TRABAJO PRÁCTICO INTEGRADOR

PARTE 2

Docente: Sanroman, Gabriela

Alumnos: Gomez, Sol; Marquez, Bautista; Varela, Lucas; Vasques, Kalil

Materia: Algoritmos y Estructuras de Datos

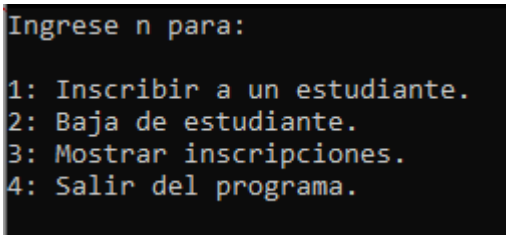
Comisión: K1022

Institución: Universidad Tecnológica Nacional

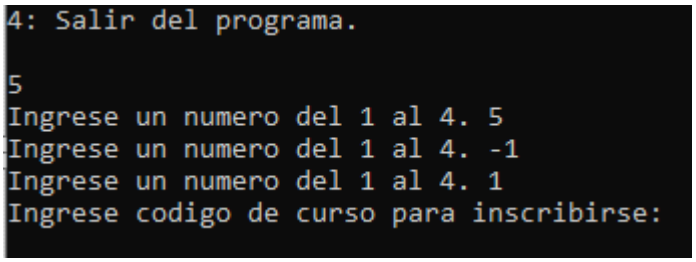
Ciclo Lectivo: 2022

Interfaz hacia el usuario

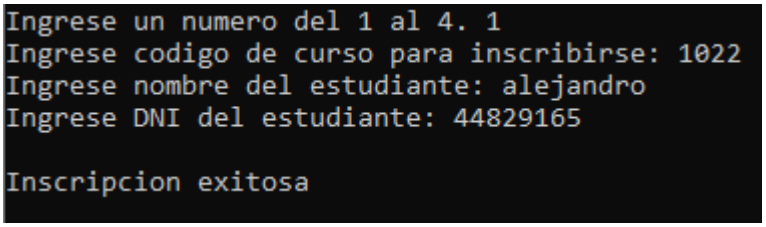
Se despliega un menú con 4 opciones para la inscripción de estudiantes.



El programa prevé la posibilidad de que se ingrese un número erróneo, solicitando otro tantas veces como sea necesario.



Cuando el usuario digita un 1 se le pide ingrese el curso al que se quiere inscribir el alumno, junto con su nombre y documento.



Cuando el usuario digita un 2 comienza el proceso de baja de estudiante, para lo que le pide el dni de la persona a dar de baja y notifica que el proceso finalizó.

```
1: Inscribir a un estudiante.
2: Baja de estudiante.
3: Mostrar inscripciones.
4: Salir del programa.

2
Ingrese documento del alumno a eliminar de la lista: 7698660
Estudiante eliminado de la lista.
Ingrese n para:

1: Inscribir a un estudiante.
2: Baja de estudiante.
3: Mostrar inscripciones.
4: Salir del programa.
```

Cuando el usuario digita un 3 se muestra una lista por curso con los estudiantes inscriptos hasta el momento (los que consiguieron cupo).

```
3
Codigo de Curso: 649
Vacantes Ocupadas: 0
Vacantes Disponibles: 30

LISTA ALUMNOS INSCRIPTOS:
----------------------------------------

Codigo de Curso: 872
Vacantes Ocupadas: 0
Vacantes Disponibles: 5

LISTA ALUMNOS INSCRIPTOS:
----------------------------------------

Codigo de Curso: 1022
Vacantes Ocupadas: 5
Vacantes Disponibles: 0

LISTA ALUMNOS INSCRIPTOS:
Nombre: halima
DNI: 4167949

Nombre: kimberley
DNI: 5300514

Nombre: essie
DNI: 6192756

Nombre: lavanya
DNI: 9134831

Nombre: alpha
DNI: 9475639

----------------------------------------

Codigo de Curso: 1024
Vacantes Ocupadas: 0
Vacantes Disponibles: 5

LISTA ALUMNOS INSCRIPTOS:
----------------------------------------

Codigo de Curso: 1093
Vacantes Ocupadas: 0
Vacantes Disponibles: 5

LISTA ALUMNOS INSCRIPTOS:
----------------------------------------
```

Cuando el usuario digita un 4 finaliza el programa, mostrando la cola de espera de cada curso y un árbol recorrido INORDEN con todos los docentes ordenados por documento, y la cantidad de alumnos a cargo de cada uno.

```
4
COLA DE ESPERA CURSO  649
----------------------------------------

COLA DE ESPERA CURSO  872
----------------------------------------

COLA DE ESPERA CURSO  1022
Nombre: titrit
DNI: 7053777

Nombre: eva
DNI: 5448956

Nombre: deb
DNI: 3114450

Nombre: eachann
DNI: 287558

Nombre: arya
DNI: 2928385

Nombre: yaqoob
DNI: 7867978

----------------------------------------

COLA DE ESPERA CURSO  1024
----------------------------------------

COLA DE ESPERA CURSO  1093
----------------------------------------

COLA DE ESPERA CURSO  1854
----------------------------------------

COLA DE ESPERA CURSO  2251
----------------------------------------

COLA DE ESPERA CURSO  3986
----------------------------------------

COLA DE ESPERA CURSO  5159
----------------------------------------

COLA DE ESPERA CURSO  7047
----------------------------------------

COLA DE ESPERA CURSO  8268
Nombre: dx
DNI: 3274583

----------------------------------------

COLA DE ESPERA CURSO  9445
----------------------------------------

COLA DE ESPERA CURSO  9989
----------------------------------------

ARBOL:
```

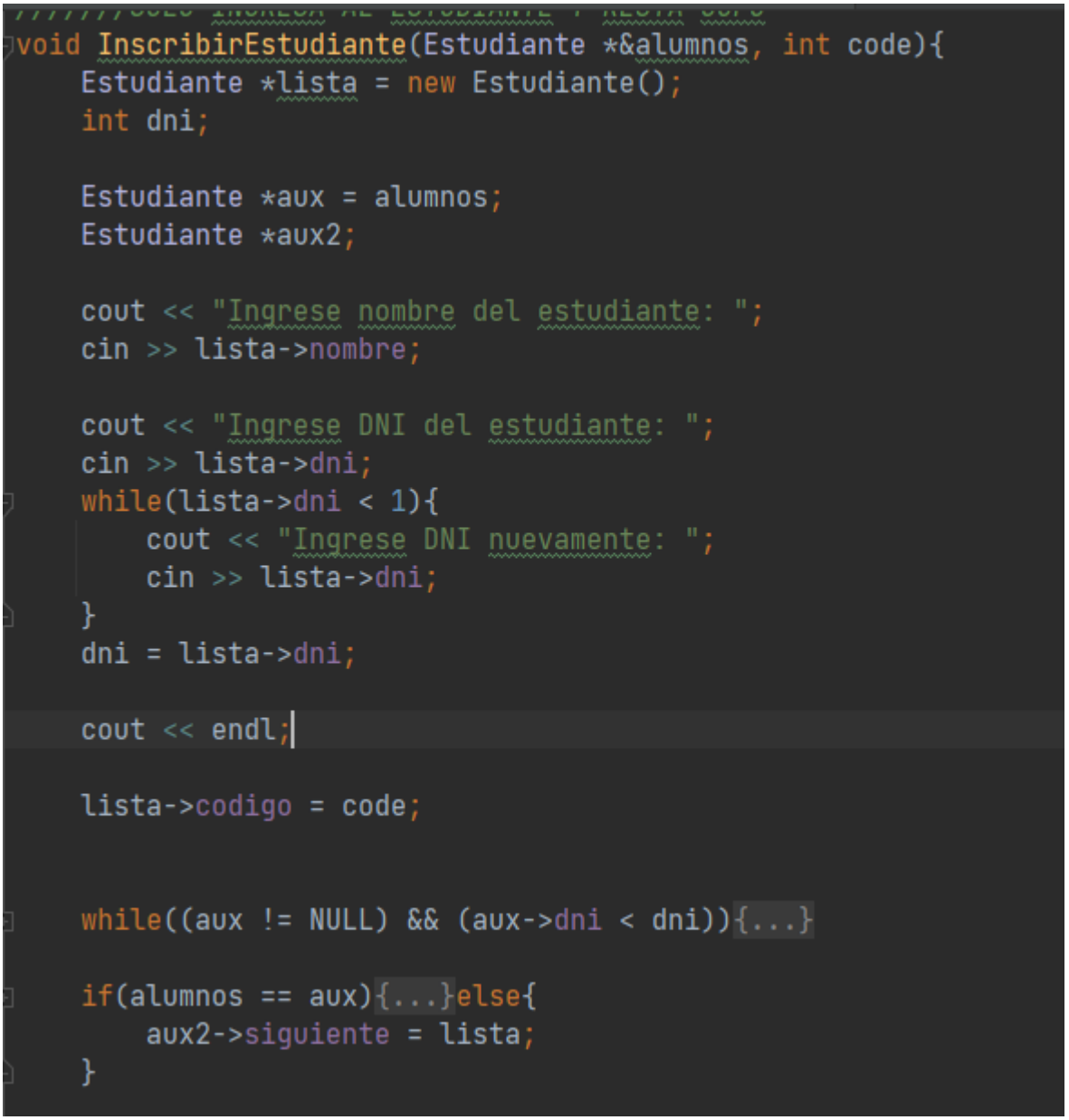
```
ARBOL:
DNI-> 44 - Curso-> 1024 - Nombre-> kal - Alumnos inscriptos-> 0
DNI-> 45 - Curso-> 1022 - Nombre-> sol - Alumnos inscriptos-> 5
DNI-> 1420412 - Curso-> 1854 - Nombre-> maja - Alumnos inscriptos-> 1
DNI-> 10198999 - Curso-> 649 - Nombre-> arnoldo - Alumnos inscriptos-> 0
DNI-> 11990117 - Curso-> 7047 - Nombre-> abena - Alumnos inscriptos-> 0
DNI-> 12645518 - Curso-> 8268 - Nombre-> khasan - Alumnos inscriptos-> 5
DNI-> 36190395 - Curso-> 2251 - Nombre-> mattis - Alumnos inscriptos-> 2
DNI-> 56291677 - Curso-> 1093 - Nombre-> caridad - Alumnos inscriptos-> 0
DNI-> 56469486 - Curso-> 5159 - Nombre-> titus - Alumnos inscriptos-> 0
DNI-> 62442788 - Curso-> 3986 - Nombre-> caridad - Alumnos inscriptos-> 0
DNI-> 64252083 - Curso-> 9989 - Nombre-> grozda - Alumnos inscriptos-> 0
DNI-> 88106696 - Curso-> 9445 - Nombre-> alexander - Alumnos inscriptos-> 0
DNI-> 93106106 - Curso-> 872 - Nombre-> loyd - Alumnos inscriptos-> 0

FIN DEL PROGRAMA
```

Funcionamiento del código

Menú para la Interfaz Usuario. 4 opciones distintas en base a lo ingresado por consola.

```
while(true){
    if(!error) {
        cout << "Ingrese n para: \n\n1: Inscribir a un estudiante.\n2: Baja de estudiante.\n3: Mostrar inscripciones.\n4: Salir del programa.\n\n";
        cin >> ingreso;
    }

    error = false;

    if (ingreso == 1) {...} else if (ingreso == 2) {...} else if (ingreso == 3) {...} else if (ingreso == 4) {
        for (int i = 0; i < 48; i++) {
            if (codigos[i].codigoDeCurso != -1) {
                cout << "COLA DE ESPERA CURSO " << " " << codigos[i].codigoDeCurso << endl;
                MostrarLista( alumnos: colaEspera, codigos,  pos: i);
                cout << "----------------------------------------\n\n";
            }
        }
        ArrayDocentes( docente: profesores,  idioma: idiomas, codigos);

        while(profesores[b].codigoDeCurso != -1){
            InsertarDocente( &: docentes,  profes: profesores,  pos: b);
            b++;
        }

        for(int i=0; i<48; i++) {
            if(codigos[i].codigoDeCurso != -1)
                ArchivosInscriptos( &: alumnos,  code: codigos[i].codigoDeCurso);
        }

        cout << "ARBOL:\n";
```

InscribirEstudiante crea un nodo nuevo cada vez que es llamada y pide por consola los datos a guardar en los campos.



```
void InscribirEstudiante(Estudiante *&alumnos, int code){
    Estudiante *lista = new Estudiante();
    int dni;

    Estudiante *aux = alumnos;
    Estudiante *aux2;

    cout << "Ingrese nombre del estudiante: ";
    cin >> lista->nombre;

    cout << "Ingrese DNI del estudiante: ";
    cin >> lista->dni;
    while(lista->dni < 1){
        cout << "Ingrese DNI nuevamente: ";
        cin >> lista->dni;
    }
    dni = lista->dni;

    cout << endl;

    lista->codigo = code;


    while((aux != NULL) && (aux->dni < dni)){...}

    if(alumnos == aux){...}else{
        aux2->siguiente = lista;
    }
```

EliminarEstudiante recibe un numero de documento como parámetro, lo busca (recorriendo la función siempre que no sea nula), y cuando encuentra una coincidencia la elimina, igualando la posición siguiente a siguiente->siguiente. Después elimina el nodo para no ocupar espacio innecesario en memoria. También incrementa y decrementa la cantidad de vacantes ocupadas y disponibles del curso en base a los estudiantes que se dan de alta o de baja.

```
void EliminarEstudiante(Estudiante *&lista, int dni, Estudiante *&colaEspera, Instituto codigos[]){

    Estudiante *nuevoNodo = new Estudiante();
    Estudiante *agregar = new Estudiante();
    Estudiante *auxCola = colaEspera;
    int dniAux, codigo, pos;

    codigo = BuscarCurso( &lista, dni);
    for(int i=0; i<48; i++){
        if(codigos[i].codigoDeCurso == codigo){
            pos = i;
            break;
        }
    }

    if(lista != NULL){
        Estudiante *anterior = NULL;
        Estudiante *aux_borrar = lista;

        while((aux_borrar != NULL) && (aux_borrar->dni != dni)){
            anterior = aux_borrar;
            aux_borrar = aux_borrar->siguiente;
        }

        if(aux_borrar == NULL){
            cout << "El elemento no existe";
        }else if(anterior == NULL){
            lista = lista->siguiente;
            delete aux_borrar;
```

```
            codigos[pos].cupo++;
            codigos[pos].alumnos--;
        }else{
            anterior->siguiente = aux_borrar->siguiente;
            delete aux_borrar;
            codigos[pos].cupo++;
            codigos[pos].alumnos--;
        }
    }

    Estudiante *aux = lista;
    Estudiante *aux2 = NULL;

    //PASAR DATOS
    while(auxCola != NULL){
        if(auxCola->codigo == codigo){
            nuevoNodo = auxCola;
            codigos[pos].cupo--;
            codigos[pos].alumnos++;
            break;
        }
        auxCola = auxCola->siguiente;
    }
    dniAux = nuevoNodo->dni;

    agregar->dni = nuevoNodo->dni;
    agregar->codigo = nuevoNodo->codigo;
    agregar->nombre = nuevoNodo->nombre;
    agregar->nivel = nuevoNodo->nivel;
```

```
    agregar->idioma = nuevoNodo->idioma;

    while((aux != NULL) && (aux->dni < dniAux)){
        aux2 = aux;
        aux = aux->siguiente;
    }

    if(lista == aux){
        lista = lista;
    }else{
        aux2->siguiente = agregar;
    }

    agregar->siguiente = aux;

    BorrarCola ( &: colaEspera,  dni: dniAux);
}
```

Esta función elimina al nodo Estudiante en cola de espera que se inscribió en el curso (porque alguien se dio de baja).

```
void BorrarCola(Estudiante *&lista, int dni){

    if(lista != NULL){
        Estudiante *auxborrar;
        Estudiante *anterior = NULL;
        auxborrar = lista;
        while((auxborrar != NULL) && (auxborrar->dni != dni)){
            anterior = auxborrar;
            auxborrar = auxborrar->siguiente;
        }
        if(auxborrar == NULL){
            cout << "El elemento no existe.";
        }else if(anterior == NULL){
            lista = lista->siguiente;
            delete auxborrar;
        }else{
            anterior->siguiente = auxborrar->siguiente;
            delete auxborrar;
        }
    }
}
```

MostrarLista crea una variable auxiliar y recorre la lista mientras no sea nula, mostrando el nombre y el documento guardados en cada caso.

```
void MostrarLista(Estudiante *alumnos, Instituto codigos[], int pos) {
    Estudiante *lista = alumnos;
    if (lista == NULL) {
        cout << "Soy una lista vacia :(\n" << endl;
    } else {
        while (lista != NULL) {
            if (lista->codigo == codigos[pos].codigoDeCurso) {
                cout << "Nombre: " << lista->nombre << endl;
                cout << "DNI: " << lista->dni << endl;
                cout << "\n";
            }
            lista = lista->siguiente;
        }
        lista = alumnos;
    }
}
```

InsertarDocente funciona con la misma lógica que **InscribirEstudiante**, pero va a crear un ARBOL ordenado por documento.

```
void InsertarDocente(Docente *&docentes, Instituto profes[], int pos){
    if(docentes == NULL){//ARBOL VACIO
        Docente *nuevoNodo = CrearDocente(profes, pos);
        docentes = nuevoNodo;
    }else{
        int valorRaiz = docentes->dni;//Valor de la raiz
        if(profes[pos].dni < valorRaiz){
            InsertarDocente( &: docentes->izq, profes, pos);

        }else{
            InsertarDocente( &: docentes->der, profes, pos);
        }
    }
}
```

Esta función realiza un recorrido **InOrden** del árbol, mostrando una lista de los docentes ordenada por número de documento. Por cada uno, a su vez, se muestra el curso que tiene a cargo, el nombre y la cantidad de alumnos inscriptos (que consiguieron vacante).

```
void InOrden(Docente *docentes){
    if(docentes == NULL){
        return;
    }else{
        InOrden( docentes: docentes->izq);
        cout << "DNI-> " << docentes->dni << " - ";
        cout << "Curso-> " << docentes->codigoDeCurso << " - ";
        cout << "Nombre-> " << docentes->nombre << " - ";
        cout << "Alumnos inscriptos-> " << docentes->alumnos;
        cout << endl;
        InOrden( docentes: docentes->der);
    }
}
```

La última función del código toma el valor entero del código de curso y lo pasa a arreglo de char, para después usarlo como nombre para crear un archivo por cada uno, con la información de los estudiantes.

```
void ArchivosInscriptos(Estudiante *&inscriptos, int code){
    Estudiante *lista = inscriptos;
    int aux, i=0;
    int code2=code;
    char x[12], datAux[4] = { [0]: '.', [1]: 'd', [2]: 'a', [3]: 't'};

    for(int a=0; a < size(x); a++){
        x[a] = 0;//Inicio a NULL
    }
    while(code2 > 0){
        aux = code2%10;//Guarda el ultimo digito de n en aux
        code2 /= 10;//Saca un digito a n
        x[i] = aux+48;//Guarda n+48 (valor de n en ASCII)
        i++;
    }
    strrev( Str: x);//Invierto arreglo

    for(int j=0; j<4; j++){
        x[i+j] = datAux[j];
    }

    FILE *f = fopen( Filename: x, Mode: "wb");
    while(lista != NULL) {
        if(lista->codigo == code){
            fwrite( Str: &lista, Size: sizeof(Estudiante), Count: 1, File: f);
        }
        lista = lista->siguiente;
    }
    lista = inscriptos;
    fclose( File: f);
}
```

Así quedan los archivos de cursos