



Act 2.2 - Verificación de las funcionalidades de una estructura de datos lineal

Programación de estructuras de datos y algoritmos fundamentales (Gpo 4)

Alumno:

Thomas Freund Paternostro //A00831997

Fecha de entrega:

30/09/2021

Introducción

Se trabajó con la estructura de datos tipo linked list, donde se lee el primer nodo como nulo y al resto se le da un valor. En este caso es un link list simple dado que solo sabe cuál es el valor siguiente pero no sabe cuál fue el valor previo. Se va a definir uno de los conceptos principales con los cuales se trabajó a lo largo del programa.

Linked List(Listas Enlazadas):

Los "Linkedlist" son estructuras de datos lineales, donde sus elementos no están almacenados en bloques contiguos de memoria, a diferencia de los array, que estos son almacenados de bloques contiguos de memoria, para entender mucho mejor vea la siguiente imagen (Medina, 2018)."

Las listas enlazadas son almacenadas en diferentes sectores de la memoria y hace referencia a sus elementos mediante punteros.

Templates (plantilla)

Se crearon templates o plantillas para regresar un tipo de dato donde el enfoque fue crear funciones CRUD (create, read, update, delete/destroy). Una plantilla por definición es "una manera especial de escribir funciones y clases para que estas puedan ser usadas con cualquier tipo de dato, similar a la sobrecarga, en el caso de las funciones, pero evitando el trabajo de escribir cada versión de la función (Hernández, 2017)."

```
1  template<class T>
2  class Node{
3  private:
4      T data; //
5      Node<T> *next; //
6  public:
7      Node(T data);
8      Node(T data, Node<T> *next);
9      T getData();
10     Node<T>* getNext();
11     void setData(T data);
12     void setNext(Node<T>* next); //variable void
13 };
```

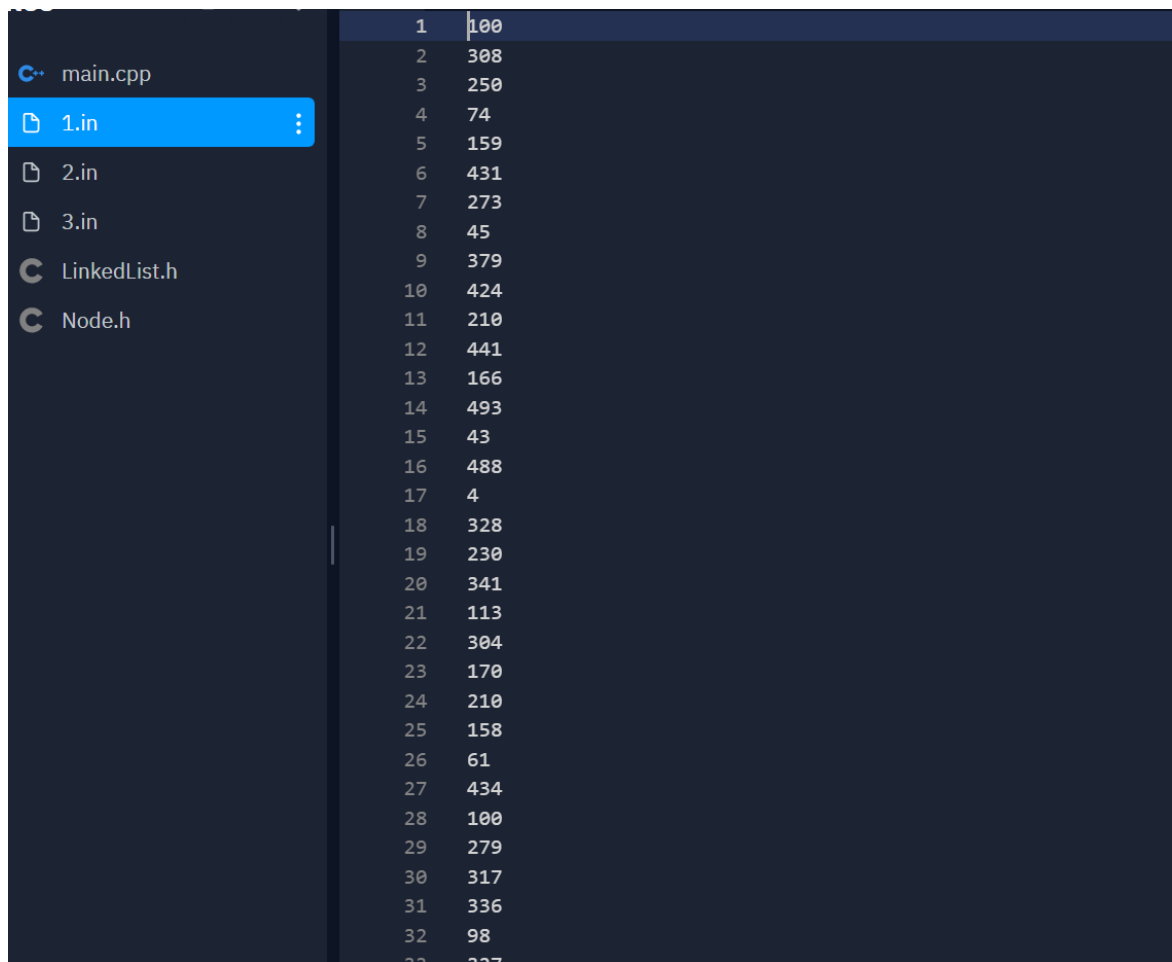
Ejemplo de Template usado en el código

CRUD

Son las funciones principales para trabajar y manipular datos con cualquier estructura de datos, dependiendo de las condiciones que esta tenga. Como fue previamente mencionado las siglas corresponden a **C** create, **R** read, **U** update y **D** delete/destroy.

Casos de prueba utilizadas y razones por las cuales esas pruebas se eligieron

Se trabajó con tres archivos tipo .in para hacer los casos de prueba, estos son los archivos dados en las actividades previas para hacer los casos de prueba:



```
1 100
2 308
3 250
4 74
5 159
6 431
7 273
8 45
9 379
10 424
11 210
12 441
13 166
14 493
15 43
16 488
17 4
18 328
19 230
20 341
21 113
22 304
23 170
24 210
25 158
26 61
27 434
28 100
29 279
30 317
31 336
32 98
33 327
```

1.in

Este primer archivo se utilizó para tener una variedad de números pequeños e integrales de todo tipo. Este archivo tiene 100 elementos

```
1 42
2 34
3 2752
4 -253
5 -1379
6 1814
7 -347
8 190
9 -1061
10 267
11 2677
12 3490
13 564
14 2588
15 812
16 -1426
17 1193
18 -974
19 2644
20 -822
21 2273
22 530
23 3328
24 1501
25 85
26 3239
27 -1150
28 197
29 874
30 1132
31 2789
32 3252
33 407
34 2546
35 3372
36 -1469
37 5
```

2.in

Este segundo archivo tiene números mayores y una variación entre números positivos y negativos.

```
1 4660
2 -741
3 -740
4 -739
5 -738
6 -737
7 -736
8 -735
9 -734
10 -733
11 -732
12 -731
13 -730
14 -729
15 -728
16 -727
17 -726
18 -725
19 -724
20 -723
21 -722
22 -721
23 -720
24 -719
25 -718
26 -717
27 -716
28 -715
29 -714
30 -713
31 -712
32 -711
33 -710
34 -709
35 -708
36 -707
37 -706
38 -705
```

3.in

El tercer archivo tiene números mixtos pero estos tienden a ser mayores y también negativos.

Solo se trabaja con números enteros dado que la lista enlazada espera este tipo de dato.

Resultados de los casos de prueba aplicados al ADT

1) Leer archivo correctamente:

```
❏ clang++-7 -pthread -std=c++17 -o main main.cpp
❏ ./main
Nombre del archivo(Opciones: 1.in, 2.in, 3.in):
2342
Nombre del archivo(Opciones: 1.in, 2.in, 3.in):
324242
Nombre del archivo(Opciones: 1.in, 2.in, 3.in):
432
Nombre del archivo(Opciones: 1.in, 2.in, 3.in):
4234
Nombre del archivo(Opciones: 1.in, 2.in, 3.in):
werwer
Nombre del archivo(Opciones: 1.in, 2.in, 3.in):
sr32
Nombre del archivo(Opciones: 1.in, 2.in, 3.in):
4.in
Nombre del archivo(Opciones: 1.in, 2.in, 3.in):
█
```

Cuando el archivo correcto es introducido se lo lee y se genera el linkedlist y un menu con opciones

```

Nombre del archivo(Opciones: 1.in, 2.in, 3.in):
4.in
Nombre del archivo(Opciones: 1.in, 2.in, 3.in):
1.in
El archivo tiene los siguientes elementos:
1) 100, 2) 308, 3) 250, 4) 74, 5) 159, 6) 431, 7) 273, 8) 45, 9) 379, 10) 424, 11) 210, 12) 441, 13) 166,
14) 493, 15) 43, 16) 488, 17) 4, 18) 328, 19) 230, 20) 341, 21) 113, 22) 304, 23) 170, 24) 210, 25) 15,
26) 61, 27) 434, 28) 100, 29) 279, 30) 317, 31) 336, 32) 98, 33) 327, 34) 13, 35) 268, 36) 311, 37) 1,
38) 480, 39) 150, 40) 80, 41) 322, 42) 468, 43) 173, 44) 394, 45) 337, 46) 486, 47) 246, 48) 229, 49)
92, 50) 195, 51) 358, 52) 2, 53) 154, 54) 209, 55) 445, 56) 169, 57) 491, 58) 125, 59) 197, 60) 31, 61)
404, 62) 223, 63) 167, 64) 50, 65) 25, 66) 302, 67) 354, 68) 478, 69) 409, 70) 229, 71) 434, 72) 299,
73) 482, 74) 136, 75) 14, 76) 366, 77) 315, 78) 64, 79) 37, 80) 426, 81) 170, 82) 116, 83) 95, 84) 130,
85) 2, 86) 18, 87) 196, 88) 106, 89) 405, 90) 452, 91) 299, 92) 189, 93) 124, 94) 6, 95) 383, 96) 253, 97)
67, 98) 217, 99) 338, 100) 439, 101) 145,
-----MENU-----
1) Create
2) Read
3) Update
4) Delete
5) Salir

```

2) En caso de que se introduzca un valor que no está en el menú este no será reconocido y se pedirá de nuevo una opción válida

```

Nombre del archivo(Opciones: 1.in, 2.in, 3.in):
1.in
El archivo tiene los siguientes elementos:
1) 100, 2) 308, 3) 250, 4) 74, 5) 159, 6) 431, 7) 273, 8) 45, 9) 379, 10) 424, 11) 210, 12) 441, 13) 166,
14) 493, 15) 43, 16) 488, 17) 4, 18) 328, 19) 230, 20) 341, 21) 113, 22) 304, 23) 170, 24) 210, 25) 15,
26) 61, 27) 434, 28) 100, 29) 279, 30) 317, 31) 336, 32) 98, 33) 327, 34) 13, 35) 268, 36) 311, 37) 1,
38) 480, 39) 150, 40) 80, 41) 322, 42) 468, 43) 173, 44) 394, 45) 337, 46) 486, 47) 246, 48) 229, 49)
92, 50) 195, 51) 358, 52) 2, 53) 154, 54) 209, 55) 445, 56) 169, 57) 491, 58) 125, 59) 197, 60) 31, 61)
404, 62) 223, 63) 167, 64) 50, 65) 25, 66) 302, 67) 354, 68) 478, 69) 409, 70) 229, 71) 434, 72) 299,
73) 482, 74) 136, 75) 14, 76) 366, 77) 315, 78) 64, 79) 37, 80) 426, 81) 170, 82) 116, 83) 95, 84) 130,
85) 2, 86) 18, 87) 196, 88) 106, 89) 405, 90) 452, 91) 299, 92) 189, 93) 124, 94) 6, 95) 383, 96) 253, 97)
67, 98) 217, 99) 338, 100) 439, 101) 145,
-----MENU-----
1) Create
2) Read
3) Update
4) Delete
5) Salir
6
El valor no es una opción válida. Digite de nuevo el valor.
-----MENU-----
1) Create
2) Read
3) Update
4) Delete
5) Salir
sdfsd
El valor no es una opción válida. Digite de nuevo el valor.
-----MENU-----
1) Create
2) Read
3) Update
4) Delete
5) Salir

```

3) **C (create)** Se oprime la opción del menú 1 y se pide el valor a crear que en este caso arbitrariamente se decidió el valor 514. De los 101 elementos, se puede observar en la segunda impresión el índice 103) 545

```

El valor no es una opcion valida. Digite denuevo el valor.
-----MENU-----
1) Create
2) Read
3) Update
4) Delete
5) Salir
1
El archivo tiene los siguientes elementos:
1) 100, 2) 308, 3) 250, 4) 74, 5) 159, 6) 431, 7) 273, 8) 45, 9) 379, 10) 424, 11) 210, 12) 441, 13) 166
, 14) 493, 15) 43, 16) 488, 17) 4, 18) 328, 19) 230, 20) 341, 21) 113, 22) 304, 23) 170, 24) 210, 25) 15
8, 26) 61, 27) 434, 28) 100, 29) 279, 30) 317, 31) 336, 32) 98, 33) 327, 34) 13, 35) 268, 36) 311, 37) 1
34, 38) 480, 39) 150, 40) 80, 41) 322, 42) 468, 43) 173, 44) 394, 45) 337, 46) 486, 47) 246, 48) 229, 49
) 92, 50) 195, 51) 358, 52) 2, 53) 154, 54) 209, 55) 445, 56) 169, 57) 491, 58) 125, 59) 197, 60) 31, 61
) 404, 62) 223, 63) 167, 64) 50, 65) 25, 66) 302, 67) 354, 68) 478, 69) 409, 70) 229, 71) 434, 72) 299,
73) 482, 74) 136, 75) 14, 76) 366, 77) 315, 78) 64, 79) 37, 80) 426, 81) 170, 82) 116, 83) 95, 84) 130,
85) 2, 86) 18, 87) 196, 88) 106, 89) 405, 90) 452, 91) 299, 92) 189, 93) 124, 94) 6, 95) 383, 96) 253, 9
7) 67, 98) 217, 99) 338, 100) 439, 101) 145,
Que valor deseas agregar: 543
El archivo tiene los siguientes elementos:
1) 100, 2) 308, 3) 250, 4) 74, 5) 159, 6) 431, 7) 273, 8) 45, 9) 379, 10) 424, 11) 210, 12) 441, 13) 166
, 14) 493, 15) 43, 16) 488, 17) 4, 18) 328, 19) 230, 20) 341, 21) 113, 22) 304, 23) 170, 24) 210, 25) 15
8, 26) 61, 27) 434, 28) 100, 29) 279, 30) 317, 31) 336, 32) 98, 33) 327, 34) 13, 35) 268, 36) 311, 37) 1
34, 38) 480, 39) 150, 40) 80, 41) 322, 42) 468, 43) 173, 44) 394, 45) 337, 46) 486, 47) 246, 48) 229, 49
) 92, 50) 195, 51) 358, 52) 2, 53) 154, 54) 209, 55) 445, 56) 169, 57) 491, 58) 125, 59) 197, 60) 31, 61
) 404, 62) 223, 63) 167, 64) 50, 65) 25, 66) 302, 67) 354, 68) 478, 69) 409, 70) 229, 71) 434, 72) 299,
73) 482, 74) 136, 75) 14, 76) 366, 77) 315, 78) 64, 79) 37, 80) 426, 81) 170, 82) 116, 83) 95, 84) 130,
85) 2, 86) 18, 87) 196, 88) 106, 89) 405, 90) 452, 91) 299, 92) 189, 93) 124, 94) 6, 95) 383, 96) 253, 9
7) 67, 98) 217, 99) 338, 100) 439, 101) 145, 102) 543,

```

4) R read, se oprime la opción del menú 2 y se pide el número que desea ser solicitado. Arbitrariamente se puso el número 2 y se leyó la lista enlazada y se lo encontró y para demostrar que funciona cuando no existe el valor en la lista en el segundo caso se puso 32342 que claramente no está y sale que el elemento no fue encontrado.

```

1) Create
2) Read
3) Update
4) Delete
5) Salir
2
El archivo tiene los siguientes elementos:
1) 100, 2) 308, 3) 250, 4) 74, 5) 159, 6) 431, 7) 273, 8) 45, 9) 379, 10) 424, 11) 210, 12) 441, 13) 166
, 14) 493, 15) 43, 16) 488, 17) 4, 18) 328, 19) 230, 20) 341, 21) 113, 22) 304, 23) 170, 24) 210, 25) 15
8, 26) 61, 27) 434, 28) 100, 29) 279, 30) 317, 31) 336, 32) 98, 33) 327, 34) 13, 35) 268, 36) 311, 37) 1
34, 38) 480, 39) 150, 40) 80, 41) 322, 42) 468, 43) 173, 44) 394, 45) 337, 46) 486, 47) 246, 48) 229, 49
) 92, 50) 195, 51) 358, 52) 2, 53) 154, 54) 209, 55) 445, 56) 169, 57) 491, 58) 125, 59) 197, 60) 31, 61
) 404, 62) 223, 63) 167, 64) 50, 65) 25, 66) 302, 67) 354, 68) 478, 69) 409, 70) 229, 71) 434, 72) 299,
73) 482, 74) 136, 75) 14, 76) 366, 77) 315, 78) 64, 79) 37, 80) 426, 81) 170, 82) 116, 83) 95, 84) 130,
85) 2, 86) 18, 87) 196, 88) 106, 89) 405, 90) 452, 91) 299, 92) 189, 93) 124, 94) 6, 95) 383, 96) 253, 9
7) 67, 98) 217, 99) 338, 100) 439, 101) 145, 102) 543,
Cual es el numero que deseas buscar:
2
Elemento encontrado!

-----MENU-----
1) Create
2) Read
3) Update
4) Delete
5) Salir
2
El archivo tiene los siguientes elementos:
1) 100, 2) 308, 3) 250, 4) 74, 5) 159, 6) 431, 7) 273, 8) 45, 9) 379, 10) 424, 11) 210, 12) 441, 13) 166
, 14) 493, 15) 43, 16) 488, 17) 4, 18) 328, 19) 230, 20) 341, 21) 113, 22) 304, 23) 170, 24) 210, 25) 15
8, 26) 61, 27) 434, 28) 100, 29) 279, 30) 317, 31) 336, 32) 98, 33) 327, 34) 13, 35) 268, 36) 311, 37) 1
34, 38) 480, 39) 150, 40) 80, 41) 322, 42) 468, 43) 173, 44) 394, 45) 337, 46) 486, 47) 246, 48) 229, 49
) 92, 50) 195, 51) 358, 52) 2, 53) 154, 54) 209, 55) 445, 56) 169, 57) 491, 58) 125, 59) 197, 60) 31, 61
) 404, 62) 223, 63) 167, 64) 50, 65) 25, 66) 302, 67) 354, 68) 478, 69) 409, 70) 229, 71) 434, 72) 299,
73) 482, 74) 136, 75) 14, 76) 366, 77) 315, 78) 64, 79) 37, 80) 426, 81) 170, 82) 116, 83) 95, 84) 130,
85) 2, 86) 18, 87) 196, 88) 106, 89) 405, 90) 452, 91) 299, 92) 189, 93) 124, 94) 6, 95) 383, 96) 253, 9
7) 67, 98) 217, 99) 338, 100) 439, 101) 145, 102) 543,
Cual es el numero que deseas buscar:
32342
El elemento no fue encontrado.

```

5) U update, se digita la opción 3 donde se pide el valor que se desea actualizar y para trabajar con el valor previamente dado se digito 542 en el índice 102 (empezando de 1). Este valor se sustituye por 69 como se evidencia en la imagen inferior.


```

-----MENU-----
1) Create
2) Read
3) Update
4) Delete
5) Salir
3
El archivo tiene los siguientes elementos:
1) 100, 2) 308, 3) 250, 4) 74, 5) 159, 6) 431, 7) 273, 8) 45, 9) 379, 10) 424, 11) 210, 12) 441, 13) 166
, 14) 493, 15) 43, 16) 488, 17) 4, 18) 328, 19) 230, 20) 341, 21) 113, 22) 304, 23) 170, 24) 210, 25) 15
8, 26) 61, 27) 434, 28) 100, 29) 279, 30) 317, 31) 336, 32) 98, 33) 327, 34) 13, 35) 268, 36) 311, 37) 1
34, 38) 480, 39) 150, 40) 80, 41) 322, 42) 468, 43) 173, 44) 394, 45) 337, 46) 486, 47) 246, 48) 229, 49
) 92, 50) 195, 51) 358, 52) 2, 53) 154, 54) 209, 55) 445, 56) 169, 57) 491, 58) 125, 59) 197, 60) 31, 61
) 404, 62) 223, 63) 167, 64) 50, 65) 25, 66) 302, 67) 354, 68) 478, 69) 409, 70) 229, 71) 434, 72) 299,
73) 482, 74) 136, 75) 14, 76) 366, 77) 315, 78) 64, 79) 37, 80) 426, 81) 170, 82) 116, 83) 95, 84) 130,
85) 2, 86) 18, 87) 196, 88) 106, 89) 405, 90) 452, 91) 299, 92) 189, 93) 124, 94) 6, 95) 383, 96) 253, 9
7) 67, 98) 217, 99) 338, 100) 439, 101) 145, 102) 543,
Que valor desea actualizar?:
543

Valor encontrado, por cual valor lo desea sustituir?
69
El archivo tiene los siguientes elementos:
1) 100, 2) 308, 3) 250, 4) 74, 5) 159, 6) 431, 7) 273, 8) 45, 9) 379, 10) 424, 11) 210, 12) 441, 13) 166
, 14) 493, 15) 43, 16) 488, 17) 4, 18) 328, 19) 230, 20) 341, 21) 113, 22) 304, 23) 170, 24) 210, 25) 15
8, 26) 61, 27) 434, 28) 100, 29) 279, 30) 317, 31) 336, 32) 98, 33) 327, 34) 13, 35) 268, 36) 311, 37) 1
34, 38) 480, 39) 150, 40) 80, 41) 322, 42) 468, 43) 173, 44) 394, 45) 337, 46) 486, 47) 246, 48) 229, 49
) 92, 50) 195, 51) 358, 52) 2, 53) 154, 54) 209, 55) 445, 56) 169, 57) 491, 58) 125, 59) 197, 60) 31, 61
) 404, 62) 223, 63) 167, 64) 50, 65) 25, 66) 302, 67) 354, 68) 478, 69) 409, 70) 229, 71) 434, 72) 299,
73) 482, 74) 136, 75) 14, 76) 366, 77) 315, 78) 64, 79) 37, 80) 426, 81) 170, 82) 116, 83) 95, 84) 130,
85) 2, 86) 18, 87) 196, 88) 106, 89) 405, 90) 452, 91) 299, 92) 189, 93) 124, 94) 6, 95) 383, 96) 253, 9
7) 67, 98) 217, 99) 338, 100) 439, 101) 145, 102) 69,

```

6) **D** delete, se teclea la opción 4 donde se pide el valor que se desea eliminar. Se escogió arbitrariamente 308 que está en el índice 2. Se puede observar en la parte inferior que el nuevo valor en este índice es 250 que estaba en la tercera posición y que todos los valores de índice retrocedieron uno indicando que la lista entrelazada fue reducida.

```

-----MENU-----
1) Create
2) Read
3) Update
4) Delete
5) Salir
4
El archivo tiene los siguientes elementos:
1) 100, 2) 308, 3) 250, 4) 74, 5) 159, 6) 431, 7) 273, 8) 45, 9) 379, 10) 424, 11) 210, 12) 441, 13) 166
, 14) 493, 15) 43, 16) 488, 17) 4, 18) 328, 19) 230, 20) 341, 21) 113, 22) 304, 23) 170, 24) 210, 25) 15
8, 26) 61, 27) 434, 28) 100, 29) 279, 30) 317, 31) 336, 32) 98, 33) 327, 34) 13, 35) 268, 36) 311, 37) 1
34, 38) 480, 39) 150, 40) 80, 41) 322, 42) 468, 43) 173, 44) 394, 45) 337, 46) 486, 47) 246, 48) 229, 49
) 92, 50) 195, 51) 358, 52) 2, 53) 154, 54) 209, 55) 445, 56) 169, 57) 491, 58) 125, 59) 197, 60) 31, 61
) 404, 62) 223, 63) 167, 64) 50, 65) 25, 66) 302, 67) 354, 68) 478, 69) 409, 70) 229, 71) 434, 72) 299,
73) 482, 74) 136, 75) 14, 76) 366, 77) 315, 78) 64, 79) 37, 80) 426, 81) 170, 82) 116, 83) 95, 84) 130,
85) 2, 86) 18, 87) 196, 88) 106, 89) 405, 90) 452, 91) 299, 92) 189, 93) 124, 94) 6, 95) 383, 96) 253, 9
7) 67, 98) 217, 99) 338, 100) 439, 101) 145, 102) 69,

Que valor de la lista deseas eliminar
308
Elemento encontrado y eliminado.
La nueva lista sin el valor 308 es:
El archivo tiene los siguientes elementos:
1) 100, 2) 250, 3) 74, 4) 159, 5) 431, 6) 273, 7) 45, 8) 379, 9) 424, 10) 210, 11) 441, 12) 166, 13) 493
, 14) 43, 15) 488, 16) 4, 17) 328, 18) 230, 19) 341, 20) 113, 21) 304, 22) 170, 23) 210, 24) 158, 25) 61
, 26) 434, 27) 100, 28) 279, 29) 317, 30) 336, 31) 98, 32) 327, 33) 13, 34) 268, 35) 311, 36) 134, 37) 4
80, 38) 150, 39) 80, 40) 322, 41) 468, 42) 173, 43) 394, 44) 337, 45) 486, 46) 246, 47) 229, 48) 92, 49)
195, 50) 358, 51) 2, 52) 154, 53) 209, 54) 445, 55) 169, 56) 491, 57) 125, 58) 197, 59) 31, 60) 404, 61
) 223, 62) 167, 63) 50, 64) 25, 65) 302, 66) 354, 67) 478, 68) 409, 69) 229, 70) 434, 71) 299, 72) 482,
73) 136, 74) 14, 75) 366, 76) 315, 77) 64, 78) 37, 79) 426, 80) 170, 81) 116, 82) 95, 83) 130, 84) 2, 85
) 18, 86) 196, 87) 106, 88) 405, 89) 452, 90) 299, 91) 189, 92) 124, 93) 6, 94) 383, 95) 253, 96) 67, 97
) 217, 98) 338, 99) 439, 100) 145, 101) 69,

```

Sección de la calidad de software del ADT

- Identación

4 ejemplos que evidencian buenas prácticas de indentación

1) Main

```
3 //Implementación de un ADT de estructura de datos lineales
4
5 #include <iostream>
6 #include <fstream>
7 #include <string>
8 #include "LinkedList.h"
9
10 using namespace std;
11
12 void create(int data ,LinkedList<int> list);
13 void update(int data ,LinkedList<int> list,int updateValue );
14 void read(int data ,LinkedList<int> list);
15 void del(int data ,LinkedList<int> list);
16
17
18
19 int main(){
20
21     //crear estructura linkedlist de data tipo int
22     LinkedList<int> list;
23     int data, index; // variable data y index tipo int
24     int updateValue; // valor actualizado usando funcion update
25     string fileName = " ", s; //
26     int num;
27     bool fileNameF =0;
28     string opcion;
29
30     while( fileNameF == 0){
31         cout << "Nombre del archivo(Opciones: 1.in, 2.in, 3.in):"
32         << endl;
33         cin >> fileName;
34         if(fileName == "1.in" || fileName == "2.in" || fileName ==
35         "3.in" ){
36             fileNameF = 1;
37         }
38         else{
39             fileNameF =0;
40         }
41     }
```

2) Menú

```
56 do{
57     cout<< "\t-----MENU-----\t"<<endl;
58     cout<<" 1) Create"<<endl;
59     cout<<" 2) Read"<<endl;
60     cout<<" 3) Update"<<endl;
61     cout<<" 4) Delete"<<endl;
62     cout<<" 5) Salir"<<endl;
63
64     cin>> opcion;
65     if(opcion == "1"){
66         //create
67         list.print();
68         create(data ,list);
69         cout << endl;
70     }else if(opcion == "2"){
71         //read
72         list.print();
73         read(data ,list);
74         cout << endl;
75     }else if(opcion == "3"){
76         //update
77         list.print();
78         cout<<endl;
79         cout << "Que valor desea actualizar?: " << endl;
80         cin >> data;
81         update( data, list, updateValue );
82         cout << endl;
83     }else if(opcion == "4"){
84         //delete
85         list.print();
86         cout<<endl;
87         del( data , list);
88         cout << endl;
89     }else if(opcion == "5"){
90         //delete
91         cout << "Hasta luego"<< endl;
92
93     }else{
94         cout<< "El valor no es una opcion valida. Digite denuevo el valor." <<endl;
95
96     }
```

3) Funciones void

```
121 void read(int data ,LinkedList<int> list){
122     cout<< endl;
123     cout << "Cual es el numero que deseas buscar: " << endl;
124     cin >> data;
125
126     if (list.read(data) != 0){
127         cout << "Elemento encontrado!" << endl;
128     } else {
129         cout << "El elemento no fue encontrado." << endl;
130     }
131 }
132
133 void del(int data ,LinkedList<int> list){
134     cout<< endl;
135     cout << "Que valor de la lista deseas eliminar " << endl;
136     cin >> data;
137     if (list.read(data) != 0){
138         list.del(data);
139         cout << "Elemento encontrado y eliminado." << endl;
140         cout << "La nueva lista sin el valor " << data<< " es: " << endl;
141         list.print();
142     } else {
143         cout << "El elemento que desea eliminar no fue encontrado." << endl;
144     }
145 }
146
147     cout << endl;
148 }
149
150
151 void create(int data ,LinkedList<int> list){
152     cout<< endl;
153     cout << "Que valor deseas agregar: " ;
154     cin >> data;
155     list.create(data);
156     list.print();
157     cout << endl;
158 }
```

4) templates

```
107  template <class T>
108  void LinkedList<T>::addFirst(T data){
109      head = new Node<T>(data, head); // crea un objeto dinámico en el tope
110      size++; //incrementa el tam del linked list
111  }
112
113  //create template
114  template <class T>
115  void LinkedList<T>::create(T data){
116
117      if(size == 0){
118          //llama a add first y crea un objeto
119          addFirst(data);
120          return; //salimos de la funcion
121      }
122      Node<T> *aux = head;
123      while(aux->getNext() != nullptr){
124          aux = aux->getNext();
125          size++;
126      }
127      aux->setNext(new Node<T>(data));
128
129      //current->setNext(*aux);
130  }
131
132  template <class T>
133  void LinkedList<T>::del(T data){
134      Node<T> *previous = nullptr; //empezamos con la pos nula
135      Node<T> *current = head; // apuntamos al siguiente como tope
136      //mientras que siguiente no sea igual al valor buscado y no este vacío
137      while ((current->getData() != data) && (current != nullptr)) {
138          //el anterior tope lo pasamos al siguiente
139          previous = current;
140          //pasamos el valor del siguiente nodo
141          current = current->getNext();
142      }
143  }
```

• Documentación

4 ejemplos que demuestran la documentación

1)

```
107  template <class T>
108  void LinkedList<T>::addFirst(T data){
109      head = new Node<T>(data, head); // crea un objeto dinámico en el tope
110      size++; //incrementa el tam del linked list
111  }
112
113  //create template
114  template <class T>
115  void LinkedList<T>::create(T data){
116
117      if(size == 0){
118          //llama a add first y crea un objeto
119          addFirst(data);
120          return; //salimos de la función
121      }
122      Node<T> *aux = head;
123      while(aux->getNext() != nullptr){
124          aux = aux->getNext();
125          size++;
126      }
127      aux->setNext(new Node<T>(data));
128
129      //current->setNext(*aux);
130  }
131
132  template <class T>
133  void LinkedList<T>::del(T data){
134      Node<T> *previous = nullptr; //empezamos con la pos nula
135      Node<T> *current = head; // apuntamos al siguiente como tope
136      //mientras que siguiente no sea igual al valor buscado y no este vacío
137      while ((current->getData() != data) && (current != nullptr)) {
138          //el anterior tope lo pasamos al siguiente
139          previous = current;
140          //pasamos el valor del siguiente nodo
141          current = current->getNext();
142      }
143  }
```

2)

```
1  #ifndef __LinkedList_h__
2  #define __LinkedList_h__
3
4  #include "Node.h"
5
6  using namespace std;
7
8  template <class T>
9  class LinkedList{
10     private:
11         Node<T> *head;
12         int size;
13     public:
14         LinkedList();
15         // ~LinkedList();//Delete
16         void print();
17         int getSize();
18         bool isEmpty();
19         void addFirst(T data);
20         //CRUD
21         void create(T data);
22         bool read(T data);
23         void update(T data, T updatecurrentValue);
24         void del(T data);
25         // void deleteAll();
26
27
28 };
29
30 //incicializar template
31 template <class T>
32 LinkedList<T>::LinkedList(){
33     head = nullptr;
34     size = 0;
35 }
36
37 //esta vacio el linkedin list? template
38 template <class T>
39 bool LinkedList<T>::isEmpty()
40 {
```

3)

```
void setNext(Node<T>* next); // variable void
};
//inicializar template Node<T>
template <class T>
Node<T>::Node(T data){
    this->data = data;
    this->next = nullptr; // NULL
}
//inicializar template Node<T> con variables
template <class T>
Node<T>::Node(T data, Node<T> *next){
    this->data = data;
    this->next = next;
}
//getData
template <class T>
T Node<T>::getData(){
    return data;
}

template <class T>
Node<T> * Node<T>::getNext(){
    return this->next;
}

//Setters

//setData
template <class T>
void Node<T>::setData(T data){
    this->data = data;
}
//SetNext
template <class T>
void Node<T>::setNext(Node<T>* next){
    this->next = next;
}
```


4)

```
int main(){

    //crear estructura linkedlist de data tipo int
    LinkedList<int> list;
    int data, index; // variable data y index tipo int
    int updateValue; // valor actualizado usando funcion update
    string fileName = " ", s; //
    int num;
    bool fileNameF =0;
    string opcion;

    while( fileNameF == 0){
        cout << "Nombre del archivo(Opciones: 1.in, 2.in, 3.in):" << endl;
        cin >> fileName;
        if(fileName == "1.in" || fileName == "2.in" || fileName == "3.in" ){
            fileNameF = 1;
        }
        else{
            fileNameF =0;
        }
    }

    //leer archivo
    ifstream in;
    in.open(fileName);
    while(in >> s){
        num = stoi(s);
        list.create(num);
    }

    in.close();
    list.print();
    cout << endl;
```

● Mantenibilidad y facilidad de lectura y entendimiento

Cómo se puede evidenciar los diferentes casos de prueba y documentación, el código cumple con los estándares de programación tanto de identificación y de entendimiento. Dado que este está documenta claramente lo que sucede en todas las funciones requeridas en este programa y se hacen reseñas en las partes básicas para explicar las acciones del código. El mantenimiento por esta razones no debe ser dificultoso dado que el código está construido con plantillas que sustentan buenas prácticas y se trata de usar la mayor cantidad de funciones para evitar repetir código innecesario.

Conclusión

A lo largo de esta actividad se trabajó con la estructura de datos que son las listas entrelazadas. Donde se utilizaron plantillas para crear las funciones requeridas para esta actividad y se pudo considerar las diferentes necesidades para trabajar con archivos y poder modificarlos o leerlos.

Referencias

Hernández, V. (2017, 11 mayo). *C/C++: plantillas (templates) en C++*.

codingornot. <https://codingornot.com/cc-plantillas-templates-en-c>

Medina, R. (2020, 18 diciembre). *Estructura de Datos - Linked List (Lista Enlazada)*. DEV Community.

<https://dev.to/ronnymedina/estructura-de-datos-linked-list-lista-enlazada-2h9>