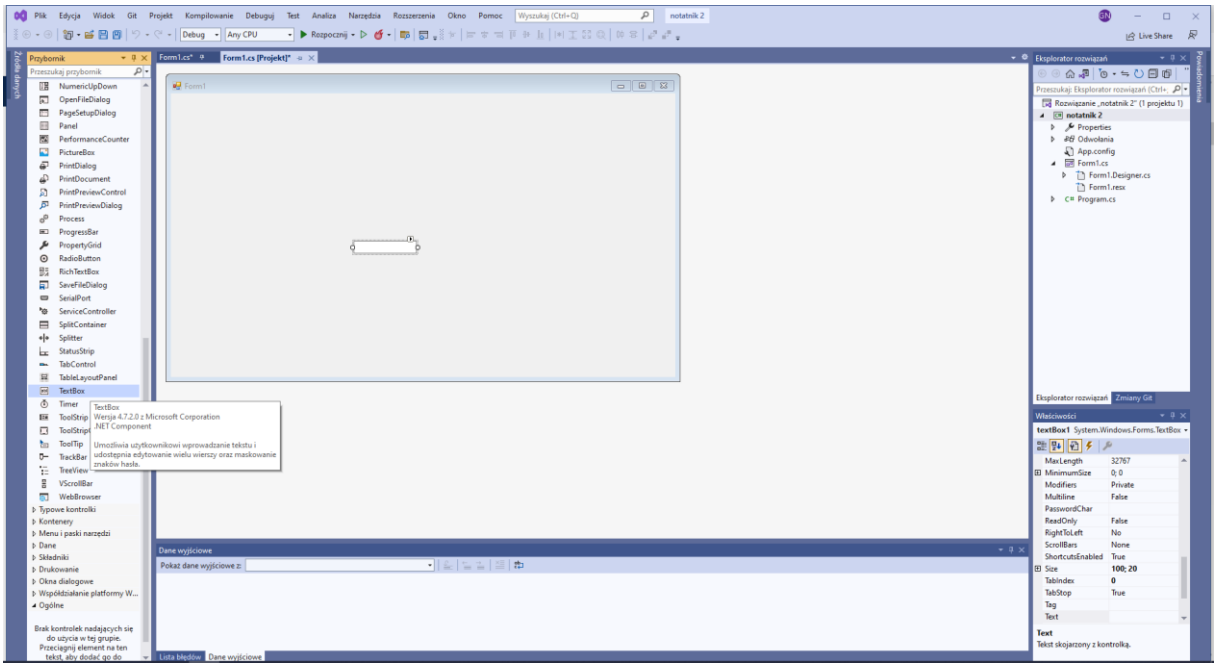
Celem jest stworzenie prostego edytora tekstowego - odpowiednika systemowego notatnika.

W środowisku Visual Studio zaznacz pozycję Aplikacja Windows Forms.

W polu Nazwa wpisz nazwę aplikacji Notatnik. Wybierz odpowiednią lokalizację i kliknij *OK*.

W widoku projektowania (zakładka *Form1.cs*), w palecie komponentów, odnajdź komponent TextBox i umieść go na powierzchni formy.



Zaznacz dodany do formy komponent i w oknie Właściwości zmień własność Multiline na True (w obiekcie będzie można wpisywać więcej niż jedną linię).

W grupie *Układ* odnajdź własność Dock.

Z własnością tą związane jest proste okno dialogowe, które pozwala na wybór położenia komponentu na formie.

Wybierz Fill — pole tekstowe wypełni całą dostępną przestrzeń okna.

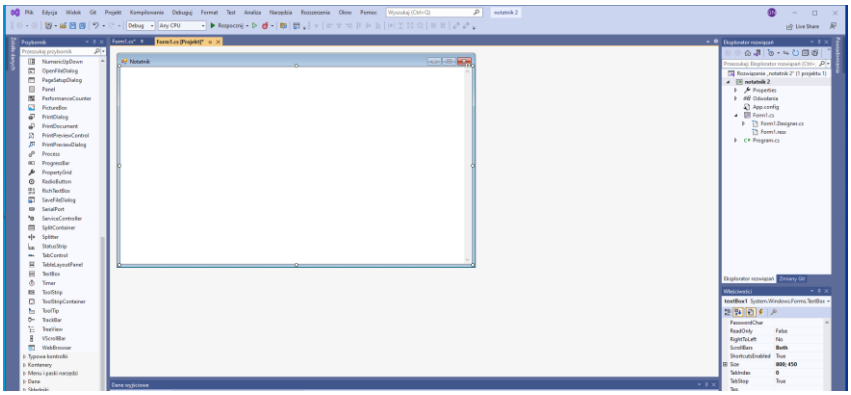
Dzięki ustawieniu własności Dock nie musimy się martwić o zachowanie pola tekstowego w razie zmiany rozmiaru formy przez użytkownika — komponent powinien zmieniać swój rozmiar tak, żeby dopasować go do nowego rozmiaru okna.

Zmień właściwość ScrollBars na Both – dostępne będą paski przewijania.

W widoku projektowania zaznacz klasę okna Form1.

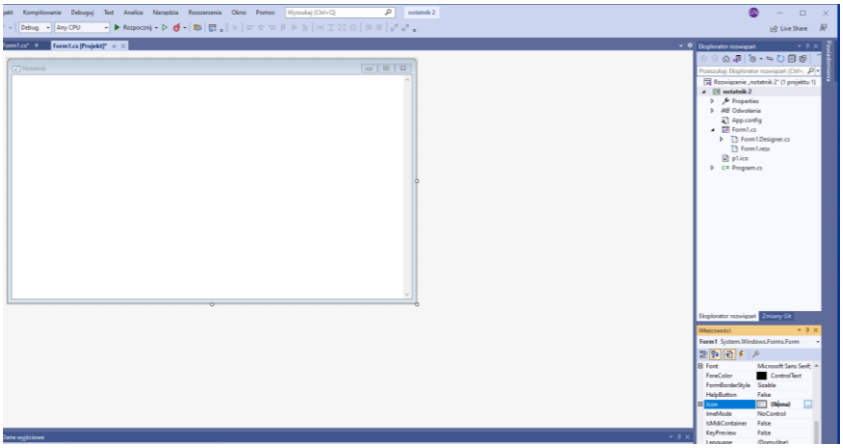
Zmień właściwość Text formy i w związanym z nią polu edycyjnym wpisz Notatnik.

Nazwa okna pozwala na identyfikację aplikacji. Warto zatem umieścić na pasku tytułu nazwę naszej aplikacji, tj. Notatnik.



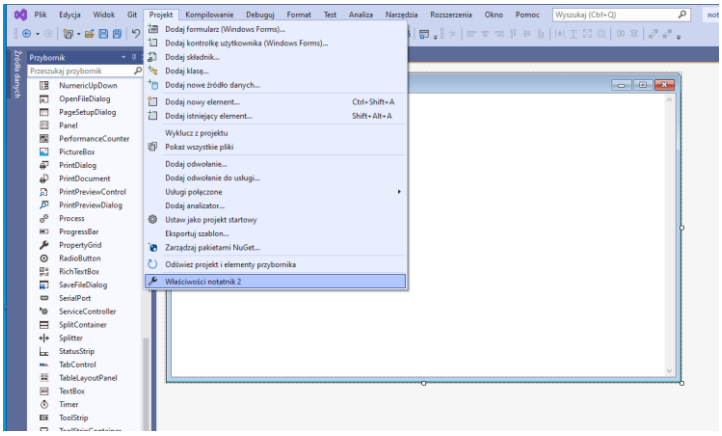
Ustawienie ikony formy i projektu.

Kliknij przycisk z trzema kropkami na Właściwości Icon Formy.



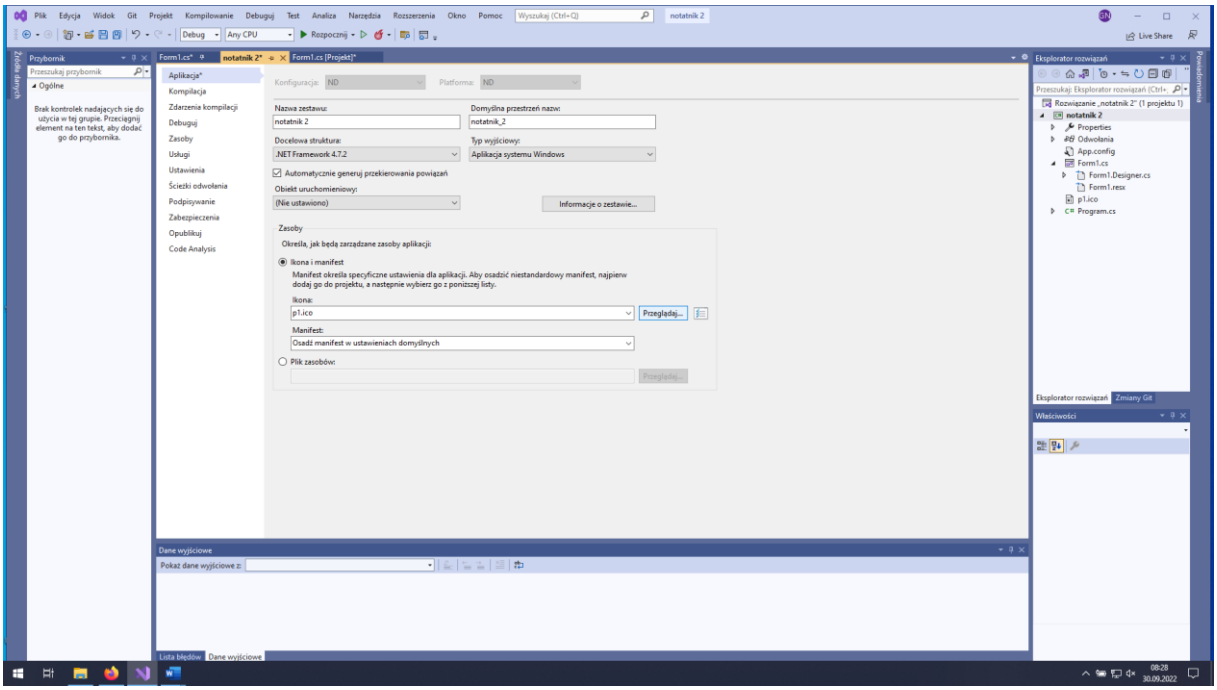
Otworzy się okno dialogowe, w którym można wskazać plik o rozszerzeniu *.ico*. Zostanie z niego pobrana ikona do wyświetlana na pasku tytułu formy. Po kliknięciu *Otwórz* wybierz jakiś plik *.ico* i zamknij okno dialogowe.

Z menu *Projekt* wybierz pozycję Właściwości *Notatnik*.



Pojawi się zakładka *Notatnik*, na której kliknij przycisk „Przeglądaj” przy polu *Ikona*.

Wybierz plik zawierający ikonę aplikacji. Zostanie ona włączona włączoną do skompilowanego pliku *.exe*.



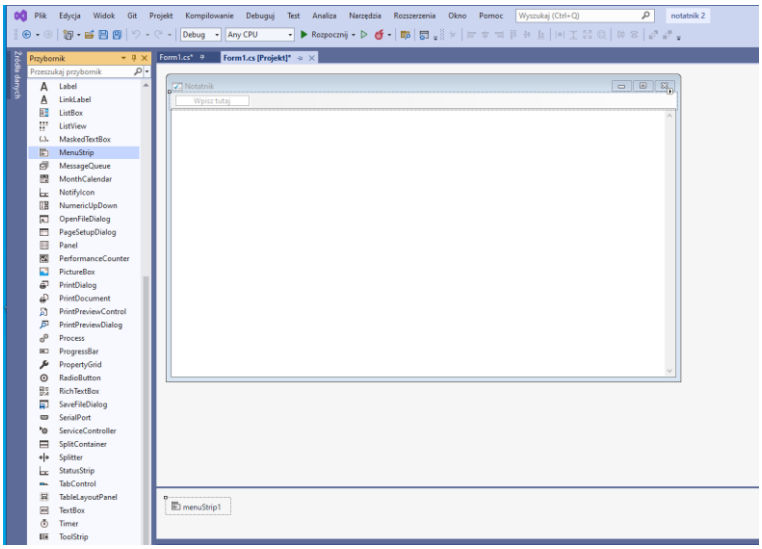
Zamknij zakładkę z własnościami projektu.

W obu przypadkach ikona zostanie umieszczona w zasobach aplikacji, a więc oryginalny plik *.ico* nie będzie więcej potrzebny. Nie trzeba go dołączać do gotowej aplikacji.

Menu główne aplikacji

Kolejnym typowym elementem, który dodajemy do aplikacji, jest menu główne. Umieśćmy je na formie i wyposażmy w pozycje, jakie znajdują się w standardowym notatniku systemu Windows.

W widoku projektowania, zaznacz komponent MenuStrip i umieść go na formie. Komponent znajdzie się na dodatkowym pasku pod podglądem formy. Ten komponent ma jednak także reprezentację na podglądzie formy — jest to zarazem edytor menu.



Jeżeli komponent menuStrip1 jest zaznaczony, w podglądzie okna widoczny jest edytor menu. Miejsce, w którym można wpisać nazwy podmenu, jest oznaczone napisem *Wpisz tutaj*.

Wprowadźmy nazwę pierwszego podmenu: &Plik. Po rozpoczęciu pisania natychmiast pojawią się dodatkowe miejsca pozwalające na utworzenie kolejnego podmenu oraz pierwszej pozycji w podmenu *Plik*.

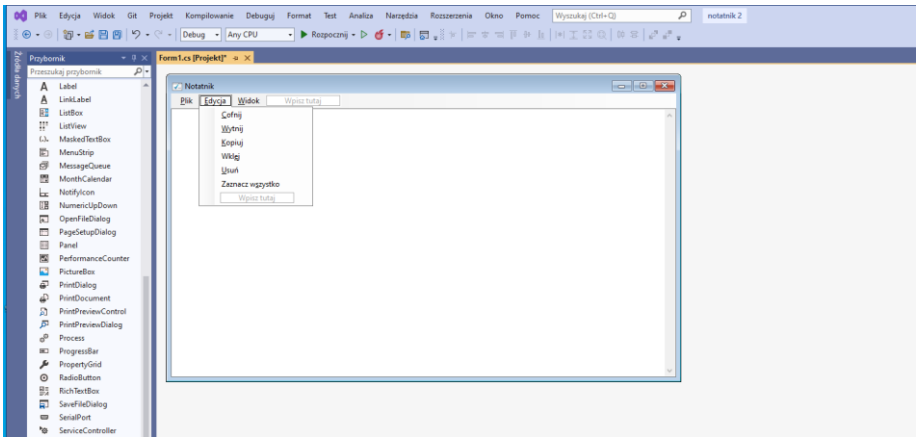
Umieszczony w nazwie pozycji menu znak & wskazuje na aktywny klawisz (oznaczona w ten sposób litera będzie podkreślona po naciśnięciu lewego klawisza *Alt* i uaktywnieniu menu).

Klawisze aktywne pozwalają na nawigację w menu bez użycia myszy - wystarczy nacisnąć klawisz odpowiadający podkreślonej literze (P w przypadku Plik), żeby uruchomić związaną z nią metodę zdarzeniową.

W podmenu *Plik* umieść cztery pozycje: *&Otwórz…*, *&Zapisz jako…*, separator (należy wpisać pojedynczy znak myślnika) i *Zamknij*.

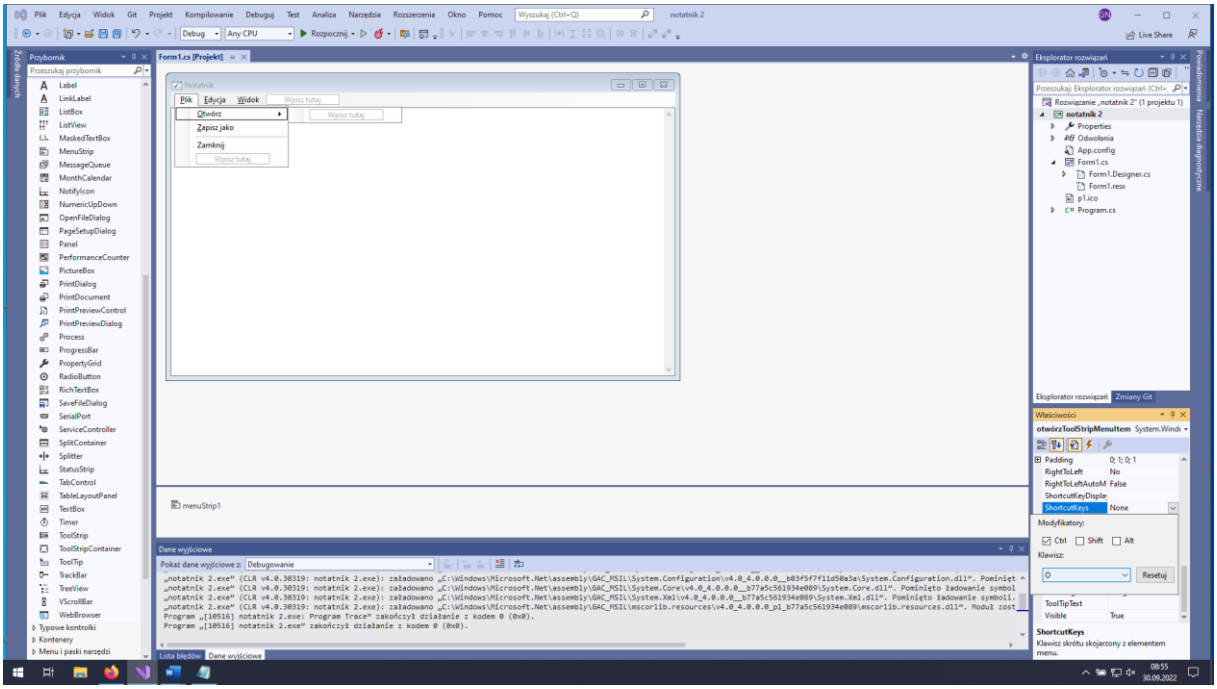
Następnie dodaj podmenu *&Edycja* z pozycjami *&Cofnij*, *&Wytnij*, *&Kopiuj*, *Wkl&ej*, *&Usuń* oraz *Zaznacz w&szystko*.

Dodaj podmenu *&Widok*, w którym umieść pozycje *&Czcionka…*, *&Tło…* i *Pasek &stanu*.



Klawisze skrótu.

Zaznacz w edytorze pozycję *Otwórz…* i w oknie własności odnajdź ShortcutKeys. Po uruchomieniu edytora tej własności zaznacz w nim opcję *Ctrl*, a z rozwijanego menu wybierz *O*. Analogicznie powiąż z pozycją *Zapisz jako…* kombinację klawiszy *Ctrl+S*.



Z pozycją *Zamknij* w menu *Plik* wiążemy metodę zdarzeniową zamykającą okno aplikacji. W widoku projektowania na podglądzie formy kliknij dwukrotnie pozycję menu *Zamknij*. W powstałej w ten sposób metodzie zdarzeniowej wpisz polecenie Close().

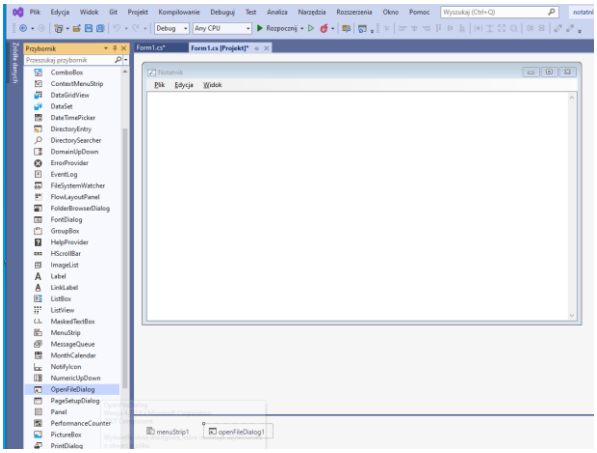
```
private void zamknijToolStripMenuItem_Click(object sender, EventArgs e)
{
   Close();
}
```

Polecenie Close(); umieszczone w metodzie zamknijToolStripMenuItem_Click jest wywołaniem metody formy, a więc metody instancji klasy Form1. Metoda ta została zdefiniowana w jej klasie bazowej, tj. w System.Windows.Forms.Form. Skutkiem jej działania jest zamknięcie okna aplikacji. Zgodnie z filozofią systemu Windows zamknięcie głównego okna (w tym przypadku jedynego) oznacza także zamknięcie samej aplikacji.

Odczytywanie tekstu z pliku

Przygotujemy teraz metodę pozwalającą na wybór pliku za pomocą standardowego okna dialogowego, a następnie wczytującą zawartość wybranego pliku do pola tekstowego.

Zaznaczmy komponent *OpenFileDialog* i umieśćmy go na podglądzie formy. Nowy obiekt openFileDialog1 nie pojawi się na samej formie, ale pod nią, na liście komponentów „niewizualnych", razem z menuStrip1.



Klasy FileSystemInfo, DirectoryInfo i FileInfo pozwalają na wykonywanie na plikach i katalogach podstawowych operacji, takich jak ich tworzenie i usuwanie, operacje na nazwach czy pobieranie parametrów, jak np. czas utworzenia bądź modyfikacji. Obejmuje ona właściwości i metody wspólne dla plików i katalogów.
Wymienione klasy znajdują się w przestrzeni nazw System.IO, tak więc w programach będzie stosowana dyrektywa using w postaci:

```
using System.IO;
```

Dwukrotnie kliknij pozycję menu *Otwórz…* w podglądzie menu, tworząc w ten sposób metodę zdarzeniową uruchamianą w przypadku wybrania tej pozycji menu w działającej aplikacji. Do nowej metody wpisz kod:

```
if (openFileDialog1.ShowDialog() == DialogResult.OK)
{
    string nazwaPliku = openFileDialog1.FileName;
    StreamReader sr = new StreamReader(nazwaPliku);
    textBox1.Text = sr.ReadToEnd();
}
```

Do zmiennej `nazwaPliku` przypisujemy nazwę pliku i używamy jako argumentu konstruktora klasy StreamReader. Utworzony obiekt jest przypisywany zmiennej sr. Jeśli utworzenie obiektu typu StreamReader się powiedzie, jest on używany do odczytu danych. Wykorzystana została w tym celu metoda `ReadToEnd` odczytująca ze strumienia wszystkie dane. Odczytane dane są przypisane właściwości Text komponentu textBox.

Uruchom program i wczytaj plik tekstowy.
Uzupełnijmy ostatni kod o kontrolę błędów

```
try
{
   if (openFileDialog1.ShowDialog() == DialogResult.OK)
   {
      string nazwaPliku = openFileDialog1.FileName;
      StreamReader sr = new StreamReader(nazwaPliku);
      textBox1.Text = sr.ReadToEnd();
   }
}
catch (Exception info)
{
   MessageBox.Show("Błąd odczytu pliku " + info.Message);
}
```

Ustawmy opcje okna dialogowego. Zaznaczmy komponent okna dialogowego openFileDialog1 i korzystając z okna właściwości:
- zmieńmy tytuł okna, ustawiając własność Title na *Wczytaj plik*;
- własność DefaultExt zmieńmy na *txt* (jest to rozszerzenie dopisywane do nazwy pliku, jeżeli użytkownik żadnego rozszerzenia nie doda sam);
- wyczyśćmy zawartość pola przy własności FileName;
- edytujmy pole przy własności InitialDirectory, podając katalog domyślny;
- wpiszmy łańcuch określający filtry (własność Filter), np.: Pliki tekstowe (*.txt)|*.txt|Pliki ini (*.ini)|*.ini|Pliki źródłowe (*.cs)|*.cs|Wszystkie pliki (*.*)|*.*

Własność Filter jest zwykłym łańcuchem, czyli zmienną typu string. Fragment tego łańcucha definiujący jeden filtr składa się z dwóch segmentów oddzielonych znakiem |. W pierwszym określamy opis wyświetlany w oknie dialogowym, a w drugim — maskę filtru. Kolejne filtry również oddzielane są znakiem |

Jeżeli z jakiś względów interesowałby nas podział na akapity (wiersze) wczytywanego tekstu, to możemy skorzystać z Listy.
Do gromadzenia i przechowywania tekstu wewnątrz metody skorzystamy z typu ogólnego List sparametryzowanego typem string. Każdy łańcuch będzie przechowywał osobny wiersz (akapit) odczytanego z pliku tekstu. W przypadku powodzenia (tj. braku wyjątków) metoda konwertuje ową listę na tablicę łańcuchów. Dzięki temu wartość zwracaną przez metodę będzie można wprost przypisać do własności Lines komponentu TextBox, która jest właśnie tablicą łańcuchów.

Zmodyfikowany kod

```
List<string> tekst = new List<string>();
try
{
    if (openFileDialog1.ShowDialog() == DialogResult.OK)
    {
        string nazwaPliku = openFileDialog1.FileName;
        using (StreamReader sr = new StreamReader(nazwaPliku))
        {
            string wiersz;
            while ((wiersz = sr.ReadLine()) != null)
                tekst.Add(wiersz);
        }

    }
    textBox1.Lines = tekst.ToArray();
}
catch (Exception info)
{
    MessageBox.Show("Błąd odczytu pliku " + info.Message);
}
```

Zapisywanie tekstu do pliku

Przygotujmy teraz metodę pomocniczą ZapiszDoPlikuTekstowego zapisującą wskazaną w argumencie tablicę łańcuchów do pliku tekstowego. Następnie w metodzie zdarzeniowej związanej z pozycją *Zapisz jako...* z menu *Plik* wykorzystamy tę metodę do zapisania do pliku zawartości komponentu TextBox.

W obrębie klasy Form1 zdefiniuj metodę

```
public static void ZapiszDoPlikuTekstowego(string nazwaPliku, string[] tekst)
{
   using (StreamWriter sw = new StreamWriter(nazwaPliku))
   foreach (string wiersz in tekst)
        sw.WriteLine(wiersz);
}
```

W widoku projektowania umieść na formie komponent SaveFileDialog z zakładki *Dialogs*. Zaznacz go i w oknie właściwości:

- zmień własność Title na *Zapisz do pliku*;
- DefaultExt na *txt*;
- we własności zdefiniuj filtr: Pliki tekstowe (*.txt)|*.txt|Pliki ini (*.ini)|*.ini|Pliki źródłowe (*.cs)|*.cs| Wszystkie pliki (*.*)|*.*;
- zmień InitialDirectory na kropkę.

Utwórz metodę zdarzeniową do pozycji *Zapisz jako…* w menu *Plik* i umieść w niej kod

```
{
   string nazwaPliku = openFileDialog1.FileName;
   if (nazwaPliku.Length > 0) saveFileDialog1.FileName = nazwaPliku;
   if (saveFileDialog1.ShowDialog() == DialogResult.OK)
   {
      nazwaPliku = saveFileDialog1.FileName;
      ZapiszDoPlikuTekstowego(nazwaPliku, textBox1.Lines);
   }
}
```

Na początku metody podejmowana jest próba pobrania z własności FileName obiektu openFileDialog ścieżki dostępu do pliku, z którego ewentualnie wcześniej wczytany był tekst. Odczytanie łańcucha z tej własności może zwrócić łańcuch pusty jedynie wtedy, gdy wcześniej nie został wczytany żaden plik. Odczytana ścieżka pliku przypisywana jest własności saveFileDialog1.FileName.

Okna dialogowe wyboru czcionki i koloru

Przygotujemy teraz metody związane z pozycjami menu *Widok*, które będą pozwalać na wybór koloru tła notatnika i użytej w nim czcionki. Użyjemy standardowych okien dialogowych.

Na podglądzie formy umieśćmy okno dialogowe ColorDialog. Dla pozycji Tło w menu Widok utwórzmy metodę zdarzeniową:

```
colorDialog1.Color = textBox1.BackColor;
if (colorDialog1.ShowDialog() == DialogResult.OK)
    textBox1.BackColor = colorDialog1.Color;
```

Na podglądzie formy umieśćmy okno dialogowe FontDialog.
Ustawmy jego własność ShowColor na True. Dzięki temu będziemy mogli zmieniać także kolor czcionki.

Dla pozycji *Czcionka* w menu Widok utwórzmy metodę zdarzeniową:

```
fontDialog1.Font = textBox1.Font;
fontDialog1.Color = textBox1.ForeColor;
if (fontDialog1.ShowDialog() == DialogResult.OK)
{
    textBox1.Font = fontDialog1.Font;
    textBox1.ForeColor = fontDialog1.Color;
}
```

Edycja i korzystanie ze schowka

Kolejna grupa poleceń związana jest z menu *Edycja*.

Komponent TextBox posiada już odpowiednie metody.

Tworzymy metody zdarzeniowe do wszystkich pozycji z menu *Edycja* i odpowiednio wywołujemy w nich następujące metody

```
private void cofnijToolStripMenuItem_Click(object sender, EventArgs e)
 {
     textBox1.Undo();
 }

 private void wytnijToolStripMenuItem_Click(object sender, EventArgs e)
 {
     textBox1.Cut();
 }

 private void kopiujToolStripMenuItem_Click(object sender, EventArgs e)
 {
     textBox1.Copy();
 }

 private void wklejToolStripMenuItem_Click(object sender, EventArgs e)
 {
     textBox1.Paste();
 }

 private void usuńToolStripMenuItem_Click(object sender, EventArgs e)
 {
     textBox1.SelectedText = "";
 }
private void zaznaczWszystkoToolStripMenuItem_Click(object sender, EventArgs e)
 {
     textBox1.SelectAll();
 }
```