**KALKULATOR DLA ELEKTRONIKÓW/INFORMATYKÓW**

Autor: Kamil Musiałkowski

Kraków 2025

**SPIS TREŚCI**

1. Wstęp 3
2. Wymagania systemowe 4
3. Analiza problemu 6
4. Funkcjonalność
5. Projekt techniczny
6. Opis realizacji
7. Opis wykonanych testów
8. Bibliografia

1. **Wstęp**

Celem projektu jest stworzenie aplikacji typu kalkulator, która umożliwia wykonywanie podstawowych operacji matematycznych oraz konwersji liczbowych pomiędzy różnymi systemami liczbowymi (binarnym, ósemkowym, dziesiętnym i szesnastkowym). Program będzie obsługiwał liczby całkowite, co pozwala na dokładne obliczenia i konwersje w kontekście obliczeń wykorzystywanych przez elektroników i informatyków. Kalkulator będzie również obsługiwał nawiasy, umożliwiając wykonywanie działań z zachowaniem priorytetów operatorów, co zapewni poprawność wyników w bardziej złożonych wyrażeniach matematycznych.

Aplikacja składa się z dwóch głównych modułów:

* **Moduł logiki (systemu klas)** – odpowiedzialny za implementację funkcjonalności kalkulatora, takich jak operacje arytmetyczne (dodawanie, odejmowanie, mnożenie, dzielenie) oraz konwersje pomiędzy systemami liczbowymi. Moduł ten będzie zaprojektowany w sposób modułowy i zorientowany obiektowo, co zapewni łatwość rozbudowy oraz niezależność od interfejsu użytkownika.
* **Moduł interfejsu użytkownika** – zapewniający komunikację z użytkownikiem poprzez prosty, tekstowy interfejs dialogowy. Dzięki temu użytkownik będzie mógł intuicyjnie korzystać z dostępnych funkcji kalkulatora, takich jak operacje na liczbach całkowitych oraz konwersje między systemami liczbowymi, a także wykonywanie obliczeń z uwzględnieniem nawiasów.

Program będzie obsługiwał podstawowe operacje matematyczne, takie jak **dodawanie, odejmowanie, mnożenie i dzielenie**, a także **nawiasy**, umożliwiające zachowanie odpowiedniej kolejności wykonywania działań. Kalkulator będzie obsługiwał liczby całkowite w różnych systemach liczbowych.

Projekt będzie realizowany w środowisku **Visual Studio 2022** z wykorzystaniem języka programowania **C++**. Do zarządzania strukturą projektu oraz jego kompilacją zastosowany zostanie **CMake** – narzędzie umożliwiające definiowanie procesu budowania aplikacji w sposób przenośny i niezależny od konkretnego środowiska.

Kod źródłowy aplikacji zostanie podzielony na osobne pliki, co jest zgodne z dobrymi praktykami programowania w C++. Logika aplikacji zostanie zaimplementowana w plikach **.cpp** (zawierających definicje funkcji i klas), a interfejsy oraz deklaracje klas zostaną umieszczone w plikach **.h**. Takie podejście zapewnia przejrzystość i modularność kodu, ułatwiając zarówno jego rozwój, jak i późniejszą konserwację.

Dzięki zastosowaniu opisanej struktury oraz narzędzi projekt będzie elastyczny, łatwy w rozbudowie i dostosowany do potrzeb użytkowników oraz wymagań współczesnego oprogramowania.

1. **Wymagania systemowe**

### Podstawowe założenia projektu:

1. **Obsługa liczb całkowitych**
   * Kalkulator będzie operował wyłącznie na liczbach całkowitych, co zapewni precyzyjne obliczenia i ułatwi konwersje między różnymi systemami liczbowymi.
   * Wyniki operacji będą zaokrąglane w dół do najbliższej liczby całkowitej w przypadku dzielenia.
2. **Obsługa nawiasów**
   * Program musi umożliwiać użycie nawiasów w wyrażeniach matematycznych, co pozwala na modyfikowanie domyślnej kolejności wykonywania działań.
   * Kalkulator powinien rozpoznawać poprawnie zagnieżdżone nawiasy oraz sygnalizować błędy w ich zastosowaniu.
3. **Obsługa różnych systemów liczbowych**
   * Kalkulator będzie umożliwiał konwersje pomiędzy systemami binarnym, ósemkowym, dziesiętnym i szesnastkowym.
   * Operacje arytmetyczne (dodawanie, odejmowanie, mnożenie, dzielenie) będą mogły być wykonywane w wybranym systemie liczbowym.
4. **Priorytety operatorów**
   * Kalkulator będzie przestrzegał matematycznych zasad priorytetu działań:
     + Mnożenie i dzielenie mają wyższy priorytet niż dodawanie i odejmowanie.
     + Nawiasy mają najwyższy priorytet i pozwalają na modyfikowanie kolejności działań.
5. **Obsługa błędów**
   * Program powinien obsługiwać typowe błędy, takie jak:
     + Nieprawidłowe dane wejściowe (np. litery w liczbie binarnej).
     + Dzielenie przez zero.
     + Błędne nawiasy w wyrażeniu matematycznym.
   * W przypadku wykrycia błędu aplikacja powinna generować odpowiedni komunikat dla użytkownika.
6. **Prostota interfejsu**
   * Interfejs użytkownika będzie w formie tekstowego menu dialogowego, zapewniając przejrzystość i intuicyjną obsługę.
   * Komunikaty powinny być czytelne i jednoznacznie informować użytkownika o dostępnych funkcjach oraz ewentualnych błędach.
7. **Zgodność z językiem C++**
   * Kod programu musi być napisany w standardzie **C++11** lub nowszym, co umożliwia wykorzystanie nowoczesnych funkcji języka.
   * Aplikacja powinna być zgodna z zasadami dobrych praktyk programowania w C++.
8. **Modularność**
   * Struktura projektu powinna być modularna i oparta na zasadach programowania obiektowego.
   * Oddzielenie logiki aplikacji od interfejsu użytkownika zapewni elastyczność i możliwość rozbudowy w przyszłości.
9. **Środowisko programistyczne**
   * Projekt będzie realizowany w środowisku **Visual Studio 2022** z wykorzystaniem narzędzia **CMake** do zarządzania procesem budowy aplikacji.
   * Kod źródłowy będzie podzielony na pliki **.cpp** (implementacje) i **.h** (deklaracje).
10. **Testowalność**

* Projekt zostanie wyposażony w zestaw testów automatycznych opartych na frameworku **Google Test (GTest)**, który zapewnia elastyczne i czytelne środowisko do pisania oraz wykonywania testów jednostkowych.
* Testy będą obejmować kluczowe funkcjonalności, takie jak obsługa operacji matematycznych, priorytetów operatorów, konwersji systemów liczbowych oraz obsługi błędów.
* Framework GTest pozwoli na przeprowadzanie szczegółowej analizy wyników testów, co zwiększy stabilność i poprawność działania programu w różnych scenariuszach użytkowania.

1. **Analiza problemu**

Elektronicy i informatycy często operują na liczbach w różnych systemach liczbowych, takich jak binarny, ósemkowy czy szesnastkowy. Tradycyjne kalkulatory zazwyczaj nie oferują pełnego wsparcia dla takich operacji. Ponadto, konwersje pomiędzy różnymi systemami mogą być czasochłonne i podatne na błędy, szczególnie gdy wymagają ręcznych obliczeń.

|  |  |  |  |
| --- | --- | --- | --- |
| Nazwa | Znaki występujące | Nazwa naukowa | Skrót |
| dwójkowy | 0,1 | Binarny | bin |
| ósemkowy | 0,1,2,3,4,5,6,7 | Oktalny | oct |
| dziesiętny | 0,1,2,3,4,5,6,7,8,9 | Decymalny | dec |
| szesnastkowy | 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F | Hexalny | hex |

**Tabela 1.** Spis systemów liczbowych

Przy konwersji między systemami pozycyjnymi istnieją dwa przypadki, które rozwiązujemy w następujący sposób:

* **Przeliczanie z systemu o mniejszej bazie na system o większej bazie:**

W takim przypadku analizujemy kolejne pozycje liczby, zapisujemy je w postaci potęg liczby będącej podstawą systemu docelowego, a następnie sumujemy te wartości, aby uzyskać wynik.

* **Przeliczanie z systemu o większej bazie na system o mniejszej bazie:**

W tym przypadku dzielimy liczbę z systemu o większej bazie przez podstawę systemu docelowego. Reszty z kolejnych dzielen zapisujemy w odwrotnej kolejności, co daje nam wynik w nowym systemie.

Metody przeliczania poszczególnych konwersji:

### **binarny (2) na dziesiętny (10)**

Dla liczby w systemie binarnym, np. *bnbn-1…b0​*, przeliczamy ją na system dziesiętny jako:

*wynik = b0⋅20 + b1⋅21 + b2⋅22 + ⋯ + bn⋅2n*

* **dziesiętny (10) na binarny (2)**

Liczbę dziesiętną przeliczamy na binarną, dzieląc ją przez 2, a reszty zapisujemy w odwrotnej kolejności:

1. Podzielenie liczby przez 2
2. Zapisanie reszty
3. Powtarzanie kroków, aż liczba wyniesie 00
4. Odczytanie reszty od końca

* **ósemkowy (8) na dziesiętny (10)**

Dla liczby w systemie ósemkowym, *onon-1…o0*​, przeliczamy na dziesiętny:

*wynik = o0⋅80 + o1⋅81 + o2⋅82 + ⋯ + on⋅8n*

### **dziesiętny (10) na ósemkowy (8)**

Liczbę dziesiętną przeliczamy na ósemkowy, dzieląc ją przez 8, a reszty zapisujemy w odwrotnej kolejności:

1. Podzielenie liczby przez 8
2. Zapisanie reszty
3. Powtarzanie kroków, aż liczba wyniesie 00
4. Odczytanie reszty od końca

### **szesnastkowy (16) na dziesiętny (10)**

Dla liczby w systemie szesnastkowym, *hnhn-1…h0*​, przeliczamy na dziesiętny jako:

*wynik = h0⋅160 + h1⋅161 + h2⋅162 + ⋯ + hn⋅16n*

gdzie: znaki A−F oznaczają wartości 10−15

### **dziesiętny (10) na szesnastkowy (16)**

Liczbę dziesiętną przeliczamy na szesnastkowy, dzieląc ją przez 16, a reszty zapisując w odwrotnej kolejności:

1. Podzielenie liczby przez 16
2. Zapisanie reszty, jeśli reszta wynosi więcej niż 99, przekształcenie jej na odpowiedni znak (np. 10→A, 11→B).
3. Powtarzanie kroków, aż liczba wyniesie 00
4. Odczytanie reszty od końca

Kolejnym ważnym aspektem poprawnego działania kalkulatora jest kolejność wykonywania działań. W procesie wykonywania działań arytmetycznych w dowolnym systemie liczbowym istotne jest przestrzeganie zasad priorytetów operatorów, które określają kolejność wykonywania operacji. Jedną z podstawowych zasad matematyki, stosowaną w praktycznie każdym systemie liczbowym, jest wyższy priorytet mnożenia i dzielenia w porównaniu z dodawaniem i odejmowaniem. Oznacza to, że w wyrażeniach zawierających różne operatory najpierw wykonywane są operacje mnożenia i dzielenia, a dopiero potem dodawanie i odejmowanie.

Priorytet ten wynika z hierarchii działań arytmetycznych, która umożliwia jednoznaczną interpretację wyrażeń. Na przykład w wyrażeniu a + b x c, zgodnie z zasadami priorytetów, najpierw zostanie obliczone b x c, a następnie wynik tej operacji zostanie dodany do a. Gdyby priorytety były inne, możliwe byłyby różne interpretacje wyrażenia, co prowadziłoby do błędów lub niejednoznacznych wyników.

Dodatkowo, w matematyce kluczową rolę odgrywają nawiasy, które mają najwyższy priorytet i umożliwiają zmianę kolejności wykonywania operacji. Nawiasy pozwalają na "wyprzedzenie" innych operacji, nawet jeśli mają one niższy priorytet. Dzięki nim możemy kontrolować, która operacja zostanie wykonana jako pierwsza w bardziej złożonych wyrażeniach. Na przykład w wyrażeniu (a + b) x c, zgodnie z zasadami matematycznymi, najpierw zostanie obliczone (a + b) wewnątrz nawiasów, a dopiero potem wynik tej operacji zostanie pomnożony przez c.

Aby uwzględnić priorytety podczas obliczeń w systemach liczbowych, stosuje się różne podejścia. Jednym z nich jest analiza wyrażenia i budowa tzw. stosu operatorów, gdzie mnożenie i dzielenie są obsługiwane jako pierwsze, zanim przejdzie się do dodawania i odejmowania. Dodatkowo, nawiasy są traktowane jako wyznaczający granice, w obrębie których operacje są wykonywane w pierwszej kolejności, przed innymi operacjami.

Zastosowanie zasad priorytetów w kalkulatorze pozwala na poprawne interpretowanie i wykonywanie wyrażeń matematycznych, niezależnie od systemu liczbowego, w którym są one zapisywane. Priorytety operatorów, takie jak wyższy priorytet mnożenia i dzielenia w stosunku do dodawania i odejmowania, oraz szczególne traktowanie nawiasów, są niezbędne do zachowania spójności i zgodności z regułami matematycznymi, zapewniając prawidłowe wyniki obliczeń.

<https://www.geeksforgeeks.org/number-system-and-base-conversions/#what-is-number-system->

https://matkaprogramistka.pl/2018/04/03/przeliczanie-systemow-liczbowych/