Instituto Tecnológico de Tijuana

T12: Método peek en pilas.

Materia:

Estructura de Datos

Profesor(a):

Ray Brunett Parra Galaviz

Alumno(a):

Jiménez Mayoral Gloria Alejandra – 17212146

Fecha:

26 de septiembre de 2018

Pila: método peek.

```
1  #include <iostream>
2  #include<stdlib.h>
3  using namespace std;
```

Las líneas **1** y **2** implementan las librerías: iostream, para poder utilizar operaciones de entrada y salida. Y stdlib para poder utilizar funciones como system("CLS") para limpiar la pantalla. La línea **3** implementa el namespace para el uso de las palabras reservadas: cout (para imprimir datos) y cin (para ingresar datos).

```
4  class Pila
5  {
6      int arraypi[5], index;
7      public:
8      Pila()
9      {
10         index = -1;
11     }
12     void push(int x)
13     {
14         if(index>5)
15         {
16             system("CLS");
17             cout<<"Pila llena!! No puedes agregar mas elementos"<<endl;
18             return;
19         }
20         arraypi[++index]=x;
21         cout<<"Elemento "<<x<<" ingresado"<<endl;
22     }
```

A partir de la línea **4** hasta la línea **70**, se crea la clase Pila que contiene un arreglo con el tamaño de la pila de tipo entero y un índice del mismo tipo. Dentro de la clase, se crea un constructor y los métodos que se utilizarán, todos públicos. En el constructor (línea **8**) se inicializa el valor del índice en -1 para considerarlo como un inicio cero. El método push se utilizará para introducir los elementos a la pila.

A continuación, se crearán tres métodos **peek** los cuales realizarán la misma función de imprimir los elementos en pantalla, pero con algunas variaciones.

peek

```
24         void peek() //ultima opcion del usuario
25         {
26             if(index<0)
27             {
28                 system("CLS");
29                 cout<<"Pila vacia"<<endl;
30                 return;
31             }
32             cout<<"El ultimo elemento ingresado es: "<<arraypi[index]<<endl;
33         }
```

Para este primer método, se imprimirá en pantalla el último elemento que se introdujo por el teclado. Cuenta con una condicional en caso de que la pila esté vacía; esto evitará un error en el programa conocido como underflow. Si la condición no se cumple, se procederá al método de imprimir, el cual considera la posición del índice para indicar el elemento (línea **32**).

userpeek

```
34          void userpeek() //eleccion del usuario
35          {
36              int op;
37              system("CLS");
38              cout<<"Escoge la posicion del elemento que deseas ver"<<endl;
39              cout<<"[0]-[1]-[2]-[3]-[4]-[5]"<<endl;
40              cout<<"Tu opcion es: ";
41              cin>>op;
42              if(index < op)
43              {
44                  system("CLS");
45                  cout<<"Posicion vacia de la pila"<<endl;
46                  return;
47              }
48              if(op > 5)
49              {
50                  system("CLS");
51                  cout<<"Seleccion fuera de rango"<<endl;
52                  return;
53              }
54              cout<<arraypi[op];
55              return;
56          }
```

El segundo método **userpeek** tiene como objetivo imprimir el elemento de la posición que el usuario seleccione. Primero se va a declarar una variable de tipo entero que funcionará para guardar la opción de posición del usuario. Se le imprimirá en pantalla las posiciones existentes. Cuenta con una condicional (línea **42**) en caso de que su opción señale una posición vacía.

La otra condición es para cuando el usuario seleccione una posición inexistente (línea **48**), es decir, mayor al límite de posiciones de la pila.

Si las condiciones no se cumplen, se procederá a imprimir en pantalla la opción de la posición que escogió el usuario.

allpeek

Este último método imprime en pantalla todos los elementos de la pila.

Cuenta con la misma condicional en caso de que la pila esté vacía.

Si la condición no se cumple, se procederá a imprimir los elementos, los cuales se manejarán con un bucle for para ir mostrando cada uno de los elementos de la pila.

```
57          void Allpeek()
58          {
59              if(index<0)
60              {
61                  system("CLS");
62                  cout<<"Pila vacía"<<endl;
63                  return;
64              }
65              for(int i=index;i>=0;i--)
66              {
67                  cout<<arraypi[i]<<" ";
68              }
69          }
70      };
```

```
71  int main()
72  {
73      int op, op2, op3;
74      Pila pil;
75      do
76      {
77          cout<<"1.Push(ingresar un elemento) \n2.Peek(mostrar los elementos) \n3.Salir \nTu opcion es:  ";
78          cin>>op;
79          switch(op)
80          {
81              case 1:
82                  system("CLS");
83                  cout<<"Ingresa un elemento: ";
84                  cin>>op2;
85                  system("CLS");
86                  pil.push(op2);
87              break;
```

En la línea **71** del código, comienza la función principal main, en la cual se declaran variables de tipo entero para ser utilizadas dentro de un método switch. Y también el objeto para el uso de la clase Pila. La función switch (línea **79**) funciona como un menú y cuenta con dos opciones a elegir por el usuario: la número 1 para ingresar los elementos a la pila por el método push.

```
88          case 2:
89              system("CLS");
90              cout<<"1.Peek(ultimo elemento) \n2.userPeek(escoge un elemento) \n3.Allpeek(todos los elementos) \nTu opcion es: ";
91              cin>>op3;
92              switch(op3)
93              {
94                  case 1:
95                  system("CLS");
96                  pil.peek();
97                  cout<<"\n"<<endl;
98                  break;
99                  case 2:
100                 system("CLS");
101                 pil.userpeek();
102                 cout<<"\n"<<endl;
103                 break;
104                 case 3:
105                 system("CLS");
106                 cout<<"Los elementos de la pila son: ";
107                 pil.Allpeek();
108                 cout<<"\n"<<endl;
109                 break;
110             }
111         break;
112         case 3:
113         break;
114     }
115 }while(op!=3);
116 return 0;
```

La opción dos, funciona para los métodos **peek** anteriores. Primero, se agregará una nueva función switch que servirá de submenú, esto es para poder elegir de entre las tres opciones de **peek** de forma más organizada. En el submenú (línea **92**) se tienen tres opciones para cada uno de los métodos en la clase.

- **Peek**: en la línea 94.
- **Userpeek:** en la línea 99.
- **Allpeek:** en la línea 104.

Con el objeto creado, se llama a los métodos declarados en la clase en las opciones.

Por último, se tiene la opción en el switch principal para terminar el programa. El proceso se repetirá hasta que se salga de éste.

Fin del programa.