Examen Programación Para Competición Intermedio: Pregunta 1
Estudiante: Wilson Joel Valeriano Quispe

EXPLORACIÓN LINEAL

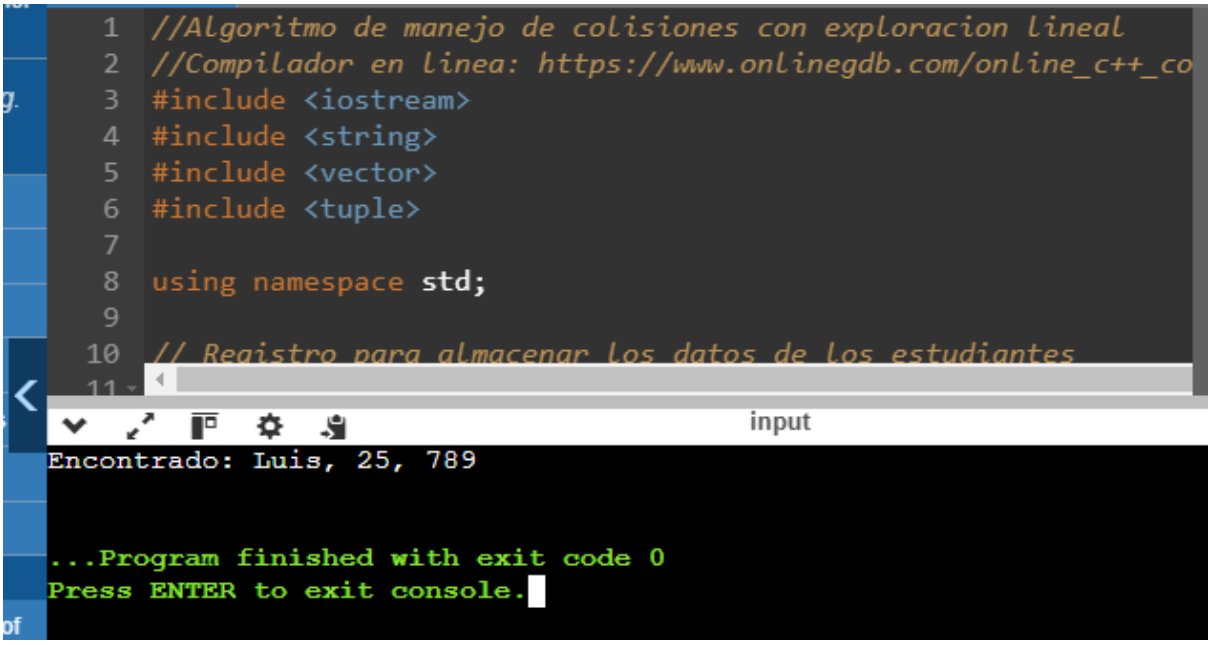
```cpp
//Algoritmo de manejo de colisiones con exploracion lineal
//Compilador en linea: https://www.onlinegdb.com/online_c++_compiler
#include <iostream>
#include <string>
#include <vector>
#include <tuple>

using namespace std;

// Registro para almacenar los datos de los estudiantes
struct Estudiante {
    string nombre;
    int numero_estudiante;
    int numero_seguro;
};

// Tabla hash con exploracion lineal
class TablaHashLineal {
    vector<Estudiante> tabla;
    vector<bool> ocupado;
    int tamaño;

public:
    TablaHashLineal(int t) : tamaño(t) {
        tabla.resize(t);
        ocupado.resize(t, false);
    }

    int funcionHash(int clave) {
        return clave % tamaño;
    }

    void insertar(Estudiante est) {
        int clave = funcionHash(est.numero_estudiante);
        int i = clave;

        // Exploracion lineal para encontrar espacio
        while (ocupado[i]) {
            i = (i + 1) % tamaño;
        }
```

Examen Programación Para Competición Intermedio: Pregunta 1
Estudiante: Wilson Joel Valeriano Quispe

```
        tabla[i] = est;
        ocupado[i] = true;
    }

    Estudiante buscar(int clave) {
        int hash = funcionHash(clave);
        int i = hash;

        while (ocupado[i]) {
            if (tabla[i].numero_estudiante == clave) {
                return tabla[i];
            }
            i = (i + 1) % tamaño;
        }

        throw runtime_error("Estudiante no encontrado.");
    }
};

int main() {
    TablaHashLineal tabla(10);

    // Insertar estudiantes
    tabla.insertar({"Juan", 1, 123});
    tabla.insertar({"Ana", 7, 456});
    tabla.insertar({"Luis", 25, 789});
    tabla.insertar({"Maria", 35, 321});
    tabla.insertar({"Pedro", 43, 654});
    tabla.insertar({"Laura", 51, 987});
    tabla.insertar({"Jose", 72, 112});
    tabla.insertar({"Sara", 65, 223});
    tabla.insertar({"Carlos", 37, 334});
    tabla.insertar({"Lucia", 54, 445});

    // Buscar estudiante
    try {
        Estudiante est = tabla.buscar(25);
        cout << "Encontrado: " << est.nombre << ", " <<
est.numero_estudiante << ", " << est.numero_seguro << endl;
    } catch (exception& e) {
        cout << e.what() << endl;
    }
```

Examen Programación Para Competición Intermedio: Pregunta 1
Estudiante: Wilson Joel Valeriano Quispe

```
    return 0;
}
```



EXPLORACIÓN CUADRÁTICA

```
//Algoritmo de manejo de colisiones con exploracion cuadratica
//Compilador en linea: https://www.onlinegdb.com/online_c++_compiler
#include <iostream>
#include <string>
#include <vector>
#include <tuple>

using namespace std;

// Registro para almacenar los datos de los estudiantes
struct Estudiante {
    string nombre;
    int numero_estudiante;
    int numero_seguro;
};

// Tabla hash con exploracion cuadratica
class TablaHashCuadratica {
    vector<Estudiante> tabla;
    vector<bool> ocupado;
    int tamaño;
```

Examen Programación Para Competición Intermedio: Pregunta 1
Estudiante: Wilson Joel Valeriano Quispe

```
public:
    TablaHashCuadratica(int t) : tamaño(t) {
        tabla.resize(t);
        ocupado.resize(t, false);
    }

    int funcionHash(int clave) {
        return clave % tamaño;
    }

    void insertar(Estudiante est) {
        int clave = funcionHash(est.numero_estudiante);
        int i = 0;

        // Exploracion cuadratica para encontrar espacio
        while (ocupado[(clave + i * i) % tamaño]) {
            i++;
        }

        int pos = (clave + i * i) % tamaño;
        tabla[pos] = est;
        ocupado[pos] = true;
    }

    Estudiante buscar(int clave) {
        int hash = funcionHash(clave);
        int i = 0;

        while (ocupado[(hash + i * i) % tamaño]) {
            int pos = (hash + i * i) % tamaño;
            if (tabla[pos].numero_estudiante == clave) {
                return tabla[pos];
            }
            i++;
        }

        throw runtime_error("Estudiante no encontrado.");
    }
};

int main() {
    TablaHashCuadratica tabla(10);
```

Examen Programación Para Competición Intermedio: Pregunta 1
Estudiante: Wilson Joel Valeriano Quispe

```
    // Insertar estudiantes
    tabla.insertar({"Juan", 1, 123});
    tabla.insertar({"Ana", 7, 456});
    tabla.insertar({"Luis", 25, 789});
    tabla.insertar({"Maria", 35, 321});
    tabla.insertar({"Pedro", 43, 654});
    tabla.insertar({"Laura", 51, 987});
    tabla.insertar({"Jose", 72, 112});
    tabla.insertar({"Sara", 65, 223});
    tabla.insertar({"Carlos", 37, 334});
    tabla.insertar({"Lucia", 54, 445});

    // Buscar estudiante
    try {
        Estudiante est = tabla.buscar(25);
        cout << "Encontrado: " << est.nombre << ", " <<
est.numero_estudiante << ", " << est.numero_seguro << endl;
    } catch (exception& e) {
        cout << e.what() << endl;
    }

    return 0;
}
```

```
1  //Algoritmo de manejo de colisiones con exploracion cuadratica
2  //Compilador en linea: https://www.onlinegdb.com/online_c++_co
3  #include <iostream>
4  #include <string>
5  #include <vector>
6  #include <tuple>
7
8  using namespace std;
9
10 // Registro para almacenar los datos de los estudiantes
```

input

```
Encontrado: Luis, 25, 789
double free or corruption (out)


...Program finished with exit code 134
Press ENTER to exit console.
```

DOBLE HASH

Examen Programación Para Competición Intermedio: Pregunta 1
Estudiante: Wilson Joel Valeriano Quispe

```cpp
//Algoritmo de manejo de colisiones con doble hash
//Compilador en linea: https://www.onlinegdb.com/online_c++_compiler
#include <iostream>
#include <string>
#include <vector>
#include <tuple>

using namespace std;

// Registro para almacenar los datos de los estudiantes
struct Estudiante {
    string nombre;
    int numero_estudiante;
    int numero_seguro;
};

// Tabla hash con doble hash
class TablaHashDoble {
    vector<Estudiante> tabla;
    vector<bool> ocupado;
    int tamaño;

public:
    TablaHashDoble(int t) : tamaño(t) {
        tabla.resize(t);
        ocupado.resize(t, false);
    }

    // Primera funcion hash
    int funcionHash1(int clave) {
        return clave % tamaño;
    }

    // Segunda funcion hash
    int funcionHash2(int clave) {
        return 7 - (clave % 7); // Valor primo menor al tamaño
    }

    void insertar(Estudiante est) {
        int hash1 = funcionHash1(est.numero_estudiante);
        int hash2 = funcionHash2(est.numero_estudiante);
        if (hash2 == 0) hash2 = 1; // Evitar que el avance sea 0
```

Examen Programación Para Competición Intermedio: Pregunta 1
Estudiante: Wilson Joel Valeriano Quispe

```
        int i = 0;

        // Doble hash para encontrar espacio
        while (ocupado[(hash1 + i * hash2) % tamaño]) {
            i++;
        }

        int pos = (hash1 + i * hash2) % tamaño;
        tabla[pos] = est;
        ocupado[pos] = true;

    }

    Estudiante buscar(int clave) {
        int hash1 = funcionHash1(clave);
        int hash2 = funcionHash2(clave);
        if (hash2 == 0) hash2 = 1; // Evitar avance 0
        int i = 0;

        while (ocupado[(hash1 + i * hash2) % tamaño]) {
            int pos = (hash1 + i * hash2) % tamaño;
            if (tabla[pos].numero_estudiante == clave) {
                return tabla[pos];
            }
            i++;
        }

        throw runtime_error("Estudiante no encontrado.");
    }
};

int main() {
    // Crear tabla hash con tamaño primo
    TablaHashDoble tabla(11);

    // Insertar estudiantes
    tabla.insertar({"Juan", 1, 123});
    tabla.insertar({"Ana", 7, 456});
    tabla.insertar({"Luis", 25, 789});
    tabla.insertar({"Maria", 35, 321});
    tabla.insertar({"Pedro", 43, 654});
    tabla.insertar({"Laura", 51, 987});
    tabla.insertar({"Jose", 72, 112});
```

Examen Programación Para Competición Intermedio: Pregunta 1
Estudiante: Wilson Joel Valeriano Quispe

```
    tabla.insertar({"Sara", 65, 223});
    tabla.insertar({"Carlos", 37, 334});
    tabla.insertar({"Lucia", 54, 445});

    // Buscar estudiante
    try {
        Estudiante est = tabla.buscar(72);
        cout << "Encontrado: " << est.nombre << ", " <<
est.numero_estudiante << ", " << est.numero_seguro << endl;
    } catch (exception& e) {
        cout << e.what() << endl;
    }

    return 0;
}
```