Documentación Examen ED-Unidad 2.

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

Ejercicio 4: hacer un programa que simule la fila de clientes de una tienda de supermercado, considerando que solo hay una caja que está activa. La fila solo puede tener como máximo 5 clientes.

```
1  #include<iostream>
2  #include<stdlib.h>
3  #define TAM 5
4  using namespace std;
5  class Fila
6  {
7      int cola[TAM];
8      int fin=0;
9      int frente=0;
```

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. Por último se define el tamaño *TAM* que será igual a 5 por los cinco clientes admitidos; esto será para el tamaño del array.
Se declara la clase *Fila*, la cual posee los siguientes atributos: la declaración del array llamado *cola* con tamaño *TAM* (o cinco) y variables de tipo entero *fin* y *frente* con valor de cero.

Función ***push***

```
10     public:
11         void push(int cliente)
12         {
13             if(fin==TAM)
14             {
15                 system("CLS");
16                 cout<<"Cliente "<<cliente<<", ya no hay lugar en la fila :("<<endl;
17                 return;
18             }
19             system("CLS");
20             cola[fin++]=cliente;
21             cout<<"El cliente numero "<<cliente<<" entro a la fila."<<endl;
22         }
```

A continuación se comienzan a implementar los métodos, el primero es el método *push* que en este caso, será para ingresar a la fila del supermercado. Recibe un entero llamado *cliente* que indica el número del mismo. Cuenta con la condición para evitar un desbordamiento u overflow, en caso de que *fin* ya sea igual al tamaño *TAM* máximo del array. Si esto sucede, se le imprimirá en pantalla que “ya no hay lugar en la fila”, ya no hay espacio para ingresar otro elemento en el array. Y regresa al usuario (o cliente) a las otras funciones.
Si la condición no se cumple, se pasa a acomodar al *cliente* a la fila o array y se aumenta el *fin* del array para ir llenando los espacios.

Función ***pop***

```
23      void pop()
24      {
25          if(frente==fin)
26          {
27              system("CLS");
28              cout<<"Parece que no hay nadie en la fila!"<<endl;
29              return;
30          }
31          system("CLS");
32          cout<<"El cliente numero "<<cola[frente]<<" salio de la fila"<<endl;
33          frente++;
34      }
```

El siguiente método es *pop*, el cual será para salir de la fila. Éste cumple con una condición para evitar un underflow en caso de que no haya ningún cliente en la fila. Si el *frente* de la fila es igual que *fin*, se imprime en pantalla que “no hay nadie en la fila”, no hay ningún elemento para sacar del array. Y regresa al usuario a las otras opciones que se verán más adelante.
Si la condición no se cumple, se procederá a eliminar al primer elemento del array ya que fue el primero en ingresar; o bien, se atendió al primer cliente y ya puede salir de la fila. Esto debido al seguimiento de la estructura FIFO (*first-in, first-out*) de las colas. El *frente* de la fila aumenta para ir recorriendo a los otros clientes o elementos en el array.

Función **peek**

```
35      void peek()
36      {
37          if(frente==fin)
38          {
39              system("CLS");
40              cout<<"Parece que no hay nadie en la fila!"<<endl;
41              return;
42          }
43          system("CLS");
44          cout<<"Los siguientes clientes estan en la fila "<<endl;
45          for(int i=frente;i<fin;i++)
46          {
47              cout<<"Posicion: "<<i<<" Cliente: "<<cola[i]<<endl;
48          }
49      }
50  };
```

El último método es *peek* que servirá para poder mostrar en este caso, a todos los clientes de la fila y su posición. Cumple también con la condición para evitar un underflow en caso de que no haya ningún cliente en la fila. 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 a los clientes mediante un ciclo for, donde la variable del ciclo será igual al *frente* de la fila hasta que sea menor que *fin* e irá aumentando para ir mostrando cada una de las posiciones. Se imprime en pantalla posición y número de cliente.
Se termina la clase Fila.

```
51  int main()
52  {
53      int op;
54      int op1;
55      Fila co;
56      do
57      {
58      cout<<"1.Pagar(entrar a la fila). \n2.Salir de la fila. \n3.Ver quienes estan en la fila. \n4.Irte a casa(salir del mercado)."<<endl;
59      cout<<"\nQue vas a hacer? ";
60      cin>>op;
```

En la función principal main(), se declaran dos tres variables, las primeras de tipo entero *op* y *op1* que servirán para almacenar las respuestas del usuario. Y de tipo Fila *co* para poder llamar a las funciones en este main. 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 cuatro opciones para el usuario:

- Número 1: para ingresar a la fila (método *push*).
- Número 2: para salir de la fila (método *pop*).
- Número 3: para mostrar a los clientes de la fila (método *peek*).
- Número 4: para salir del supermercado (terminar el programa).

La respuesta del usuario se guardará en la variable *op*.

```
61      switch(op)
62      {
63          case 1:
64              system("CLS");
65              cout<<"Que numero de cliente entro a la fila? ";
66              cin>>op1;
67              co.push(op1);
68          break;
69          case 2:
70              system("CLS");
71              co.pop();
72          break;
73          case 3:
74              system("CLS");
75              co.peek();
76          break;
77          case 4:
78          break;
79      }
80      }while(op!=4);
81  }
```

Las opciones para el usuario se llevarán a cabo gracias a una función switch. En caso de seleccionar la opción 1 (método *push*), se le pregunta al usuario que número de cliente entró a la fila y la variable auxiliar *op1* almacenará este dato. Después se llama al método *push* de la clase y se ejecuta todo el proceso.

La opción 2 (método *pop*) llama al método de la clase y se ejecutan de igual forma todos los procesos. Igual el caso 3 ejecuta el método *peek* de la clase.
Finalmente, el caso 4 simplemente termina el programa.
Se cierran las llaves y termina la función principal main().
Fin del programa.