



SISTEMA DE GESTIÓN DE PROCESOS

INTEGRANTES:

-JEREMY ALDAIR BANEGAS ALANGUIA

-JHON HERMES VALDIVIA TEJADA

-JHON BRANDER CCOPA QQUENTA

-DARGEN LEO SULLASI TECSI

• **-LEOVIGILDO DOMINGO CAYA UMIYAURI**

¿QUÉ ES ESTE PROGRAMA?

El sistema busca simular el comportamiento de un sistema operativo al gestionar procesos. Creamos un programa que permita registrar procesos, almacenarlos en estructuras de datos dinámicas, y simular operaciones como el manejo de memoria (pila) y planificación de procesos (cola por prioridad). Esto permite comprender de forma didáctica cómo se administran los procesos en un entorno multitarea.

FUNCIONES DE SISTEMA

```
men Menu 1 men
1. Lista de Tareas
2. Pila de Tareas
3. Cola de Tareas
4. Salir
Seleccione una opcion si estimado: 1

--- menu 2 lista ---
1. Agregar tarea
2. Eliminar tarea
3. Verificar y mostrar tareas si estimado
4. Buscar tarea
5. Modificar tarea
6. Volver al menu principal
Seleccione una opcion: 1
Ingresar tarea: prueba de validacion de listas
Tarea agregada

--- menu 2 lista ---
1. Agregar tarea
2. Eliminar tarea
3. Verificar y mostrar tareas si estimado
4. Buscar tarea
5. Modificar tarea
6. Volver al menu principal
Seleccione una opcion: 3
Tareas en lista:
1. prueba de validacion de listas

--- menu 2 lista ---
1. Agregar tarea
2. Eliminar tarea
3. Verificar y mostrar tareas si estimado
4. Buscar tarea
5. Modificar tarea
6. Volver al menu principal
Seleccione una opcion: 4
Ingresar la tarea a buscar (exacta porfavor): prueba de validacion de listas
Tarea encontrada en la posicion 1: prueba de validacion de listas

--- menu 2 lista ---
1. Agregar tarea
2. Eliminar tarea
3. Verificar y mostrar tareas si estimado
4. Buscar tarea
5. Modificar tarea
6. Volver al menu principal
Seleccione una opcion: 5
Ingresar la tarea a modificar (exacta porfavor): prueba de validacion de listas
Tarea encontrada: prueba de validacion de listas
```

- Registrar procesos con nombre y prioridad.
-
- Agregar, buscar, eliminar y modificar procesos en una lista.
-
- Simular gestión de memoria utilizando una pila LIFO.
-
- Simular la planificación de procesos con una cola ordenada por prioridad.
-
-
- Mostrar información clara y estructurada al usuario

DIAGRAMA DE FLUJO



MENU DE PROGRAMA

```
C:\Users\EstudianteUC\Docum X + v

=== Menu 1 ===
1. Lista de Tareas
2. Pila de Tareas
3. Cola de Tareas
4. Salir
Seleccione una opcion mi estimado: 1

--- menu 2 lista ---
1. Agregar tarea
2. Eliminar tarea
3. Verificar y mostrar tareas mi estimado
4. Buscar tarea
5. Modificar tarea
6. Volver al menu principal
Seleccione una opcion: 1
Ingrese tarea: prueba de validacion de listas
Tarea agregada

--- menu 2 lista ---
1. Agregar tarea
2. Eliminar tarea
3. Verificar y mostrar tareas mi estimado
4. Buscar tarea
5. Modificar tarea
6. Volver al menu principal
Seleccione una opcion: 3
Tareas en lista:
1. prueba de validacion de listas

--- menu 2 lista ---
1. Agregar tarea
2. Eliminar tarea
3. Verificar y mostrar tareas mi estimado
4. Buscar tarea
5. Modificar tarea
6. Volver al menu principal
Seleccione una opcion: 4
Ingrese la tarea a buscar (exacta porfavor): prueba de validacion de listas
Tarea encontrada en la posicion 1: prueba de validacion de listas

--- menu 2 lista ---
1. Agregar tarea
2. Eliminar tarea
3. Verificar y mostrar tareas mi estimado
4. Buscar tarea
5. Modificar tarea
6. Volver al menu principal
Seleccione una opcion: 5
Ingrese la tarea a modificar (exacta porfavor): prueba de validacion de listas
Tarea encontrada: prueba de validacion de listas
```


CODIGO

```
[*] expo.cpp
1  #include <iostream>
2  #include <fstream>
3  #include <string>
4  using namespace std;
5
6  struct Proceso {
7      string nombre;
8      int prioridad;
9      Proceso* siguiente;
10 };
11
12 Proceso* listaProcesos = NULL;
13
14 void agregarProceso() {
15     cin.ignore();
16     Proceso* nuevo = new Proceso();
17     cout << "Ingrese nombre del proceso: ";
18     getline(cin, nuevo->nombre);
19     cout << "Ingrese prioridad del proceso (1-10): ";
20     cin >> nuevo->prioridad;
21     nuevo->siguiente = NULL;
22
23     if (listaProcesos == NULL) {
24         listaProcesos = nuevo;
25     } else {
26         Proceso* actual = listaProcesos;
27         while (actual->siguiente != NULL) {
28             actual = actual->siguiente;
29         }
30         actual->siguiente = nuevo;
31     }
32     cout << "Proceso agregado.\n";
33 }
34
35 void eliminarProceso() {
36     cin.ignore();
37     if (listaProcesos == NULL) {
38         cout << "No hay procesos registrados.\n";
39         return;
40     }
41     string nombre;
42     cout << "Ingrese el nombre del proceso a eliminar: ";
43     getline(cin, nombre);
44
45     Proceso* actual = listaProcesos;
46     Proceso* anterior = NULL;
```

```
[*] expo.cpp
46     Proceso* anterior = NULL;
47
48     while (actual != NULL && actual->nombre != nombre) {
49         anterior = actual;
50         actual = actual->siguiente;
51     }
52
53     if (actual == NULL) {
54         cout << "Proceso no encontrado.\n";
55         return;
56     }
57
58     if (anterior == NULL) {
59         listaProcesos = actual->siguiente;
60     } else {
61         anterior->siguiente = actual->siguiente;
62     }
63     delete actual;
64     cout << "Proceso eliminado.\n";
65 }
66
67 void mostrarProcesos() {
68     if (listaProcesos == NULL) {
69         cout << "No hay procesos registrados.\n";
70         return;
71     }
72     Proceso* actual = listaProcesos;
73     cout << "\nLista de procesos:\n";
74     while (actual != NULL) {
75         cout << "- Nombre: " << actual->nombre << " | Prioridad: " << actual->prioridad << endl;
76         actual = actual->siguiente;
77     }
78 }
79
80 void buscarProcesoPorNombre() {
81     cin.ignore();
82     if (listaProcesos == NULL) {
83         cout << "No hay procesos registrados.\n";
84         return;
85     }
86     string nombre;
87     cout << "Ingrese el nombre del proceso a buscar: ";
88     getline(cin, nombre);
89
90     Proceso* actual = listaProcesos;
91     while (actual != NULL) {
92         if (actual->nombre == nombre) {
93             cout << "Proceso encontrado: " << actual->nombre << " | Prioridad: " << actual->prioridad << endl;
94             return;
95         }
96         actual = actual->siguiente;
97     }
98     cout << "Proceso no encontrado.\n";
99 }
100
101 void modificarPrioridad() {
102     cin.ignore();
103     if (listaProcesos == NULL) {
104         cout << "No hay procesos registrados.\n";
105         return;
106     }
107     string nombre;
108     cout << "Ingrese el nombre del proceso a modificar: ";
109     getline(cin, nombre);
110
111     Proceso* actual = listaProcesos;
112     while (actual != NULL) {
113         if (actual->nombre == nombre) {
114             cout << "Prioridad actual: " << actual->prioridad << endl;
115             cout << "Ingrese nueva prioridad: ";
116             cin >> actual->prioridad;
117             cout << "Prioridad modificada con exito.\n";
118             return;
119         }
120         actual = actual->siguiente;
121     }
122     cout << "Proceso no encontrado.\n";
123 }
124
125 Proceso* frenteCola = NULL;
126 Proceso* finCola = NULL;
127
128 void encolarProcesoCPU() {
129     cin.ignore();
130     Proceso* nuevo = new Proceso();
131     cout << "Nombre del proceso: ";
132     getline(cin, nuevo->nombre);
133     cout << "Prioridad (1-10): ";
134     cin >> nuevo->prioridad;
```

```
[*] expo.cpp
88     getline(cin, nombre);
89
90     Proceso* actual = listaProcesos;
91     while (actual != NULL) {
92         if (actual->nombre == nombre) {
93             cout << "Proceso encontrado: " << actual->nombre << " | Prioridad: " << actual->prioridad << endl;
94             return;
95         }
96         actual = actual->siguiente;
97     }
98     cout << "Proceso no encontrado.\n";
99 }
100
101 void modificarPrioridad() {
102     cin.ignore();
103     if (listaProcesos == NULL) {
104         cout << "No hay procesos registrados.\n";
105         return;
106     }
107     string nombre;
108     cout << "Ingrese el nombre del proceso a modificar: ";
109     getline(cin, nombre);
110
111     Proceso* actual = listaProcesos;
112     while (actual != NULL) {
113         if (actual->nombre == nombre) {
114             cout << "Prioridad actual: " << actual->prioridad << endl;
115             cout << "Ingrese nueva prioridad: ";
116             cin >> actual->prioridad;
117             cout << "Prioridad modificada con exito.\n";
118             return;
119         }
120         actual = actual->siguiente;
121     }
122     cout << "Proceso no encontrado.\n";
123 }
124
125 Proceso* frenteCola = NULL;
126 Proceso* finCola = NULL;
127
128 void encolarProcesoCPU() {
129     cin.ignore();
130     Proceso* nuevo = new Proceso();
131     cout << "Nombre del proceso: ";
132     getline(cin, nuevo->nombre);
133     cout << "Prioridad (1-10): ";
134     cin >> nuevo->prioridad;
```

CODIGO

```
[*] expo.cpp
133 cout << "Prioridad (1-10): ";
134 cin >> nuevo->prioridad;
135 nuevo->siguiente = NULL;
136
137 if (frenteCola == NULL) {
138     frenteCola = finCola = nuevo;
139 } else {
140     finCola->siguiente = nuevo;
141     finCola = nuevo;
142 }
143 cout << "Proceso encolado.\n";
144 }
145
146 void desencolarProcesoCPU() {
147     if (frenteCola == NULL) {
148         cout << "La cola de procesos esta vacia.\n";
149         return;
150     }
151     Proceso* temp = frenteCola;
152     frenteCola = frenteCola->siguiente;
153     cout << "Ejecutando proceso: " << temp->nombre << endl;
154     delete temp;
155 }
156
157 void mostrarColaCPU() {
158     if (frenteCola == NULL) {
159         cout << "La cola de procesos esta vacia.\n";
160         return;
161     }
162     Proceso* actual = frenteCola;
163     cout << "\nCola de procesos:\n";
164     while (actual != NULL) {
165         cout << "- " << actual->nombre << " | Prioridad: " << actual->prioridad << endl;
166         actual = actual->siguiente;
167     }
168 }
169
170 Proceso* topePila = NULL;
171
172 void asignarMemoria() {
173     cin.ignore();
174     Proceso* nuevo = new Proceso();
175     cout << "Proceso a asignar memoria: ";
176     getline(cin, nuevo->nombre);
177     nuevo->siguiente = topePila;
178     topePila = nuevo;
179     cout << "Memoria asignada al proceso.\n";
180 }
```

```
[*] expo.cpp
178 topePila = nuevo;
179 cout << "Memoria asignada al proceso.\n";
180 }
181
182 void liberarMemoria() {
183     if (topePila == NULL) {
184         cout << "No hay bloques de memoria asignados.\n";
185         return;
186     }
187     Proceso* temp = topePila;
188     topePila = topePila->siguiente;
189     cout << "Memoria liberada del proceso: " << temp->nombre << endl;
190     delete temp;
191 }
192
193 void mostrarMemoria() {
194     if (topePila == NULL) {
195         cout << "No hay memoria asignada.\n";
196         return;
197     }
198     Proceso* actual = topePila;
199     cout << "\nPila de memoria:\n";
200     while (actual != NULL) {
201         cout << "- " << actual->nombre << endl;
202         actual = actual->siguiente;
203     }
204 }
205
206 void menuPrincipal() {
207     int opcion;
208     do {
209         cout << "\n===== SISTEMA DE GESTION DE PROCESOS =====\n";
210         cout << "1. Gestor de Procesos (Lista)\n";
211         cout << "2. Planificador de CPU (Cola)\n";
212         cout << "3. Gestor de Memoria (Pila)\n";
213         cout << "4. Salir\n";
214         cout << "Seleccione una opcion: ";
215         cin >> opcion;
216
217         switch(opcion) {
218             case 1: {
219                 int op;
220                 do {
221                     cout << "\n--- GESTOR DE PROCESOS ---\n";
222                     cout << "1. Agregar\n2. Eliminar\n3. Mostrar\n4. Buscar proceso\n5. Modificar prioridad\n6. Volver\nOpcion: ";
223                     cin >> op;
224                     switch(op) {
225                         case 1: {
226                             asignarMemoria();
227                             break;
228                         }
229                         case 2: {
230                             liberarMemoria();
231                             break;
232                         }
233                         case 3: {
234                             mostrarColaCPU();
235                             break;
236                         }
237                         case 4: {
238                             mostrarMemoria();
239                             break;
240                         }
241                         case 5: {
242                             desencolarProcesoCPU();
243                             break;
244                         }
245                         case 6: {
246                             return;
247                         }
248                     }
249                 } while (op != 6);
250             }
251             default: {
252                 cout << "Opcion no valida.\n";
253             }
254         }
255     } while (opcion != 4);
256 }
```

CODIGO

```
[*] expo.cpp
223     cin >> op;
224     switch(op) {
225         case 1: agregarProceso(); break;
226         case 2: eliminarProceso(); break;
227         case 3: mostrarProcesos(); break;
228         case 4: buscarProcesoPorNombre(); break;
229         case 5: modificarPrioridad(); break;
230     }
231     } while(op != 6);
232     break;
233 }
234 case 2: {
235     int op;
236     do {
237         cout << "\n--- PLANIFICADOR DE CPU ---\n";
238         cout << "1. Encolar\n2. Desencolar\n3. Mostrar\n4. Volver\nOpcion: ";
239         cin >> op;
240         switch(op) {
241             case 1: encolarProcesoCPU(); break;
242             case 2: desencolarProcesoCPU(); break;
243             case 3: mostrarColaCPU(); break;
244         }
245     } while(op != 4);
246     break;
247 }
248 case 3: {
249     int op;
250     do {
251         cout << "\n--- GESTOR DE MEMORIA ---\n";
252         cout << "1. Asignar\n2. Liberar\n3. Mostrar\n4. Volver\nOpcion: ";
253         cin >> op;
254         switch(op) {
255             case 1: asignarMemoria(); break;
256             case 2: liberarMemoria(); break;
257             case 3: mostrarMemoria(); break;
258         }
259     } while(op != 4);
260     break;
261 }
262 case 4: cout << "Saliendo del sistema...\n"; break;
263 default: cout << "Opcion invalida. Intente de nuevo.\n";
264 }
265 } while(opcion != 4);
266 }
267
268 int main() {
269     menuPrincipal();
270     return 0;
271 }
```

```
266 }
267
268 int main() {
269     menuPrincipal();
270     return 0;
271 }
```

Compile Log Debug Find Results Close

**MUCHAS
GRACIAS**

