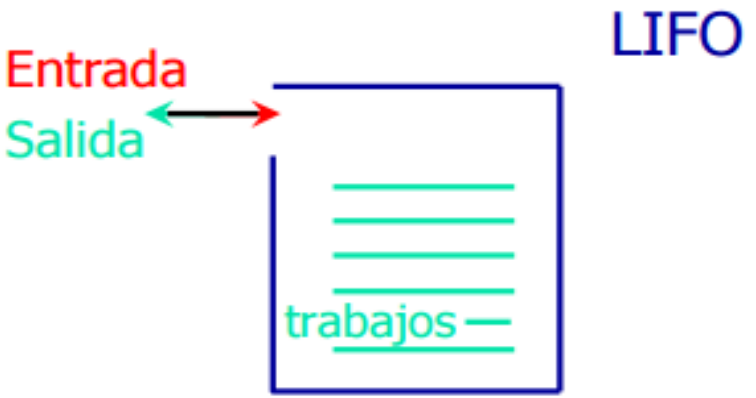
S5: Pilas (Stacks)

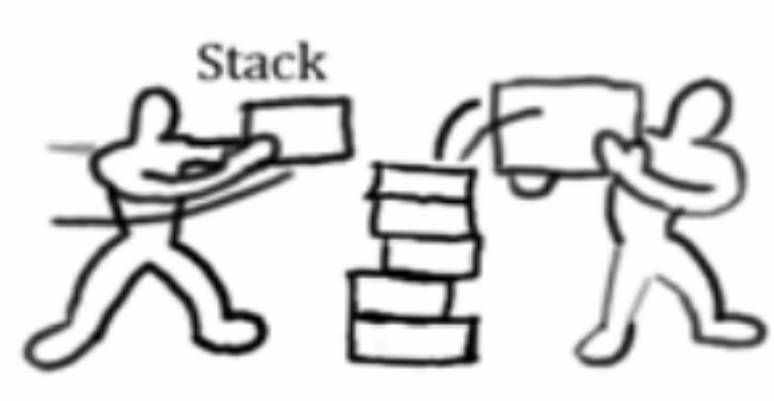
Estructuras de Datos

Docente: Pamela Landero Sepúlveda
pamelalandero@Gmail.com

Pilas - Stacks

- Una de las estructuras de datos más primitivas y más antiguas en la ciencia de los computadores.
- A pesar de ser tan simple, es esencial en los compiladores, SS.OO., etc.
- Se pueden representar como una lista lineal que crece y decrece por el mismo extremo.
  - El último elemento que se incorpora a la estructura, es el único que se puede consultar y eliminar.





Pilas - Stack

- Es un tipo lineal de datos, secuencia de elementos de un tipo, una estructura tipo **LIFO** (**L**ast **I**n **F**irst **O**ut) último en entrar primero en salir.
- Son un subconjunto de las listas, en donde las eliminaciones e inserciones se realizan en un solo extremo.

Pilas - Aplicaciones

- Histórico de páginas visitadas en un browser de web.
- Secuencia de "*undo*" en un editor de textos.
- Cadena de llamadas a métodos en JVM o medioambiente *runtime* en C++
- *Parsers* en Compiladores (reconocedores sintácticos).
- SSOO.
- Convertir notación infija a posfija o prefija.
- Implementación de recursividad.

Ejemplo de Aplicación

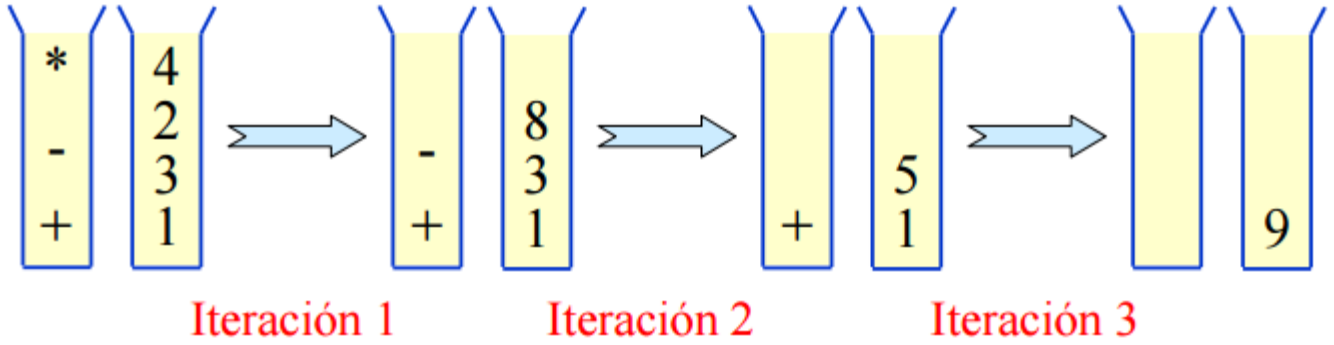
- Suponga que se quiere evaluar una expresión aritmética, mediante un proceso que se basa en guardar dicha expresión en dos pilas:
  - una de operadores (+, -, * y /) y
  - otra de operandos (números naturales).

Pilas – Ejemplo de Aplicación

- El método consiste en seguir el siguiente proceso de forma reiterada:

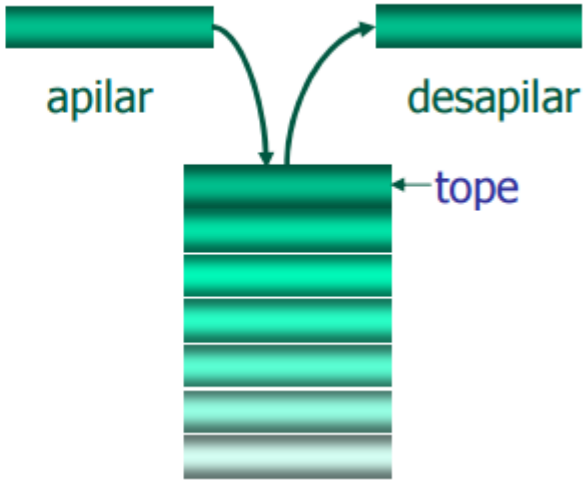
1. Si la pila de operadores está vacía, la cima de la de operandos contiene el resultado final.
2. En caso contrario, tomar los dos operandos de la cima de la pila y operarlos según el operador de la cima de la otra pila.

   El resultado colocarlo en la cima de la pila de operandos.

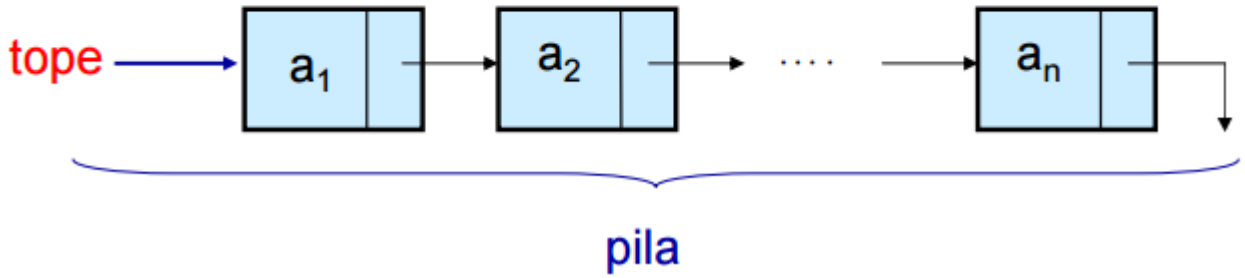


Pilas – Operaciones

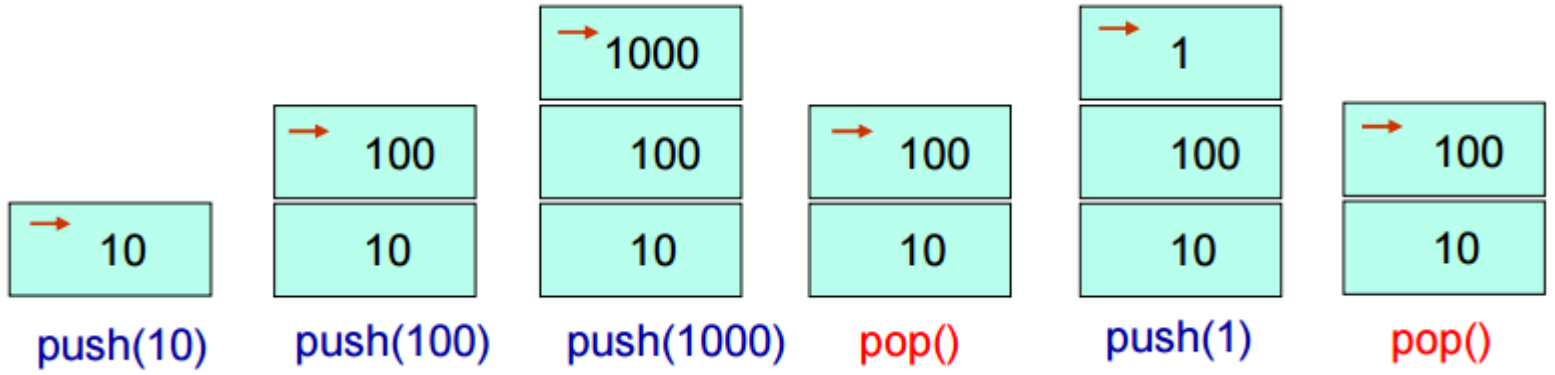
- Operaciones:
  - CrearPilaVacía(P)
  - PilaVacía(P) → B
  - Apilar(x, P)
  - Desapilar(P)
  - Tope(P) → E

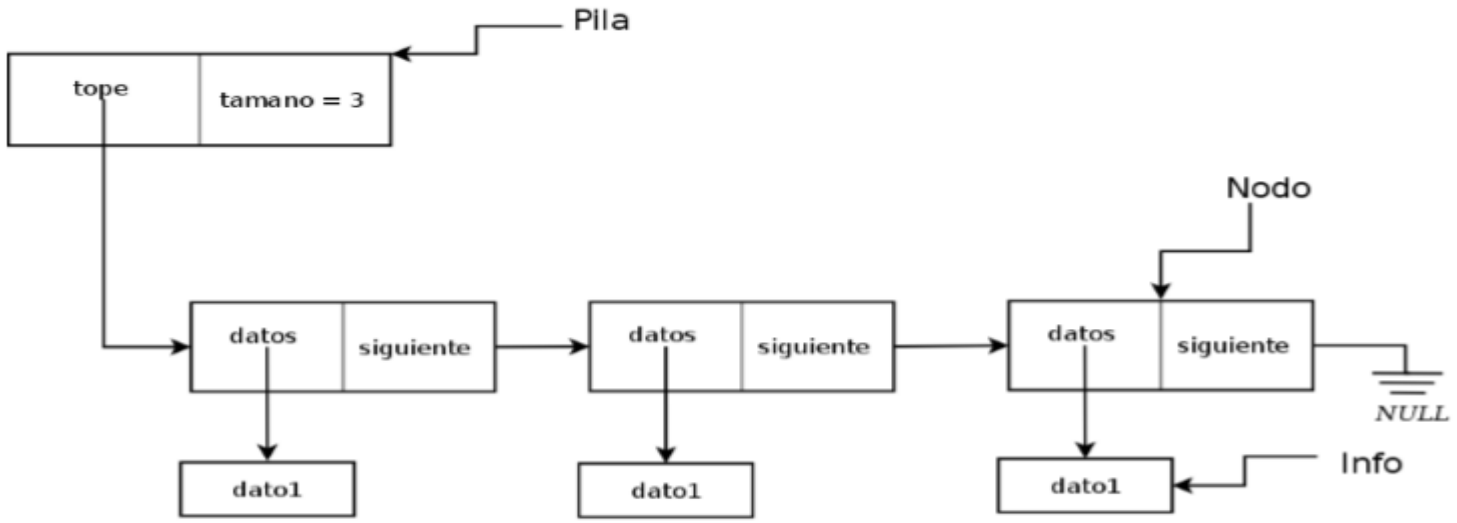


- Las pilas se implementan como una lista enlazada simple.
- El puntero de la lista (el tope) apunta al primer nodo de la pila.



Pilas – Implementación






```
typedef struct info{
        int valor;
}Info;

typedef struct nodo{
        Info *elem;
        struct nodo *sgte;
}Nodo;

typedef struct pila{
        Nodo *tope;
        int tam;
}Pila;
```

Pilas - Implementación

```c
int main(int argc, char *argv[]){
    Pila *P;
    P = crearPila();
    apilar(P,15); //push
    apilar(P,4);
    apilar(P,5);
    apilar(P,10);
    recorrerPila(P);
    printf("\n Valor desapilado: %d", desapilar(P)->valor);
    recorrerPila(P);
    printf("\n Tamaño de la pila: %d", P->tam);
    destruirPila(P);
}
```

```c
//Crea la información del nodo
Info *crearInfo(int pval){
    Info *newInfo;
    if (newInfo= (Info *) malloc(sizeof(Info))){
        newInfo->valor = pval;
    }else{
        printf("ERROR: MEMORIA INFO NO ASIGNADA");
    }
    return newInfo;
}
```

```c
//Crear Pila
Pila *crearPila(){
    Pila *p;
    if (p = (Pila *) malloc(sizeof(Pila)))
    {
        p->tope=NULL;
        p->tam=0;
    }else{
        printf("ERROR: MEMORIA PILA NO ASIGNADA");
    }
    return p;
}
```

```c
//crear nodo e insertar su valor
void apilar(Pila *p, int pval){
    //Crear Info
      Info *pinfo;
      pinfo = crearInfo(pval);
    //crear Nodo
    Nodo  *newNodo;
    if (newNodo = (Nodo *) malloc(sizeof(Nodo))) {
        newNodo->elem = pinfo;
        newNodo->sgte = p->tope;
        p->tope = newNodo;
        p->tam++;
    }else{
        printf("ERROR: MEMORIA NODO NO ASIGNADA");
        free(newNodo);
    }
}
```

Pilas – Implementación

```
//Desapilar
Info *desapilar(Pila *p){  //pop
    Nodo *aux;
    Info *inf;
    if (esVacia(p))
        printf("La pila está vacía no se puede desapilar\n");
    else
    {
      aux = p->tope; //aux apunta al primer elemento de la pila
      inf = aux->elem;
      p->tope = aux->sgte; //El inicio de p se asigna al nodo sgte (segundo nodo)
      p->tam--;
      free(aux);
    }
    return inf;
}
```

```
//Función que retorna el nodo del tope
Nodo *nodoTope(Pila *p){
    return p->tope;
}
```

```
//Función que reivisa si la Pila está vacía
bool esVacia(Pila *p){
    if(p->tope == NULL) return true;
    else return false;
}
```

TAREA:

Implemente las funciones recorrer pila, destruir pila y el main

¿ cuál es la eficiencia de las operaciones de una pila?

Ejercicios Propuestos:

1. Evaluar, indicando en cada paso el estado de la pila:
   Pila p = (Pila *)malloc(sizeof(Pila));

   a) Pop (Pop (Push (3, Pop (Push (4, Push (5, Push (6, p)))))))

   b) EstaVacia(Pop(Pop(Push(8,p))))

   c) InvPila(Push (3, Pop (Push (4, Push (5, Push (6, p)))))))

2. Escribir una función que permita contar todos los elementos pares que están en una pila dada.

3. Mediante el uso de pilas verifique si los paréntesis de una función están balanceados, es decir, que a cada paréntesis abierto le corresponde un paréntesis cerrado. Considere que la función llega como parámetro, y es un arreglo dinámico de largo n.

4. Escribir una función Reemplazar que tenga como argumentos una pila con tipo de elemento int y dos valores int: nuevo y viejo de forma que si el segundo valor aparece en algún lugar de la pila, sea reemplazado por el segundo.