Universidad San Carlos de Guatemala

Facultad de Ingeniería

Estructura de Datos, Sección B

Ing. Álvaro Obrayan Hernández García

Auxiliar: Carlos Javier Castro González

Primer Semestre 2024

Tema:

Manuel Técnico – Proyecto Fase 1

Nombre: Carlos Manuel Lima y Lima

Registro Académico: 202201524

CUI: 3009368850101

Guatemala, 28 de febrero del 2024.

MODULO COLA DE CLIENTES

El módulo modulo_cola_cliente implementa una estructura de datos de cola para almacenar información sobre los clientes. Cada nodo en la cola representa a un cliente y contiene información como el ID del cliente, el nombre, las imágenes grandes y pequeñas, y los conteos de imágenes grandes y pequeñas.

Las subrutinas en este módulo son:

- ***push_cliente:*** Agrega un nuevo nodo al final de la cola. El nuevo nodo se llena con la información del cliente proporcionada.
- ***pop_cliente:*** Elimina el nodo en la parte frontal de la cola y devuelve la información del cliente en ese nodo.
- ***print_cliente:*** Imprime la información de todos los clientes en la cola.
- ***graphic_cliente:*** Genera un gráfico de la cola de clientes utilizando la herramienta Graphviz. Cada nodo en el gráfico representa a un cliente en la cola. La subrutina escribe el gráfico en un archivo .dot y luego lo convierte a un archivo .pdf utilizando el comando dot -Tpdf.
- ***top5_img_grandes:*** Imprime los nombres de los cinco clientes que tienen la mayor cantidad de imágenes grandes. Si hay menos de cinco clientes en la cola, imprime la información de todos los clientes.
- ***top5_img_pequenas:*** Imprime los nombres de los cinco clientes que tienen la menor cantidad de imágenes pequeñas. Si hay menos de cinco clientes en la cola, imprime la información de todos los clientes.

```
module modulo_cola_cliente
    implicit none
    type :: cola_cliente
    type(nodo_cola_cliente), pointer :: cabeza => null()
    contains
        procedure :: push_cliente
        procedure :: pop_cliente
        procedure :: print_cliente
        procedure :: top5_img_grandes
        procedure :: top5_img_pequenas
        procedure :: graphic_cliente
    end type cola_cliente

    type :: nodo_cola_cliente
        character(len=:), allocatable :: id_cliente
        character(len=:), allocatable :: nombre
        character(len=:), allocatable :: img_grande
        character(len=:), allocatable :: img_pequena
        integer :: pequena, grande
        type(nodo_cola_cliente), pointer :: siguiente
    end type nodo_cola_cliente

    type :: cliente_conteo
        character(len=:), allocatable :: nombre
        integer :: grande, pequena
    end type cliente_conteo
```

MODULO PILA IMÁGENES

El módulo modulo_pila_imagenes en implementa una estructura de datos de pila para almacenar información sobre las imágenes. Cada nodo en la pila representa una imagen y contiene información sobre el tipo de imagen.

Las subrutinas en este módulo son:

- ***push_imagen:*** Agrega un nuevo nodo al principio de la pila. El nuevo nodo se llena con el tipo de imagen proporcionado.
- ***pop_imagen:*** Elimina el nodo en la parte superior de la pila y devuelve el tipo de imagen en ese nodo.
- ***print_imagen:*** Imprime el tipo de todas las imágenes en la pila.
- ***clean_imagen:*** Elimina todos los nodos de la pila.

```
module modulo_pila_imagenes
    type :: pila_imagenes
    type(nodo_pila_imagen), pointer :: cabeza => null()
    contains
        procedure :: push_imagen
        procedure :: pop_imagen
        procedure :: print_imagen
        procedure :: clean_imagen
    end type pila_imagenes
    type :: nodo_pila_imagen
        character(len=:), allocatable :: tipo_imagen
        type(nodo_pila_imagen), pointer :: siguiente
    end type nodo_pila_imagen

    contains
    subroutine push_imagen(self,tipo_imagen)
        class(pila_imagenes), intent(inout) :: self
        character(len=*), intent(in) :: tipo_imagen
        type(nodo_pila_imagen), pointer :: nuevo_nodo
        allocate(nuevo_nodo)
        nuevo_nodo%tipo_imagen = tipo_imagen
        nuevo_nodo%siguiente => self%cabeza
        self%cabeza => nuevo_nodo
    end subroutine push_imagen
```

```
subroutine pop_imagen(self, tipo_imagen)
    class(pila_imagenes), intent(inout) :: self
    character(len=20), dimension(:), intent(out) :: tipo_imagen
    type(nodo_pila_imagen), pointer :: temp
    if (.not. associated(self%cabeza)) then
        print *, "PILA VACIA"
    else
        temp => self%cabeza
        tipo_imagen(1) = self%cabeza%tipo_imagen
        self%cabeza => self%cabeza%siguiente
        deallocate(temp)
    end if
end subroutine pop_imagen

subroutine print_imagen(self)
    class(pila_imagenes), intent(in) :: self
    type(nodo_pila_imagen), pointer :: actual
    actual => self%cabeza
    print *, "Pila De Imagenes:"
    do while (associated(actual))
        print *, actual%tipo_imagen
        actual => actual%siguiente
    end do
end subroutine print_imagen

subroutine clean_imagen(self)
    class(pila_imagenes), intent(inout) :: self
    type(nodo_pila_imagen), pointer :: temp
    do while (associated(self%cabeza))
        temp => self%cabeza
        self%cabeza => self%cabeza%siguiente
        deallocate(temp)
    end do
end subroutine clean_imagen
```

MODULO COLA DE IMPRESORA PEQUEÑA

El módulo modulo_cola_impresora_pequena implementa una estructura de datos de cola para almacenar información sobre las imágenes pequeñas. Cada nodo en la cola representa una imagen y contiene información sobre el tipo de imagen.

Las subrutinas en este módulo son:

- ***push_img_pequena:*** Esta subrutina agrega un nuevo nodo al final de la cola. El nuevo nodo se llena con la información de la imagen proporcionada si es de tipo “Pequena”. Esta subrutina es útil para agregar nuevas imágenes pequeñas a la cola.
- ***pop_img_pequena:*** Esta subrutina elimina el nodo en la parte frontal de la cola. Esta subrutina es útil para eliminar la imagen pequeña más antigua de la cola cuando ya no es necesaria.
- ***print_img_pequena:*** Esta subrutina imprime la información de todas las imágenes en la cola. Esta subrutina es útil para visualizar todas las imágenes pequeñas en la cola.
- ***graphic_cola_imgPequena:*** Esta subrutina genera un gráfico de la cola de imágenes pequeñas utilizando la herramienta Graphviz. Cada nodo en el gráfico representa una imagen en la cola. La subrutina escribe el gráfico en un archivo .dot y luego lo convierte a un archivo .pdf utilizando el comando dot -Tpdf. Esta subrutina es útil para visualizar la estructura de la cola de imágenes pequeñas.

```
module modulo_cola_impresora_pequena
    use modulo_pila_imagenes
    implicit none
    type :: cola_impresora_pequena
    type(nodo_impresora_pequena), pointer :: cabeza => null()
    contains
        procedure :: push_img_pequena
        procedure :: pop_img_pequena
        procedure :: print_img_pequena
        procedure :: graphic_cola_imgPequena
    end type cola_impresora_pequena
    type :: nodo_impresora_pequena
        character(len=:), allocatable :: tipo_imagen
        type(nodo_impresora_pequena), pointer :: siguiente
    end type nodo_impresora_pequena
```

```
subroutine push_img_pequena(self, pila)
    class(cola_impresora_pequena), intent(inout) :: self
    type(pila_imagenes), intent(in) :: pila
    type(nodo_impresora_pequena), pointer :: actual, nuevo_nodo
    type(nodo_pila_imagen), pointer :: temp
    character(len=:), allocatable :: tipo_imagen
    temp => pila%cabeza
    do while (associated(temp))
        tipo_imagen = temp%tipo_imagen
        if (tipo_imagen == "Pequena") then
            allocate(nuevo_nodo)
            nuevo_nodo%tipo_imagen = tipo_imagen
            nuevo_nodo%siguiente => null()
            if (.not. associated(self%cabeza)) then
                self%cabeza => nuevo_nodo
            else
                actual => self%cabeza
                do while (associated(actual%siguiente))
                    actual => actual%siguiente
                end do
                actual%siguiente => nuevo_nodo
            end if
        end if
        temp => temp%siguiente
    end do
end subroutine push_img_pequena
```

MODULO COLA DE IMPRESORA GRANDE

El módulo modulo_cola_impresora_grande implementa una estructura de datos de cola para almacenar información sobre las imágenes grandes. Cada nodo en la cola representa una imagen y contiene información sobre el tipo de imagen.

Las subrutinas en este módulo son:

- ***push_img_grande:*** Esta subrutina agrega un nuevo nodo al final de la cola. El nuevo nodo se llena con la información de la imagen proporcionada si es de tipo “Grande”. Esta subrutina es útil para agregar nuevas imágenes grandes a la cola.
- ***pop_img_grande:*** Esta subrutina elimina el nodo en la parte frontal de la cola. Esta subrutina es útil para eliminar la imagen grande más antigua de la cola cuando ya no es necesaria.
- ***print_img_grande:*** Esta subrutina imprime la información de todas las imágenes en la cola. Esta subrutina es útil para visualizar todas las imágenes grandes en la cola.
- ***graphic_cola_imgGrande:*** Esta subrutina genera un gráfico de la cola de imágenes grandes utilizando la herramienta Graphviz. Cada nodo en el gráfico representa una imagen en la cola. La subrutina escribe el gráfico en un archivo .dot y luego lo convierte a un archivo .pdf utilizando el comando dot -Tpdf. Esta subrutina es útil para visualizar la estructura de la cola de imágenes grandes.

```
module modulo_cola_impresora_grande
    use modulo_pila_imagenes
    implicit none
    type :: cola_impresora_grande
    type(nodo_impresora_grande), pointer :: cabeza => null()
    contains
        procedure :: push_img_grande
        procedure :: pop_img_grande
        procedure :: print_img_grande
        procedure :: graphic_cola_imgGrande
    end type cola_impresora_grande
    type :: nodo_impresora_grande
        character(len=:), allocatable :: tipo_imagen
        type(nodo_impresora_grande), pointer :: siguiente
    end type nodo_impresora_grande
```

```
contains
subroutine push_img_grande(self, pila)
    class(cola_impresora_grande), intent(inout) :: self
    type(pila_imagenes), intent(inout) :: pila
    type(nodo_impresora_grande), pointer :: actual, nuevo_nodo
    character(len=:), allocatable :: tipo_imagen
    do while (associated(pila%cabeza))
        tipo_imagen = pila%cabeza%tipo_imagen
        if (tipo_imagen == "Grande") then
            allocate(nuevo_nodo)
            nuevo_nodo%tipo_imagen = tipo_imagen
            nuevo_nodo%siguiente => null()
            if (.not. associated(self%cabeza)) then
                self%cabeza => nuevo_nodo
            else
                actual => self%cabeza
                do while (associated(actual%siguiente))
                    actual => actual%siguiente
                end do
                actual%siguiente => nuevo_nodo
            end if
        end if
        pila%cabeza => pila%cabeza%siguiente
    end do
end subroutine push_img_grande
```

MODULO LISTA DE IMÁGENES IMPRESAS

El módulo modulo_lista_imagen_impresa implementa una estructura de datos de lista para almacenar información sobre las imágenes impresas. Cada nodo en la lista representa una imagen impresa y contiene información sobre el tipo de imagen.

Las subrutinas en este módulo son:

- ***append_imagen_impresa:*** Esta subrutina agrega un nuevo nodo al principio de la lista. El nuevo nodo se llena con la información de la imagen proporcionada. Esta subrutina es útil para agregar nuevas imágenes impresas a la lista.
- ***print_lista_imagen_impresa:*** Esta subrutina imprime la información de todas las imágenes en la lista. Esta subrutina es útil para visualizar todas las imágenes impresas en la lista.

```
module modulo_lista_imagen_impresa
    implicit none
    type :: nodo_imagen_impresa
        character(len=:), allocatable :: tipo_imagen
        type(nodo_imagen_impresa), pointer :: siguiente
    end type nodo_imagen_impresa
    type :: lista_imagen_impresa
        type(nodo_imagen_impresa), pointer :: cabeza => null()
    contains
        procedure :: append_imagen_impresa
        procedure :: print_lista_imagen_impresa
    end type lista_imagen_impresa
    contains
    subroutine append_imagen_impresa(self, tipo_imagen)
        class(lista_imagen_impresa), intent(inout) :: self
        character(len=*), intent(in) :: tipo_imagen
        type(nodo_imagen_impresa), pointer :: nuevo_nodo
        allocate(nuevo_nodo)
        nuevo_nodo%tipo_imagen = tipo_imagen
        nuevo_nodo%siguiente => self%cabeza
        self%cabeza => nuevo_nodo
    end subroutine append_imagen_impresa
    subroutine print_lista_imagen_impresa(self)
        class(lista_imagen_impresa), intent(in) :: self
        type(nodo_imagen_impresa), pointer :: actual
        actual => self%cabeza
        if (.not. associated(actual)) then
            print *, "LISTA IMAGENES IMPRESAS VACIA"
            return
        end if
        do while (associated(actual))
            print *, "Imagen: ", actual%tipo_imagen
            actual => actual%siguiente
        end do
    end subroutine print_lista_imagen_impresa
end module modulo_lista_imagen_impresa
```

MODULO LISTA DE CLIENTES EN ESPERA

El módulo modulo_lista_cliente_espera implementa una estructura de datos de lista circular doblemente enlazada para almacenar información sobre los clientes en espera. Cada nodo en la lista representa a un cliente en espera y contiene información como el ID del cliente, el nombre, las imágenes grandes y pequeñas, el número de ventanilla y la cantidad de pasos.

Las subrutinas en este módulo son:

- ***append_cliente_espera:*** Esta subrutina agrega un nuevo nodo al final de la lista. El nuevo nodo se llena con la información del cliente proporcionada. Esta subrutina es útil para agregar nuevos clientes en espera a la lista.
- ***delete_cliente_espera:*** Esta subrutina elimina un nodo específico de la lista basándose en el ID del cliente. Esta subrutina es útil para eliminar un cliente en espera de la lista cuando ya no es necesario.
- ***print_lista_cliente_espera:*** Esta subrutina imprime la información de todos los clientes en la lista. Esta subrutina es útil para visualizar todos los clientes en espera en la lista.
- ***graphic_cliente_espera:*** Esta subrutina genera un gráfico de la lista de clientes en espera utilizando la herramienta Graphviz. Cada nodo en el gráfico representa a un cliente en la lista. La subrutina escribe el gráfico en un archivo .dot y luego lo convierte a un archivo .pdf utilizando el comando dot -Tpdf. Esta subrutina es útil para visualizar la estructura de la lista de clientes en espera.

```
module modulo_lista_cliente_espera
    use modulo_lista_imagen_impresa
    implicit none
    type :: nodo_cliente_espera
    integer :: numero_ventanilla, cantidad_paso, pequena, grande
    character(len=:), allocatable :: id_cliente
    character(len=:), allocatable :: nombre
    character(len=:), allocatable :: img_grande
    character(len=:), allocatable :: img_pequena
    type(lista_imagen_impresa) :: lista_imagen_cliente
    type(nodo_cliente_espera), pointer :: anterior, siguiente
    end type nodo_cliente_espera

    type :: lista_cliente_espera
        type(nodo_cliente_espera), pointer :: cabeza => null()
    contains
        procedure :: append_cliente_espera
        procedure :: delete_cliente_espera
        procedure :: print_lista_cliente_espera
        procedure :: graphic_cliente_espera
    end type lista_cliente_espera
```

```
subroutine append_cliente_espera(self, id_cliente, nombre, img_pequena, img_grande, numero_ventanilla, cantidad_paso)
    class(lista_cliente_espera), intent(inout) :: self
    character(len=*), intent(in) :: id_cliente, nombre, img_pequena, img_grande
    integer, intent(in) :: numero_ventanilla, cantidad_paso
    integer :: pequena, grande
    type(nodo_cliente_espera), pointer :: nuevo_nodo
    READ(img_pequena, *) pequena
    READ(img_grande, *) grande
    allocate(nuevo_nodo)
    nuevo_nodo%id_cliente = id_cliente
    nuevo_nodo%nombre = nombre
    nuevo_nodo%img_pequena = img_pequena
    nuevo_nodo%img_grande = img_grande
    nuevo_nodo%pequena = pequena
    nuevo_nodo%grande = grande
    nuevo_nodo%numero_ventanilla = numero_ventanilla
    nuevo_nodo%cantidad_paso = cantidad_paso
    if (.not. associated(self%cabeza)) then
        self%cabeza => nuevo_nodo
        nuevo_nodo%anterior => nuevo_nodo
        nuevo_nodo%siguiente => nuevo_nodo
    else
        nuevo_nodo%anterior => self%cabeza%anterior
        nuevo_nodo%siguiente => self%cabeza
        self%cabeza%anterior%siguiente => nuevo_nodo
        self%cabeza%anterior => nuevo_nodo
    end if
end subroutine append_cliente_espera
```

MODULO LISTA DE CLIENTES ATENDIDOS

El módulo modulo_lista_cliente_atendido en implementa una estructura de datos de lista para almacenar información sobre los clientes atendidos. Cada nodo en la lista representa a un cliente atendido y contiene información como el ID del cliente, el nombre, las imágenes grandes y pequeñas, el número de ventanilla y la cantidad de pasos.

Las subrutinas en este módulo son:

- ***append_cliente_atendido:*** Esta subrutina agrega un nuevo nodo al principio de la lista. El nuevo nodo se llena con la información del cliente proporcionada. Esta subrutina es útil para agregar nuevos clientes atendidos a la lista.
- ***print_cliente_atendido:*** Esta subrutina imprime la información de todos los clientes en la lista. Esta subrutina es útil para visualizar todos los clientes atendidos en la lista.
- ***cliente_mayor_pasos:*** Esta subrutina encuentra e imprime la información del cliente que tiene la mayor cantidad de pasos en la lista. Esta subrutina es útil para identificar al cliente que ha realizado la mayor cantidad de pasos.
- ***graphic_clientes_atentido:*** Esta subrutina genera un gráfico de la lista de clientes atendidos utilizando la herramienta Graphviz. Cada nodo en el gráfico representa a un cliente en la lista. La subrutina escribe el gráfico en un archivo .dot y luego lo convierte a un archivo .pdf utilizando el comando dot -Tpdf. Esta subrutina es útil para visualizar la estructura de la lista de clientes atendidos.

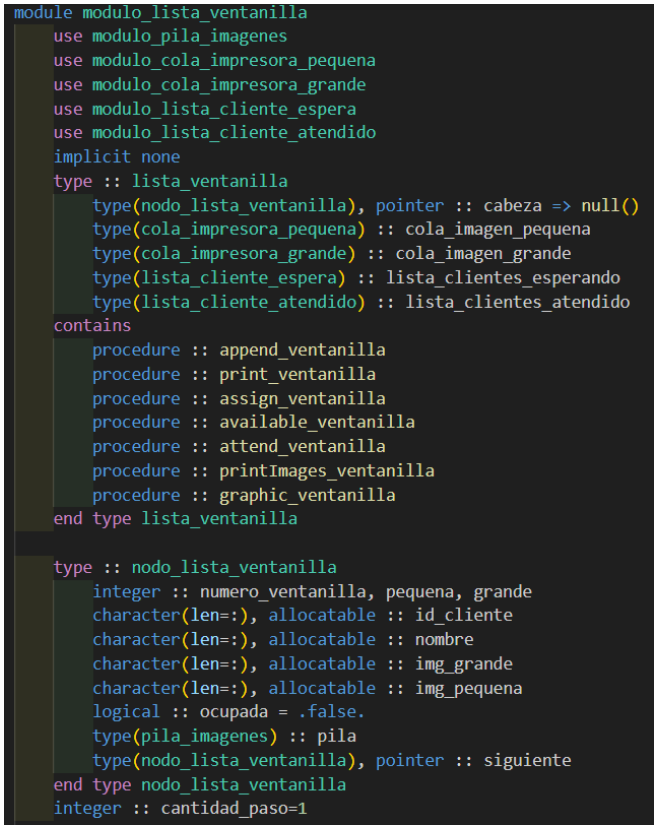
```
module modulo_lista_cliente_atendido
    implicit none
    type :: nodo_cliente_atendido
        integer :: numero_ventanilla
        character(len=:), allocatable :: id_cliente
        character(len=:), allocatable :: nombre
        character(len=:), allocatable :: img_pequena
        character(len=:), allocatable :: img_grande
        integer :: cantidad_pasos
        type(nodo_cliente_atendido), pointer :: siguiente => null()
    end type nodo_cliente_atendido

    type :: lista_cliente_atendido
        type(nodo_cliente_atendido), pointer :: cabeza => null()
    contains
        procedure :: append_cliente_atendido
        procedure :: print_cliente_atendido
        procedure :: cliente_mayor_pasos
        procedure :: graphic_clientes_atentido
    end type lista_cliente_atendido

    contains
    subroutine append_cliente_atendido(self, numero_ventanilla, id_cliente, nombre, img_pequena, img_grande, cantidad_pasos)
        class(lista_cliente_atendido), intent(inout) :: self
        character(len=*), intent(in) :: id_cliente, nombre, img_pequena, img_grande
        integer, intent(in) :: cantidad_pasos
        integer, intent(in) :: numero_ventanilla
        type(nodo_cliente_atendido), pointer :: nuevo_nodo
        allocate(nuevo_nodo)
        nuevo_nodo%numero_ventanilla = numero_ventanilla
        nuevo_nodo%id_cliente = id_cliente
        nuevo_nodo%nombre = nombre
        nuevo_nodo%img_pequena = img_pequena
        nuevo_nodo%img_grande = img_grande
        nuevo_nodo%cantidad_pasos = cantidad_pasos
        nuevo_nodo%siguiente => self%cabeza
        self%cabeza => nuevo_nodo
```

MODULO LISTA DE VENTANILLAS

El módulo modulo_lista_ventanilla implementa una estructura de datos de lista para almacenar información sobre las ventanillas. Cada nodo en la lista representa a una ventanilla y contiene información como el número de ventanilla, el ID del cliente, el nombre, las imágenes grandes y pequeñas, y si la ventanilla está ocupada o no.

Las subrutinas en este módulo son:

- ***append_ventanilla:*** Esta subrutina agrega un nuevo nodo al final de la lista. El nuevo nodo se llena con la información de la ventanilla proporcionada. Esta subrutina es útil para agregar nuevas ventanillas a la lista.
- ***print_ventanilla:*** Esta subrutina imprime la información de todas las ventanillas en la lista. Esta subrutina es útil para visualizar todas las ventanillas en la lista.
- ***available_ventanilla:*** Esta función verifica si hay alguna ventanilla disponible en la lista y devuelve un valor booleano.
- ***assign_ventanilla:*** Esta subrutina asigna un cliente a una ventanilla disponible. Actualiza la información de la ventanilla con la información del cliente proporcionada.
- ***attend_ventanilla:*** Esta subrutina procesa las imágenes de los clientes en las ventanillas. Si hay imágenes disponibles, las agrega a la pila de imágenes de la ventanilla y actualiza la cantidad de imágenes pequeñas o grandes.
- ***printImages_ventanilla:*** Esta subrutina imprime las imágenes de las ventanillas. Si hay imágenes pequeñas o grandes disponibles, las agrega a la lista de imágenes del cliente y actualiza la cantidad de imágenes pequeñas o grandes.
- ***graphic_ventanilla:*** Esta subrutina genera un gráfico de la lista de ventanillas utilizando la herramienta Graphviz. Cada nodo en el gráfico representa a una ventanilla en la lista. La subrutina escribe el gráfico en un archivo .dot y luego lo convierte a un archivo .pdf utilizando el comando dot -Tpdf.



```
module modulo_lista_ventanilla
    use modulo_pila_imagenes
    use modulo_cola_impresora_pequena
    use modulo_cola_impresora_grande
    use modulo_lista_cliente_espera
    use modulo_lista_cliente_atendido
    implicit none
    type :: lista_ventanilla
        type(nodo_lista_ventanilla), pointer :: cabeza => null()
        type(cola_impresora_pequena) :: cola_imagen_pequena
        type(cola_impresora_grande) :: cola_imagen_grande
        type(lista_cliente_espera) :: lista_clientes_esperando
        type(lista_cliente_atendido) :: lista_clientes_atendido
    contains
        procedure :: append_ventanilla
        procedure :: print_ventanilla
        procedure :: assign_ventanilla
        procedure :: available_ventanilla
        procedure :: attend_ventanilla
        procedure :: printImages_ventanilla
        procedure :: graphic_ventanilla
    end type lista_ventanilla

    type :: nodo_lista_ventanilla
        integer :: numero_ventanilla, pequena, grande
        character(len=:), allocatable :: id_cliente
        character(len=:), allocatable :: nombre
        character(len=:), allocatable :: img_grande
        character(len=:), allocatable :: img_pequena
        logical :: ocupada = .false.
        type(pila_imagenes) :: pila
        type(nodo_lista_ventanilla), pointer :: siguiente
    end type nodo_lista_ventanilla
    integer :: cantidad_paso=1
```