Clase #21 de 27
Stack Frames, Estructuras Enlazadas & Nodos

Septiembre 28, Jueves

Agenda para esta clase

- Stack Frames
- Heap
  - Reserva dinámica de memoria desde el stack
  - Reserva dinámica de memoria desde el heap
  - Operador new
  - Operador delete
  - Diferencias entre Heap y Stack
- Estructuras Enlazadas & Nodos
  - Nodo
  - Recorrido de lista enlazada

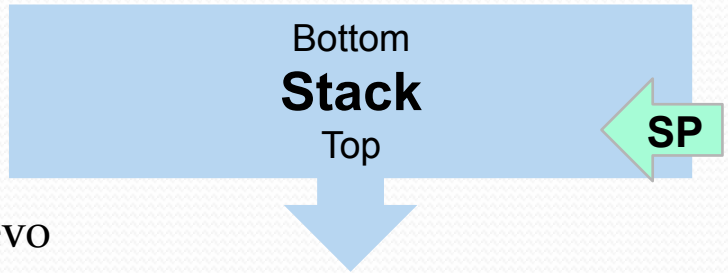
Stack Frames

Invocaciones a funciones

Stack Frame (Activation Record)

Cuadro de Pila (Registro de Activación)

- Los Stack Frames (activation records) son los elementos de la **Call Stack** (**Pila de Invocaciones**)
- Cada stack frame se corresponde a la **invocación** de una función que todavía no finalizó
- En el top está la función que se está ejecutando
- El contenido es dependiente de la plataforma, pero en general, un stack frame mantiene:
  - Variables locales o automáticas
  - Argumentos
  - Dirección de retorno para continuar la ejecución
- Cada invocación a función agrega, i.e. **Push**, un nuevo **Stack Frame** (**Activation Record**)
- Cada retorno de función saca, i.e. **Pop**, un stack frame, dejando el valor de retorno disponible
- El **Stack Pointer** (**SP**) apunta a la cima, **Top**, de la pila.
- ¿Qué es **el stack overflow**?
- ¿Es posible retornar un puntero a una variable local?.



Ejercicio – Call Stack:

Basado en hoja 6 de Memory Management in C

- Dado el call stack
  1. main
  2. foo
  3. bar
- ¿Cuántos stack frames hay?
- ¿Cuál función llamó a cuál?
- Dibuje un digrafo que represente el call stack y las relación "invocó-a"
- ¿Cuántas funciones hay en ejecución?
- ¿Cuál es la invocación activa?
- Diagrame y codifique un programa que se corresponda a este call stack.

Heap

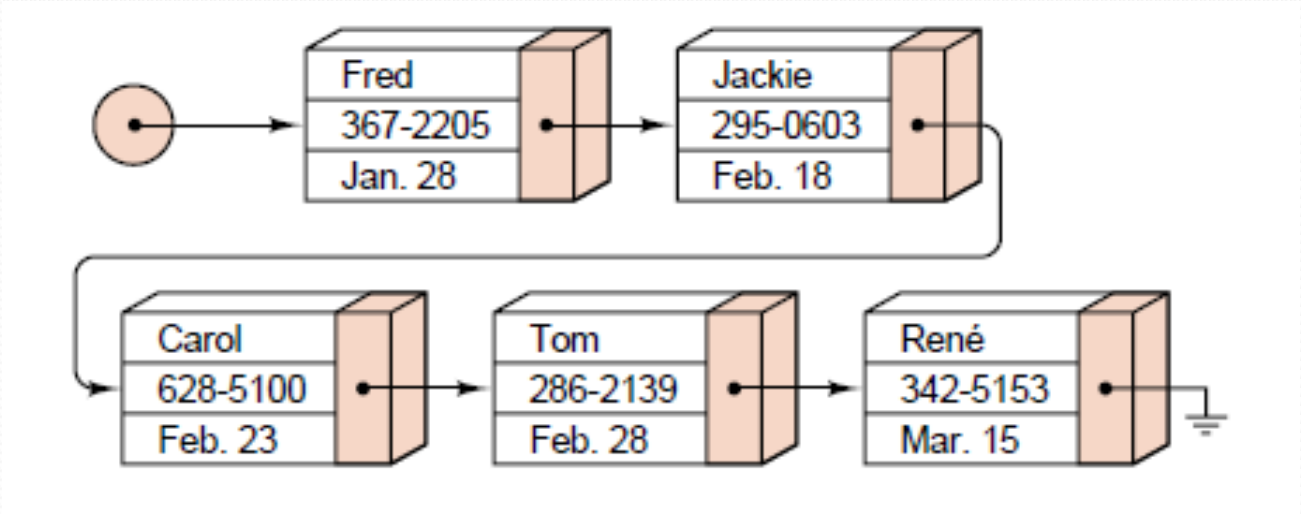
Reserva dinámica de memoria con los operadores new & delete

Reserva, Analogía con Restarurant

- Llamado para hacer una reserva
  - ¿Qué se pide? ¿Nombre? ¿Cantidad?
  - ¿Qué se registra?
  - ¿Qué se retorna?
  - new
- Llamado para cancelar reserva
  - delete
- Diferencias con Stack
  - Duración
  - Nombre
  - Liberación
  - Garbage Collector

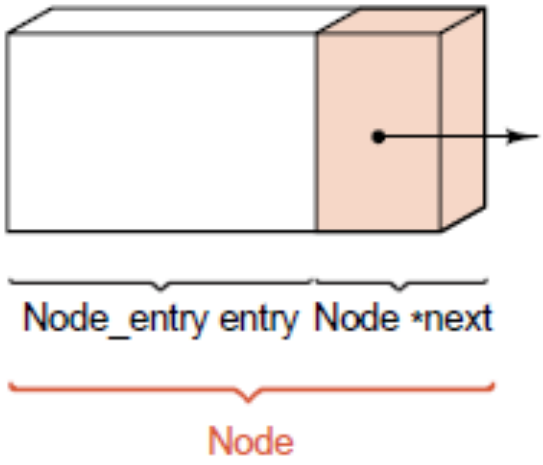
Estructuras Enlazadas & Nodos

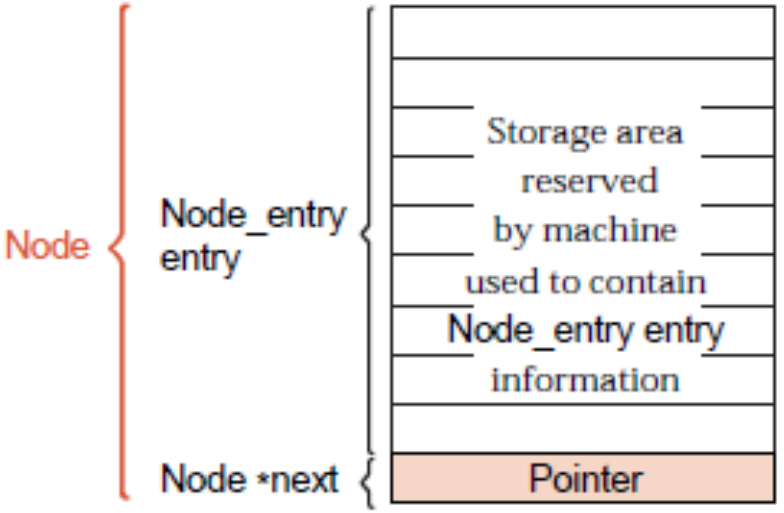
Introducción a Estructura Enlazada



- Repaso: Estructura Estática versus Dináminca
  - Stack, Queue, Array
- Estructura Enlazada
  - Stack, Queue, Linked List
- Problemas que resuelven
- Ventajas y Desventajas

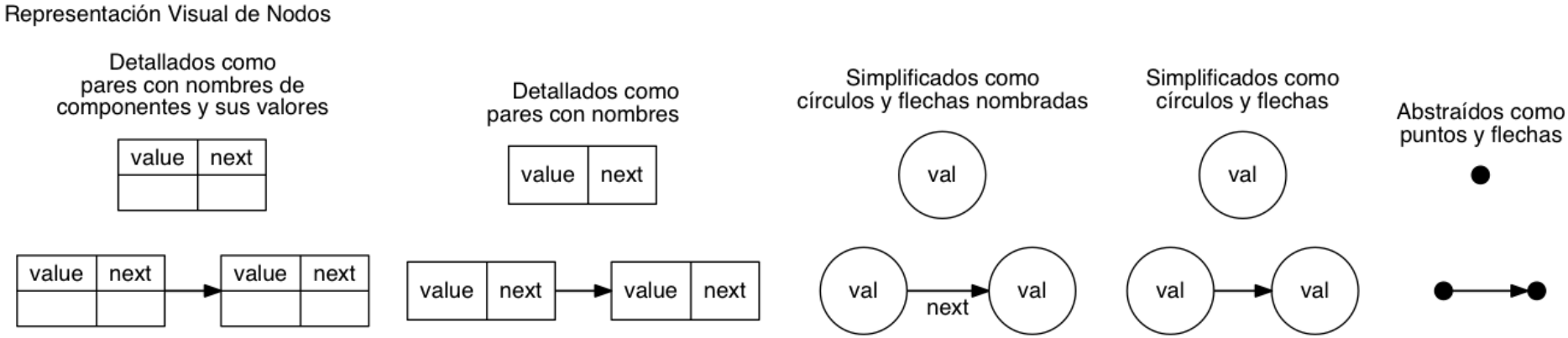
¿Qué es un Nodo? *node*=(*data*,*link*)





- Otros nombres para *data*
  - car
  - datos
  - entry, entrada
  - payload, carga
  - info, información
  - val, **value**, valor
- Otros nombres para *link*
  - cdr
  - enlace
  - **next**, siguiente, próximo
  - nxt, sgte

Representaciones de Nodos

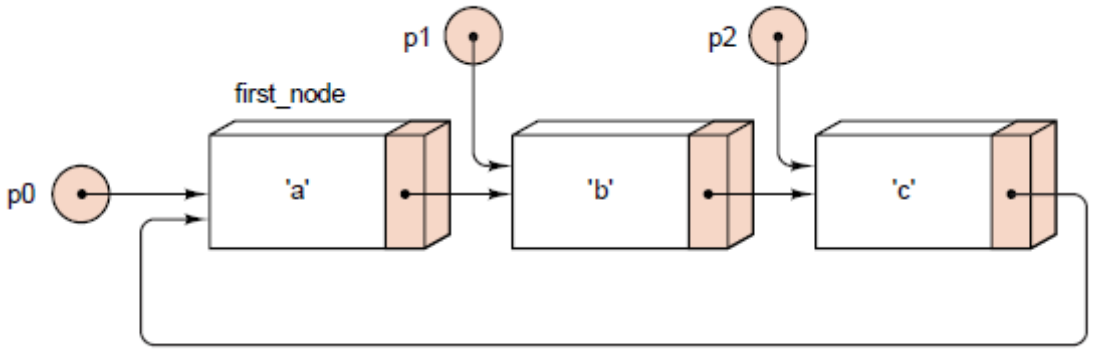



```
// Node of int
struct Node{
   int  value;
   Node *next;
};
```

```
// Node of Type
struct Node{
   Type value;
   Node *next;
};
```

```
// Template Node
struct Node<T>{
   T        value;
   Node<T> *next;
};
```

Ejemplo: Tres Nodos Enlazados

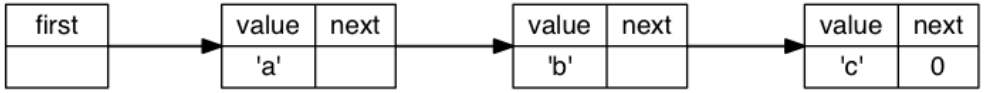



```
Node first_node;
first_node.value = 'a';

Node *p0 = &first_node;
Node *p1 = new Node;

(*p1).value = 'b';
p1->value = 'b';

p0->next = p1;

Node *p2 = new Node;
p2->value = 'c';
p2->next = p0;

p1->next = p2;

// Recorrido
Node *p=p0; // actual
do{
  cout << p->value;
  p=p->next;
}while( p != p0);
```

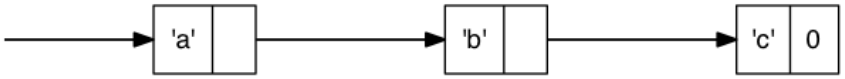
Recorrer la lista enlazada

Representación Visual de secuencia de caracteres (a,b,c) como lista enlazada

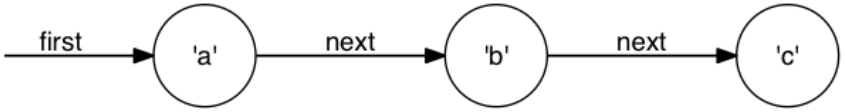
Detallados como pares



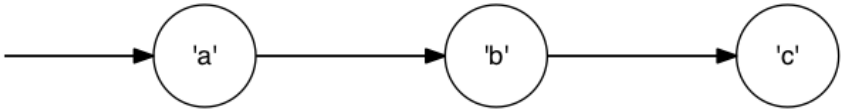
Detallados como pares sin nombres



Simplificados como círculos y flechas nombradas



Simplificados como círculos y flechas



Abstraídos como puntos y flechas

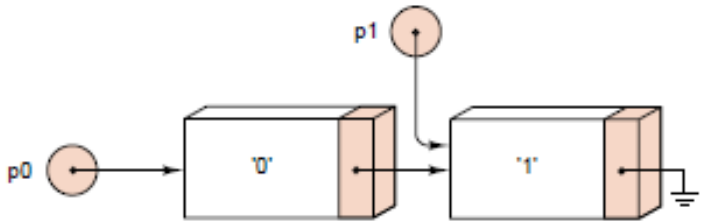
```
Node *first = …;
Node *p;

for(p=first; p!=nullptr; p=p->next)
  cout << p->value;
```

Ejercicio – Dibuje el diagrama para el siguiente código C++

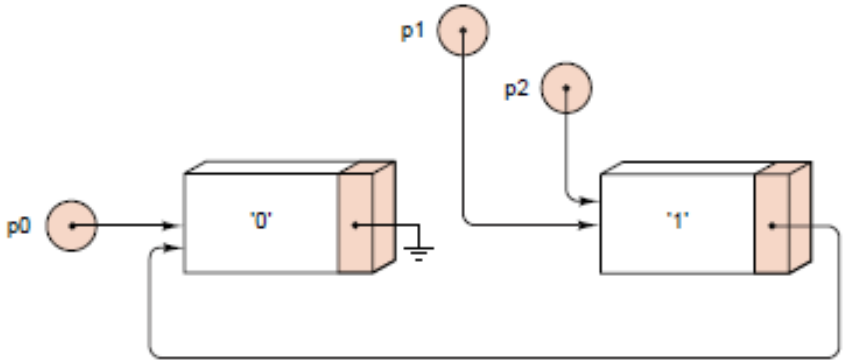
```
Node *p0 = new Node;
po->value ='0';

Node *p1 = p0->next = new Node;
p1->value ='1';

Node *p2 = p1->next = new Node;
p2->value ='2';
p2->next = p1;
```

Ejercicios – Escriba las sentencias

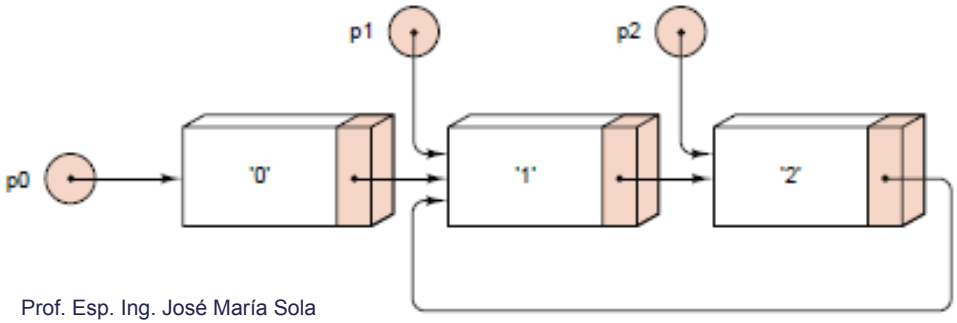
(a)



(b)



(c)



Términos de la clase #21

Definir cada término con la bibliografía

- Secciones de Memoria
  - Parte estática
  - Datos estáticos ó Pila Estática
  - Código ó Texto
  - Parte dinámica
  - Stack (Pila)
- Call Stack
  - Call Stack
  - Variables automáticas
  - Argumentos
  - IP
  - Stack Frame (Activation Record)
  - Stack pointer
  - Diagrama de call stack y las relación "invocó-a"
  - Invocación activa
  - Push
  - Pop
- Heap (Montículo)
  - Puntero
  - Declaración de puntero
  - Operador *
  - Operador &
  - Operador new
  - Operador delete

Tareas para la próxima clase

1. Implementar versiones enlazadas de Stack y Queue, con declaraciones y definiciones en archivos separados
2. Leer http://josemariasola.wordpress.com/aed/papers#StringPackerAndBlockStream

¿Consultas?

Fin de la clase