

OBJETIVOS

- Utilizar punteros para manejar vectores, arreglos bidimensionales.
- Aplicar aritmética de punteros.
- Manejar diferentes branch en git
- Utilizar estructuras como tipo de datos.

Ejercicios:

- 1) Copie el siguiente enlace en su navegador: <https://tinyurl.com/tl1-2024-tp2> para crear el repositorio donde subirá el **Trabajo Práctico Nro. 2**. Realice los pasos ya aprendidos para clonar el repositorio en su máquina y poder comenzar a trabajar de forma *local*.

Nota. No se olvide de incluir el archivo `.gitignore` en la raíz del repositorio para excluir los archivos `.exe`, `.obj` y `.tds` del mismo.

Tip: Puede utilizar el sitio gitignore.io para generar el contenido del archivo `.gitignore` que excluye todos los archivos pertinentes.

Recordatorio en caso de estar utilizando una PC del laboratorio: no olvide eliminar las credenciales de windows antes de clonar el repositorio. Asimismo, verifique que su nombre de usuario y email de git sean correctos y el proxy esté configurado. Para cambiar el nombre de usuario y contraseña en git bash escriba los siguientes comandos (incluir las comillas).

```
$ git config --global user.name "Su Nombre"
```

```
$ git config --global user.email "su e-mail"
```

En caso de ser necesario, configure a su vez el proxy, utilizando el comando:

```
$ git config --global http.proxy 10.10.0.31:80
```

Para confirmar que su configuración es la correcta, ejecute la siguiente instrucción:

```
$ git config --list
```

- 2) En el siguiente código se accede a los elementos de un vector.

```
// codigo a completar
#define N 20

int i;
double vt[N];
for(i = 0; i < N; i++)
{
    vt[i] = 1 + rand() % 100;
    printf("%f  ", vt[i]);
}
```

- a) Complete el código anterior para que el mismo funcione en un archivo nuevo que se llame `tp2_1_1.c` y agregue el archivo a su repositorio local (commit) y luego al repositorio remoto (push).

- b) Cree un nuevo Branch de forma local llamado `Opcion_2` para ello utilice los comandos:

- `git branch Opcion_2` → crea un nuevo branch

Taller de Lenguajes I

2024 - Programador Universitario / Licenciatura en informática / Ingeniería en Informática
Trabajo Práctico N° 2

- `git checkout Opcion_2` → Pone el branch [Opcion_2] como directorio de trabajo

Tip: El comando `git checkout -b Opcion_2` realiza los dos comandos anteriores simultáneamente (crea un branch nuevo llamado `Opcion_2` y nos lleva a esa rama)

- c) Dentro del Branch `Opcion_2` cree un nuevo archivo que se llame `tp2_1_2.c`. Para asegurarse que está trabajando en el branch correspondiente ejecute el siguiente comando

- `git status`

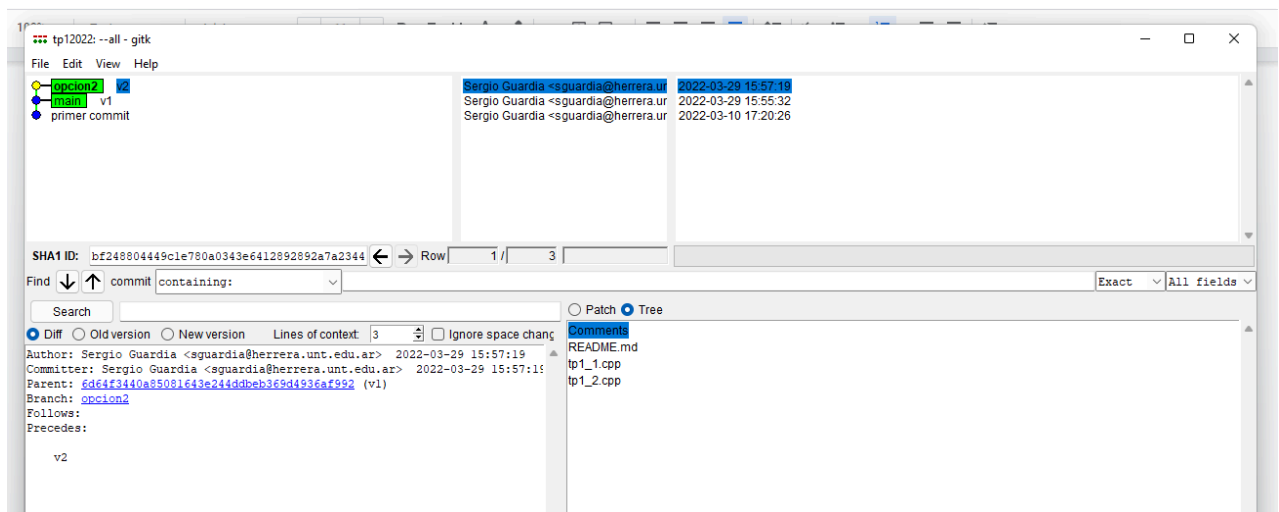
- d) En el archivo `tp2_1_2.c` modifique el código anterior para utilizar **aritmética de punteros (notación indexada)** para recorrer el vector.

Committee los cambios al repositorio local.

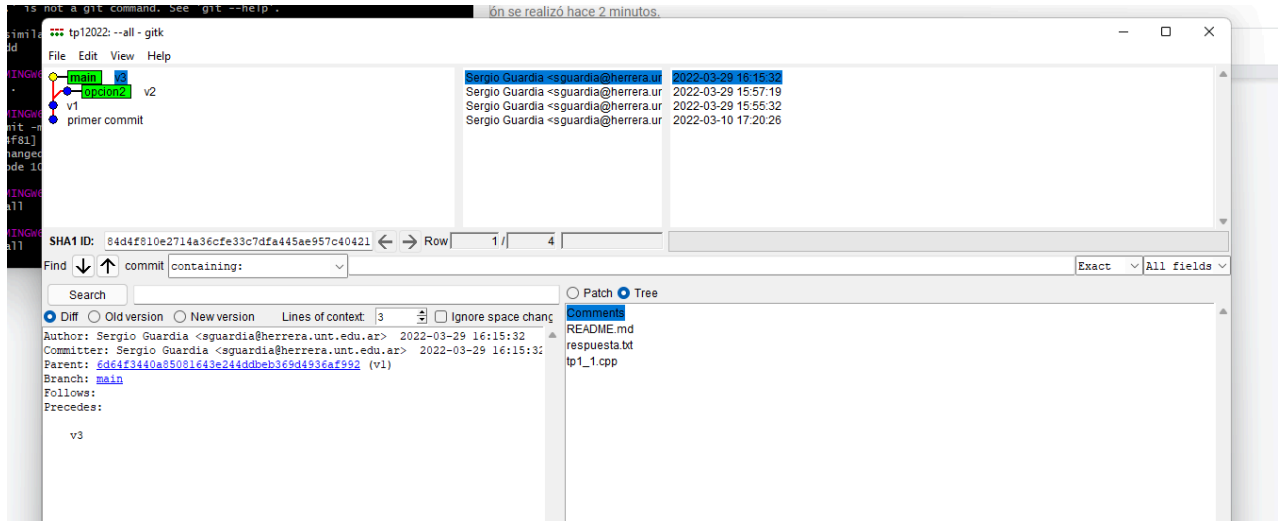
- e) Salte a la línea *main* utilizando el comando `git checkout main`

- f) Inspeccione desde `git bash` o la terminal de Windows (PowerShell) la carpeta donde inicializó el repositorio:

- ¿Puede ver el archivo `tp2_1_2.c`? ¿Por qué?
- Utilice el comando `gitk --all` para ver una representación gráfica del historial de versiones del repositorio.



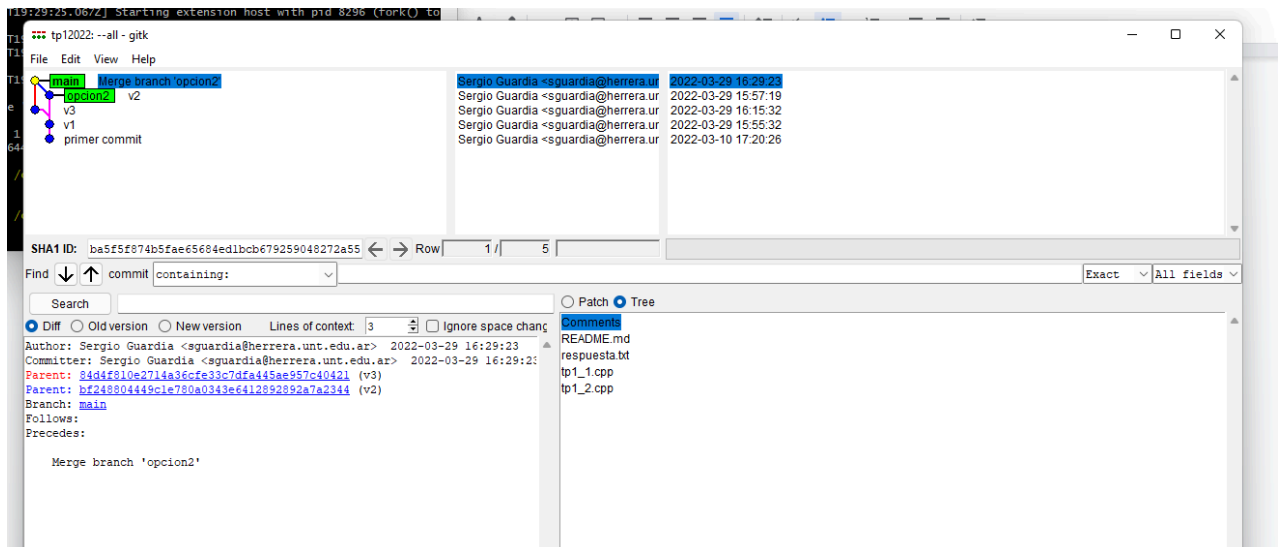
- En la línea principal agregue un nuevo archivo que se llame `Respuestas.txt` y realice un commit para agregar este archivo al repositorio.
- vuelva a ejecutar el comando `gitk --all` ¿Qué diferencias nota?



- En el Branch *main* se va a combinar (*merge*) ambos repositorios. Para esto, utilice el siguiente comando:

```
o git merge Opcion_2 -m "merge opcion 2" → combinamos master con Opcion_2
```

- Luego utilice el comando **gitk -all** para ver los cambios, ¿Qué nota?



- En el archivo respuestas.txt escriba las respuestas a las preguntas anteriores.
- Realice el push para llevar sus cambios al repositorio Remoto

Por último, haga un checkout para saltar nuevamente a la línea principal (Master o Main).

- 3) Reimplemente el siguiente código utilizando aritmética de punteros (tp2_3.c) y súbalo al repositorio.

```
#define N 5
#define M 7
Int i,j;
int mt[N][M];

...
for(i = 0;i<N; i++)
{
    for(j = 0;j<M; j++)
    {
        mt[i][j]=1+rand()%100;
        printf("%lf    ", mt[i][j]);
    }
    printf("\n");
}
```

- 4) Declara un **tipo de dato estructura**:

- Para representar a una PC; los campos serán: velocidad de procesamiento en GHz, año de fabricación, tipo de procesador, cantidad de núcleos.
- Considera valores enteros aleatorios para la velocidad: entre 1 y 3, para el año: entre 2015 y 2024, para la cantidad de núcleos: entre 1 y 8.
- Para evitar ingresar por teclado los tipos de procesador, considera que estos se encuentran en un arreglo de cadenas de caracteres:

```
char tipos[6][10]={“Intel”, “AMD”, “Celeron”, “Athlon”, “Core”, “Pentium”}
```

La **estructura** será la siguiente:

```
struct compu {
    int velocidad;//entre 1 y 3 GHz
    int anio;//entre 2015 y 2023
    int cantidad;//entre 1 y 8
    char *tipo_cpu; //valores del arreglo tipos
};
```

- Define una variable del tipo arreglo de estructura para guardar las características de 5 PC que cargará el usuario.
- Escribe una función que presente la lista de las PC, cada una con sus características.
- Escribe una función que presente la PC más vieja.
- Escribe una función que presente la PC que tiene mayor velocidad.