```
#include <iostream>
#include <fstream>
#include <cstring>
#include <climits>
#include <map>
#include <vector>
using namespace std;

void readFromConsole(char buffer[], int size){
    string temp;
    getline (cin,temp);
    for(int i=0; i<size; i++)
        //llenar con espacios en blanco si el tamaño del string recibido en menor
que el tamaño asignado para ese campo (size)
        buffer[i] = (i < temp.size())? temp[i] : ' ';
    //se llena el último espacio con \0 porque es un char
    buffer[size-1] = '\0';
    cin.clear();
}

struct Record
{
    char nombre[30];
    char carrera[20];
    int ciclo;
    void setData(){
        cout<<"Nombre: ";
        readFromConsole(this->nombre,30);

        cout<<"Carrera: ";
        readFromConsole(this->carrera,20);

        cout<<"Ciclo: ";
        cin >> this->ciclo;
        cin.ignore();
    }
    string getKey(){
        return nombre;
    }

    void print(){
        cout << nombre <<" "<< carrera <<" " << ciclo <<endl;
    }
};

//Sobrecarga cout
ostream & operator << (ostream & stream, Record & p)
{
    stream.write ((char*)&p,sizeof(Record));
    //stream << flush;
    return stream;
}

//Sobrecarga cin
istream & operator >> (istream & stream, Record & p)
{
    stream.read ((char*)&p, sizeof(Record));
    return stream;
}
```

```
class RandomFile{
private:
    string filename;
    string indexname;
    map<string, long> index;
public:
    RandomFile(string filename){
        this->filename = filename;
        this->indexname =  "ind_" + filename ;
        readIndex();
    }

    void readIndex(){
        ifstream indexFile(indexname, ios::binary| ios::app);
        if (!indexFile.is_open())
            cerr << "No se pudo abrir el archivo\n";
        indexFile.seekg(0, ios::end);
        if (indexFile.tellg() == 0){
            indexFile.close();
            cout << "Archivo de índices vacío\n";
        }
        else{
            indexFile.seekg(0, ios::beg);
            char nombre_key[30];
            long pos_fisica;
            while (indexFile.read(nombre_key,30)){
                indexFile.read((char*)&pos_fisica, sizeof(pos_fisica));
                index[nombre_key]=pos_fisica;
            }
            indexFile.close();
        }
    }

    void writeIndex(){
        ofstream indexFile(indexname, ios::binary | ios::app);
        if (!indexFile.is_open())
            cerr << "No se pudo abrir el archivo\n";
        indexFile.seekp(0, ios::end);
        for (auto& index_ : index){
            indexFile.write(index_.first.c_str(), 30);
            indexFile.write((char*)& index_.second, sizeof(long));
        }
        indexFile.close();
    }

    void write(Record data){

        ofstream file(filename, ios::binary | ios::app);
        if (!file.is_open())
            cerr << "No se pudo abrir el archivo\n";

        file.seekp(0, ios::end);
        long pos_Fisica = file.tellp();

        file << data;
        index[data.getKey()] = pos_Fisica;
        file.close();
    }
```

```
    void scanAll(){
        Record record;
        ifstream file(filename, ios::binary);
        if (!file.is_open())
            cerr << "No se pudo abrir el archivo\n";
        while (!file.eof()){
            file >> record;
            record.print();
        }
        file.close();
    }

    //recorrido en O(n) usando el index
    void scanAllByIndex(){
        Record record{};
        ifstream file(filename, ios::binary | ios::in);
        if (!file.is_open())
            cerr << "No se pudo abrir el archivo\n";
        file.seekg(0, ios::beg);
        for (auto& index_ : index){
            file.seekg(index_.second, ios::beg);
            file >> record;
            record.print();
        }
        file.close();
    }

    void searchByKey(){
        char nombre_alumno[30];
        Record record;
        cout << "Ingrese el nombre del alumno: ";
        readFromConsole(nombre_alumno,30);
        if (index.find(nombre_alumno) == index.end()){
            cerr << "No se pudo encontrar al alumno\n";
        }
        long posFis = index [nombre_alumno];
        ifstream file(filename, ios::binary);
        if (!file.is_open())
            cerr << "No se pudo abrir el archivo\n";
        file.seekg(posFis, ios::beg);
        file >> record;
        record.print();
        file.close();

    }

    ~RandomFile(){
        writeIndex();
    }

};
```