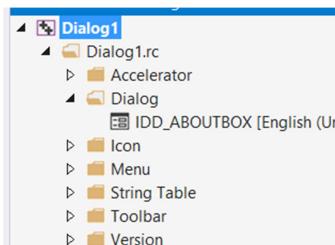


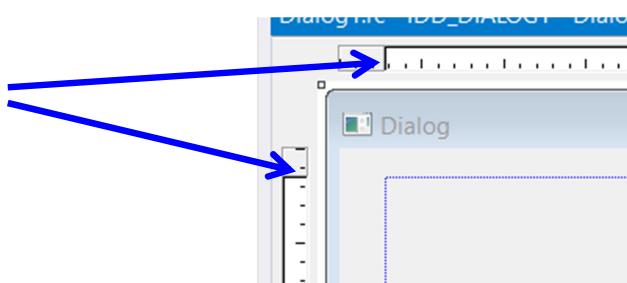
===== CZEŚĆ PIERWSZA =====

Robimy okno dialogowe modalne

- Wykreować nowy projekt MFC APP o nazwie **Dialog1**, Typ aplikacji SDI, Styl aplikacji: MFC Standard, wyłączyć Printing and print preview
- Otworzyć okno z zasobami oraz podgrupę Dialog

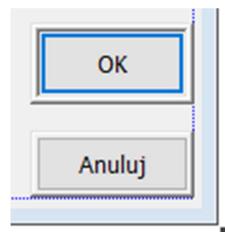


- Aplikacja już ma jedno okno dialogowe IDD_ABOUTBOX
- Dodać nowe okno dialogowe: prawy klawisz na grupie zasobów Dialog i z menu wybrać **Insert Dialog**. Nowe okno dialogowe ma identyfikator **IDD_DIALOG**. Proszę zmienić na **IDD_TEST** (prawy klawisz na identyfikatorze i wybrać **Properties**). Pokazało się jednocześnie okno dialogowe z przyciskami **OK** i **Cancel**. Jednocześnie widać jakie są marginesy (można je modyfikować – niebieskie strzałki wskazują gdzie złapać myszką), a przyciski można przesuwać. Jeżeli kontrolka w oknie dialogowym przylega do linii marginesu to gdy zwiększamy rozmiar okna dialogowego to kontrolka będzie również się przemieszczać (może być niekorzystne gdy już mamy ustawione i wyrównane kontrolki i potrzeba jeszcze trochę miejsca to się może rozjechać).

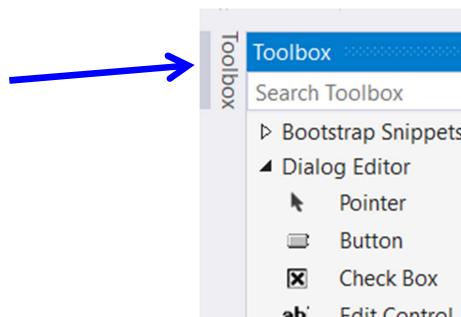


- Chwycić myszką za prawy dolny róg okna dialogowego i lekko zwiększyć
- Ustawić wszystkie marginesy na 3 (przsuwając myszką margines wyświetla się wartość)

- Zmienić właściwość **Caption** dla przycisku **ID_CANCEL** na **Anuluj**. Lekko zwiększyć wysokość i zmienić właściwości: **Modal Frame**, **Static Edge** na **TRUE**.
- Przycisk **OK** lekko zwiększyć (wysokość) i zmienić mu właściwości: **Client Edge**, **Modal Frame** na **TRUE**



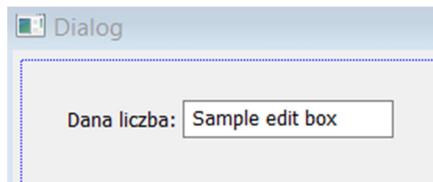
- Modyfikacja ustawień tych trzech właściwości plus **Sunken** dają różne efekty wizualne. Popróbować różnych kombinacji!
- Wstawianie okienek edycyjnych: Trzeba wstawić najpierw okno statyczne z opisem jaka dana będzie w oknie edycyjnym a potem okno edycyjne (najczęściej i je odpowiednio wyrównać. Z lewej (VS 2017) jest zwinięte okno **Toolbox** – należy go rozwinąć. Zawiera ono kontrolki jakie mogą być wstawione do ona dialogowego



- Wstawić do okna dialogowego **Static Text** i na prawo **Edit Control**, ustawić identyczną wysokość najpierw a potem wyrównać w poziomie (static będzie minimalnie wyżej) – do operacji potrzebne zaznaczenie kilku kontrolek – lewy klawisz myszy + CTRL, jedna kontrolka jest zaznaczana wypełnionymi granatowymi kwadracikami – do tego elementu będzie wyrównywanie. Na górze ekranu jest dodatkowy toolbar z przyciskami dla różnych operacji wyrównywania i modyfikowania ułożenia względnego kontrolek.



- Można lekko poprawić static aby napisy były równo w poziomie.
- Dla static-a zmodyfikować atrybuty:
 - ✓ Caption: Dana liczba
- Dla edit-a zmodyfikować atrybuty:
 - ✓ ID: IDC_DATANUM
 - ✓ Number: TRUE
- Modyfikacje spowodowały ze należy poprawić rozmiar static-a oraz odsunąć edit-a.



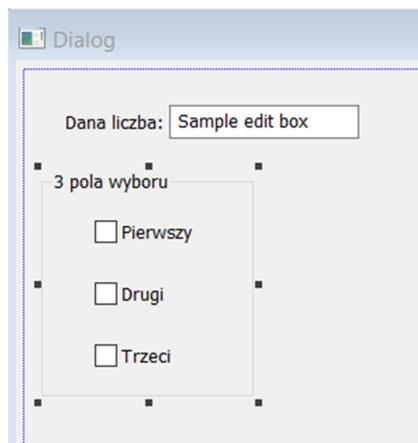
- Przetestować wciskając zaznaczony strzałką przycisk na toolbarze:



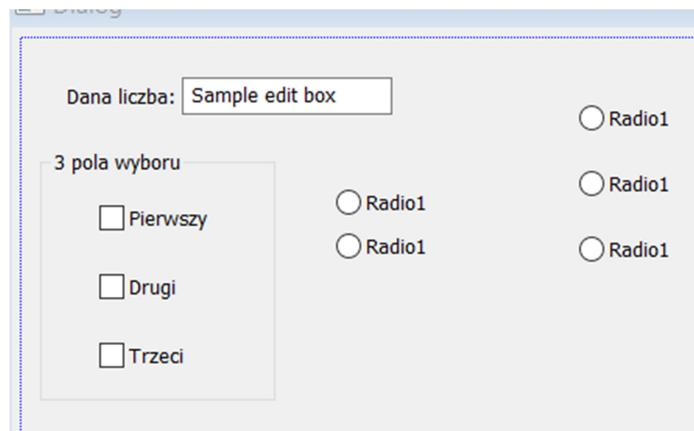
- Proszę spróbować wpisać dowolne znaki, potem, liczbę ujemną, potem liczbę z kropką – nie da się. Tylko cyfry liczby naturalnej. Przyciśnięcie OK lub Anuluj zamyka okno.
- Teraz wstawiamy pola check box: wstawić trzy po kolej i jeden nad drugim
- Wyrównać do lewej (wszystkie trzy zaznaczone), i ustalić identyczne odległości między nimi



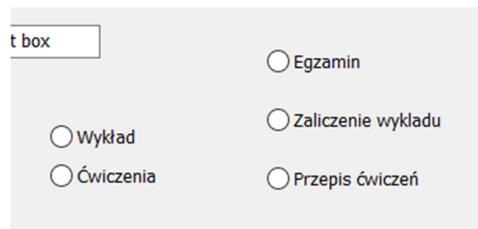
- Ustalić Caption i ID dla pól wyboru odpowiednio:
 - ✓ Pierwszy, IDC_FIRST
 - ✓ Drugi, IDC_SECOND
 - ✓ Trzeci, IDC_THIRD
- Dodać kontrolkę Group Box i zmienić mu Caption na 3 pola wyboru.



- Następnie dodajemy dwie grupy pól Radiobox. Można wstawić jedną kontrolkę i potem je skopiować: CTRL+C -> CTRL+V. przesuwanie można zrealizować też strzałkami na klawiaturze co nieraz jest znacznie bardziej precyzyjne. Aby zaznaczyć kilka kontrolek to można przycisnąć (nie puszczać) lewy klawisz myszki np. powyżej i na lewo od skrajnej kontrolki i pociągnąć myszką w prawo w dół i puścić. Wszystkie kontrolki będące wewnętrz zaznaczonego obszaru się zaznaczą. Ponieważ były kopiowane to wszystkie mają Caption Radio1.

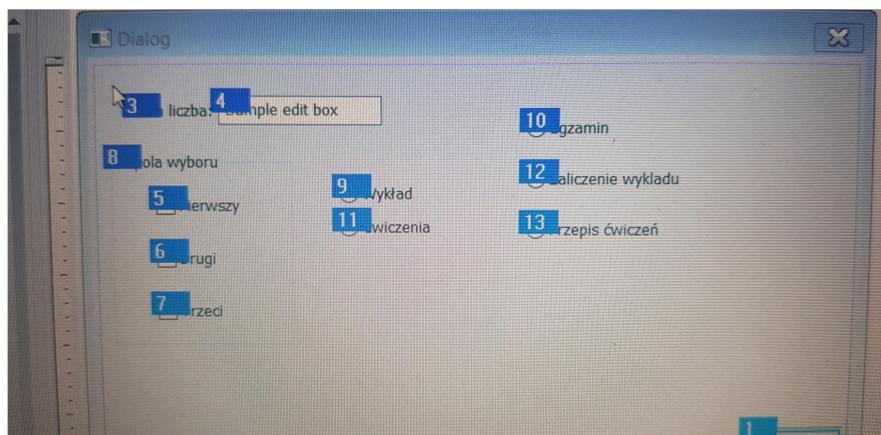


- Teraz pozmieniać napisy (Caption):

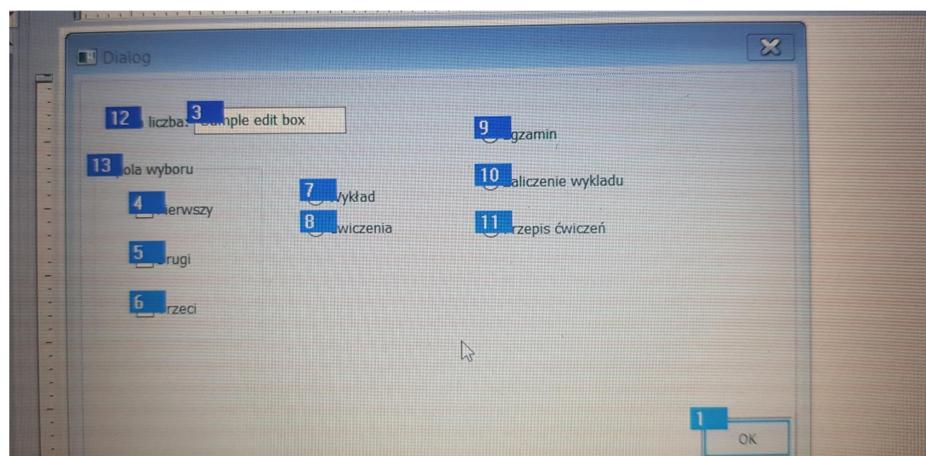


- Identyfikatory: ID_WYKL, IDC_CW, IDC_EGZ, IDC_ZALWYK, IDC_PRZEPIS

- Można uruchomić testowo. Grupy radiowe będą źle działać. Powinny być niezależne od siebie. Aby to zrobić należy najpierw wyświetlić kolejne numery przeskakiwania tabulatora.
- Z menu środowiska - Format wybrać Tab Order. Wyświetli się koło każdej kontrolki kolejny numer (kolejność przeskoku tabulatora). **Każdy z Państwa może mieć trochę inaczej w zależności od kolejności wstawiania kontrolek!!!**

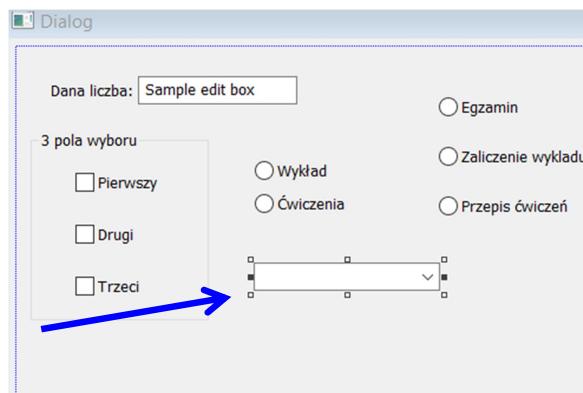


- Teraz należy zrobić tak aby w grupie RadioBox kontrolki były po **kolei**. Robi się to poprzez przeklikanie kolejnych kontrolek – będą się ustawiać numerki w kolejności kliknięć. Ja nie brałem pod uwagę okien statycznych ani Group Box (nieistotne).

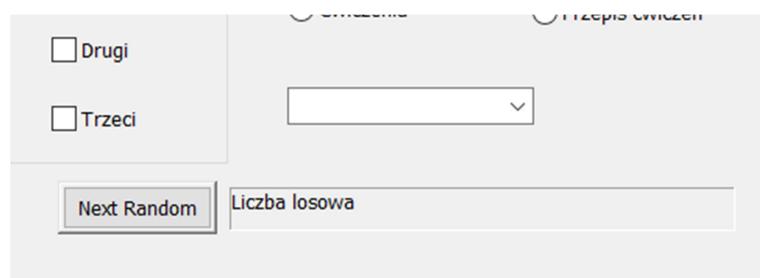


- Teraz dla **każdego pierwszego** elementu grupy radiowej (Wykład i Egzamin) należy zdefiniować własność Group – TRUE. Przetestować.

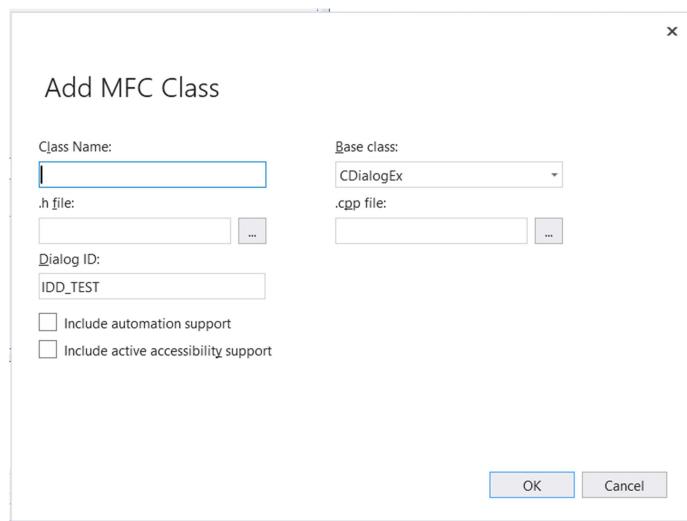
- Następnie dodać do okna dialogowego kontrolkę Combo Box – zmienić identyfikator na IDC LETTERS. Następnie w polu właściwości **Data** wpisać po kilka razy duże litery alfabetu oddzielone średnikami: AAAAA;BBBBB;CCCC; DDDD;EEEEEE;FFFFF;GGGGG.



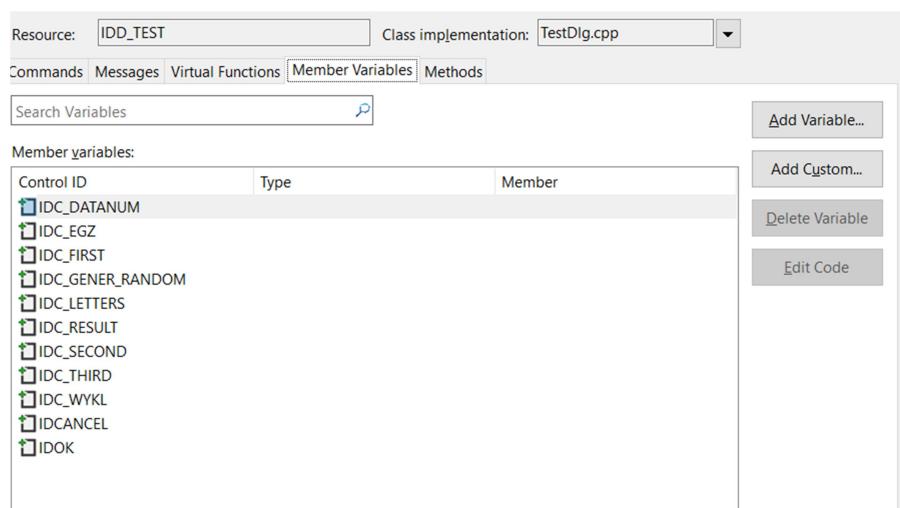
- Uruchomić i przetestować
- Następnie wstawimy jeszcze:
 - ✓ kontrolkę Button z identyfikatorem IDC_GENER_RANDOM i Caption: Next Random, własność Modal Frame na TRUE
 - ✓ okno Static Text, do którego wpiszemy potem liczbę losową (nie jest to okno edycyjne ale można do niego wpisać tekst). Caption: Liczba losowa, Sunken: TRUE. Po wstawieniu należy zmienić identyfikator: IDC_RESULT. Nie może zostać IDC_STATIC bo takich kontrolek może być w oknie dialogowym kilka.



- Teraz jak obsłużyć takie okno dialogowe programowo? Jak ustawić kontrolki przed pierwszym pojawiением się okna, jak ściągnąć z niego dane i wyświetlić dane?. Jak sprawdzić czy okno dialogowe zostało zamknięte przyciskiem OK?
- W pierwszej kolejności należy wygenerować klasę dla zaprojektowanego okna dialogowego. Należy kliknąć na tle okna dialogowego lewym klawiszem myszki i pokaże się okienko:



- Wpisać nazwę klasy Class Name: CTestDlg oraz Base Class można zmienić na Dialog. Usunąć literkę C z nazw plików (.h i .cpp) i zaakceptować. Wygenerują się dwa pliki: TestDlg.h i TestDlg.cpp czyli definicja klasy i jej implementacja.
- Wracamy do okna dialogowego. Należy dodać do klasy okna dialogowego odpowiednie zmienne (odpowiednich typów w zależności jak będą obsługiwane kontrolki). Można to zrobić klikając kontrolkę np. IDC_DATANUM i wybrać Add Variable. Ponieważ jednak będziemy dodawać więcej zmiennych **lepiej** po kliknięciu prawym klawiszem na tło okna dialogowego wybrać **Class Wizard** i następnie wybrać zakładkę Member Variables:



- Następnie zaznaczyć identyfikator IDC_DATANUM i przycisk z prawej Add Variable. Pojawi się okno do wstawiania do klasy zmiennych. Zmienne mają być **prywatne**. Nazwę

wpisujemy ręcznie, mają mieć przedrostek **m_ !!** (**moje wymaganie co do projektów!!**) Ustala się czy zmienna jest typu Control (kontrolka) czy Value (wartość z kontrolki), typ wpisać ręcznie, można wpisać komentarz. Zaakceptować przyciskiem **Finish**.

Add Control Variable

General settings

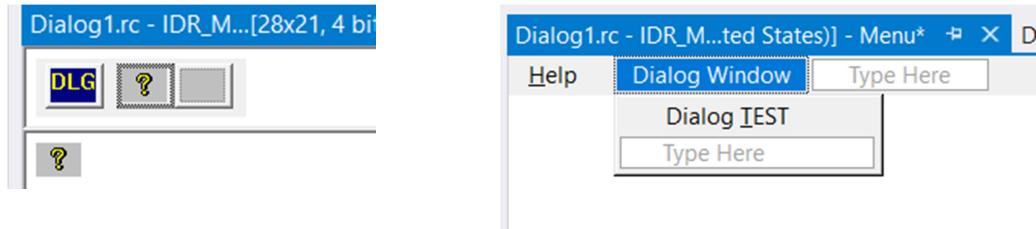
Control	Control ID:	Control Type:
Other	IDC_DATANUM	EDIT
	Category:	Name:
	Value	m_nNum
	Access:	Variable Type:
	public	int
	Comment:	Liczba naturalna

- Dodać zmienne typów jak na obrazku poniżej (**wszystkie prywatne**). Ważne:
 - ✓ z polami wyboru łączymy zmienne typu BOOL i po wartościach TRUE i FALSE będzie wiadomo czy dany Check Box został zaznaczony czy nie.
 - ✓ Dla grup radiowych łączymy tylko **jedną zmienną** typu int z pierwszą kontrolką Radio Box (z właściwością Group == TRUE). Będzie to wartość „0-based” (jak indeks tablicy). Czyli jest to numer kontrolki radiowej numerowany od 0.
 - ✓ Dla Combo dodajemy zmienną, która jest typu Control (automatycznie będzie CComboBox).

Member variables:		
Control ID	Type	Member
IDC_DATANUM	int	m_nNum
IDC_EGZ	int	m_nAcceptanceKind
IDC_FIRST	BOOL	m_bFirst
IDC_GENER_RANDOM		
IDC_LETTERS	CComboBox	m_ComboLetters
IDC_RESULT		
IDC_SECOND	BOOL	m_bSecond
IDC_THIRD	BOOL	m_bThird
IDC_WYKL	int	m_nLectureKind
IDCANCIFI		

- Zmienne zostały defaultowo zainicjowane w konstruktorze. Można to zmienić np. dla **m_bSecond** zmienić na TRUE, **m_nNum** zmienić na 1 oraz **m_nAcceptanceKind** na 2.

- Dla uruchomienia okna dialogowego należy zdefiniować nowy przycisk na toolbarze (usunąć wszystkie z wyjątkiem pytajnika) aplikacji o identyfikatorze ID_DLG_TEST. Następnie dodać w menu aplikacji element o takim samym identyfikatorze (usunąć wszystkie poza Help).



- Teraz można dodać **do klasy Widoku** metodę `OnDlgTest()` (automatycznie proponowana nazwa) reagującą na wybór albo elementu menu albo przycisku na toolbarze. Prawy klawisz myszy na nowym elemencie menu i Add Event Handler (komunikat typu COMMAND).
- Teraz w klasie widoku musimy **zainkludować plik nagłówkowy** z definicją klasy okna dialogowego!!
- W metodzie `OnDlgTest()` należy wykreować obiekt klasy `CTestDlg` i wywołać dla niego metodę `DoModal()`;

```
void CDialog1View::OnDlgTest()
{
    CTestDlg dlg;
    dlg.DoModal();
}
```

- Uruchomić aplikację i uruchomić okno dialogowe. Po wyświetleniu okna dialogowego w oknie edycyjnym będzie 1, pole wyboru Drugi będzie zaznaczone, oraz w pierwszej grupie radiowej będzie wybrany Wykład a w drugiej grupie radiowej Przepis ćwiczeń.

===== CZĘŚĆ DRUGA – OBSŁUGA OKNA DIALOGOWEGO =====

- Po pierwsze jak ustawać kontrolki w zależności od wartości zmiennych z nimi związanych. Proste ustawienia robi się w konstruktorze nadając zmiennym odpowiednie wartości jak zrobiono w części pierwszej ćwiczenia. Natomiast dla zmiennych typu Control należy np. dodać do klasy implementację metody wirtualnej `OnInitDialog()` (np. z Class Wizard-a).

W niej można zmienić ustawienia kontrolek przed pokazaniem się okna dialogowego. Przykładowo, wywołanie metody `SetCurSel(3)` dla zmiennej kontrolnej związanej z `ComboBox`, spowoduje ustawienie w oknie `ComboBox` czwartego napisu (0-based) czyli DDDD. Można też, gdy okno już jest wyświetcone modyfikować kontrolki poprzez najpierw zmianę zmiennych z nimi związanych a potem wywołać metodę `UpdateData(FALSE)`.

```
BOOL CTestDlg::OnInitDialog()
{
    CDialog::OnInitDialog();

    m_ComboLetters.SetCurSel( 3 );

    return TRUE; // return TRUE unless you set
                 // EXCEPTION: OCX Property Pa
}
```

- Po drugie, jak odczytać wartości zmiennych związanych z kontrolkami? Można to zrobić przed zamknięciem okna dialogowego w metodzie reagującej na wybór przycisku OK i zdefiniować odpowiednie metody publiczne dające dostęp do tych składowych – po zamknięciu okna dialogowego obiekt okna istnieje dalej więc można poprzez niego wywołać odpowiednie metody publiczne zwracające wartości kontrolek. Albo bez zamykania okna dialogowego w dowolnej metodzie reagującej na jakieś zdarzenie od kontrolek. Należy w obu przypadkach wywołać metodę `UpdateData()` (parametr TRUE domyślny).
- Dodać do klasy okna dialogowego funkcję reagującą na przycisk OK oraz na przycisk (Button IDC_GENER_RANDOM). Można kliknąć prawym klawiszem w button IDC_GENER_RANDOM w oknie dialogowym i z wyświetlonego menu wybrać Add Event Handler -> Message type: BN_CLICKED, Function handler name: OnGenerRandom. Sprawdzić czy wstawi się do klasy CTestDlg. Dla przycisku OK proszę dodać metodę OnOK.

```
void CTestDlg::OnGenerRandom()
{
    UpdateData(); // default param = TRUE i.e. data is being retrieved
    // UpdateData( FALSE ); // dialog box is being initialized
}

void CTestDlg::OnOK()
{
    UpdateData();
    CDialog::OnOK();
}
```

- Proszę zaimplementować metodę `OnGenerRandom()` tak aby generowała kolejną liczbę losową i następnie ją wyświetli w oknie statycznym `IDC_RESULT` (zainicjować generator w konstruktorze - `srand((unsigned int)time(NULL));`)

```
void CTestDlg::OnGenerRandom()
{
    int randNum = rand();
    CString str; str.Format( L"%d", randNum );
    CWnd* pWnd = GetDlgItem( IDC_RESULT );
    pWnd->SetWindowTextW( str );
}
```

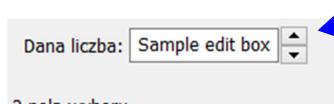
- Metoda `GetDlgItem()` zwraca wskaźnik na okno kontrolki o identyfikatorze będącym parametrem. Metoda `Format()` obiektu `CString` działa podobnie jak `sformat()` zwykłego C.
- Następnie wyświetlimy wartości zmiennych związanych z kontrolkami. Podczas testu należy poustawić kontrolki (nie działa na liczbę losową bo nie ma zmiennej związanej z polem statycznym) i przycisnąć przycisk OK. Wtedy wyświetli się okno informacyjne zawierające wartości odpowiednich zmiennych. Implementacja w metodzie `OnOK()` jak poniżej:

```
void CTestDlg::OnOK()
{
    UpdateData();
    CString str;

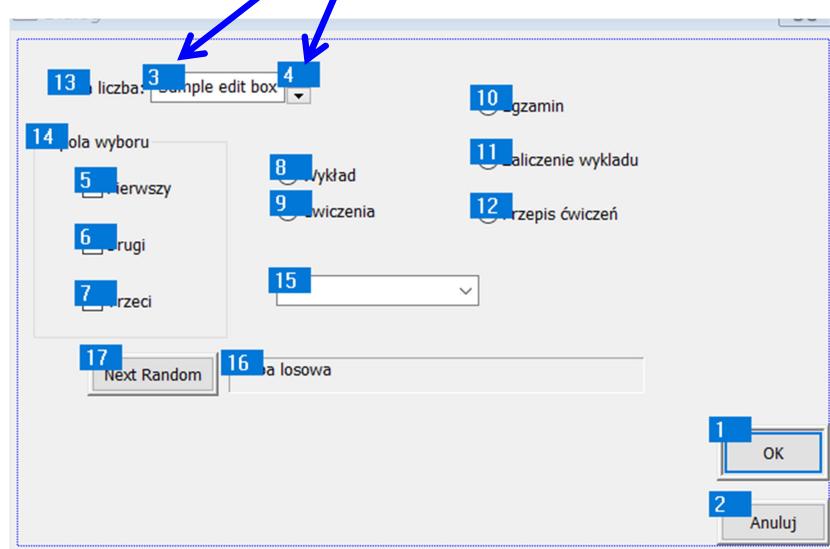
    int pos = -1;
    CString s;
    pos=m_ComboLetters.GetCurSel();
    m_ComboLetters.GetLBText( pos, s );

    str.Format( L"num=%d, first=%d, second=%d, third=%d, LKind=%d, AKind=%d, Combo_pos=%d Combo_str=%s",
               m_nNum, m_bFirst, m_bSecond, m_bThird, m_nLectureKind, m_nAcceptanceKind,
               pos, s );
    AfxMessageBox( str );
    CDialog::OnOK();
}
```

- Wartości logiczne wypiszą się jako 0 lub 1, wybrana pozycja Combo wyświetli się jako indeks pozycji oraz jako string. Uruchomić aplikację i przetestować!!!
- Teraz dodamy do okna edycyjnego (`IDC_DATANUM`) kontrolkę typu SpinControl i ustawiemy ją z prawej strony okna dialogowego. Ustawić identyfikator na `IDC_SPIN_NUM`.



- Na razie jeszcze nie będzie działać. Najpierw należy zmienić ustawienia i kolejność kontrolek Tab Order.
- Ustawić dla kontrolki Spin Control właściwości: Auto Buddy na TRUE oraz Set Buddy Integer na TRUE.
- Następnie wyświetlić Menu ->Format-> Tab Order. Przekliknąć kolejność tak aby Spin Control miał **kolejny** numer po oknie edycyjnym!! (wystarcz kliknięcie kolejno w: OK, Anuluj, okno edycyjne i spin)



- Przetestować okno dialogowe i jak działa spin. Niestety domyślnie strzałka w dół zwiększa liczbę w oknie edycyjnym a w górę zwiększa. Jest tak zrobione celowo aby wymusić na użytkowniku ustawienie wartości minimalnej i maksymalnej.
- Do tego najpierw dodać do klasy okna dialogowego zmienną kontrolną związaną z Spin Control o nazwie np. `m_Spin` (będzie typu `CSpinButtonCtrl`). Następnie w metodzie `OnInitDialog()` użyć metody `SetRange()`:

```
CDialog::OnInitDialog();

m_ComboLetters.SetCurSel( 3 );
m_Spin.SetRange( 0, UD_MAXVAL );
```

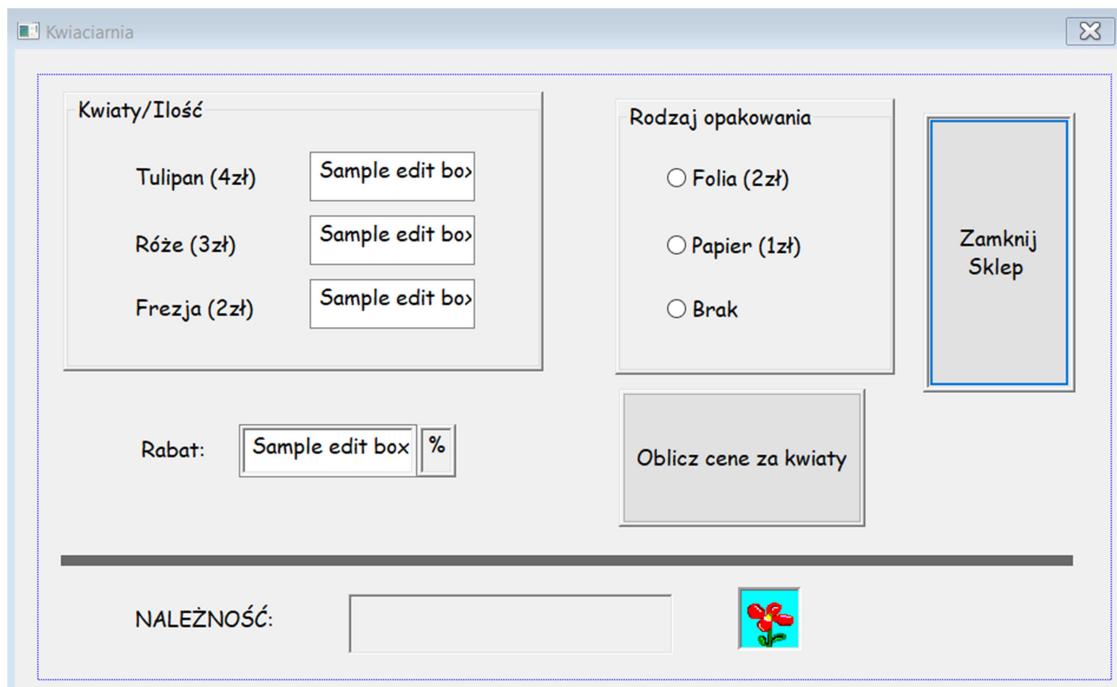
- Jeśli pierwszy parametr jest mniejszy od drugiego to strzałka w góre będzie zwiększać liczbę w oknie edycyjnym, jeśli większy to będzie zmniejszać jak jest default-owo.

- Przetestować zaprojektowane okno dialogowe uruchamiając aplikację.
- Na koniec jak sprawdzić w aplikacji, że okno dialogowe zostało zaakceptowane buttonem OK? Poprawić implementację metody klasy widoku `OnDlgTest()` jak poniżej:

```
void CDlg1View::OnDlgTest()
{
    CTestDlg dlg;
    if( dlg.DoModal() == IDOK )
        AfxMessageBox( L"Ustawienia w oknie dialogowym zaakceptowane!" );
    // majac zdefiniowane metody publiczne w klasie okna dialogowego zwracajace
    // wartosci skladowych zwiiezanych z kontrolkami
    // mozna teraz pobrac wartosci ustawien kontrolek okna dialogowego
    else
        AfxMessageBox( L"Rezygnacja z ustawien" );
}
```

Zadanie 1:

- Zadanie do zrobienia jako ćwiczenie po powyżej opisanym programie. Napisać aplikację dla sklepu z kwiatami sprzedający 3 rodzaje kwiatów, w 2 rodzajach opakowania lub bez, z uwzględnieniem ewentualnego rabatu (domyślnie 0%).



- W zaprezentowanym oknie dialogowym zwiększo czcionkę (właściwości okna dialogowego) na Comic Sans MS (11). Kreska pozioma to obiekt Picture Control z zmienioną właściwością Type ma Rectangle (domyślnie jest Frame). Obrazek kwiatka to również Picture Control ale z właściwością Type ustawioną na Bitmap i Image na identyfikator bitmapy (48x48) zdefiniowanej wcześniej w zasobach aplikacji.

Zadanie 2:

- Zrealizować w oknie dialogowym kalkulator liczb zespolonych.
- Dla realizacji kalkulatora należy zaimplementować klasę liczb zespolonych CComplex ze składowymi m_re oraz m_im typu double oraz zaimplementować:
 - ✓ Konstruktor z parametrami domyślnymi,
 - ✓ Konstruktor kopiący (musi wywołać operator = - bo tak należy podstawnienie pisać co wynika z tego jak jest wykonywana druga instr. CComplex z1(1,2); CComplex z2=z1;). Tu nie ma to znaczenia ale przy składowych dynamicznie alokowanych ma.
 - ✓ Operator =
 - ✓ funkcje operatorowe: + - * / (friend-y obowiązkowo – metody globalne). Dla implementacji wywołać odpowiadające operatory <op>= (obowiązkowo) np. + wywołuje +=.
 - ✓ funkcje operatorowe: +=, -=, *=, /= (metody klasy).
 - ✓ Funkcje operatorowe == i != (friend). Implementacja tylko np. pierwszej a druga zwraca wartość pierwszej z zaprzeczeniem.
 - ✓ Metody set (trzy) i get (dwie) ustawiające składowe liczby zespolonej i zwracające składowe
 - ✓ Operatory we/wy (friend). Wypisywanie w postaci pary lub w postaci kanonicznej - niezbedna klasa CCanonComplex pochodna od CComplex mająca jedynie konstruktor z parametrem const CComplex& oraz globalną (friend) operacją wypisywania.

- ✓ Metody obliczające moduł (`Module()`) oraz liczbę sprzężoną (`Coupled()`)
- Zaprojektować odpowiednio okno dialogowe. W oknie dialogowym kalkulatora musi być możliwość wpisania dwóch liczb zespolonych, możliwość wyboru operatora +, - *, / oraz obliczenia modułu i liczby sprzężonej. Wynik jest wyświetlany w oknie dialogowym po kliknięciu w stosowny Button.
- Należy **zachować podane nazwy składowych, metod i klas oraz typy metod!**
Warunek zaliczenia programu.