

Universidad Nacional del Altiplano
Facultad de Ingeniería Estadística e Informática
Docente: Fred Torres Cruz
Estudiante 1 : Ruth Karina Apaza Solis

<https://github.com/R-Karina-A-Solis/lectura-de-datos.git>

Trabajo Encargado: Lista enlazadas

```
1 #include <iostream>
2 #include <fstream>
3 #include <string>
4 #include <ctime> // Para medir tiempo
5 #include <cstdlib> // Para atoi
6 #include <thread>
7 using namespace std;
8
9 struct Player {
10     int id;
11     string name;
12     int score;
13     Player* next;
14 };
15
16
17 Player* createNode(int id, string name, int score) {
18     Player* newNode = new Player();
19     newNode->id = id;
20     newNode->name = name;
21     newNode->score = score;
22     newNode->next = NULL;
23     return newNode;
24 }
25
26
27 void appendNode(Player*& head, int id, string name, int score) {
28     Player* newNode = createNode(id, name, score);
29     if (head == NULL) {
30         head = newNode;
31         return;
32     }
33     Player* temp = head;
34     while (temp->next != NULL) {
35         temp = temp->next;
36     }
37     temp->next = newNode;
38 }
39
40 double calculateAverage(Player* head) {
41     double sum = 0;
```

```
42     int count = 0;
43     Player* temp = head;
44     while (temp != NULL) {
45         sum += temp->score;
46         count++;
47         temp = temp->next;
48     }
49     return (count == 0) ? 0 : sum / count;
50 }
51
52 Player* findHighestScore(Player* head) {
53     Player* highest = head;
54     Player* temp = head;
55     while (temp != NULL) {
56         if (temp->score > highest->score) {
57             highest = temp;
58         }
59         temp = temp->next;
60     }
61     return highest;
62 }
63
64 Player* findLowestScore(Player* head) {
65     Player* lowest = head;
66     Player* temp = head;
67     while (temp != NULL) {
68         if (temp->score < lowest->score) {
69             lowest = temp;
70         }
71         temp = temp->next;
72     }
73     return lowest;
74 }
75
76 void removeBelowAverage(Player*& head, double average) {
77     Player* temp = head;
78     Player* prev = NULL;
79     while (temp != NULL) {
80         if (temp->score < average) {
81             if (prev != NULL) {
82                 prev->next = temp->next;
83             } else {
84                 head = temp->next;
85             }
86             Player* toDelete = temp;
87             temp = temp->next;
88             delete toDelete;
89
90             // Introduce una pausa de 1000 microsegundos (1 milisegundo)
91             this_thread::sleep_for(chrono::microseconds(1000)); // Pausa
92                                 de 1000 microsegundos (1 milisegundo)
93         } else {
94             prev = temp;
95             temp = temp->next;
```

```
95     }
96 }
97 }
98
99 int main() {
100     Player* head = NULL;
101     ifstream inputFile("jugadores.txt");
102     string line;
103
104     while (getline(inputFile, line)) {
105         int id, score;
106         string name;
107         size_t firstSpace = line.find(' ');
108         size_t lastSpace = line.rfind(' ');
109
110         id = atoi(line.substr(0, firstSpace).c_str());
111         name = line.substr(firstSpace + 1, lastSpace - firstSpace - 1);
112         score = atoi(line.substr(lastSpace + 1).c_str());
113
114         appendNode(head, id, name, score);
115     }
116     inputFile.close();
117
118
119     double average = calculateAverage(head);
120     cout << "Puntuacion media: " << average << endl;
121
122     Player* highest = findHighestScore(head);
123     Player* lowest = findLowestScore(head);
124
125     cout << "Puntuacion mas alta: ID=" << highest->id
126         << ", Nombre=" << highest->name
127         << ", Puntuacion=" << highest->score << endl;
128
129     cout << "Puntuacion mas baja: ID=" << lowest->id
130         << ", Nombre=" << lowest->name
131         << ", Puntuacion=" << lowest->score << endl;
132
133     clock_t start = clock();
134     removeBelowAverage(head, average);
135     clock_t end = clock();
136
137     double elapsed = double(end - start) / CLOCKS_PER_SEC;
138     cout << "Tiempo para eliminar jugadores por debajo del promedio: " <<
139         elapsed << " segundos" << endl;
140
141     return 0;
142 }
```

```
l13     string name;  
l14     size_t firstSpace = line.find(' ');  
l15     size_t lastSpace = line.rfind(' ');  
l16  
l17  
l18  
l19  
l20     Puntuacion media: 72.3039  
l21     Puntuacion mas alta: ID=12, Nombre=Fernanda, Puntuacion=106  
l22     Puntuacion mas baja: ID=55, Nombre=David, Puntuacion=35  
l23     Tiempo para eliminar jugadores por debajo del promedio: 0.501 segundos  
l24  
l25  
l26     Process returned 0 (0x0)   execution time : 0.613 s  
l27     Press any key to continue.  
l28  
l29  
l30  
l31  
l32  
l33     cout << "Puntuacion mas alta: ID=" << highest->id  
l34           << ", Nombre=" << highest->name  
l35           << ", Puntuacion=" << highest->score << endl;  
l36  
l37     cout << "Puntuacion mas baja: ID=" << lowest->id  
l38           << ", Nombre=" << lowest->name  
l39           << ", Puntuacion=" << lowest->score << endl;  
l40  
l41     // Eliminar jugadores por debajo del promedio y medir el tiempo
```

Figura 1: Búsqueda lineal