## 1. Wprowadzenie do przedbiegów

Budując kalkulator liczb zespolonych obsługę zdarzeń realizowaliśmy najprawdopodobniej bez wsparcia klasy (klas) zdefiniowanych przez nas na potrzeby realizowanej aplikacji. Po pierwsze dlatego, że nie było takiego polecenia ze strony prowadzącego, po drugie bo byliśmy skupieni na realizacji zadania, którym była budowa samego interfejsu. Przy wykorzystywaniu biblioteki *wxWidgets* do realizacji graficznego interfejsu użytkownika dla dużych projektów, wygodniej jest trzon oprogramowania zapisać w plikach dodatkowych dołączając je do projektu *wxWidgets* dyrektywą *#include*.

## 2. Studium przypadku – kalkulator liczb zespolonych

W przypadku kalkulatora liczb zespolonych można wykorzystać na przykład strukturę o nazwie *Zespolona* umożliwiającą przy pomocy dwóch pól typu double zdefiniowanie części rzeczywistej i urojonej liczby zespolonej oraz klasę o nazwie *Obliczenia*, która przy pomocy struktury *Zespolona* realizuje w odpowiednich metodach podstawowe operacje matematyczne wykonywane na liczbach zespolonych typu *Zespolona*.

Przykładowo klasa (*Obliczenia*) wraz ze strukturą (*Zespolona*) może się znajdować w pliku o nazwie

*Zespolone.h*

który dołączymy dyrektywą

*#include "Zespolone.h"*

do pliku o nazwie

*kalkulatorMain.h*

oczywiście przy założeniu, że projekt nazywa się *kalkulator* (wobec tego podstawowe pliki projektu otrzymają nazwy: *kalkulatorApp.h*, *kalkulatorApp.cpp*, *kalkulatorMain.h* oraz *kalkulatorMain.cpp*).

Jeśli teraz chcielibyśmy w pliku głównym aplikacji powołać do życia obiekt o nazwie *Oblicz* będący instancją klasy *Obliczenia*, czyli

*Obliczenia Oblicz;*

najlepiej zrobić to w pliku *kalkulatorMain.h* w ciele klasy *kalkulatorFrame* dziedziczącej po klasie *wxFrame* w części prywatnej. Oczywiście plik *Zesplone.h* powinien znajdować się w katalogu projektu *kalkulator* środowiska *Code::Blocks*.

Po wykonaniu tych czynności jesteśmy gotowi, aby zająć się programowaniem poszczególnych zdarzeń okna tworzonej aplikacji. Ale z tym nie powinniśmy mieć większych problemów dlatego, że czynności te były już wykonywane na poprzednich zajęciach.

## 3. Aplikacje graficzne z wykorzystaniem kontenerów

Teraz przystępujemy do zasadniczej części zajęć, dla której aplikacja kalkulator stanowiła tylko wprowadzenie. W tym celu wykorzystamy jedno z zadań zrealizowanych w ramach instrukcji 6, którego celem było wykorzystanie kontenera *list* do zaimplementowania stosu (zadanie nr 7), kontenera *set* do implementacji listy uczniów (zadanie nr 8) oraz kontenera *map* do utworzenia słownika angielsko-polskiego (zadanie nr 9).

Niniejszy opis będzie dotyczył ostatniego przypadku, czyli wykorzystania kontenera *map* do utworzenia słownika angielsko-polskiego, z tym, że tym razem celem będzie wykonanie graficznego interfejsu użytkownika przy wykorzystaniu biblioteki *wxWidgets*.

Przyjmujemy założenie, że projekt będzie nosił nazwę *slownik*. W związku z tym podstawowe cztery pliki aplikacji będą nosiły nazwy: *slownikApp.h*, *slownikApp.cpp*, *slownikMain.h* oraz *slownikMain.cpp*. Pliki projektu będą się znajdować w katalogu roboczym o nazwie *slownik*. W tymże katalogu powinien się znaleźć plik

*Slownik.h*

zawierający odpowiednią klasę, na przykład o nazwie *Slownik*, która w części publicznej posiada odpowiednie metody umożliwiające dodawanie nowego słowa z tłumaczeniem do słownika, usuwanie wybranego słowa z tłumaczeniem ze słownika itd., które będą obsługiwać zdarzenia pojawiające się w oknie tworzonej aplikacji. Plik *Slownik.h*

dołączamy do projektu dyrektywą

*#include Slownik.h*

którą umieszczamy w pliku o nazwie

*slownikMain.h*

W tymże pliku w ciele klasy *slownikFrame*, która dziedziczy po klasie *wxFrame*, w części prywatnej powołujemy do życia obiekt *Slow* będący instancją klasy *Slownik*, czyli

*Slownik Slow;*

Teraz jesteśmy przygotowani do budowy graficznego interfejsu użytkownika wykorzystując w tym celu plik *slownikframe.wxs*. Aby zapewnić dopasowanie rozmiarów okna aplikacji do znajdujących się w niej kontrolek wykorzystamy elastyczność sizerów poznanych na poprzednich zajęciach. Sugerowany wybór to sizer *wxFlexGridSizer*

(znajdujący się w palecie *Layout*), od którego zaczynamy budowanie interfesju. Następnie można użyć notatnika, czyli *wxNotebook* (paleta *Standard*). Przeglądanie stron notatnika z kontrolkami zapewni nam klasa *wxPanel* (paleta *Standard*). W klasę *wxNotebook* upychamy tyle paneli *wxPanel*, ile stron zamierzamy wykorzystać w tworzonej aplikacji. Proponuje się jedną stronę wykorzystać na wprowadzanie nowej pary wyrazów do słownika, drugą na tłumaczenie wyrazu i ostatnią – trzecią stronę – na listę, dzięki której będzie można wyświetlić zawartość słownika razem z przyciskiem do kasowania wybranego elementu listy. Chociaż równie dobrze na stronie z listą mogą znajdować się dwa przyciski (jeden do kasowania pojedynczego zestawu słów, a drugi do kasowania całej listy), ale to już pozostawia się decyzji wykonawcy projektu. Aby umieścić w sposób samookreślający się przyciski w panelu proponuje się użyć ponownie sizera *wxFlexGridSizer*, w którym przy pomocy opcji *cols* decydujemy w ilu kolumnach chcemy umieścić kontrolki w każdym z paneli. Sugerowane kontrolki, to znane z poprzednich projektów *wxStaticText* – do tworzenia etykiet, *wxTextCtlr* – do pobierania i wyprowadzania tekstu oraz klawisz *wxButton* – do generowania odpowiednich zdarzeń. Wszystkie wymienione kontrolki znajdują się w palecie *Standard*. W drugim panelu przed umieszczeniem kontrolek umieszczamy sizer *wxFlexGridSizer*. To samo powtarzamy dla trzeciego panelu. Z tym, że w trzecim panelu powinnyśmy umieścić listę. Sugerowany typ listy to *wxListCtrl* (paleta *Standard*). W przypadku tego typu listy ważne jest żeby w pliku *slownikMain.cpp* w ciele konstruktora okna aplikacji (*slownikFrame*) zainicjować dwie kolumny listy *wxListCtrl* stosując metodę *InsertColumn()*, podając w nawiasie metody między innymi numer kolumny, jej nazwę, format wyświetlanych danych oraz rozmiar kolumny. Ponieważ mamy dwie kolumny, ich rozmiar powinien stanowić połowę rozmiaru szerokości listy (w tym przypadku 220), którą znajdziemy w jej konstruktorze. Proponowany fragment kodu może wyglądaj na przykład tak:

```
ListCtrl1->InsertColumn(0,"Polski",wxLIST_FORMAT_LEFT,110);
ListCtrl1->InsertColumn(1,"Angielski",wxLIST_FORMAT_LEFT,110);
```

Poza tym ważne jest, aby listę uzupełniać podczas każdego dodawania do słownika nowej pary wyrazów. Należy jednak pamiętać, że jest to tylko propozycja rozwiązania zadania.

## 4. Zadania do wykonania

1. Zmodyfikować obsługę zdarzeń w kalkulatorze z zadania nr 3 z instrukcji laboratoryjnej 10, zgodnie z sugestiami dotyczącymi przedbiegów.
2. Zaimplementować *GUI* przy pomocy biblioteki *wxWidgets* do jednego z programów utworzonych w ramach realizacji instrukcji 6, wykorzystując do tego wskazówki zawarte w sekcji dotyczącej aplikacji graficznych z wykorzystaniem kontenerów z niniejszej instrukcji laboratoryjnej.