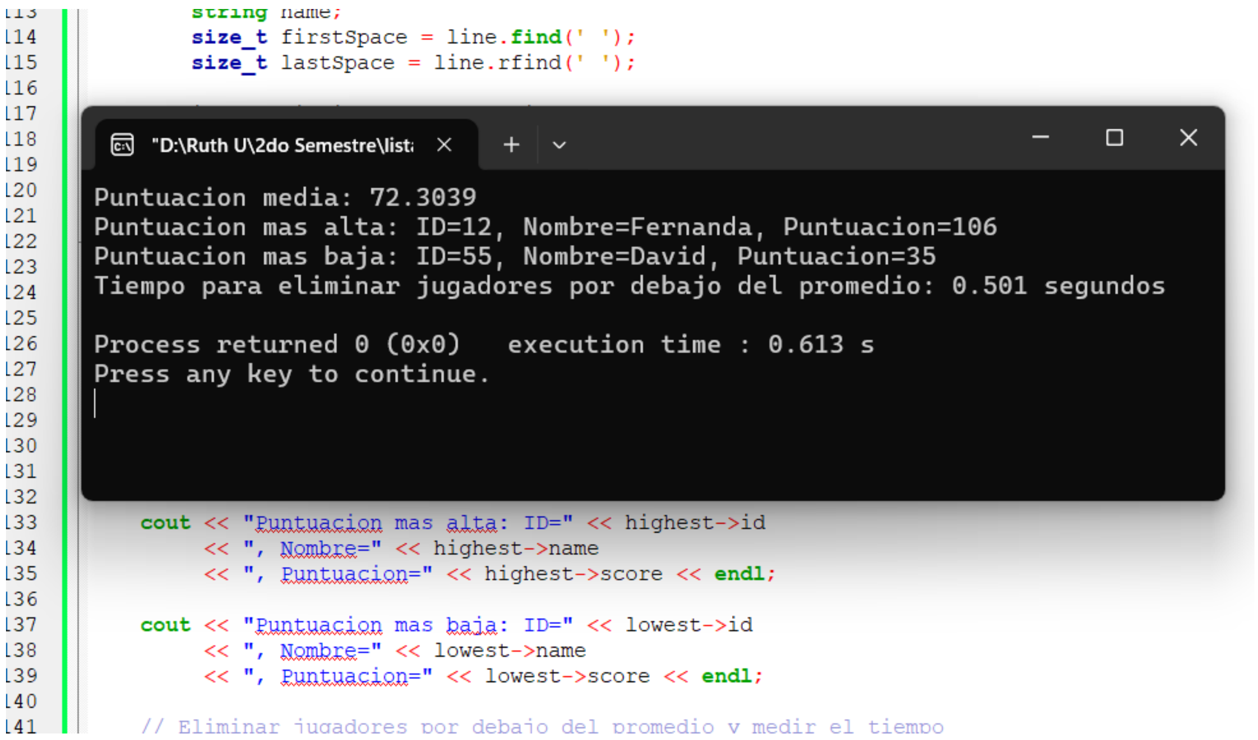
Universidad Nacional del Altiplano
Facultad de Ingeniería Estadística e Informática
Docente: Fred Torres Cruz
Estudiante 1 : Ruth Karina Apaza Solis

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

Trabajo Encargado: Lista enlazadas

```
#include <iostream>
#include <fstream>
#include <string>
#include <ctime> // Para medir tiempo
#include <cstdlib> // Para atoi
#include <thread>
using namespace std;

struct Player {
    int id;
    string name;
    int score;
    Player* next;
};


Player* createNode(int id, string name, int score) {
    Player* newNode = new Player();
    newNode->id = id;
    newNode->name = name;
    newNode->score = score;
    newNode->next = NULL;
    return newNode;
}


void appendNode(Player*& head, int id, string name, int score) {
    Player* newNode = createNode(id, name, score);
    if (head == NULL) {
        head = newNode;
        return;
    }
    Player* temp = head;
    while (temp->next != NULL) {
        temp = temp->next;
    }
    temp->next = newNode;
}

double calculateAverage(Player* head) {
    double sum = 0;
    int count = 0;
    Player* temp = head;
    while (temp != NULL) {
        sum += temp->score;
        count++;
        temp = temp->next;
    }
    return (count == 0) ? 0 : sum / count;
}

Player* findHighestScore(Player* head) {
    Player* highest = head;
    Player* temp = head;
    while (temp != NULL) {
        if (temp->score > highest->score) {
            highest = temp;
        }
        temp = temp->next;
    }
    return highest;
}

Player* findLowestScore(Player* head) {
    Player* lowest = head;
    Player* temp = head;
    while (temp != NULL) {
        if (temp->score < lowest->score) {
            lowest = temp;
        }
        temp = temp->next;
    }
    return lowest;
}

void removeBelowAverage(Player*& head, double average) {
    Player* temp = head;
    Player* prev = NULL;
    while (temp != NULL) {
        if (temp->score < average) {
            if (prev != NULL) {
                prev->next = temp->next;
            } else {
                head = temp->next;
            }
            Player* toDelete = temp;
            temp = temp->next;
            delete toDelete;

            // Introduce una pausa de 1000 microsegundos (1 milisegundo)
            this_thread::sleep_for(chrono::microseconds(1000)); // Pausa
                de 1000 microsegundos (1 milisegundo)
        } else {
            prev = temp;
            temp = temp->next;
        }
    }
}

int main() {
    Player* head = NULL;
    ifstream inputFile("jugadores.txt");
    string line;

    while (getline(inputFile, line)) {
        int id, score;
        string name;
        size_t firstSpace = line.find(' ');
        size_t lastSpace = line.rfind(' ');

        id = atoi(line.substr(0, firstSpace).c_str());
        name = line.substr(firstSpace + 1, lastSpace - firstSpace - 1);
        score = atoi(line.substr(lastSpace + 1).c_str());

        appendNode(head, id, name, score);
    }
    inputFile.close();


    double average = calculateAverage(head);
    cout << "Puntuacion media: " << average << endl;

    Player* highest = findHighestScore(head);
    Player* lowest = findLowestScore(head);

    cout << "Puntuacion mas alta: ID=" << highest->id
         << ", Nombre=" << highest->name
         << ", Puntuacion=" << highest->score << endl;

    cout << "Puntuacion mas baja: ID=" << lowest->id
         << ", Nombre=" << lowest->name
         << ", Puntuacion=" << lowest->score << endl;

    clock_t start = clock();
    removeBelowAverage(head, average);
    clock_t end = clock();

    double elapsed = double(end - start) / CLOCKS_PER_SEC;
    cout << "Tiempo para eliminar jugadores por debajo del promedio: " <<
        elapsed << " segundos" << endl;

    return 0;
}
```



```
        string name;
        size_t firstSpace = line.find(' ');
        size_t lastSpace = line.rfind(' ');
```

```
Puntuacion media: 72.3039
Puntuacion mas alta: ID=12, Nombre=Fernanda, Puntuacion=106
Puntuacion mas baja: ID=55, Nombre=David, Puntuacion=35
Tiempo para eliminar jugadores por debajo del promedio: 0.501 segundos

Process returned 0 (0x0)   execution time : 0.613 s
Press any key to continue.
```

```
    cout << "Puntuacion mas alta: ID=" << highest->id
         << ", Nombre=" << highest->name
         << ", Puntuacion=" << highest->score << endl;

    cout << "Puntuacion mas baja: ID=" << lowest->id
         << ", Nombre=" << lowest->name
         << ", Puntuacion=" << lowest->score << endl;

    // Eliminar jugadores por debajo del promedio y medir el tiempo
```

Figura 1: Busqueda lineal