

Kalkulator Dla Elektroników/Informatyków

Autor: Hubert Gąsior

Spis treści

| | |
|--------------------------------|---|
| 1. Wstęp..... | 2 |
| 2. Wymagania Systemowe..... | 2 |
| 3. Funkcjonalność..... | 2 |
| 4. Analiza Problemu..... | 3 |
| 5. Projekt Techniczny..... | 5 |
| 6. Opis Realizacji..... | 6 |
| 7. Opis wykonanych testów..... | 7 |
| 8. Podręcznik użytkownika..... | 8 |
| 9. Bibliografia..... | 9 |

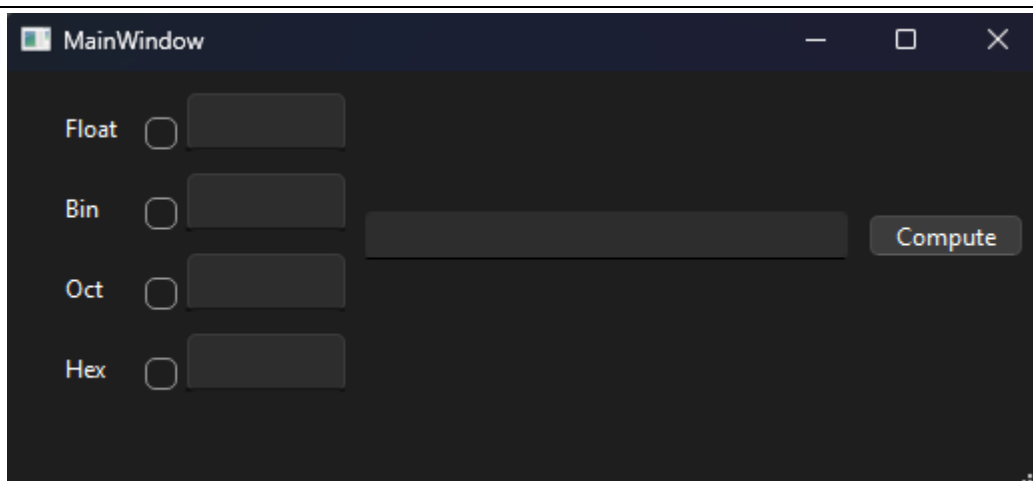
1. Wstęp

Niniejszy dokument dotyczy projektu kalkulatora przeznaczonego dla elektroników i informatyków. Program oferuje funkcje wykonywania podstawowych operacji matematycznych, takich jak dodawanie, odejmowanie, mnożenie, dzielenie. Dodatkowo umożliwia konwersję liczb między systemami liczbowymi: dziesiętnym, binarnym, ósemkowym i szesnastkowym. Kalkulator wspiera wykonywanie operacji w każdym z wymienionych systemów liczbowych oraz uwzględnia hierarchię wykonywania działań obsługując również nawiasy.

2. Wymagania systemowe

- System operacyjny Windows.
- Środowisko Qt lub Visual studio 2022 wraz z biblioteką MFC.

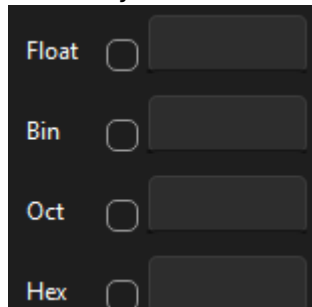
3. Funkcjonalność



Rys. 1 interfejs graficzny kalkulatora

1. Wyświetlanie wyników

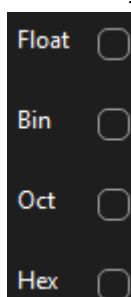
Rysunek poniżej przedstawia pola „Float”, „Bin”, „Oct” i „Hex”, w których wyświetlany jest wynik obliczonego wyrażenia po naciśnięciu przycisku „Compute”. Wyniki są prezentowane w odpowiednich systemach liczbowych.



Rys. 2 Pola wyświetlające wyniki

2. Wybór systemu liczbowego

Na rysunku trzecim przedstawiono checkboxy, które odpowiadają za wybór systemu liczbowego. Zaznaczenie checkboxa przy odpowiedniej etykiecie systemu liczbowego umożliwia wprowadzenie wyrażenia w wybranym formacie.



Rys. 3 Pola wyświetlające wyniki oraz checkboxy wyboru systemu liczbowego

3. Wprowadzenie oraz obliczanie wyrażenia

Poniższy rysunek przedstawia pole tekstowe, w którym użytkownik może wprowadzić wyrażenie matematyczne do obliczenia w wybranym systemie liczbowym. Po prawej stronie pola znajduje się przycisk „Compute”. Naciśnięcie tego przycisku powoduje obliczenie wprowadzonego wyrażenia.



Rys. 4 okno do wprowadzania wyrażenia oraz przycisk „Compute” służący do obliczania wyrażeń

4. Analiza problemu

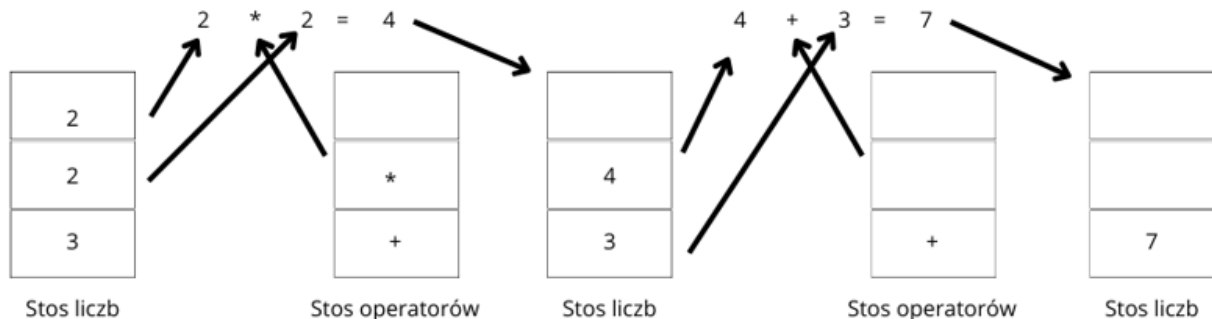
Równanie zapisane w klasycznej notacji infiksowej, np. „ $3+2*2$ ”, można przedstawić w odwrotnej notacji polskiej (RPN) jako „ $322*+$ ”. Po przekonwertowaniu równania na RPN, obliczenia można przeprowadzić w prosty sposób, stosując następujący algorytm:

1. Wszystkie liczby i operatory są umieszczane na stosie w kolejności, w jakiej występują w wyrażeniu RPN.
2. Ze stosu zdejmowane są dwie liczby oraz operator.
3. Wykonywane jest działanie określone przez operator, a wynik jest ponownie wrzucany na stos liczb.

4. Proces jest powtarzany, aż stos operatorów będzie pusty.

W ten sposób obliczenie wyrażenia zostaje zakończone.

$$3+2*2 \rightarrow 322*+$$



Rys. 5 algorytm obliczania wyrażenia zapisanego przy pomocy RPN

Algorytm konwersji wyrażenia z postaci infiksowej do RPN

Algorytm „Shunting Yard” zastosowany w projekcie kalkulatora przekształca wyrażenie matematyczne z notacji infiksowej na RPN, uwzględniając priorytet działań. Jego działanie można opisać w następujących krokach:

1. Liczba:

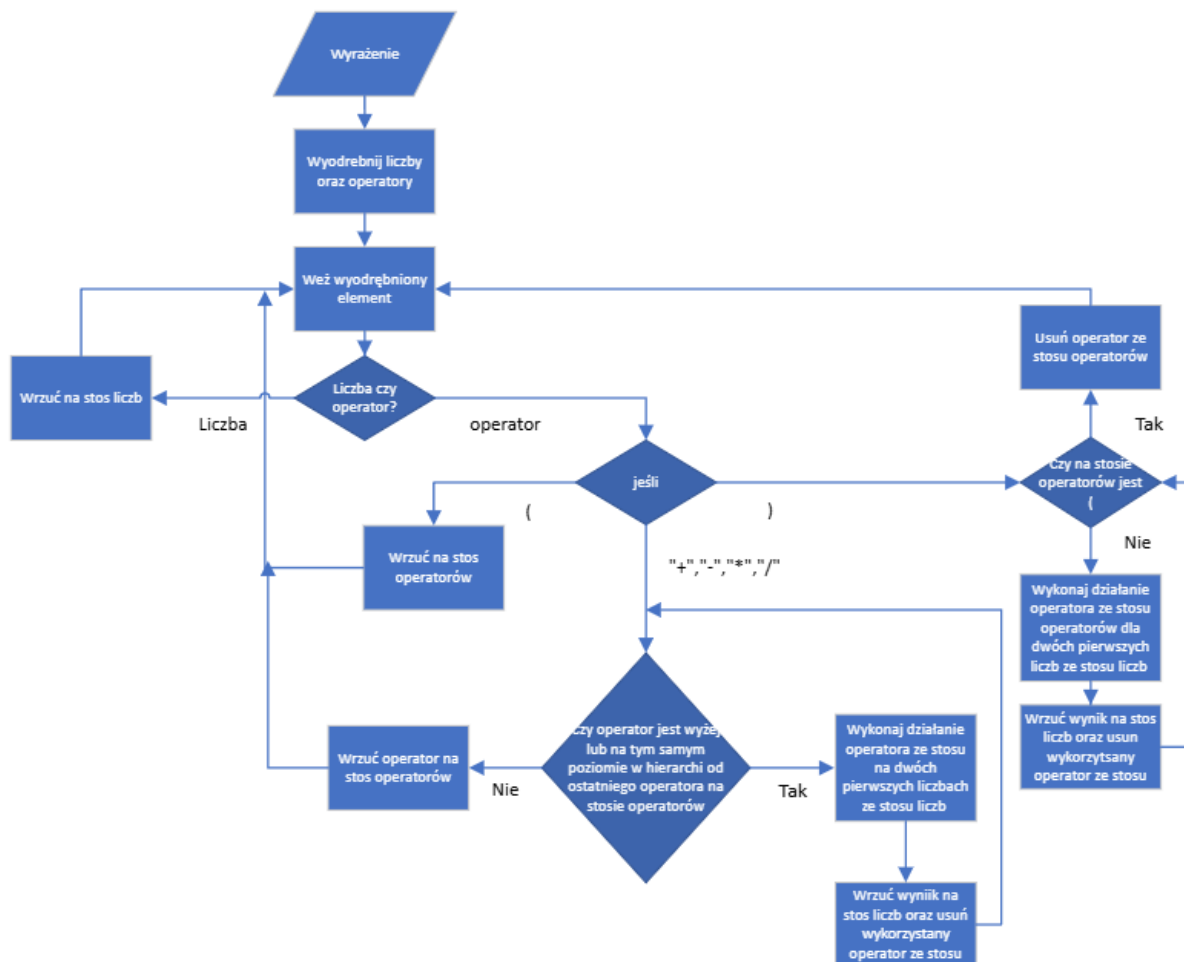
Jeśli element jest liczbą, odłóż go na stos liczb.

2. Operator:

- Jeśli stos operatorów jest pusty, odłóż operator na stos.
- W przeciwnym razie:
 - Określ priorytet bieżącego operatora.
 - Jeśli operator na szczycie stosu operatorów ma równy lub wyższy priorytet, wykonaj działanie operatora ze stosu na dwóch pierwszych liczbach ze stosu liczb.
 - Powtarzaj tę procedurę, aż bieżący operator będzie miał wyższy priorytet niż operator na stosie. Następnie odłóż bieżący operator na stos operatorów.

3. Powtarzanie:

Powtarzaj kroki 1 i 2 dla wszystkich elementów wyrażenia, aż wszystkie liczby i operatory zostaną przetworzone.



Rys. 7 Diagram algorytmu „shunting yard” wraz z obsługą nawiasów

5. Projekt techniczny

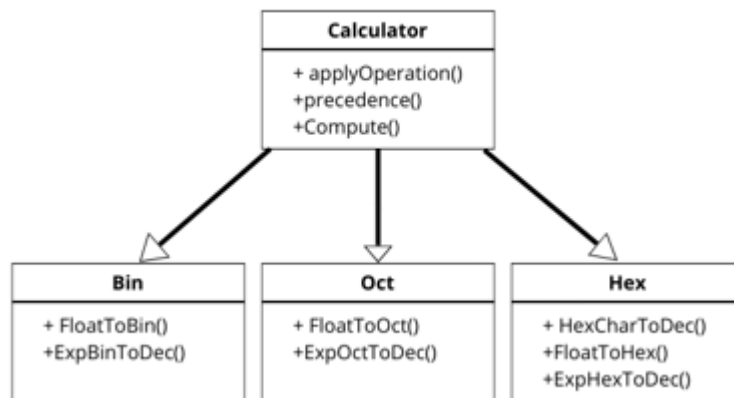
1. Hierarchia klas

Poniższy rysunek przedstawia hierarchię klas w zaimplementowanym kalkulatorze. Główną klasą projektu jest „Calculator”, która zawiera publiczne metody umożliwiające obliczanie wyrażeń wprowadzonych przez użytkownika. Klasa ta dziedziczy publiczne metody z klas „Bin”, „Oct” i „Hex”.

Metody te umożliwiają:

1. Konwersję wprowadzonego wyrażenia z wybranego systemu liczbowego (binarny, ósemkowy, szesnastkowy) na system dziesiętny, który jest używany przez funkcję „Compute()” kalkulatora.
2. Konwersję wyniku obliczeń z systemu dziesiętnego na odpowiedni system liczbowy.

Dzięki takiej strukturze kalkulator zapewnia elastyczność i wsparcie dla różnych systemów liczbowych.



Rys. 6 Diagram UML kalkulatora

6. Opis realizacji

1. Klasa Calculator

- Metoda applyOperation (funkcja inspirowana ChatemGPT)**
 Funkcja wykonująca działanie matematyczne w zależności od operatora.
Input: Dwie liczby zmiennoprzecinkowe oraz operator w formacie char.
Output: Wynik operacji w formacie zmiennoprzecinkowym.
- Metoda precedence (funkcja inspirowana ChatemGPT)**
 Funkcja zwracająca hierarchię danego operatora.
Input: Operator w formacie char.
Output:
 - 2 dla operatorów mnożenia (*) i dzielenia (/),
 - 1 dla operatorów dodawania (+) i odejmowania (-),
 - 0 dla operatorów nieobsługiwanych.
- Metoda Compute**
 Funkcja obliczająca wyrażenie matematyczne. Wykorzystuje algorytm „shuntingyard” do konwersji wyrażenia infiksowego na RPN, a następnie oblicza wynik.
Input: Wyrażenie matematyczne w formacie std::string.
Output: Wynik w formacie zmiennoprzecinkowym.

2. Klasa Bin

- Metoda FloatToBin**
 Funkcja konwertująca liczbę zmiennoprzecinkową na liczbę w systemie binarnym.
Input: Liczba zmiennoprzecinkowa.
Output: Liczba binarna w formacie std::string.
- Metoda ExpBinToDec**
 Funkcja przekształcająca wyrażenie matematyczne wprowadzone w systemie binarnym na wyrażenie w systemie dziesiętnym.

Input: Wyrażenie w systemie binarnym w formacie std::string.

Output: Wyrażenie w systemie dziesiętnym w formacie std::string.

3. Klasa Oct

- **Metoda FloatToOct**

Funkcja konwertująca liczbę zmiennoprzecinkową na liczbę w systemie ósemkowym.

Input: Liczba zmiennoprzecinkowa.

Output: Liczba w systemie ósemkowym w formacie std::string.

- **Metoda ExpOctToDec**

Funkcja przekształcająca wyrażenie matematyczne wprowadzone w systemie ósemkowym na wyrażenie w systemie dziesiętnym.

Input: Wyrażenie w systemie ósemkowym w formacie std::string.

Output: Wyrażenie w systemie dziesiętnym w formacie std::string.

4. Klasa Hex

- **Metoda HexCharToDec**

Funkcja zwracająca wartość pojedynczego znaku w systemie szesnastkowym.

Input: Znak w formacie char.

Output: Wartość w systemie dziesiętnym w formacie int.

- **Metoda FloatToHex**

Funkcja konwertująca liczbę zmiennoprzecinkową na liczbę w systemie szesnastkowym.

Input: Liczba zmiennoprzecinkowa.

Output: Liczba w systemie szesnastkowym w formacie std::string.

- **Metoda ExpHexToDec**

Funkcja przekształcająca wyrażenie matematyczne wprowadzone w systemie szesnastkowym na wyrażenie w systemie dziesiętnym.

Input: Wyrażenie w systemie szesnastkowym w formacie std::string.

Output: Wyrażenie w systemie dziesiętnym w formacie std::string.

7. Opis wykonanych testów

W celu weryfikacji poprawności działania funkcji użytych w projekcie kalkulatora przeprowadzono testy jednostkowe z wykorzystaniem biblioteki **GTest**. Testy obejmowały odpowiednie przypadki zgodne z zamierzonym zastosowaniem funkcji. Szczegóły testów dla poszczególnych klas i funkcji przedstawiono poniżej:

Klasy Bin, Oct, Hex

- **Metoda FloatToX() (konwersja liczby zmiennoprzecinkowej na różne systemy liczbowe):**
 - Standardowa konwersja liczby.
 - Konwersja liczby zero.
 - Skrajne przypadki, w tym liczby z częścią ułamkową.

- **Metoda ExpXToDec() (konwersja wyrażeń na system dziesiętny):**
 - Konwersja prostych wyrażeń z operatorami.
 - Konwersja złożonych wyrażeń z wieloma operatorami.
 - Wyrażenia bez operatorów (pojedyncze liczby).
 - Obsługa pustych wyrażeń.
 - Obsługa błędnych argumentów.
- **Metoda HexCharToDec() (konwersja pojedynczego znaku w systemie szesnastkowym):**
 - Poprawność konwersji dla argumentów w różnych formatach (małe i wielkie litery).
 - Obsługa błędnych argumentów.

Klasa Calculator

- **Metoda applyOperation() (wykonywanie działań matematycznych):**
 - Poprawność wyników dla podstawowych operacji matematycznych (dodawanie, odejmowanie, mnożenie, dzielenie).
 - Obsługa przypadku dzielenia przez zero.
- **Metoda precedence() (priorytet operatorów):**
 - Sprawdzenie poprawności zwracanych wartości dla operatorów +, -, *, /.
 - Obsługa operatorów nieobsługiwanych.
- **Metoda Compute() (obliczanie wyrażeń matematycznych):**
 - Poprawność wyników dla prostych wyrażeń.
 - Poprawność wyników dla złożonych wyrażeń, uwzględniających hierarchię działań oraz nawiasy.
 - Obsługa pustego argumentu.

Przeprowadzone testy wykazały poprawne działanie funkcji dzięki czemu kalkulator działa zgodnie z założeniami projektu oraz jest odporny na błędne dane wejściowe.

8. Podręcznik użytkownika

Budowanie aplikacji przy pomocy CMake:

1. MFC (Możliwe jedynie z zainstalowanym Visual Studio z biblioteką MFC)
Aby zbudować projekt z interfejsem graficznym MFC, należy wykonać następujące kroki:
 1. Stwórz folder „build” w katalogu projektu.
 2. Otwórz wiersz polecenia (CMD) i przejdź do katalogu „build”, używając komendy: „cd build”

3. wykonaj polecenie „cmake ..” komenda ta wykorzysta plik CMakeLists.txt stworzony przez prof. dr hab. inż. Bogusława Cyganka i wygeneruje plik Calculator.sln.
 4. Otwórz plik Calculator.sln w Visual Studio.
 5. Uruchom kompilację projektu w Visual Studio (np. używając opcji "Build Solution").
 6. Po zakończeniu kompilacji powstanie plik wykonywalny Calculator.exe, który będzie gotowy do użycia.
2. Qt (Możliwe jedynie z zainstalowanym środowiskiem Qt)
- Aby zbudować projekt z interfejsem graficznym Qt przy użyciu MinGw32, należy wykonać następujące kroki:
1. Stwórz folder „build” w katalogu projektu.
 2. Otwórz wiersz polecenia (CMD) i przejdź do katalogu „build”, używając komendy: „cd build”
 3. wykonaj polecenie „cmake -G "MinGW Makefiles" ..” przed wykonaniem polecenia należy upewnić się że w pliku CMakeLists.txt są ustawione odpowiednie ścieżki kompilatorów dostarczonych przez Qt.
 4. Następnie należy wykonać polecenie „mingw32-make” po zakończeniu kompilacji powstanie plik wykonywalny Calculator.exe, który będzie gotowy do użycia.

9. Bibliografia

”Introduction to Programming with C++ for Engineers” by Bogusław Cyganek

<https://doc.qt.io/qt-6/cmake-get-started.html#building-a-c-gui-application>

<https://chatgpt.com/>

https://en.wikipedia.org/wiki/Shunting_yard_algorithm

https://en.wikipedia.org/wiki/Reverse_Polish_notation