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

Índice General de Temas

FUNDAMENTOS DE UN LENGUAJE DE ALTO NIVEL

1

1.1 Introducción al lenguaje de programación C

Algunas aplicaciones importantes de C

C se utiliza para cuestiones de alto rendimiento, como por ejemplo en supercomputadoras para realizar cálculos muy rápidos, también para computadoras muy pequeñas como Arduino.

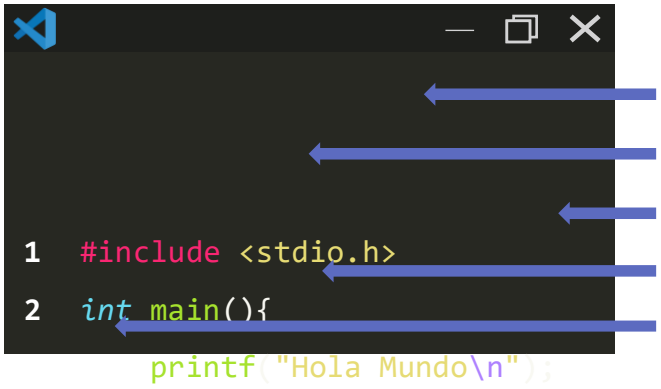
Cuestiones que tienen que ver con precisión y exactitud, el robot que mandaron a Marte esta programado en C, entonces es un lenguaje usado en la NASA.

También hay egresados de la carrera que trabajan utilizando el lenguaje C para hacer controladores para turbinas de aviones.

El kernel de Linux esta escrito en C.

Estructura básica de un programa en C

Comenzaremos mencionando las partes que componen un programa básico en C, para ello utilizaremos el algoritmo para explicarlo.



Incluimos la librería.

Declaramos la función principal.

Imprimimos texto en pantalla.

Verificamos que el programa termino satisfactoriamente.

Se cierra la función principal.

Al principio de nuestro código debemos agregar las librerías que vamos a necesitar, por ejemplo, para poder imprimir en pantalla, así como para poder leer datos de variables, necesitamos la librería stdio.h, para agregarla debemos escribir “#include <stdio.h>”, tal y como se muestra en la línea 1 del código.

A continuación debemos declarar la función principal en la que vamos a escribir el código que vamos a ejecutar, para hacerlo, primero declaramos el tipo de función que será, en este caso de tipo entero (int), luego le damos un nombre a la función, en este caso “main”, que significa “principal” en inglés, a continuación abrimos paréntesis y colocamos las variables que se ocuparan en la función, en este caso no se necesita ninguna, así que lo podemos dejar vacio, o bien, escribir “void” dentro de los paréntesis, de la siguiente manera:

```
int main(void){
```

Esto se explicara más detalladamente en la sección **1.8 Funciones y paso de parámetros.**

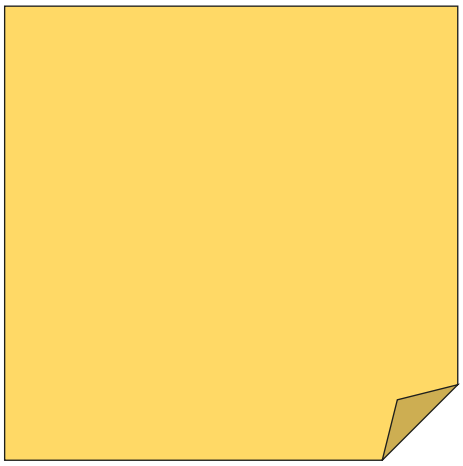
Al final abrimos corchetes, y dentro de los corchetes escribimos el código de la función.

En el renglón 3 escribimos el texto que queremos mostrar en pantalla, para hacerlo, escribimos `printf("Hola Mundo\n");` lo veremos más detalladamente en la sección **1.6 Funciones de entrada y salida.**

En el renglón 4 regresamos el valor 0 para indicar que el programa terminó correctamente.

Finalmente, en el renglón 5 cerramos el corchete, dando por terminado el código de la función, y este caso, del programa.

1.2 Instalación del entorno de programación en Windows



En este video puedes ver como hacer la instalación de codeblocks en Windows, así como también una breve introducción a como utilizar el programa para crear nuevos archivos de código, así como también compilarlos y ejecutarlos.

El programa se puede descargar del siguiente link.

Link de descarga

Es un archivo ejecutable, simplemente hay que ejecutarlo y darle siguiente en las opciones que aparecen hasta que quede instalado.

Para más información, ver el video de la izquierda.

1.3 Instalación del entorno de programación en Android

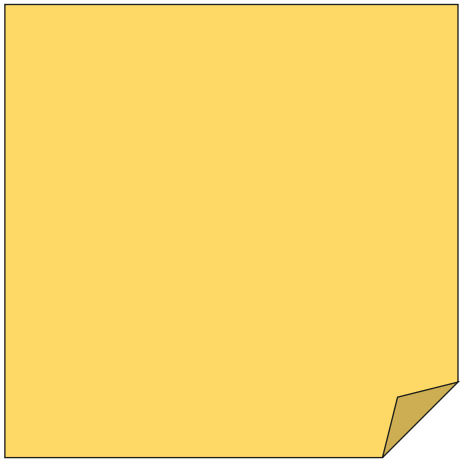
En este video puedes ver como hacer la instalación de Cxxdroid en un celular android, así como también una breve introducción a como utilizar el programa para crear nuevos archivos de código, así como también compilarlos y ejecutarlos.

El programa se puede descargar del siguiente link.

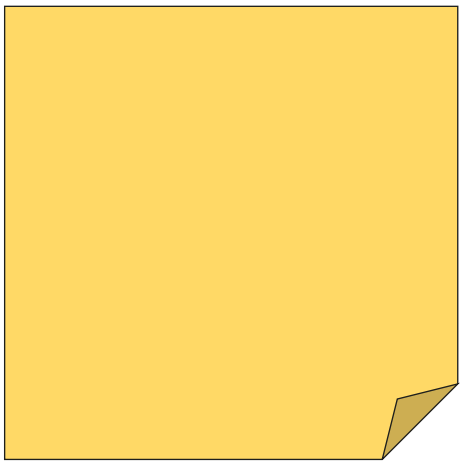
Link de descarga

Simplemente hay que instalarlo dese la play store en nuestro celular, abrir la aplicación, y comenzar a escribir código.

Para más información, ver el video de la derecha.



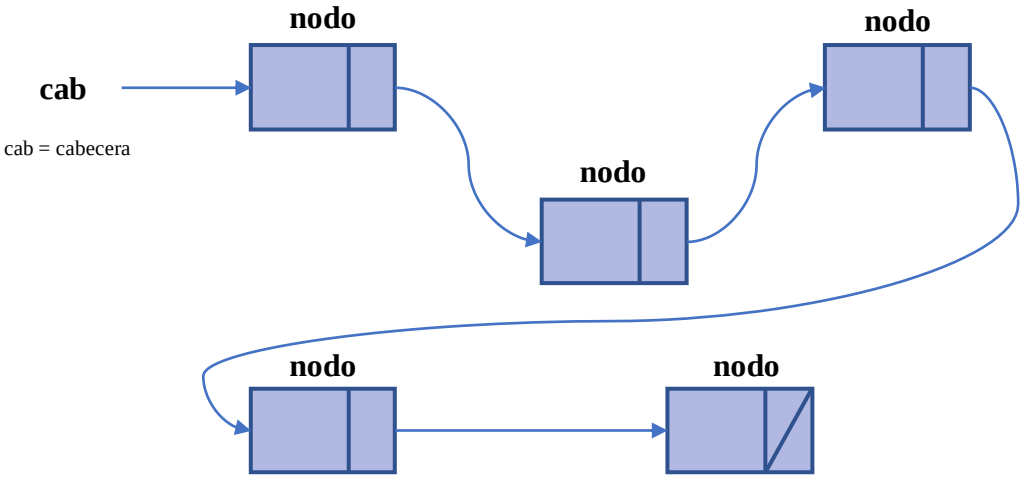
1.4 Variables y tipos de datos



LISTAS ENLAZADAS

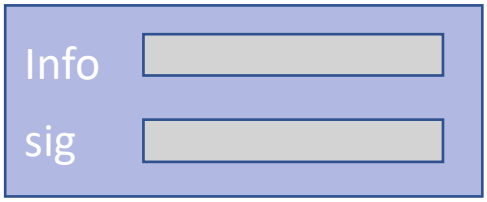
2

2.1 Listas simples



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.

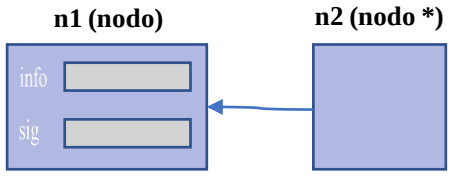
Struct nodo

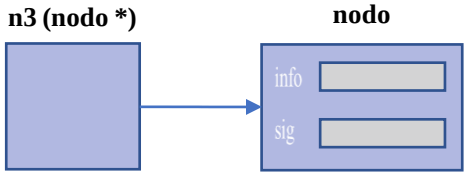


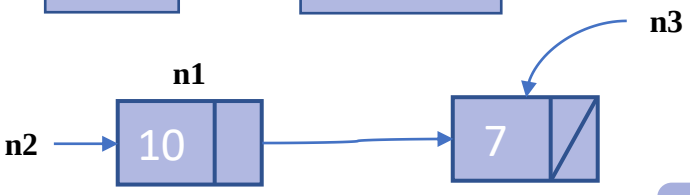
2.1.1 Definir la estructura del nodo

```c
struct nodo{
    int info;
    struct nodo *sig;
};
typedef struct nodo nodo_t;
```

```c
int main(){
    nodo_t n1;
    nodo_t *n2, *n3;
    n1.info = 5;
    n1.sig = NULL;
    n2 = &n1;
    n2->info = 10;
    printf("%d", n1.info);
    n3 = (nodo_t *)malloc(sizeof(nodo_t));
    assert(n3 != NULL);
    n3->info = 7;
    n3->sig = NULL;
    n2->sig = n3;
    return 0;
}
```







Output

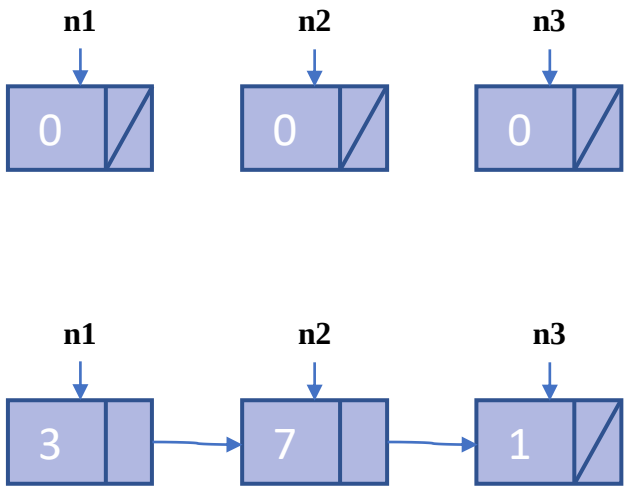
10

2.1.2 Función para crear un nodo

```c
nodo_t *crea_nodo(){
    nodo_t *nodo = NULL;
    nodo = (nodo_t*)malloc(sizeof(nodo_t));
    if(nodo == NULL){
        printf("Error: no hay memoria suficiente");
        exit(EXIT_FAILURE);
    }
    nodo->info = 0;
    nodo->sig = NULL;
    return nodo;
}
```

Ejemplo Burdo

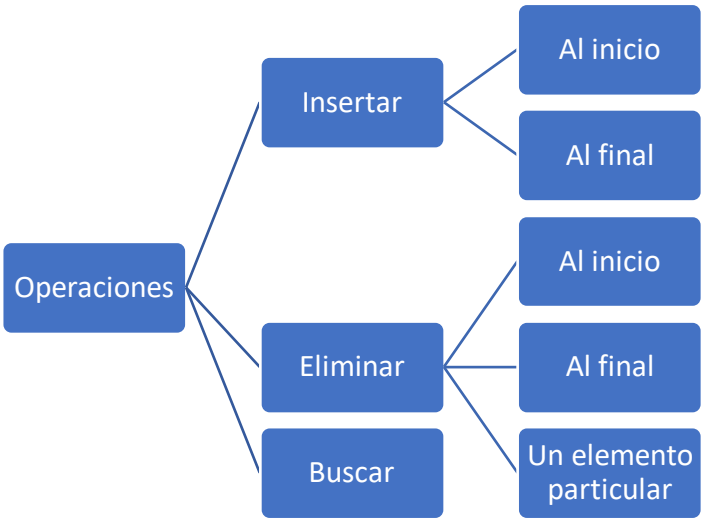
```c
int main(){
    nodo_t *n1 = crea_nodo();
    nodo_t *n2 = crea_nodo();
    nodo_t *n3 = crea_nodo();
    n1->info = 3;
    n2->info = 7;
    n3->info = 1;
    n1->sig = n2;
    n2->sig = n3;
    printf("%d ", n1->info);
    printf("%d ", n1->sig->info);
    printf("%d", n1->sig->sig->info);
    free(n1);
    free(n2);
    free(n3);
    return 0;
}
```



Output

3 7 1

2.1.3 Operaciones en listas enlazadas



2.1.3 Creando un menú de opciones

```c
int main(){
    nodo_t *lista = NULL;
    int opcion = 0, num = 0;
    do{
        opcion = selecciona_opcion();
        switch(opcion){
            case 1: printf("Valor del numero: ");
                scanf("%d", &num);
                insertar_inicio(&lista, num);
                break;
            case 2: printf("Valor del numero: ");
                scanf("%d", &num);
                insertar_final(&lista, num);
                break;
            case 3: eliminar_inicio(&lista);
                break;
            case 4: eliminar_final(&lista);
                break;
            case 5: imprimir_lista(lista);
                break;
            case 6: printf("Valor del numero: ");
                scanf("%d", &num);
                elimina_nodo(&lista, num);
                break;
            case 7: liberar_lista(&lista);
                break;
            case 8: printf("Valor del numero: ");
                scanf("%d", &num);
                printf("%p\n", buscar_dato(lista, num));
                break;
            default: puts("Opcion no valida");
                break;
        }
    }while(opcion != 0);
    return 0;
}
```

2.1.5 Función “int selecciona_opción();”

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

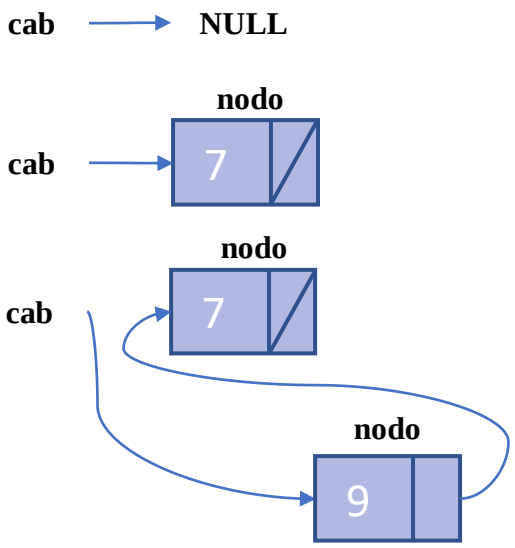
```c
int selecciona_opcion(){
    puts("Selecciona una opcion");
    puts("0. Salir");
    puts("1. Insertar inicio");
    puts("2. Insertar al final");
    puts("3. Eliminar al inicio");
    puts("4. Eliminar al final");
    puts("5. Imprimir toda la lista");
    puts("6. Eliminar nodo arbitrario");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

2.1.5 Función “void insertar_inicio(nodo_t **, int);”

```c
void insertar_inicio(nodo_t **cab, int dato){
    nodo_t *nodo = crea_nodo();
    nodo->info = dato;
    nodo->sig = *cab;
    *cab = nodo;
}
```

Ejemplo:

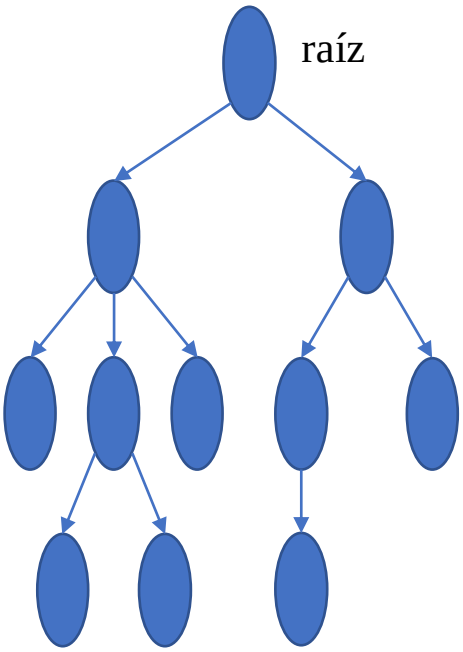
```c
int main(){
    nodo_t *lista = NULL;
    insertar_inicio(&lista, 7);
    insertar_inicio(&lista, 9);
    return 0;
}
```



ÁRBOLES BINARIOS

4

4.1 Conceptos básicos de árboles

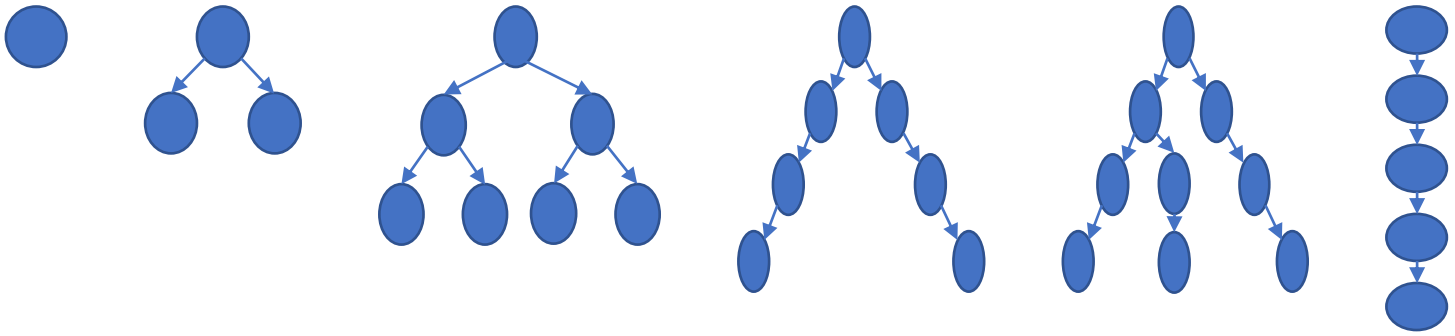


Terminología	
Raíz	Nodo principal, cabecera.
Hijos	Nodos izquierdo y derecho de un nodo.
Padre	Nodo con hijos.
Hermanos	Nodos que tienen el mismo padre.
Antecesor y descendiente	Si podemos ir desde el nodo A hacia el nodo B, entonces A es un antecesor de B, y B es un descendiente de A.
Hoja	Nodos sin 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:



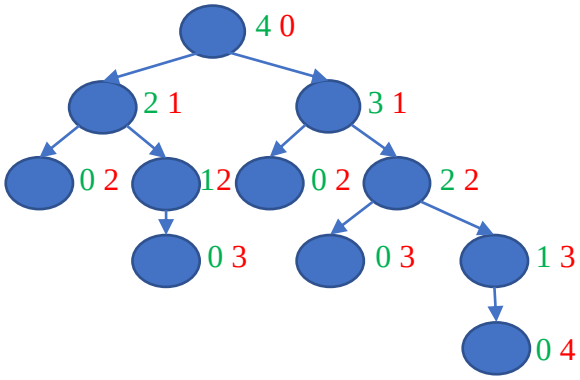
Aplicaciones de los árboles:

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

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



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 $2^{n+1} - 1$ nodos.

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

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

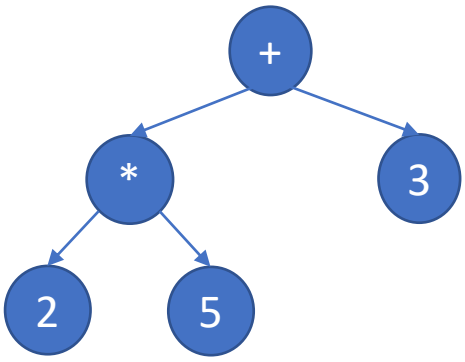
$$\log a^b = b \log a \qquad \log_x x = 1$$

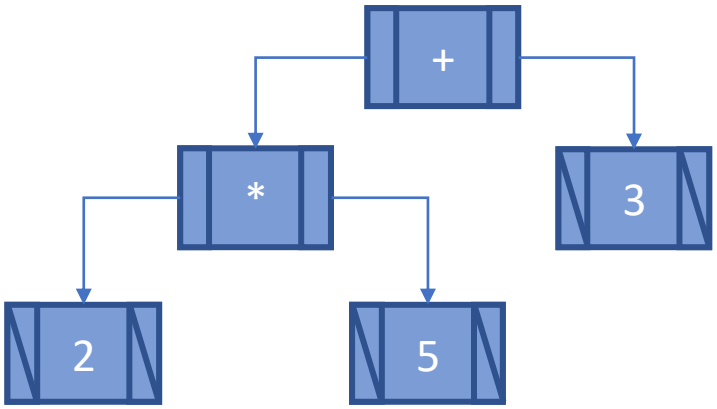
$$\log_2(2^{h+1}) = \log_2(N+1)$$

$$(h+1)\log_2 2 = \log_2(N+1)$$

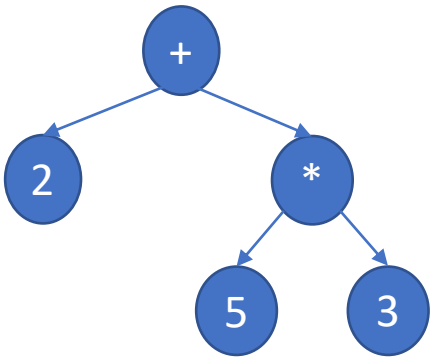
$$h = \log_2(N+1) - 1$$

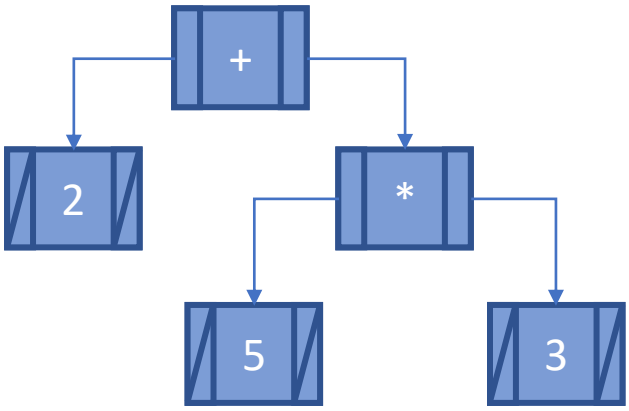
4.2 Árboles binarios de expresiones



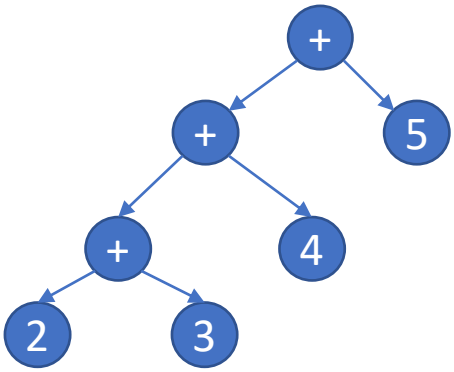


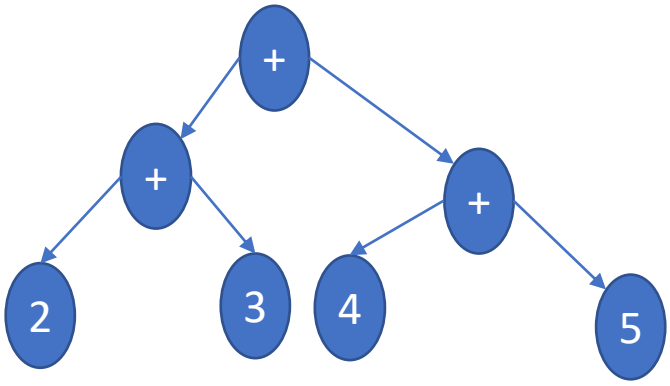
$$2 * 5 + 3 \Rightarrow 13$$





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





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

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

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

4.2.1 Definir la estructura del nodo

```c
struct nodo{
    int info;
    struct nodo *izq;
    struct nodo *der;
};
typedef struct nodo nodo_t;
```