Sprawozdanie końcowe
Projekt indywidualny

Łukasz Knigawka

29 listopada 2018

Spis treści

1 Wprowadzenie **2**

2 Efekty działania programu **2**

3 Zmiany względem specyfikacji **20**
3.1 Zmiany względem specyfikacji funkcjonalnej 20
3.2 Zmiany względem specyfikacji implementacyjnej 21

4 Wnioski **22**

1 Wprowadzenie

Dokument opisuje działanie programu w momencie zakończenia pracy nad projektem. Porównano założenia projektowe z faktycznym sposobem zachowania programu. Dokonano krytycznej analizy realizacji wymagań a także przedstawiono krótką retrospekcję. Zakładane funkcje programu przedstawiono w specyfikacji funkcjonalnej. Szkic realizacji problemu przedstawiono w specyfikacji implementacyjnej.

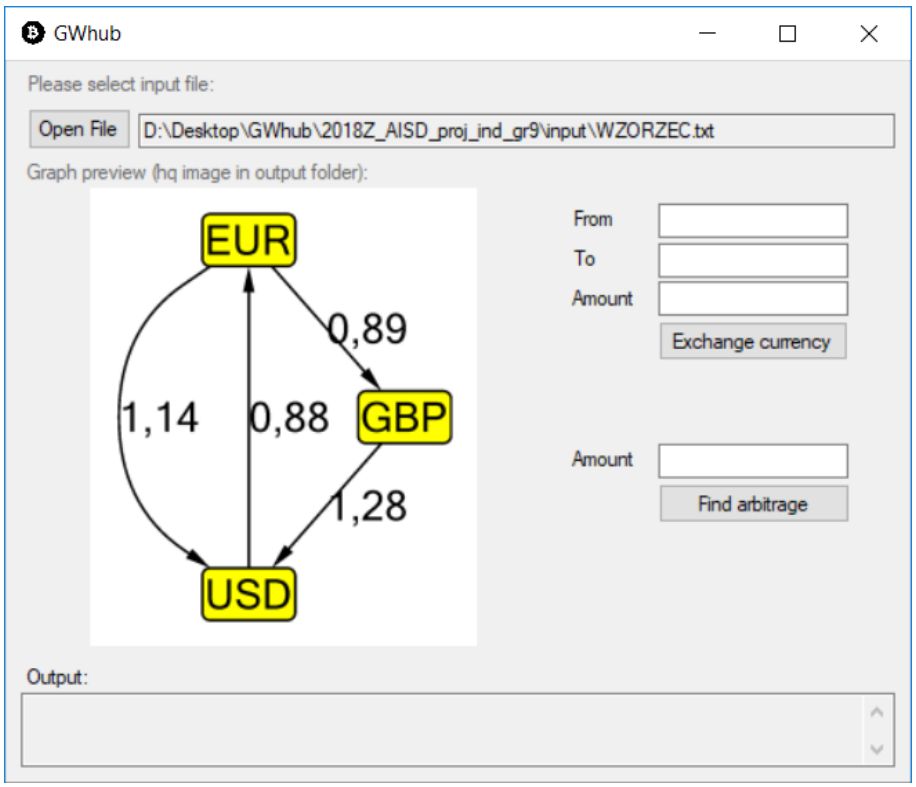
2 Efekty działania programu

W tej sekcji przyjrzymy się zarówno zachowaniu programu w przypadku podania przez użytkownika bezbłędnych danych wejściowych, na podstawie których da się wyświetlić ścieżkę wynikową, a także reakcji programu na otrzymanie danych, na podstawie których niemożliwe jest wyznaczenie ścieżki wynikowej. Sposób edycji danych wejściowych zaprezentowany w specyfikacji funkcjonalnej nie różni się poza jedną, lecz dość istotną różnicą: dane liczbowe - czy to kursy wymiany, czy to wartości opłat - należy podawać stosując jako separator dziesiętny bezwzględnie przecinek. Omówmy więc sposób zachowania programu dla różnych danych wejściowych.

Załóżmy dane wejściowe jak poniżej. Należy zwrócić uwagę, iż linia, w której znajduje się drugi komentarz, czyli druga linia, która rozpoczyna się od znaku #, kończy się dopiero po słowie *opłata*. W podanym przykładzie jedynym powodem, dla którego można byłoby odnieść inne wrażenie, jest fakt, że szerokość strony dokumentu jest ograniczona.

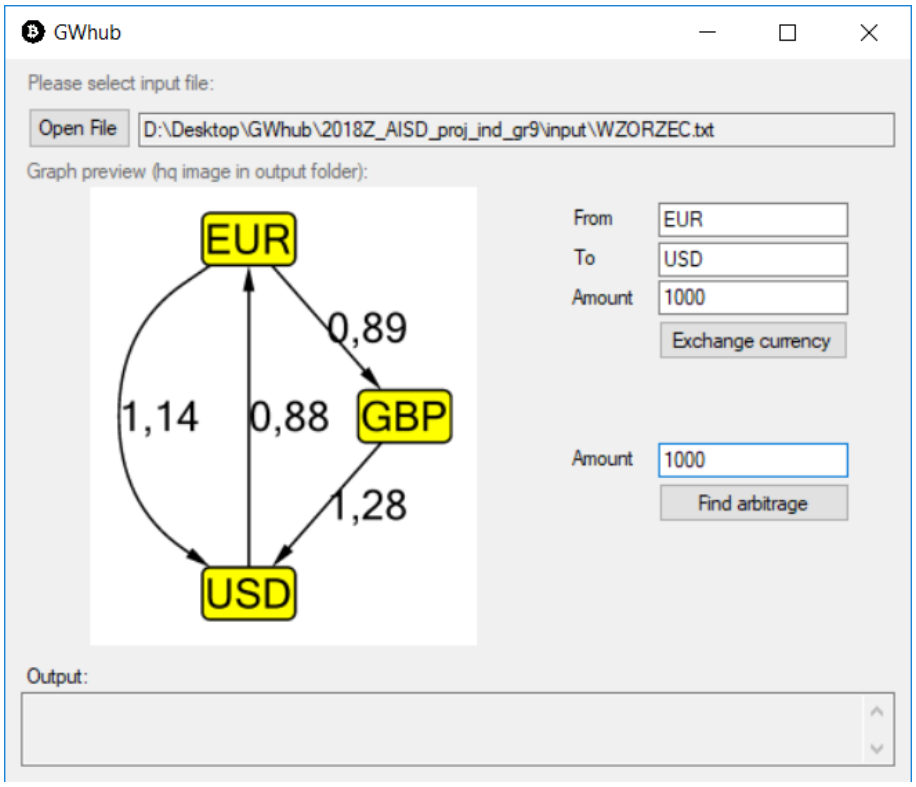
 # Waluty (id | symbol | pełna nazwa)
0 EUR euro
1 GBP funty brytyjski
2 USD dolar amerykański
 # Kursy walut (id | symbol (waluta wejściowa) | symbol (waluta wyjściowa) | kurs | typ opłaty | opłata
0 EUR GBP 0,8889 PROC 0,0001
1 GBP USD 1,2795 PROC 0
2 EUR USD 1,137 STAŁA 0,025
3 USD EUR 0,8795 STAŁA 0,01

W przypadku takich danych program stwierdzi, iż są one poprawne - zostanie utworzony odpowiadający tym danym obraz. Użytkownik, po wybraniu pliku tekstowego (z zawartością taką jak powyższy przykład) poprzez kliknięcie przycisku *Open File* oraz wybranie odpowiedniego pliku w eksploratorze, otrzyma okno programu takie jak przedstawia poniższy rysunek.



Rysunek 1: Okno główne programu po wprowadzeniu poprawnych danych wejściowych

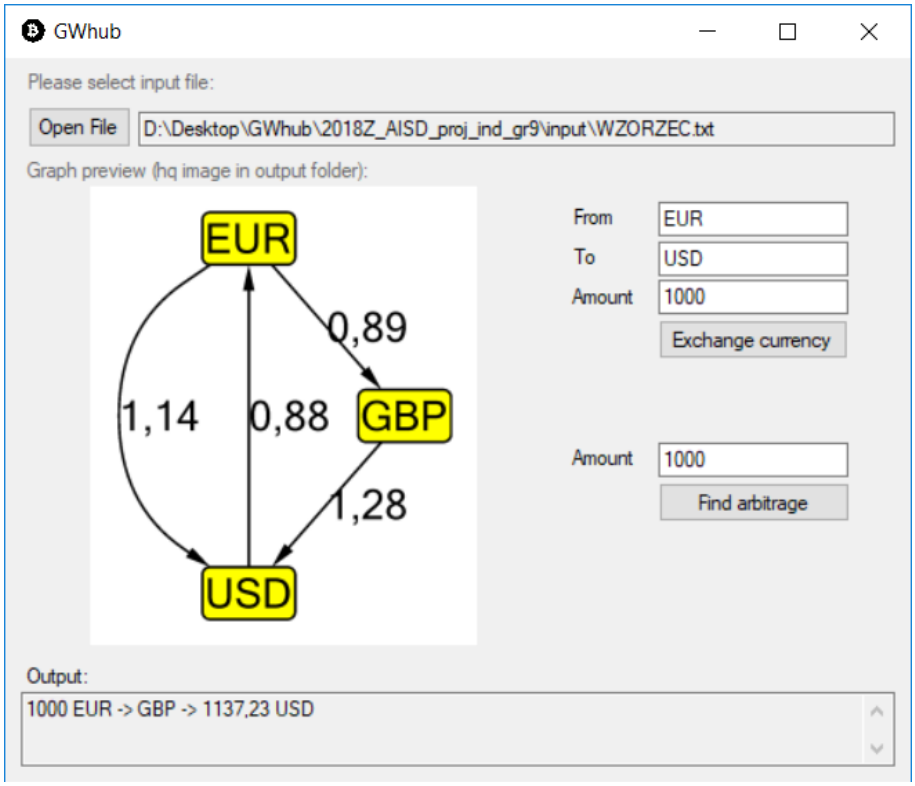
Spróbujmy więc dla otrzymanego grafu odnaleźć dowolny arbitraż a także kilka najbardziej opłacalnych wymian. Załóżmy wpisanie takich danych jak na poniższym rysunku.



Rysunek 2: Okno główne programu po wprowadzeniu walut początkowej i końcowej, a także wartości pieniężnych dla wymiany i arbitrażu

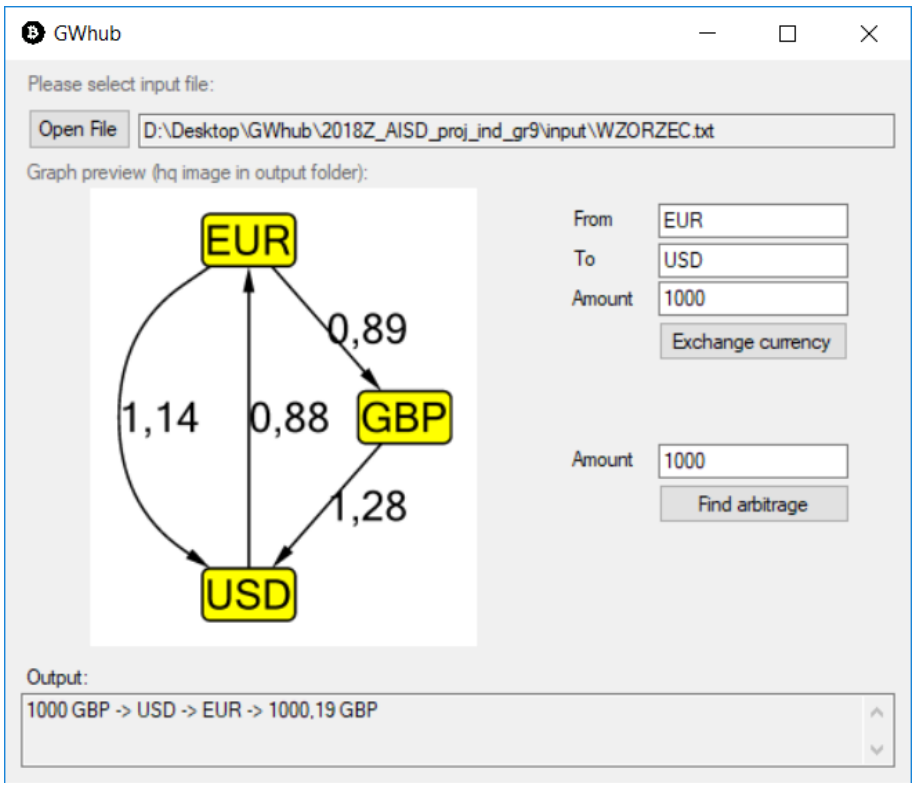
Dla takich danych, uda się odnaleźć ścieżkę pomiędzy walutą oznaczoną symbolem *EUR*,

a także walutą oznaczoną symbolem *USD*. Co można sprawdzić poprzez test manualny, czyli eufemistyczne wyrażenie na wpisanie ciągu znaków do kalkulatora, korzystniejszą wymianą, po uwzględnieniu opłat, jest wymiana w pierwszej kolejności waluty oznaczonej symbolem *EUR* na walutę oznaczoną symbolem *GBP*, a następnie wymiana tej waluty pośredniej na walutę oznaczoną symbolem *USD*. W tym przypadku, po kliknięciu przycisku z napisem *Exchange currency*, okno główne programu wyglądać będzie jak poniżej.



Rysunek 3: Okno główne programu po wybraniu opcji wyszukania najkorzystniejszej wymiany waluty

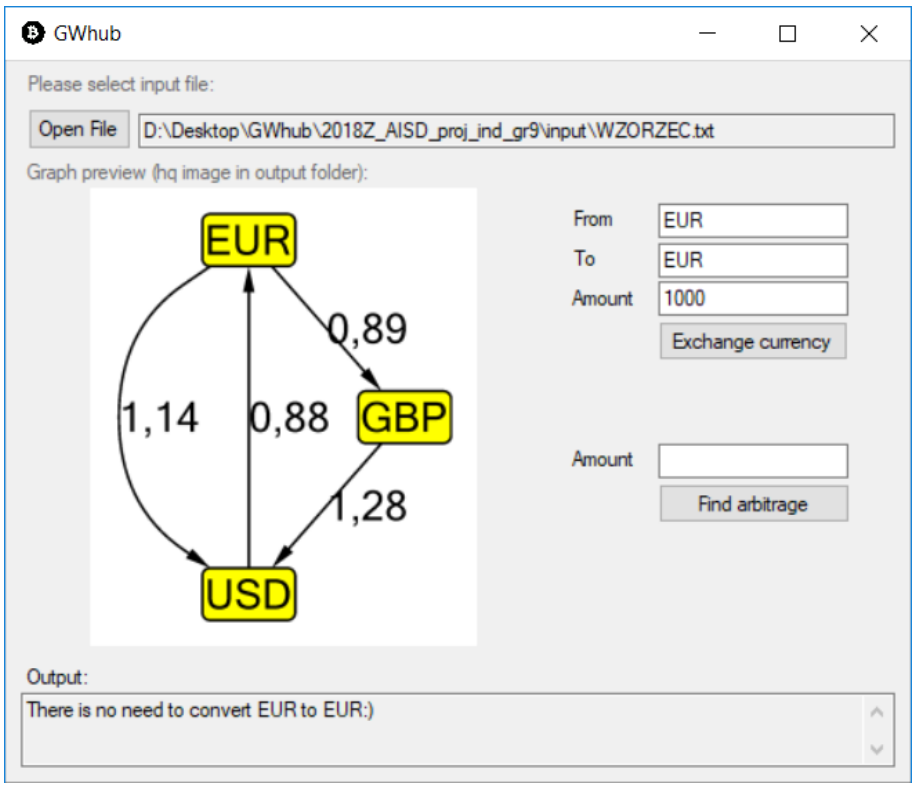
W przypadku wybrania opcji wykrywania arbitrażu, dla podanej wartości pieniężnej wykryta zostanie ścieżka arbitrażu przedstawiona poniżej.



Rysunek 4: Okno główne programu po wybraniu opcji wyszukania dowolnego arbitrażu

Zaprezentowane wyniki zostały porównane z wynikami otrzymanymi podczas wykonywania testów manualnych. Bardzo niewielkie różnice wynikały z zaokrąglania wyników podczas liczenia manualnego. Należy zauważyć, że mimo że wyniki są przedstawiane ostatecznie z dwoma miejscami po przecinku, wartości nie zostają ograniczane takim ograniczeniem. Zdecydowano się na takie rozwiązanie, gdyż nie odnaleziono w treści polecenia informacji o zaokrąglaniu wyników przy każdej zachodzącej wymianie. Zmniejsza to realizm przedstawionej sytuacji, ale zważywszy na fakt, że cała sytuacja nie jest realistyczna, takie rozwiązanie jest dopuszczalne.

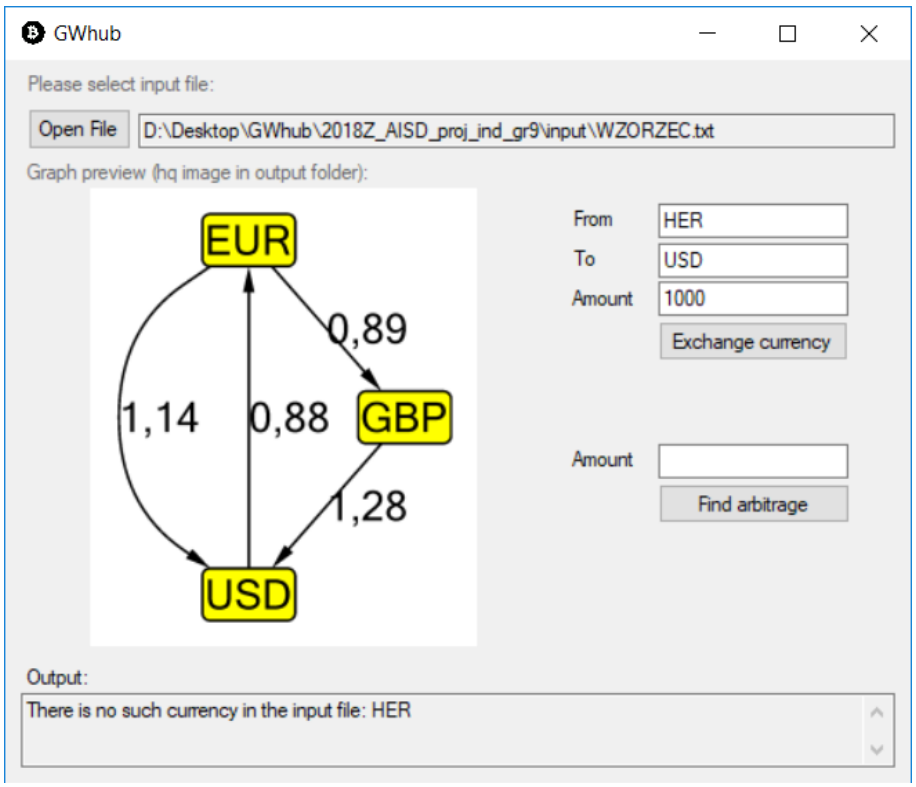
Spróbujmy teraz wprowadzić w polu waluty docelowej najkorzystniejszej wymiany symbol waluty podanej wcześniej jako waluta początkowa wymiany. Sprawdźmy, czy program został odpowiednio zabezpieczony na wypadek wystąpienia takiej sytuacji, tzn. co się stanie, gdy klikniemy przycisk opatrzony etykietą *Exchange currency*.



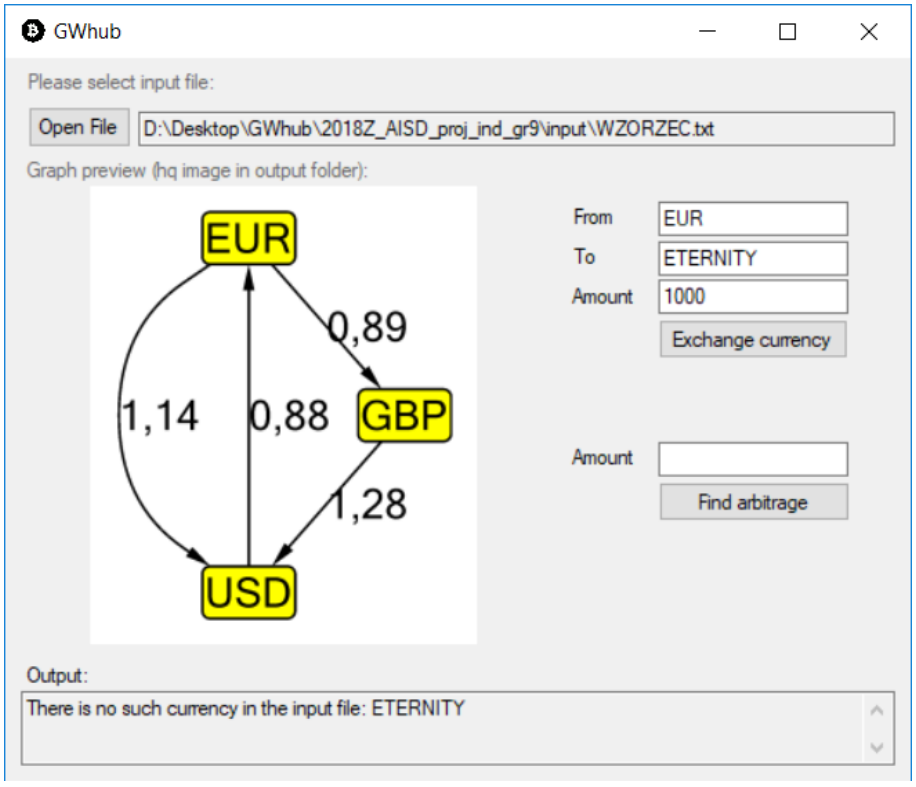
Rysunek 5: Okno główne programu po wprowadzeniu kontrowersyjnych danych

Ku naszej uciesze, program nie miał problemów z rozpatrzeniem tak niewygodnych danych. Bo i owszem - do poszukiwań arbitrażu lepiej sprawdza się przycisk *Find arbitrage*.

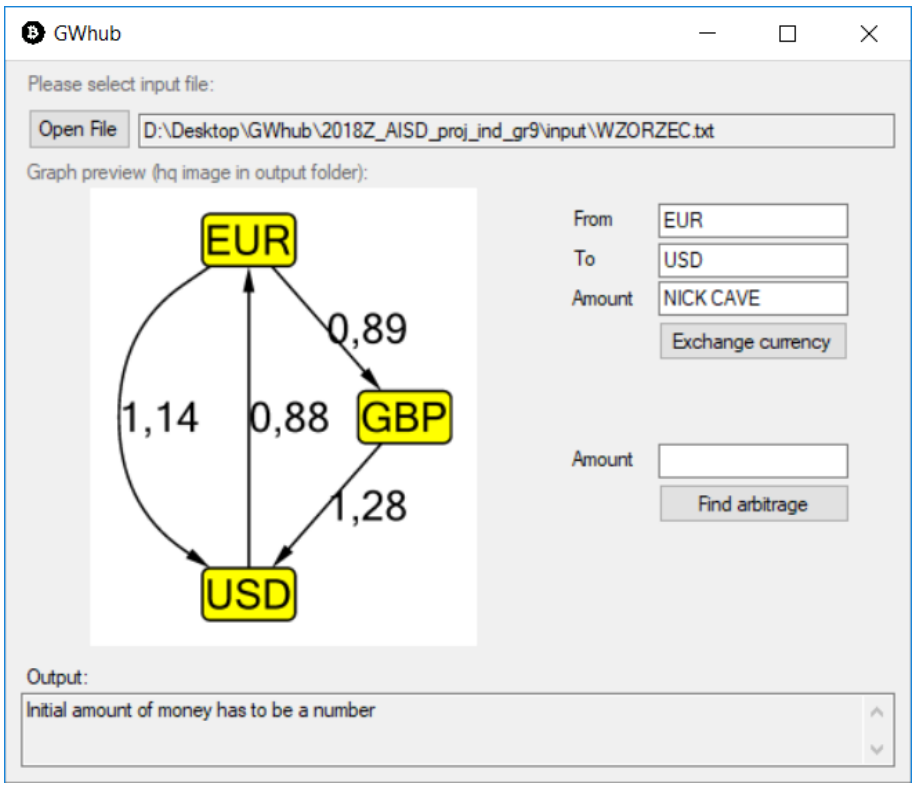
Sprawdźmy także, czy przypadkowe (lub nie) wystąpienie literówek w polach może zaszkodzić programowi. Poniżej przedstawiono serię rysunków przedstawiającą wprowadzone dane oraz reakcję programu.



Rysunek 6: Okno główne programu po wprowadzeniu nieistniejącej waluty jako waluta wejściowa



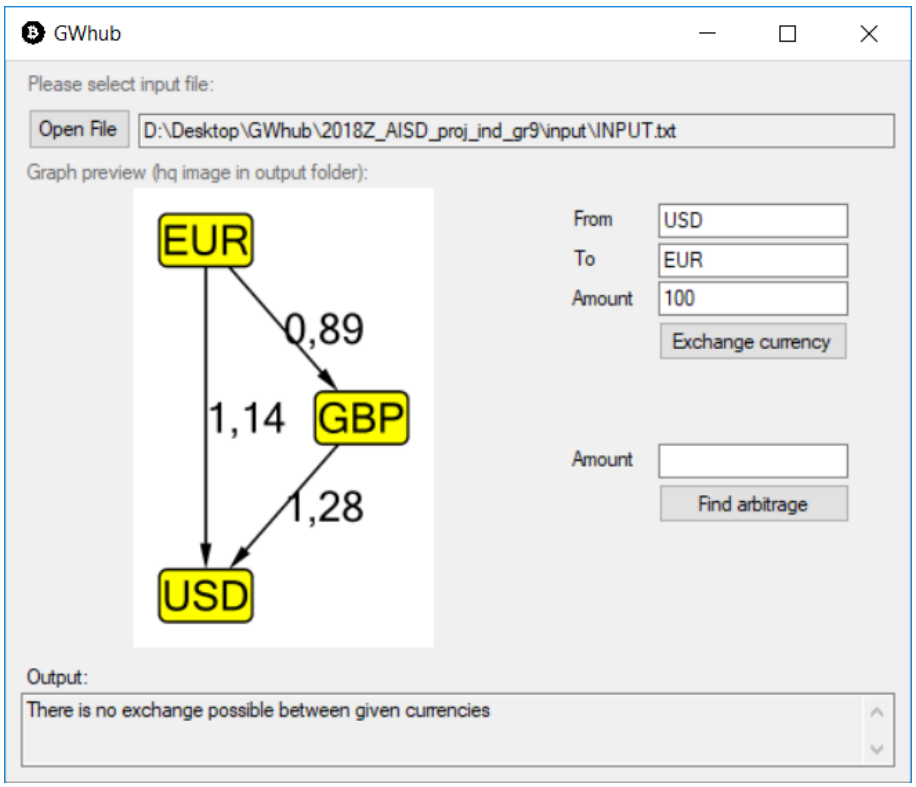
Rysunek 7: Okno główne programu po wprowadzeniu nieistniejącej waluty jako waluta wyjściowa



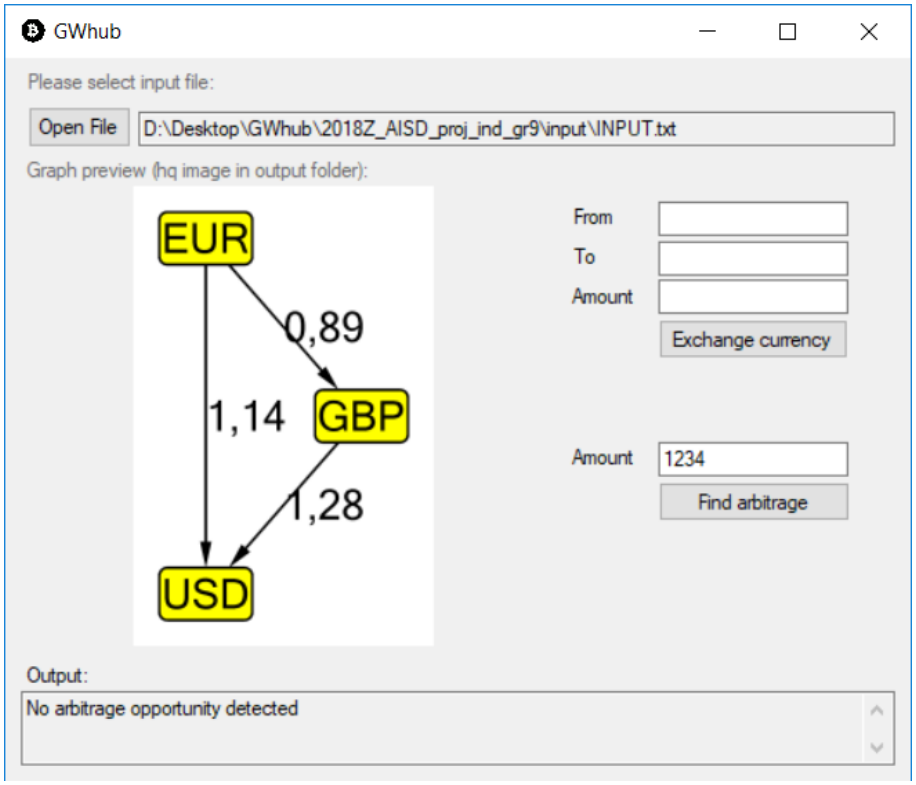
Rysunek 8: Okno główne programu po wprowadzeniu nieistniejącej waluty jako waluta wyjściowa

Posłużymy się także plikiem wejściowym o poniższej zawartości. Spróbujmy dokonać rzeczy niemożliwych.

 # Waluty (id | symbol | pełna nazwa)
0 EUR euro
1 GBP funty brytyjski
2 USD dolar amerykański
 # Kursy walut (id | symbol (waluta wejściowa) | symbol (waluta wyjściowa) | kurs | typ opłaty | opłata
0 EUR GBP 0,8889 PROC 0,0001
1 GBP USD 1,2795 PROC 0
2 EUR USD 1,137 STAŁA 0,025



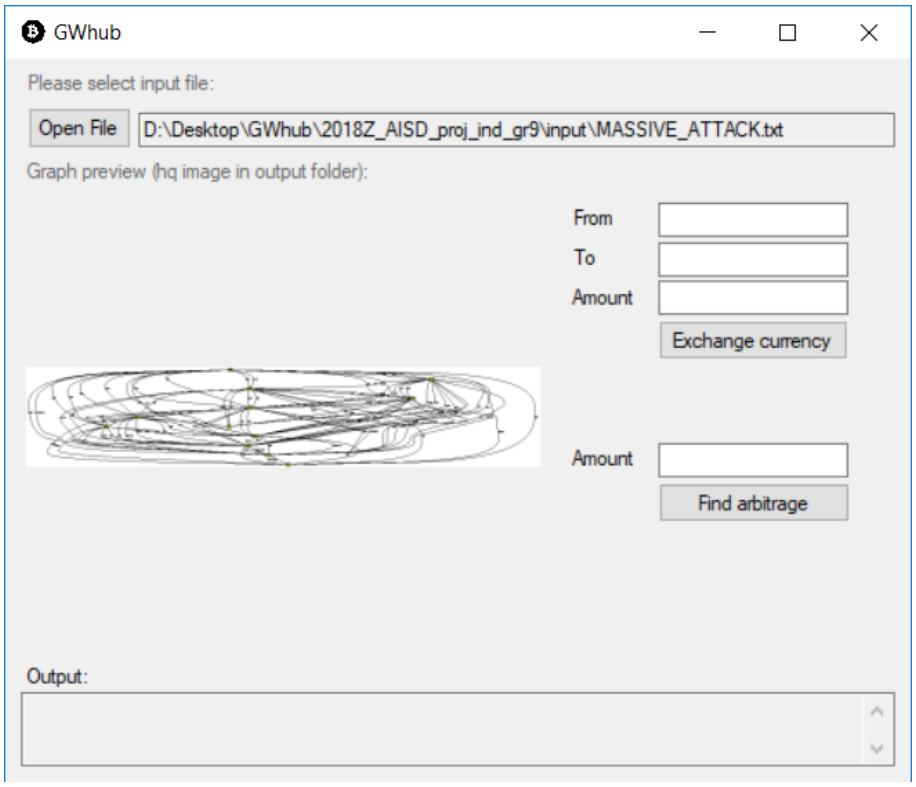
Rysunek 9: Okno główne programu po próbie wymiany między walutami, gdzie waluta docelowa nie jest osiągalna z waluty wejściowej



Rysunek 10: Okno główne programu po próbie wyszukania arbitrażu w grafie, w którym nie ma cykli

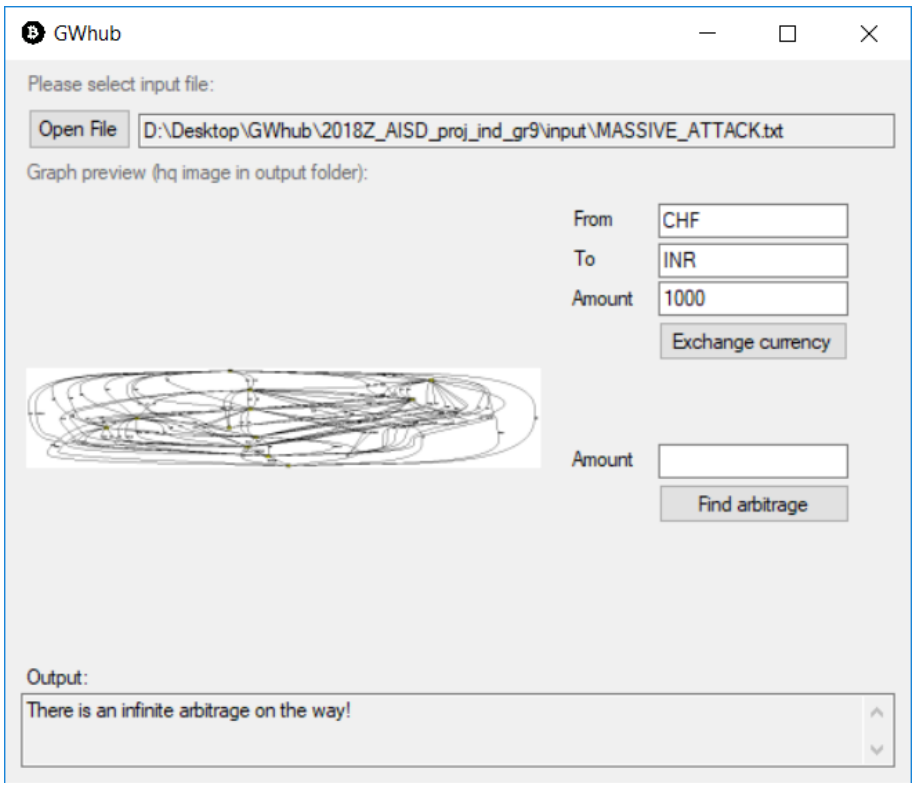
Rozważmy teraz wczytanie bardzo dużego grafu. Jego podgląd w oknie głównym programu jest bardzo nieczytelny, jednak bez problemu możemy graficzną reprezentację odnaleźć

w folderze z obrazami wynikowymi.



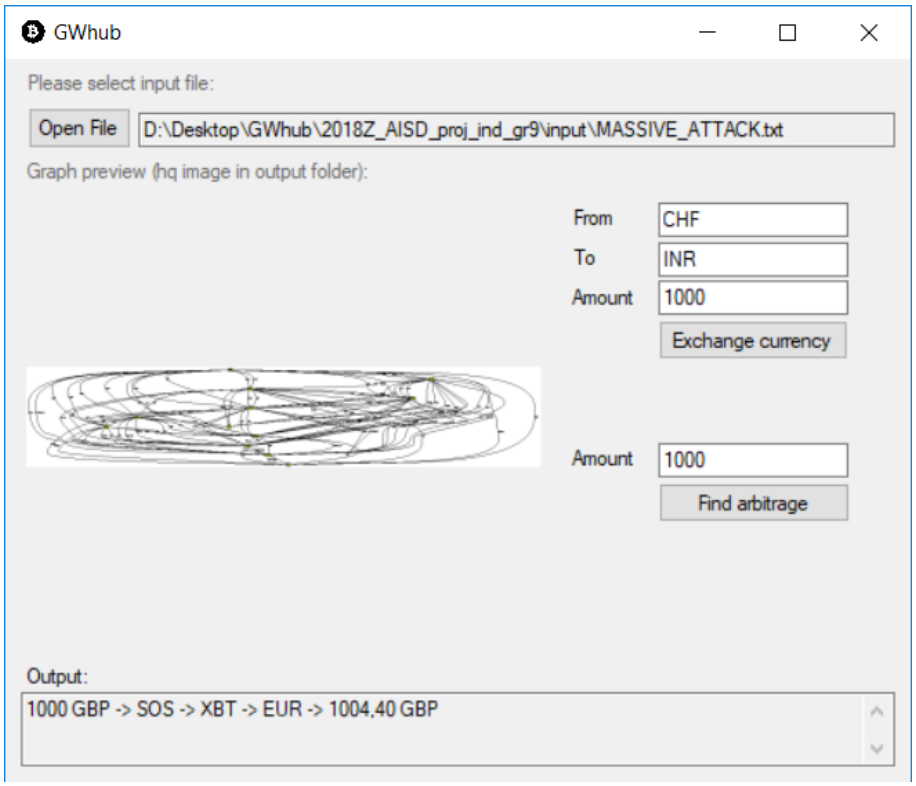
Rysunek 11: Wczytano duży graf, nie bez kozery nazwany *MASSIVE_ATTACK*

Przypadek tego grafu jest o tyle ciekawy, że w wielu sytuacjach chcąc odnaleźć najkorzystniejszą ścieżkę wymiany z danego wierzchołka do innego danego wierzchołka, po drodze napotykamy na możliwość wejścia w nieskończoną pętlę, przynoszącą nam zysk - znajdujemy po drodze arbitraż. W tej sytuacji, bez założenia jednokrotnego odwiedzania wierzchołków, którego w założeniach projektu nie odnaleziono, nie da się odnaleźć wartości pieniężnej w wierzchołku docelowym po dokonaniu najkorzystniejszej wymiany - gdyż zawsze da się znaleźć wymianę korzystniejszą.



Rysunek 12: Program jest zabezpieczony na takie sytuacje

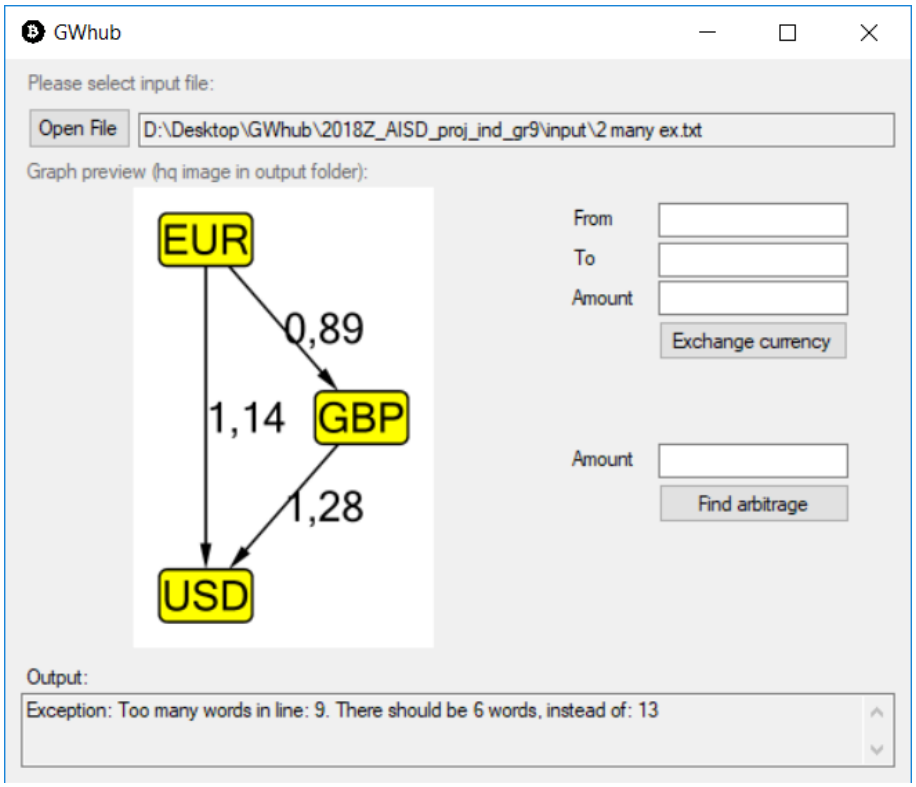
Spróbujmy więc znaleźć jeszcze w takim grafie arbitraż.



Rysunek 13: Jak można było odgadnąć, udało się wykryć arbitraż

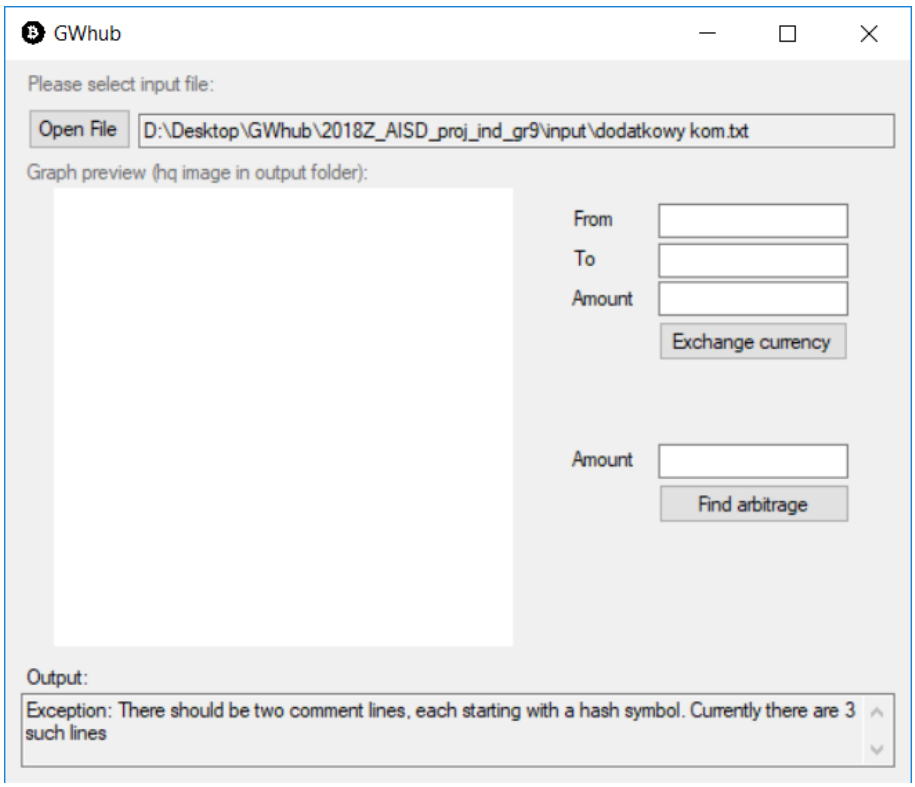
Pora przedstawić sposób reakcji programu na otrzymanie pliku wejściowego zawierającego błędy. Gdyby w pliku wejściowym, wśród linii opisujących kursy wymiany, znalazła się podana poniżej linia, program wypisałby na pole wynikowe informację o błędzie.

3 USD EUR 0,8795 STAŁA 0,01 ala ma kota i brata lukasza



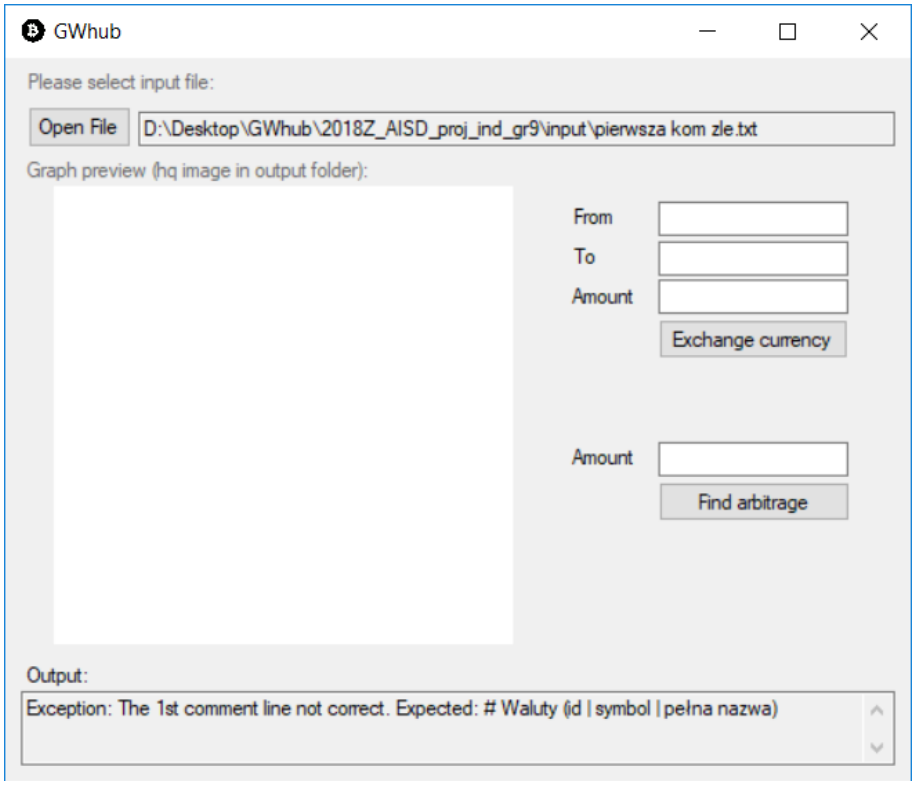
Rysunek 14: Oczywiście program wykrył obecność zbyt wielu słów w omawianej linii

Natomiast gdy zbyt wiele linii będzie zaczynało się od znaku symbolizującego początek linii zawierającej komentarz, błąd zostanie wyświetlony przed wczytaniem czegokolwiek z tego pliku.



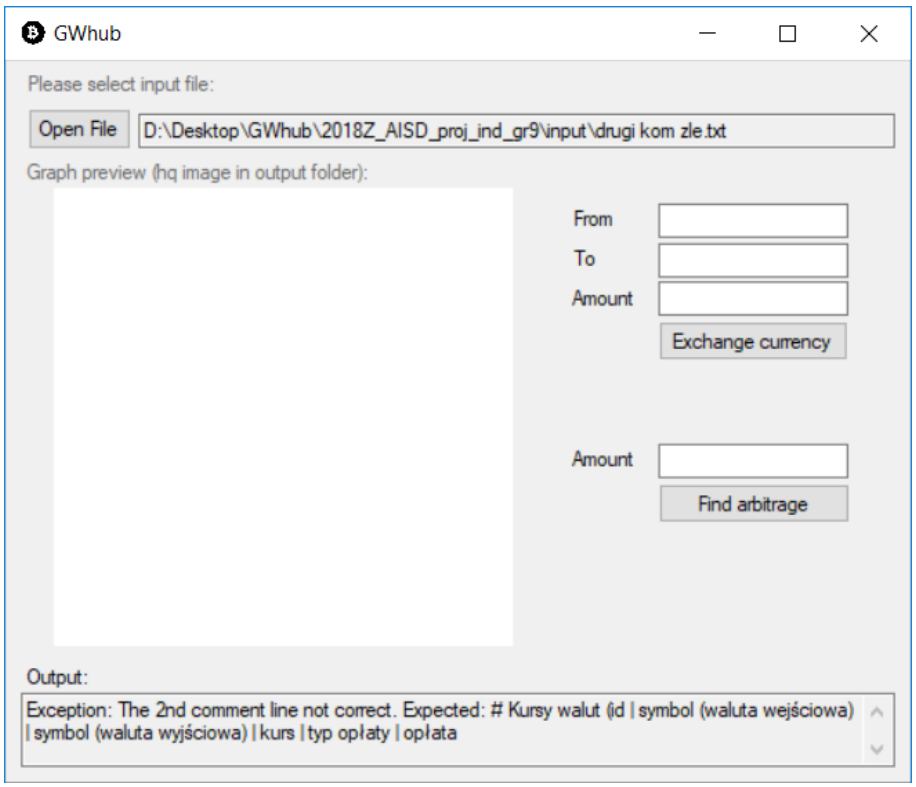
Rysunek 15: Dodatkowy komentarz

Zakładamy, że linie zawierające komentarze zawsze są takie same, co do znaku. Jeśli choćby jedna spacja nie będzie się zgadzać, wyświetlona zostanie informacja o błędnie sformatowanym pliku wejściowym.



Rysunek 16: Wadliwy pierwszy komentarz

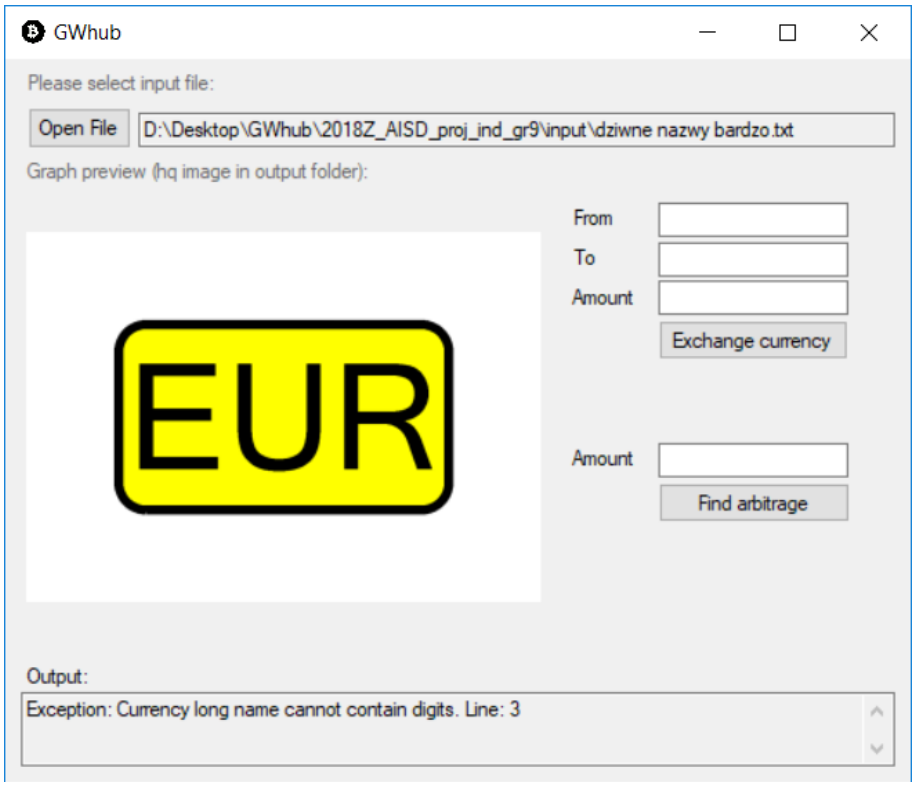
Analogicznie sytuacja wygląda przy drugiej linii zawierającej komentarz.



Rysunek 17: Wadliwy drugi komentarz

Nazwy pełne walut mogą być jedynie ciągami dowolnej długości, składającymi się z słów pisanych małymi literami. Poniższa linijka z pewnością łamie takie zasady.

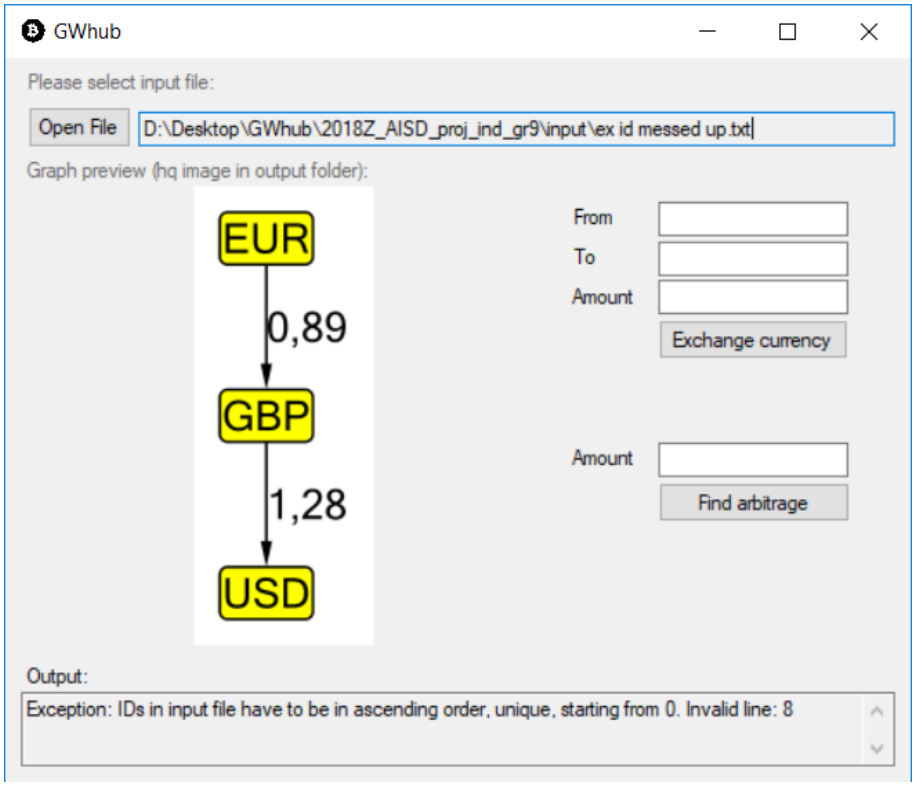
1 GBP funty 123



Rysunek 18: Wadliwa pełna nazwa waluty

Indeksy możliwych wymian muszą być wpisane w zachowanej kolejności, zaczynając od zera. W tym przypadku błędna była trzecia z przedstawionych poniżej linijek.

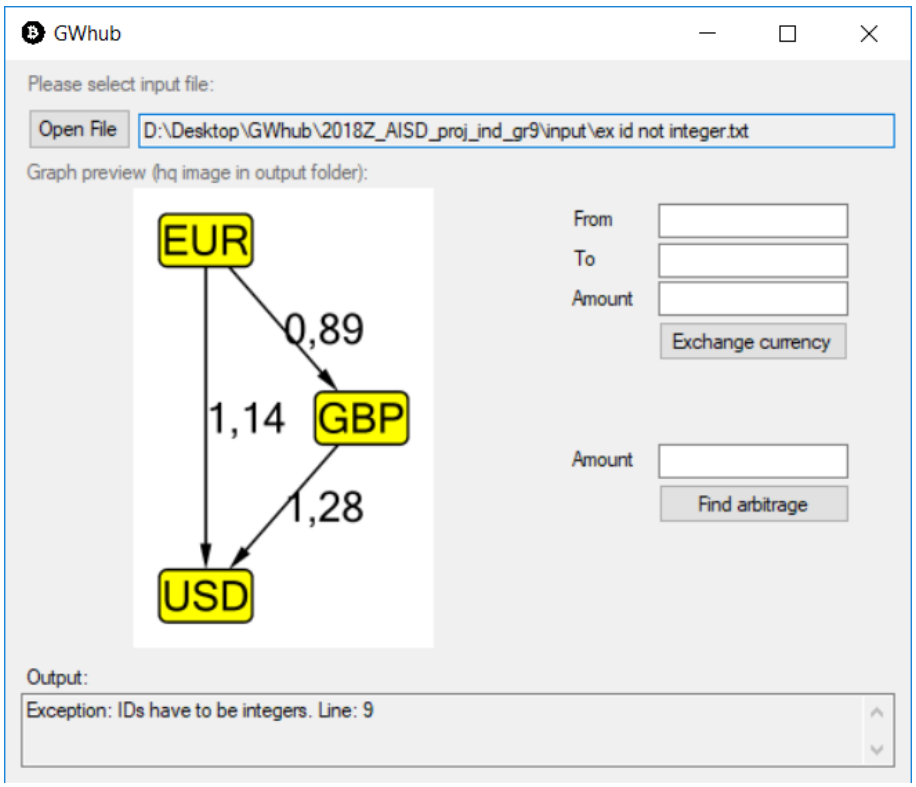
0 EUR GBP 0,8889 PROC 0,0001
1 GBP USD 1,2795 PROC 0
88 EUR USD 1,1370 STAŁA 0,025
3 USD EUR 0,8795 STAŁA 0,01



Rysunek 19: Wadliwa pełna nazwa waluty

Co oczywiste, indeks wymiany musi być liczbą całkowitą.

szalalamruta USD EUR 0,8795 STAŁA 0,01

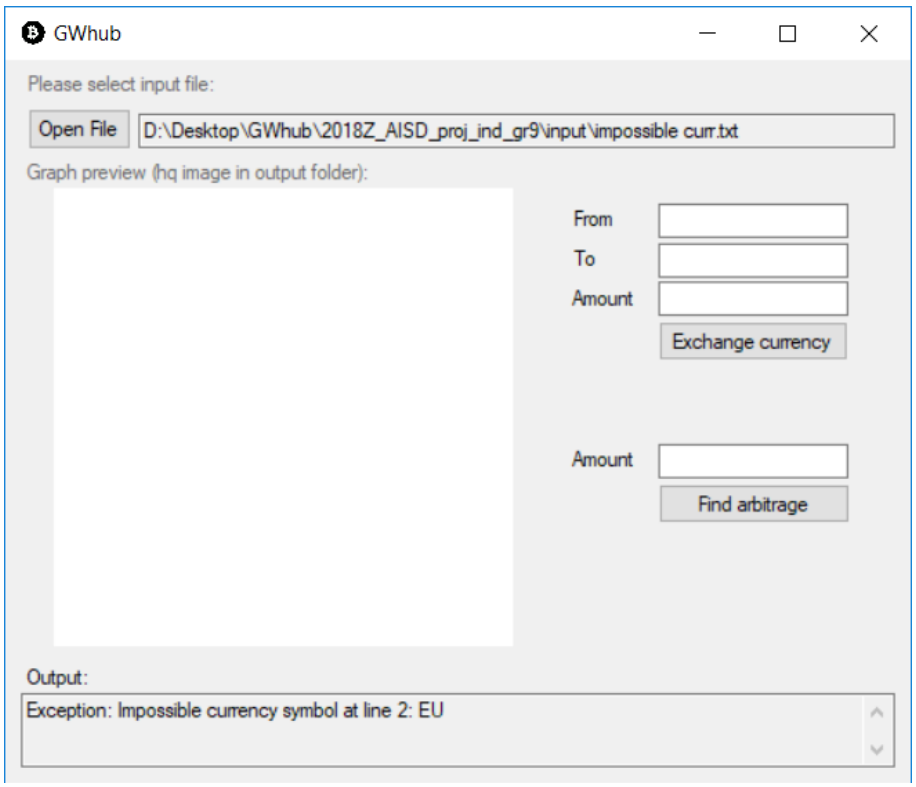


Rysunek 20: Indeks takiej wymiany nie spełniałby wymogów

Analogicznie rozwiązano problem nieuporządkowanych indeksów walut, a także indeksów walut nie będących liczbami całkowitymi.

W pliku z danymi wejściowymi, nie może pojawić się symbol waluty inny, niż składający się z trzech wielkich liter. Rozpatrzmy poniższą linię.

0 EU euro

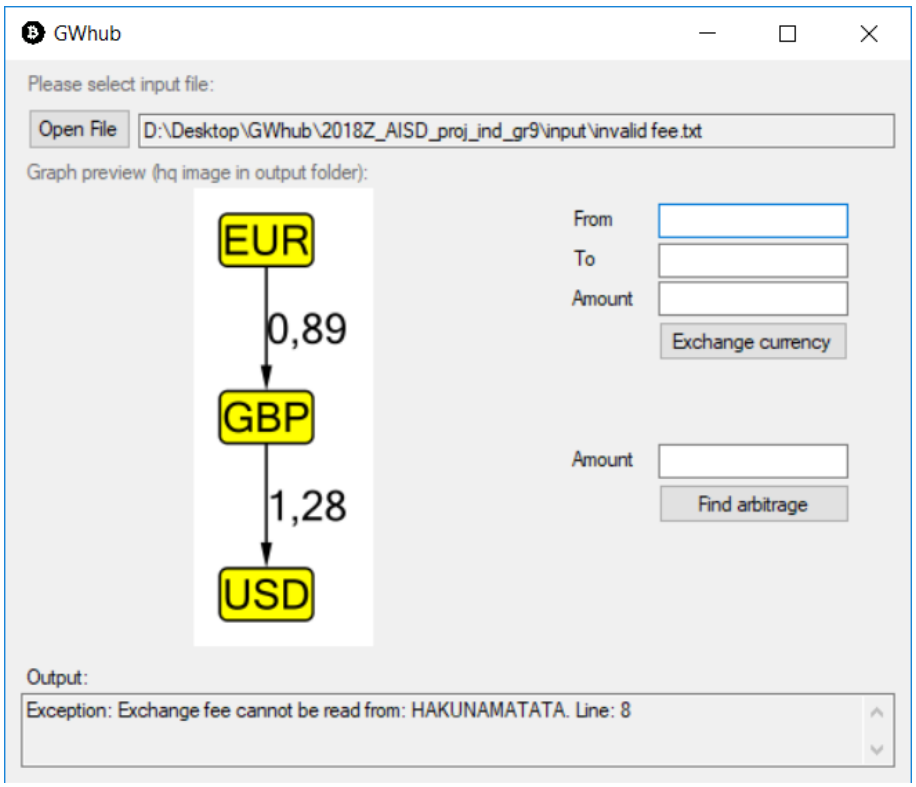


Rysunek 21: Niemożliwy symbol

Analogicznie, odpowiednia informacja o błędzie zostałaby wyświetlona, gdyby symbol był trzyliterowy, lecz litery byłyby małe.

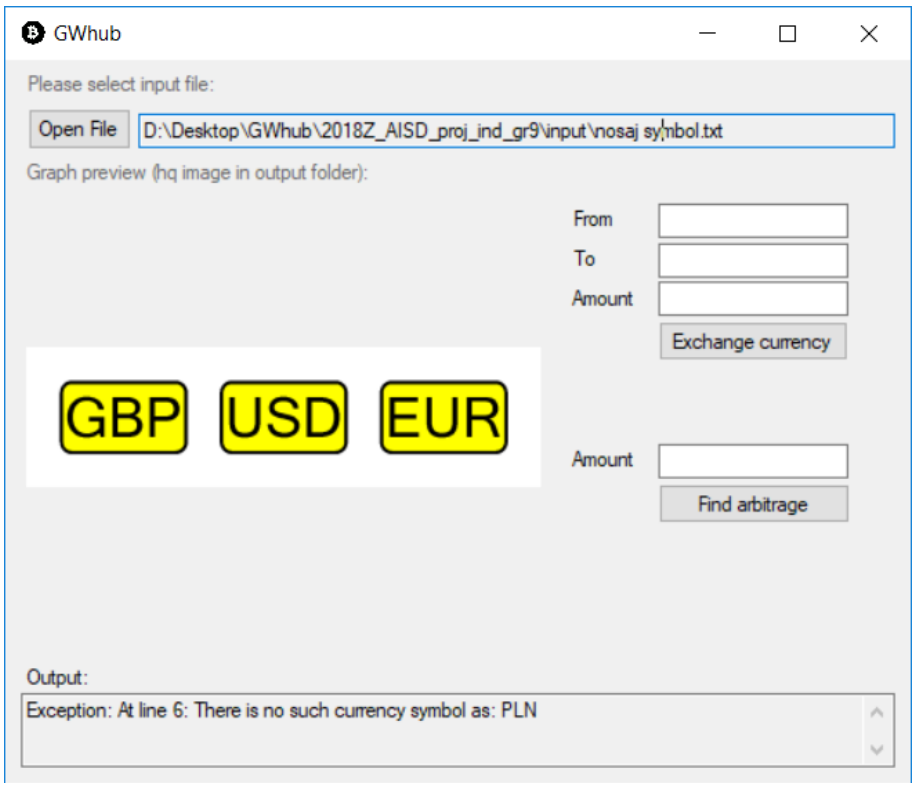
Odpowiednie błędy zostanę także ukazane, jeśli wartość kursu lub wartość opłaty nie będzie liczbą. Weźmy dla przykładu taką linię jak poniższa.

2 EUR USD 1,1370 STAŁA HAKUNAMATATA



Rysunek 22: W tym przypadku, mamy do czynienia z nierealną wartością opłaty stałej

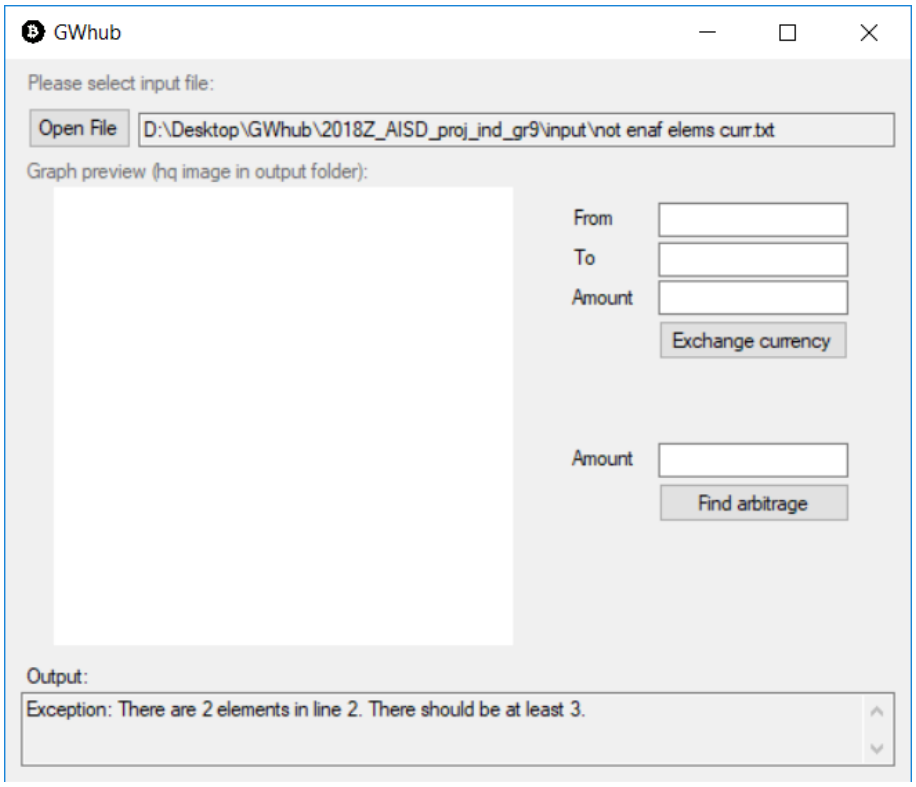
Jeśli podany symbol dla waluty początkowej lub końcowej danej wymiany nie został wcześniej dodany do symboli walutowych, wykryty zostaje błąd. Przykład reakcji programu poniżej.



Rysunek 23: W jednej z wymian użyto symbolu waluty, która nie została zdefiniowana

Jeśli w linii walutowej lub w linii wymiany liczba wyrazów będzie zbyt mała, program z

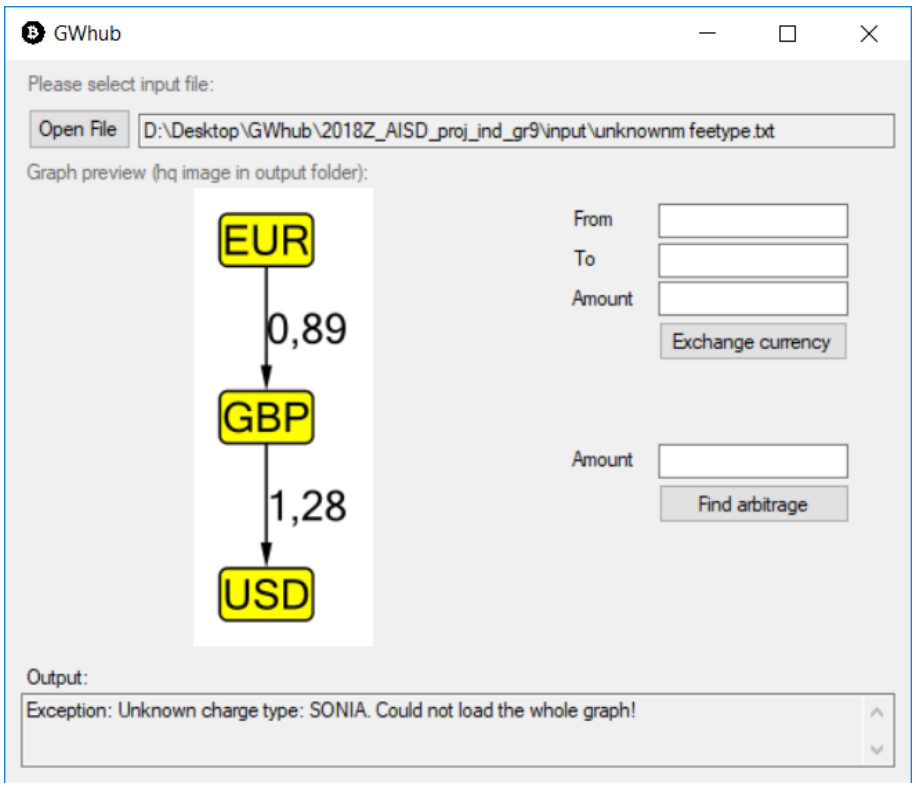
góry wykryje podstęp.



Rysunek 24: Zbyt mało elementów w linii

Została obsłużona także sytuacja, w której jako rodzaj opłaty podano słowo inne niż *PROC* lub *STAŁA*. Załóżmy poniższą linijkę.

2 EUR USD 1,1370 SONIA 0,025



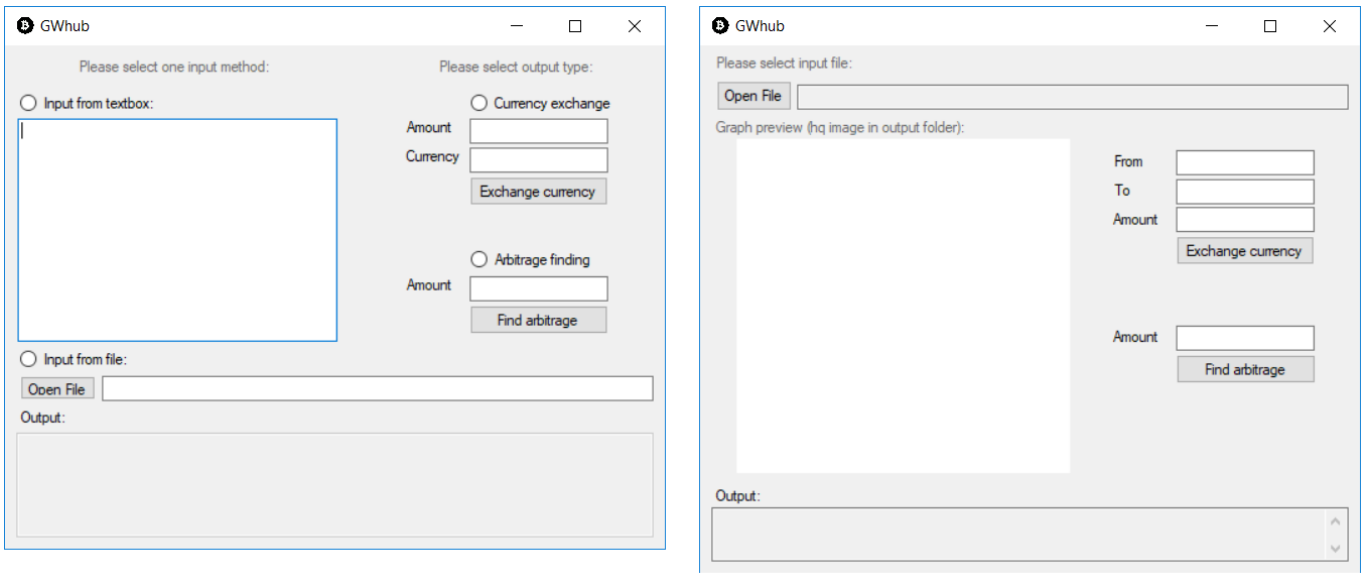
Rysunek 25: Program nie zna takiego rodzaju opłaty

3 Zmiany względem specyfikacji

3.1 Zmiany względem specyfikacji funkcjonalnej

Nie powstało wiele różnic pomiędzy założeniami specyfikacji funkcjonalnej a faktyczną realizacją projektu. Nieco inaczej wygląda okno główne programu, przy czym drobne zmiany były przewidywane, stąd też wygląd okna głównego w specyfikacji funkcjonalnej został przedstawiony jako prototyp. Na poniższym rysunku przedstawiono porównanie wersji okna. Najbardziej istotną zmianą jest zrezygnowanie z możliwości wprowadzania danych wejściowych poprzez pole tekstowe będące częścią okna głównego. W zamian w tym samym miejscu umieszczono pole, w którym wyświetlana może być graficzna reprezentacja wczytanego grafu.

Pozostałe zmiany dotyczące wyglądu graficznego interfejsu użytkownika programu są kosmetyczne. Dodano pole, w które należy wpisać walutę początkową przy wybraniu opcji konwersji walut. Pole, w którym pojawiają się dane wyjściowe, zostało wzbogacone o pasek boczny, który umożliwiłby odczytanie całego wyniku działania programu, gdyby ten nie zmieścił się w polu. Zrezygnowano z jakichkolwiek zsynchronizowanych przycisków (jak te przy opcji wymiany walut oraz opcji arbitrażu w prototypie okna głównego), gdyż zastosowanie takiego rozwiązania wydaje się być bezzasadne. Poszczególne komponenty mogły zostać przemieszczone w obrębie okna.



Rysunek 26: Porównanie okna głównego programu w wersji prototypowej oraz ostatecznej

Zaszły liczne zmiany w stosunku do *specyfikacji funkcjonalnej*, jeśli chodzi o sposób obsługi sytuacji wyjątkowych. Jakiekolwiek komunikaty o błędach, czy to związanych z wadliwym plikiem wejściowym, czy też z niepoprawnymi wartościami wpisywanymi w pola graficznego interfejsu, zdecydowano się wyświetlać w tym samym polu, w którym wyświetlane są dane wynikowe. Wszystkie przewidziane w *specyfikacji funkcjonalnej* sytuacje wyjątkowe udało się obsłużyć, generując inne komunikaty, lecz tożsame w sensie znaczeniowym. Dodano więcej przewidzianych sytuacji wyjątkowych, a wszystkie zostały krótko omówione na przykładzie w sekcji *Efekty działania programu*.

3.2 Zmiany względem specyfikacji implementacyjnej

Zdecydowanie więcej różnic niż pomiędzy wersją ostateczną projektu, a założeniami specyfikacji funkcjonalnej, można odnaleźć pomiędzy ostateczną wersją programu, a założeniami specyfikacji implementacyjnej.

Klasę *Visionary*, która pierwotnie miała zawierać metodę odnajdującą arbitraż, oraz metodę odnajdującą najkorzystniejszą ścieżkę pomiędzy danymi walutami, zdecydowano się podzielić na dwie klasy, obarczone węższym zakresem odpowiedzialności. Idąc tym tropem, klasa *Arbitrage* umożliwia odnalezienie ścieżki arbitrażu, a klasa *Best Exchange* wspiera wyszukiwanie najkorzystniejszej ścieżki wymiany. Obie te klasy posiadają swoje metody do wygenerowania napisu będącego efektem wyszukiwania - sklejane są symbole kolejnych walut, a także na początku dodawana jest wartość pieniężna w walucie początkowej, a na końcu wartość pieniężna w walucie końcowej.

W klasie *Parser* zrezygnowano z wykorzystania typu wyliczeniowego przy rozróżnianiu rodzaju danej linii. Ze względu na nieskomplikowaną strukturę plików wejściowych, zdecydowano się problem rodzaju linii rozwiązać znacznie prościej, wykorzystują fakt, iż linie pomiędzy pierwszym a drugim komentarzem to linie przedstawiające waluty, a linie po drugim komentarzu przedstawiają połączenia między tymi walutami - kursy wymiany. Sama metoda wczytująca dane wejściowe w pierwotnej wersji nie zwracała danych, natomiast w wersji ostatecznej zwraca obiekt typu *Digraph*. Obiekt ten jest reprezentacją grafu skierowanego.

Wspomniana powyżej klasa *Digraph* jest odpowiednikiem klasy *Graph* ze specyfikacji implementacyjnej. Pierwotna nazwa klasy była identyczna jak nazwa klasy reprezentującej graf

w bibliotece wspomagające rysowanie grafu, stąd też zmiana. W celu umożliwienia rysowania grafu, do klasy reprezentującej graf skierowany w omawianym projekcie zostały dodane metody: *Draw*, *AddEdges*, *AddNodes* oraz *SaveGraphAsImg*. Dzięki zaimportowaniu *Microsoft.Msagl.Drawing* oraz *Microsoft.Msagl.GraphViewerGdi*, tworzenie graficznych reprezentacji grafów nie zajęło ani dużo czasu, ani nie zajmuje wielu linijek w kodzie.

4 Wnioski

1. Dodanie do programu interfejsu graficznego można uznać za udany eksperyment. Z pewnością takie rozwiązanie dla wielu użytkowników końcowych byłoby wygodniejsze niż konieczność posługiwania się oknem konsoli.

2. Przygotowanie specyfikacji funkcjonalnej oraz specyfikacji implementacyjnej przed właściwym tworzeniem kodu aplikacji znacznie usprawniło proces pisania kodu.

3. Pomimo wielu nieznacznych różnic obecnej implementacji z założeniami specyfikacji implementacyjnej, udało się podtrzymać wszystkie główne idee z obu specyfikacji.

4. Kosztem stosunkowo niewielkiego wysiłku program można rozszerzyć o funkcję zaznaczania na podglądzie grafu (tym samym także na obrazach znajdujących się w folderze z obrazami wynikowymi) wierzchołków lub krawędzi obecnych w wynikowej ścieżce - czy to symbolizującej wierzchołki arbitrażu, czy to kolejne wierzchołki biorące udział w wymianie.

5. Dzięki dodaniu funkcji graficznego przedstawienia danych wejściowych można szybko upewnić się, czy nie popełniło się błędu przy wprowadzaniu danych

6. Konieczność stosowania się do bardzo ścisłych reguł przy tworzeniu pliku wejściowego, między innymi konieczność nadawania walutom pełnych nazw zawierających jedynie małe litery, może wprowadzić w złość użytkownika, który dopiero zaczął pracę z programem. Jednak po jego dłuższym użytkowaniu, taki porządek można uznać za zaletę, pliki wejściowe są bardzo czytelne.

7. Algorytm *Bellmana-Forda* pozwolił uporać się z problemem, jednak absolutnie nie jest jedynym rozwiązaniem. Możliwe byłoby otrzymanie takich samych rezultatów przy zastosowaniu *przeszukiwania wszerz* bądź *przeszukiwania w głąb*. W wielu sytuacjach korzystniejsze pod względem złożoności czasowej byłoby zapewne zastosowanie wersji algorytmu *Bellmana-Forda* nieco bliższej programowaniu dynamicznemu.

8. Program można bardzo łatwo rozbudować o możliwość wpisywania danych wejściowych bezpośrednio do pola będącego częścią okna głównego programu. Ułatwiłoby to pracę z programem użytkownikom posługującym się jednym monitorem.

9. W przypadku, gdy wprowadzanych walut i/lub kursów jest wiele, podgląd grafu dostępny z poziomu okna głównego programu staje się bardzo nieczytelny. W ramach polepszania doświadczeń użytkownika z programem można dość łatwo zaimplementować możliwość przybliżania i oddalania wygenerowanego obrazu, a także poruszania się po nim dzięki poziomemu i pionowemu paskowi. Jednak podczas gdy wszystkie wygenerowane obrazy i tak trafiają do specjalnego folderu, takie rozszerzenie wydaje się być mało istotne. Niemniej jednak, ułatwiłoby ono pracę z programem użytkownikom posługującym się jednym monitorem.

10. Chęć napisania szybkiego prototypu programu, który miał być sukcesywnie usprawniany, wpłynęła negatywnie na jakość generowanego kodu. Założenie, że dokonamy w przyszłości refaktoryzacji niskiej jakości kodu jest bardzo ryzykowne - należało od samego początku przykładać dużą wagę do jakości kodu.

11. Pisanie testów jednostkowych do programów z graficznym interfejsem użytkownika nie jest łatwe, a taki interfejs użytkownika często ułatwia przeprowadzanie testów manualnych.

12. Rzeczywisty czas działania programu nie jest prosty do zmierzenia, lecz w praktyce jakiekolwiek operacje na realnych zbiorach danych (tzn. liczba walut nie przekracza liczby walut dostępnych obecnie na świecie) nie zajmują dłużej niż 1 sekunda. Znaczna część czasu działania programu przypada na wygenerowanie obrazu i podmienienie obecnie prezentowanego obrazu w graficznym interfejsie na nowy obraz.