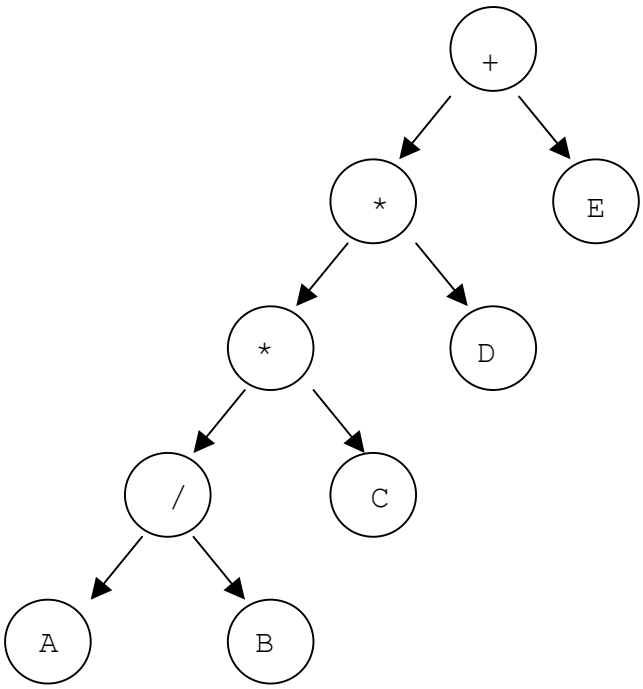
STRUTTURE DATI e LABORATORIO II
Esercitazione n° 12

Albero binario contenente una espressione aritmetica

Scrivere un algoritmo in C che legge una espressione numerica in formato postfix con gli operatori "+, -, *, /" e la rappresenta secondo un albero binario.
Per esempio l'espressione AB/C*D*E+ viene rappresentata come



Visitare poi l'albero con gli algoritmi inorder, preorder, postorder, level_order.

Suggerimento. Rappresentare un nodo dell'albero mediante la seguente dichiarazione

```
struct nodo
      { char dati; struct nodo *sinistro; struct nodo *destro; };
```

Per la creazione dell'albero si può usare il seguente blocco di istruzioni

```
      char c, espr[20];
      int i=0, flag, top=-1;

      scanf("%s", espr);
      while((c=espr[i]) != '\0')
       {
            x=(tree_pointer) malloc(sizeof(struct nodo));
             x->dati = c; x->sinistro= NULL; x->destro =NULL;
             if (c=='+' || c=='-' || c=='*' || c=='/')
                  {     x->destro = deletex(&top);
                        x->sinistro=deletex(&top);
                   }
             add(&top,x);
             ++i;
            }
        root=stack[top];
```

per cui valgono le dichiarazioni

```
   #define MAX_STACK 100
```

```
typedef struct nodo *tree_pointer;
tree_pointer stack[MAX_STACK];
tree_pointer root;

void add(int *top, tree_pointer item)
        /* aggiunge un elemento allo stack globale */
 {
   if(*top >=MAX_STACK-1) {
          printf("\nstack_full");      return;     }
   stack[++*top] = item;
   return;
 }

tree_pointer deletex(int *top)
 {
   if(*top == -1) return NULL;
   else        return stack[(*top)--];
 }
```

L'algoritmo level_order può essere

```
void level_order(tree_pointer ptr)
  {
    void addc(int davanti, int *dietro, tree_pointer item);
    tree_pointer deletec(int *davanti, int dietro);
    int davanti=0;
    int dietro=0;

    if(!ptr) return;
    addc(davanti, &dietro, ptr);
    for(;;) {
          ptr=deletec(&davanti,dietro);
          if(ptr) {
                printf("%c",ptr->dati);
                if(ptr->sinistro)
                      addc(davanti, &dietro, ptr->sinistro);
                if(ptr->destro)
                      addc(davanti, &dietro, ptr->destro);
                }
          else break;
    }
  }
```

con addc() e deletec() dati da

```
#define MAX_CODA 100

void addc(int davanti, int *dietro, tree_pointer item)
  {
    *dietro=(*dietro+1)% MAX_CODA;
    if (davanti==*dietro)  {  return;   }
    coda[*dietro]=item;
 }

 tree_pointer deletec(int *davanti, int dietro)
  {
    if (*davanti == dietro)  {    return NULL;    }
    *davanti=(*davanti+1)%MAX_CODA;
    return coda[*davanti];
  }
```

Buon lavoro!