Podstawy programowania – projekt

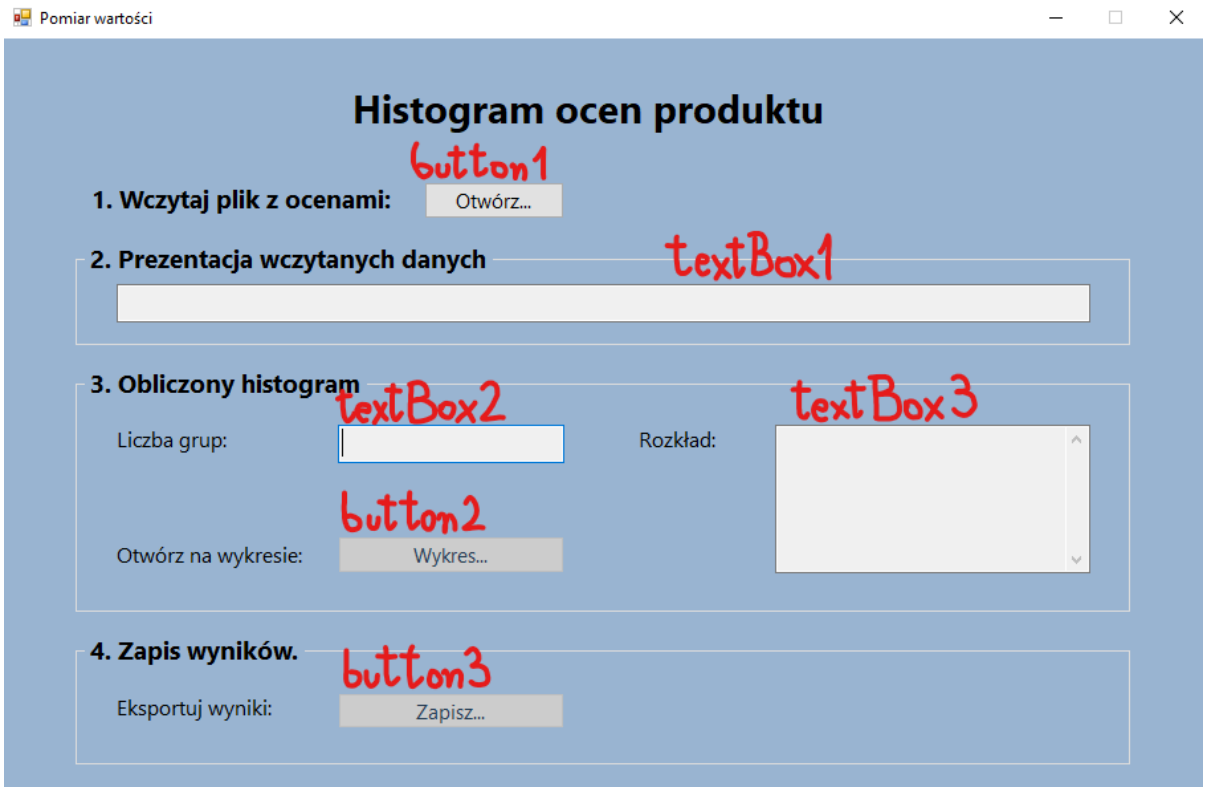
Histogram ocen – Sprawozdanie

Paweł Frąckowiak - gr. Lab. 2

1. Wpierw podszedłem do projektu od strony graficznej. Stworzyłem pierwsze okno i wypełniłem je potrzebnymi kontrolkami. Potem stworzyłem drugie okno i zrobiłem to samo.

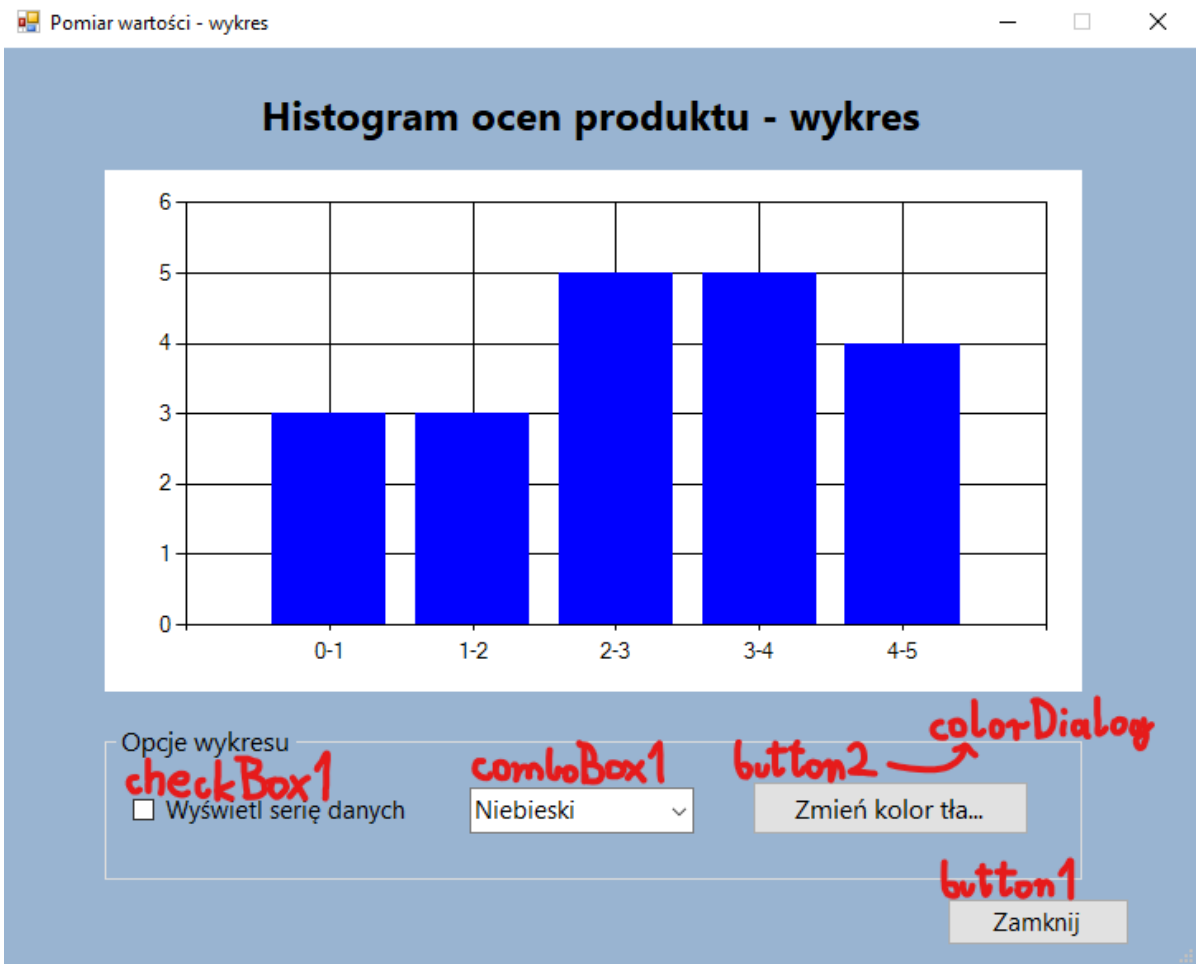
A) Pierwsze okienko – Form1

```
public Form1()
{
    InitializeComponent();
}
```



B) Drugie okienko – Form2

```
private void button2_Click(object sender, EventArgs e)
{
    Form2 F2 = new Form2();
    F2.ShowDialog();
}
```



2. Następnie utworzyłem zmienne typu **public**, które będą dostępne w Form1 jak i w Form2 (mimo bycia zadeklarowanymi w Form1).

```
public static int dolneGrupy = 100, gorneGrupy = 0, iloscGrup = 0, kolor;
public static bool check1;
public static string stringPrzekazany = "", Dane;
public static Color kolortla;
```

- **dolneGrupy**: zawiera zaokrągloną w dół wartość najniższej oceny pobranej z pliku tekstowego.
- **gorneGrupy**: zawiera zaokrągloną w górę wartość najwyższej oceny pobranej z pliku tekstowego.
- **iloscGrup**: zawiera wyliczoną później ilość przedziałów ocen.
- **kolor**: zawiera numer indexu opcji wybranej później w ComboBox1 w Form2.
- **check1**: zawiera wartość true lub false zależnie, czy checkBox1 jest zaznaczony czy nie.
- **kolortla**: zawiera kolor wybrany później w colorDialog1 w Form2.

3. Zaraz potem zająłem się kodem, który był przypisywany do poszczególnych zdarzeń.
    a. **Button1** – Otwarcie pliku

```
private void button1_Click(object sender, EventArgs e)
{
    using (OpenFileDialog openFileDialog = new OpenFileDialog())
    {
        openFileDialog.InitialDirectory = "c:\\";
        openFileDialog.Filter = "txt files (*.txt)|*.txt";
        openFileDialog.FilterIndex = 2;
        openFileDialog.RestoreDirectory = true;

        if (openFileDialog.ShowDialog() == DialogResult.OK)
        {
            //Ścieżka do pliku, którą podaje użytkownik
            string sciezka = openFileDialog.FileName;

            if (openFileDialog.CheckFileExists)
            {
                //Odczytanie zawartości pliku
                stringPrzekazany = File.ReadAllText(sciezka);
                try
                {
                    WyswietlanieDanych();
                    button2.Enabled = true;
                    button3.Enabled = true;
                    textBox1.Enabled = false;
                    textBox2.Enabled = false;
                }
                catch
                {
                    MessageBox.Show("Niewłaściwe dane w pliku.", "Błąd!", MessageBoxButtons.OK, MessageBoxIcon.Error);
                }
            }
            else MessageBox.Show("Nie można otworzyć pliku.", "Błąd!", MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
    }
}
```

Początek kodu służy do stworzenia obiektu i przypisania mu ustawień tak by później móc otworzyć okienko, w którym będą wyszukiwane np. pliki **.txt** na dysku **C:**. Dalej otwieramy okienko i pobieramy ścieżkę pliku podaną przez użytkownika. Jeśli plik istnieje i jest z nim wszystko w porządku to odczytujemy zawartość pliku do publicznej zmiennej **stringPrzekazany**. Potem łapiemy wyjątki za pomocą **try {} catch {}.** Robimy tak, aby zapobiec wysypaniu się programu, gdy będziemy odczytywać dane z pliku i wyświetlać je w kontrolkach w **Form1**. W **try{…}** znajduje się metoda **WyswietlanieDanych**, która właśnie odpowiada za ich poprawne ukazanie na ekranie, a pod tą metodą są włączane lub wyłączane przyciski dla użytkownika by mógł je klikać wtedy i tylko wtedy gdy dane zostaną załadowane poprawnie.

b. **Button3** – Eksport wyniku

```
private void button3_Click(object sender, EventArgs e)
{
    SaveFileDialog saveFileDialog1 = new SaveFileDialog();

    saveFileDialog1.Filter = "txt files (*.txt)|*.txt";
    saveFileDialog1.FilterIndex = 2;
    saveFileDialog1.RestoreDirectory = true;

    if (saveFileDialog1.ShowDialog() == DialogResult.OK)
    {
        String zapisDirectory = saveFileDialog1.InitialDirectory;
        String zapisFileName = saveFileDialog1.FileName;
        String zapisPath = Path.Combine(zapisDirectory, zapisFileName);

        using (StreamWriter eksport = new StreamWriter(zapisPath))
        {
            eksport.WriteLine(label6.Text + " " + textBox2.Text);
            eksport.WriteLine(label8.Text);
            eksport.WriteLine(textBox3.Text);
            eksport.Close();
        }
        if (File.Exists(zapisPath)) MessageBox.Show("Plik zapisano pomyślnie.", "Sukces!", MessageBoxButtons.OK, MessageBoxIcon.Information);
        else MessageBox.Show("Błąd przy zapisie pliku.", "Błąd!", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}
```

Tak jak przy otwarciu, podobnie jest przy zapisie. Tworzymy obiekt, za pomocą którego otwieramy okienko, w którym możemy wybrać lokalizacje pliku do zapisania. W tym okienku możemy również nazwać nasz wyeksportowany plik. Dalej za pomocą **StreamWriter**'a tworzymy plik i zapisujemy w nim nasze wyniki (tu posłużyłem się wartościami w **label**ach i **textBox**ach). Jeśli wszystko podczas tego procesu przebiegło zgodnie to informujemy użytkownika o pomyślnym zapisaniu pliku, a jeśli niezgodnym to o błędzie.

4. **WyswietlanieDanych()** - Odczytywanie danych z pliku i wyświetlnenie ich w poszczególnych kontrolkach.

```
public void WyswietlanieDanych()
{
    //-------------------------------------
    //Resetowanie wcześniejszych ustawień
    textBox1.Text = null;
    textBox2.Text = null;
    textBox3.Text = null;
    dolneGrupy = 100;
    gorneGrupy = 0;
    //-------------------------------------
```

Najpierw zaczynamy od zresetowania ustawień, by przy powtórnym odczytaniu pliku nie pozostały jakiekolwiek wartości ze starego odczytywania.

```
textBox1.Text = stringPrzekazany;
char charSeparator = ';';
string[] result;
result = stringPrzekazany.Split(charSeparator, (char)StringSplitOptions.None);
Double ocena;
int k = 0;
Double[] tablicaOcen = new Double[result.Length - 2];
foreach (var value in result)
{
    Double.TryParse(value, out ocena);
    if (ocena < dolneGrupy) dolneGrupy = Convert.ToInt32(Math.Floor(ocena));
    if (ocena > gorneGrupy) gorneGrupy = Convert.ToInt32(Math.Ceiling(ocena));
    if (Double.TryParse(value, out ocena))
    {
        tablicaOcen[k] = ocena;
        k++;
    }
}
iloscGrup = gorneGrupy - dolneGrupy;
textBox2.Text = Convert.ToString(iloscGrup);
```

Dalej uzupełniamy jednym stringiem **stringPrzekazany** nasz **textBox1** zawartością otwieranego pliku, żebyśmy mieli aktualny wgląd w niego. Potem za pomocą **Split()** rozdzielamy string (separatorem jest **char ‘;’**). Następnie tworzymy tablice z ocenami **tablicaOcen[]**, która będzie już zawierała same wartości liczbowe. Robimy to za pomocą **foreach()** i **TryParse**. Dalej chcemy wyłonić najniższą ocenę zaokrągloną w dół i najwyższą ocenę zaokrągloną w górę, aby znać przedziały ocen (przyda się to do wykresu oraz do wyliczenia ilości grup).

```
//Sortowanie Ocen
double temp = 0;
for (int i = 0; i < tablicaOcen.Length; i++)
{
    for (int j = 0; j < tablicaOcen.Length - 1; j++)
    {
        if (tablicaOcen[j] > tablicaOcen[j + 1])
        {
            temp = tablicaOcen[j + 1];
            tablicaOcen[j + 1] = tablicaOcen[j];
            tablicaOcen[j] = temp;
        }
    }
}
```

Dalej mamy zaimplementowanie najprostszego typu sortowanie, czyli **BubbleSort** (sortowanie bąbelkowe), aby posortować wszystkie nasze oceny od najmniejszej do największej w **tablicaOcen[]**. Wybrałem go, bo jest najprostszy, a naszych ocen nie jest dużo, dlatego nie trzeba będzie długo czekać na wynik.

```
//Liczenie ile jest ocen w poszczególnych grupach
Double dolne = dolneGrupy;
int iloscOcen = 0;
for (int i = 0; i < tablicaOcen.Length; i++)
{
    if (Math.Floor(tablicaOcen[i]) == dolne) iloscOcen++;
    else
    {
        textBox3.Text = textBox3.Text + Convert.ToString(dolne) + "-" + Convert.ToString(dolne + 1) + ": " + Convert.ToString(iloscOcen) + Environment
        iloscOcen = 0;
        dolne++;
        i--;
    }
}
textBox3.Text = textBox3.Text + Convert.ToString(dolne) + "-" + Convert.ToString(dolne + 1) + ": " + Convert.ToString(iloscOcen);
Dane = textBox3.Text;
//-------------------------------------------------

MessageBox.Show("Plik odczytano pomyślnie.", "Sukces!", MessageBoxButtons.OK, MessageBoxIcon.Information);
```

Następnie za pomocą prostej pętli **for()** i **if()**'a możemy policzyć wszystkie nasze oceny w poszczególnych grupach i je wypisać w **textBox3**. Zawartość tego **textBox'**a zapisujemy jako jeden string do publicznej zmiennej **Dane**. Jeśli wszystko do tego momentu w tej metodzie się powiedzie i nie wysypie programu to dostajemy informacje o pomyślnym odczytaniu pliku.

5. Niebawem zająłem się pisaniem kodu dla drugiego okienka, który był odpowiedzialny za wyświetlanie wykresu i edytowanie go, z możliwością zapisania wcześniejszych ustawień wykresu.

```
private void Form2_Load(object sender, EventArgs e)
{
    //wczytanie ustawień z przed zamknięcia okna
    checkBox1.Checked = Form1.check1;
    comboBox1.SelectedIndex = Form1.kolor;
    chart1.ChartAreas["ChartArea1"].BackColor = Form1.kolortla;
    //-----------------------------------------
    string[] result = Form1.Dane.Split(new Char[] { ':', '\n' });
    for (int i = 0; i < result.Length; i+=2)
        chart1.Series["Histogram"].Points.AddXY(Convert.ToString(result[i]), Convert.ToString(result[i + 1]));
}
```

Jak okno się załaduje, pierwsze co robi, to odczytuje wcześniejsze ustawienia dla naszego wykresu, które zapisaliśmy podczas bieżącego uruchomienia programu. Następnie rozdzielamy **Dane** za pomocą **Split()**. Separatorami są ':' i '\n', czyli enter (nowa linijka). Tworzymy wykres **chart1** i dodajemy pierwszą wartość tabeli na oś X, a drugą wartość tabeli na oś Y. Wszystko to zostało zapętlone **for**em zwiększającym się o 2.

a. **CheckBox1**

```
private void checkBox1_CheckedChanged(object sender, EventArgs e)
{
    if (chart1.Legends[0].Enabled == true) chart1.Legends[0].Enabled = false;
    else chart1.Legends[0].Enabled = true;
}
```

Tu znajduje się zaznaczanie oraz odznaczanie legendy z wykresu (**seria1**). Prostym ifem sprawdzamy, czy jest widoczna legenda, jeśli tak to ją chowamy, a jak nie to pokazujemy.

b. **ComboBox1**

```
private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
{
    Form1.kolor = comboBox1.SelectedIndex;
    switch (Form1.kolor)
    {
        case 0:
            chart1.Series["Histogram"].Color = Color.Blue;
            break;
        case 1:
            chart1.Series["Histogram"].Color = Color.Green;
            break;
        case 2:
            chart1.Series["Histogram"].Color = Color.Red;
            break;
    }
}
```

Tu znajduje się wybieranie koloru i przypisywanie go do koloru słupków w wykresie. **Switch** odczytuje jaki jest numer indeksu w **comboBox'**ie i wybiera dla niego odpowiednią przypisaną odpowiedź w **case**.

c. **Button2 (colorDialog1)**

```
private void button2_Click(object sender, EventArgs e)
{
    if (colorDialog1.ShowDialog() == DialogResult.OK)
    {
        colorDialog1.ShowHelp = true;
        chart1.ChartAreas["ChartArea1"].BackColor = colorDialog1.Color;
    }
}
```

Tu znajduje się zmiana koloru tła wykresu (za pomocą użycia **colorDialog1**). Ta metoda pozwala na wyświetlenie dodatkowego okienka, w którym możemy wybrać niestandardowy kolor jaki tylko chcemy.

d. **Button1**

```
private void button1_Click(object sender, EventArgs e)
{
    Form1.check1 = checkBox1.Checked;
    Form1.kolor = comboBox1.SelectedIndex;
    Form1.kolortla = colorDialog1.Color;
    this.Close();
}
```

Tu znajduje się zamknięcie okienka z wykresem (**Form2**) za pomocą użycia **this.Close()**. Ale najpierw zapisuje wszystkie nasze wcześniejsze ustawienia dotyczące wykresu, czyli zaznaczenie **checkBox**'a, wybranie koloru (indeksu) w **comboBox**'ie oraz wybranie koloru tła w **colorDialog**. To wszystko zostaje zapisane w publicznych zmiennych zadeklarowanych w **Form1**, żeby potem móc ponownie otworzyć nasz taki sam wykres, który już wcześniej ustawialiśmy.