עץ חיפוש בינארי

אוסף פעולות

בניית עץ חיפוש בינארי:

```
//--- בנית עץ חיפוש בינארי ---
public static BinNode<int> BuildSearchTree()
{
	//--- בניית צומת ראשון בעץ ---
	Console.Write ("type 1st number (-1 to finish) --> ");
	int num = int.Parse(Console.ReadLine());
	BinNode<int> bt = new BinNode<int>(num);

	//--- קלט והוספת הצמתים הבאים ---
	Console.Write ("type next number (-1 to finish) --> ");
	num = int.Parse(Console.ReadLine());
	while (num !=  -1)
	{
		Add(bt, num);

		Console.Write ("type next number --> ");
		num = int.Parse(Console.ReadLine());
	}
	return bt;
}

//--- פעולה המוסיפה צומת לעץ חיפוש בינארי באופן איטרטיבי ---
public static void Add (BinNode<int> bt, int x)
{
	BinNode<int> t = new BinNode<int>(x);

	while (x < bt.GetValue()  &&  bt.HasLeft() ||
	       x >= bt.GetValue()  &&  bt.HasRight())
	{
		if (x < bt.GetValue())
			bt = bt.GetLeft();
		else
			bt = bt.GetRight();
	}
	if (x < bt.GetValue())
		bt.SetLeft(t);
	else
		bt.SetRight(t);

}
//--- הצגת צמתי העץ בסדר תוכי ---
public static void PrintInOrder (BinNode<int> bt)
{
	if (bt != null)
	{
		PrintInOrder(bt.GetLeft());
		Console.Write (bt.GetValue() + ", ");
		PrintInOrder(bt.GetRight());
	}

}
```

הוספת צומת לעץ חיפוש בינארי - פתרון רקורסיבי

```
//--- פעולה המוסיפה צומת לעץ חיפוש בינארי באופן רקורסיבי ---
public static void Add (BinNode<int> bt, int x)
{
     if (x < bt.GetValue())
     {
        if (! bt.HasLeft())          //    אם אין בן שמאלי
                bt.SetLeft(new BinNode<int>(x));  // הוספת הצומת כבן שמאלי
        else
            Add (bt.GetLeft(), x);          //  המשך חיפוש המקום המתאים בצד שמאל
     }
     else    // x >= bt.GetValue()
     {
        if (! bt.HasRight())         //    אם אין בן ימני
            bt.SetRight(new BinNode<int>(x));   // הוספת הצומת כבן ימני
        else
            Add (bt.GetRight(), x);         //  המשך חיפוש המקום המתאים בצד ימין
     }
}
```

חיפוש איבר בעץ חיפוש בינארי:

- **פתרון רקורסיבי:**

```
//--- פעולה המחזירה "אמת" אם x נמצא בעץ חיפוש בינארי bt ---
//---                                                  אחרת "שקר" ו-"שקר" ---
public static bool Exist (BinNode<int> bt, int x)
{
      if (bt == null)
            return false;
      if (bt.GetValue() == x)
            return true;
      if (x < bt.GetValue())
            return Exist (bt.GetLeft(), x);
      return Exist (bt.GetRight(),x);
}
```

- **פתרון איטרטיבי:**

```
//--- פעולה המחזירה "אמת" אם x נמצא בעץ חיפוש בינארי bt ---
//---                                                  אחרת "שקר" ו-"שקר" ---
public static bool Exist (BinNode<int> bt, int x)
{
      while (bt != null)
      {
            if (bt.GetValue() == x)
                  return true;

            if (x < bt.GetValue())
                  bt = bt.GetLeft();
            else
                  bt = bt.GetRight();
      }
      return false;

}
```

החזרת הפנייה לצומת:

```
//--- פעולה המקבלת עץ חיפוש ומספר, ומחזירה הפנייה לצומת. ---
//---                    אם הערך לא נמצא בעץ יוחזר null ---
public static BinNode<int> GetNode (BinNode<int> bt, int x)
{
      if (bt == null  ||  bt.GetValue() == x)
            return bt;
      if (x < bt.GetValue())
            return GetNode (bt.GetLeft(), x);
      return GetNode (bt.GetRight(),x);

}
```

מציאת הערך הגדול ביותר בעץ חיפוש:

- **פתרון רקורסיבי:**

```
//--- מציאת האיבר הגדול ביותר בעץ חיפוש ---
//---                 הנחה: העץ אינו null ---
public static int Biggest (BinNode<int> bt)
{
      if (bt.HasRight())
            return bt.GetValue();
      return Biggest(bt.GetRight());
}
```

מציאת הערך הקטן ביותר בעץ חיפוש:

- **פתרון איטרטיבי:**

```
//---  מציאת האיבר הקטן ביותר בעץ חיפוש ---
//---                 הנחה: העץ אינו null ---
public static int Smallest (BinNode<int> bt)
{
      while (bt.HasLeft())
            bt = bt.GetLeft();
      return bt.GetValue();
}
```

מציאת ההורה הישיר של הצומת בעץ חיפוש:

- **פתרון רקורסיבי:**

```
//--- פעולה המחזירה הפנייה להורה של צומת נתון בעץ חיפוש --
public static BinNode<int> Parent
                  (BinNode<int>bt, BinNode<int>node)
{
      if (bt==null || bt.GetLeft() == node || bt.GetRight() == node)
            return bt;
      if (node.GetValue() < bt.GetValue())
            return Parent(bt.GetLeft());
      return Parent(bt.GetRight());
}
```

- **פתרון איטרטיבי:**

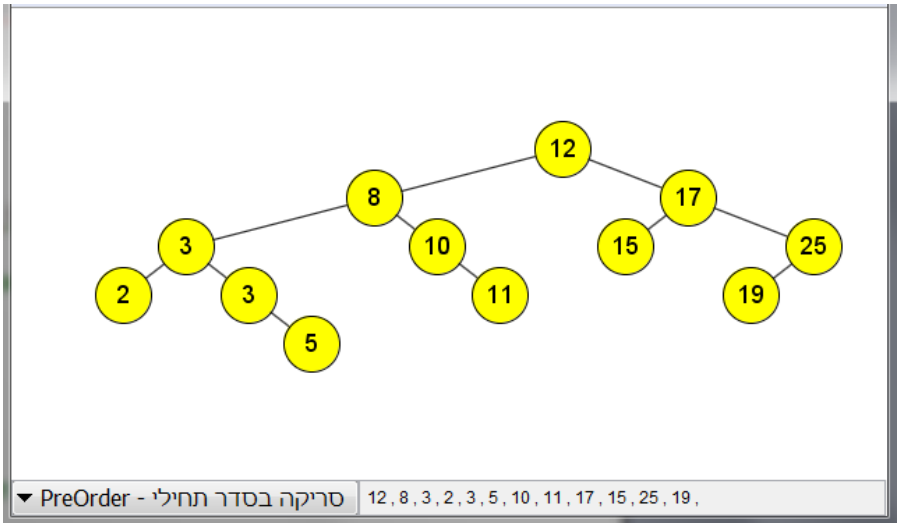
```
//--- פעולה המחזירה הפנייה להורה של צומת נתון בעץ חיפוש --
public static BinNode<int> parent
                  (BinNode<int>bt, BinNode<int>node)
{
      while (bt != null)
      {
            if (bt.GetLeft() == node || bt.GetRight() == node)
                  return bt;

            if (node.GetValue() < bt.GetValue())
                  bt = bt.GetLeft();
            else
                  bt = bt.GetRight();
      }
      return bt;

}
```

הפנייה לצומת בעל הערך העוקב בסדר תחילי בצומת בעץ חיפוש:

```
//--- האיבר העוקב - פעולה המקבלת הפנייה לצומת בעץ חיפוש ---
//---        ומחזירה הפניה לצומת העוקב לו  (בסדר עולה).  ---
//---
//---      bt העץ עצמו , node הצומת שאת העוקב לו מחפשים ---
//---                 null יוחזר ,אם אין עוקב לערך בעץ זה ---
public static BinNode<int> Successor
                  (BinNode<int> bt, BinNode<int> node)
{
      //--- אם הכי גדול בעץ, אין לו עוקב ---
      if (node.GetValue() == Biggest(bt))
            return null;

      BinNode<int> t = node.GetRight();

      //--- אם יש בן ימני, נחפש את המינימום בעץ שבראשו הבן הימני ---
      if (t != null)
      {
            while (t.HasLeft())
                  t = t.GetLeft();
      }
      else //--- חיפוש ההורה באבות הקדמונים ---
      {
            t = Parent (bt, node);
            while (t != bt && t.GetRight() == node)
            {
                  node = t;
                  t = Parent (bt, node);
            }
      }
      return t;

}
```

למשל: העוקב ל- 5 הוא 8

למשל: העוקב ל- 17 הוא 19



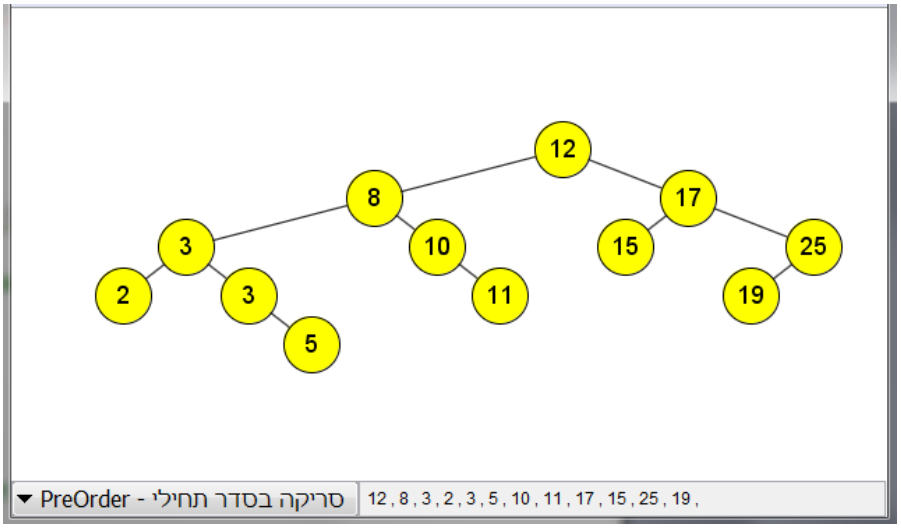
פתרון פשוט יותר: סרוק את העץ לתוך רשימה ומצא את העוקב-ברשימה לצומת.

הפנייה לצומת בעל הערך הקודם בסדר תחילי בצומת בעץ חיפוש:

```
//--- האיבר הקודם - פעולה המקבלת הפנייה לצומת בעץ חיפוש ---
//---        ומחזירה הפניה לצומת הקודם לו  (בסדר עולה).    ---
//---
//---      bt העץ עצמו , node הצומת שאת הקודם לו מחפשים ---
//---                 null אם אין קודם לערך בעץ זה, יוחזר ---
public static BinNode<int> Predeccessor
                  (BinNode<int> bt, BinNode<int> node)
{
      //--- אם הכי קטן בעץ, אין לו קודם ---
      if (node.GetValue() == Smallest(bt))
            return null;

      BinTreeNode<int> t = node.GetLeft();

      //--- אם יש בן שמלי, נחפש את המקסימום בעץ שבראשו הבן השמלי ---
      if (t != null)
      {
            while (t.HasRight())
                  t = t.GetRight();
      }
      else //--- חיפוש ההורה באבות הקדמונים ---
      {
            t = Parent (bt, node);
            while (t != bt && t.GetLeft() == node)
            {
                  node = t;
                  t = Parent (bt, node);
            }
      }
      return t;
}
```

למשל: הקודם ל- 8 הוא 5

למשל: הקודם ל- 19 הוא 17



פתרון פשוט יותר: סרוק את העץ לתוך רשימה ומצא את הקודם-ברשימה לצומת.