Реализация алгоритма включения элемента в сбалансированное дерево:

```
Procedure Search(x : integer; var Q : P_tr; var H : boolean);
Var
  Q1, Q2 : P_tr;   //H - false
Begin
   if Q = Nil
    then  begin
        New (Q); H := true;
        with Q^ do begin
           key   := x;   count := 1;
           left  := Nil; right := Nil;
           bal   := 0;
                  end;
         end
    else
     if x < Q^.key
      then  begin
        search (x, Q^.left, H); //выросла левая ветвь
        if H then               //если высота увеличилась
          case Q^.bal of
          1 : begin Q^.bal := 0; H := false; end;
          0 : Q^.bal := -1;
         -1 : begin             //балансировка
                Q1 := Q^.left;
                if Q1^.bal = -1
                 then  begin     //однократный LL-поворот
                    Q^.left := Q1^.right;
                    Q1^.right := Q;
                    Q^.bal := 0; Q := Q1;
                       end
                 else   begin     //двукратный LR-поворот
                    Q2 := Q1^.right;
                    Q1^.right:= Q2^.left;  Q2^.left := Q1;
                    Q^.left := Q2^.right;  Q2^.right := Q;
                    if Q2^.bal = -1 then Q^.bal := 1
                                    else Q^.bal := 0;
                   if Q2^.bal = 1 then Q1^.bal := -1
                                  else Q1^.bal := 0;
                    Q := Q2;
                       end;
                 Q^.bal := 0; H := false;
             end                  //балансировка
          end                     //case
             end                  //if x < Q^.key
        else
         if x > Q^.key
           then begin
             search (x, Q^.right, H); //выросла правая ветвь
             if H then            //если высота увеличилась
             case Q^.bal of
             -1 : begin  Q^.bal := 0; H := false; end;
              0 : Q^.bal := 1;
              1 : begin             //балансировка
                    Q1 := Q^.right;
                    if Q1^.bal = 1
                     then begin      //однократный RR-поворот
                      Q^.right := Q1^.left;
                      Q1^.left := Q;
                      Q^.bal := 0; Q := Q1;
                          end
                     else begin      //двукратный RL-поворот
                       Q2 := Q1^.left;
```

```
                        Q1^.left := Q2^.right;  Q2^.right := Q1;
                        Q^.right := Q2^.left;
                        Q2^.left := Q;
                        if Q2^.bal = 1 then Q^.bal := -1
                                       else Q^.bal := 0;
                       if Q2^.bal = -1 then Q1^.bal := 1;
                                        else Q1^.bal := 0;
                        Q := Q2;
                             end;
                        Q^.bal := 0; H := false;
                     end                //балансировка
                end                     //case}
                    end                 //if x > Q^.key
                else
                  Q^.count := Q^.count + 1;

End;
```

Реализация алгоритма удаления элемента из сбалансированного дерева:

```
     Procedure L_Balance(var Q : P_Tr; var H : boolean);
               //H = true, левая ветвь стала короче
      Var
       Q1, Q2 : P_tr;
       B1, B2 : -1..1;
     Begin
        case Q^.bal of
        -1 : Q^.bal := 0;
         0 : begin  Q^.bal := 1; H := false;end;
         1 : begin     //балансировка
                  Q1 := Q^.right;  B1 := Q1^.bal;
                  if B1 >= 0
                   then  begin  //однократный RR-поворот
                           Q^.right := Q1^.left; Q1^.left := Q;
                           if B1 = 0
                            then begin
                               Q^.bal := 1;Q1^.bal := -1;
                               H := false;
                                  end
                            else  begin
                                Q^.bal := 0;Q1^.bal := 0;
                                  end;
                           Q := Q1;
                          end
                     else  begin   //двукратный RL-поворот
                           Q2:= Q1^.left;  B2 := Q2^.bal;
                           Q1^.left := Q2^.right;
                           Q2^.right := Q1;
                           Q^.right  := Q2^.left;Q2^.left  := Q;
                           if B2 = 1 then Q^.bal := -1
                                     else Q^.bal := 0;
                           if B2 = -1 then Q1^.bal := 1;
                                      else Q1^.bal := 0;
                           Q := Q2; Q2^.bal := 0;
                          end;
                end;     //балансировка
               end;      //case
End;    //L_balance
```

```
{_____________________________________}
  Procedure R_Balance(var Q : P_Tr; var H : boolean);
               //H = true, правая ветвь стала короче
  Var
    Q1, Q2 : P_tr;
    B1, B2 : -1..1;
  Begin
      case Q^.bal of
        1 : Q^.bal := 0;
        0 : begin  Q^.bal := -1; H := false;end;
       -1 : begin     //балансировка
             Q1 := Q^.left;  B1 := Q1^.bal;
             if B1 <= 0
                then  begin  //однократный LL-поворот
                       Q^.left := Q1^.right;
                       Q1^.right := Q;
                       if B1 = 0
                          then begin
                                Q^.bal := -1;
                                Q1^.bal := 1;
                                H := false;
                               end
                          else begin
                                Q^.bal := 0;  Q1^.bal := 0;
                               end;
                          Q := Q1;
                            end
                 else  begin   //двукратный LR-поворот
                          Q2 := Q1^.right; B2 := Q2^.bal;
                          Q1^.right := Q2^.left;
                          Q2^.left  := Q1;
                          Q^.left   := Q2^.right;
                          Q2^.right := Q;
                          if B2 = -1 then Q^.bal := 1
                                     else Q^.bal := 0;
                          if B2 = 1 then Q1^.bal := -1;
                                    else Q1^.bal := 0;
                        Q := Q2; Q2^.bal := 0;
                      end;
              end;  //балансировка
         end;  //case
End;    //R_balance

  Procedure Delete(x : integer; var Q : P_tr; var H : boolean);
     Var
       P : P_Tr;      {H = false}
{_____________________________________}
Procedure Del (var R : P_Tr; var H : boolean);
              //H = false
 Begin
   if R^.right <> Nil
      then  begin
               Del(R^.right, H);
               if H then R_Balance(R, H);
            end
      else   begin
                P^.key := R^.key;
                P^.count := R^.count;
                P:=R;
                R := R^.left;
                H := true;
             end;
```

```
   End;     //Del
{_________________________}
 Begin
   if Q = Nil
    then  begin
            writeln ('Дерево пусто.');
            H := false1
          end
     else
         if x < Q^.key
          then begin
             Delete (x, Q^.left, H);
             if H then L_Balance (Q, H);
               end
           else
             if x > Q^.key
               then begin
                  Delete (x, Q^.right, H);
                  if H then R_Balance (Q, H);
                    end
               else begin    //удаление Q^
                  P := Q;
                  if P^.right = Nil
                     then  begin
                         Q := P^.left;  H := true;
                           end
                     else
                      if P^.left = Nil
                         then   begin
                              Q := P^.right; H := true;
                                end
                         else   begin
                             Del (P^.left, H);
                             if H then L_Balance(Q, H);
                                end;
                    End;
  End; //Delete
```

Д,Э. Кнут т 3 «Сортировка и поиск» стр 492-510 – алгоритм и реализация на С