**Final Algoritmos y estructura de datos 24-07-2023**

*Temas evaluados: estructura de datos, estructuras enlazadas, resolucion de problemas*

Usted es parte de un equipo de desarrollo que colabora con el hospital publico de pinamar y debe desarrollar una aplicación para administrar la cola de pacientes que requieren atencion en la guardia del mismo según la prioridad de su estado

**Descripcion del problema**

*Analisis del contexto:* la guardia del hospital publico atiende pacientes con diferentes grados de complejidad y dispone del siguiente protocolo de prioridades:

|  |  |  |
| --- | --- | --- |
| Prioridad | Estado | Tiempo de atencion |
| 0 | Riesgo de vida | Asistencia inmediata |
| 1 | Emergencia grave | Tiempo asistencia < 15 minutos |
| 2 | Emergencia leve | Tiempo asistencia < 30 minutos |
| 3 | Urgencia | Tiempo asistencia < 50 minutos |
| 4 | Dolencia menor | Tiempo asistencia < 4 horas |

La estructura ***Cola de Prioridad*,** se caracteriza porque en el proceso de insercion se agrega un elemento considerando su prioridad y al seleccionar uno se elige el de mayor prioridad. Por convencion, en programacion, la maxima prioridad se le asigna la clave 0 (cero), a la siguiente 1 (uno) y asi sucesivamente, por lo que la clave de minima prioridad es “N”.

El hospital contempla las cinco prioridades descriptas y las colas deben ser implementadas en una lista de cinco nodos cada uno de los cuales con el puntero de control de cada cola de la prioridad que corresponda, con el nombre del paciente (cadena de 40 caracteres) como informacioa.

Las *Operaciones a implementar* son las siguientes:

|  |  |
| --- | --- |
| *crearColaPrioridad* | Inicializa la estructura sin elementos según prioridad y categoria |
| *insertarEnPrioridad* | Añade un elemento según prioridad y categoria |
| *elementoMinimo* | Retorna el elemento de la cola con mas alta prioridad, según cat. |
| *quitarMinimo* | Quita de la cola el elemento con mas alta prioridad |
| *colaVacia* | Comprueba si una cola determinada no tiene elementoz |
| *prioridadVacia* | Comprueba si todas las colas de la estructura estan vacial |

1. *La estructura colaDePrioridades debe implementarse en una estructura enlazada, con un nodo para cada cola que controla, conteniendo la prioridad y los punteros de control de la misma*
2. *El campo de informacion del nodo es una cadena de 40 caracteres*
3. *Las funciones* ***insertarOrdenado, queue, unQueue, atenderPaciente(nombre)****, de necesitarlas pueden invocarse sin desarrollar*

*Se pide:*

1. Declare el prototipo de las operaciones descriptas con una definicion precisa del valor de retorno y los parametros de cada una de ellas, con pre y poscondiciones. (4 puntos)
2. Implemente un programa interactivo con invocaciones correctas a las operaciones que simule el proceso completo (4 puntos)
3. Declare y defina TODAS las estructuras de datos necesarias para la correcta implementacion de lo pedido (2 puntos)

1

Declare el prototipo de las operaciones descriptas con una definicion precisa del valor de retorno y los parametros de cada una de ellas, con pre y poscondiciones. (4 puntos)

**nodoPrioridad\* crearColaPrioridad();**

Retorna el puntero a una lista con cinco nodos, en cada uno de los cuales tiene un puntero al frente y otro al fin de la cola de pacientes según la prioridad. El primer nodo corresponde a prioridad 0 (cero) y el quinto nodo al de prioridad 4 (cuatro).

**void insertarEnPrioridad(char\* , int);**

Inserta la cadena representada con char\* (el nombre de un paciente) en la prioridad indicada por el entero. Precondicion: la cadena no debe ser vacia y la prioridad esta en el rango [0..4].

**nodoPrioridad\* elementoMinimo(nodoPrioridad\*);**

Recibe el puntero a la lista que contiene las colas. Retorna el puntero al nodo que contiene el puntero a la cola no vacia de la mayor prioridad. Precondicion: existe al menos una cola No vacia.

**char\* quitarMinimo( nodoPrioridad\*);**

Retorna el nombre del paciente de la cola de mayor prioridad. Precondicion: existe al menos una cola no vacia.

**int colaVacia(int);**

Recibe como parametro un entero en el rango [1..4]. Retorna(colavacia?1:0).

**int prioridadVacia(nodoPrioridad\*);**

Recibe como parametro un puntero al inicio de la lista que contiene las colas de prioridades. Retorna(todas las colas vacias?1:0).

2

Implemente un programa interactivo con invocaciones correctas a las operaciones que simule el proceso completo (4 puntos)

int main(){

char opcion; nodoColaPrioridad prioridades,ptr = NULL; int prioridad;

/\*a:crearColaPrioridad, b:insertarEnPrioridad, c:elementoMinimo, d:quitarMinimo, e:colaVacia, f:prioridadVacia\*/

cout<<” Ingrese una opcion “; cin>>opcion;

switch(opcion){

case ‘a’: prioridades = crearColaPrioridad();break;

case ‘b’:cin>>nombre;cin>>prioridad;insertarPrioridad(nombre, prioridad;break;

case’c’:ptr = elementoMinimo(prioridades);break;

case ‘d’: atenderPaciente(quitarMinimo(prioridades)); break;

case ‘e’: cin>>prioridad;if(colaVacia(prioridad);cout<<”sin pacientes”;breck;

case ‘f’: if (prioridadVacia(prioridades)) cout<<”guardia sin pacientes”;breack;

default: cout<<”error”;

}

return 0;

}

Declare y defina TODAS las estructuras de datos necesarias para la correcta implementacion de lo pedido (2 puntos)

struct nodoCola{

char nombre[41];

nodoCola\* sgte;

}

struct tipoInfoPrioridades{

nodoCola\* frente;

nodoCola\* fin;

}

structNodoPrioridad{

tipoInfoPrioridades info;

tipoNodoPrioridades\* sgte;

}