

REV	DATA	ZMIANY

KONWERTER WALUT

Autor: Wojciech Rębisz
Akademia Górniczo-Hutnicza

Kraków (C) 2019

Spis treści

1. WSTĘP.....	3
2. WYMAGANIA SYSTEMOWE (<i>REQUIREMENTS</i>).....	4
3. FUNKCJONALNOŚĆ (<i>FUNCTIONALITY</i>).....	5
4. ANALIZA PROBLEMU (<i>PROBLEM ANALYSIS</i>).....	6
5. PROJEKT TECHNICZNY (<i>TECHNICAL DESIGN</i>).....	7
6. PODRĘCZNIK UŻYTKOWNIKA (<i>USER'S MANUAL</i>).....	8

1. Wstęp

Dokument dotyczy programu służącego do obliczania walut. Celem aplikacji jest zamiana jednej waluty, na drugą walutę (np. Dolary na Euro). Dodatkowo podajemy liczbę, która odzwierciedlała będzie ilość gotówki.

2. Wymagania systemowe (*requirements*)

Podstawowe założenia projektu

1. Celem projektu jest stworzenie prostej aplikacji – kalkulatora walut, który poprzez Internet łąduje bieżące kursy walut, później na ich podstawie oblicza wynik.
2. Aplikacja składa się zasadniczo z dwóch modułów:
 - klasy currency, hierarchii, funkcji pomocniczych, design patterns realizujących działania kalkulatora (jak najbardziej niezależne od platformy i języka)
 - interfejsu użytkownika (prosty dialog pod Windows)

3. Funkcjonalność (*functionality*)

Konwerter służy do zamiany jednej waluty na drugą. Całość składa się z dwóch plików klasy `currency` oraz funkcji pomocniczych `helpFunction`

- **Klasa `currency` (`currency.cpp`)**

Klasa przechowuje 4 zmienne;

dwie typu `string`:

`code`, `name`

dwie typu `double`:

`buy`, `sell`

oraz funkcje zaprzyjaźnione:

```
friend double exchange_values()
```

```
operatory strumieniowania (ostream&, istream&)
```

- **Funkcje pomocnicze (`helpFunction.cpp`)**

```
void download_current_state();
```

-pobiera plik `current_rates.xml` z obecnymi kursami walut ze strony <https://www.nbp.pl/kursy/xml/lastc.xml>

```
double comma_to_dot(string com);
```

-funkcja zamienia napis `com` na `double` niezależnie od tego czy jest podany z kropką czy z przecinkiem

```
vector<currency> load_data(vector<string> *s);
```

-otwiera wcześniej pobrany plik oraz ładuje dane do klasy `currency` za pomocą funkcji `regex`. Efektem tego jest utworzenie tablicy z wartościami. Zmienna `s` jest wskaźnikiem do kodów użytych w programie (tak abyśmy mogli operować na oryginałach)

```
double exchange_values(string code, string code_2, double cash);
```

-oblicza na podstawie algorytmu wartość po zamianie(algorytm w punkcie 5 dokumentacji)

```
vector<string> giveCodes();
```

-zwraca `vector` kodów które zostaną użyte w combobox

4. Analiza problemu (*problem analysis*)

Mając dwie klasy currency A, B każda z nich posiada zawarte w niej atrybuty (tabela 1-1) :

- kod (3-literowe np. USD)
- nazwa (dosłowna np. dolar amerykański)
- kurs kupna (liczba podawana z dokładnością do 4 miejsc po przecinku)
- kurs sprzedaży (patrz wyżej)

Pobieramy odpowiednio kurs kupna pierwszej oraz kurs sprzedaży drugiej (patrz rysunek 1-1 w rozdziale 4).

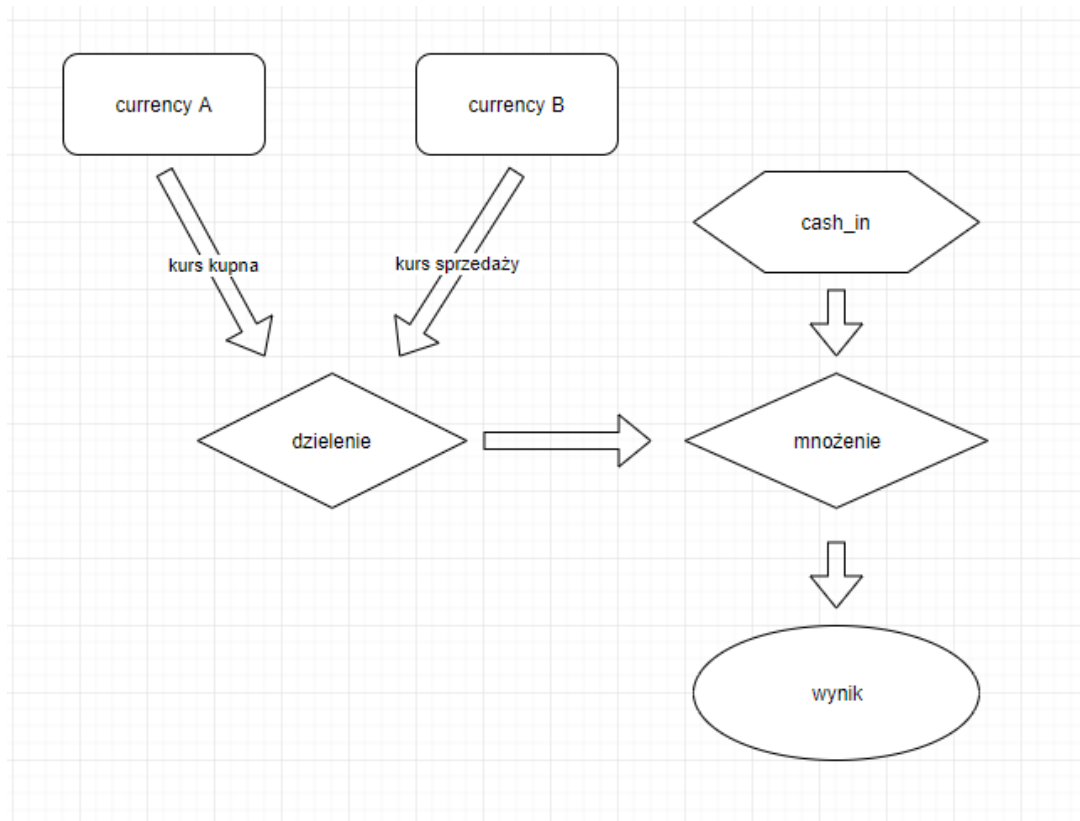
Dodatkowo należy podać jaką wartość chcemy obliczyć (cash_in).

Wynik otrzymamy wykonując odpowiednie działania.

Tabela 1-1. Przykładowa tabela wartości klasy currency

Kod	Nazwa	Kurs kupna	Kurs sprzedaży
USD	Dolar amerykański	3,7448	3,8204
EUR	Euro	4,2559	4,3419
...

5. Projekt techniczny (*technical design*)

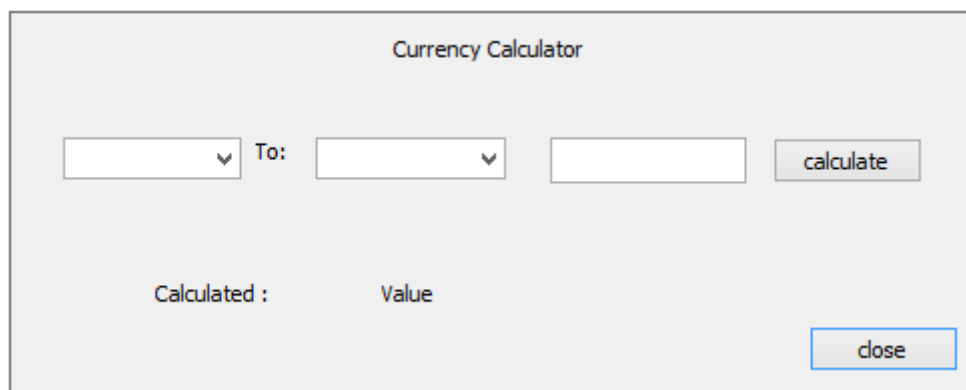


Rysunek 1-1. Schemat konwersji

Algorytm w postaci diagramu przedstawia kolejne etapy obliczeń występujących w programie-
funkcja `double exchange_values()` (patrz rysunek 3-1)

6. Podręcznik użytkownika (*user's manual*)

Całkowity wygląd aplikacji korzystającej z MFC jest pokazane niżej:



Rysunek 2-1. Ostateczny wygląd aplikacji MFC

- Lewy comboBox służy do wybrania waluty którą posiadamy z rozwijanej listy.
- Prawy comboBox (To:) służy do wybrania waluty na którą chcemy zamienić również w postaci rozwijanej listy.
- Następne jest pole w którym wprowadzamy kwotę opcjonalnie z kropką, bądź przecinkiem.
- Przycisk *calculate* służy do dokonania końcowych obliczeń.
- Wynik obliczeń zostanie ukazany w miejscu *Value*
- Przycisk *close* służy do wyjścia z aplikacji

W przypadku gdy nie podamy któregoś pola , system poinformuje nas o niepoprawności parametrów.


```
void CprojektDlg::OnBnClickedbtncalculate()
{
    CString _cashIN, _textLEFT, _textRIGHT, _textOUTPUT;

    GetDlgItemText(txtInput, _cashIN);
    m_comboBoxCtrl1.GetLBText(m_comboBoxCtrl1.GetCurSel(), _textLEFT);
    m_comboBoxCtrl2.GetLBText(m_comboBoxCtrl2.GetCurSel(), _textRIGHT);
    double cash_IN = _ttof(_cashIN);
    double result;
    CT2CA lewy(_textLEFT);
    CT2CA prawy(_textRIGHT);

    // construct a std::string using the LPCSTR input
    string kurs_kupna(lewy);
    string kurs_sprzedazy(prawy);

    result = exchange_values(kurs_kupna, kurs_sprzedazy, cash_IN);

    _textOUTPUT.Format(_T("%f"), result);

    SetDlgItemText(txtOutput, _textOUTPUT);
}
```

Rysunek 3-1. Kod wykonujący główną część programu

Powyższy kod przedstawia procedurę po wciśnięciu przycisku *calculate*. Do zmiennych typu CString wpisywane są wartości kolejnych zmiennych z pól:

comboBox:

_textLEFT

_textRIGHT

Pola tekstowego:

_cashIN

Pola wypisującego wynik:

_textOUTPUT