Teoretyczne i technologiczne podstawy multimediów	**TITPM04**	**Akademia Nauk Stosowanych**
06.11.2022	**L2**	**Szymon Zwoliński**

1. Kodowanie Kodowanie Arytmetyczne:

Kodowanie arytmetyczne polega na utworzeniu podziału ciągu wejściowego na odpowiednie części dziesiętne w zakresie [0,1] w zależności od częstotliwości występowania znaku. Następnie dla każdego znaku z osobna poszerzamy zakres.
Na początku tworzony jest zakres ogólny (A, B, R, D) i częstotliwości występowania. Następnie dla każdego znaku w danym słowie (ABRAADABRA), zakres każdej pozostałej liczby jest odpowiednio zmniejszony.
W podanym przypadku na początku zmniejszamy zakres A, aby wpasować tam częstotliwości występowania znaków (A, B, R, D). Następnym krokiem jest zmniejszenie zakresu B, aby wpasować tam częstotliwości występowania znaków. Krok ten powtarzamy do skończenia się danego słowa. Uzyskujemy przedział, w którym podany znak koduje podane przez nas słowo.

```
Podaj ciąg znaków:
ABRAADABRA
{A ; 0,5}
{B ; 0,2}
{R ; 0,2}
{D ; 0,1}
A{0 , 0,5}
B{0,5 , 0,7}
R{0,7 , 0,9}
D{0,9 , 1,0}
```

```
ilosc operacji: 11
A{0 , 0,5}
B{0,5 , 0,7}
R{0,7 , 0,9}
D{0,9 , 1,0}

A{0 , 0,25}
B{0,25 , 0,35}
R{0,35 , 0,45}
D{0,45 , 0,50}

A{0,50 , 0,625}
B{0,625 , 0,675}
R{0,675 , 0,725}
D{0,725 , 0,750}
```

```
R{0,838809555950011484198529407O , 0,8388095559652702732610294070}
D{0,8388095559652702732610294070 , 0,8388095559728996677922794070}
A{0,8388095559728996677922794070 , 0,8388095559919731541204044070}
B{0,8388095559919731541204044070 , 0,8388095559996025486516544070}
```

```
public List<Dic> createDic(string input)
{
    List<Dic> symbolDic = new List<Dic>();
    int inputLen = input.Length;
    decimal intervalTempValue;
    int i = 0;
    foreach (char oneChar in input.Distinct())
    {
        intervalTempValue = (decimal)(input.Count(x => x == oneChar)) / (decimal)inputLen;
        if (i == 0)
        {
            symbolDic.Add(new Dic { symbol = oneChar, interval = intervalTempValue, intervalFirstValue = 0, intervalSecondValue = Math.Abs(1 - intervalTempValue) });
        }
        else
        {
            symbolDic.Add(new Dic { symbol = oneChar, interval = intervalTempValue, intervalFirstValue = symbolDic[i-1].intervalSecondValue, intervalSecondValue = symbolDic[i-1].intervalSecondValue+ intervalTempValue });

        }
        i++;
    }
    return symbolDic;
}
```

```
public void divideDic(char input, List<Dic> symbolDic, int start, int end)
{
    int actualPositionOfDivide = 0;
    for (int i = start; i != end; i++)
    {
        if (symbolDic[i].symbol == input && !usedPositions.Contains(i))
        {
            actualPositionOfDivide = i;
            usedPositions.Add(i);
            break;
        }
    }

    int tempIToRecreateDic = 0;
    int counterOfElementsInList = symbolDic.Count();
    for (int i = start; i != end; i++)
    {
        if (tempIToRecreateDic == 0)
        {
            symbolDic.Add(new Dic
            {
                symbol = symbolDic[tempIToRecreateDic].symbol,
                interval = symbolDic[actualPositionOfDivide].interval * symbolDic[tempIToRecreateDic].interval,
                intervalFirstValue = symbolDic[actualPositionOfDivide].intervalFirstValue,
                intervalSecondValue = symbolDic[actualPositionOfDivide].intervalFirstValue + (symbolDic[actualPositionOfDivide].interval / (i+2))
            });
        }
        else
        {
            symbolDic.Add(new Dic
            {
                symbol = symbolDic[tempIToRecreateDic].symbol,
                interval = symbolDic[actualPositionOfDivide].interval * symbolDic[tempIToRecreateDic].interval,
                intervalFirstValue = symbolDic[counterOfElementsInList-1].intervalSecondValue,
                intervalSecondValue = symbolDic[counterOfElementsInList - 1].intervalSecondValue  + (symbolDic[actualPositionOfDivide].interval * symbolDic[tempIToRecreateDic].interval)
            });
        }
        counterOfElementsInList++;
        tempIToRecreateDic++;
    }


}
```

```
public static void Main()
{
    Program program = new Program();
    List<Dic> symbolDic = new List<Dic>();
    Console.WriteLine("Podaj ciąg znaków: ");
    string input = Console.ReadLine();
    symbolDic = program.createDic(input);
    foreach (var line in symbolDic)
    {
        line.readDic();
    }
    foreach(var line in symbolDic)
    {
        line.readDicFirstSecondValue();
    }
    int i = 1;
    foreach (var character in input)
    {
        program.divideDic(character, symbolDic, ((input.Count() * i) - input.Count()), (input.Count() * i));
        i++;
    }
    Console.WriteLine("ilosc operacji: " + (i));

    int cntr = 0;
    foreach (var line in symbolDic)
    {
        line.readDicFirstSecondValue();
        cntr++;
        if(cntr==4)
        {
            cntr = 0;
            Console.Write("\n");
        }
    }
}
```