TRABAJO PRÁCTICO ANUAL - Parte 1

ALGORITMOS Y ESTRUCTURA DE DATOS

INTEGRANTES

Ramiro Alonso Alvite

Gonzalo Agustin Marolda

Luciano Riente

Lara Irina Tartaglia

CURSO

K1022

FECHA DE ENTREGA

08/08/2022

EXPLICACIÓN DEL PROGRAMA:

Elegimos guardar los datos ingresados por teclado en un struct ya que el mismo engloba todo lo que contiene una clase (**Imagen 1**).
A lo largo del programa se hará uso de varias clases que tienen un número limitado de niveles e idiomas, 6 idiomas y 8 niveles, los cuales establecimos previamente con un *define* (**Imagen 2**). Como solo puede haber una clase por cada idioma y por cada nivel, elegimos guardar cada struct **Clase** en una matriz con una cantidad de filas igual a la cantidad de niveles y una cantidad de columnas igual a la cantidad de idiomas (**Imagen 3**).

```
struct Clase
{
    int codigo, cupos, dniProfe, nivel;
    char nombre[25];
};
```

Imagen 1

```
#define niveles 8
#define idiomas 6
```

Imagen 2

```
Clase matClases[niveles][idiomas];
```

Imagen 3

Se ingresa el código de curso(1022) y se indica al usuario la finalización de ingreso de datos con 0:

```
Ingresar codigo del curso (finaliza con 0): 1022
```

El código obtiene el dato ingresado por teclado y entra a un ciclo while de ingreso de datos que finaliza cuando el código ingresado sea 0.

```
cout << "Ingresar codigo del curso (finaliza con 0): ";
cin >> codigo;

while (codigo != 0)
```

---

*Se ingresa el idioma. En caso de ser **incorrecto**, le pide al usuario que ingrese uno válido :*

```
Ingresar idioma: Banana
Ingrese un idioma valido: Ingles
```

El idioma ingresado es guardado como un número ya que lo hace más sencillo de guardar y de buscar en la matriz. Para ello, se hace uso de la función *numeroDeIdioma()* la cual contiene un switch que asigna un número al idioma ingresado y, en el caso de ser ***incorrecto***, la función retornará -1 y le pedirá al usuario que ingrese un idioma válido.

```
cout << "Ingresar idioma: ";
cin >> idioma;
numIdioma = numeroDeIdioma(idioma);
while (numIdioma == -1)
{
    cout << "Ingrese un idioma valido: ";
    cin >> idioma;
    numIdioma = numeroDeIdioma(idioma);
}
```

```
int numeroDeIdioma(char i[])
{
    if (strcmpi(i, "ingles") == 0)
        return 0;
    else if (strcmpi(i, "frances") == 0)
        return 1;
    else if (strcmpi(i, "portugues") == 0)
        return 2;
    else if (strcmpi(i, "italiano") == 0)
        return 3;
    else if (strcmpi(i, "aleman") == 0)
        return 4;
    else if (strcmpi(i, "chino") == 0)
        return 5;
    else
        return -1;
}
```

---

*Se ingresa el nivel y, **si sobrepasa el límite de nivel** establecido, se le solicita al usuario que ingrese un nivel válido:*

```
Ingresar nivel (de 1 a 8) : 30
El nivel ingresado no es valido, ingrese otro 5
```

Se guarda el dato ingresado en la variable nivel y se verifica que dicha variable entre dentro del rango establecido de niveles, en el caso de haber ingresado un nivel ***incorrecto***, se le pedirá al usuario que ingrese otro.
Luego se verifica que el curso no exista utilizando el nivel y el idioma ya ingresados. Para ello, anteriormente se ejecuta la función *inicializarMatriz()* que carga todos los códigos de curso con 0, entonces, si el código del curso no es 0 en ese espacio de la matriz[nivel][idioma] significa que ya existe un curso y se le pedirá al usuario que ingrese todo nuevamente.
Finalmente, si se ingresa todo correctamente, se le asigna el nivel y el código ingresados al espacio de la matriz.

La razón detrás de la elección de guardar los datos de forma ordenada es que de esta forma se utiliza mucho menos código, optimizando el programa, y además, el trabajo de asignación y búsqueda es bastante más simple.

```
    cout << "Ingresar nivel (de 1 a 8) : ";
    cin >> nivel;
    while (nivel < 1 || nivel > 8)
    {
        cout << "El nivel ingresado no es valido, ingrese otro ";
        cin >> nivel;
    }

    if (m[nivel - 1][numIdioma].codigo != 0)
        cout << "Ya existe un curso" << endl;

} while (m[nivel - 1][numIdioma].codigo != 0);

m[nivel - 1][numIdioma].codigo = codigo;
m[nivel - 1][numIdioma].nivel = nivel;
```

Se ingresan los demás datos necesarios (cantidad de cupos del curso correspondiente, DNI y nombre del Profesor):

```
Ingresar cantidad de cupos disponibles: 50
Ingresar DNI del profesor: 33457986
Ingresar nombre del profesor: Jorge
```

Se guardan en la matriz los datos ingresados y se le vuelve a pedir el ingreso de cupos y DNI si el usuario no los ingresa correctamente.

```
cout << "Ingresar cantidad de cupos disponibles: ";
cin >> m[nivel - 1][numIdioma].cupos;
while (m[nivel - 1][numIdioma].cupos < 0)
{
    cout << "La cantidad de cupos ingresados no es valida, debe ser mayor a 0, ingrese otra ";
    cin >> m[nivel - 1][numIdioma].cupos;
}

cout << "Ingresar DNI del profesor: ";
cin >> m[nivel - 1][numIdioma].dniProfe;
while (m[nivel - 1][numIdioma].dniProfe < 0)
{
    cout << "Debe ingresar un DNI mayor a 0, ingrese otro ";
    cin >> m[nivel - 1][numIdioma].dniProfe;
}

cout << "Ingresar nombre del profesor: ";
cin >> m[nivel - 1][numIdioma].nombre;
```

Se ingresan dos cursos más y se finaliza con 0:

```
Ingresar codigo del curso (finaliza con 0): 1125
Ingresar idioma: Aleman
Ingresar nivel (de 1 a 8) : 8
Ingresar cantidad de cupos disponibles: 60
Ingresar DNI del profesor: 33795685
Ingresar nombre del profesor: Esteban
Ingresar codigo del curso (finaliza con 0): 1166
Ingresar idioma: Ingles
Ingresar nivel (de 1 a 8) : 8
Ingresar cantidad de cupos disponibles: 30
Ingresar DNI del profesor: 33864579
Ingresar nombre del profesor: Saul
Ingresar codigo del curso (finaliza con 0): 0
```

DATOS INGRESADOS:

Código	Idioma	Nivel	Cupos	DNI	Nombre
1022	Inglés	5	50	33457986	Jorge
1125	Alemán	8	60	33795685	Esteban
1166	Inglés	8	30	33864579	Saul

Se informa la cantidad de cursos que se dictan por idioma:

```
La cantidad de cursos que se dictaran del idioma Ingles es: 2
La cantidad de cursos que se dictaran del idioma Frances es: 0
La cantidad de cursos que se dictaran del idioma Portugues es: 0
La cantidad de cursos que se dictaran del idioma Italiano es: 0
La cantidad de cursos que se dictaran del idioma Aleman es: 1
La cantidad de cursos que se dictaran del idioma Chino es: 0
```

Se recorre la matriz primero por columnas (**idiomas**) y se inicializa en cero un contador de la cantidad de cursos de cada idioma.
Luego se recorre por cada fila (**niveles**) y mediante un if se compara si el código dentro de dicha matriz es distinto de cero, significa que se ha ingresado un código, por lo tanto se incrementa el contador de cantidad de cursos.

```
void punto2(Clase matClase[niveles][idiomas])
{
    for (int i = 0; i < idiomas; i++)
    {
        int cantCursos = 0;
        for (int j = 0; j < niveles; j++)
            if (matClase[j][i].codigo != 0)
                cantCursos++;
        cout << "La cantidad de cursos que se dictaran del idioma " << idiomaString(i) << " es: " << cantCursos << endl;
    }
    cout << endl;
}
```

Se informan los niveles en los que no se dictarán clases:

```
En el nivel 1 No se dictaran clases
En el nivel 2 No se dictaran clases
En el nivel 3 No se dictaran clases
En el nivel 4 No se dictaran clases
En el nivel 6 No se dictaran clases
En el nivel 7 No se dictaran clases

Process returned 0 (0x0)   execution time : 281.277 s
Press any key to continue.
```

Se crea una bandera llamada “v”.
Se recorre la matriz primero por filas (**niveles**) y se inicializa en cero un contador de la cantidad de cursos de cada nivel. En cada vuelta del for, se transforma a la bandera v en **true**.
Luego se recorre por columna (**idiomas**) y mediante un if se evalúa si la casilla tiene un valor distinto de cero (es decir, si tiene un curso o más). En caso de no haber columna con curso en esa fila la bandera mantendrá su valor y el if se ejecutará indicando el nivel en cual no se dictará clases.
Caso contrario, v pasaría a ser **false** y el if sería salteado.

```
void punto4(Clase matClase[niveles][idiomas])
{
    bool v = false;
    for (int i = 0; i < niveles; i++)
    {
        v = true;
        for (int j = 0; j < idiomas; j++)
        {
            if (matClase[i][j].codigo != 0)
            {
                v = false;
                break;
            }
        }
        if (v)
            cout << "En el nivel " << i + 1 << " No se dictaran clases" << endl;
    }
}
```

Se ingresan un curso en todos los niveles:

```
Ingresar codigo del curso (finaliza con 0): 1000
Ingresar idioma: Ingles
Ingresar nivel (de 1 a 8) : 1
Ingresar cantidad de cupos disponibles: 50
Ingresar DNI del profesor: 33457985
Ingresar nombre del profesor: Jorge
Ingresar codigo del curso (finaliza con 0): 1001
Ingresar idioma: Ingles
Ingresar nivel (de 1 a 8) : 2
Ingresar cantidad de cupos disponibles: 60
Ingresar DNI del profesor: 33457985
Ingresar nombre del profesor: Kim
Ingresar codigo del curso (finaliza con 0): 1002
Ingresar idioma: Ingles
Ingresar nivel (de 1 a 8) : 3
Ingresar cantidad de cupos disponibles: 90
Ingresar DNI del profesor: 33758446
Ingresar nombre del profesor: Alex
Ingresar codigo del curso (finaliza con 0): 1003
Ingresar idioma: Ingle
Ingrese un idioma valido: Ingles
Ingresar nivel (de 1 a 8) : 4
Ingresar cantidad de cupos disponibles: 60
Ingresar DNI del profesor: 22422422
Ingresar nombre del profesor: Marcelo
Ingresar codigo del curso (finaliza con 0): 1004
Ingresar idioma: Ingles
Ingresar nivel (de 1 a 8) : 5
Ingresar cantidad de cupos disponibles: 70
Ingresar DNI del profesor: 22357689
Ingresar nombre del profesor: Lara
Ingresar codigo del curso (finaliza con 0): 1005
Ingresar idioma: Ingles
Ingresar nivel (de 1 a 8) : 6
Ingresar cantidad de cupos disponibles: 75
Ingresar DNI del profesor: 99854637
Ingresar nombre del profesor: Alfredo
Ingresar codigo del curso (finaliza con 0): 1006
Ingresar idioma: ingles
Ingresar nivel (de 1 a 8) : 7
Ingresar cantidad de cupos disponibles: 32
Ingresar DNI del profesor: 22147536
Ingresar nombre del profesor: Sofia
Ingresar codigo del curso (finaliza con 0): 1007
Ingresar idioma: Ingles
Ingresar nivel (de 1 a 8) : 8
Ingresar cantidad de cupos disponibles: 60
Ingresar DNI del profesor: 23456789
Ingresar nombre del profesor: Maximo
Ingresar codigo del curso (finaliza con 0): 0
```

Se informa que dicho idioma tiene cursos en todos sus niveles:

```
El idioma Ingles tiene cursos en los ocho niveles
```

Se recorre la matriz por columna (**idiomas**), y se compara con un if si hay un cero en el código dentro de dicha columna, de ser así se iguala la j a los niveles para finalizar el ciclo. Se compara con un nuevo if, y en el caso de que la j sea igual a los niveles quiere decir que recorrió todo el **ciclo for** anterior

(no encontró ningún código cero, todos estaban con su código determinado), por lo tanto el idioma correspondiente tiene cursos en los ocho niveles.

```
void punto3(Clase matClase[niveles][idiomas])
{
    for (int i = 0; i < idiomas; i++)
    {
        int j = 0;
        for (j; j < niveles; j++)
            if (matClase[j][i].codigo == 0)
                j = niveles;
        if (j == niveles)
            cout << "El idioma " << idiomaString(i) << " tiene cursos en los ocho niveles" << endl;
    }
    cout << endl;
}
```

Teniendo en cuenta que la columna 0 es respectivamente el idioma Inglés, la columna 1, el idioma Francés, y así sucesivamente; Creamos una función denominada **idiomaString**, la cual devuelve el nombre del idioma dependiendo del número la columna en la que esté posicionado el **ciclo for**.

```
string idiomaString(int i)
{
    string idioma;
    switch (i)
    {
    case 0:
        idioma = "Ingles";
        break;
    case 1:
        idioma = "Frances";
        break;
    case 2:
        idioma = "Portugues";
        break;
    case 3:
        idioma = "Italiano";
        break;
    case 4:
        idioma = "Aleman";
        break;
    case 5:
        idioma = "Chino";
        break;
    }
    return idioma;
}
```

Se toma la decisión de generar una copia de la matriz que se guardará en oMatClase utilizando la función *copiarMatriz()*. Esto se hace para que, al realizar un burbujeo que ordene las clases por código de curso, no se desordene nuestra matriz de clases principal ya que al estar ordenada por nivel y por idioma nos servirá en los puntos posteriores.
A partir de esta copia, ordenada por código utilizando la función de *burbujeo()*, se ingresa en un doble ciclo for que le asigna a la variable idioma un nombre nuevo, generando así un nuevo archivo, dependiendo del idioma actual del for, definido por k. Por cada archivo de idioma se recorre la matriz por nivel y, si la clase existe, se guarda en el archivo. Finalmente, se cierra el archivo.

```
void punto1(Clase matClase[niveles][idiomas])
{
    char idioma[20];
    Clase oMatClase[niveles][idiomas];
    copiarMatriz(oMatClase, matClase);
    burbujeo(oMatClase);
    for (int k = 0; k < idiomas; k++)
    {
        switch (k)
        {
        case 0:
            strcpy(idioma, "Ingles.dat");
            break;
        case 1:
            strcpy(idioma, "Frances.dat");
            break;
        case 2:
            strcpy(idioma, "Portugues.dat");
            break;
        case 3:
            strcpy(idioma, "Italiano.dat");
            break;
        case 4:
            strcpy(idioma, "Aleman.dat");
            break;
        case 5:
            strcpy(idioma, "Chino.dat");
            break;
        }

        FILE *archCurso = fopen(idioma, "wb");
        if (archCurso == NULL)
            cout << "ERROR" << endl;
        else
        {
            for (int i = 0; i < niveles; i++)
                if (oMatClase[i][k].codigo != 0)
                    fwrite(&oMatClase[i][k], sizeof(Clase), 1, archCurso);
        }
        fclose(archCurso);
    }
    cout << endl;
}
```

Utilizamos la función burbujeo para ordenar por código de curso:

```
void burbujeo(Clase v[niveles][idiomas])
{
    Clase aux;
    bool cambio;
    for (int k = 0; k < idiomas; k++)
    {
        unsigned i = 1;
        do
        {
            cambio = false;
            for (int j = 0; j < niveles - i; j++)
            {
                if (v[j][k].codigo > v[j + 1][k].codigo)
                {
                    aux = v[j][k];
                    v[j][k] = v[j + 1][k];
                    v[j + 1][k] = aux;
                    cambio = true;
                }
            }
            i++;
        } while (i < niveles && cambio);
    }
}
```