**САА – Упражнение 9**

***Дърво***

Дървото е съвкупност от върхове и клони, които удовлетворяват определени условия.


Двоично дърво е това дърво, в което всеки връх може да има максимум два наследника.

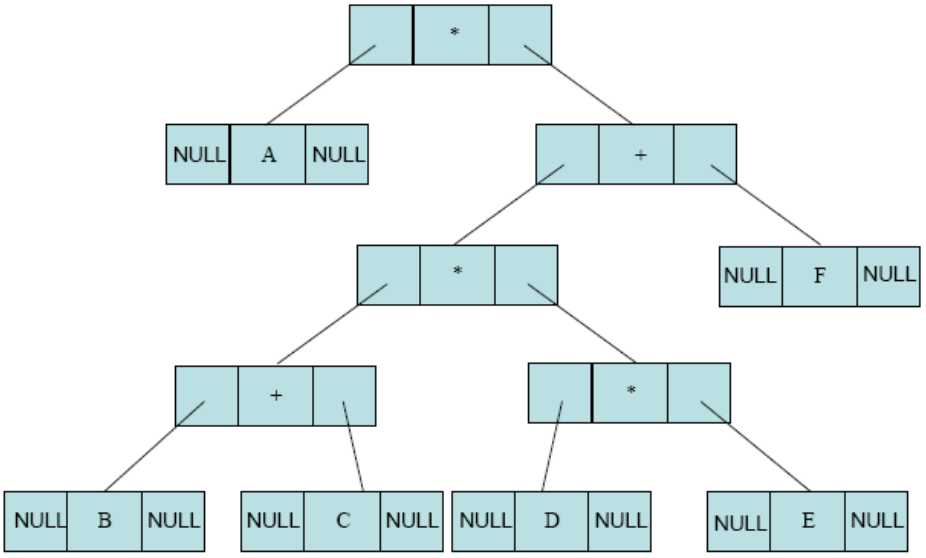
## Начини на обхождане на двоично дърво

- **Префиксно** (обхождане в прав ред): корен, ляво поддърво, дясно поддърво;

- **Инфиксно** (обхождане в междинен ред): ляво поддърво, корен, дясно поддърво;

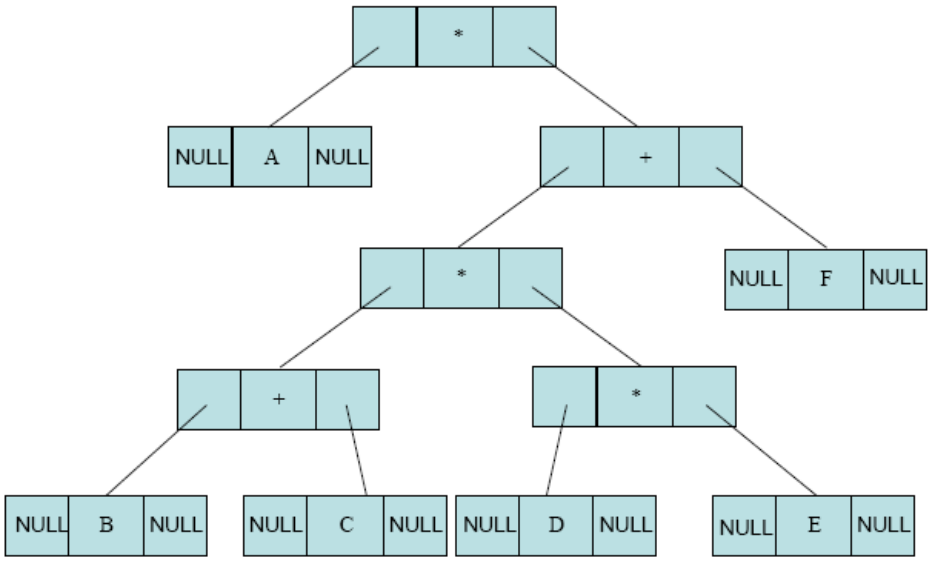
- **Постфиксно** (обхождане в обратен ред): ляво поддърво, дясно поддърво, корен.

## Префиксно обхождане на двоично дърво



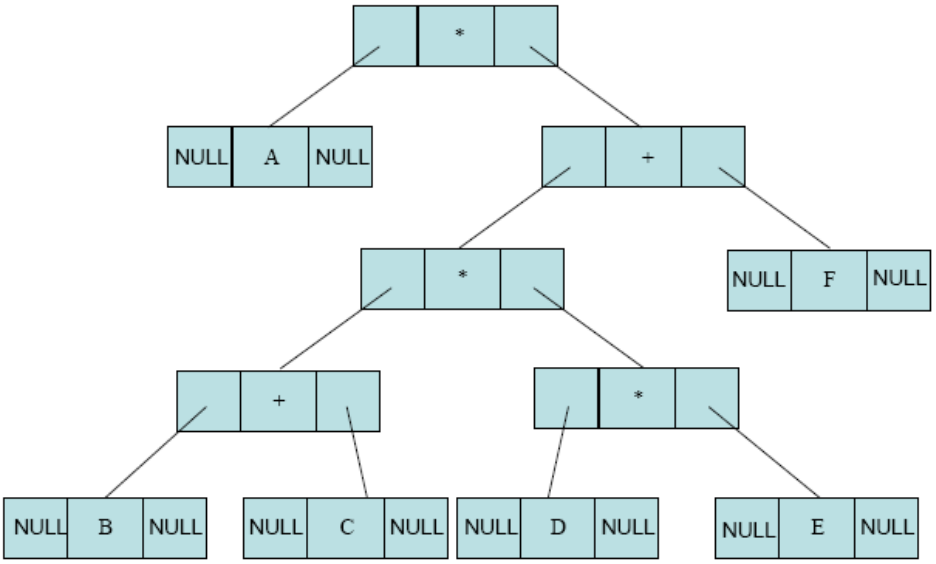

A * (((B + C) * (D * E)) + F) ⇨ * A + * + B C * D E F

## Инфиксно обхождане на двоично дърво



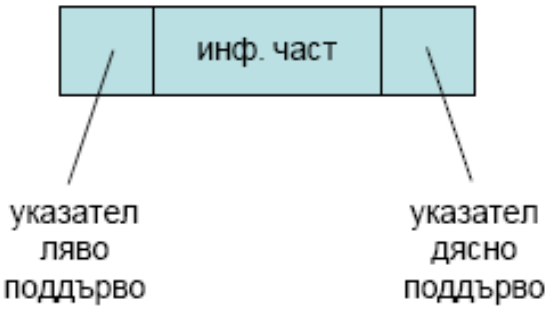

A * (((B + C) * (D * E)) + F) ⇨ A * B + C * D * E + F

## Постфиксно обхождане на двоично дърво




A * (((B + C) * (D * E)) + F) ⇨ A B C + D E * * F+

***Динамично представяне на двоично дърво-*** всеки връх в двоичното дърво има следната структура:




```
struct  node
{
        char key;
        struct node * left;
        struct node * right;
};
```

***Задачи:***

1. Съставете програма, която създава двоично дърво и отпечатва дървото. Използвайте функциите readtree за въвеждане на данни в двоичното дърво и printtree за разпечатване на дървото.

```
void readtree(struct node ** p, char &x )
{
	char y, z;
	struct node * q1, *q2;

	// създава корена, ако няма такъв до момента

	if( *p == NULL )
	{
		printf( "Vyvedi korena" );
			scanf( "%c", &x );
		(*p)=(struct node *)malloc(sizeof(struct node));
		(*p)->key=x;
	}
	getchar();

	// въвежда ляв наследник на текущия връх

	printf("\nVyvedi liav naslednik na %c", x);
	z = getchar();
	getchar();

	// въвежда десен наследник на текущия връх

	printf("\nVyvedi desen naslednik na %c", x);
	y = getchar();

	// създава ляв наследник на текущия връх, ако стойността, въведена за него е
	// различна от '0'

	if (z!='0')
	{
		q1=(struct node *)malloc(sizeof(struct node));
		q1->key=z;
		(*p)->left=q1;
			readtree(&q1,z);
	}
	else
		(*p)->left=NULL;

	// създава десен наследник на текущия връх, ако стойността, въведена за него е
	// различна от '0'

	if (y!='0')
	{
		q2=(struct node *)malloc(sizeof(struct node));
		q2->key=y;
		(*p)->right=q2;
			readtree(&q2,y);
	}
	else
		(*p)->right=NULL;
}
```

```
void printtree(struct node * p, int delta)
{
    int i;
    if (p!=NULL)
    {
            printtree(p->left,delta+1);
                for (i=1;i<=delta; i++)
                printf("\t");
            printf("%c",p->key);
            printf("\n");
                printtree(p->right, delta+1);

    }
}
```

2. Реализирайте програмно всяка от функциите за обхождане на двоично дърво. Разпечатайте резултатите от всяко от обхожданията.
3. Създайте дърво, за което всяко от листата съдържа цифра от 1 до 9, а останалите върхове и корена съдържат операциите +, - и *. Обходете дървото по един от описаните начини, за да получите аритметичния израз във вид, удобен за възприемане от човек (операнд операция операнд). След края на израза изведете знака ‘=’ и отпечатайте стойността на израза.