

| REV  | DATA       | ZMIANY  |
|------|------------|---|
| 0.1  | 15.01.2023 | <i>Adam Andrzej Keków (kekow@student.agh.edu.pl)</i>  |
| 0.11 | 20.01.2023 | Całkowita zmiana koncepcji przechowywania wielomianów |
| 0.5  | 22.01.2023 | <i>Aktualizacja rozpoznanych błędów</i>               |
|      |            |   |

# SYMBOLICZNE OBLICZENIE POCHODNYCH WYRAŻENIA

Autor: Adam Andrzej Keków  
Akademia Górniczo-Hutnicza

Kraków 2023

## Spis treści

|   |   |
|---|---|
| Lista oznaczeń:.....  | 3 |
| Wstęp: .....  | 4 |
| Wymagania systemowe: .....  | 4 |
| Funkcjonalność: .....   | 4 |
| Analiza Problemu: .....   | 4 |
| Projekt Techniczny: .....   | 5 |
| Opis realizacji: .....  | 6 |
| Opis wykonanych testów (testing report) - lista buggów, uzupełnień, itd. .... | 7 |
| Podręcznik użytkownika: .....   | 8 |
| Metodologia rozwoju i utrzymania systemu .....                                | 8 |
| Bibliografia .....  | 8 |

## Lista oznaczeń:

|       |   |
|-------|---|
| tg()  | Tangens   |
| ctg() | Cotangens   |
| sin() | Sinus   |
| cos() | Cosinus   |
| ln()  | Logarytm naturalny ( o podstawie z liczby e)        |
| $x^n$ | Potęgowanie - „x” do potęgi „n”                     |
| *     | Znak mnożenia                                       |
| /     | Znak dzielenia                                      |
| mapa  | Słownik przekształcający jeden typ zmiennej w drugą |
| SDK   | System Development Kit                              |

Tabela 1. Lista oznaczeń

## Wstęp:

Projekt zakłada obliczanie pochodnych metodą symboliczną z interfejsem w postaci wiersza poleceń.

## Wymagania systemowe:

Podstawowe założenia projektu:

Przygotowanie bazy pod obliczanie pochodnej dowolnej funkcji.

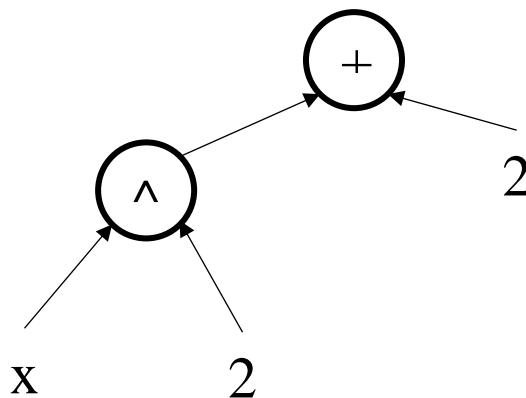
Analiza matematyczna problemu w celu znalezienia rozwiązania jak najbliższego optimum.

## Funkcjonalność:

Wejście oraz wyjście jako ciąg znaków w konsoli. Obliczanie pochodnych metodą symboliczną. Upraszczanie wyrażeń. Prosty interfejs tekstowy.

## Analiza Problemu:

Równanie matematyczne możemy zdefiniować jako drzewo, gdzie każde działanie matematyczne tworzy jedną lub dwie gałęzie dla przykładu  $x^2+2$  można zapisać jako:



Uwzględniając kolejność działań, można zbudować drzewo które przy zachowaniu kilku zasad da poprawną odpowiedź.

Obliczenia pochodnej zawsze zaczynają się od najdalszej gałęzi drzewa, przy czym lewa gałąź jest wcześniejsza.

Korzystając z stabilizowanych wzorów łatwo zauważyć że metoda rekurencyjna jest możliwa.

Tak więc każde obliczenie pochodnej może oznaczać tylko i wyłącznie obliczenie algebraicznego wyrażenia lub pochodnej.

Przechowywanie wyniku, c++ nie posiada typu zdolnego przechowywać wielomiany lub funkcje matematyczne. Stworzono strukturę mogącą przechowywać wielomiany jako parę potęgi oraz liczby, resztę funkcji jako ciąg znaków.

Możliwość w przyszłości zastosowania vectora stringów do znajdowania identycznych zapisów i zamiany w mnożenie, dzielenie i potęgowanie.

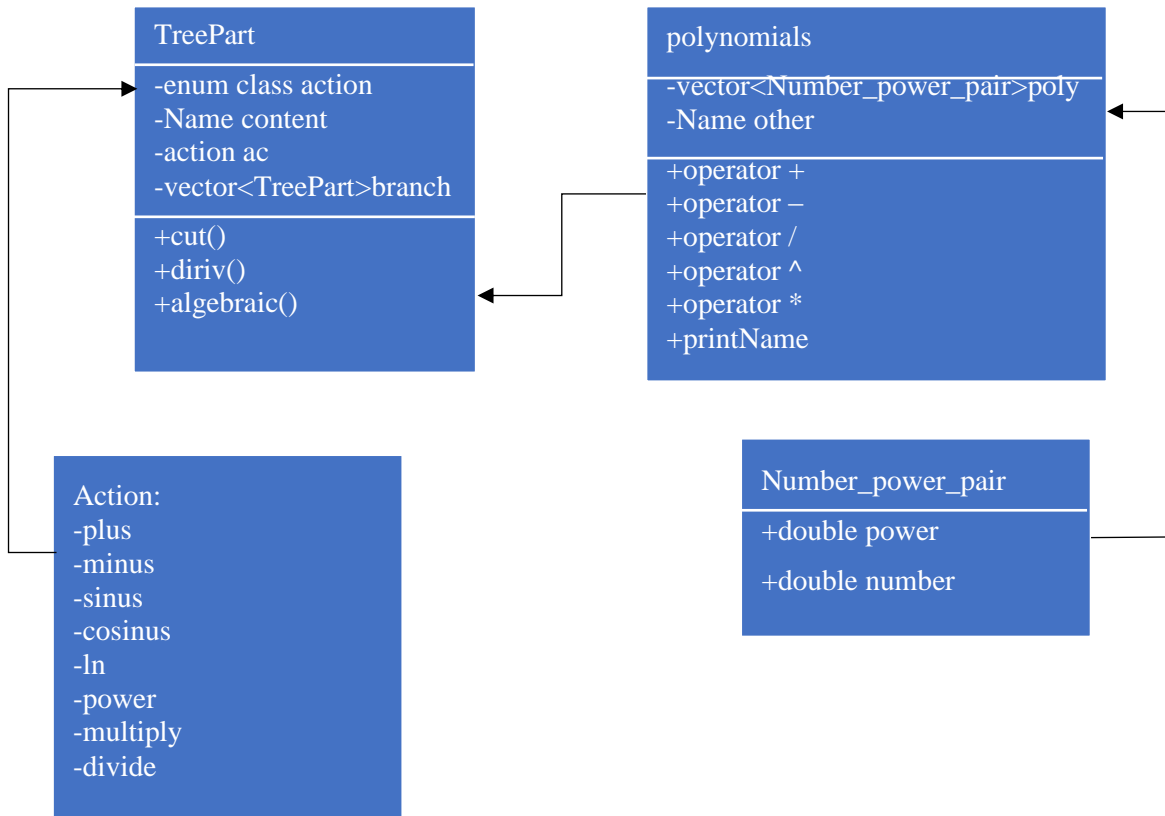
## Projekt Techniczny:



Rys.2.1

Diagram (Rys 2.1) przedstawia przepływ programu. W nieskończonej pętli czeka na nowe informacje od użytkownika by po zatwierdzeniu przygotować odpowiedź.

Program zaczyna od pozbycia się niechcianych znaków z wprowadzonego tekstu. Dzielenie na drzewo polega na rekurencyjnym sprawdzaniu czy znajdują się znaki działania i jeżeli się znajdują to sprawdza czy podział nie przeciął by granicy nawiasów co zaburzyło by kolejność działań.



Rys 2.2 Hierarchia klas

Sporym problemem jest przechowywanie zapisu matematycznego, na rys.2.2 widzimy klasę polynomials posiada ona jak nazwa wskazuje wielomian, ale także tekst reprezentujący inne wyrażenia matematyczne. Wielomian jest tworzony z wektora par liczb reprezentujących potęgę oraz wartość liczby. Na strukturze polynomials można wykonywać podstawowe operacje dzięki przeciążeniu operatorów. PrintName pozwala drukować polynomials do stringa.

TreePart jako struktura cechuje się posiadaniem „content” i jest to zawartość tekstowa którą wykorzystujemy przy dzieleniu na mniejsze kawałki. W action zapisywany jest znalezione działanie które ma być wykonane na gałęziach.

Funkcja Cut rekurencyjnie dzieli drzewo i zapisuje w gałęziach(branch) coraz to mniejsze fragmenty wyrażenia.

Funkcja diriv() składa pochodną odnosząc się do gałęzi(branch) i operacji(action). Korzystając z struktury switch(action) możliwe jest zaprogramowanie dowolnego zachowania dla każdej operacji. Wykorzystano tablicę najpopularniejszych funkcji.

## Opis realizacji:

Platforma testowa:

Procesor x86, amd c70, Visual Studio wersja 17.3.6, wersja zestawu SDK Windows 10.0.19041.0, kompilator Microsoft (R) C/C++ wersja kompilatora optymalizującego 19.33.31630 dla x64. System operacyjny Windows Server 2012 r2.

Procesor Ryzen 5 3500, system operacyjny: Linux Ubuntu 22.04 oraz Windows 11.

System kontroli źródeł: git z kopią online w github, cmake jako system budowy projektu.

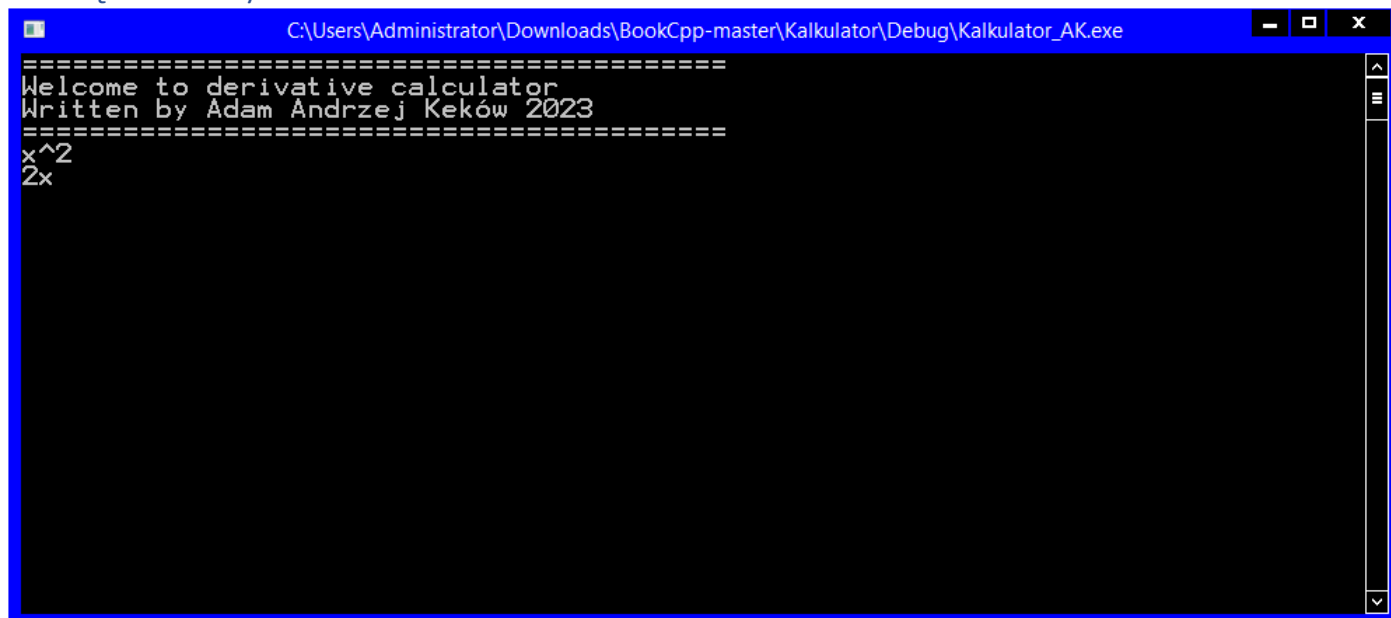
## Opis wykonanych testów (testing report) - lista buggów, uzupełnień, itd.

Sprawdzono działanie programu pod systemem linux.

| Kod usterki | Data     | Autor            | Opis  | Stan                    |
|-------------|----------|------------------|---|-------------------------|
| 0x0         |          | Adam Keków       | Nazwisko się nie wyświetla  | Naprawione              |
| 0x01        | 21.01.23 | Wiktor Pantak    | Automatyczne usuwanie nawiasów nie uwzględnia specjalnych przypadków      | Naprawiono              |
| 0x02        | 21.01.23 | Szymon Szczerbik | Logarytm nie przyjmuje funkcji złożenia                                   | Naprawione              |
| 0x03        | 21.01.23 | Szymon Szczerbik | Czasem kolejność działań jest błędna                                      | Naprawione              |
| 0x04        |          | Adam Keków       | Poprawne działanie tylko przy zaimplementowanych funkcjach matematycznych | Nie dokończone          |
| 0x05        | 22.01.   | Szymon Szczerbik | Znaki +/- znajdują się wewnątrz nawiasu zamiast przed nim                 | Nie udało się odtworzyć |
|             |          |                  |   |                         |

Tabela 2. Negatywne wyniki testów.

## Podręcznik użytkownika:



```

C:\Users\Administrator\Downloads\BookCpp-master\Kalkulator\Debug\Kalkulator_AK.exe
=====
Welcome to derivative calculator
Written by Adam Andrzej Keków 2023
=====
x^2
2x

```

Użytkownik wprowadza wyrażenie które chce zróżniczkować i po zatwierdzeniu enterem otrzymuje wynik w następnej linii.

Niektóre funkcje, takie jak np.: tangens, można zapisać jako sinus przez cosinus.

Ważne jest by wprowadzać tylko zaimplementowane funkcje, ponieważ w przeciwnym wypadku wynik nie będzie poprawny, różniczkowanie zawsze jest po zmiennej „x”. Nazwy funkcji znajdują się w Lista oznaczeń:

Zapis „sinx” jest równoważny „sin(x)”, tak samo „2x” jest równoznaczne z „(2\*x)”. Oznacza to że „2x^2” nie jest równe „2\*x^2”.

## Metodologia rozwoju i utrzymania systemu

Nowe funkcje matematyczne mogą być uzupełniane wraz z upływem czasu poprzez uzupełnianie słownika mapy. Program może być rozszerzony o interfejs graficzny zawierający symbole bardziej zbliżone do prawidłowego matematycznego opisu. Metoda wprowadzania może zostać rozszerzona o klawiaturę z widocznymi symbolami. Możliwe jest też wprowadzenie oznaczenia stałej i zmiennej do różniczkowania innej niż „x”.

## Bibliografia:

- M. Gewert, Z. Skoczylas, Analiza matematyczna 1, Definicje, twierdzenia, wzory, Oficyna Wydawnicza GiS, Wrocław, 2009
- Bogusław Cyganek, introduction-to-programming-with-c-for-engineers, Wiley, 2021