

Universidad Nacional del Altiplano  
Facultad de Ingeniería Estadística e Informática  
Docente: Fred Torres Cruz  
Alumno: Roberto Angel Ticona Miramira

### Práctica Calificada: Listas Enlazadas

Enlace al repositorio de Github: Práctica de laboratorio Listas enlazadas

#### Definición

Se presentará el código que se solicitó para la práctica de laboratorio.

```
Practica.cpp
1  #include <iostream>
2  #include <fstream>
3  #include <string>
4  #include <ctime>
5  #include <cstdlib>
6
7
8  using namespace std;
9
10 // Definición de la estructura del nodo
11 struct Player {
12     int id;
13     string name;
14     int score;
15     Player* next;
16 };
17
18 // Función para crear un nuevo nodo
19 Player* createNode(int id, string name, int score) {
20     Player* newNode = new Player();
21     newNode->id = id;
22     newNode->name = name;
23     newNode->score = score;
24     newNode->next = NULL;
25     return newNode;
26 }
27
```

Figura 1: Código

```
28 // Función para agregar un nodo al final de la lista
29 void appendNode(Player*& head, int id, string name, int score) {
30     Player* newNode = createNode(id, name, score);
31     if (head == NULL) {
32         head = newNode;
33         return;
34     }
35     Player* temp = head;
36     while (temp->next != NULL) {
37         temp = temp->next;
38     }
39     temp->next = newNode;
40 }
41
42 // Función para calcular la puntuación promedio
43 double calculateAverage(Player* head) {
44     double sum = 0;
45     int count = 0;
46     Player* temp = head;
47     while (temp != NULL) {
48         sum += temp->score;
49         count++;
50         temp = temp->next;
51     }
52     return (count == 0) ? 0 : sum / count;
53 }
54
```

Figura 2: Código

```
55 // Función para encontrar al jugador con la puntuación más alta
56 Player* findHighestScore(Player* head) {
57     Player* highest = head;
58     Player* temp = head;
59     while (temp != NULL) {
60         if (temp->score > highest->score) {
61             highest = temp;
62         }
63         temp = temp->next;
64     }
65     return highest;
66 }
67
68 // Función para encontrar al jugador con la puntuación más baja
69 Player* findLowestScore(Player* head) {
70     Player* lowest = head;
71     Player* temp = head;
72     while (temp != NULL) {
73         if (temp->score < lowest->score) {
74             lowest = temp;
75         }
76         temp = temp->next;
77     }
78     return lowest;
79 }
80
```

Figura 3: Código

```
81 // Función para eliminar jugadores con puntuaciones por debajo del promedio
82 void removeBelowAverage(Player*& head, double average) {
83     Player* temp = head;
84     Player* prev = NULL;
85     while (temp != NULL) {
86         if (temp->score < average) {
87             if (prev != NULL) {
88                 prev->next = temp->next;
89             } else {
90                 head = temp->next;
91             }
92             Player* toDelete = temp;
93             temp = temp->next;
94             delete toDelete;
95         } else {
96             prev = temp;
97             temp = temp->next;
98         }
99     }
100 }
101
102 // Función principal
103 int main() {
104     Player* head = NULL;
105     ifstream inputFile("jugadores.txt");
106     string line;
107
```

Figura 4: Código

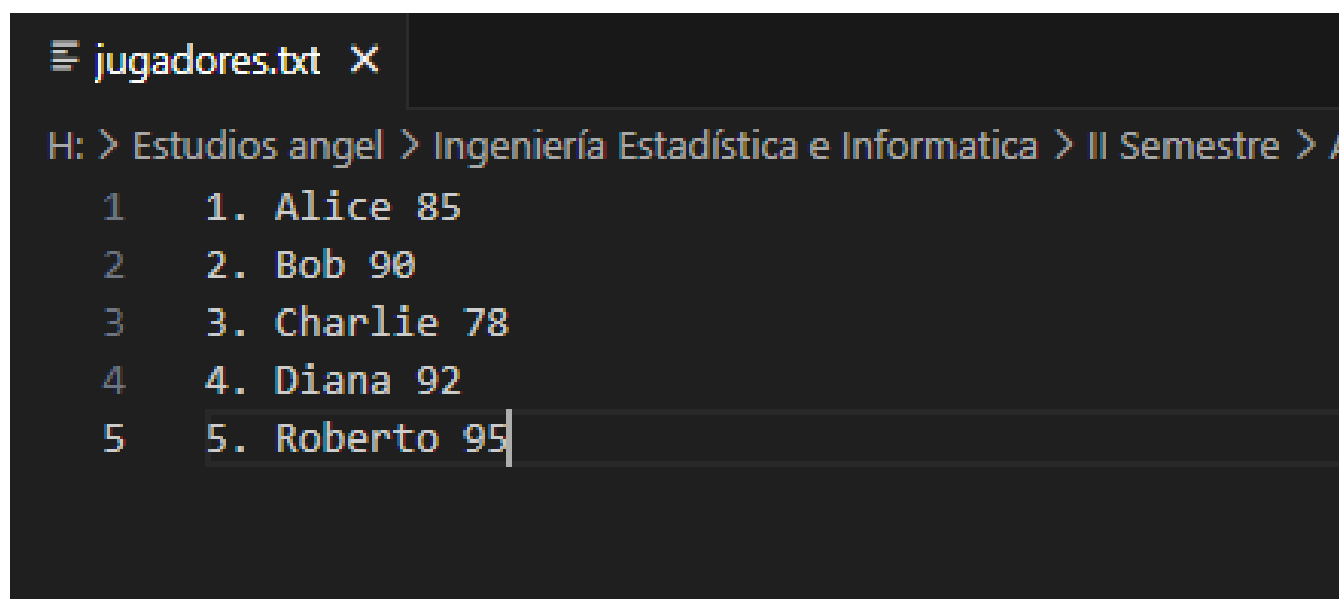
```
108 // Leer datos desde el archivo
109 while (getline(inputFile, line)) {
110     int id, score;
111     string name;
112     size_t firstSpace = line.find(' ');
113     size_t lastSpace = line.rfind(' ');
114
115     id = atoi(line.substr(0, firstSpace).c_str());
116     name = line.substr(firstSpace + 1, lastSpace - firstSpace - 1);
117     score = atoi(line.substr(lastSpace + 1).c_str());
118
119     appendNode(head, id, name, score);
120 }
121 inputFile.close();
122
123 // Calcular la puntuación promedio
124 double average = calculateAverage(head);
125 cout << "Puntuacion Promedio: " << average << endl;
126
```

Figura 5: Código

```
127 // Encontrar al jugador con la puntuación más alta y más baja
128 Player* highest = findHighestScore(head);
129 Player* lowest = findLowestScore(head);
130
131 cout << "Mayor Puntuacion: JugadorID=" << highest->id << ", Nombre de jugador=" << highest->name
132 | << ", Score=" << highest->score << endl;
133 cout << "Menor Puntuacion: JugadorID=" << lowest->id << ", Nombre de jugador=" << lowest->name
134 | << ", Score=" << lowest->score << endl;
135
136 // Medir tiempo para eliminar jugadores por debajo del promedio
137 clock_t start = clock();
138 removeBelowAverage(head, average);
139 clock_t end = clock();
140
141 double elapsed = double(end - start) / CLOCKS_PER_SEC;
142 cout << "Tiempo para eliminar jugadores por debajo del promedio: " << elapsed << " segundos" << endl;
143
144 return 0;
145 }
```

Figura 6: Código

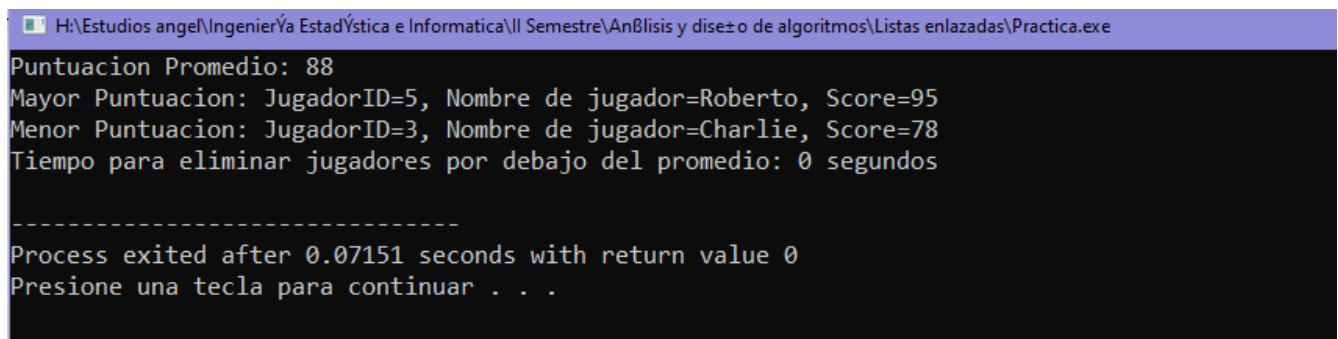
Datos que se leyeron



```
jugadores.txt X
H: > Estudios angel > Ingeniería Estadística e Informatica > II Semestre > A
1 1. Alice 85
2 2. Bob 90
3 3. Charlie 78
4 4. Diana 92
5 5. Roberto 95
```

Figura 7: Datos

Resultados



```
H:\Estudios angel\Ingeniería Estadística e Informática\II Semestre\Análisis y diseño de algoritmos>Listas enlazadas\Practica.exe
Puntuacion Promedio: 88
Mayor Puntuacion: JugadorID=5, Nombre de jugador=Roberto, Score=95
Menor Puntuacion: JugadorID=3, Nombre de jugador=Charlie, Score=78
Tiempo para eliminar jugadores por debajo del promedio: 0 segundos

-----
Process exited after 0.07151 seconds with return value 0
Presione una tecla para continuar . . .
```

Figura 8: Compilación