

LAB 10

Celem zadania jest wykonanie obliczeń numerycznych dla **zdefiniowanych funkcji** (tutaj wielomianów) w dziedzinie **rzeczywistej** lub **zespólonej** (`complex<double>`).

Jeśli definiowana funkcja jest **wielomianem**, to można ją „**poskładać**” z funkcji **identycznościowej** (klasa **X**) oraz **stałej** (klasa **Const**) wykonując odpowiednie ich **iloczyn** (klasa **Iloczyn**) oraz **dodawanie** (klasa **Plus**), czy **odejmowanie** (klasa **Minus**)

Klasa **Wielomian**, jak i wszystkie powyższe klasy tworzą **polimorficzną hierarchię**, dla których **Function** jest abstrakcyjną klasą **bazową**.

Etap 1

W pliku **function.h** dana jest abstrakcyjna klasa bazowa **Function**

Obejrzyj klasę (jest gotowa).

Wyprowadź (wszystko w **function.h**) **publiczne klasy pochodne**: **Const**, **X**, **Plus**, **Minus**, **Iloczyn** oraz **Wielomian** realizujące odpowiednie działania narzucone w klasie bazowej, tj. **wartość funkcji** (metoda **value**) oraz **wartość jej pochodnej** (metoda **prim**).

Zaimplementuj klasy hierarchii realizujące obliczenia dla typu **double**.

- klasa **Const** obsługuje funkcje stałe, np. $f(x)=4.5$
 - konstruktor pobiera stałą, którą przechowuje w prywatnym polu klasy (typu **double**)
 - metody **value** oraz **prim** zwracają odpowiednio wartość funkcji oraz jej pochodnej
- klasa **X** obsługuje funkcję identycznościową $f(x)=x$
 - klasa **nie** posiada żadnych pól
 - metody **value** oraz **prim** zwracają odpowiednio wartość funkcji oraz jej pochodnej
- klasa **Plus** obsługuje **sumę** dwóch funkcji
 - konstruktor pobiera 2 wskaźniki klasy bazowej **Function**, które reprezentują dodawane 2 funkcje i wskaźniki te przechowuje w 2 prywatnych polach wskaźnikowych klasy (pola typu **Function***)
 - met. **value** oblicza **wartość sumy** wykorzystując wywołania met. **value** dla jej składników
 - met. **prim** oblicza **wartość pochodnej sumy** wykorzystując wywołania met. **prim** dla jej składników
 - **destruktor** zwalniający pola wskaźnikowe klasy
- klasa **Minus** obsługuje **różnicę** dwóch funkcji;
 - *zaimplementuj w podobnej konwencji jak **Plus***
- klasa **Iloczyn** obsługuje **iloczyn** dwóch funkcji;
 - *zaimplementuj w podobnej konwencji jak **Plus***

- Do hierarchii dołącz klasę **Wielomian**.
 - Klasa posiada prywatne pole wskaźnikowe typu **Function*** , konstruktor bezparametrowy tej klasy ustawia je na **nullptr**
 - **destruktor** zwalniający pola wskaźnikowe klasy
 - zaimplementuj metody **value** i **prim**
 - napisz **dwie metody** do generowania wielomianów:

generuj1 , która (na podstawie algorytmu Hornera) „składa” wielomian za pomocą klas **Const, X, Plus, Iloczyn**.

Metoda przyjmuje: tablicę **współczynników** oraz jej rozmiar (*poprawne*).

Metoda ustawia wskaźnik utworzonego wielomianu w prywatnym polu.

Schemat Hornera: $W(x) = a_0 + x(a_1 + x(a_2 + \dots + x(a_{n-2} + x(a_{n-1} + xa_n)) \dots))$

generuj2 , która „składa” wielomian za pomocą klas **Const, X, Minus, Iloczyn**.

Metoda przyjmuje: tablicę **zer** wielomianu oraz jej rozmiar (*poprawne*).

Metoda ustawia wskaźnik utworzonego wielomianu w prywatnym polu.

$W(x) = (x - z_0)(x - z_1) \dots (x - z_{n-1})$

Etap 2

Proszę utworzyć (przerobić) hierarchię klas **function.h** na klasy wzorcowe zależne od jednego parametru **T**, który będzie konkretyzowany dla typu **double** oraz **complex<double>**

UWAGA zakomentuj **Etap1** w funkcji main

Etap 3

Zaimplementuj **wzorzec** funkcji **pierwiastek**, realizującej algorytm **metody Newtona** do rozwiązania równania **f(x)=0** w dziedzinie zespolonej.

Szablon funkcji **pierwiastek** zależy od jednego parametru **T**, który określa typ dla szablonu wielomianu (czyli **double** lub **complex<double>**)

Dokładność obliczeń **EPS=1e-6** oraz liczba iteracji **MAX_IT=50** są wewnętrznymi parametrami funkcji zainicjowane podanymi wartościami.

Funkcja **pierwiastek** zwraca znalezione rozwiązanie (jeśli algorytm był zbieżny).

Parametry funkcji reprezentują: przekazana funkcja (wej), punkt startowy (wej), oraz wykonaną liczbę iteracji (wyj).

Do obliczenia wartości bezwzględnej w algorytmie użyj funkcji **abs**.

Algorytm

Startujemy z x_0 (wejściowy **drugi** parametr funkcji). Kolejne przybliżenia są dane wzorem:

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$$

Wykonaj obliczenia (maksymalnie **MAX_ITER=50**) do momentu, gdy

- $|f(x_k)| \leq EPS$
- gdy **osiągnięto** założoną liczbę iteracji **MAX_ITER** **lub** $|f'(x_k)| < EPS$, generuj **wyjątek** (komunikat „Brak zbieżności”)

Etap 4

Dany jest **gotowy** wzorzec funkcji **fraktal** zależny od parametru **N** (tzn. **template <int N>**), która wykonuje obliczenia w celu utworzenia fraktala Newtona (parametr definiuje rozmiar wewnętrznej tablicy).

Funkcja ta korzysta z szablonu tablicy 2-wymiarowej **tab2d**, który posiada 2 parametry:

template <typename T, int N>

Szkielet klasy **tab2d** jest **gotowy**. Dostarcza pewną podstawową funkcjonalność tablicy dwuwymiarowej. Tylko obejrzyj.

Proszę „przerobić” ją na wymaganą klasę szablonoową.

W definicji klasy proszę pozostawić **tylko nagłówki** składowych i funkcji friend.

```
Microsoft Visual Studio Debug Console

----- ETAP 1: double -----
funkcja w1<x1> :
w1<x1> =34
w1'<x1>=20
funkcja w11<x> :
w11<x11> =48
w11'<x11>=44
----- ETAP 2: function template -----
----- ETAP 3 - pierwiastek -----
----- ETAP 4 - fraktale -----

D:\EWA\C++_lato2021\project10_test\Debug\project10_test.exe (process 10936)
exited with code 0.
Press any key to close this window . . .
```

