Gliwice, 11.06.2014

Politechnika Śląska w Gliwicach

Programowanie Komputerów 4

**Klient FTP**

Autor: Prowadzący przedmiot:

Jakub Bujny dr inż. Roman Starosolski

Grupa 3, sekcja 1

## Temat

Zadaniem programu jest komunikacja z serwerem za pośrednictwem protokołu FTP, wysyłanie, pobieranie oraz usuwanie plików, poruszanie się po strukturze katalogów oraz tworzenie nowych folderów.

## Analiza tematu

Program udostępnia użytkownikowi podstawowe operacje związane z komunikacją z serwerem FTP. Za pośrednictwem graficznego interfejsu możemy uzyskać dostęp do danych znajdujących się zarówno na lokalnym dysku jak i na serwerze. Program umożliwia swobodne poruszanie się po strukturze katalogów, pobieranie i wysyłanie danych.

Graficzny interfejs został stworzony, tak aby być jak najbardziej intuicyjny. Zdecydowałem się wybrać bibliotekę GTK+ ponieważ:

* Nigdy wcześnie jeszcze nie pracowałem z tą biblioteką a uważam, że należy poszerzać horyzonty
* Jest to biblioteka, która została stworzona z myślą o Linuxie więc chciałem przetestować jej wersję Windowsową
* W Internecie udostępniona jest dość dobra dokumentacja opisująca działanie biblioteki
* Zawiera wszystkie potrzebne komponenty do budowy programu objętego moim tematem

Do obsługi protokołu FTP wybrałem bibliotekę libcurl ponieważ:

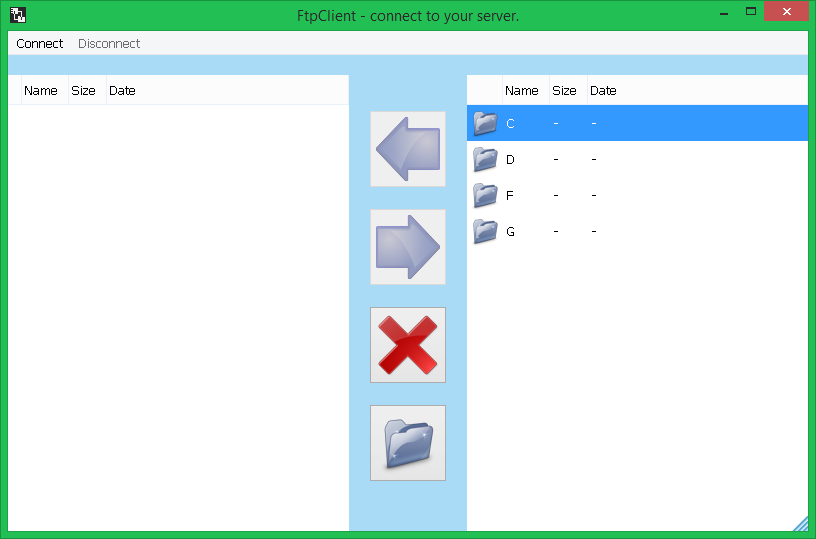
* Jest to bardzo dojrzała biblioteka wywodząca się z języka C
* Jej obsługa jest bardzo intuicyjna
* W Internecie można znaleźć wiele przykładów, które znacznie ułatwiają naukę

Do operacji na dysku lokalnym wybrałem bibliotekę winapi ponieważ już jej kiedyś wcześniej używałem.

Do odczytu polecenia LIST użyty został prosty parser ftpparse.

## Specyfikacja zewnętrzna

Główne okno programu



Przycisk

połączenia

Przycisk

wysyłania pliku

Przycisk

ściągania

pliku

Przycisk

Usuwania

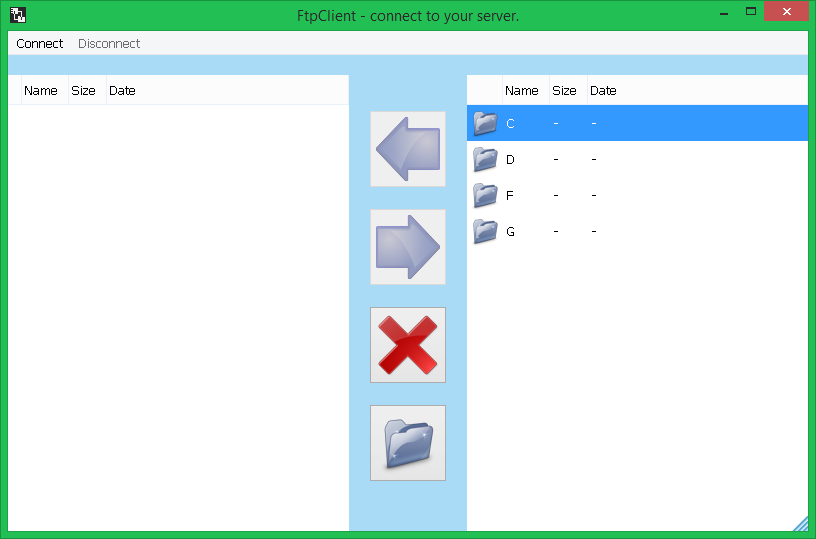
Pliku/folderu

Przycisk

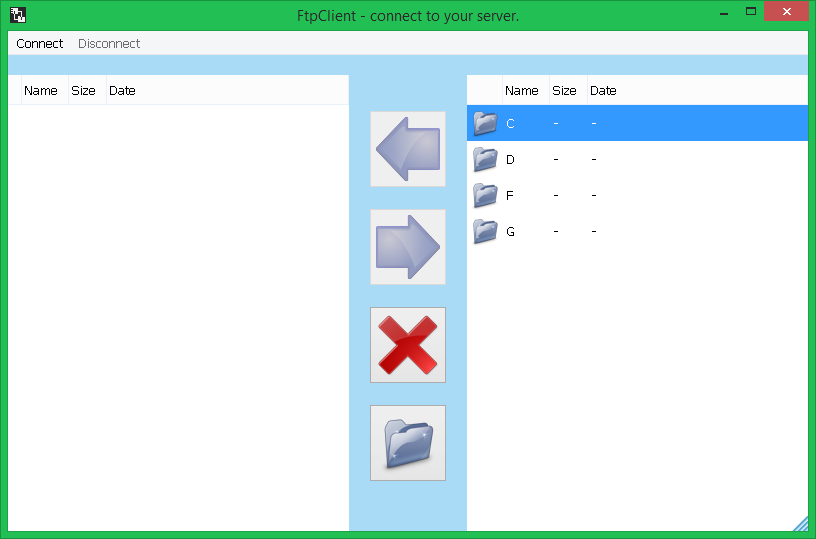
tworzenia folderu

Lista plików /folderów na serwerze

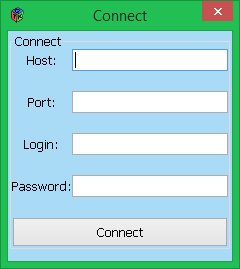
Lista plików /folderów lokalnych



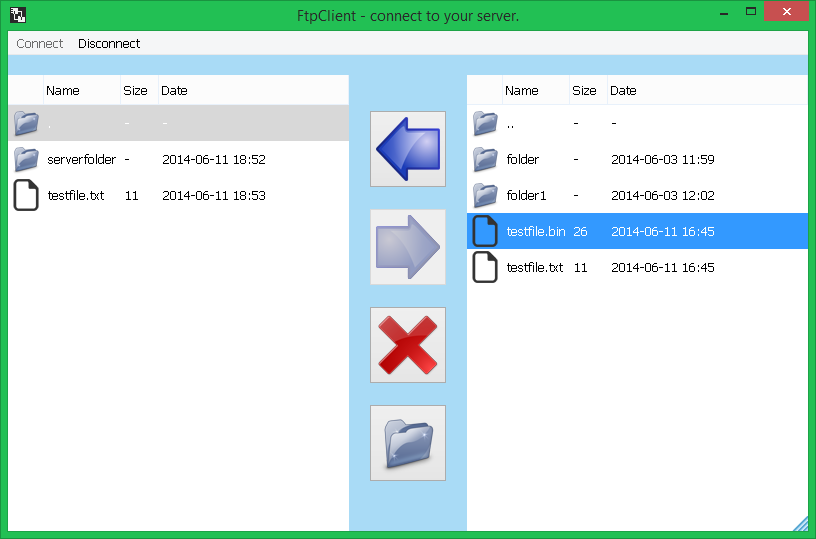
Aby połączyć się z serwerem w celu przesłania/ściągnięcia plików należy kliknąć przycisk Connect.



Pokaże się wtedy następujące okno, w którym należy wypełnić dane potrzebne do połączenia a następnie kliknąć przycisk Connect:



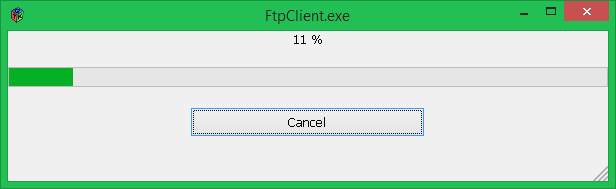
Przykład połączenia:



**Podstawowe akcje:**

* Aby poruszać się po strukturze katalogów należy dwa razy kliknąć folder.
* Aby wysłać plik należy zaznaczyć go na liście lokalnej i kliknąć strzałkę w lewo.
* Aby pobrać plik należy zaznaczyć go na liście serwerowej i kliknąć strzałkę w prawo.
* Aby usunąć plik/folder należy zaznaczyć go na liście i kliknąć przycisk z krzyżykiem.
* Aby stworzy folder należy zaznaczyć plik/folder na jednej z list (tam gdzie ma być utworzony folder) a następnie wpisać jego nazwę.

Podczas pobierania pliku możemy obserwować postęp lub anulować akcję:



## Specyfikacja wewnętrzna

### Opis klas

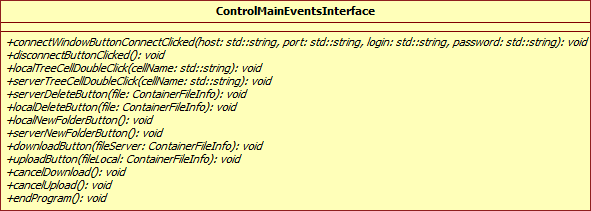
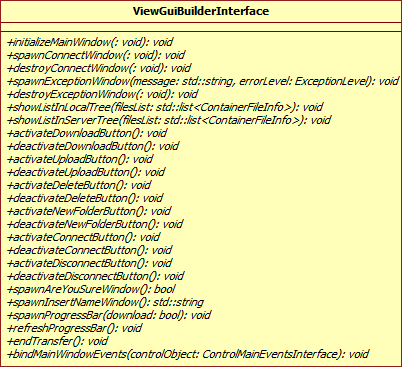
* ControlMain: główna klasa odpowiedzialna za sterowanie programem. Zawiera w sobie metody do obsługi zdarzeń udostępnione w interfejsie ControlMainEventsInterface. Za pośrednictwem wstrzykiwania zależności zawiera w sobie obiekty odpowiedzialne za dostęp do danych i budowanie graficznego interfejsu – dostępne są za pomocą interfejsów. Gdy złapie jakiś wyjątek kieruje go do obiektu ControlExceptionManager, który podejmuje stosowne działania.
* ViewGuiBuilder: największa klasa w całym programie odpowiedzialna za obsługę biblioteki GTK+. Zawiera w sobie metody do budowania graficznego interfejsu: tworzenie okien, tworzenie przycisków, tworzenie paska postępu, aktualizacja paska postępu. Swoje metody udostępnia za pomocą interfejsu ViewGuiBuilderInterface. Warto zwrócić uwagę na obsługę zdarzeń. Ta klasa zawiera odpowiednie, statyczne metody do obsługi zdarzeń – statyczność tych metod wymagana jest przez bibliotekę. Gdy zdarzenie pojawi się, wewnątrz statycznej metody następuje odwołanie do interfejsu ControlMainEventsInterface gdzie następuje przekazanie sterowania do klasy ControlMain.
* ModelDAO: klasa odpowiedzialna za dostęp do danych na dysku lokalnym i na serwerze. Swoje metody udostępnia za pomocą interfejsu ModelDAOInterface. Zawiera w sobie metody potrzebne do przeglądania struktury katalogów na dysku, tworzenia nowych katalogów i usuwania plików/katalogów. Posiada w sobie obiekt klasy ModelConnection, który odpowiedzialny jest za wszystkie operacje na danych na serwerze – od połączenia do wysyłania/pobierania plików. W tej klasie pojawia się sporo wyjątków opakowanych w kontener ContainerException.

### Istotne struktury danych

Struktura plików i katalogów, którą uda się odczytać z dysku lokalnego lub serwera przechowywana jest w liście dwukierunkowej. Informacje o każdym pliku/folderze opakowane są w specjalny kontener ContainerFileInfo. W tym kontenerze znajdują się informacje m.in. o nazwie, ścieżce, dacie, rozmiarze. W obiekcie dostępu do danych pobierana jest struktura plików/katalogów a następnie jest przerabiana na listę dwukierunkową zawierającą kontenery, która następnie wysyłana jest do wyższych warstw – najpierw do warstwy kontroli a następnie do warstwy GUI. W warstwie GUI następuje transformacja listy dwukierunkowej do struktury wymaganej przez bibliotekę graficzną i wreszcie następuje wyświetlenie żądanych danych.

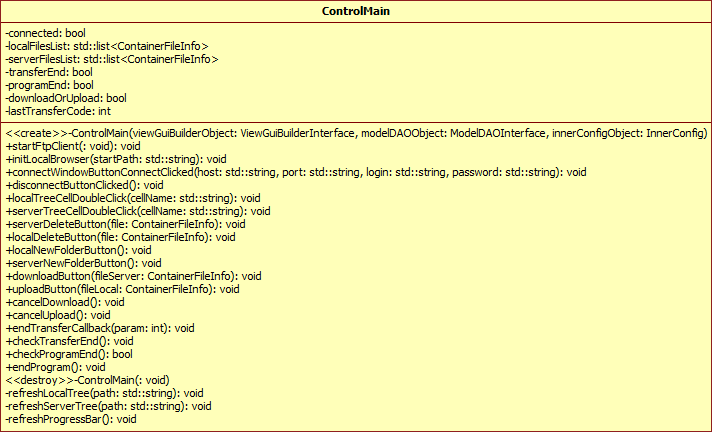
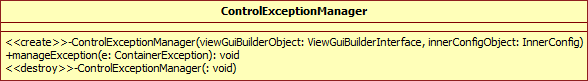
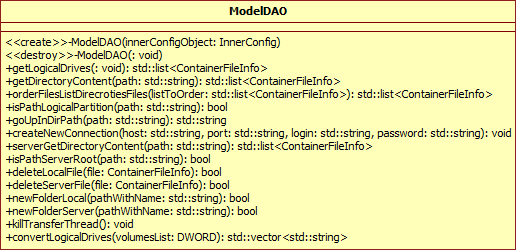
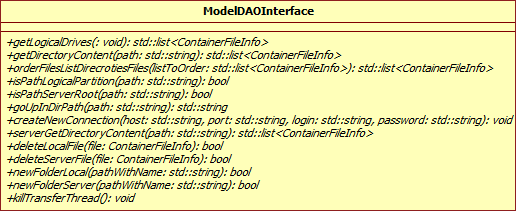
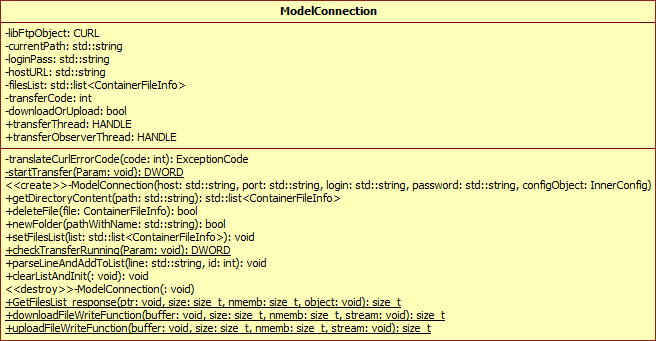
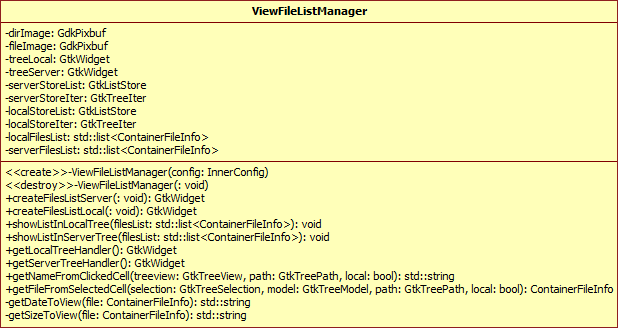
### Diagram klas:

Diagram klas z uwagi na swój rozmiar znajduje się na następnej stronie.



C:\Users\Jakubek\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Untitled-1.png

C:\Users\Jakubek\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Untitled-1.png



### Wykorzystane techniki programowania obiektowego

1. Dziedziczenie
2. Polimorfizm
3. Wyjątki
4. Strumienie

## Testowanie i uruchamianie

Podczas testów wykrytych i usuniętych została wiele błędów.

|  |  |  |
| --- | --- | --- |
| Lp. | Opis błędu | Rozwiązanie |
| 1. | Zawieszanie się okna głównego podczas wywoływania paska postępu transferu. | Program potrzebuje oddzielnych wątków do pobierania i rysowania interfejsu więc jeden wątek aktualizuje pole atomowe zawierające postęp transferu a drugi wątek pobiera tą wartość i rysuje interfejs |
| 2. | Brak plików na serwerze w katalogu głównym uniemożliwiał zaznaczenie jakiegokolwiek pliku w celu utworzenia katalogu | Dodano plik-fantom o nazwie „.” do katalogu root |
| 3. | Problem z zarządzaniem listą dwukierunkową w momencie zmiany katalogu (usuwanie starej listy i tworzenie nowej) | Całość operacji została przerzucona do klasy odpowiedzialnej za dostęp do danych. |
| 4. | Brak jednolitego standardu zwracania odpowiedzi od serwera za pomocą polecenia LIST | Dołączenie specjalnej biblioteki znalezionej w Internecie, która wykonuje parsowanie danych do jednolitego standardu. |
| 5. | Problem z wyciekami pamięci związany z nieprawidłowym zarządzaniem destruktorami | W destruktorach klas zaimplementowano usuwanie dynamicznie tworzonych obiektów |
| 6. | Zbyt mały maksymalny rozmiar zmiennej przechowującej rozmiar plików w bajtach – następowało przepełnienie | Użycie typu uinit64\_t |
| 7. | Problem ze zbyt dużym użyciem procesora podczas wykonywania programu | Użycie specjalnej metody biblioteki graficznej, która zawiesza wykonywanie głównej pętli programu aż nie nastąpi jakieś zdarzenie wykonywane przez użytkownika. |

Liczba błędów była znacznie większa (wymieniłem te najważniejsze) i wiązała się głównie z nieumiejętnym stosowaniem biblioteki graficznej co wynikało z tego, że był to mój pierwszy kontakt z nią.

## Wnioski

1. Stosując programowanie obiektowe można dużo łatwiej zarządzać swoim projektem. Każdy obiekt stanowi oddzielną logikę i oddzielny byt dzięki czemu testowanie staje się dużo łatwiejsze a program przyjmuje charakter modułowy. Klasy można postrzegać jako swego rodzaju klocki przy pomocy których składamy program w całość.
2. Posługiwanie się protokołem FTP przy pomocy odpowiedniej biblioteki nie jest takie trudne o ile uważnie czytamy dokumentacje.
3. Podczas tworzenia programu należy na bieżąco zwracać uwagę na usuwanie elementów, które są tworzone dynamicznie by uniknąć wycieków pamięci.
4. Każda biblioteka graficzna jest inna jednak wiele bibliotek ma wspólne cechy dzięki czemu tworzenie programów jest dużo łatwiejsze. Jako przykład mogę tu podać model zdarzeń oraz główną pętle programu.
5. Budując graficzny interfejs należy zwracać uwagę na przyzwyczajenia i rozumowanie użytkowników. Dzięki temu interfejs przyjmuje intuicyjny charakter i jest łatwy w użyciu.
6. Dokumentacja programu powinna być tworzona równolegle do implementacji kodu.