



KALKULATOR WALUTOWY

Autor: Krzysztof Liszka
18.01.2020r.

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>).....	10
6. GRAFICZNE PRZEDSTAWIENIE KODU	14
7. PODRĘCZNIK UŻYTKOWNIKA (<i>USER'S MANUAL</i>)	15

1.Wstęp

Dokument dotyczy opracowania kalkulatora walutowego. Celem tego projektu jest przekonwertowanie kwoty z jednej waluty do drugiej według wprowadzonych wcześniej kursów. Klient może skorzystać z 10 wprowadzonych wcześniej do programu walut wraz z ich aktualnymi kursami na dzień 17.01.2020r.

Kod programu stworzony został w języku C++ w środowisku Visual Studio 2019.

2. Wymagania systemowe (*requirements*)

1. Program powinien składać się z kilku niezależnych od siebie elementów, aby w razie implementacji zmian nie było konieczności modyfikacji całego kodu źródłowego, a tylko jego części w poszczególnych plikach. Złożenie funkcjonalnego kalkulatora walut pomaga również w analizie działania poszczególnych operacji, które wykonuje.

Jeżeli kursy, nazwy oraz symbole walut będą wprowadzone wcześniej w programie, powinny znajdować się w osobnym pliku. Ułatwi to ich modyfikację i pozwoli na łatwiejsze edytowanie, bądź dodawanie nowych.

Problemem przy operacjach walutowych i wszystkich innych związanych z pieniędzmi są typy zmiennoprzecinkowe (floating types), których powinniśmy unikać przy tego typu operacjach. Mogą się one przyczyniać do gubienia poszczególnych zmiennych w pamięci, a działanie całego programu byłoby uzależnione od danych wprowadzonych przez klienta. Aby temu zapobiec, powinna zostać stworzona dodatkowa funkcjonalność. Dzięki temu, kalkulator walutowy będzie poprawnie przeliczał wartości.

2. Architektura systemu:

- menu,
- struktura przechowująca waluty,
- plik zawierający aktualne kursy,
- konwertowanie walut,
- zapobieganie liczbom zmiennoprzecinkowym.

3.Funkcjonalność (*functionality*)

- a) Przedstawienie użytkownikowi wszystkich dostępnych walut, którymi można operować,
- b) Możliwość łatwej edycji kursów oraz ewentualne dodanie innych dodatkowych opcji wymiany,
- c) Możliwość konwersji walut,
- d) Umożliwienie wprowadzenia kwoty w dowolnym formacie,
- e) Zabezpieczenie użytkownika przed podaniem błędnego symbolu waluty,
- f) Zabezpieczenie użytkownika przed wprowadzeniem błędnej kwoty,
- g) Kontrola konwersji przy wprowadzeniu tych samych walut.

Z powyższego obrazka widać, że pamięć przechowująca wartość zmiennej jest podzielona na trzy części:

- bit znaku – 1bit,
- zestaw bitów określający wykładnik potęgi – 8 bitów,
- zestaw bitów określający mantysę.

Wartości z obrazka: 0, 123 oraz 5033165 są binarnymi reprezentacjami odpowiednio bitu znaku, wykładnika oraz mantysy. Na początku należy znaleźć znormalizowaną wartość liczby zmiennoprzecinkowej – dla bitu nieustawionego jest wartość dodatnia (wartość znaku równa 0), a dla bitu ustawionego jest to wartość ujemna (wartość znaku równa 1). W przypadku wykładnika, jego wartość binarna to 123. Do znalezienia właściwej wartości, należy odjąć od jego binarnej wartości wartość stałej, która umownie nazwana została **K**. Wynosi ona:

- 127 dla liczb pojedynczej precyzji,
- 1023 dla liczb podwójnej precyzji,
- 16383 dla liczb podwójnej precyzji rozszerzonej.

W przypadku 32-bitowego przykładu z obrazka będzie to $123 - 127 = -4$.

Następnym krokiem jest praca nad mantysą. Z perspektywy poszukiwania jej wartości, jest to liczba, która zawsze z przodu zaczyna się od 1. Stąd wiemy, że jest wartością z zakresu $(2.0, 1.0>$. Mając tę informację nie ma sensu zapisywać jej wartości w pamięci, w której faktycznie mantysa to liczby z zakresu $<1.0, 0.0>$. W tym miejscu można skorzystać z algorytmu na znalezienie wartości mantysy:

1. Mantysa przyjmuje wartość 0.0.
2. Wartość składnika przyjmuje wartość 1.0.
3. Przesunięcie się na pierwszy bit z lewej strony.

4. Wartość składnika dzielony jest przez 2.
5. Jeżeli bit jest ustawiony, to do mantysy dodaje się wartość składnika.
6. Przesunięcie się o jeden bit w prawo.
7. Powrót do punktu nr 4, jeżeli nie zaszło wyjście poza zakres bitów.

W analizowanym przykładzie mantysa będzie więc miała następującą postać:

```
0.5 + 0.0625 + 0.03125 + 0.00390625 + 0.001953125 +
0.000244140625 + 0.0001220703125 +
0.0000152587890625 + 0.0000076293945312 +
0.0000009536743164 + 0.0000004768371582 +
0.0000001192092896 = 0.600000023841858
```

Pozostaje jeszcze dodanie zawsze występującej jedynki i ostatecznie:

1.600000023841858

Wykorzystując ten zapis wartość konkretnej liczby zmiennoprzecinkowej można osiągnąć poprzez wykonanie obliczenia na podstawie wzoru:

$$L = (-1)^Z * M * 2^{K-e}$$

Po podstawieniu wyznaczonych przez nas wartości otrzymujemy:

$$L = (-1)^0 * 1.600000023841858 * 2^{127-123}$$

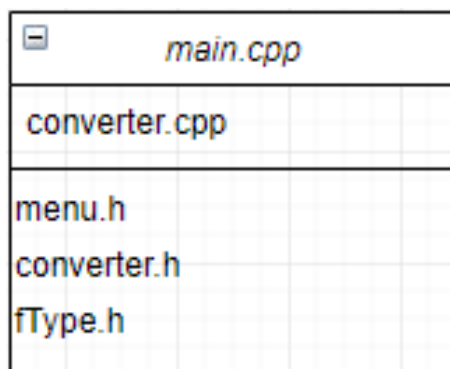
Ostateczna wartość to: **0.100000001490116119384765625**

Powyższa analiza problematycznej wartości 0,1 bardzo dobrze ukazuje skalę problemu stosowania typów zmiennoprzecinkowych do

operacji przy raportach sprzedażowych i innych różnych finansowych aspektach. Rozwiązaniem na ten problem jest użycie typu całkowitego, gdzie jako jednostkę potraktujemy najmniejszą możliwą wartość. W przypadku polskiego złotego jest to grosz, będąc w Wielkiej Brytanii mówimy o pensie, a jeśli chcemy korzystać z amerykańskiego dolara bądź najpopularniejszego w Unii Europejskiej euro – mamy do czynienia z centem.

5. Projekt techniczny (*technical design*)

Program składa się z 5 niezależnych od siebie elementów – trzy z nich to pliki nagłówkowe (menu.h, converter.h, fType.h), a dwa pozostałe są plikami źródłowymi (main.cpp, converter.cpp).



Rys 2. Schematyczne przedstawienie struktury programu.

1. menu.h

W pliku menu.h stworzone zostało menu do całego kalkulatora walutowego, które ukazuje się użytkownikowi bezpośrednio po uruchomieniu. Przedstawione zostały w nim dane o autorze, data sporządzenia kodu źródłowego, wszystkie dostępne waluty, z których klient może korzystać przy konwertowaniu, oraz opcje wyboru. Dzięki tym ostatnim, po udanej bądź nieudanej konwersji, użytkownik może wykonać akcję ponownie bez potrzeby ponownego uruchamiania programu. Ten plik nagłówkowy ma wyłącznie funkcję informacyjną dla klienta i ma za zadanie ułatwić poruszanie się po programie, nie wykonując przy tym żadnych operacji obliczeniowych.

2. Converter.h

Plik converter.h ma za zadanie utworzyć klasę walut i klasę symboli. Klasa map znajdująca się w pliku nagłówkowym map i biblioteczce standardowej jest

posortowanym kontenerem zawierającym pary klucz-wartość z unikalnymi kluczami. Jej cechą jest jednocześnie sortowanie podczas dodawania kolejnych elementów albo za pomocą domyślnego porównywania binarnego, albo za pomocą funkcji porównującej lub własnej klasy.

Konstruktor klasy `map` wymaga jedynie określenia typu dla klucza i wartości przechowywanej.

W klasie `currency`, obiekt klasy `map` przyjmuje jako klucz obiekty klasy `string`, natomiast jako wartości przechowywane pod danym kluczem zmienne typu `double`.

W klasie `symbol`, obiekt klasy `map` przyjmuje jako klucz i jako wartości przechowywane pod tym kluczem zmienne typu `string`.

3. `fType.h`

Plik nagłówkowy `fType.h` zawiera klasę `fType`, która powstała aby wyeliminować liczby zmiennoprzecinkowe z programu. Wcześniejsza analiza z pkt. 4 wykazała, że w przypadku pracy polegającej na zliczaniu pieniędzy, mogą być one niezwykle niebezpieczne. Aby temu zapobiec stworzono klasę, która zamienia typ zmiennoprzecinkowy na typ całkowity, w którym jako jednostkę przyjmujemy najmniejszą wartość. W praktyce operacja ta polega na podzieleniu wartości przez 100 – 1zł to 100 groszy, itd. Wejście i wyjście liczby zmiennoprzecinkowej znajdują się w przestrzeni `private` klasy, aby nie było możliwości ich modyfikacji. W klasie `fType` skorzystano z przyjaźni klas, bez której niemożliwy byłby dostęp do prywatnych składowych.

4. `Converter.cpp`

Plik źródłowy `converter.cpp` korzysta z utworzonego wcześniej nagłówka `converter.h`. Zostały wprowadzone do

niego wszystkie obsługiwane przez kalkulator walutowy kursy oraz mechanizm wyszukiwania symbolu podanej przez użytkownika waluty. Instrukcja `if` w funkcji `convertCurrency()` przeszukuje klasę `map`, w celu odnalezienia waluty. W tym pliku został również uwzględniony przypadek, w którym użytkownik chce konwertować walutę X na tę samą walutę X poprzez wprowadzenie kursu 1:1, co znacznie usprawnia funkcjonowanie programu.

5.Main.cpp


Główny plik programu – plik źródłowy `main.cpp` zbiera wszystkie wymienione powyżej funkcje i stanowi trzon kalkulatora walutowego. W tym miejscu, po wprowadzeniu przez użytkownika wszystkich potrzebnych informacji, program dokonuje przekonwertowania walut. W programie zaimplementowany został test wprowadzenia symboli – w przypadku wprowadzenia nieobsługiwanego symbolu następuje powrót do menu, z którego użytkownik może spróbować jeszcze raz wprowadzić interesujące go dane bez ponownego uruchamiania programu.

W pliku `main.cpp` zastosowano również zabezpieczenie, które pozwala użytkownikowi wprowadzać symbole walut zarówno wielkimi, jak i małymi literami, za co odpowiedzialna jest funkcja `transform()`. Przedwcześnie zapobiega ona ewentualnemu konfliktowi i od razu po wprowadzeniu przez użytkownika, zamienia symbole na małe litery.

Kolejnym elementem, który chroni działanie programu przed wpisaniem niepożądanych w danym momencie danych jest instrukcja warunkowa `if`, która w momencie wpisywania kwoty sprawdza, czy nie zostały podane litery bądź inne znaki. W przypadku wystąpienia tego błędu, użytkownik zostaje poinformowany o takiej sytuacji odpowiednim komunikatem, a program kasuje flagi błędu strumienia oraz

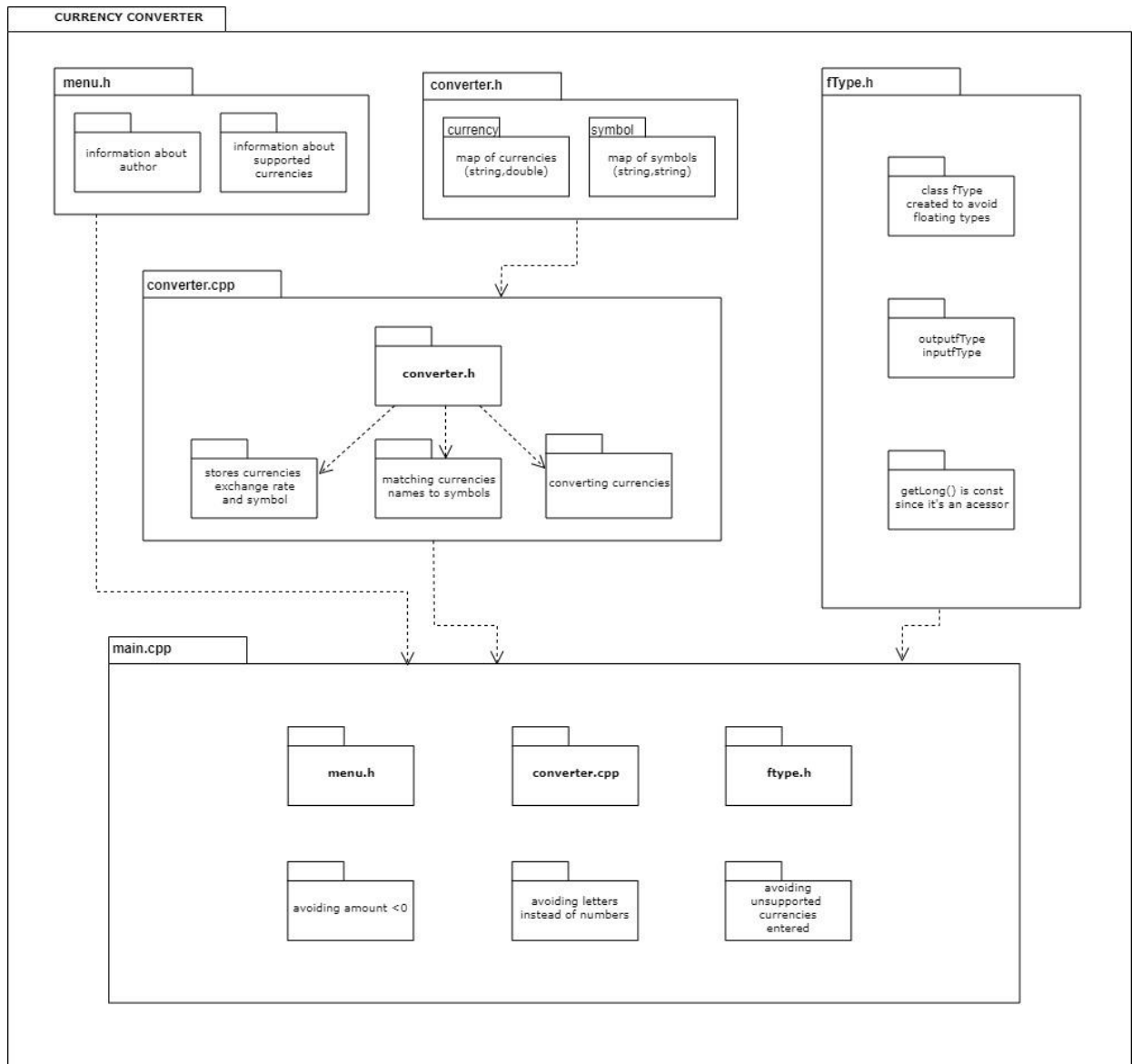
zbędne znaki z bufora wykorzystując funkcje `cin.clear()` oraz `cin.sync()` oraz wyłącza się.

Na samym końcu znajduje się zabezpieczenie przed wprowadzeniem ujemnej kwoty do programu. Nie jest niczym szczególnie odkrywczym stwierdzenie, że pieniądze są zawsze większe od zera, jednak program powinien być odpowiedni zabezpieczony na wypadek przeróżnych sytuacji. Dlatego też po raz kolejny skorzystano z instrukcji warunkowej `if`, dzięki której po ewentualnym wprowadzeniu ujemnej kwoty do kalkulatora, program nie wykona się.

 fType.h
class fType
private: long outputfType; double inputfType;
public: fType(long nfType = 0) long getLong() const
friend std::ostream
friend std::istream

Rys 3. Schematyczne przedstawienie klasy `fType`.

6. Graficzne przedstawienie kodu



Rys 4. Graficzne przedstawienie kodu w UML

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

Wdrażanie programu jest intuicyjne i proste. Po kompilacji i uruchomieniu wyświetla się ekran startowy (menu), w którym zawierają się informacje o autorze oraz lista obsługiwanych walut, z których użytkownik może skorzystać. Jest tam również menu wyboru – po naciśnięciu „1”, program od razu przechodzi do kalkulatora walut, natomiast po wciśnięciu „0” wyłącza się.

```
-----
----Currency Converter----
-----
01.2020 (C) Krzysztof Liszka
-----
1. USD - US dollar,
2. GBP - British pound,
3. CAD - Canadian dollar,
4. EUR - Euro,
5. PLN - Polish zloty,
6. AUD - Australian dollar,
7. CHF - Swiss franc,
8. RUB - Russian ruble,
9. CNY - Chinese youan renminbi,
10. AED - Emirati dirham.
-----
Press 1 to start currency converter
Press 0 to exit
```

Po uruchomieniu konwertera, program prosi użytkownika o podanie symbolu waluty, z której chce konwertować kwotę. Symbol ten może być wprowadzony do systemu zarówno małymi, jak i wielkimi literami, ponieważ program obsługuje wszystkie typy i nie stanowi to dla niego różnicy. Przy podawaniu kwoty, użytkownik może podać zarówno wartość w typie float, jak i int – program zapobiega gubieniu danych w pamięci, ponieważ nie wykorzystuje liczb zmiennoprzecinkowych.

```
-Enter currency to convert from (symbol XXX):
PLN
-Enter currency to convert to (symbol XXX):
usd
-Enter amount:
1000
Amount after exchange: USD 263.70
```

```
-Enter currency to convert from (symbol XXX):
PLn
-Enter currency to convert to (symbol XXX):
Gbp
-Enter amount:
100.000
Amount after exchange: GBP 20.21
```

Po wprowadzeniu odpowiednich symboli oraz kwoty, na ekranie wyświetla się przekonwertowana kwota, co widać na powyższych zrzutach z konsoli.

W przypadku wprowadzenia błędnego symbolu waluty, użytkownik zostanie natychmiastowo o tym poinformowany i wróci do menu.

```
-Enter currency to convert from (symbol XXX):  
pln  
-Enter currency to convert to (symbol XXX):  
lll  
Wrong currency typed! Returning...
```

W przypadku wprowadzenia błędnej kwoty, użytkownik zostanie natychmiastowo o tym poinformowany i wróci do menu.

```
-Enter currency to convert from (symbol XXX):  
pln  
-Enter currency to convert to (symbol XXX):  
usd  
-Enter amount:  
-199  
Entered amount is lower than 0! Try again.
```

W przypadku wprowadzenia liter bądź innych znaków w pole kwoty, użytkownik zostanie natychmiastowo o tym poinformowany, a program zakończy działanie.

```
-Enter currency to convert from (symbol XXX):  
PlN  
-Enter currency to convert to (symbol XXX):  
GBp  
-Enter amount:  
sss  
Error! You didn't enter an amount.
```