Estructuras autoreferenciadas
Listas simplemente enlazadas

Sofía Beatriz Pérez
Daniel Agustín Rosso

sperez@iua.edu.ar
drosso@iua.edu.ar

Centro Regional Univesitario Córdoba
Instituto Univeristario Areonáutico

Clase número 6 - Ciclo lectivo 2023

Agenda

Disclaimer

Los siguientes slides tienen el objetivo de dar soporte al dictado de la asignatura. De ninguna manera pueden sustituir los apuntes tomados en clases y/o la asistencia a las mismas.
Es importante mencionar que todos este material se encuentra en un proceso de mejora continua.
Si encuentra bugs, errores de ortografía o redacción, por favor repórtelo a sperez@iua.edu.ar y/o drosso@iua.edu.ar. También puede abrir issues en el repositorio de este link: ▸ infoI_IUA_GitLab

Estructuras autoreferenciadas I

Una estructura autoreferenciada contiene un puntero que apunta a una estructura del mismo tipo de estructura:

```
1  struct node
2  {
3      int data;
4      struct node *p_next;
5  };
```

Esta estructura tiene dos miembros, el miembro entero "data" es quien almacena la información útil del nodo y el miembro puntero p_next que apunta a una estructura del tipo node y es por esto que se las conoce como estructuras autoreferenciadas.

Listas simplemente enlazadas: introducción I

Una lista enlazada es una colección lineal de estructuras autoreferenciadas conectadas entre si mediante enlaces de punteros.

Consideraciones:

- Se tiene acceso a la lista mediante un puntero al primer nodo de la lista
- Para marcar el fin de una lista, el apuntador de enlace del último nodo, debe apuntar a NULL.
- Un nodo puede almacenar datos de cualquier tipo, esto incluye a otras estructuras.

Listas simplemente enlazadas: introducción II

- La implementación de listas dinámicas es particularmente útil cuando no se conoce previamente la cantidad de datos a ser almacenados en la estructura
- El número de nodos que conforma una lista puede variar en tiempo de ejecución, es decir, pueden crearse y destruirse nodos a demanda
- Normalmente, los nodos de la listas enlazadas no están almacenados en memoria de forma contigua [1]. Sin embargo, los nodos de una lista pueden ser recorridos de forma contigua.

Listas simplemente enlazadas: introducción III

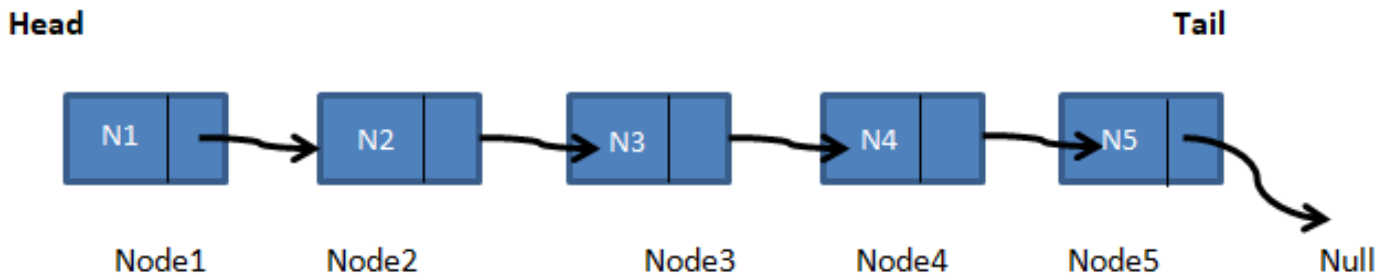
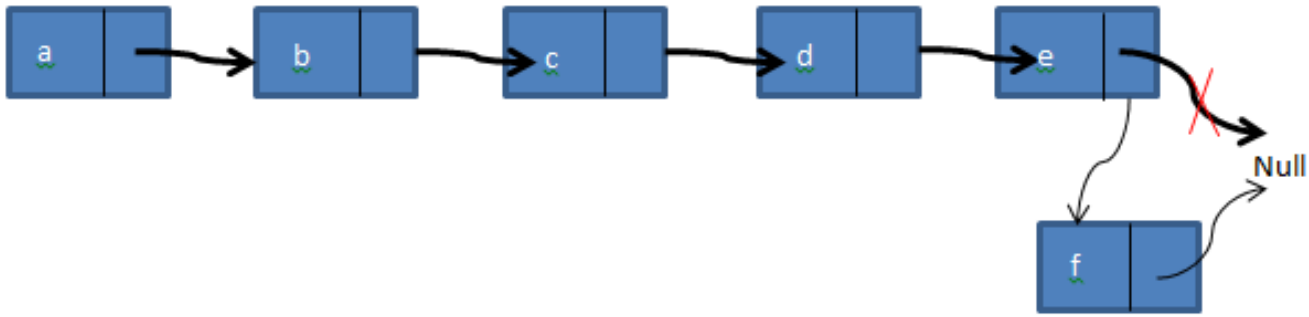


Figure: Representación gráfica de una lista simplemente enlazada.

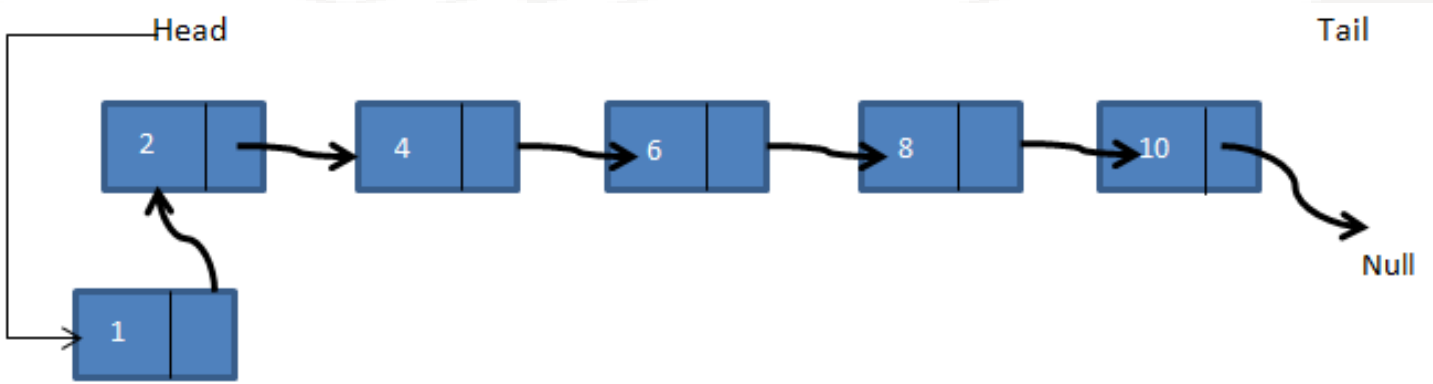
[1] Recordar que los arreglos si. Es por esto que se puede utilizar aritmética de punteros para recorrerlos.

Listas simplemente enlazadas: operaciones I

- Inserción de un nodo al final

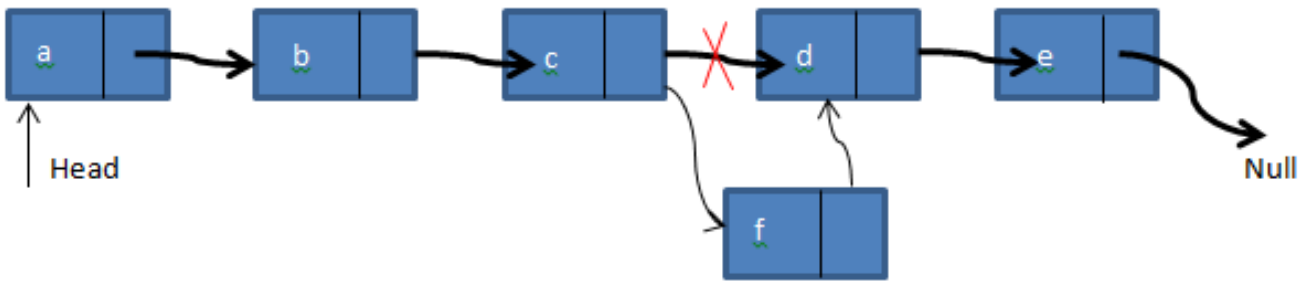


- Inserción de un nodo al comienzo



Listas simplemente enlazadas: operaciones II

- Inserción de un nodo después de un cierto nodo



- Eliminación de un nodo
- Impresión de todos los nodos
- Contar cantidad de nodos

Listas simplemente enlazadas: creación

```
1  struct node
2  {
3      int data;
4      struct Node *p_next;
5  };
6
7
8  int main()
9  {
10 /* puntero al comienzo de la lista */
11 struct node* head = NULL;
12
13 ...
```

Listas simplemente enlazadas: insertando un nodo al final I

```
1  /*Asignacion del dato al nuevo nodo*/
2  printf("Ingrese un dato\n");
3  scanf("%d", &dato);
4
5  new_node = malloc(sizeof(struct Node));
6  new_node = (struct Node * ) new_node
7
8  if (new_node == NULL)
9  {
10    printf("No hay memoria disponible");
11    exit(0);
12 }
13
14
15
```

Listas simplemente enlazadas: insertando un nodo al final II

```
16  new_node -> data = dato;
17
18  /*Como va al final de la lista
19  este nodo apunta a NULL*/
20  new_node -> next = NULL;
21
22  /*Si la lista esta vacia, el nodo ingresado
23  es el primero de la lista */
24  if (head == NULL)
25  {
26     head = new_node;
27  }
28
29
```

Listas simplemente enlazadas: insertando un nodo al final III

```
30  else
31  {
32
33    /*Buscamos cual es el ultimo*/
34    temp = head;
35    while (temp -> next != NULL)
36      temp = temp -> next;
37
38    /*Hacemos que el que era ultimo
39    apunte al nuevo nodo*/
40    temp -> next = new_node;
41  }
```

Listas simplemente enlazadas: imprimiendo la lista I

```
1  temp = head;
2  while (temp != NULL)
3  {
4     printf("%d\n", temp -> data);
5     temp = temp -> next;
6  }
```

Listas simplemente enlazadas: borrando un nodo la lista I

```
1  if (head == NULL)
2  {
3     printf("Lista vacia\n");
4  }
5  else
6  {
7     printf("Ingrese el dato a borrar\n");
8     scanf("%d", &dato);
9     /*el dato a borrar es el primer nodo*/
10    if (dato == head -> data)
11    {
12       temp = head;
13       head = head -> next;
14       free(temp);
15    }
```

Listas simplemente enlazadas: borrando un nodo la lista II

```
16      else
17      {
18
19          prev = head;
20          current = head -> next;
21          while (current != NULL && current -> data != dato)
22          {
23              prev = current;
24              current = current -> next;
25          }
26
27
28
29
30
```

Listas simplemente enlazadas: borrando un nodo la lista III

```
31          /*El dato existe*/
32          if (current != NULL)
33          {
34              temp = current;
35              prev -> next = current -> next;
36              free(temp);
37          }
38
39      }
40  }
```

▸ Ver ejemplo completo en gitlab

Listas simplemente enlazadas: implementación con funciones I

- ¿Cómo implementamos una función para agregar un nodo al comienzo de la lista?
- Se necesita modificar el contenido, es decir la dirección de memoria almacenada en head
- ¿Cómo permitimos que una función modifique el contenido de una variable puntero?

Listas simplemente enlazadas: implementación con funciones II

- Propuestas de prototipos para insertar un nodo al comienzo de la lista
  - void push(struct Node , int);
  - void push(struct Node *, int);
  - void push(struct Node **, int);
  - void push(struct Node ***, int);

Listas simplemente enlazadas: implementación con funciones III

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  struct Node {
5    int data;
6    struct Node * next;
7  };
8
9
10 void push      (struct Node **, int);
11 void append    (struct Node **, int);
12 void print_list(struct Node*);
13
14
```

Listas simplemente enlazadas: implementación con funciones IV

```
15  void append(struct Node** head, int node_data)
16  {
17    struct Node* new_node = NULL;
18    new_node = (struct Node * ) malloc(sizeof(struct Node)
19    if(new_node==NULL)
20    {
21      printf("No hay memoria disponible");
22      exit(0);
23    }
24    struct Node *temp = *head;
25    new_node->data = node_data;
26
27    new_node->next = NULL;
28
```

Listas simplemente enlazadas: implementación con funciones V

```
29      if (*head == NULL)
30      {
31          *head = new_node;
32          return;
33      }
34
35      while (temp->next != NULL)
36          temp = temp->next;
37
38      temp->next = new_node;
39  }
40
41
42
```

Listas simplemente enlazadas: implementación con funciones VI

```
43  void push(struct Node** head, int node_data)
44  {
45    struct Node* new_node = NULL;
46    new_node = (struct Node * ) malloc(sizeof(struct Node));
47    if (new_node == NULL)
48    {
49      printf("No hay memoria disponible");
50      exit(0);
51    }
52    new_node->data = node_data;
53    new_node->next = (*head);
54    (*head) = new_node;
55  }
56
```

Listas simplemente enlazadas: implementación con funciones VII

```
57
58  void print_list (struct Node *head)
59  {
60     struct Node* temp = NULL;
61     temp=head;
62     while (temp != NULL)
63     {
64        printf("%d\n", temp -> data);
65        temp = temp -> next;
66     }
67  }
68
69
70
```

Listas simplemente enlazadas: implementación con funciones VIII

```
71  void menu(void)
72  {
73      printf("1.- Agregar un nodo al final\n");
74      printf("2.- Agregar un nodo al comienzo\n");
75      printf("3.- Impresion de la lista\n");
76      printf("4.- Salir\n");
77  }
78
79
80
81
82
83
84
```

Listas simplemente enlazadas: implementación con funciones IX

```
85  int main(void)
86  {
87    int dato = dato = 0,op=0;
88    /*Puntero al comienzo de la lista*/
89    struct Node * head = NULL;
90    do {
91      menu();
92      scanf("%d", &op);
93      switch (op) {
94      case 1:
95        printf("Ingrese un dato\n");
96        scanf("%d", &dato);
97        append(&head,dato);
98      break;
```

Listas simplemente enlazadas: implementación con funciones X

```
99          case 2:
100            printf("Ingrese un dato\n");
101            scanf("%d", &dato);
102            push(&head,dato);
103            break;
104         case 3:
105            print_list(head);
106            break;
107            }
108       }while (op != 4);
109      return (0);
110   }
```

▸ Ver ejemplo completo en gitlab

¡Muchas gracias!

Consultas:

sperez@iua.edu.ar
drosso@iua.edu.ar