## Documentación Examen ED-Unidad 2.

**Alumno(a):** Jiménez Mayoral Gloria Alejandra 17212146-ISC.

**Ejercicio 3:** Hacer un programa que lleve el control de versiones de un proyecto en una empresa, la estructura de datos debe controlar el número de migraciones realizadas en el proyecto. Utilizar una pila para introducir las migraciones una por una y obtener las migraciones una por una empezando por la migración más actual y terminando por la migración más antigua.

```
1   #include <iostream>
2   #include<stdlib.h>
3   #include<string>
4   #define TAM 10000
5   using namespace std;
```

Al inicio del programa se implementan las librerías iostream, para poder leer (cout) e introducir datos (cin) así como el uso de lenguaje estándar using namespace std y la librería stdlib.h para poder usar funciones como system(“CLS”) para limpiar la pantalla y la librería string para poder introducir cadenas de texto. Por último se define el tamaño TAM del array.

```
6   class Migracion
7   {
8       string pila[TAM];
9       int  index;
10      public:
11          Pila()
12          {
13              index = -1;
14          }
```

A continuación se declara la clase *Migración* , donde se declara el arreglo que lleva por nombre pila con el tamaño TAM y un índice index de tipo entero. A su vez, se declara un constructor donde se inicializará el index en -1.

Función **push**

```
15          void push (string fecha)
16          {
17              if(index>TAM)
18              {
19                  system("CLS");
20                  cout<<"No hay espacio para agregar mas migraciones"<<endl;
21                  return;
22              }
23              pila[++index]=fecha;
24              cout<<"Migracion "<<fecha<<" ingresada"<<endl;
25          }
```

Se comienzan a implementar los métodos, el primero es el método push que en este caso, será para ingresar una nueva fecha de la migración. Recibe un string llamado *fecha*. Cuenta con la condición para evitar un desbordamiento u overflow, en caso de que el index apunte más allá del TAM máximo del array. Si esto sucede, se le imprimirá en pantalla que ya no hay espacio para más migraciones, o sea ya no hay espacio para ingresar otro elemento al array. Y regresa al usuario a las otras funciones. Si la condición no se cumple, se pasa a acomodar la fecha y se aumenta el index para recorrerlo a la siguiente posición y así sucesivamente.

Función **peekAll**

```
26          void peekAll()
27          {
28              if(index<0)
29              {
30                  system("CLS");
31                  cout<<"No hay migraciones para mostrar"<<endl;
32                  return;
33              }
34              for(int i=index;i>=0;i--)
35              {
36                  cout<<"\n";
37                  cout<<"Numero de migracion"<<"["<<i<<"]"<<" Fecha:"<<pila[i]<<" ";
38              }
39          }
40  };
```

El último método es peekAll que servirá para poder mostrar todas las migraciones ingresadas y su posición o número de éstas. Cumple también con la condición para evitar un underflow en caso de que no haya ninguna migración guardada. O sea, no hay ningún elemento para sacar del array. Y regresa al usuario a las otras opciones que se verán más adelante. En caso contrario, muestra todas las migraciones mediante un ciclo for, donde la variable del ciclo será igual al index de la fila hasta que sea mayor o igual que 0 o el fin, e irá disminuyendo para ir mostrando cada una de las posiciones, puesto que se mostrará desde la última migración ingresada hasta la más antigua, siguiendo así la estructura LIFO (Last in-first out) de las pilas.
Se termina la clase Migración.

```
41  int main()
42  {
43      int op;
44      string op2;
45      Migracion pil;
46      cout<<"                                    MIGRACIONES DEL PROYECTO"<<endl;
47      do
48      {
49          cout<<"1.Ingresar una nueva migracion \n2.Mostrar migraciones \n3.Salir \nTu opcion es:  ";
50          cin>>op;
```

En la función principal main(), se declaran tres variables, las primera de tipo entero op que servirá para almacenar las respuestas del usuario. Y la segunda op2 de tipo string como auxiliar para ingresar las fechas. Y de tipo Migración pil para poder llamar a las funciones en este main. Se imprime en pantalla el nombre del programa “MIGRACIONES DEL PROYECTO”.
A continuación se abren las llaves de un ciclo do-while para llevar a cabo las funciones hasta que sea necesario. Se imprime en pantalla tres opciones para el usuario:

- Número 1: para ingresar una nueva migración (método push ).
- Número 2: para mostrar el total de migraciones ingresadas (método peek ).
- Número 3: para salir (terminar el programa).

La respuesta del usuario se guardará en la variable op.

```
51          switch(op)
52          {
53              case 1:
54                  system("CLS");
55                  cout<<"Ingresar fecha de la migracion (dia-mes-anio): ";
56                  cin>>op2;
57                  system("CLS");
58                  pil.push(op2);
59              break;
60              case 2:
61                  system("CLS");
62                  cout<<"Total de migraciones"<<endl;
63                  pil.peekAll();
64                  cout<<"\n"<<endl;
65              break;
66              case 3:
67              break;
68          }
69      }while(op!=3);
70  }
```

Las opciones para el usuario se llevarán a cabo gracias a una función switch.

Opción 1: En caso de seleccionar la opción 1 (método push), se le solicita al usuario que ingrese la fecha de la migración la cual se guarda en la variable auxiliar op2. Después se llama al método push de la clase y se ejecuta todo el proceso.

Opción 2: la opción desplegará en orden en pantalla todas las migraciones ingresadas por el usuario, ejecutando la el método peekAll de la clase.

Opción 3: la opción 3 termina permite salir del programa y terminarlo.

Se cierran las llaves y termina la función principal main().

Fin del programa.