones de seguridad que se describen a continuación:
- Debe tener al menos 8 caracteres.
 - No debe tener espacios.
 - Debe contener letras mayúsculas y minúsculas.
 - Debe contener alguno de estos símbolos ^!@#\$%^&* _+= ' | \ \ () {} \ [] ; : " ' < > , . ? /
 - Debe contener un dígito base 10. (0 - 9)

El prototipo de la función es el siguiente

```
int validarPassword(char *dataPtr)
```

Donde:

- dataPtr: Es el puntero al password a validar.
- Devuelve:
 - 1: El password cumple las cuatro recomendaciones de seguridad.
 - 1: Si no cumple la recomendación a
 - 2: Si no cumple la recomendación b
 - 3: Si no cumple la recomendación c
 - 4: Si no cumple la recomendación d
 - 5: Si no cumple la recomendación e

11. Implemente una función que obtenga el dígito verificador de un número de CUIT pasado como parámetro, el cálculo se realiza utilizando el algoritmo módulo 11. El prototipo de la función es el siguiente:

```
int cuitValida (char *cuit);
```

Parámetros

- cuit: puntero al vector que contiene el número de CUIT terminado en '\0'

Devuelve:

- Un número positivo indicando el dígito verificador.
- 1: Cuando la cantidad de dígitos es distinto de 10
- 2: Indica que el número de CUIT es inválido (contiene algo distinto a números)

Algoritmo módulo 11

- Multiplique los dígitos desde el menos significativo por la serie 2,3,4,5,6,7.
- Sume el resultado de las multiplicaciones anteriores.
- Calcule el módulo 11 de la suma anterior.
- Calcule 11 menos el resultado anterior, si el resultado es menor que 10 lo obtenido es el dígito verificador. En cambio si vale 10 el dígito verificador es 9. Si vale 11 el dígito verificador es 0

Ejemplo:

| CUIT | 2 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | Suma | %11 | Dígito |
|--------------------------------|----|----|----|----|----|----|----|----|----|----|------|--------|--------|
| Valor a multiplicar Por dígito | X5 | X4 | X3 | X2 | X7 | X6 | X5 | X4 | X3 | X2 | | 148%11 | 11-5 |
| Resultado de la multiplicación | 10 | 0 | 3 | 4 | 21 | 24 | 25 | 24 | 21 | 16 | =148 | =5 | 6 |