PRÁCTICA/LABORATORIO Nº 03

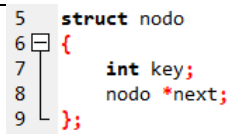
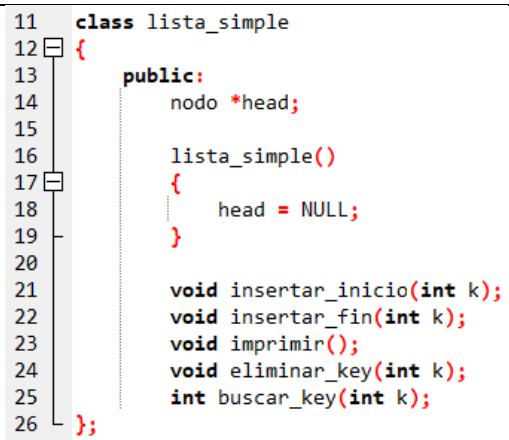
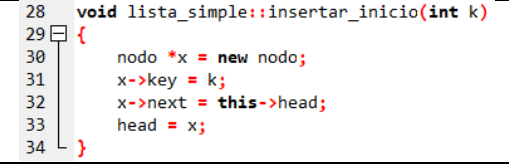
LISTAS SIMPLEMENTE ENLAZADAS

Objetivos:

- Utilizar Dev-C++
- Implementar estructuras (**nodo**) y clases (**lista_simple,lista_simple_ordenada**).
- Resolver problemas utilizando listas simplemente enlazadas.

0. Ejercicios con listas simplemente enlazadas

A continuación, se presentan y describen fragmentos del archivo **lista_simple.hpp**:

Código	Descripción
Estructura "nodo":	
	La estructura **nodo** solo contiene la clave (**key**) y el **puntero** al siguiente elemento.
Clase "lista_simple"	
	La clase **lista_simple**, es en la que implementaremos los **métodos** de esta. Como atributos de la **lista_simple**, solo tenemos el **nodo cabeza (head)** de la lista simplemente enlazada.
Función "insertar_inicio" de la clase "lista_simple"	
	Es un **método** de la clase **lista_simple** que permite **crear un nodo, cargarlo de información** e **insertarlo al inicio** de la lista simplemente enlazada.

Función "insertar_fin" de la clase "lista_simple"

```
36  void lista_simple::insertar_fin(int k)
37  {
38      if(this->head == NULL)
39      {
40          this->insertar_inicio(k);
41      }
42      else
43      {
44          nodo *x = this->head;
45          while(x->next != NULL)
46          {
47              x = x->next;
48          }
49          nodo *y = new nodo;
50          y->key = k;
51          y->next = x->next;
52          x->next = y;
53      }
54  }
```

Es un **método** de la clase **lista_simple** que permite **crear un nodo, cargarlo de información** e **insertarlo al final** de la lista simplemente enlazada.

Función "imprimir" de la clase "lista_simple"

```
56  void lista_simple::imprimir()
57  {
58      nodo *x = this->head;
59      int count = 0;
60      while(x != NULL)
61      {
62          cout<<x->key<<' ';
63          x = x->next;
64          count++;
65      }
66      cout<<endl;
67  }
```

Es un **método** de la clase **lista_simple** que permite **imprimir** los elementos de la lista simplemente enlazada separados por un espacio.

Función "eliminar_key" de la clase "lista_simple"

```
69  void lista_simple::eliminar_key(int k)
70  {
71      if(this->head != NULL)
72      {
73          nodo *x = this->head;
74          if(x->key == k)
75          {
76              this->head = this->head->next;
77              delete x;
78          }
79          else
80          {
81              while(x->next != NULL and x->next->key != k)
82              {
83                  x = x->next;
84              }
85              if(x->next != NULL)
86              {
87                  nodo *y = x->next;
88                  x->next = x->next->next;
89                  delete y;
90              }
91          }
92      }
93  }
```

Es un **método** de la clase **lista_simple** que permite **eliminar** la primera aparición de un nodo con key = **k** de la lista simplemente enlazada.

Función "buscar_key" de la clase "lista_simple"

```
95  int lista_simple::buscar_key(int k)
96  {
97      nodo *x = this->head;
98      while(x != NULL and x->key != k)
99      {
100         x = x->next;
101     }
102     if(x != NULL)
103     {
104         return 1;
105     }
106     else
107     {
108         return 0;
109     }
110 }
```

Es un **método** de la clase **lista_simple** que permite **buscar** un nodo con key = **k** de la lista simplemente enlazada.

Devuelve 1 si lo encuentra, caso contrario devuelve 0.

A continuación, se presenta y describe el código del archivo **main.cpp**:

Función main()

```
1   #include<iostream>
2   #include "lista_simple.hpp"
3
4   using namespace std;
5
6   int main()
7   {
8       lista_simple lista;
9       lista.insertar_inicio(55);
10      lista.imprimir(); //55
11      lista.insertar_inicio(44);
12      lista.imprimir(); //44 55
13      lista.insertar_inicio(33);
14      lista.imprimir(); //33 44 55
15      lista.insertar_inicio(22);
16      lista.imprimir(); //22 33 44 55
17      lista.insertar_inicio(11);
18      lista.imprimir(); //11 22 33 44 55
19
20      lista.eliminar_key(11);
21      lista.imprimir(); //22 33 44 55
22      lista.eliminar_key(33);
23      lista.imprimir(); //22 44 55
24      lista.eliminar_key(55);
25      lista.imprimir(); //22 44
26
27      cout<<lista.buscar_key(22)<<endl; //1
28      cout<<lista.buscar_key(33)<<endl; //0
29
30      lista.insertar_fin(66);
31      lista.imprimir(); //22 44 66
32  }
```

Es la función del archivo **main.cpp** en la que probamos nuestra clase **lista_simple**.

1. Implementar en el archivo lista_simple.hpp los siguientes métodos:

- insertar_antes_de_key(int k) //inserta un nodo anterior a key, key está en lista.
- insertar_despues_de_key(int k) // inserta un nodo posterior a key, key está en lista.
- eliminar_inicio() //elimina el primer elemento de la lista.
- eliminar_fin() //elimina el último elemento de la lista.
- eliminar_antes_de_key(int k) // elimina el nodo anterior a key, key está en lista.
- eliminar_despues_de_key(int k) // elimina el nodo posterior a key, key está en lista.

2. Implementar en un NUEVO archivo lista_simple_ordenada.hpp la clase lista_simple_ordenada que mantenga en una lista simple el orden de nodos que almacenan un entero key:

- insertar(int k) //considerar que se debe mantener el orden ascendente.
- eliminar(int k) //considerar que puede ser el primer nodo
- buscar(int k) //considerar que la lista está ordenada
- imprimir() //imprimir los nodo en orden ascendente
- imprimir_descendente() //imprimir los nodo de forma descendente (usar memoria auxiliar)
- sumar () //sumar todos nodos
- sumar_posiciones_par() //sumar todos cuya posición sea par, se enumera desde 0.
- invertir() //invertir el orden de los NODOS (luego de esto, solo funcionará imprimir()).