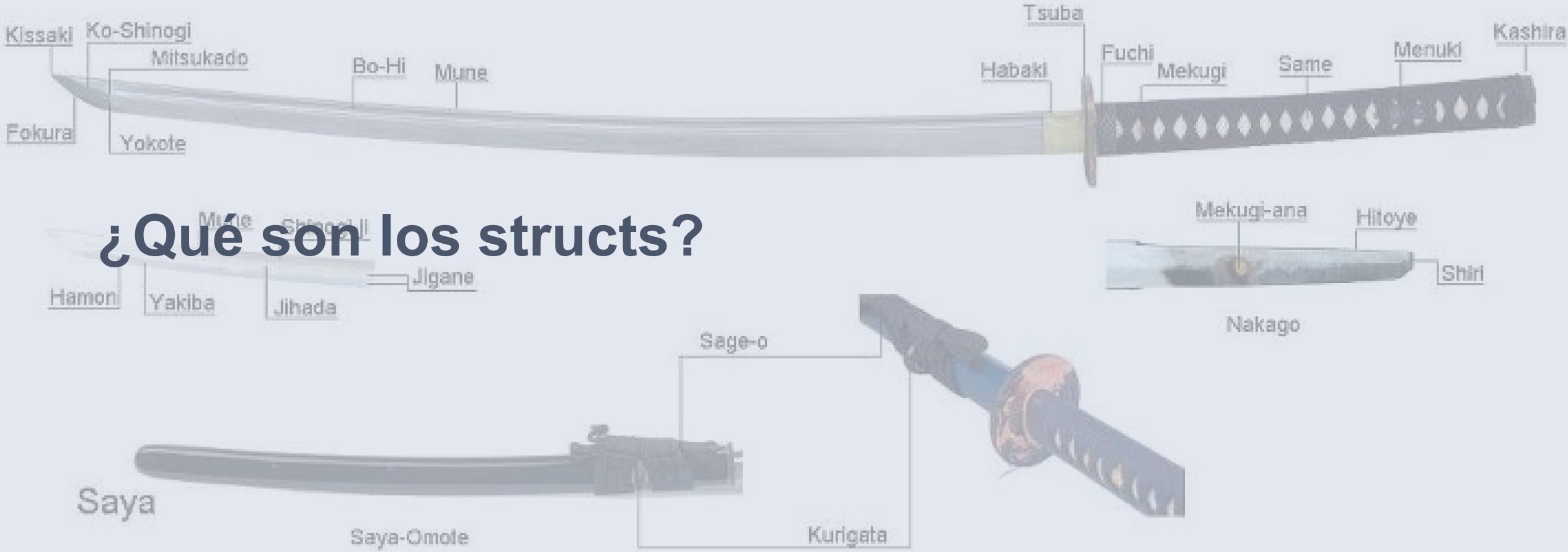
Structs, nodos, listas y pilas

Por Ariel Parra.

¿Qué son los structs?



Son tipos de datos definidos por el usuario que se utilizan para almacenar o agrupar diferentes tipos de datos ordenados. Estos se declaran fuera del main y se pueden asignar valores a las variables del struct desde C++11, pero no se recomienda; las variables creadas a partir de un struct se pueden acceder a travez de punto "." y si se trabaja con punteros de estructuras con el operador flecha "->".

```
struct identificacion{
    string nombre;
    int celular;
} usuario1;

int main(){
    usuario1.nombre="ariel"; usuario1.celular=449;
    identificacion usuario2;
    usuario2.nombre="chisrra"; usuario2.celular=449;
    cout<<usuario1.nombre<<" "<<usuario1.celular<<endl;
    cout<<usuario2.nombre<<" "<<usuario2.celular<<endl;
return 0;
}
```

structs anidados

```
struct carro {
    struct m {
       int modelo;
       int cilindros;
    } motor;
    string color;
};

int main() {
    carro ferrari;
    ferrari.motor.modelo=1;
    ferrari.motor.cilindros=6;
    ferrari.color="rojo";
    cout<<ferrari.motor.modelo<<
    " " <<ferrari.motor.cilindros<<
    " " <<ferrari.color<<endl;
return 0;
}
```

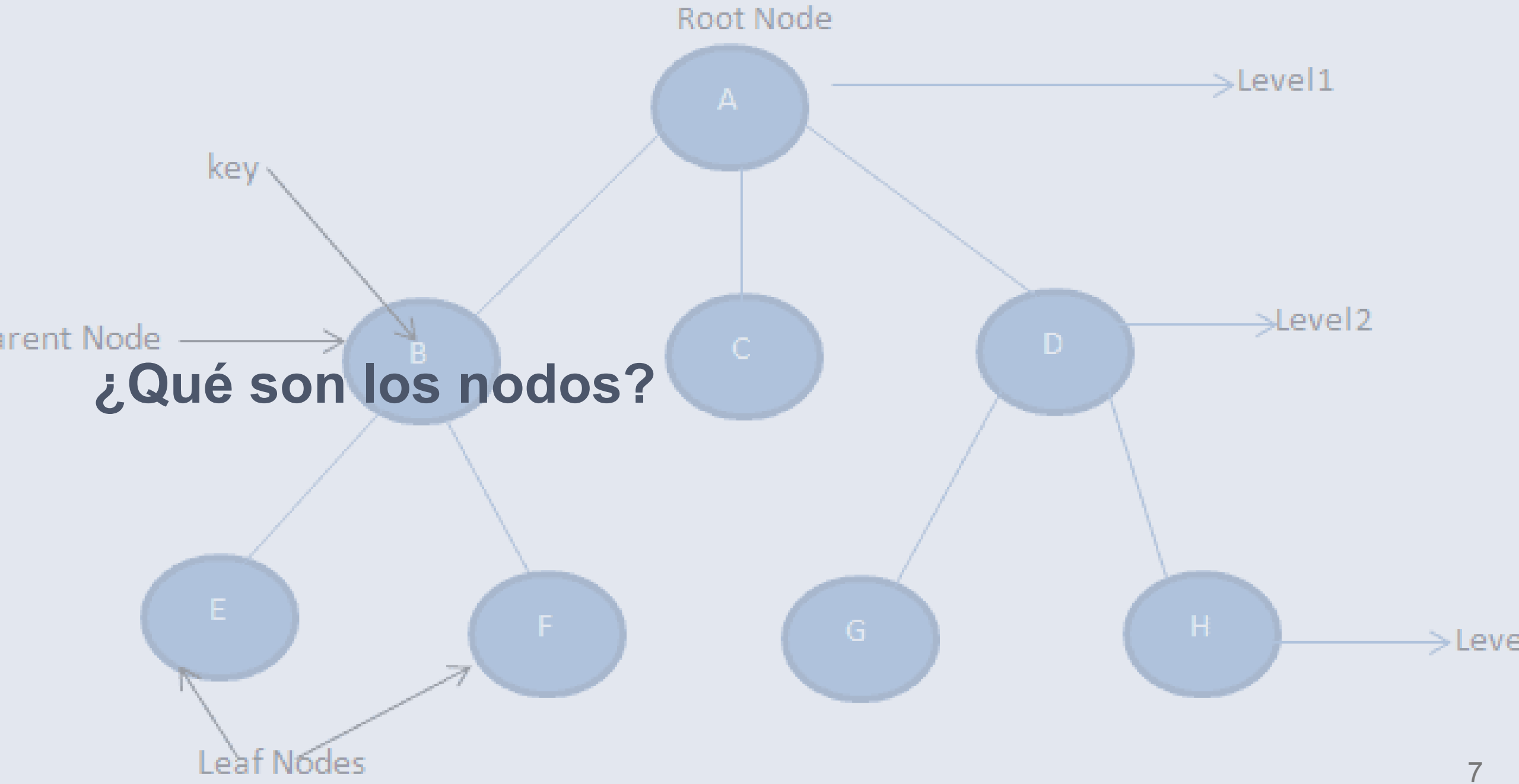
punteros de structs

```
struct Sptr {
    int x, y;
};

int main(){
    Sptr p1 = { 1, 2 };

    Sptr* p2 = &p1;

    cout<< p2->x <<" "<< p2->y;
return 0;
}
```

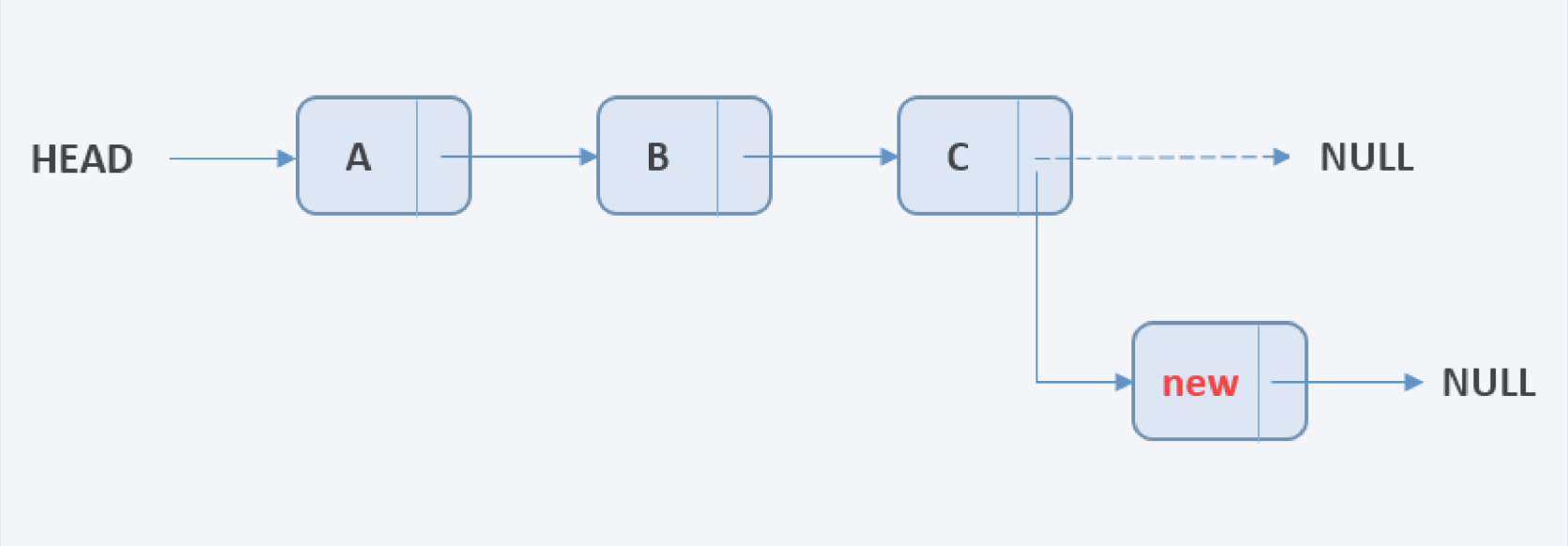
structs con memoria dinamica

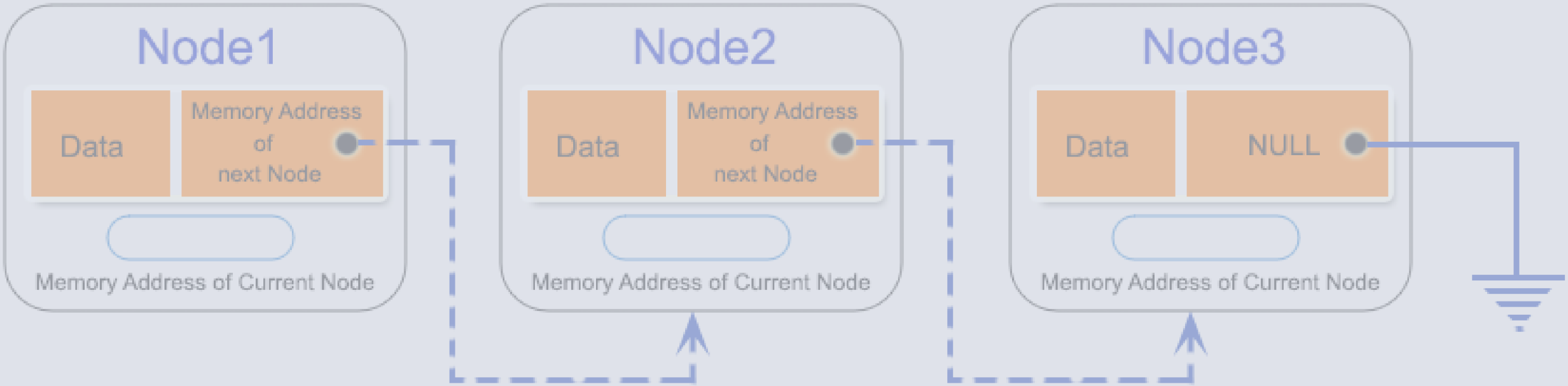
```
struct id{
    string nombre;
};
int main(){
    id *escuela=new id[2];
    vector <id> vec (2);
    escuela[0].nombre="ariel";
    vec[0].nombre=escuela[0].nombre;
    escuela[1].nombre="chisrra";
    vec[1].nombre=escuela[1].nombre;
    delete []escuela;
return 0;
```

¿Qué son los nodos?

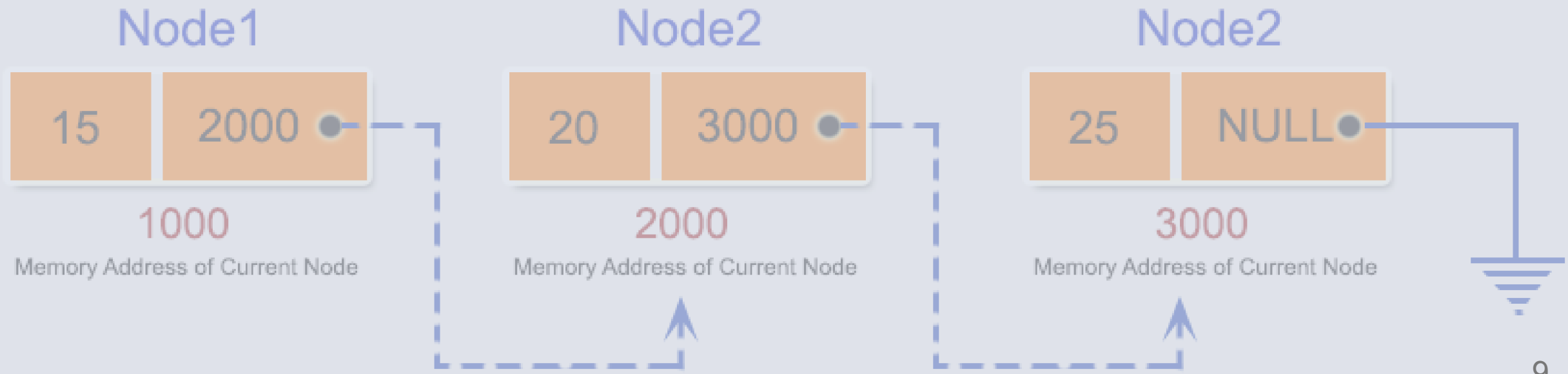


Un nodo es una unidad básica de una estructura de datos, como los arboles y las listas. Los nodos contienen datos y también pueden vincularse con otros nodos. Estos se crean atravez de structs y pueden ser enlazados con punteros.





¿Qué son las listas?



Las listas son estructuras en secuencia que permiten la asignación de memoria no contigua. los algoritmos de inserción e eliminación son rápidas O(1). Normalmente, cuando se habla de listas se habla de una lista doblemente ligada. A estas les podemos meter y eliminar elementos por delante y por detras, saber su longitud, darnos iteradores de posicion, etc.

lista simplemente ligada

```
struct Nodo{
    int dato;
    Nodo* siguiente;
};
int main(){
    Nodo primer=new Nodo;
    Nodo Segundo=new Nodo;
    primer->dato=1;
    primer->siguiente=segundo;
    segundo->dato=1;
    segundo->siguiente=NULL; // ó =nullptr;
    delete primer;
    delete segundo;
return 0;
}
```

lista doblemente ligada

```
struct Nodo{
    int dato;
    Nodo* anterior;
    Nodo* siguiente;
};
int main(){
    Nodo primer=new Nodo;
    Nodo Segundo=new Nodo;
    primer->dato=1;
    primer->anterior=NULL;
    primer->siguiente=segundo;
    segundo->dato=1;
    primer->anterior=primer;
    segundo->siguiente=NULL;
    delete primer;
    delete segundo;

return 0;
}
```

lista circular

```
struct Nodo{
    int dato;
    Nodo* anterior;
    Nodo* siguiente;
};
int main(){
    Nodo primer=new Nodo;
    Nodo Segundo=new Nodo;
    primer->dato=1;
    primer->anterior=segundo;
    primer->siguiente=segundo;
    segundo->dato=1;
    primer->anterior=primer;
    segundo->siguiente=primera;
    delete primer;
    delete segundo;
return 0;
}
```

Funciones de listas STD

Implentar manualmente las funciones de listas es un trabajo tedioso y complejo por lo que el uso de listas STD incluidas en la libreria <list> nos facilitaran la vida.

Funcion	Descripcion
front()	Retorna el valor del primer elemento en la lista.
back()	Retorna el valor del último elemento en la lista.
push_front()	Añade un nuevo elemento al comienzo de la lista.
push_back()	Añade un nuevo elemento al final de la lista.

Funcion	Descripcion
pop_front()	Elimina el primer elemento de la lista.
pop_back()	Elimina el último elemento de la lista.
insert()	Inserta antes del elemento en una posición especificada.
tamaño()	Retorna el número de elementos en la lista.
start()	Retorna un iterator señalando el primer elemento de la lista.
end()	Retorna iterador señalando el elemento que sigue al último.

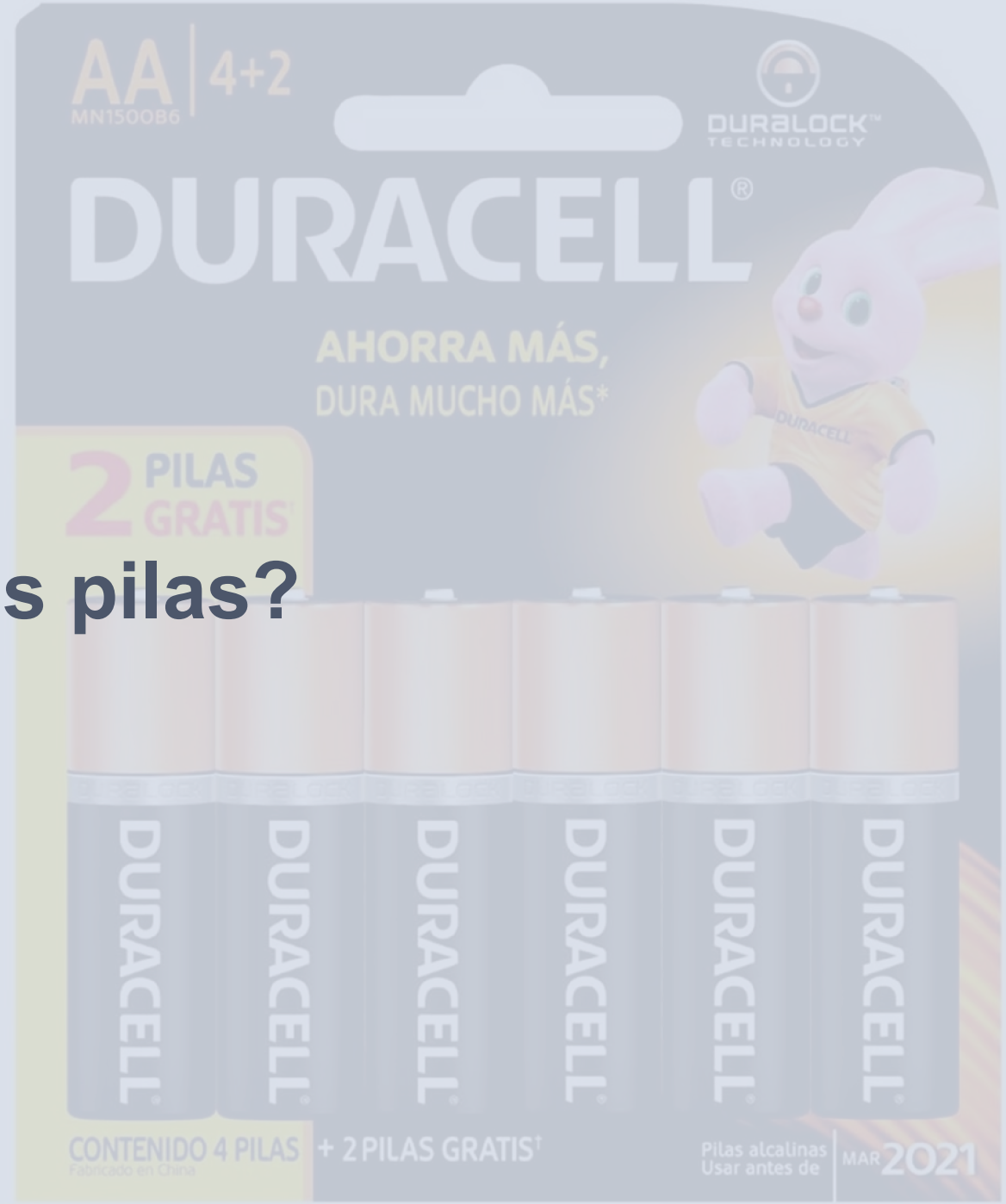
```
void showlist(list<int> g){
    list<int>::iterator it;
    for (it = g.begin(); it != g.end(); ++it)
        cout << '\t' << *it;
    cout<<endl;
}

int main(){
    list<int> gqlist1, gqlist2;
    for (int i = 0; i < 10; ++i) {
        gqlist1.push_back(i * 2);
        gqlist2.push_front(i * 3);
    }
    cout << "\nList 1 (gqlist1) is : ";
    showlist(gqlist1);
    cout << "\nList 2 (gqlist2) is : ";
    showlist(gqlist2);
```

```
    cout << "\ngqlist1.front() : " << gqlist1.front();
    cout << "\ngqlist1.back() : " << gqlist1.back();
    cout << "\ngqlist1.pop_front() : ";
    gqlist1.pop_front();
    showlist(gqlist1);
    cout << "\ngqlist2.pop_back() : ";
    gqlist2.pop_back();
    showlist(gqlist2);
    cout << "\ngqlist1.reverse() : ";
    gqlist1.reverse();
    showlist(gqlist1);
    cout << "\ngqlist2.sort(): ";
    gqlist2.sort();
    showlist(gqlist2);
return 0;
}
```

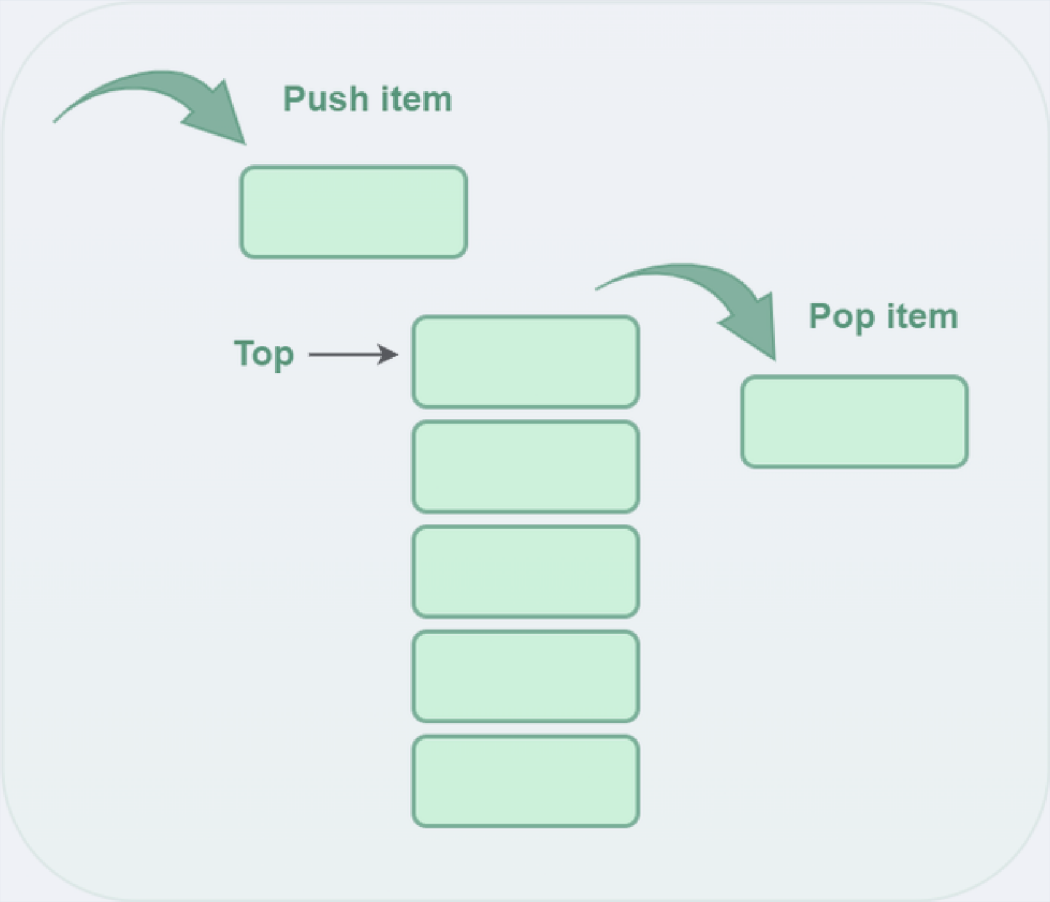
output

```
List 1 (gqlist1) is :   0    2    4    6    8    10    12    14    16    18

List 2 (gqlist2) is :   27    24    21    18    15    12    9    6    3    0

gqlist1.front() :       0
gqlist1.back() :        18
gqlist1.pop_front() :   2    4    6    8    10    12    14    16    18

gqlist2.pop_back() :    27    24    21    18    15    12    9    6    3

gqlist1.reverse() :     18    16    14    12    10    8    6    4    2

gqlist2.sort():         3    6    9    12    15    18    21    24    27
```

¿Qué son las pilas?



Las pilas o en Ingles Stack es una estructura lineal de tipo LIFO (Last Input First Output), esto significa que la inserción de un nuevo elemento y la eliminación de un elemento existente tiene lugar en el mismo extremo representado como la parte superior de la pila. Donde sus funciones basicas de push,pop,isEmpty y size tienen una complejidad lineal O(1).



Estructura de stack

```
struct Nodo {
    int dato;
    struct Nodo* siguiente;
} Nodo;

struct Pila {
    Nodo* tope;
} Pila;

Pila* crearPila() {
    Pila* nuevaPila = (Pila*)malloc(sizeof(Pila));
    nuevaPila->tope = NULL;
    return nuevaPila;
}
```

Funciones de Stacks STD

Funcion	Descripcion
empty()	Retorna si la pila está vacía
size()	Retorna el tamaño de la pila
top()	Retorna una referencia al elemento más alto de la pila
push()	Añade el elemento en la parte superior de la pila
pop()	Elimina el elemento introducido más reciente de la pila

```
int main() {
    stack<int> pila;
    pila.push(21);
    pila.push(22);
    pila.push(24);
    pila.push(25);
    int num=0;
    pila.push(num);
    pila.pop();
    pila.pop();
    pila.pop();
    while (!pila.empty()) {
        cout << pila.top() <<" ";
        pila.pop();
    }
return 0;
}
```

Referencias

https://www.geeksforgeeks.org/structures-in-cpp/
https://www.w3schools.com/cpp/cpp_structs.asp
https://en.wikipedia.org/wiki/Node_(computer_science)
https://www.geeksforgeeks.org/list-cpp-stl/
https://www.geeksforgeeks.org/list-of-stacks-in-c-stl/
https://www.geeksforgeeks.org/data-structures/linked-list/
https://www.geeksforgeeks.org/stack-in-cpp-stl/
https://www.geeksforgeeks.org/introduction-to-stack-data-structure-and-algorithm-tutorials/