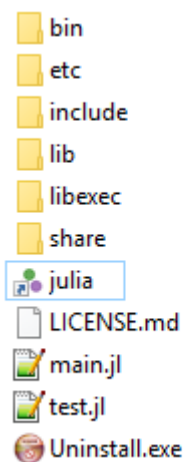


# 1) DOKUMENTACJA UŻYTKOWNIKA ORAZ OPIS INTERFEJSU

## A) Uruchomienie aplikacji

By uruchomić program należy wcześniej zainstalować środowisko Julia na swoim komputerze. Dokładny opis znajduje się w dokumentacji dołączonej na stronie języka (<https://julialang.org/>).

Mając już zainstalowane i skonfigurowane środowisko należy skopiować pliki main.jl oraz test.jl do głównego katalogu języka julia. Następnie powinniśmy uruchomić REPL oraz wykonać polecenie: `include(„main.jl”)`



Rys1. Katalog języka Julia wraz z plikami algorytmu

## B) Menu aplikacji

Po prawidłowym uruchomieniu powinniśmy ujrzeć okno wraz z początkowym menu :

```
julia> include("main.jl")
      MENU Przeszukiwania drzewa binarnego :

Podaj odpowiedni numer czynności, którą chciałbyś wykonać.
1. Wyszukiwanie elementu wśród podanych 5 liczb i klucza
2. Wyszukiwanie elementu wśród znacznej ilości losowych liczb
3. Wykonaj Testy aplikacji
4. ZAKOŃCZ
```

Rys2. Fragment początkowego widoku menu

Gdy wybierzemy na klawiaturze numer jeden oraz zatwierdzimy wybór klawiszem ENTER, zostaniemy poproszeni o podanie klucza głównego oraz pięć kolejnych wartości. Na końcu zapytani zostaniemy jakiej wartości poszukujemy. Przebieg funkcji nr 1 zaprezentowany został na rysunku (Rys3.)

```

Podaj klucz główny:
50
Podaj pierwszą wartość do dodania:
32
Podaj drugą wartość do dodania:
43
Podaj pierwszą trzecią do dodania:
89
Podaj pierwszą czwartą do dodania:
87
Podaj pierwszą piątą do dodania:
80
Podaj jakiej wartości szukamy w drzewie :
80
Znaleziono szukaną wartosc, występuje w drzewie
MENU Przeszukiwania drzewa binarnego :

```

Rys3. Fragment funkcji numer 1

Użytkownikowi wyświetlany jest komunikat sukcesu przeszukiwania wcześniej zdefiniowanego drzewa w wypadku znalezienia szukanej wartości lub komunikat negatywny, dla rezultatu przeciwnego. Jest to zgodnie z wymaganiami łatwa, podstawowa funkcjonalność.

W wypadku wyboru opcji drugiej, użytkownikowi zostaje zaprezentowana już bardziej skomplikowana funkcjonalność. Losowane zostają liczby z przedziału od -100 do 100. Domyślnie korzeń, tworzonego na potrzeby opcji nr 2 funkcjonalności, zostaje ustawiony pośrodku przedziału – na liczbie zero.

```

Losujemy 100 liczb wśród wartości od -100 do 100 , klucz glowny to 0
Następnie sprawdzamy czy wartosci 3, 5, 54, -39, -27, -34 wystąpiły
Znaleziono wartosc 5 w drzewie
Wylosowane drzewo to : [-99,21,-38,-35,-45,-26,2,46,-75,-73,-21,31,-40,-9,66,-58,68,6,29,-22,-77,-
94,10,-56,-64,97,92,-8,96,1,-76,86,25,79,-78,-2,-1,90,99,-32,-50,51,37,-41,91,-61,-28,-54,12,84,-9
1,-24,-96,56,15,-47,100,35,-53,74,45,47,83,-36,98,63,5,36,-69,14,-23,23,59,-14,80,19,-92,52,-30]
MENU Przeszukiwania drzewa binarnego :

```

Rys4. Fragment funkcji numer 2

Jak widzimy na powyższym rysunku (Rys4.) program wylosował i zaprezentował w formie tablicy sto liczb. Po wylosowaniu sprawdził czy wartości 3,5,54,-39,-27,-34 wystąpiły w drzewie. W wypadku wystąpienia wyświetlany jest stosowny komunikat. W losowanej tablicy jedynie znaleziono wartość 5.

Trzecia opcja dostarcza użytkownikowi informacjach o testach. Aplikacja tworzona była zgodnie z metodyką TDD, przeprowadzane testy, a właściwie ich rezultat wyświetlany jest użytkownikowi. Do testowania została użyta biblioteka FactCheck. Testy wykonywane są zgodnie z kolejnością z jaką tworzone było oprogramowanie. Pierw testowana była logika i algorytm drzewa, następnie dodane funkcjonalności dodawania oraz przeszukiwania drzewa. Elementy menu zostały dodane na końcu i zostały wyłączone z metodyki TDD. Wynik testów oraz funkcjonalności pod opcją trzecią widzimy na rysunku na kolejnej stronie (Rys5.).

```

Testowanie wartosci obiektu BST
11 facts verified.
Testowanie lewego i prawego liscia obiektu typu bst
7 facts verified.
Testy funkcji dodajacej elementy do drzewa
Wartosc juz wystapila- nie dodano
Wartosc juz wystapila- nie dodano
6 facts verified.
Testy funkcji zwracajacej true przy wystepowaniu danego elementu w drzewie, false w wypadku przeciwnym
Wartosc juz wystapila- nie dodano
8 facts verified.
MENU Przeszukiwania drzewa binarnego :

```

*Rys5. Fragment funkcji numer 3 - testy*

Ostatnia – czwarta opcja w menu zamyka program. Gdy użytkownik wybierze opcję różną od 1,2,3,4 wtedy wyświetlany jest komunikat, iż wybór jest błędny oraz zostaje ponowione wyświetlenie menu

### **C)Krótki opis algorytmu**

Program oparty jest o rozwiązania związane z drzewami. W tym wypadku zgodnie z wymaganiami zostało zaimplementowane drzewo przeszukiwań binarnych. Każdy element drzewa, również początkowy – zwany korzeniem, posiada swój lewy i prawy element – zwany liśćmi. Elementy większe od korzenia dodawane są do prawego liścia, mniejsze natomiast dodawane są do lewego.

Algorytm – zgodnie z wymaganiami – ma wyszukiwać czy dana wartość pojawiła się w drzewie. Duplikaty czyli elementy nie większe i nie mniejsze od tych znajdujących się w drzewie nie są dodawane.