Algoritmos y Estructuras de Datos II

Pilas y su implementación en C++

Dr. Edgard I. Benítez-Guerrero

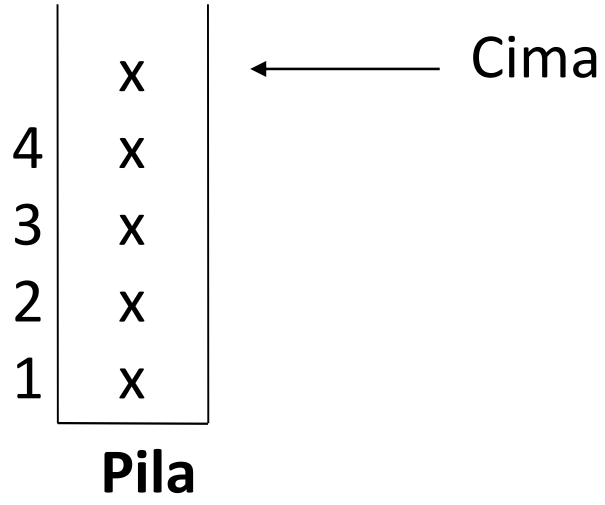
cursofei@gmail.com

Contenido

- Definición
- Operaciones
- Implementación estática en C++
- Implementación dinámica en C++

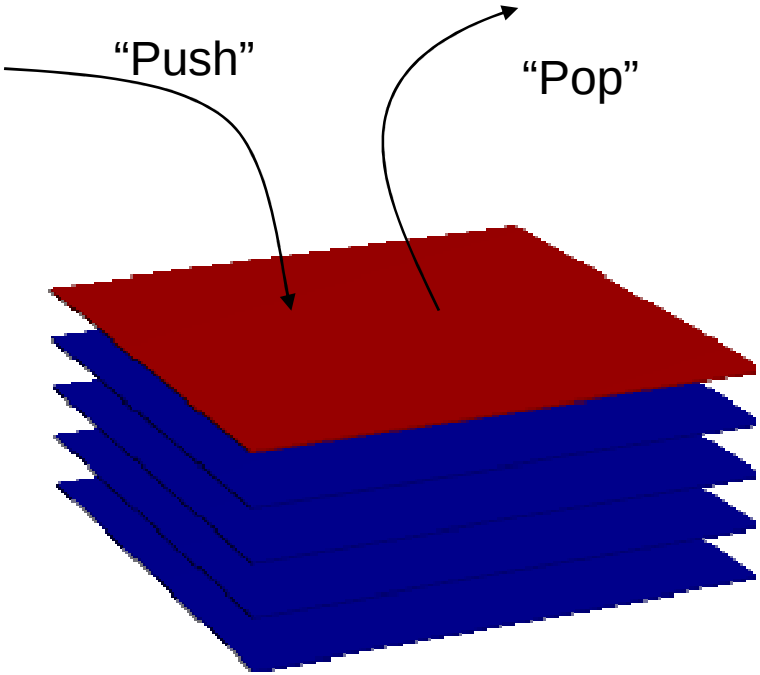
Pila

- Colección ordenada de elementos accesible por un único punto llamado cima o top.
- Los elementos en una pila tienen un orden LIFO (Last-In First-Out, último en entrar primero en salir)



Operaciones básicas

- Push: Añadir un elemento al final de la pila
- Pop: Leer y eliminar un elemento del final de la pila



Ejemplo de funcionamiento

Operación	Pila	Elemento extraído
Push(a)	a	
Push(b)	ab	
Pop()	a	b
Push(c)	ac	
Push(d)	acd	
Pop()	ac	d

Implementación estática (1/2)

```
class Pila {
   private:
           static const int MAX = 3;
           int tope;
           int valores[MAX];
   public:
   Pila() {
      tope = -1;
   }

   int empty() {
       if (tope==-1) return 1;
       else return 0;
   }

   int full() {
       if (tope== MAX-1) return 1;
       else return 0;
   }
```

```
   void push(int v) {
       if (! full()) {
         valores[++tope]=v;}
       else {
           cout << "No es posible
   agregar un elemento" << endl;
           }
   }
   int pop() {
      if (! empty()) {
           return (valores[tope--]);}
     else {
           cout << "No es posible
   extraer un elemento" << endl;
               return (0);
          }
    }
};
```

Implementación estática (2/2)

```
int main(int argc, char *argv[])
{
   Pila p;
   p.push(5);
   p.push(10);
   p.push(15);
   int x = p.pop();
   if (x != 0)
      cout << x << endl;

    system("PAUSE");
    return EXIT_SUCCESS;
}
```

Implementación dinámica: nodo

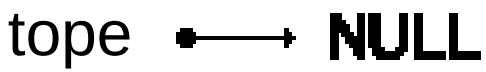
Nodo es una clase que permitirá crear estructuras con dos atributos*:* un contenido y un apuntador al siguiente Nodo


```
class nodo {
   private:
    int valor;
    nodo *siguiente;
   public:
    nodo(int v, nodo *sig)
    {
       valor = v;
       siguiente = sig;
    }
    int getValor() {return (valor);}

    nodo *getSiguiente() {
              return(siguiente); };
};
```

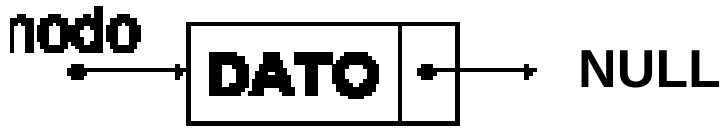
Implementación dinámica: crear una pila

- Hacer que *tope* apunte a NULL

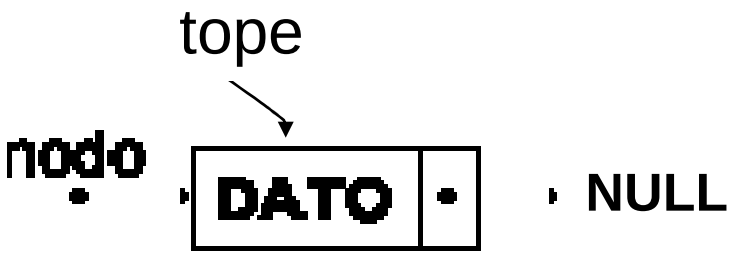


“Push”: Insertar en una pila vacía

1. Crear un nodo y hacer que su siguiente apunte a NULL

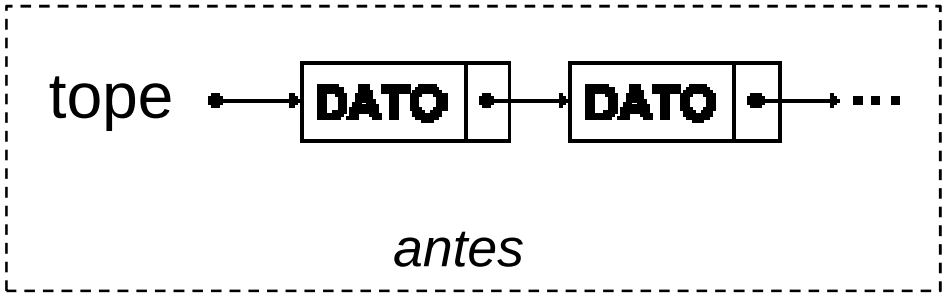


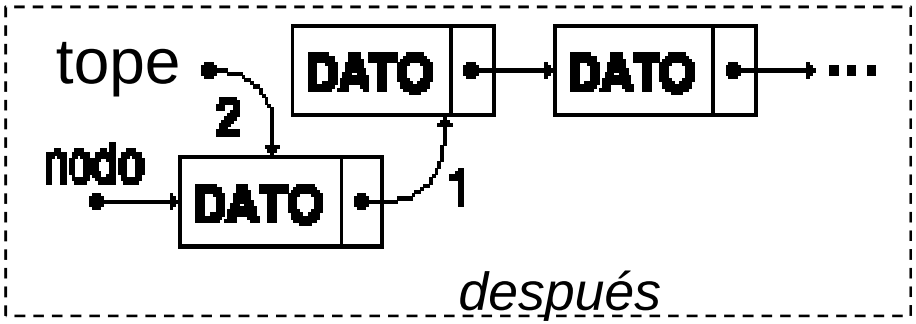
2. Hacer que tope apunte a nodo.



“Push”: Inserta en una pila no vacía

1. Crear un nodo y hacer que su siguiente apunte al tope
2. Hacer que tope apunte al nodo.

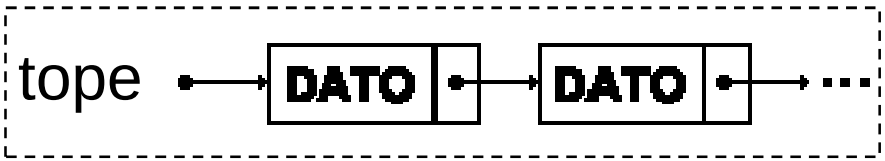




Pop: leer y eliminar un elemento

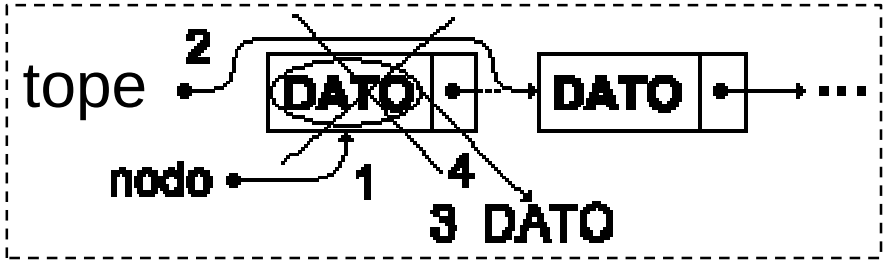
Suponiendo que se parte de una pila con uno o más nodos, considere un apuntador auxiliar *nodo*:

1. Hacer que *nodo* apunte al primer elemento de la pila, es decir a *tope*
2. Asignar a *tope* la dirección del segundo nodo de la pila; es decir, el de su nodo siguiente
3. Guardar el contenido de *nodo* para devolverlo como retorno,
4. Liberar la memoria asignada a *nodo*, que es el que se desea eliminar



antes

después



Si la pila sólo tiene un nodo, el proceso sigue siendo válido, ya que el siguiente del tope es NULL, y después de eliminar el último nodo la pila quedará vacía y el valor de tope será NULL.

Implementación dinámica (1/2)

```
class pila {
   private:
    nodo *tope;

   public:
    pila();
    ~pila();

    void Push(int v);
    int Pop();

};
pila::pila() {
     tope = NULL;
}
pila::~pila(){
   nodo *aux;
   while(tope!=NULL) {
      aux = tope;
      tope = (*tope).getSiguiente();
      delete aux;
   }
}
```

```
void pila::Push(int v){
   /* Crear un nodo nuevo */
   nodo *n = new nodo(v, tope);
   /* el comienzo de la pila es el nuevo
nodo */
   tope = n;
}
int pila::Pop(){
   int v; /*variable aux para retorno*/
   /*Si la pila está vacía,regresar 0*/
   if(tope == NULL) return 0;
   nodo *n = tope; /* nodo apunta al
         primer elemento de la pila */
   tope = (*n).getSiguiente(); /* el nuevo
         tope es el siguiente del tope
         actual.Con esto nos lo saltamos*/
    v = (*n).getValor();  /* Guardamos el
         valor de retorno que es el
         contenido del antiguo tope */
   delete n; /* Borrar el nodo */
   return v;
}
```

Implementación dinámica (2/2)

```
int main()
{
   pila Pila;

   Pila.Push(20);
   cout << "Push(20)" << endl;
   Pila.Push(10);
   cout << "Push(10)" << endl;

   cout << "Pop() = " << Pila.Pop() << endl;
   Pila.Push(40);
   cout << "Push(40)" << endl;
   Pila.Push(30);
   cout << "Push(30)" << endl;
   cout << "Pop() = " << Pila.Pop() << endl;
   cout << "Pop() = " << Pila.Pop() << endl;
   Pila.Push(90);
   cout << "Push(90)" << endl;
   cout << "Pop() = " << Pila.Pop() << endl;
   cout << "Pop() = " << Pila.Pop() << endl;

   cin.get();
   return 0;
}
```