Taller de Lenguajes I
2025 - Programador Universitario / Licenciatura en informática / Ingeniería en Informática
Trabajo Práctico N° 2

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-tp2-2025** 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 --global --list
```

   Para borrar alguna línea del archivo de configuración

```
$ git config --global --unset-all <registro a borrar>
Ej.
$ git config --global --unset-all user.name → borra todas las instancias
donde figure "user.name"
```

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).

Ejecute los siguientes comandos

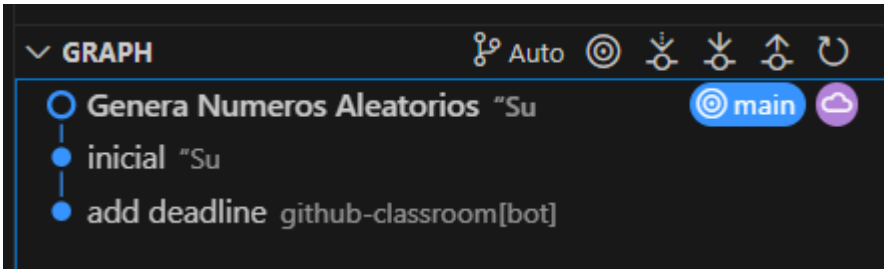
- git add .
- git commit -m 'inicial'
- git push

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

- `git branch Opcion_2` → **crea un nuevo branch**
- `git checkout Opcion_2` → **Pone el branch [Opcion_2]** como directorio de trabajo

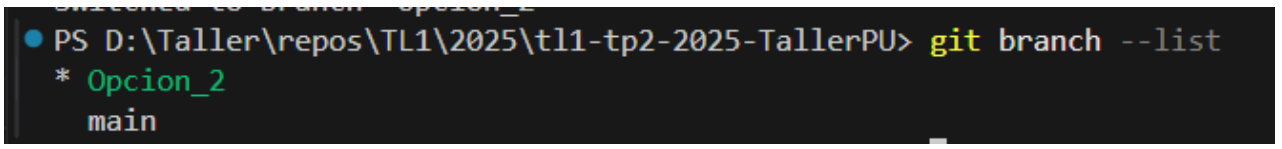
Tip: El comando `git checkout -b Opcion_2` realiza los dos comandos anteriores simultáneamente (crea el branch nuevo llamado Opcion_2 y nos lleva a esa rama)

Antes de crear el branch

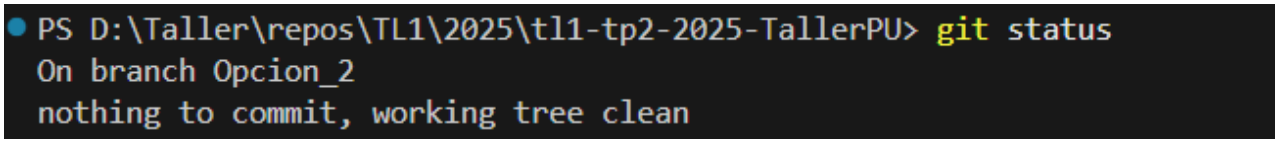


c) Dentro del Branch Opcion_2 cree un nuevo archivo que se llame tp2_1_2.c. Para asegurarse que está trabajando en el branch correspondiente ejecute el siguiente comando

- `git branch --list` → le indicará en qué rama está parado



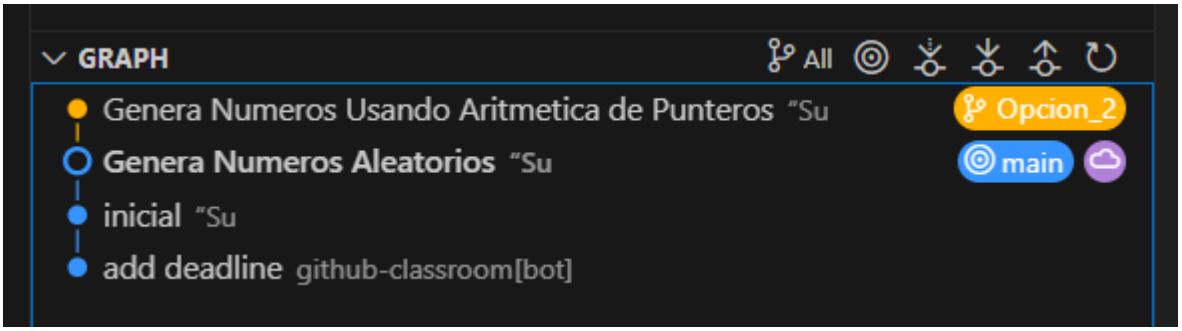
También podría usar el comando git status para verificar que este en la rama correcta



d) En el archivo tp2_1_2.c copie código del archivo tp2_1_1.c y luego haga las modificaciones necesarias para utilizar **aritmética de punteros (notación indexada) para recorrer el vector.**

Commitee los cambios al repositorio local.

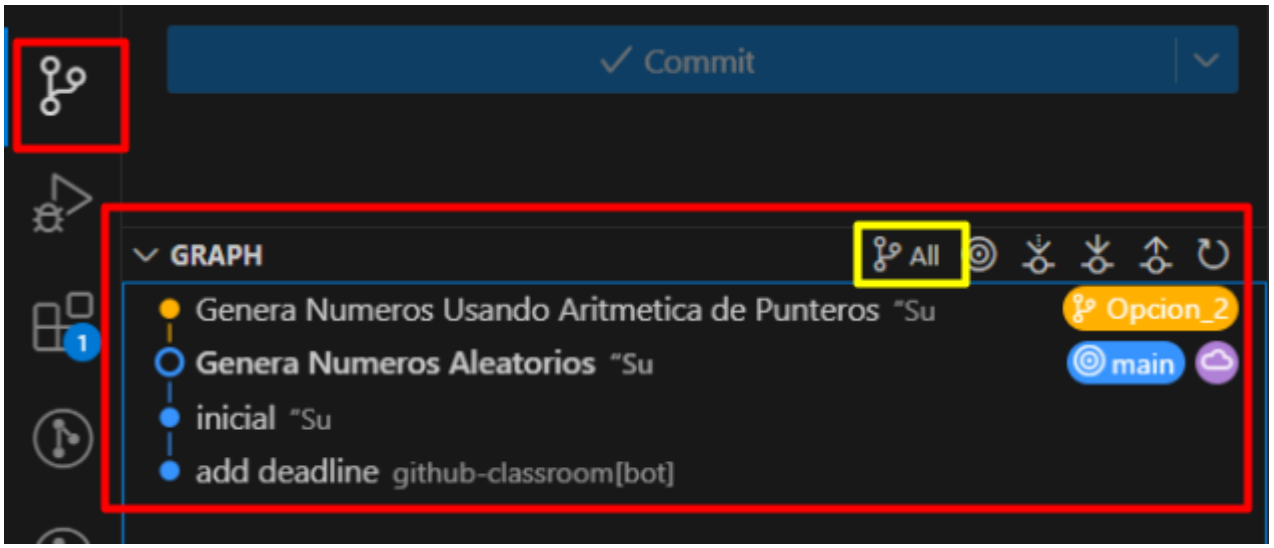
Después de hacer el commit en el Branch Opcion_2



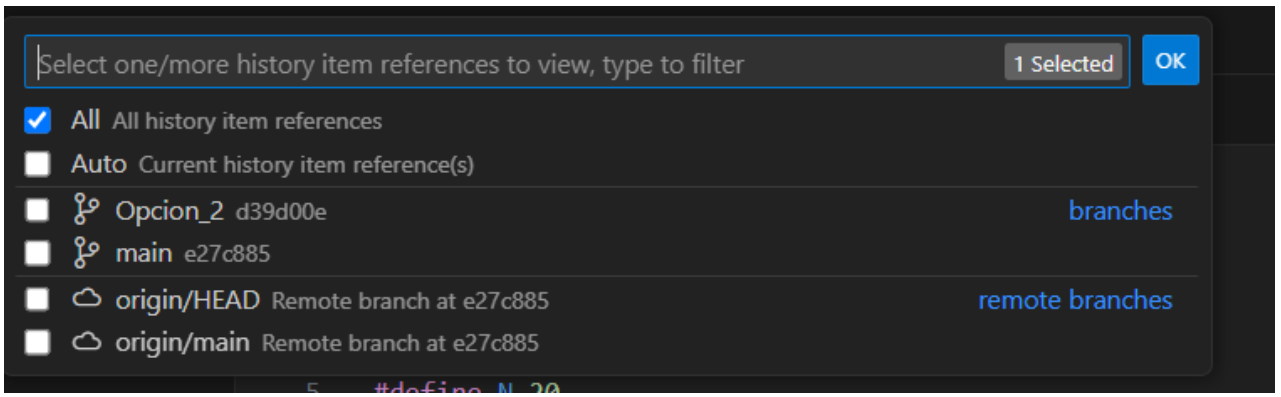
e) Salte a la línea *main* utilizando el comando `git checkout main`

f) Inspeccione la carpeta donde inicializó el repositorio:

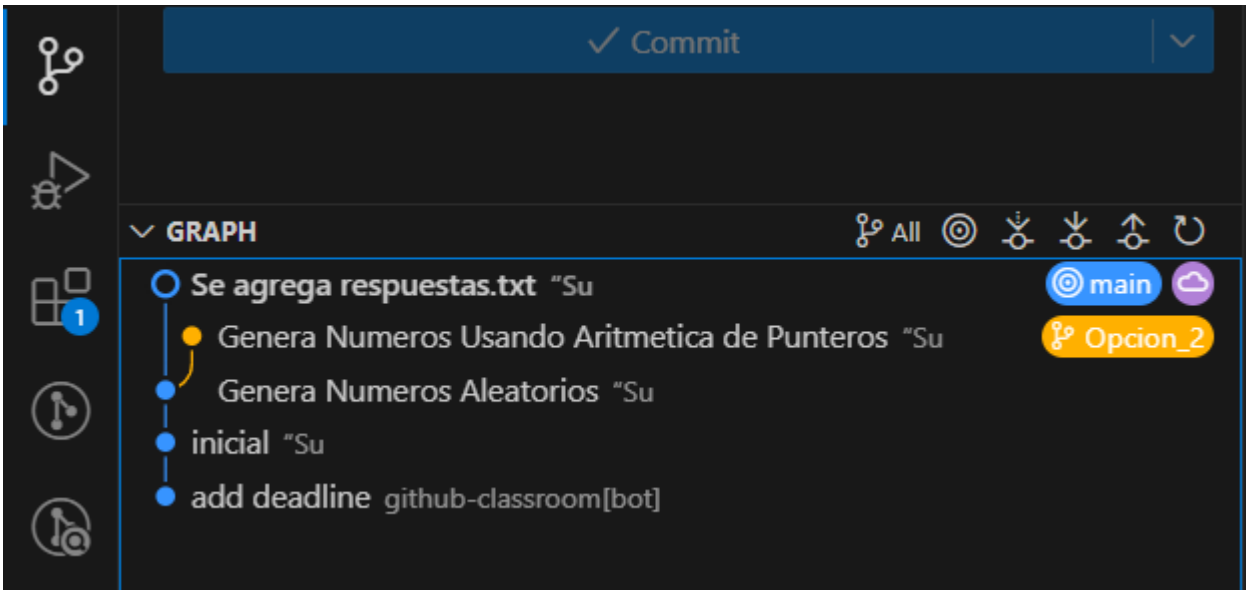
- ¿Puede ver el archivo tp2_1_2.c? ¿Por qué?
- En el VSCode vaya al “Source Control” para ver una representación gráfica del historial de versiones del repositorio.



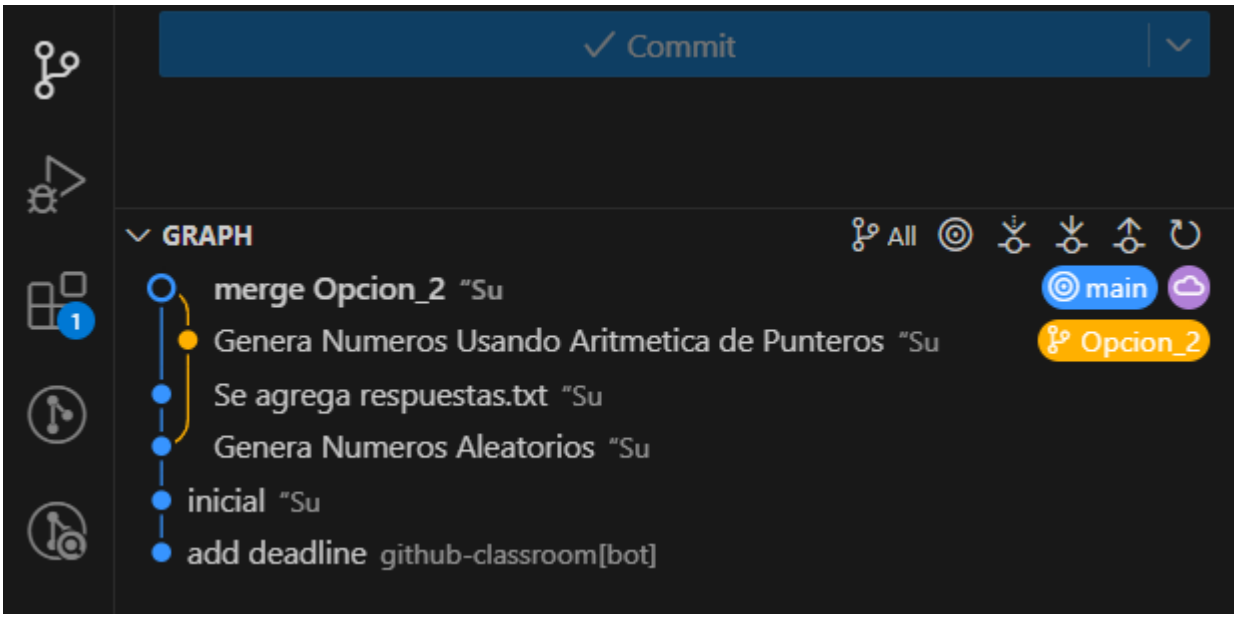
Es importante que marque All (recuadro amarillo) para que vea todos los branch disponibles



- En la rama **main** agregue un nuevo archivo que se llame Respuestas.txt y realice un commit para agregar este archivo al repositorio.
- En VSCode vuelva al “Source Control” ¿Qué diferencias nota?



- En el Branch *main* se va a combinar (*merge)* ambos repositorios. Para esto, utilice el siguiente comando:
    - `git merge Opcion_2 -m "merge Opcion_2"` → combinamos main con Opcion_2
- En VSCode vuelva al "Source Control" vea 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

Nota:

Comandos útiles relacionados a ramas

Para Borrar una rama

- git branch -d nombre-rama

Para Renombrar una rama

- git branch -m nuevo-nombre-rama

Para Listar todas las ramas locales

- git branch --list

3) Modifique 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) Desarrollar una aplicación en C que permita gestionar información sobre computadoras (PC) utilizando estructuras y funciones. La aplicación deberá generar datos aleatorios para un conjunto de PCs y luego implementar funciones para mostrar la información y encontrar características específicas.

a. Definición de la Estructura de Datos

Deberás declarar un tipo de dato `struct` para representar una PC. La estructura se llamará `compu` y contendrá los siguientes campos:

```
struct compu {
   int velocidad;      // Velocidad de procesamiento en GHz (valor entre 1 y 3)
   int anio;           // Año de fabricación (valor entre 2015 y 2024)
   int cantidad_nucleos; // Cantidad de núcleos (valor entre 1 y 8)
   char *tipo_cpu;     // Tipo de procesador (apuntador a cadena de caracteres)
};
```

b. Generación de Datos Aleatorios

La aplicación no requerirá ingreso de datos por teclado. En su lugar, generará aleatoriamente las características para 5 PCs.

- Valores Numéricos Aleatorios:
  - Velocidad: Generar un valor entero aleatorio entre 1 y 3 (inclusive).
  - Año: Generar un valor entero aleatorio entre 2015 y 2024 (inclusive).
  - Cantidad de Núcleos: Generar un valor entero aleatorio entre 1 y 8 (inclusive).
- Tipos de Procesador:
  - Utiliza el siguiente arreglo predefinido de cadenas de caracteres para los tipos de CPU:
    `char tipos[6][10] = {"Intel", "AMD", "Celeron", "Athlon", "Core", "Pentium"};`
  - Para cada PC generada, el campo `tipo_cpu` de la estructura deberá apuntar a una cadena seleccionada aleatoriamente de este arreglo `tipos`.

c. Almacenamiento de Datos
   - Define una variable que sea un arreglo de 5 elementos del tipo `struct compu`. Este arreglo almacenará las características de las 5 PCs generadas.

d. Funciones a Implementar

   Deberás escribir e implementar las siguientes funciones:

   - `void listarPCs(struct compu pcs[], int cantidad)`:
     - Recibe el arreglo de PCs y la cantidad de elementos.
     - Muestra por pantalla la lista completa de las PCs, presentando todas las características de cada una de forma clara.
   - `void mostrarMasVieja(struct compu pcs[], int cantidad)`:
     - Recibe el arreglo de PCs y la cantidad de elementos.
     - Busca la PC con el menor año de fabricación (la más vieja).
     - Muestra por pantalla las características de la PC más vieja encontrada. Si hay varias con el mismo año más antiguo, puedes mostrar la primera que encuentres.
   - `void mostrarMasVeloz(struct compu pcs[], int cantidad)`:
     - Recibe el arreglo de PCs y la cantidad de elementos.
     - Busca la PC con la mayor velocidad de procesamiento.
     - Muestra por pantalla las características de la PC más rápida encontrada. Si hay varias con la misma velocidad máxima, puedes mostrar la primera que encuentres.

El código fuente completo de la aplicación debe guardarse en un archivo llamado `tp2_4.c`.

No se olvide de ir haciendo commit a medida que vaya avanzando. Por ej. cada vez que defina una función.