Projekt

Napisanie programu pozwalającego na naszkicowanie histogramu na podstawie danych pobranych z pliku.

15.05.2021

1.Teoria

Program został stworzony w kompilatorze Visual Studio 2019 w szablonie aplikacji Windows Forms (.NET Framework). Szablon ten umożliwiał nam dodawanie rożnych elementów takich jak: wykres, przyciski, obszary tekstu oraz pozwalał nam działać nich. Użyto języka programowania C#.

W programie zostały wykorzystane różne zdarzenia:

Click – wykorzystywany głownie w przyciskach takich jak: „Wczytaj dane”, „Pokaż wykres”, „Zapisz dane”, „Zmień kolor”, „Wyłącz legendę”, ”Włącz legendę”. Umożliwiło to wykonanie konkretnych działań po kliknięciu danych przycisków.

SelectedIndexChenged – używany jedynie przy modyfikacji wykresu w momencie wyboru danych do ukrycia. Pozwala ono na działaniu w momencie wybrania konkretnego elementu z listy rozwijanej.

Load – występujący w momencie generowania wykresu na 2 oknie programu. Pozwala na działanie w momencie ładowania okna programu.

Klasy wykorzystane w programie:

- OpenFileDialog, która umożliwia otworzenie nowego okna do wyboru pliku oraz oczytanie jego nazwy, lokalizacji oraz ustawienia np. filtru wybieranych plików. W programie używana na przycisku „Wczytaj dane”.

-SaveFileDialog, która umożliwia otworzenie nowego okna do wyboru miejsca bądź konkretnego pliku do zapisu jak również ustawienia filtru. W programie używana na przycisku „Zapisz dane”.

-File, która w programie została użyta do odczytania całego pliku z podanej ścieżki.

-MessageBox, używana do wyświetlania okienek z komunikatami dla użytkownika.

-List, używana do tworzenia listy, do której z łatwością można dodawać nowe elementy.

-Color, pozwalająca na używanie kolorów odwołując się po ich nazwach lub stałych.

Główne metody wykorzystane w programie:
-Parse, użyta do zmienienia typu w tym przypadku z alfanumerycznego na liczby całkowite.
-Try Catach, wykorzystana do przechwytywania wyjątków i obsługiwania ich.

-Split, pozwalająca na podziale tekstu na mniejsze fragmenty.

-ShowDialog, otwierająca nowe okno, użyta w przypadku otwierania pliku zapisu lub wyświetlenia wykresu.

-WriteLine, zapisująca podana linie tekstu do pliku wcześniej wybranego.

-AddXY, używana w odwołaniu do wykresu pozwala na dodanie danych na wykres.

-Invalidate, na nowo szkicuje wykres.

-Add, dodaje element do listy.

2. Zadanie

Na ocenę 4.

Zadaniem programu jest otworzenie pliku, który wybierze użytkownik sprawdzenie czy plik zawiera odpowiednie dane a następnie oczytanie ich rozdzielenie i wpisanie na tablice, z której później dokonuje wyliczeń ile jest w danym przedziale. Program oferuje również wyświetlenie wykresu w osobnym oknie oraz modyfikowanie go po przez zmianę koloru danych ukrycia niektórych danych lub schowanie legendy. Mamy również możliwość na zapisanie podzielonych już ocen do pliku tekstowego.

Zadanie programu na ocenę 4:

- otwieranie pliku tekstowego z ocenami, każdorazowo otwierając okno dialogowe umożliwiające wskazanie pliku,
- parsowanie tekstu i wyodrębnienie wartości liczbowych do oddzielnej tablicy,
- obliczanie histogramu własnoręcznie napisanym algorytmem,
- zapisywanie wyników obliczeń do pliku tekstowego,
- wyświetlanie danych na wykresie, w którym można dokonać przynajmniej 3 modyfikacji.
- zadbanie o obsługę wyjątków związanym z IO,
- blokowanie niektórych pól tylko do odczytu oraz funkcjonalności zanim użytkownik ma prawo je wykonać,
- wyświetlanie komunikatów informujących o stanie IO,
- wyświetlanie wykresu w osobnym oknie.

Kod:

```
public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
        public static Int32[] zliczone;
        public Int32 rozmiar = 0;
```

Fragment kodu 1.Tworzenie zmiennych globalnych.

Będą one używane do przechowywania zliczonych elementów („zliczone”) oraz rozmiaru jaki ma przyjąć tablica przechowującą obliczone wartości („rozmiar”).

```
public Int32 sprawdzRozmiar(Int32[] tab)
        {
            int max = 0;
            for (int i = 0; i < tab.Length; i++)
            {
                if (max < tab[i]) max = tab[i];
            }

            return max;
        }
```

Fragment kodu 2. Metoda sprawdzająca rozmiar.

Sprawdza rozmiar tablicy podanej jako jej argument oraz zwraca go.

```
public Int32[] zliczanie(Int32[] tab)
        {
            Int32[] listaUlozona = new Int32[rozmiar];

            foreach (var item in listaUlozona)
            {
                listaUlozona[item] = 0;
            }

            for (int i = 0; i < tab.Length; i++)
            {
                for (int j = 0; j < rozmiar; j++)
                    if (tab[i] == j) listaUlozona[j]++;
            }
            return listaUlozona;


           }
```

Fragment kodu 3. Metoda zliczająca.

Sprawdza elementy tablicy podanej jako argument metody oraz zlicza ich ilosc zapisując w osobnej tablicy która nastepnie zwraca. W tej metodzie używany jest „rozmiar” który określa rozmiar tablicy z zliczonymi elementami.

```
private void Wczytaj_Click(object sender, EventArgs e)
        {
            OpenFileDialog otworz = new OpenFileDialog();
            otworz.CheckFileExists = true;
            otworz.Filter = "txt file (*.txt)|*.txt";

            if(otworz.ShowDialog() == DialogResult.OK)
            {
                try
                {
                    string path = otworz.FileName;
                    string czytane = File.ReadAllText(path);

                    Int32[] liczby = czytane.Split(',', ';', ' ', '.').Select(s =>
Int32.Parse(s)).ToArray();

                    rozmiar = sprawdzRozmiar(liczby) + 1;
                    zliczone = zliczanie(liczby);

                    pokazWykres.Visible = true;
                    zapisz.Visible = true;
```

```
                label1.Visible = true;
                label3.Visible = true;
                MessageBox.Show("Prawidłowo załadowano plik:" + otworz.FileName);

            }
            catch(Exception)
            {
                pokazWykres.Visible = false;
                zapisz.Visible = false;
                label1.Visible = false;
                label3.Visible = false;
                MessageBox.Show("Nie udało się załadowac pliku!");
                MessageBox.Show("Wybrano zły plik. Przykładowy plik:(1,2,4.5;6,3
7)");
            }
        }

           }
```

Fragment kodu 4.Metoda wykonywana po naciśniecu przycisku „Wczytaj dane”.

Na sam początek tworzony jest obiekt klasy OpenFileDialog który umożliwy operowanie na tej klasie, oraz otworzenie okienka z wyborem pliku do załadowania. Po załadowaniu prawidłowego pliku zawartość pliku zostaje zapisana na zmienną która następnie zostaje podzielona za pomoca metody Split na mniejsze fragmeny oraz typ tych fragmentów zostaje zmieniony z tekstowego na liczbowy oraz przypisany na tablice. Tablice tę nastepnie przekazujemy do wcześniej przedstawionych metod „sprawdzRozmiar” oraz „zliczanie” które zapisuja wyniki na zmiennych globalnych. Następnie zostają włączone pozostałe pola znajdujace się w programie oraz zostaje wyświetlona informacja o prawidłowym załadowaniu pliku dla użytkownika. Jednak w momencie załadowania złego pliku pola pozostaną niewidoczne a użytkownik dostanie informacje o nie udanym załadowaniu pliku.

```
private void pokazWykres_Click(object sender, EventArgs e)
        {
            Form2 wykres = new Form2();
            wykres.ShowDialog();

           }
```

Fragment kodu 5.Metoda wykonywana po naciśniecu przycisku „Pokaż wykres”.

Po wćiśnięciu przycisku zostanie otwarte nowe okno programu.

```
private void zapisz_Click(object sender, EventArgs e)
        {
            SaveFileDialog zapis = new SaveFileDialog();
            zapis.Filter = "txt file (*.txt)|*.txt";

            if (zapis.ShowDialog() == DialogResult.OK)
            {
                StreamWriter sw = new StreamWriter(zapis.FileName);
                for(int i = 0; i<=rozmiar; i++)
                {
                    if (i + 1 < rozmiar)
                    {
                        int z = i + 1;
                        int odp = zliczone[i] + zliczone[i + 1];
                        sw.WriteLine("Oceny" + i + "-" + z + ": " + odp);
                    }
                    else break;

                }
                MessageBox.Show("Zapisano dane do pliku: " + zapis.FileName);

                sw.Close();

            }

           }
```

Fragment kodu 6.Metoda wykonywana po naciśniecu przycisku „Zapisz dane”.

Podobnie jak w przypadku otwieranie jest tworzony obiekt tylko tym razem klas SaveFileDialog który działa podobnie jak przy otwieraniu czyli pozwala nam na odczytanie scieżki oraz nazwy do danego pliku. Następnie dzieki StreamWriter możemy zapisac wyznaczone i pogrupowane dane które znajduja się w tablicy „zliczone” oraz na koniec wyświetlic komunikat o poprawnym zapisaniu pliku.

```
private void Form2_Load(object sender, EventArgs e)
        {
            for (int i = 0; i < Form1.zliczone.Length; i++)
            {
                Wykres.Series[0].Points.AddXY(i, Form1.zliczone[i]);
                ListaDoUkrycia.Items.Add("Dana: " + i);
            }

           }
```

Fragment kodu 7.Metoda wykonywana podczas ładowania 2 formularza.

W tym momencie zostaja dodane dane na wykres oraz elementy do listy rozwijanej która będzie uzywana podczas ukrywania danych.

```
private void ZmienKolor_Click(object sender, EventArgs e)
        {
            if (radioCzerwony.Checked == true) Wykres.Series[0].Color =
Color.FromName("Red");
            else if (radioZielony.Checked == true) Wykres.Series[0].Color =
Color.FromName("Green");
            else if (radioNiebieski.Checked == true) Wykres.Series[0].Color =
Color.Blue;

               }
```

Fragment kodu 8.Metoda wykonywana w momencie klikniecia przycisku „Zmień kolor”.

Metoda ta sprawdza jaki z przyciskow typu radio jest akutalnie wybrany i w zależności od przycisku zostaje zmienony kolor seri na wykresie.

```
private void wlLegende_Click(object sender, EventArgs e)
        {
            Wykres.Legends[0].Enabled = true;
        }

        private void wylegende_Click(object sender, EventArgs e)
        {
            Wykres.Legends[0].Enabled = false;

               }
```

Fragment kodu 8. Metody od ukrywania i pokazywania legendy.

Oba przypadki działaja w taki sam sposób czyli w momencie kliknięcia w przycisk „Włącz legendę” i „Wyłącz legendę” wartosc widocznosci legendy zostanie zmieniona.

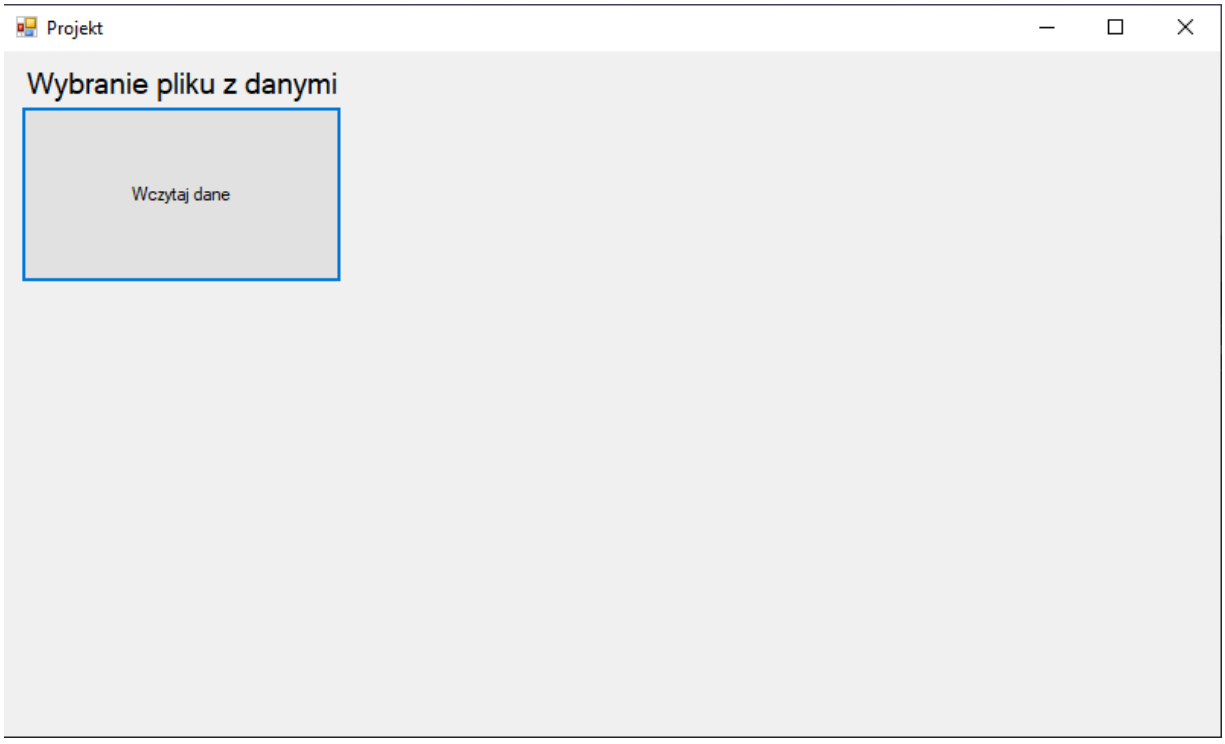
```
private void PokazDane_Click(object sender, EventArgs e)
        {
            for(int i = 0; i< wybrane.Count;i++)
            {
```

```
Wykres.Series[0].Points[wybrane[i]].SetValueY(Form1.zliczone[wybrane[i]]);
            }
            Wykres.Invalidate();
        }
        List<int> wybrane = new List<int>();

        private void ListaDoUkrycia_SelectedIndexChanged(object sender, EventArgs e)
        {
            wybrane.Add(ListaDoUkrycia.SelectedIndex);
            Wykres.Series[0].Points[ListaDoUkrycia.SelectedIndex].SetValueY(0);
            Wykres.Invalidate();

               }
```

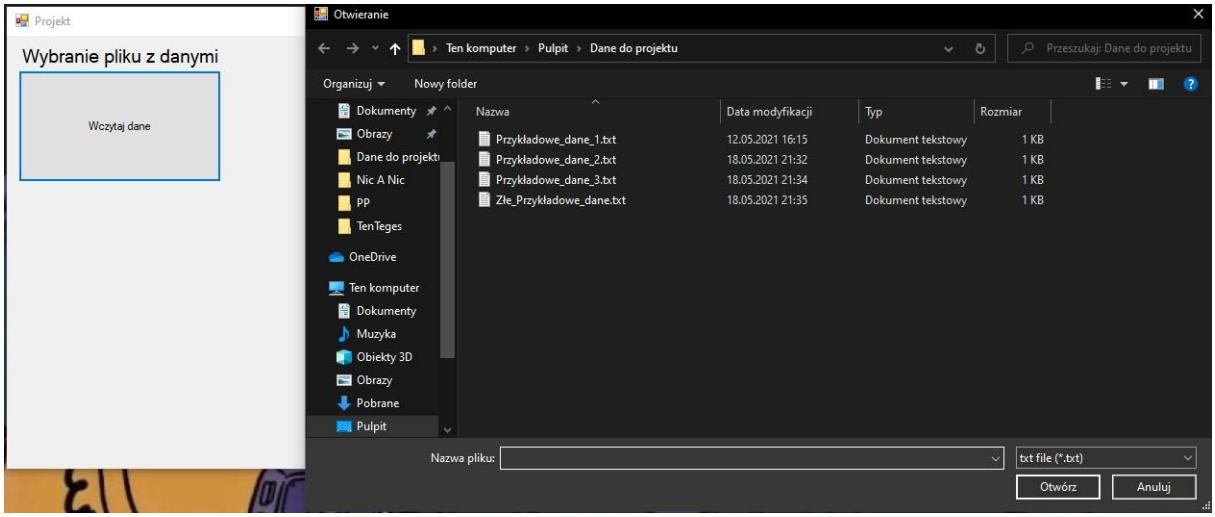
Fragment kodu 9. Ukrywanie elementów oraz ich przywracanie.

Metoda pierwsza zostaje wywołana w momencie kliknięcia w przycisk „Pokaż dane” i powoduje ustawienie podstawowych wartosci na wykresie. Następnie mamy deklaracje listy która będzie używana do zapisywania jaki element został usuniety aby łatwiej później przywracać te elementy. Kolejna metoda wywoływana jest w momencie wybrania jakiegoś elementu z listy rozwijanej i powoduje ona ustawienie wartosci wybranego punktu na 0 oraz dodanie tego punktu do listy.

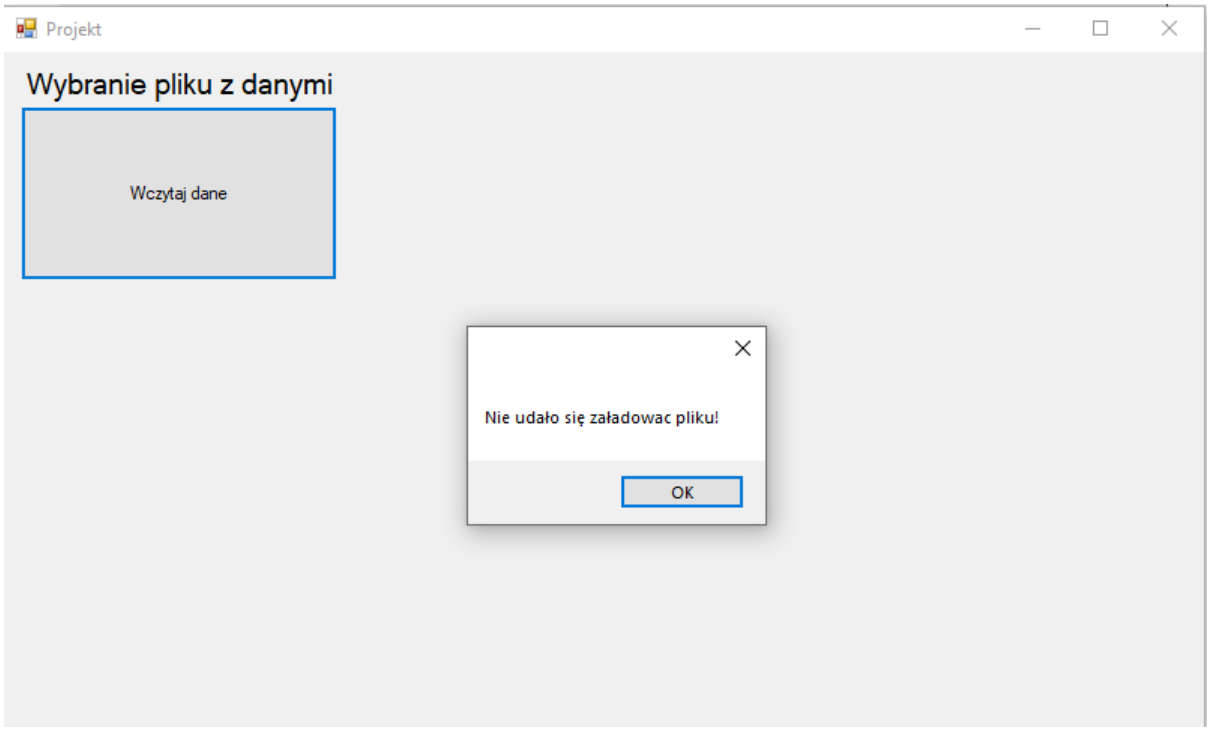
Rezultat:



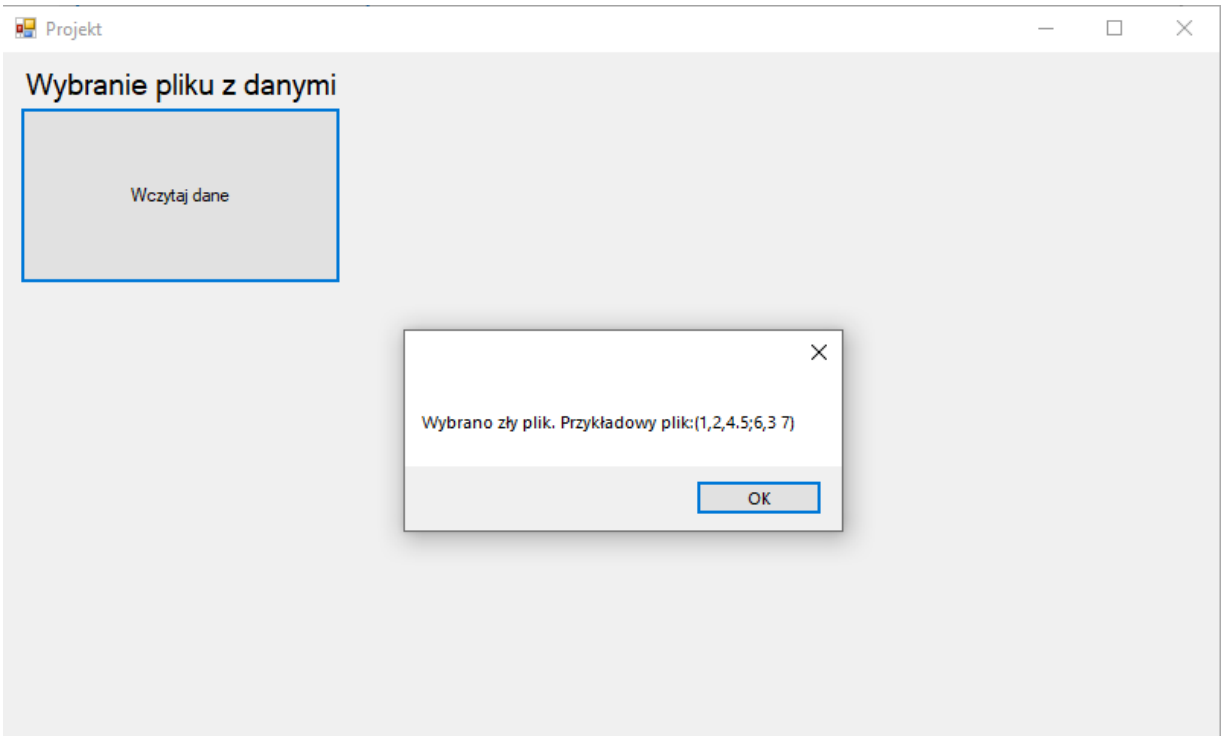
Zdj 1. Program po uruchomieniu.



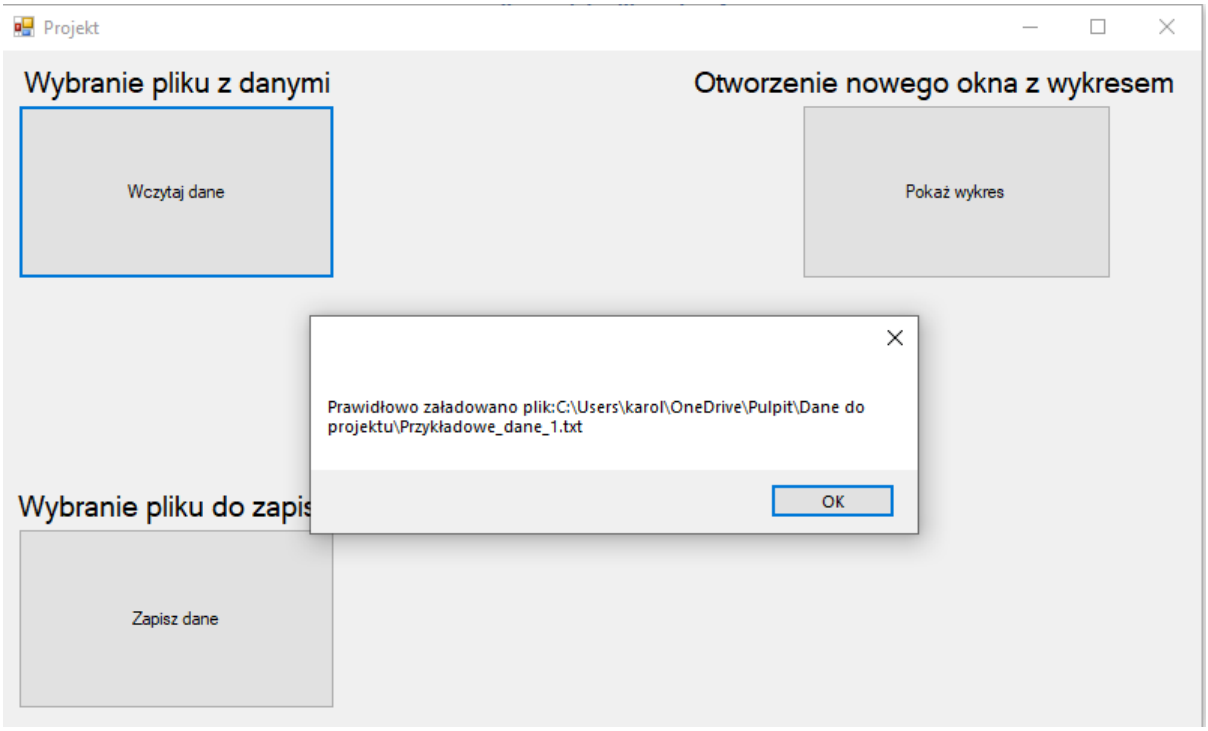
Zdj 2. Wybór pliku.



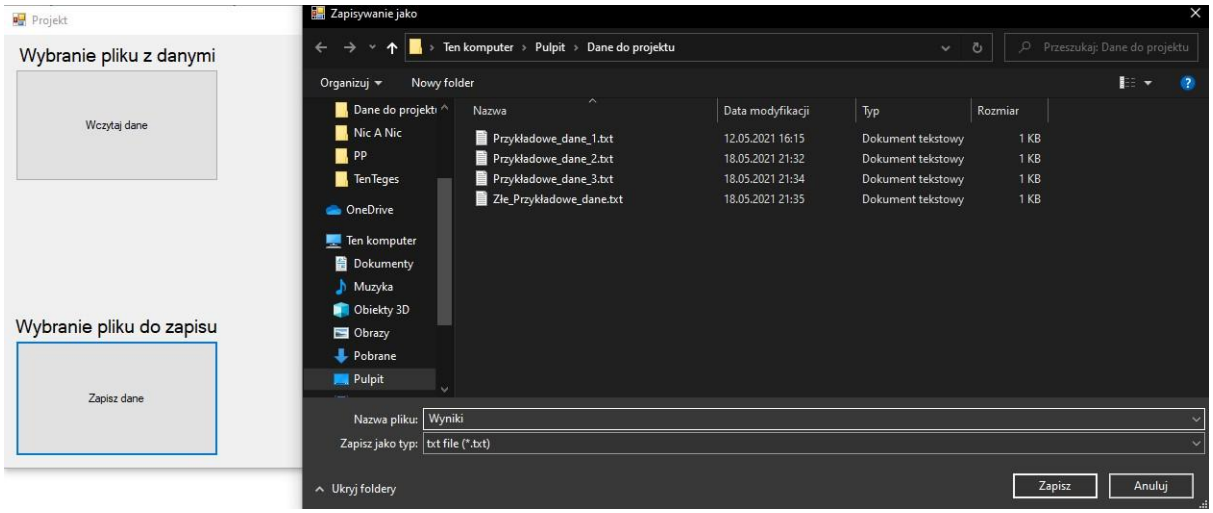
Zdj 3. Wybór pliku z złym formatem.



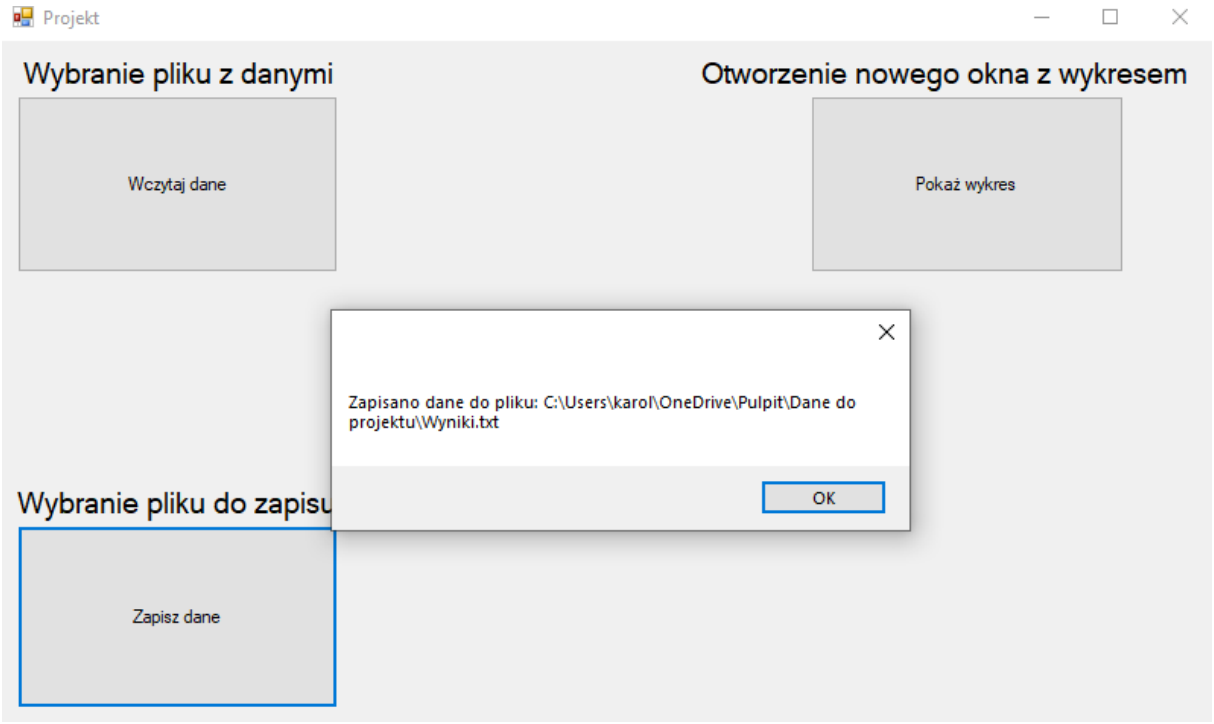
Zdj 4. Komunikat z poprawnym formatem pliku.



Zdj 5. Wybór pliku z poprawnym formatem.

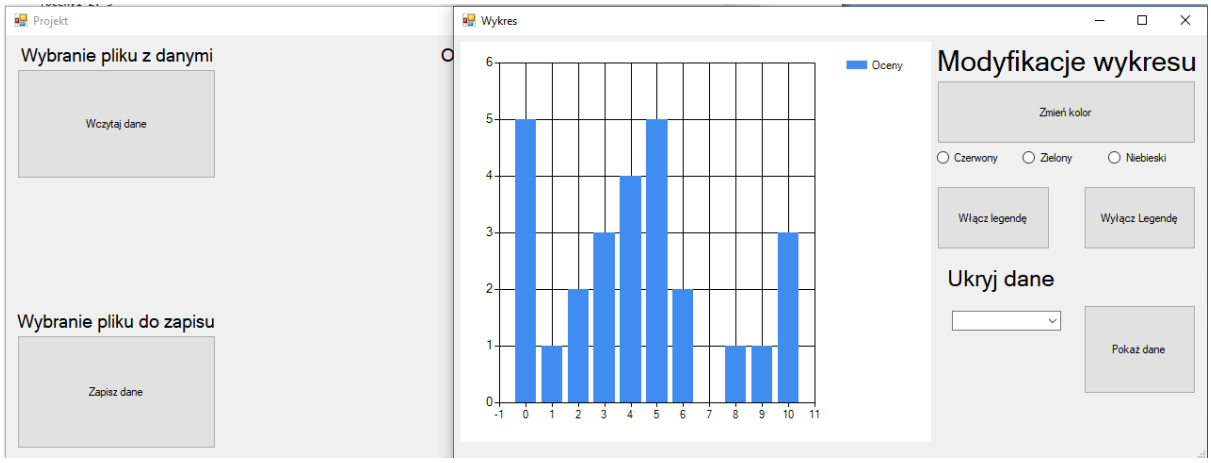


Zdj 6. Wybór miejsca oraz nazwy do zapisu.

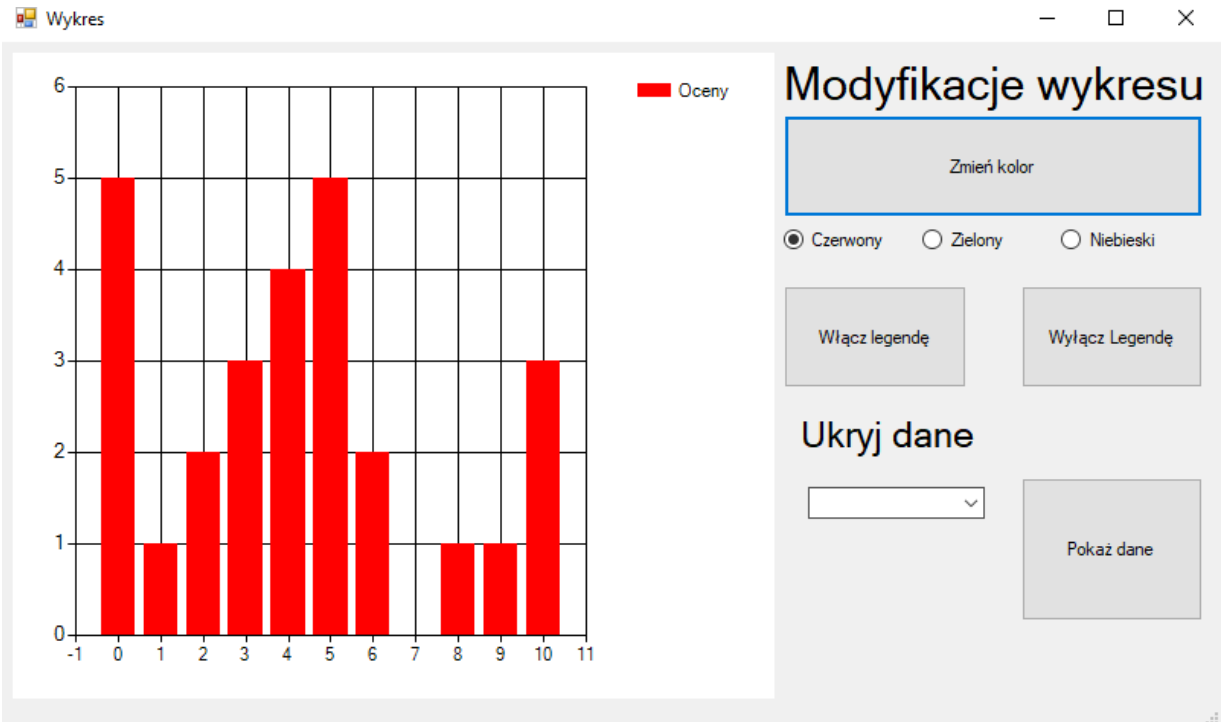


Zdj 7. Komunikat o porawnym zapisie pliku.

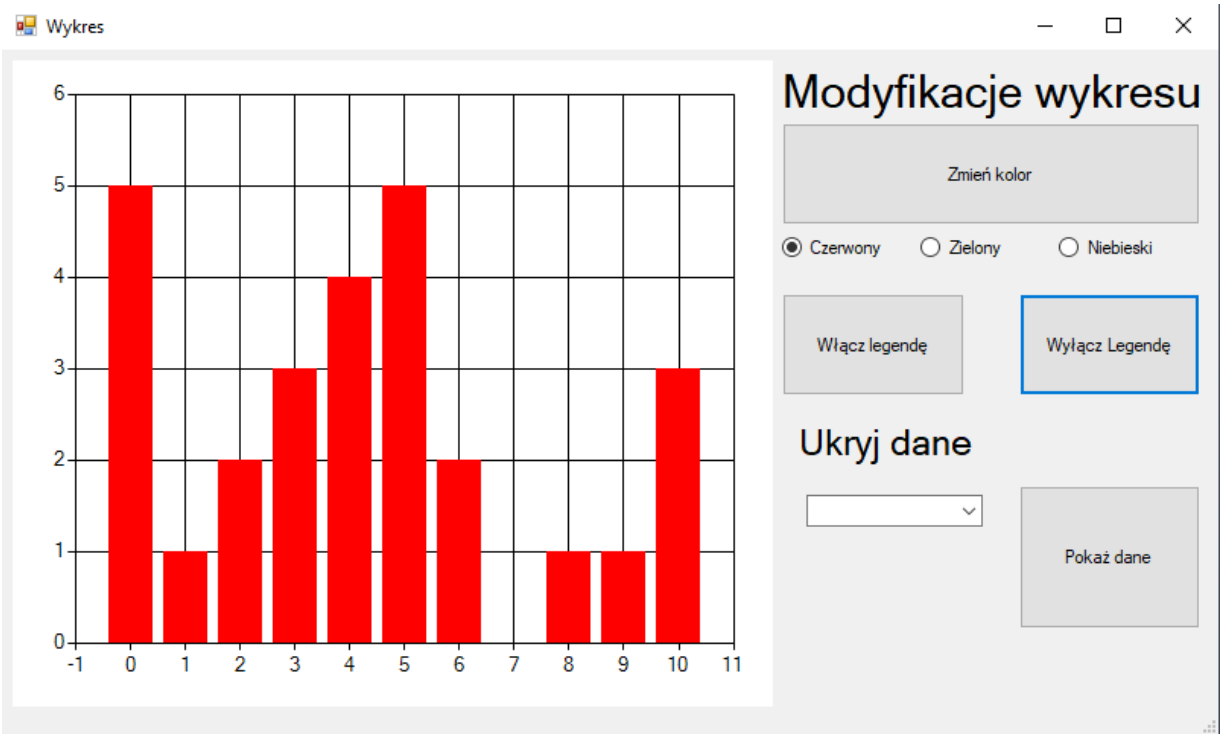
Zdj 8. Zawartość zapisanego pliku.



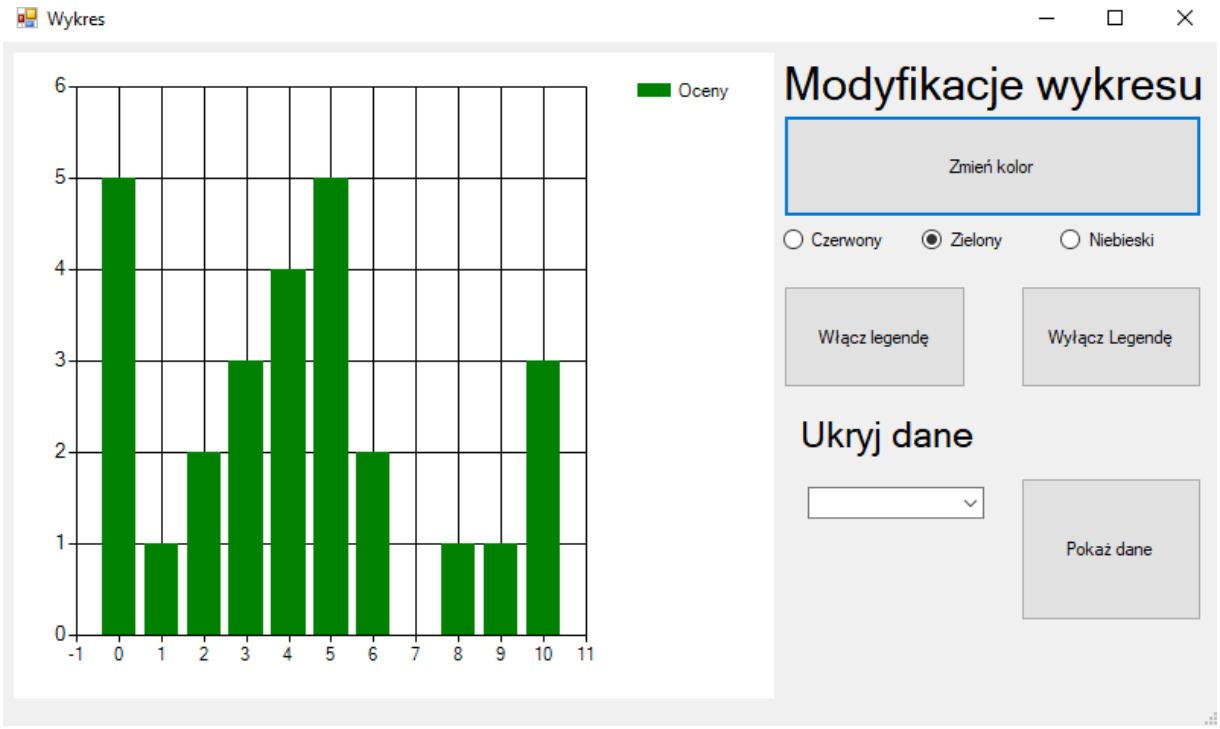
Zdj 9. Otworzenie nowego okna z wykesem.



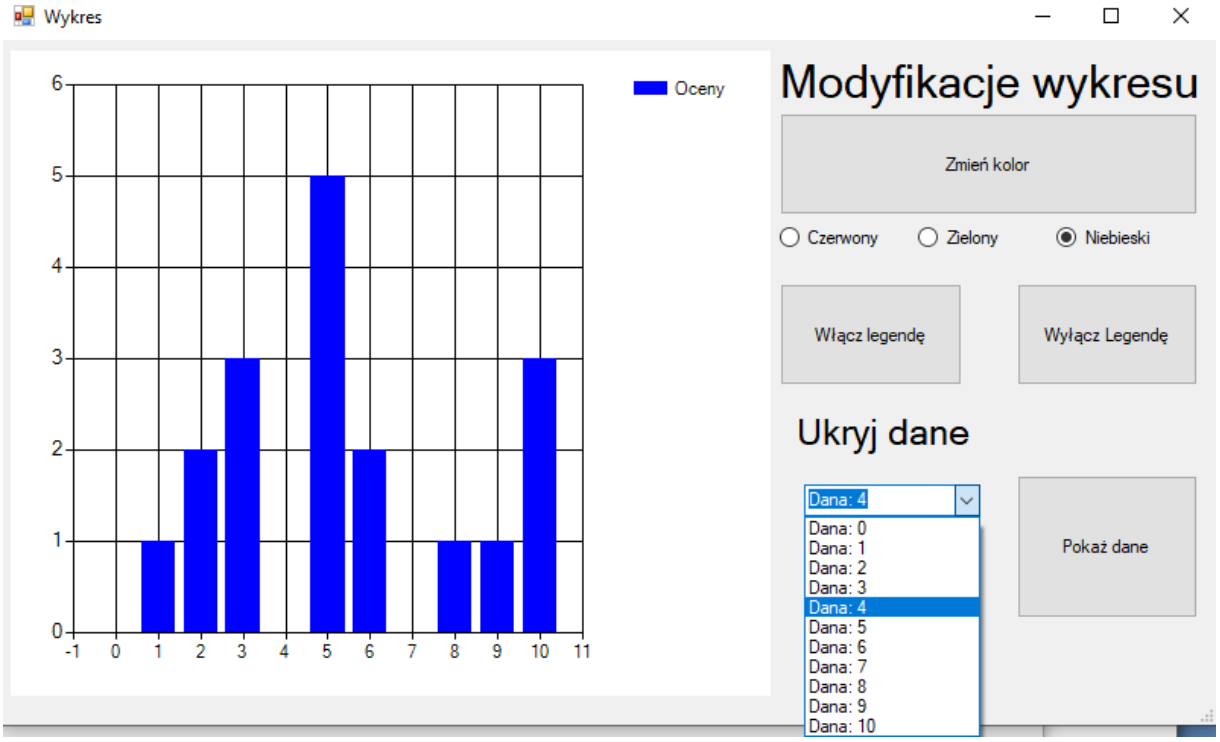
Zdj 10. Zmiana koloru danych na wykresie.



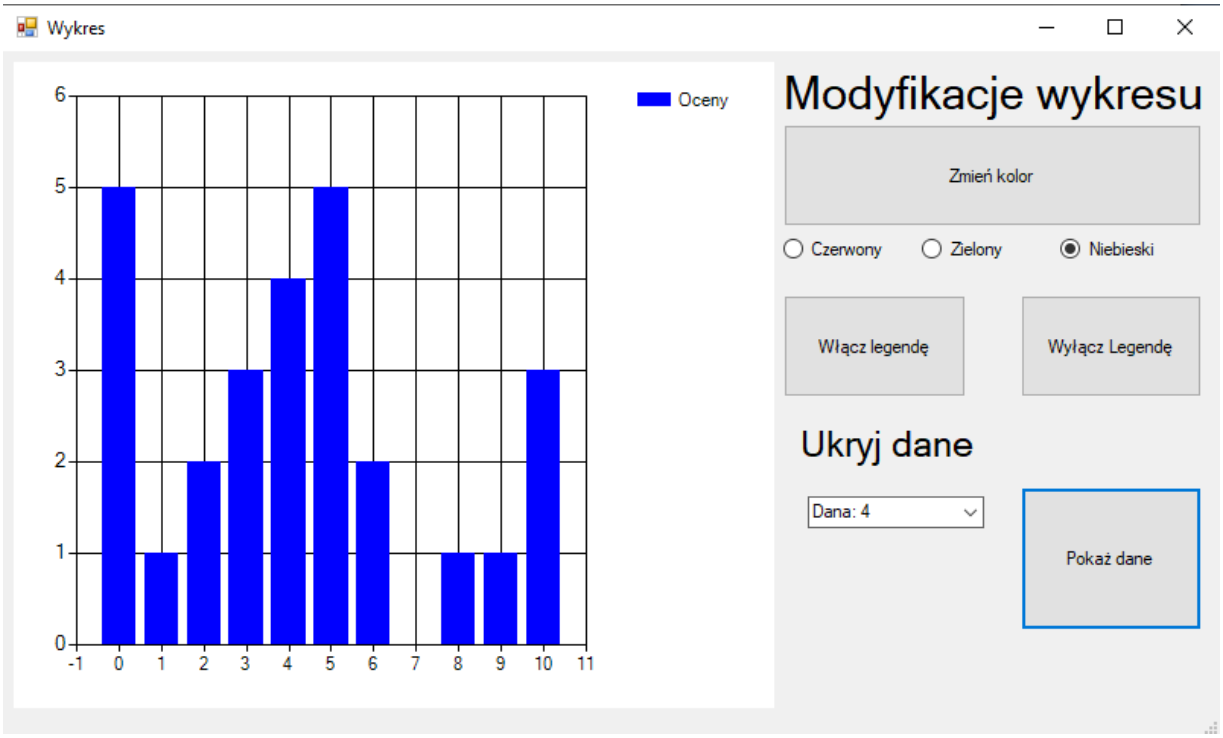
Zdj 11. Wyłączenie legendy.



Zdj 12. Ponowne włączenie legendy oraz zmiana koloru.



Zdj 13. Zmiana koloru oraz ukrycie danych.



Zdj 14. Przywrócenie danych na wykresie.

3.Podsumowanie

Program powal nam załadować plik z ocenami, który posiada odpowiedni format, o którym poinformuje nas sam program w momencie wybrania złego pliku. Realizacja wczytania pliku wykonywana jest w momencie wciśnięcia przycisku „Wczytaj dane”. Kolejnym zadaniem jest wyliczenie ilości danych ocen oraz wygenerowanie na podstawie nich wykresu, który możemy otworzyć w osobnym oknie po przez wybranie przycisku „Pokaż wykres”. Następnie mamy możliwość na proste modyfikacje wykresu dzięki panelowi po prawej stronie okna z wykresem gdzie mamy takie funkcje jak: zmianę koloru wykresu, ukrywanie legendy oraz danych. Program pozwala również zapisać obliczone przedziały ocen w pliku tekstowym. Dodatkowo w momencie wczytania błędnego pliku stracimy możliwość generowania wykresu oraz zapisu danych do pliku.

Liteatura:

1. Klasa Chart:
https://docs.microsoft.com/pl-pl/dotnet/api/system.windows.forms.datavisualization.charting.chart?view=netframework-4.8
2. Klasa OpenFileDialog:
https://docs.microsoft.com/pl-pl/dotnet/api/system.windows.forms.openfiledialog?view=net-5.0
3. Klasa SaveFileDialog:
https://docs.microsoft.com/pl-pl/dotnet/api/system.windows.forms.savefiledialog?view=net-5.0
4. MessageBox:
https://docs.microsoft.com/pl-pl/dotnet/api/system.windows.forms.messagebox?view=net-5.0
5. Klasa StreamWriter
https://docs.microsoft.com/pl-pl/dotnet/api/system.io.streamwriter?view=net-5.0
6. Obsługa try-catch
https://docs.microsoft.com/pl-pl/dotnet/csharp/language-reference/keywords/try-catch
7. Klasa list
https://docs.microsoft.com/pl-pl/dotnet/api/system.collections.generic.list-1?view=net-5.0
8. Poradnik do GUI C#
https://www.youtube.com/watch?v=U2f8A4C0rrQ
9. Klasa Color
https://docs.microsoft.com/pl-pl/dotnet/api/system.drawing.color?view=net-5.0
10. Poradnik do GUI c#
https://docs.microsoft.com/pl-pl/visualstudio/get-started/csharp/tutorial-wpf?view=vs-201