Znenie zadania:

Pridajte podporu nerekurzívneho prechodu stratégiou PREORDER. Prototypy operácií (súbor bintree.h):

```
void preorderNR(int v);
```

Doplnenie implementácie rekurzívneho prechodu ternárnym stromom.
Úloha: Pridajte implementáciu operácie SetTT pre inicializáciu ternárneho stromu a podporu rekurzívneho prechodu ternárnym stromom operáciami preorderTT a postorderTT. Prototypy operácií (súbor bintree.h):

```
void preorderTT(int root);
void postorderTT(int root);
void SetTT(int* V,int* L,int* M,int* R);
```

Opis riešenia:

Funkcia preorder bez použitia rekurzie funguje tak, že sa pozerá najprv na ľavého potomka, potom na pravého a nakoniec sa vypíše daný uzol. Funkcia najprv prechádza všetkými ľavými potomkami. To sa deje tak, že sa skontroluje či má daný prvok potomka, a ak áno tak sa vypíše a nastaví ako aktuálny uzol. Toto sa opakuje kým existuje ľavý potomok. Ak už neexistuje, tak sa iba vypíše, bez nastavenia daného potomka ako hlavného uzlu. Vtedy sa treba pozrieť či existuje pravý potomok. Ak áno, tento potomok sa nastaví ako hlavný uzol a skočí sa ne vypisovanie ľavých potomkov od tohto uzla. Ak už neexistujú potomkovia, treba sa vrátiť o uzol vyššie. Na to slúži zásobník na ktorého vrchole je uložený uzol, na ktorý sa treba vrátiť. Ak už taký uzol neexistuje, čiže zásobník je prázdny, prešiel sa celý strom a funkcia sa ukončí.

Prechádzanie ternárneho stromu je v zásade také isté ako prechádzanie binárneho stromu, iba sa pridá rekurzívne volanie pre stredného potomka.

Zdrojový kód:

funkcia preorderNR(int v):

```
void preorderNR(int v){
     Stack Tree;
     Tree = CreateStack( 10 );

 LEFT: while (left[v] != NULL){
         printf("%d ",value[v]);
         Push(v,Tree);
         v = left[v];
     }
     printf("%d ",value[v]);

NODE: if (right[v] != NULL){
         v = right[v];
         goto LEFT;
     }

     if (!IsEmpty(Tree)){
         v = Top(Tree);
         Pop(Tree);
         goto NODE;
     }

     DisposeStack( Tree );
}
```

funkcia void preorderTT(int root):

```
void preorderTT(int root){
     printf("%d ",value[root]);
     if(left[root]!=0)preorderTT(left[root]);
     if(middle[root]!=0)preorderTT(middle[root]);
     if(right[root]!=0)preorderTT(right[root]);
}
```

funkcia postorderTT(int root):

```
void postorderTT(int root){
     if(left[root]!=0)postorderTT(left[root]);
     if(middle[root]!=0)postorderTT(middle[root]);
     if(right[root]!=0)postorderTT(right[root]);
     printf("%d ",value[root]);
}
```

funkcia SetTT(int*V, int*L, int*M, int*R):

```
     void SetTT(int* V,int* L,int *M,int* R){
          value=V;
          left=L;
          middle=M;
          right=R;
     }
```