Universidad de Los Andes
Escuela de Ingeniería de Sistemas
Departamento de Computación

PROGRAMACIÓN 2

Clase 10

Junior Altamiranda
altamira@ula.ve

AGENDA

Programación Orientada a Objetos (POO)

- Sobrecarga de operadores
- TDA Cadena
- Clases Paramétricas

Sobrecarga de operadores

Mecanismo a través del cual, operadores estándar de C++ son usados para otras operaciones de clase.
Los operadores son los que ya existen: No se pueden crear nuevos operadores
Todos los operadores pueden ser sobrecargados, excepto:

., ?:, ::, .*

Un ejemplo de un operador “sobrecargado” en C es <<
Ni la precedencia, ni la asociatividad (orden), ni la cantidad de operandos de un operador, pueden ser modificados

Sobrecarga de operadores

Implementación:

```
class algo {
   public:
         int operator+(int b);
};

int algo::operator+(int b) {
     . . .
}
```

TDA Cadena
(Características a incluir)

- Propiedades
  - Longitud
  - Contenido
- Métodos
  - Constructores
  - Destructor
  - Comparar la cadena con otra cadena
  - Buscar texto en cadena
  - Substraer: Obtener cierta cantidad de caracteres, a partir de cierta posición
  - Concatena otra cadena
  - Asignación de un nuevo valor

Ejercicio: El TDA Cadena

- Deseamos definir y construir el TDA Cadena
- Una Clase cadena; que facilite la manipulación de cadena de caracteres en C++, añadiendo métodos y propiedades que nos permitan ahorrar tiempo y facilite nuestro trabajo
- La cadena, la veremos como una secuencia de caracteres

Ejercicio

Agregar un método a la clase Cadena llamado “invertir”, que coloque en orden inverso las letras de la Cadena. Por ejemplo, si c1 es un objeto Cadena que contiene el texto “casa” y aplicamos c1.invertir(), el texto de c1 sería ahora “asac”. Realizar la sobrecarga del operador ! en la clase Cadena para que reallice la función de invertir el texto de un objeto cadena

Ejercicio

Agregar un método a la clase Cadena llamado “mesh”, que permita mezclar las letras, en orden, de un texto pasado como parámetro, en una cadena. Por ejemplo, si c1 es un objeto Cadena que contiene el texto “cuadro” y aplicamos c1.mesh(“blanco”), el texto de c1 debería pasar a ser “cbulaadnrcoo”
Realizar la sobrecarga del operador *= en la clase Cadena para que realice la función anterior.

Clases paramétricas (Plantillas)

Poderosa característica de reutilización de software de C++, que permite la definición de categorías o grupos relacionados de clases

Las clases son definidas para distintos tipos de argumentos

Esta forma de programar, se conoce como **Programación Genérica**

Para definir una clase como plantilla, se debe colocar (Antes de la definición de la clase):

template <class T>

Clases paramétricas (Plantillas)

Ejemplo:

```
template <class T>
class Par  {
      private:
            T primero;
            T segundo;
      public:
            Par();
            Par(T pri, T seg);
            void setPrimero(T pri);
            void setSegundo(T seg);
            T getPrimero();
            T getSegundo();
};
```

Clases paramétricas (Plantillas)

- El instanciar una plantilla, indicando un tipo de dato dado, se denomina "**especializar**" la clase.
- Toda función miembro debe llevar el prefijo: **template <class T>**
- Toda función miembro debe llevar, como nombre de la clase; el <T>. Antes de los ::
- Ver ejemplo de clase Par
- Es posible definir más de un tipo para la especialización de la plantilla: **template <class T1, class T2>**

Clases paramétricas (Plantillas)

- **Ejemplo**

TDA PILA

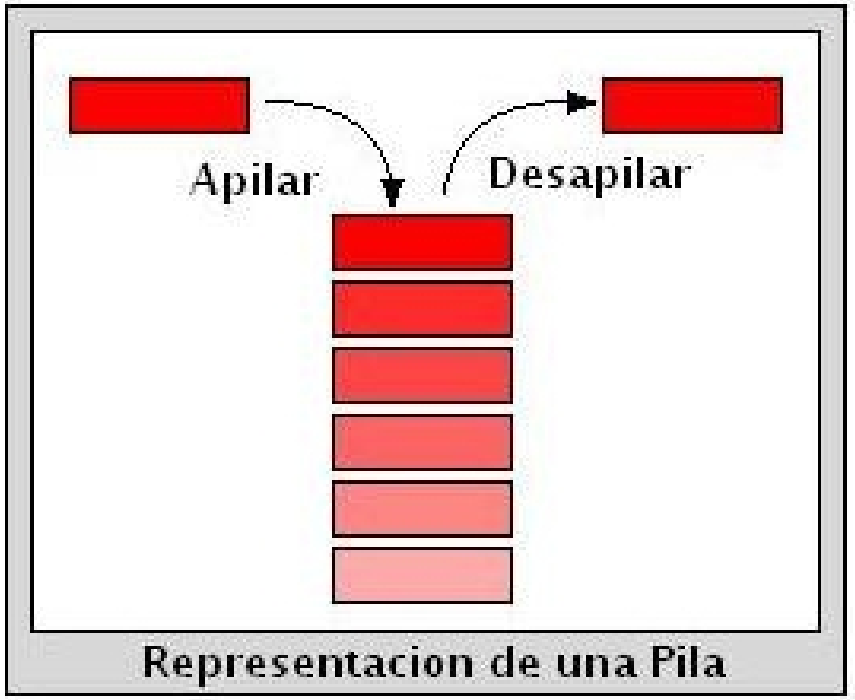
PILA: Una pila (stack en inglés) es una estructura de datos en la que el modo de acceso a sus elementos es de tipo LIFO (Last In First Out, último en entrar, primero en salir) que permite almacenar y recuperar datos
Para el manejo de los datos se cuenta con dos operaciones básicas: apilar (**push**), que coloca un objeto en la pila, y retirar (o desapilar, **pop**), que retira el último elemento apilado.
Las pilas suelen emplearse en los siguientes contextos:

- Evaluación de expresiones en notación postfija
- Reconocedores sintácticos
- Implementación de recursividad.

TDA PILA

Funciones:

- Constructor
- Destructor
- Push (Apilar)
- Pop (Desapilar)
- Tope
- Vaciar



Ejercicios

- Realizar la implementación de la clase Pila
- Implementar la clase Pila de forma paramétrica