**Programación 3**

Nombre: Santiago Vietto

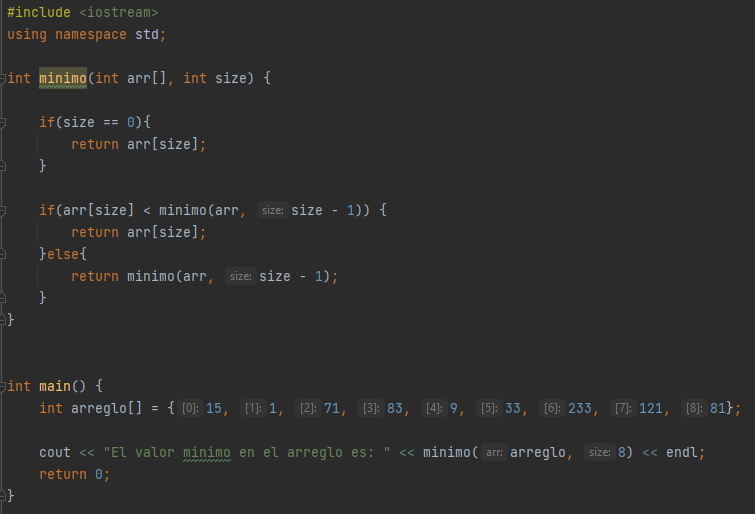
Docente: Matías Murgui

Institución: UCC

Año: 2020

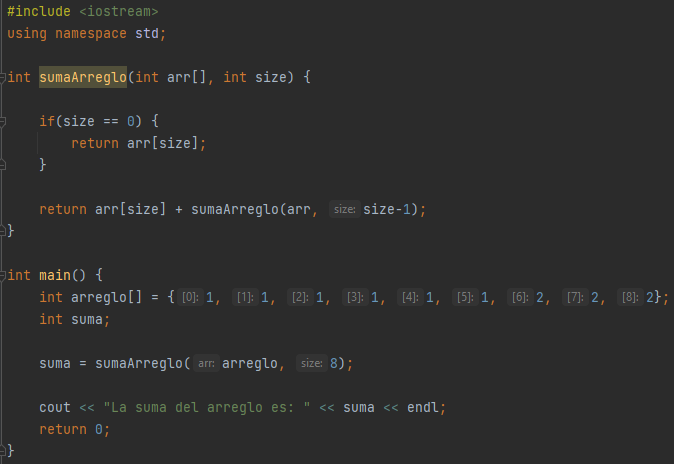
**Ejercicios tipo parcial**

1)\_ Implemente una función recursiva que reciba como parámetro un array de enteros y el tamaño y devuelva el valor más chico que encuentre en el array.

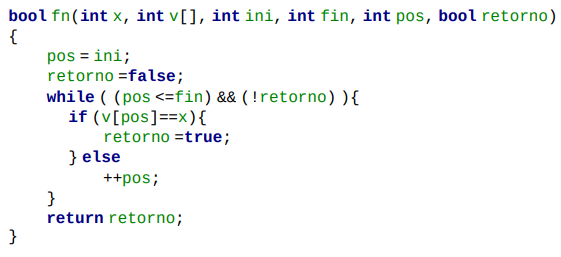


2)\_ Construir una función que sume los elementos de un arreglo de enteros recursivamente.

sumarArreglo(arr, tam);

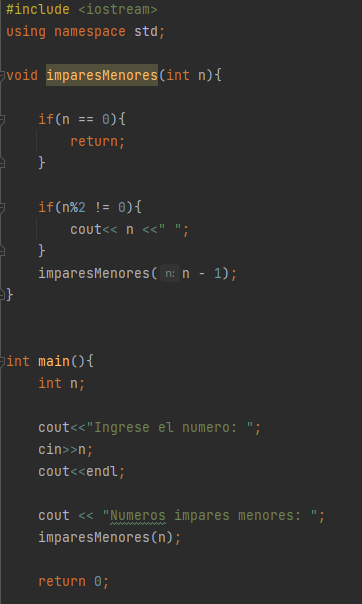


3)\_ Describir qué hace la siguiente función; si es recursiva o iterativa. Si es una función recursiva, realizar su versión iterativa. Si es una función iterativa, realizar su versión recursiva.

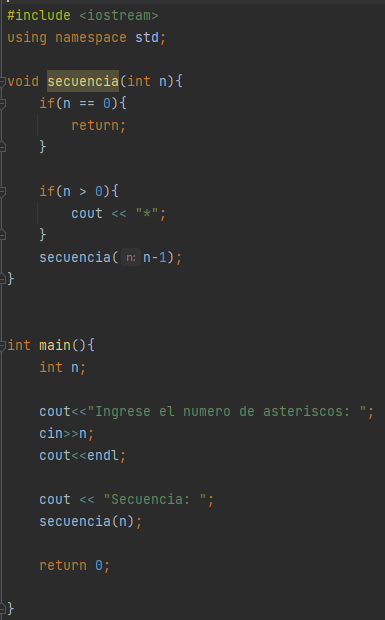




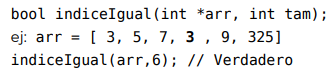
4)\_ Escriba una función void recursiva que reciba como parámetro solo un entero positivo n y que despliegue todos los enteros impares menores a n.

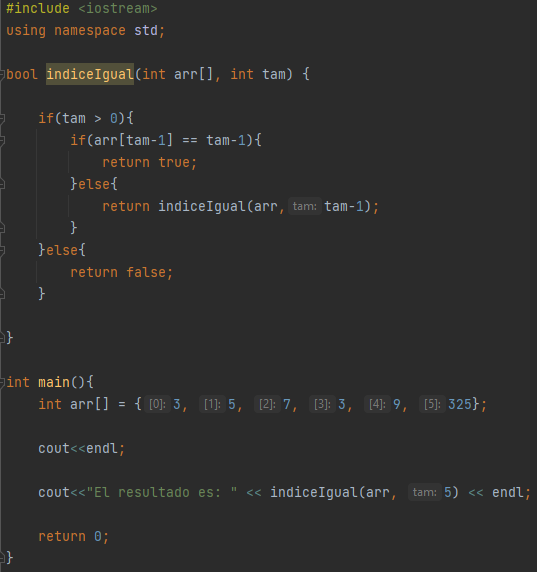


5)\_ Escribir una función recursiva que tenga solo un parámetro entero positivo n y muestre en la pantalla ese número de asteriscos “\*”, todos en una línea.



6)\_ Implemente una función recursiva que devuelva verdadero si algún valor del arreglo coincide con su índice.

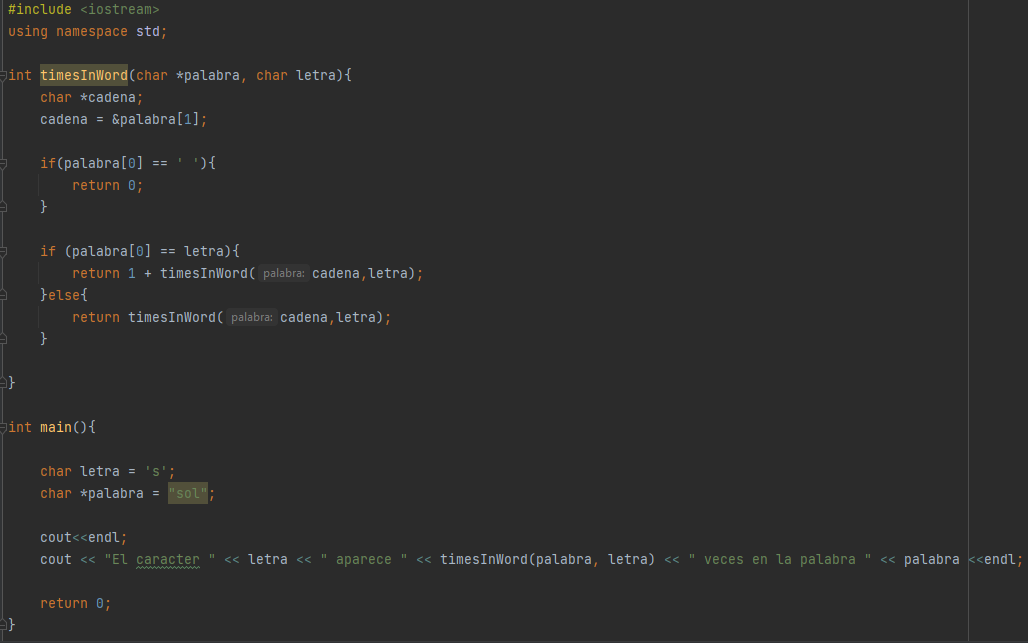




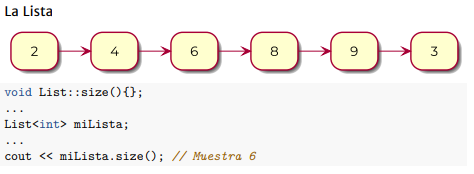
7)\_ Implementar una función recursiva que acepte por parámetro un puntero de caracteres (cadena) y un carácter, y que devuelva la cantidad de veces que aparece el carácter dentro de la cadena.

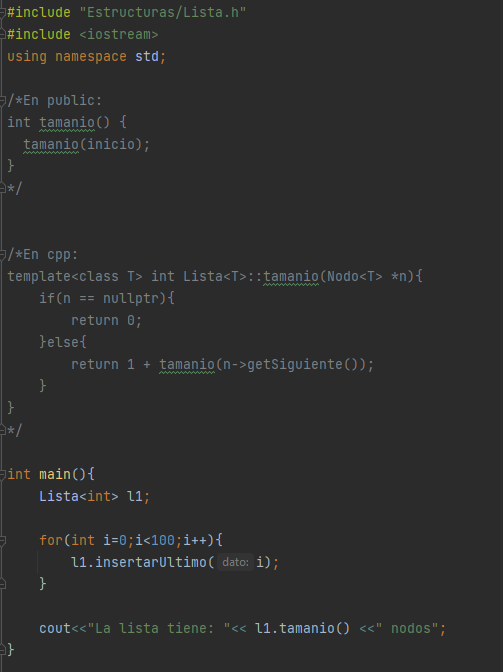
int timesInWord(char\* word, char letter){}

Ejemplo: el carácter ‘a’ aparece 2 veces en la palabra ‘programación’. Si el carácter no se encuentra en la palabra, la función deberá devolver 0.



8)\_ Escribir un método de la clase lista int tamanio(Nodo\* n) que, de manera  
recursiva, cuente cuantos nodos posee la lista enlazada.





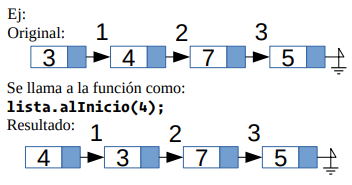
9)\_ Se pretende realizar un método en la lista enlazada que, mediante un algoritmo, calcule la suma de los enteros almacenados donde el valor sea superior a un valor umbral que se pasará como parámetro al método (utilizar los nodos de la lista):

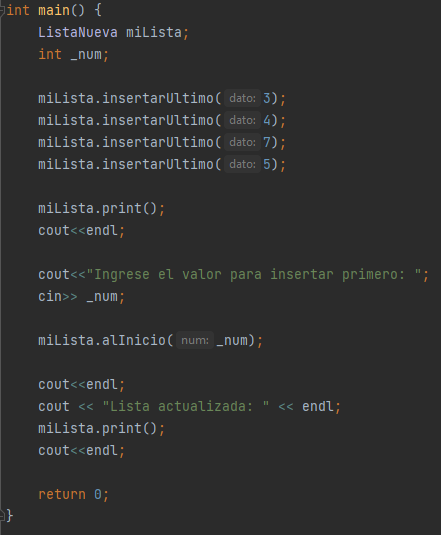
Ejemplo: A partir de la siguiente lista y el valor umbral 3.

4 → 3 → 5 → 2 → 6 → null

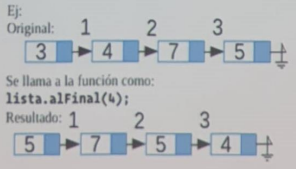
El método debe devolver el valor 15 (4 + 5 + 6)

10)\_ Implementar un método a la clase lista que reciba como parámetro un valor D. El método debe buscar este valor y moverlo a la primera posición (solo moviendo enlaces, no creando nodos nuevos).

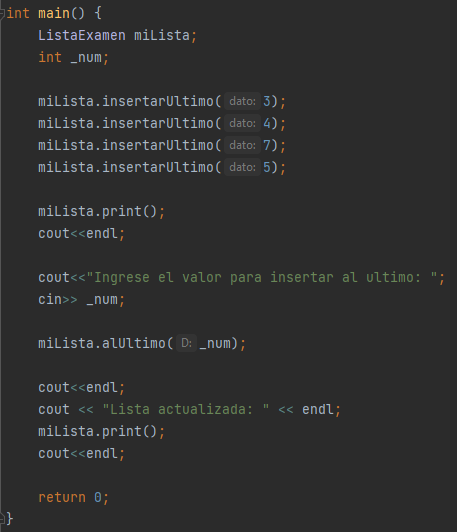


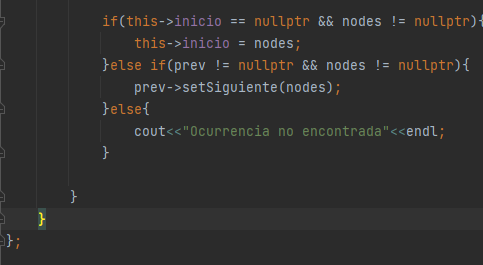


11)\_ Implementar un método a la clase lista que reciba como parámetro un valor D. El método debe buscar este valor y moverlo a la última posición (solo moviendo enlaces, no creando nodos nuevos).

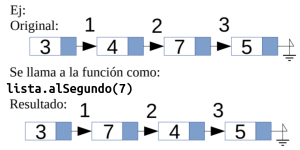




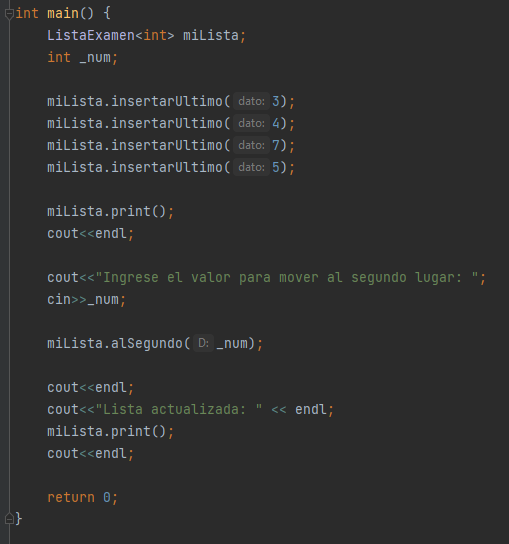




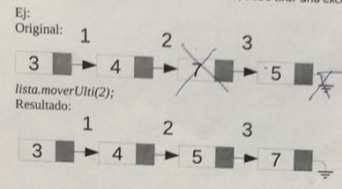
12)\_ Implementar un método a la clase lista que reciba como parámetro un valor D. El método debe buscar este valor y moverlo a la segunda posición (solo moviendo enlaces, no creando nodos nuevos).



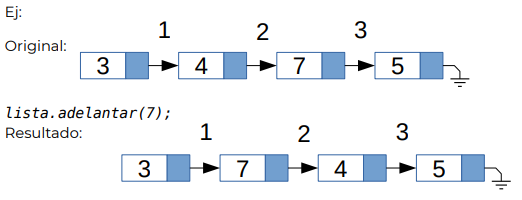


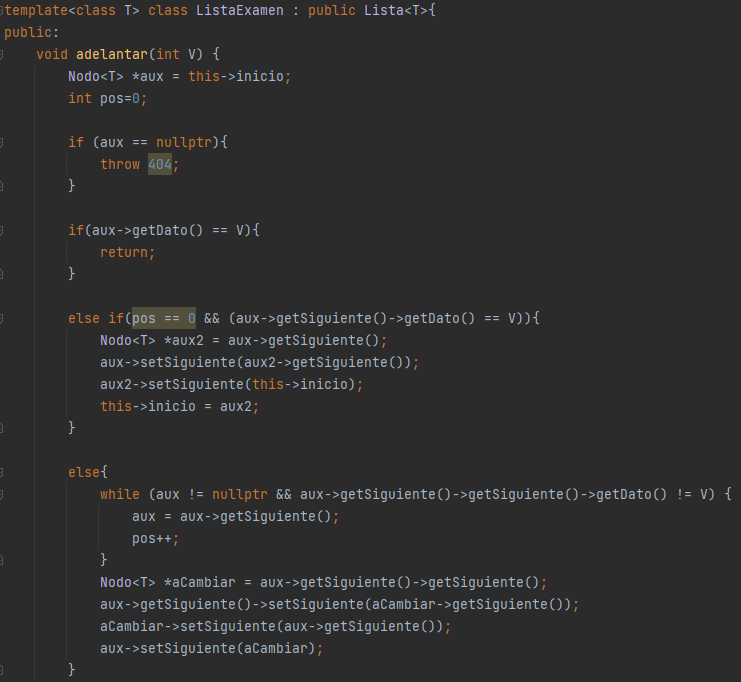


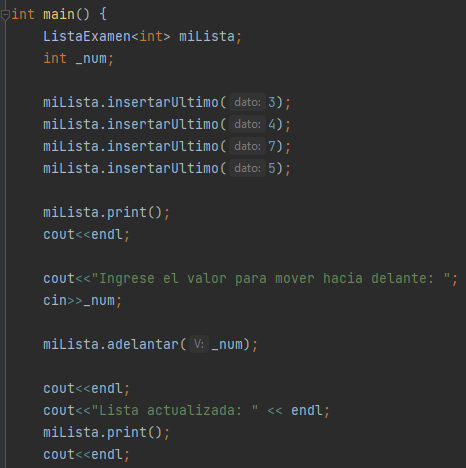
13)\_ Implementar un método a la clase lista que reciba como parámetro un valor D y mueva el elemento que está en esa posición al último lugar. Si la posición no existe, debe tirar una excepción (solo moviendo enlaces, no creando nodos nuevos).

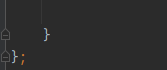


14)\_ Implementar un método a la clase lista que reciba un parámetro V y que encuentre el nodo con dicho valor y lo mueva una posición hacia adelante (no se pueden crear nuevos nodos ni eliminar nodos).









15)\_ La funcionalidad deshacer comúnmente encontrada en los programas, mantiene un historial ordenado de las operaciones que se realizaron. Cree un programa que permita elegir 4 operaciones de un menú (subir, bajar, izq y der) y una quinta opción que me permita deshacer. Al seleccionar la opción, el programa escribirá “Se realizó operación [OPERACIÓN]” y si se selecciona deshacer, deberá decir “Se deshizo [OPERACIÓN]”. Ejemplo:

MENÚ:

1. Subir
2. Bajar
3. Izq
4. Der
5. Deshacer

Se presiona la secuencia 1 3 4 5 2 3 5 5:

Se realizó operación Subir

Se realizó operación Izq

Se realizó operación Der

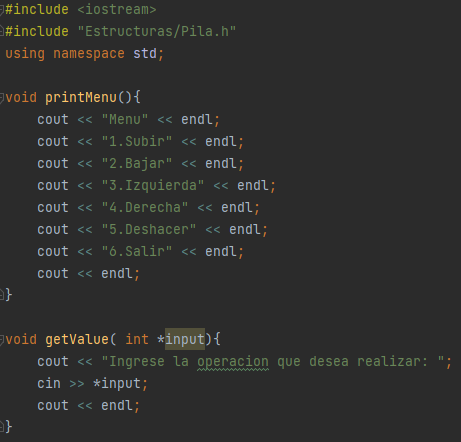
Se deshizo Der

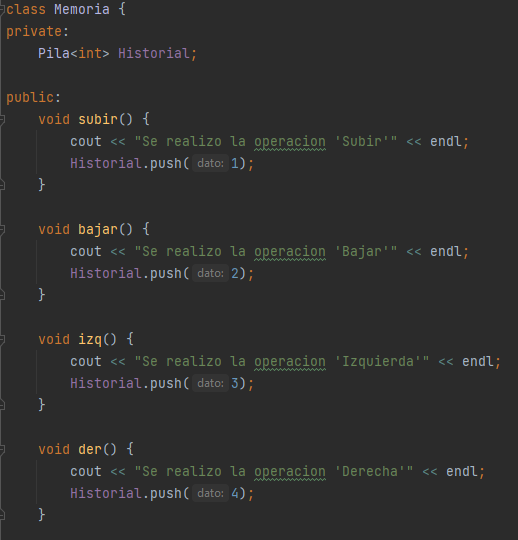
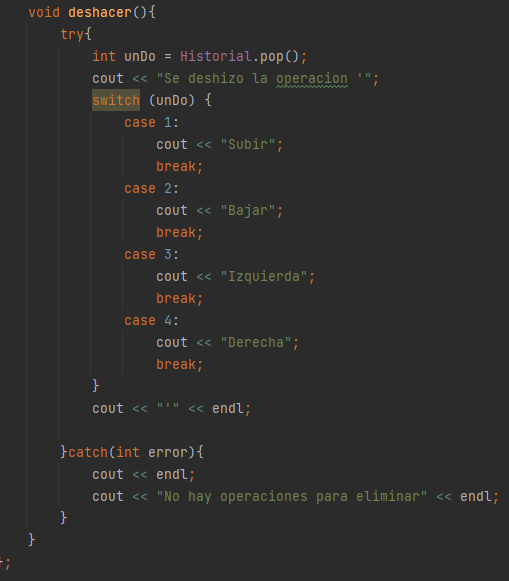
Se realizó operación Bajar

Se realizó operación Izq

Se deshizo Izq

Se deshizo Bajar

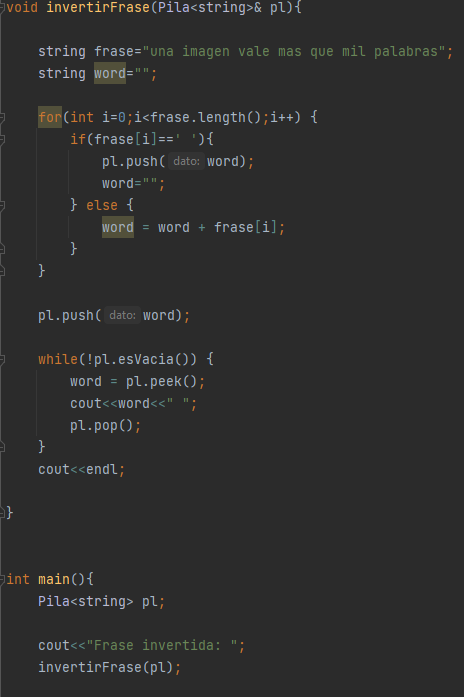




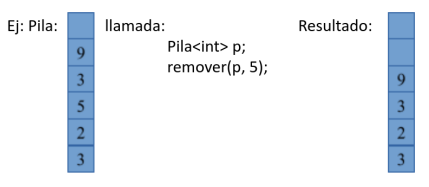


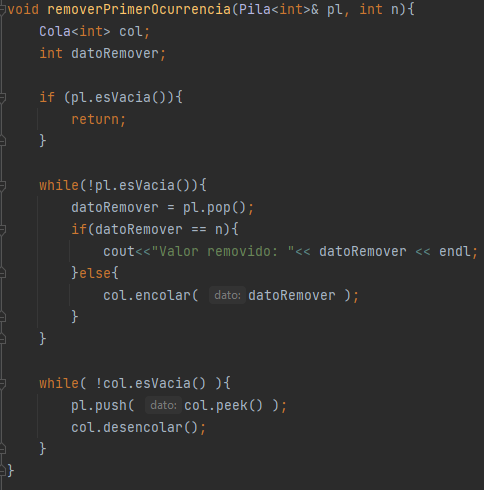
16)\_ Hacer una función que dada una frase ingresada por teclado la imprima invertida. Implementar la función con la estructura de datos vista más adecuada.

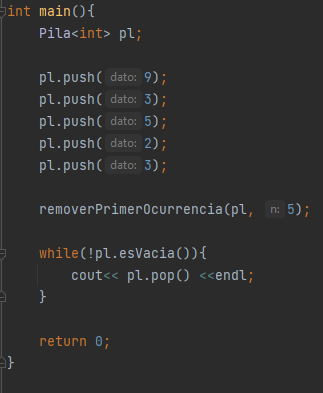
Ejemplo: “una imagen vale por mil palabras” debe convertirse en “palabras mil por vale imagen una”.



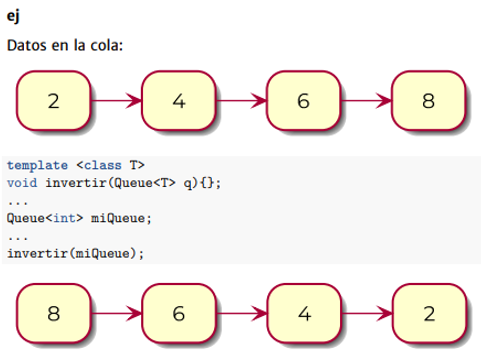
17)\_ Realice una función que reciba una pila por referencia y un número. La función debe sacar la primera ocurrencia de dicho valor de la pila.

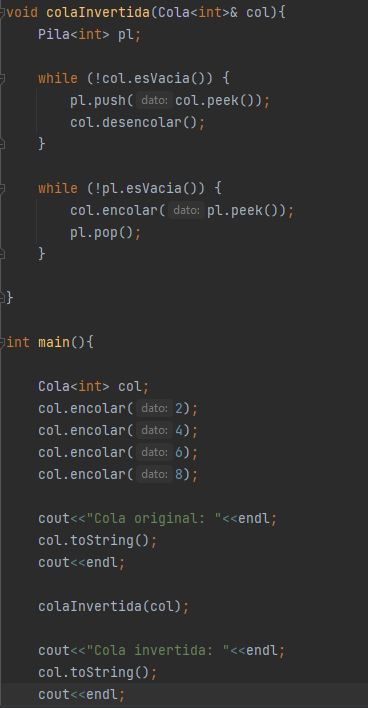






18)\_ Se pretende realizar una función que, reciba por referencia una cola e invierta todo su contenido.

 Void invertir(cola <int>&col);



19)\_ Implementar la clase PriorityQueue que seria una cola con prioridades. Dicha clase debe soportar 10 prioridades, siendo 0 la máxima prioridad. Los métodos básicos son:

Void enqueue(char data, int priority);

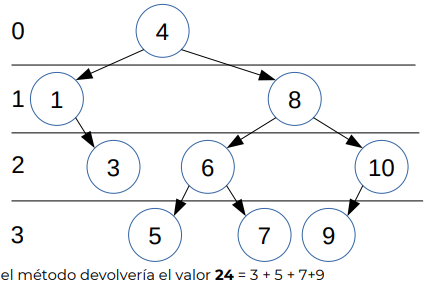
Int dequeue();

20)\_ Mostrar a través de la implementación de una clase el funcionamiento de una estructura de datos Cola utilizando exclusivamente la estructura de datos Pila vista en clase. (pista: deben utilizar métodos de la clase Pila, y no hace falta que la clase cuente con un atributo de tipo Nodo). La clase debe contar con los métodos:

* push(): permite insertar un valor en la estructura.
* pop(): permite quitar un valor de la estructura (debe quitar el valor más viejo en la estructura ya que debe responder al principio FIFO).
* esVacia(): devuelve verdadero si la estructura no tiene elementos o falso si esta tiene elementos.
* Constructor y Destructor apropiados.
* Más métodos que se les ocurran (que sean relevantes al funcionamiento de la estructura) se considerarán como bonificación extra.

21)\_ Implementar un método de la clase árbol que sume todos los valores de los nodos Hoja, es decir de los nodos que no tienen hijos. Ejemplo, para el siguiente árbol, se llama a la función de la siguiente forma:

arbol.sumarHojas();



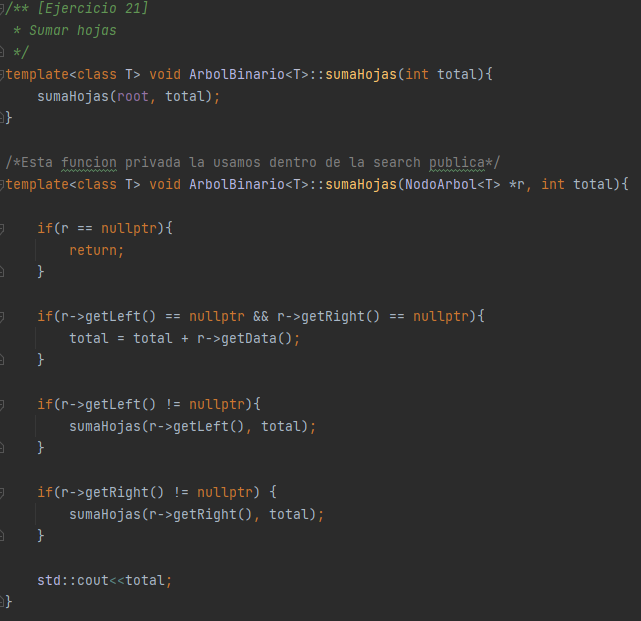
Public:

void sumaHojas(int total);

Private:

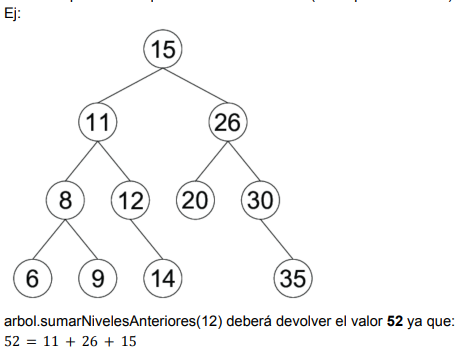
void sumaHojas(NodoArbol<T> \*r, int total);

cpp:





22)\_ Implementar un método de la clase árbol que reciba como parámetro un valor V y que devuelva la suma de los valores de todos los nodos que se encuentren en los niveles menores que el nodo que almacena el valor V (si es que este existe). Ejemplo:



Public:

int getLevel(T v);

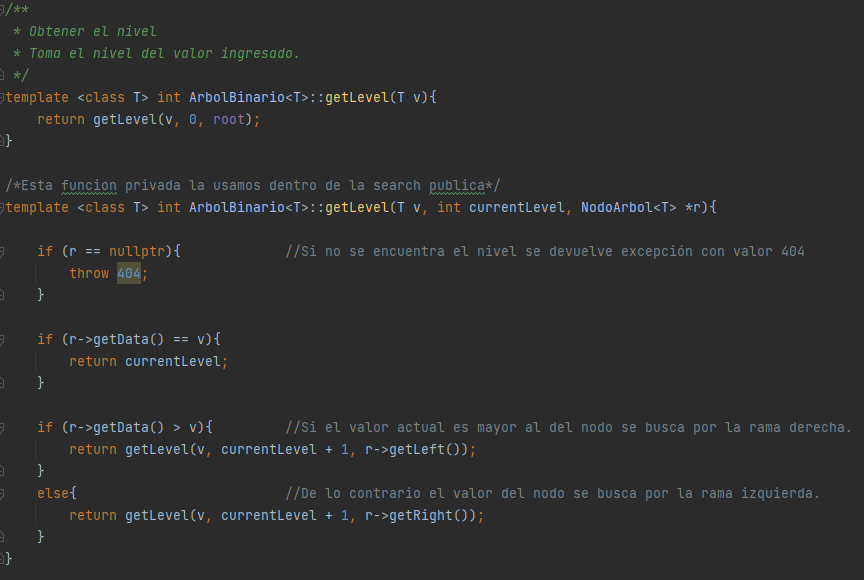
int sumarDesdeNivel(int nivel);

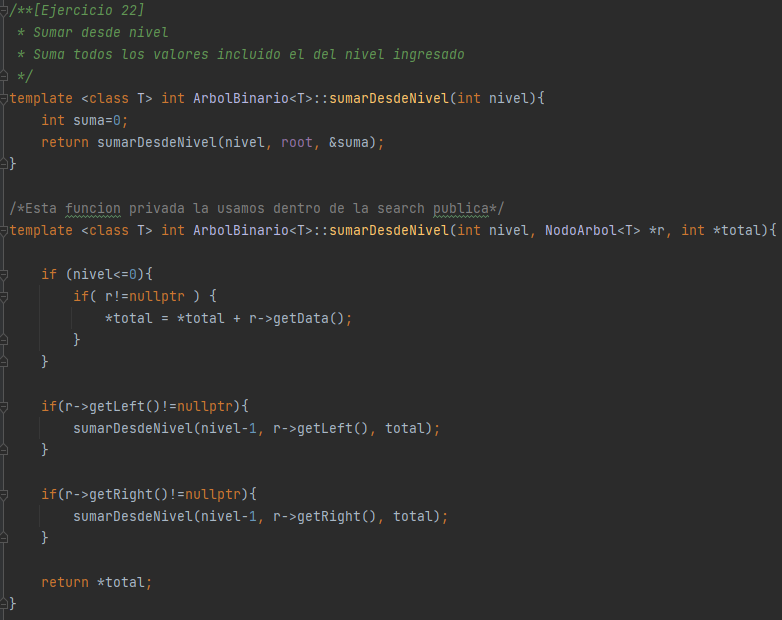
Private:

int getLevel(T v, int currentLevel, NodoArbol<T> \*r);

int sumarDesdeNivel(int nivel, NodoArbol<T> \*r, int \*total);

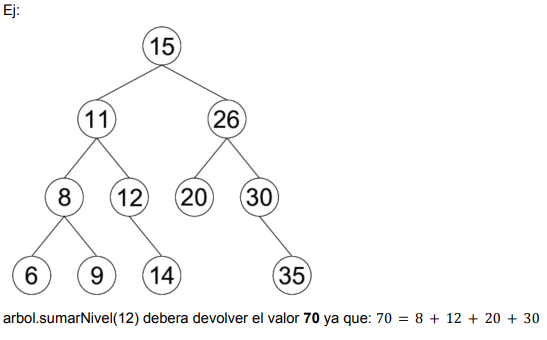
cpp:







23)\_ Implementar un método de la clase árbol que reciba como parámetro un valor V y que devuelva la suma de los valores de todos los nodos que se encuentren en el mismo nivel que el nodo que almacena el valor V (si es que este existe). Ejemplo:



Public:

int getLevel(T v);

int sumarNivel(int nivel);

Private:

int getLevel(T v, int currentLevel, NodoArbol<T> \*r);

int sumarNivel(int nivel, NodoArbol<T> \*r, int \*total);

