



Universidad Autónoma de San Luis Potosí

Facultad de Ingeniería

Área en Ciencias de la Computación

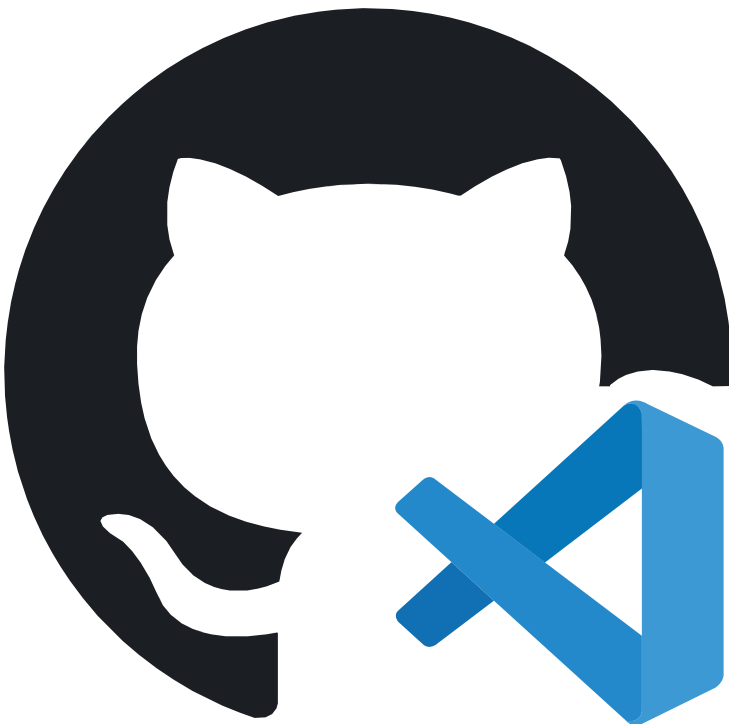


Estructuras de Datos II

Semestre: Agosto – Diciembre 2022

Alumno: Esparza Castañeda Hugo

Profesor: M. en C. Froylán Eloy Hernández Castro



Unidad 1	APUNTADORES	1
	1.1. Introducción a apuntadores	1
	1.2. Parámetros por valor/referencia	1
	1.3. Apuntadores a apuntadores	2
	1.4. Apuntadores y arreglos	2
TAREA	Tarea #0: Resumen del video “Pointers”	3
	1.5 Aritmética de apuntadores	
	1.5.1. Operaciones aritméticas sobre apuntadores	
	1.5.2. Ejemplos de aritmética de apuntadores	
	1.6. Como utilizar Codespaces y Git/GitHub	
TAREA	Tarea #1: Aritmética de apuntadores	
	1.7. Gestión de memoria dinámica	
	1.8. Liberación	
	1.9. Estados de la memoria dinámica	
	1.10. Problemas en el manejo de la memoria dinámica	
	1.11. Arreglos dinámicos	
	1.12. Matriz dinámica	
	1.13. Apuntadores por “referencia”	
TAREA	Tarea #2: Arreglos dinámicos	
	1.14. realloc	
	1.15. Pila dinámica	
	1.16. Estructuras dinámicas	
	1.16.1. Versión estática	
	1.16.2. Versión dinámica	
	1.17. Apuntadores genéricos	
TRABAJO	Trabajo en clase: “Arreglo dinámico”	
TAREA	Tarea #3: Arreglos redimensionables	
TRABAJO	Trabajo en clase: “Farmacia con medicamentos genéricos y de patente”	
EXAMEN	Examen Primer Parcial	
Unidad 2	LISTAS ENLAZADAS	
	2.1. Listas simples	
	2.1.1. Definir la estructura del nodo	



2.1.1. Definir la estructura del nodo	1
2.1.2. Función para crear un nodo	1
2.1.3. Operaciones en listas enlazadas	2
2.1.4. Creando un menú de opciones	2
2.1.5. Función “int selecciona_opción();”	2
2.1.6. Función “void insertar_inicio(nodo_t **, int);”	



1. APUNTADORES

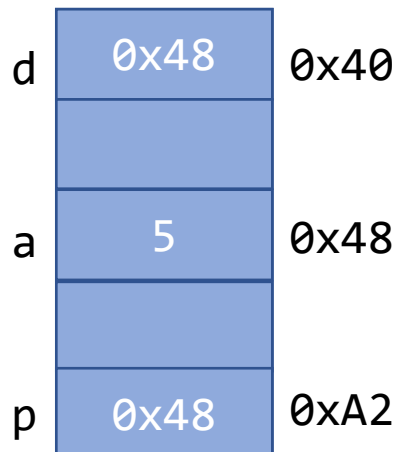


1.1 Introducción a apuntadores

Objetivo: Conocer el concepto de apuntadores en el paso de parámetros de funciones y gestión de memoria (Dinámica), así como ser capaz de utilizarlos.

Apuntador: un apuntador es una variable que contiene una dirección de memoria pero su contenido es otra dirección de memoria.

```
1 int main(){
2     int a;
3     a = 5;
4     int *p;
5     p = &a;
6     int *d;
7     d = p;
8     return 0;
9 }
```



```
printf("%d", a); //output: 5
printf("%d", *p); //output: 5
printf("%p", p); //output: 0x48
printf("%d", *d); //output: 5
printf("%p", d); //output: 0x48
```

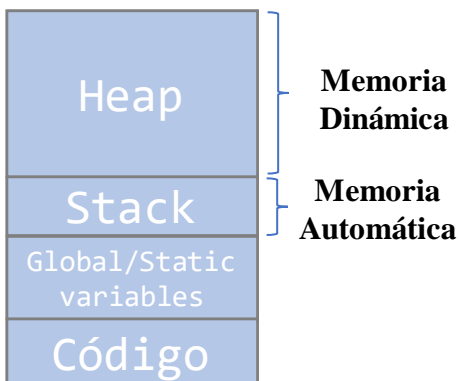
$a = 10; \Leftrightarrow *p = 10$

$\text{scanf}(\text{"\%d"}, p) \Leftrightarrow \text{scanf}(\text{"\%d"}, \&a)$

1.2 Parámetros por valor/referencia

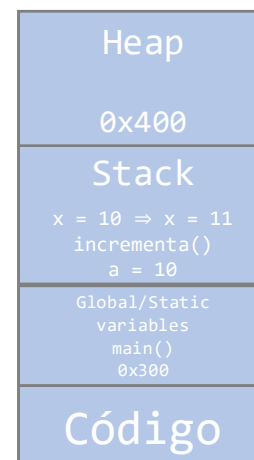
Jueves 18 Agosto 2022

Memoria

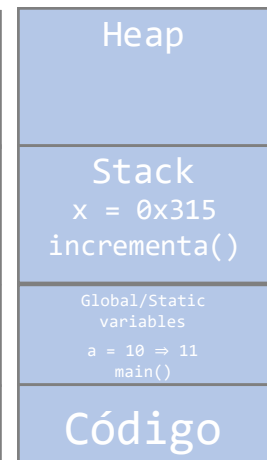


```
1 #include <stdio.h>
2 void incrementa(int *);
3 int main(){
4     int a;
5     a = 10;
6     incrementa(&a);
7     printf("%d", a);
8     return 0;
9 }
10
11 void incrementa(int *x){
12     *x = *x + 1;
13     return;
14 }
```

Por valor



Por referencia



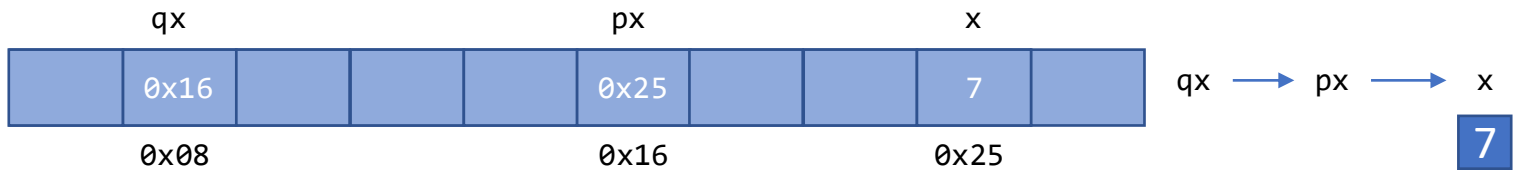


```
int *r;  ⇒  int *r = NULL;
```



1.3 Apuntadores a apuntadores

Viernes 19 Agosto 2022



```

1  #include <stdio.h>
2  int main(){
3      int x = 7; //entero
4      int *px = &x; //apuntador
5      int **qx = &px; //apuntador a apunt
6      printf("%p\n", px); //output: 0x25
7      printf("%d\n", *px); //output: 7
8      printf("%p\n", &x); //output: 0x25
9      printf("%p\n", &px); //output: 0x16
10     printf("%p\n", &qx); //output: 0x08
11     printf("%p\n", qx); //output: 0x16
12     printf("%p\n", *qx); //output: 0x25
13     return 0;
14 }
```

* = indirección = Obtener el valor almacenado en una dirección.

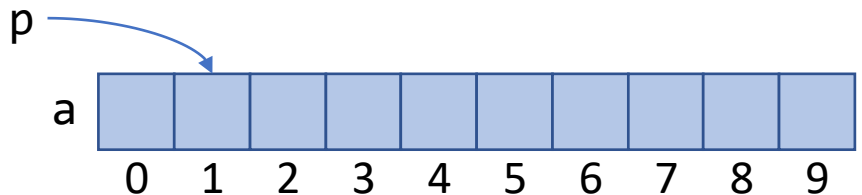
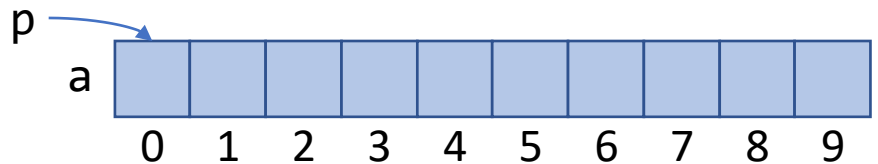
& = dirección = Obtener la dirección de una variable.

Los apuntadores a apuntadores se usan para matrices dinámicas.

1.4 Apuntadores y arreglos

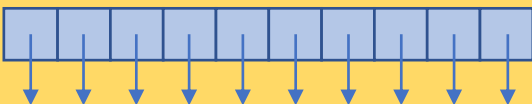
```
int a[10];
int *p = a; ⇒ int *p = &a[0];
```

```
p = p + 1;
```



Arreglo de apuntadores

```
int *b[10];
```



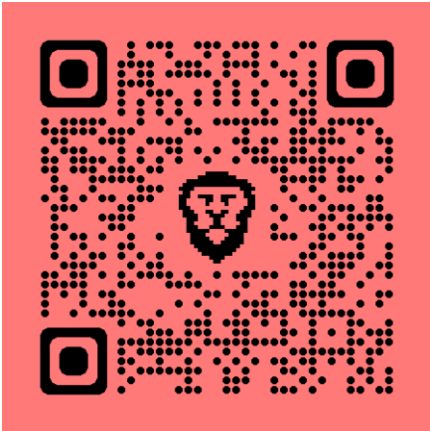
Arreglo de apuntadores apuntando a NULL

```
for(int i = 0; i < 10; i++){
    b[i] = NULL;
}
```

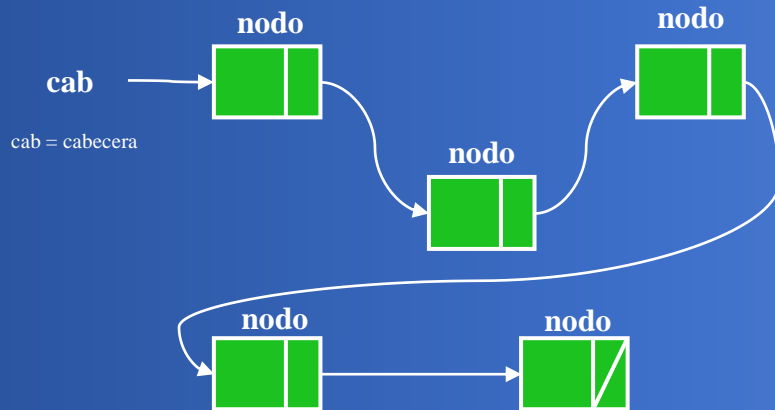
⇔ `int *b[10] = {NULL};`



Tarea #0: Resumen del video “Pointers”



2. LISTAS ENLAZADAS



2.1 Listas simples

Objetivo: Ser capaz de diseñar diversos tipos de listas enlazadas y programar las principales operaciones para su manipulación.

En una lista podemos guardar cualquier cosa, para este ejemplo de lista simple, haremos una lista de números enteros, cada casilla es un nodo, y para cada nodo hacemos una estructura con los datos que almacenara cada nodo.

2.1.1 Definir la estructura del nodo

Struct nodo

info

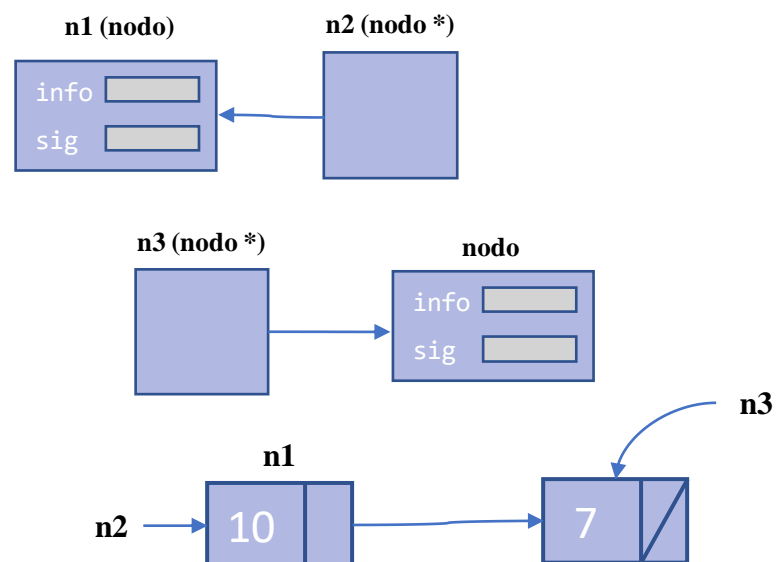
sig

```
1 struct nodo{
2     int info;
3     struct nodo *sig;
4 };
5 typedef struct nodo nodo_t;
```

```
1 int main(){
2     nodo_t n1;
3     nodo_t *n2, *n3;
4     n1.info = 5;
5     n1.sig = NULL;
6     n2 = &n1;
7     n2->info = 10;
8     printf("%d", n1.info);
9     n3 = (nodo_t *)malloc(sizeof(nodo_t));
10    assert(n3 != NULL);
11    n3->info = 7;
12    n3->sig = NULL;
13    n2->sig = n3;
14    return 0;
15 }
```

Output

10



2.1.2 Función para crear un nodo

```

1  nodo_t *crea_nodo(){
2      nodo_t *nodo = NULL;
3      nodo = (nodo_t*)malloc(sizeof(nodo_t));
4      if(nodo == NULL){
5          printf("Error: no hay memoria suficiente");
6          exit(EXIT_FAILURE);
7      }
8      nodo->info = 0;
9      nodo->sig = NULL;
10     return nodo;
11 }

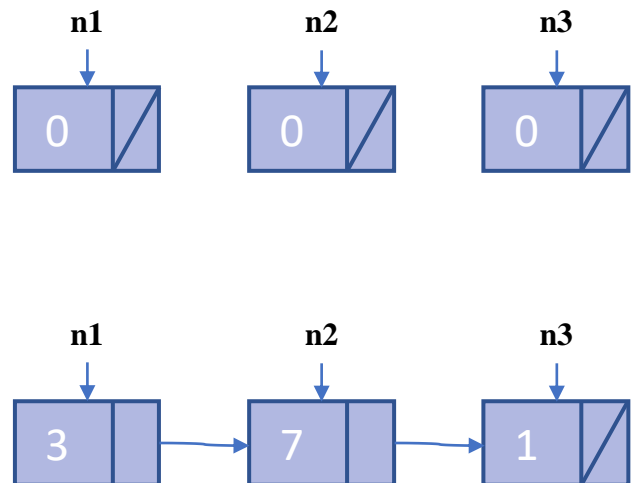
```

Ejemplo Burdo

```

1  int main(){
2      nodo_t *n1 = crea_nodo();
3      nodo_t *n2 = crea_nodo();
4      nodo_t *n3 = crea_nodo();
5      n1->info = 3;
6      n2->info = 7;
7      n3->info = 1;
8      n1->sig = n2;
9      n2->sig = n3;
10     printf("%d ", n1->info);
11     printf("%d ", n1->sig->info);
12     printf("%d", n1->sig->sig->info);
13     free(n1);
14     free(n2);
15     free(n3);
16     return 0;
17 }

```

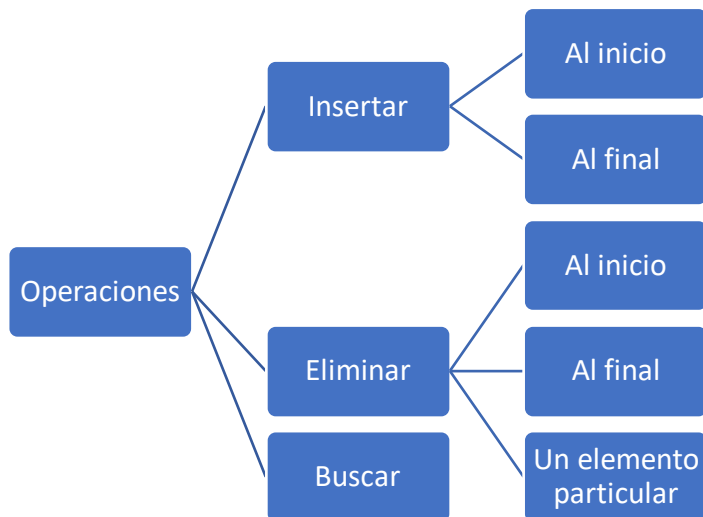


Output

3 7 1



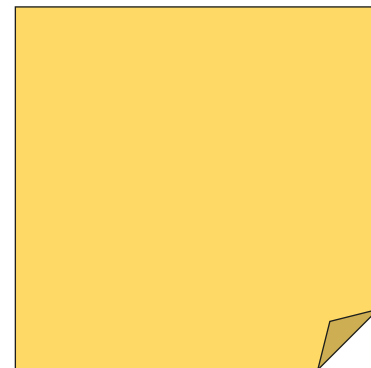
2.1.3 Operaciones en listas enlazadas



2.1.3 Creando un menú de opciones

```

1  int main(){
2      nodo_t *lista = NULL;
3      int opcion = 0, num = 0;
4      do{
5          opcion = selecciona_opcion();
6          switch(opcion){
7              case 1: printf("Valor del numero: ");
8                     scanf("%d", &num);
9                     insertar_inicio(&lista, num);
10                    break;
11             case 2: printf("Valor del numero: ");
12                    scanf("%d", &num);
13                    insertar_final(&lista, num);
14                    break;
15             case 3: eliminar_inicio(&lista);
16                    break;
17             case 4: eliminar_final(&lista);
18                    break;
19             case 5: imprimir_lista(lista);
20                    break;
21             case 6: printf("Valor del numero: ");
22                    scanf("%d", &num);
23                    elimina_nodo(&lista, num);
24                    break;
25             case 7: liberar_lista(&lista);
26                    break;
27             case 8: printf("Valor del numero: ");
28                    scanf("%d", &num);
29                    printf("%p\n", buscar_dato(lista, num));
30                    break;
31             default: puts("Opcion no valida");
32                     break;
33          }
34      }while(opcion != 0);
35      return 0;
36  }
  
```





2.1.5 Función "int selecciona_opción();"

```

1  int selecciona_opcion(){
2      puts("Selecciona una opcion");
3      puts("0. Salir");
4      puts("1. Insertar inicio");
5      puts("2. Insertar al final");
6      puts("3. Eliminar al inicio");
7      puts("4. Eliminar al final");
8      puts("5. Imprimir toda la lista");
9      puts("6. Eliminar nodo arbitrario");
10     puts("7. Eliminar/Liberar toda la lista");
11     puts("8. Buscar dato");
12     int opcion;
13     scanf("%d", &opcion);
14     return opcion;
15 }

```

puts funciona como printf, pero es para imprimir en pantalla puro texto, y ya imprime el salto de línea sin necesidad de ponerlo.

2.1.5 Función "void insertar_inicio(nodo_t **, int);"

```

1  void insertar_inicio(nodo_t **cab, int dato){
2      nodo_t *nodo = crea_nodo();
3      nodo->info = dato;
4      nodo->sig = *cab;
5      *cab = nodo;
6  }

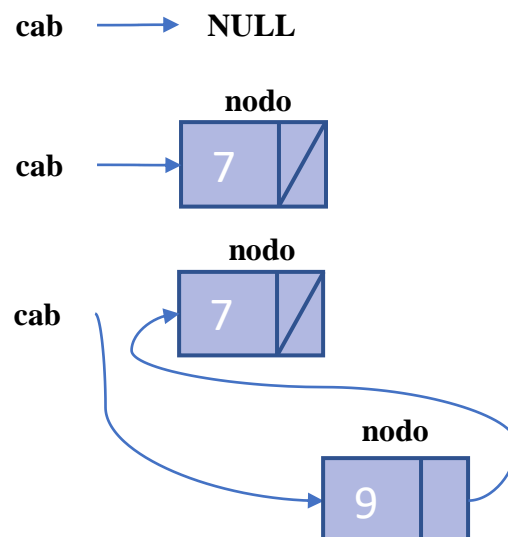
```

Ejemplo:

```

1  int main(){
2      nodo_t *lista = NULL;
3      insertar_inicio(&lista, 7);
4      insertar_inicio(&lista, 9);
5      return 0;
6  }

```



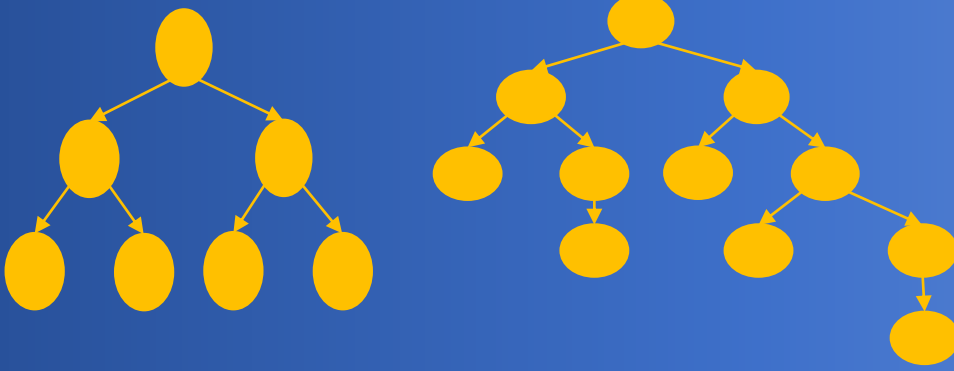




3. GRAFOS



4. ÁRBOLES BINARIOS

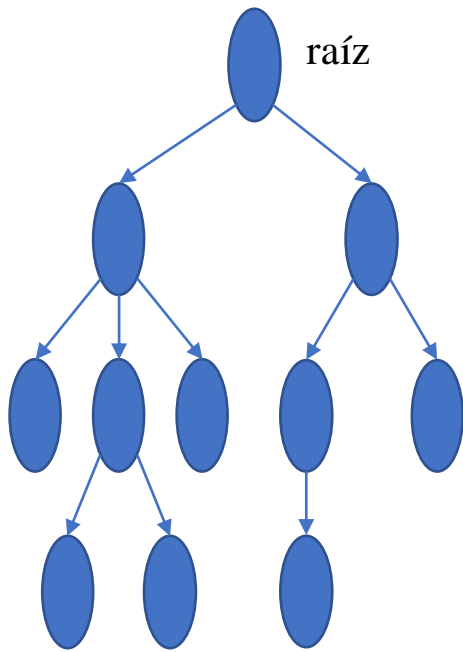


4.1

Conceptos básicos de árboles

Objetivo: Diseñar y programar las estructuras y las operaciones básicas para la manipulación de grafos.

Lunes 07 Noviembre 2022



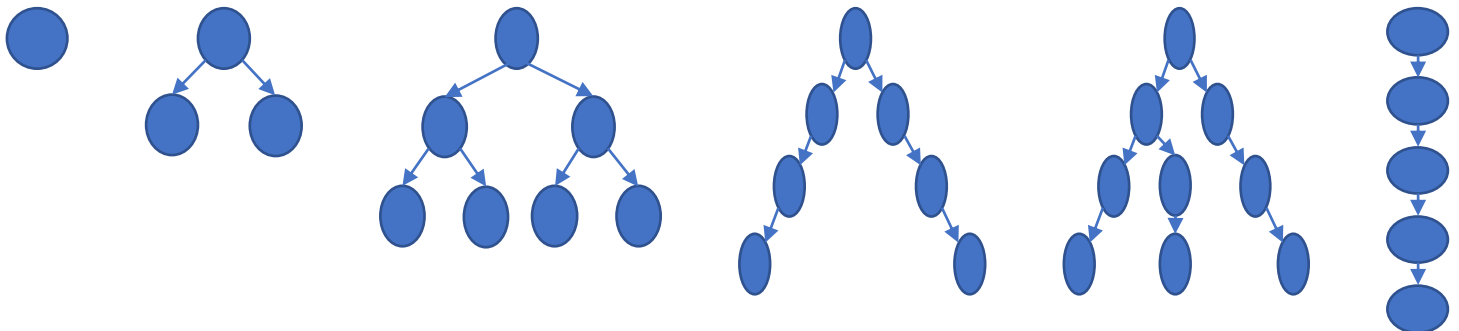
Terminología

Raíz	Nodo principal, cabecera.
Hijos	Nodos izquierdo y derecho de un nodo.
Padre	Nodo con hijos.
Hermanos	Nodos que tienen el mismo padre.
Antecesor y descendiente	Si podemos ir desde el nodo A hacia el nodo B, entonces A es un antecesor de B, y B es un descendiente de A.
Hoja	Nodos sin hijos.
Nodo interno	Nodos que no son hojas.
Camino	Secuencia de aristas desde un nodo antecesor hasta un nodo descendiente.

Definiciones y aplicaciones

Un **árbol binario** es un árbol en el cual cada nodo puede tener a lo máximo 2 hijos.

Ejemplos:





Aplicaciones de los árboles:

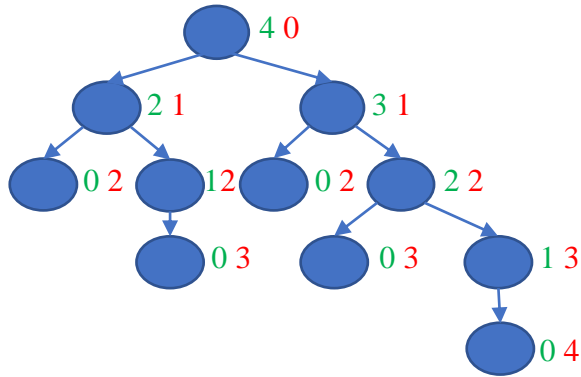
- Información con estructura jerárquica.
- Navegación web y redes.
- Bases de datos (árboles B y B+).
- Son la base para las estructuras eficientes como los conjuntos (sets).

Árbol binario completo

Todos los niveles excepto posiblemente el último están llenos y todos los nodos están lo más posible a la izquierda.

Altura de un nodo

Martes 08 Noviembre 2022



Altura de un nodo x

Es el número de aristas en el camino más largo desde x hasta una hoja. Las hojas tienen altura cero.

La altura de un árbol es la altura de la raíz (en este caso es 4).

La profundidad de un nodo x

Es la longitud del camino desde la raíz hasta el nodo x (cantidad de aristas). La profundidad de la raíz es cero.

Los nodos que tienen la misma profundidad están en un mismo nivel.

En un árbol binario perfecto, todos los niveles están completamente llenos.

El número máximo de nodos en el nivel i es 2^i . Un árbol perfecto de altura h tiene exactamente $2^{h+1} - 1$ nodos.

¿Cuál es la altura de un árbol binario perfecto que tiene N nodos?

$$2^{h+1} - 1 = N \Rightarrow 2^{h+1} = N + 1$$

$$\log a^b = b \log a$$

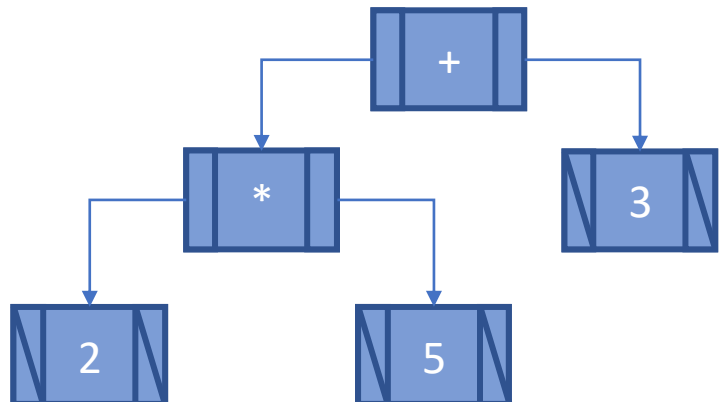
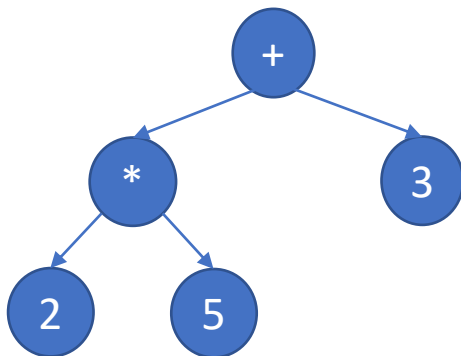
$$\log_x x = 1$$

$$\log_2(2^{h+1}) = \log_2(N + 1) \Rightarrow (h + 1) \log_2 2 = \log_2(N + 1) \Rightarrow h = \log_2(N + 1) - 1$$

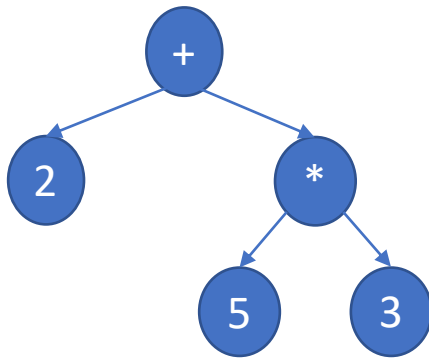
4.2

Árboles binarios de expresiones

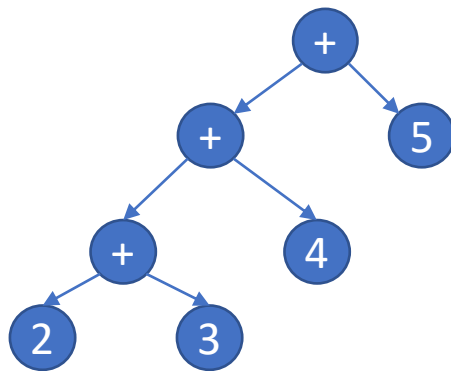
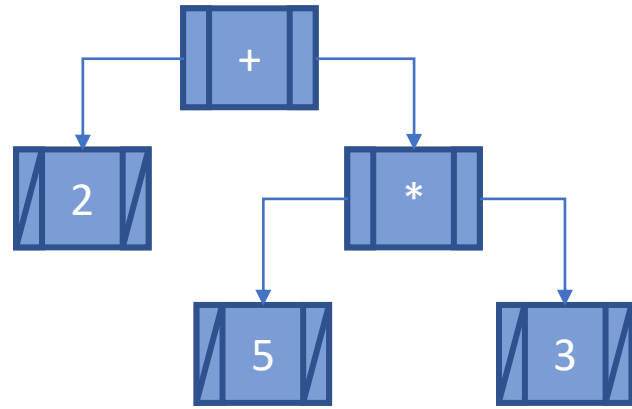
Jueves 10 Noviembre 2022



$$2 * 5 + 3 \Rightarrow 13$$

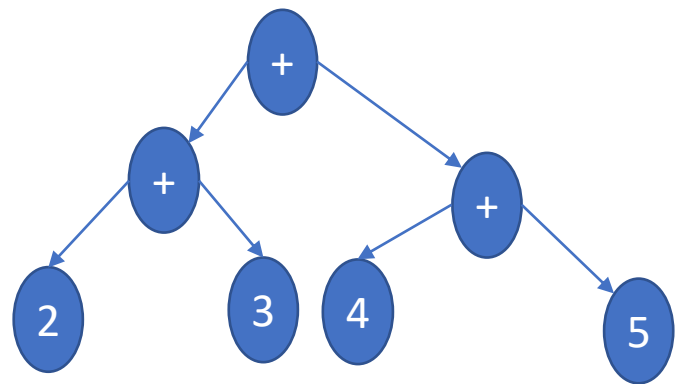


$$2 * (5 + 3) \Rightarrow 16$$



$$2 + 3 + 4 + 5 \Rightarrow 14$$

≠



$$(2 + 3) + (4 + 5) \Rightarrow 14$$

Aunque para nosotros sea la misma operación matemática, son dos árboles completamente diferentes por los paréntesis.

4.2.1 Definir la estructura del nodo

```

1 struct nodo{
2     int info;
3     struct nodo *izq;
4     struct nodo *der;
5 };
6 typedef struct nodo nodo_t;
```

5. ÁRBOLES MULTICAMINOS





