Gestor de Procesos

```
// --------------------
// GESTOR DE PROCESOS (LISTA ENLAZADA)
// --------------------
struct Proceso
{
    int id;
    string nombre;
    int prioridad;
    Proceso* siguiente;
};
Proceso* listaProcesos = NULL;
```

Declara una estructura (tipo de dato personalizado) para representar un proceso

Es el identificador único del proceso

Guarda el nombre del proceso

Representa la importancia del proceso

Es un puntero que apunta al siguiente nodo (proceso) en la lista enlazada.

Declara un puntero que apunta al primer proceso de la lista.
Al principio no hay procesos, por eso se inicializa en NULL

Insertar un Proceso

```
182  void insertarProceso()
183  {
184      int id, prioridad;
185      string nombre;
```

Declara variables locales para capturar los datos que el usuario ingresará.

Estas líneas piden al usuario los datos del proceso.
Se guardan en las variables id, nombre y prioridad

```
cout << "Ingrese ID del proceso: "; cin >> id;
cout << "Ingrese nombre del proceso: "; cin >> nombre;
cout << "Ingrese prioridad del proceso: "; cin >> prioridad;
```

Crea un nuevo nodo en memoria.
Este nuevo proceso se enlaza al proceso que estaba antes como primero (listaProcesos)

```
    Proceso* nuevo = new Proceso{ id, nombre, prioridad, listaProcesos };
    listaProcesos = nuevo;
    cout << "Proceso insertado." << '\n';
}
```

Actualiza el inicio de la lista.
Ahora el nuevo proceso es el primero (la cabeza de la lista)

Buscar un Proceso por ID

```
194 void buscarProceso() {
195     int idBuscado;
196     cout << "Ingrese ID del proceso a buscar: "; cin >> idBuscado;
```

Variable para guardar el ID que se va a buscar.

Se pide al usuario que ingrese el ID del proceso que quiere encontrar

```
197     Proceso* p = listaProcesos;
198     while (p != NULL) {
199         if (p->id == idBuscado) {
200             cout << "Proceso encontrado: " << p->nombre
201                  << " | Prioridad: " << p->prioridad << '\n';
202             return;
203         }
204         p = p->siguiente;
205     }
```

Se declara un puntero auxiliar p para recorrer la lista, empezando desde el primer nodo.

Recorre la lista hasta llegar al final (NULL).

Compara el ID del proceso actual con el que busca el usuario.

```
200             cout << "Proceso encontrado: " << p->nombre
201                  << " | Prioridad: " << p->prioridad << '\n';
202             return;
203         }
204         p = p->siguiente;
205     }
```

Si lo encuentra, muestra el nombre y la prioridad. Luego termina la función con return

Si no lo encuentra en la lista, muestra un mensaje indicando que no se encontró el proceso.

Eliminar un Proceso

Se pide el ID del proceso a eliminar.

```
209 void eliminarProceso() {
210     int idEliminar;
211     cout << "Ingrese ID del proceso a eliminar: "; cin >> idEliminar;
```

```
212     Proceso* actual = listaProcesos;
213     Proceso* anterior = NULL;
214     while (actual != NULL) {
215         if (actual->id == idEliminar) {
216             if (anterior == NULL)
217                 listaProcesos = actual->siguiente;
218             else
219                 anterior->siguiente = actual->siguiente;
220             delete actual;
221             cout << "Proceso eliminado." << '\n';
222             return;
223         }
```

Se crean dos punteros:

- actual: el proceso que estamos revisando ahora.
- anterior: el proceso anterior al actual (útil si no es el primero).

Recorremos la lista.

Si es el primer proceso, la lista ahora empieza desde el siguiente.

Si es un proceso intermedio o final, se enlaza el proceso anterior con el siguiente, quitando el actual.

Se libera la memoria del proceso y se confirma la eliminación

```
        anterior = actual;
        actual = actual->siguiente;
    }
    cout << "Proceso no encontrado." << '\n';
}
```

Si no se encuentra el ID, se muestra el mensaje correspondiente.

Modificar Prioridad

```
230 void modificarPrioridad() {
231     int idMod;
232     cout << "Ingrese ID del proceso a modificar: "; cin >> idMod;
```

Se pide el ID del proceso cuya prioridad se quiere cambiar.

```
233     Proceso* p = listaProcesos;
234     while (p != NULL) {
235         if (p->id == idMod) {
236             cout << "Prioridad actual: " << p->prioridad << '\n';
237             int nueva;
238             cout << "Ingrese nueva prioridad: "; cin >> nueva;
239             p->prioridad = nueva;
240             cout << "Prioridad modificada." << '\n';
241             return;
242         }
```

Se declara un puntero para recorrer la lista.

Se recorre la lista y se compara el ID

Si encuentra el proceso:

- Muestra su prioridad actual.
- Pide una nueva.
- La asigna y confirma el cambio.

```
243         p = p->siguiente;
244     }
245     cout << "Proceso no encontrado." << '\n';
246 }
```

Si no encuentra el proceso, muestra el mensaje correspondiente.