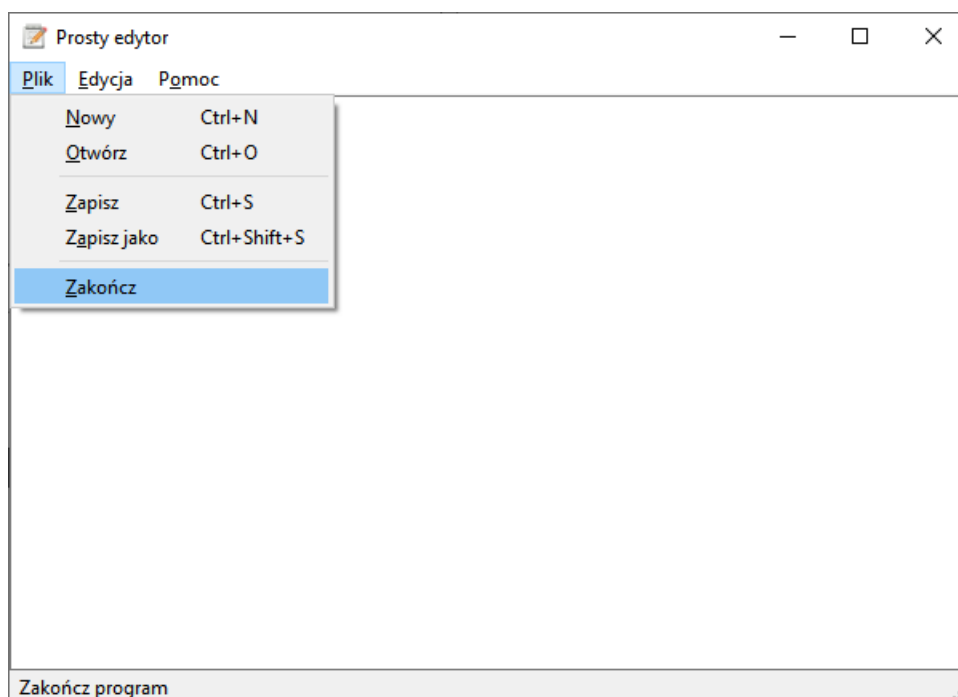


Edytor tekstu

1. Przykład

Opracuj aplikację **Edytor tekstu**.

- Do budowy interfejsu użytkownika użyj kontrolki:
 - kontrolki edycyjnej **QTextEdit**,
 - menu **QMenuBar** i pasek statusu **QStatusBar** (powinny znajdować się w domyślnym szablonie projektu).
- Ustaw układ okna na **Lay Out Horizontally**.
- Zawartość menu programu pokazana jest na kolejnych rysunkach.

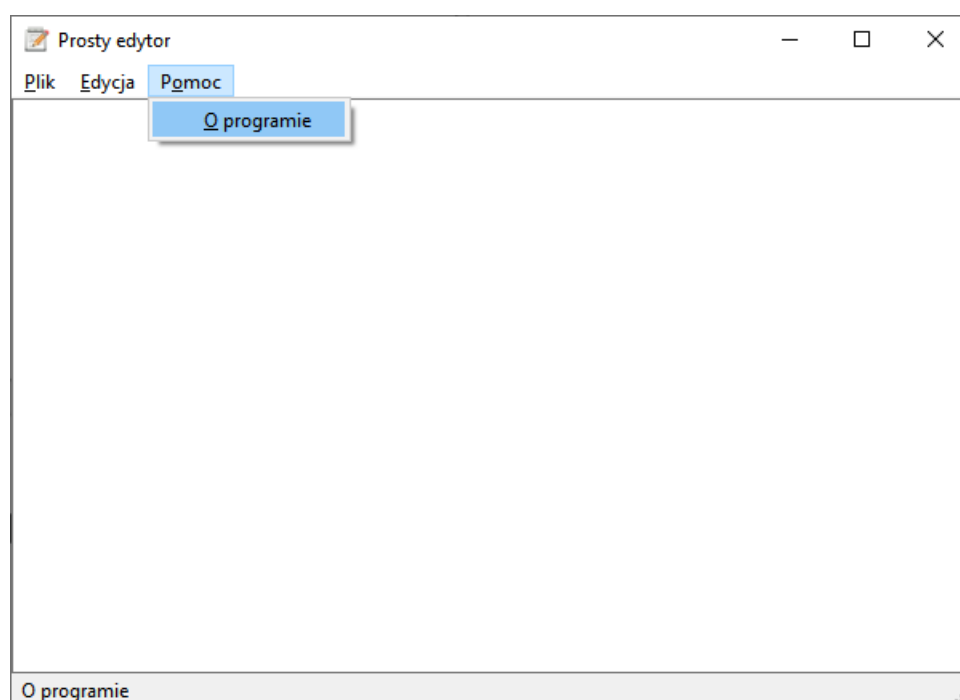
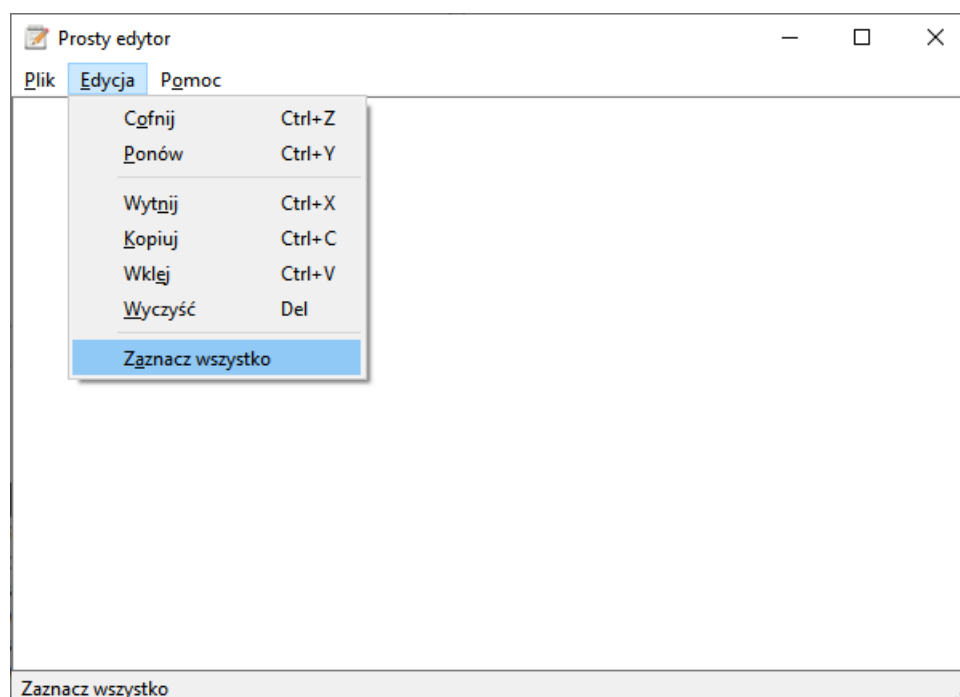


- Zwróć uwagę, że w nazwach opcji znajdują się podkreślone litery, oznacza to, że można się dostać do pozycji menu za pomocą skrótów klawiaturowych z klawiszem [ALT].
- Aby dodać taki skrót, użyj symbolu & (ampersand) przed literą w nazwie opcji.
- Dla wszystkich akcji dodaj stosowne podpowiedzi **ToolTip** (panel **Action Editor, PPM | Edit...**).
- W konstruktorze klasy okna zmień centralną kontrolkę na kontrolkę edycyjną, zwróć uwagę na zmiany w wyglądzie interfejsu programu (brak marginesów).

```
this->setCentralWidget(ui->textEdit);
```

- Dodaj prototypy metod obsługujących akcje.
- Metoda do zamykania aplikacji.

```
this->close();
```



- Metody obsługujące cofanie i ponawianie akcji.

```
| ui->textEdit->undo();
```

```
| ui->textEdit->redo();
```

- Dostępne są jeszcze metody **undoAvailable()** i **redoAvailable()**, które można wykorzystać do kontrolowania dostępności elementów menu programu.
- Metoda obsługująca kopiowanie.

```
ui->textEdit->cut();
```

- Metody dla reszty akcji edycyjnych opracuj samodzielnie, użyj odpowiednich metod klasy `QTextEdit` (`copy()`, `paste()`, `clear()`, `selectAll()`).
- Dodaj pliki nagłówkowe, potrzebne do odczytu i zapisu plików: `<QFile>`, `<QFileDialog>`, `<QTextStream>`.
- W definicji klasy okna w sekcji **private** dodaj zmienną `QString fileName`, a w konstruktorze klasy okna zainicjuj ją wartością pustą.

```
fileName = "";
```

- Propozycja metody obsługującej tworzenie nowego pliku.

```
fileName = "";  
ui->textEdit->clear();
```

- Propozycja metody obsługującej otwieranie pliku.

```
fileName = QFileDialog::getOpenFileName(this, "Otwórz plik");  
  
if(!(fileName.isEmpty() || fileName.isNull()))  
{  
    QFile textFile(fileName);  
  
    if(!textFile.open(QFile::ReadOnly | QFile::Text))  
    {  
        QMessageBox::critical(this, "Prosty edytor", "Błąd otwarcia pliku.");  
        fileName = "";  
        return;  
    }  
    QTextStream textStream(&textFile);  
    QString text = textStream.readAll();  
    textFile.close();  
  
    ui->textEdit->setText(text);  
}
```

- Propozycja metody obsługującej zapisywanie pliku z podaniem nazwy.

```
QString text = ui->textEdit->toPlainText();  
  
fileName = QFileDialog::getSaveFileName(this, "Zapisz plik");  
  
if(!(fileName.isEmpty() || fileName.isNull()))  
{  
    QFile textFile(fileName);  
  
    if(!textFile.open(QFile::WriteOnly | QFile::Text))  
    {  
        QMessageBox::critical(this, "Prosty edytor", "Błąd zapisu pliku.");  
        fileName = "";  
        return;  
    }  
  
    QTextStream textStream(&textFile);  
    textStream << text;
```

```

    textFile.flush();
    textFile.close();
}

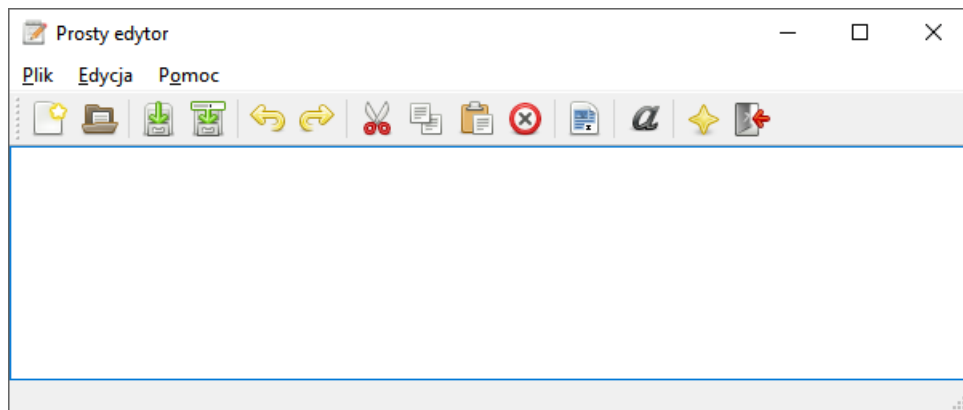
```

- Metodę zapisy pliku opracuj samodzielnie na podstawie poprzedniej metody.

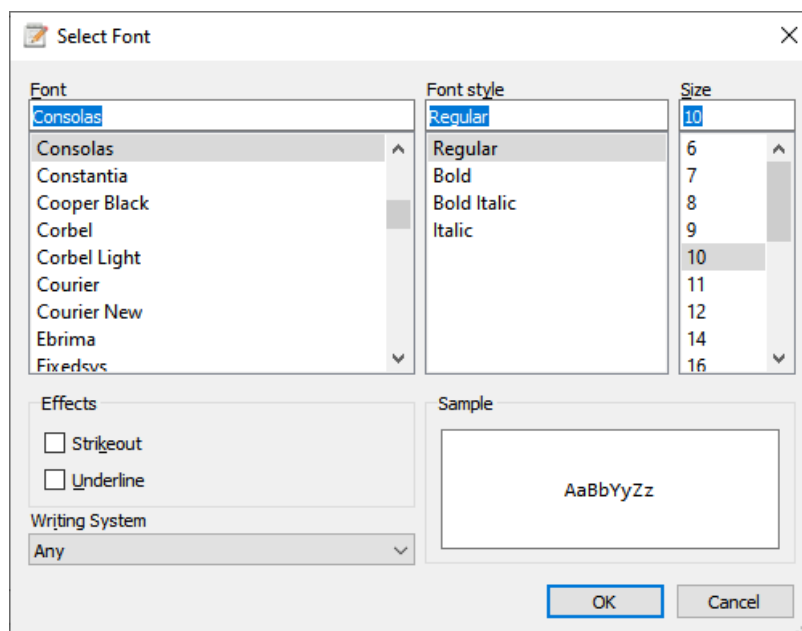
Zadania dodatkowe

1. Rozbuduj program z przykładu 1.

- Dodaj do projektu plik zasobów z ikonami (do pobrania ze strony przedmiotu).
- Zmień ikonę okna programu (właściwość **windowIcon**).
- Uzupełnij program o pasek narzędzi **QToolBar**.



- Dodaj skróty klawiaturowe widoczne w menu programu (panel **Action Editor**, **PPM | Edit...** albo właściwość **shortcut** po wybraniu akcji).
- Dodaj podpowiedzi wyświetlane na pasku statusu (właściwość **statusTip** po wybraniu akcji).
- Dodaj możliwość zmiany czcionki z wykorzystaniem klasy **QFontDialog**.



- Należy dodać pliki nagłówkowe **<QFont>** i **<QFontDialog>**.
- Propozycja metody obsługującej zmianę kroju czcionki.

```
bool fontOk;  
  
QFont newFont = QFontDialog::getFont(&fontOk, this);  
  
if(fontOk)  
{  
    ui->textEdit->setFont(newFont);  
}  
else  
    return;
```

2. Przerób program do tablicowania funkcji w taki sposób, żeby używał kontrolki **QTextEdit** do wyświetlania wyników i obsługiwał zapis do pliku typu **CSV**.