УНИВЕРСИТЕТ ПО БИБЛИОТЕКОЗНАНИЕ И ИНФОРМАЦИОННИ ТЕХНОЛОГИИ

КУРСОВА РАБОТА

Софтуерно приложение „Визуализация на двоично дърво“

„Структури от данни и алгоритми“ специалност ИКН 2020/2021

Студент:

/Филип Филипов/

ИКН, 2 група, ФН:46249з

Преподавател:

/Проф. Ив. Иванов/

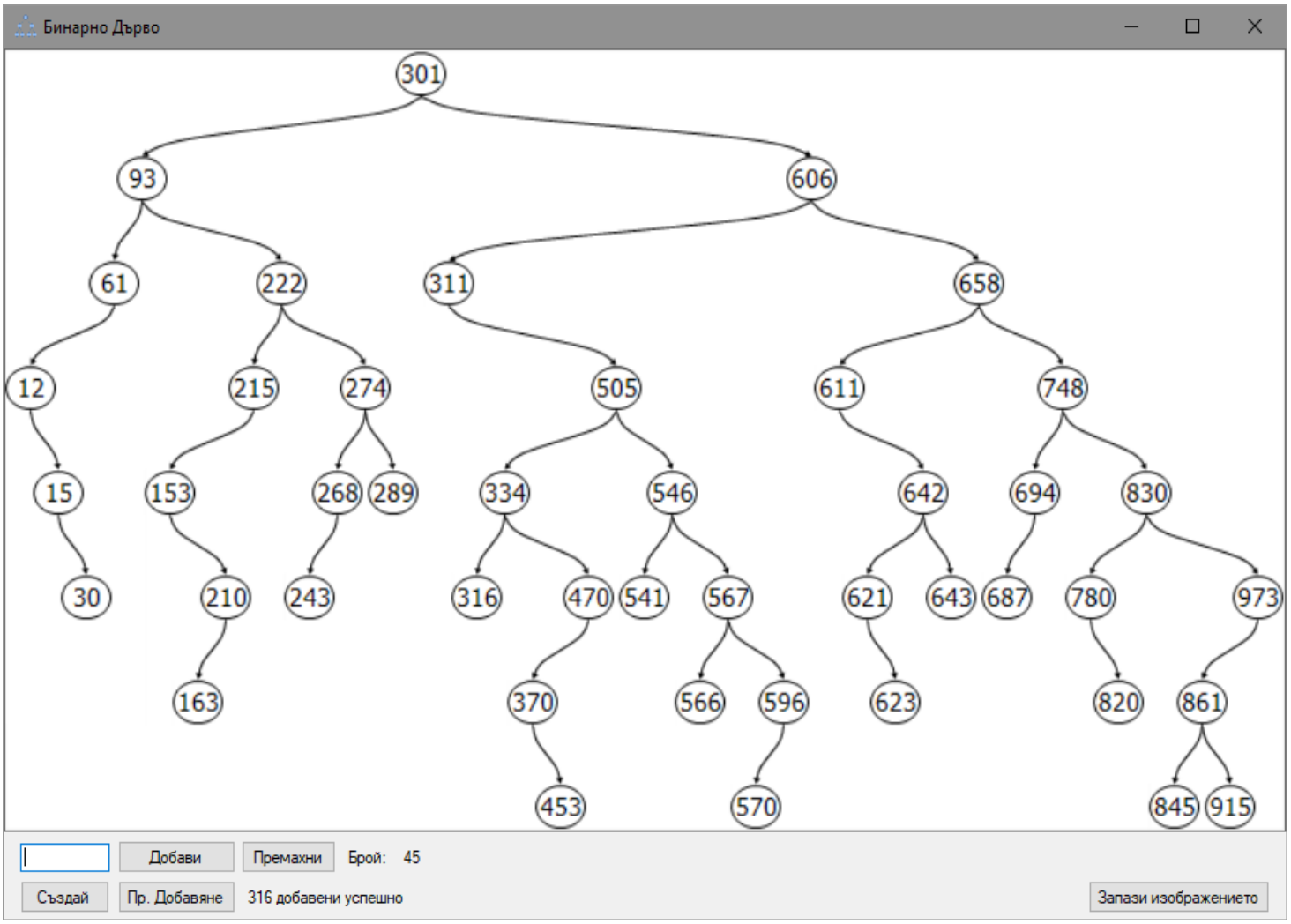
София

2021

Съдържание

Софтуерно приложение „Визуализация на двоично дърво“

Курсовата работа реализира Софтуерно приложение „Визуализация на двоично дърво“. Приложението е реализирано чрез Windows Forms, C# във Microsoft Visual Studio 2017, без да са използвани външни библиотеки растерно изображение с помощта на GDI +. Софтуерното приложение „Визуализация на двоично дърво“ създава двоичното дърво, съдържащо неограничен брой възли, възлите могат да бъдат премахнати, добавени, търсени. Приложението позволява запазване на изображението.



Фигура 1 Софтуерното приложение „Визуализация на двоично дърво“

Всеки възел в двоичното дърво има уникална стойност. например **301** в горната част на изображението е уникална стойност за кореновия възел на дървото.

Правилата за добавяне на нов възел към дървото са:

- Започвайки от основния възел:

- ако стойността на възела е по-малка от стойността на корена, тя ще бъде добавена към левия възел на коренния възел
- ако стойността на възела е по-голяма от стойността на корена, тя ще бъде добавена към десния възел на коренния възел

Този алгоритъм се прилага и за всеки следващ възел.

Правилата за премахване на възел от двоичното дърво са:

- възелът няма дете - просто се премахва
- възелът има ляв наследник- лявото дете на маркирания възел ще заеме позицията му върху дървото
- възелът има дясно дете, а дясното дете няма ляв наследник- дясното дете на възела ще заеме позицията на маркирания възел в дървото
- възелът има дясно дете, а дясното дете има ляв наследник - най-ляво позиционирания наследник на дясното дете ще бъде премахнато (премахването на този възел ще доведе до рекурсивен алгоритъм) и ще заеме позицията на маркирания възел.

Основна функционалност и примерен код

Софтуерното приложение „Визуализация на двоично дърво“ създава двоичното дърво, съдържащо неограничен брой възли и позволява запазване на изображението. Приложението създава функционалността възлите да бъдат премахнати, добавени. Чрез натискане на бутона „***Добави***“ стойността на текстовото поле се добавя като възел към двоичното дърво. След избор на бутона „***Създай***“ ще бъде създадено ново двоично дърво. Чрез натискане на бутона „***Премахни***“ от дървото се премахва възелът, съдържащ стойността на текстовото поле.

След избор на бутона "***Пр. добавяне***" към дървото ще бъде добавена произволна стойност като възел. Чрез натискане на бутона "***Запази изображението***" текущото изображение ще бъде запазено на диска.

За да се създаде дърво и да се визуализира се изпълнява следния код:

```
private BinaryTree _tree;
private bool _acting = false;
private bool _paintAgain = false;
void PaintTree()
{
```

```
        if (_tree == null) return;
        pictureBox1.Image = _tree.Draw();
    }

    private void btnCreate_Click(object sender, EventArgs e)
    {
        _tree = new BinaryTree();
        lblEvents.Text = @"ново бинарно дърво";
        PaintTree();
    }
```

Примерен код 1 „Създаване на дърво“

За да се добави възел с уникален номер се изпълнява метод ***Add*** със следния код:

```
public void Add(int val)

{
    if (val < Value)
    {
        if (Left == null)
            Left = new Node(val);
        else
            Left.Add(val);
        IsChanged = true;
    }
    else if (val > Value)
    {
        IsChanged = true;
        if (Right == null)
            Right = new Node(val);
        else
            Right.Add(val);
    }
}
```

Примерен код 2 „Добавяне на възел“

Използва се метод ***Remove*** за да се премахнат възли от дървото и метод ***Image Draw*** за визуализиране. Методът ***Image Draw*** „изрисува“ всеки възел и всички неговите деца, визуализирането е рекурсивно.

```
public Image Draw(out int center)
        {
            center = _lastCenter;
            if (!IsChanged)
                return _lastImage;

            var lCenter = 0;
            var rCenter = 0;
            Image lImg = null, rImg = null;
            if (Left != null)
                lImg = Left.Draw(out lCenter);
            if (Right != null)
                rImg = Right.Draw(out rCenter);

            var me = new Bitmap(40, 40);
            var g = Graphics.FromImage(me);
```

```
            g.SmoothingMode = SmoothingMode.HighQuality;
            var rcl = new Rectangle(0, 0, me.Width - 1, me.Height - 1);
            g.FillRectangle(Brushes.White, rcl);
            g.FillEllipse(new LinearGradientBrush(new Point(0, 0),
                          new Point(me.Width, me.Height), Color.Gold, Color.Black), rcl);
            var lSize = new Size();
            var rSize = new Size();
            var under = (lImg != null) || (rImg != null);
            if (lImg != null)
                lSize = lImg.Size;
            if (rImg != null)
                rSize = rImg.Size;
            var maxHeight = lSize.Height;
            if (maxHeight < rSize.Height)
                maxHeight = rSize.Height;

            var resSize = new Size
            {
                Width = me.Size.Width + lSize.Width + rSize.Width,
                Height = me.Size.Height + (under ? maxHeight + me.Size.Height : 0)
            };
            var result = new Bitmap(resSize.Width, resSize.Height);
            g = Graphics.FromImage(result);
            g.SmoothingMode = SmoothingMode.HighQuality;
            g.FillRectangle(Brushes.White, new Rectangle(new Point(0, 0), resSize));
            g.DrawImage(me, lSize.Width, 0);
            g.DrawString(Value.ToString(), new Font("Tahoma", 14),
              Brushes.White, lSize.Width + 5, me.Height / 2f - 12);

            center = lSize.Width + me.Width / 2;
            var pen = new Pen(Brushes.Black, 2.5f)
            {   EndCap = LineCap.ArrowAnchor,StartCap = LineCap.Round      };
            float x1 = center;
            float y1 = me.Height;
            float y2 = me.Height * 2;
            float x2 = lCenter;
            var h = Math.Abs(y2 - y1);
            var w = Math.Abs(x2 - x1);
            if (lImg != null)
            {
                g.DrawImage(lImg, 0, me.Size.Height * 2);
                var points1 = new List<PointF>
{
    new PointF(x1, y1), new PointF(x1 - w/6, y1 + h/3.5f),
    new PointF(x2 + w/6, y2 - h/3.5f), new PointF(x2, y2),
};
                g.DrawCurve(pen, points1.ToArray(), 0.5f);
            }
            if (rImg != null)
            {
                g.DrawImage(rImg, lSize.Width + me.Size.Width, me.Size.Height * 2);
                x2 = rCenter + lSize.Width + me.Width;
                w = Math.Abs(x2 - x1);
                var points = new List<PointF>
{
    new PointF(x1, y1),new PointF(x1 + w/6, y1 + h/3.5f),
    new PointF(x2 - w/6, y2 - h/3.5f),  new PointF(x2, y2)
}
                g.DrawCurve(pen, points.ToArray(), 0.5f);
            }
            IsChanged = false;
            _lastImage = result;
            _lastCenter = center;
            return result;
             }
```

Примерен код 3 „Визуализация на дърво”

Примерен код и фигури