Nombre: Posadas Flores Armin. Materia: Estrucutra de datos. ID: 6576

Tree.h

```
#ifndef TREE_H
#define TREE_H
```

Primero tenemos que ifndef se encarga de checar si el macro "TREE_H" esta definido o no, en el caso de que no este definido todo el codigo que este entre "ifndef" y "endif" sera incluido.

define se encarga de definir TREE_H lo que causa que cada vez que el header sea incluido en algun codigo no escriba todo el contenido entre "ifndef" y "endif".

```
template <typename T>
class Node {
public:
    T data;
    Node* firstChild;
    Node* nextSibling;

    explicit Node(T data);
};
```

Primero se define el siguiente nodo como un template con los siguientes datos publicos.

`T data` se encargara de almacenar datos de tipo T.

`Node* firstChild;` Es un puntero hacia el objeto de otro nodo, representando al primer child del nodo actual en una estrucutra de arbol.

`Node* nextSibling;` Es un puntero hacia a un objeto de otro nodo, representando al siguiente dato del nodo actual en una estructura de arbol.

`explicit Node(T data);` Esto se encarga de darle al nodo un solo constructor el cual toma parametros de tipo "T" para inicializar los datos de las variables de los miembros.

```
template <typename T>
class Tree {
public:
    Node<T>* root;

    Tree();

    Node<T>* addNode(T data, Node<T>* parent = nullptr);

    void printTree(Node<T>* node, int level = 0);
```

Primero se encarga de definir el siguiente arbol como un template con los siguientes datos publicos.

`Node<T>* root;` Se encarga de definir root como parte de un nodo de este template.

`Tree();` Es el constructor de tree el cual inicializara el miembro "root" con una variable null o vacia.

`Node<T>* addNode(T data, Node<T>* parent = nullptr);` Se encarga de agregar un nuevo nodo con los datos dado al arbol por lo que si se crea un nodo llamado "parent" uno nuevo sera creado como hijo de "parent", en cambio si no hay un "parent" el hijo sera enviado al root, ambos regresan un puntero al nodo recien creado.

`void printTree(Node<T>* node, int level = 0);` Se encarga de imprimir el arbol a partir de un nodo especificado incluyendo hijos de otros nodos, mientras que el parametro "level" se encarga de medir la profundidad del arbol.

```
#include "Tree.cpp"

#endif //TREE_H
```

"#include "Tree.cpp" se encarga de separar la implementacion de un class template de su declaracion y "endif" es para asegurarse que el header file solo sean incluidos una sola vez al transladarse a otro archivo.

Tree.cpp

```
#include "Tree.h"
```

Se encargara de incluir el header Tree de hace rato.

```
template <typename T>
Node<T>::Node(T data) : data(data), firstChild(nullptr), nextSibling(nullptr) {}
```

Se trata de un constructor que implementa el nodo de la class template asi como inicializar los datos y les da un puntero de valor null a firstchild y nextsibling.

```
template <typename T>
Tree<T>::Tree() : root(nullptr) {}
```

Se encarga de inicializar el puntero root con un valor null, indicando que el arbol esta vacio.

```
template <typename T>
Node<T>* Tree<T>::addNode(T data, Node<T>* parent) {
    Node<T>* newNode = new Node<T>(data);
```

Este codigo se divide en dos la primera porcion siendo esto y se encarga de agregar un nuevo nodo al arbol, para despues tomar el nuevo nodo junto con un puntero y agregarlo a un parent node.

```
 if(parent) {
        if(parent->firstChild) {
            Node<T>* sibling = parent->firstChild;
            while(sibling->nextSibling) {
                sibling = sibling->nextSibling;
            }
            sibling->nextSibling = newNode;
        } else {
            parent->firstChild = newNode;
        }
    } else {
        root = newNode;
    }

    return newNode;
}
```

La segunda parte cnsiste en crear un nuevo nodo con datos y lo agrega ya sea como un child para un nodo "parent" en especifico o al root del arbol en caso de que no halla parent, en caso de que el parent ya tenga un child el nuevo nodo es agregado como sibling al child ya existente, al final regresa un puntero al nodo recien creado.

```
template <typename T>
void Tree<T>::printTree(Node<T>* node, int level) {
    if(!node) return;

    for(int i = 0; i < level; i++) std::cout << "--";
    std::cout << node->data << '\n';

    printTree(node->firstChild, level + 1);
    printTree(node->nextSibling, level);
}
```

Se encarga de imprimir el arbol de forma jerarquica, toma un puntero al nodo en el arbol y a un integer que representa el nivel actual del nodo en el arbol.

Despues imprime el arbol desde el nodo apuntaro, imprime cada dato del nodo con sangría basada en su nivel en el árbol. durante `printTree(node->firstChild, level + 1);` imprime el nodo actual de firstchild y lo incrementa por un nivel, luego imprime el sibling al igual que el nivel en el que se encuentra.