Dokumentacja do projektu

Kalkulator dla elektroników

Z przedmiotu

Języki programowania

Autor: Mateusz Gąsiorowski

WIET Elektronika

Spis treści

1. Wstęp

Projekt Kalkulatora dla elektroników z obsługą 4 systemów liczbowych, wykonywania prostych obliczeń oraz konwersją liczb między systemami liczbowymi.

Celem projektu jest opracowanie systemu klas, poleceń oraz zaimplementowanie prostego interfejsu użytkownika (poprzez dialog w konsoli).

Całość kodu została napisana w języku C++ w środowisku Visual Studio 2022.

Jednym z głównych założeń projektu była wieloplatformowość (Windows, MacOS, Linux), uniwersalność oraz możliwość rozbudowy pisanego kodu.

2. Plan realizacji

2.1 Obsługa systemów liczbowych

Zmienna liczbowa przechowywana w typie adekwatnych do zastosowania (double lub float) a jej reprezentacje w różnych systemach liczbowych realizowane dynamicznie, gdy zachodzi taka potrzeba.

Reprezentacja wartości liczby w różnych systemach liczbowych celem szybszej implementacji wywoływana jest funkcją w typie std::string, celem bezpośredniego wyświetlenia wartości użytkownikowi.

2.2 Obliczenia matematyczne

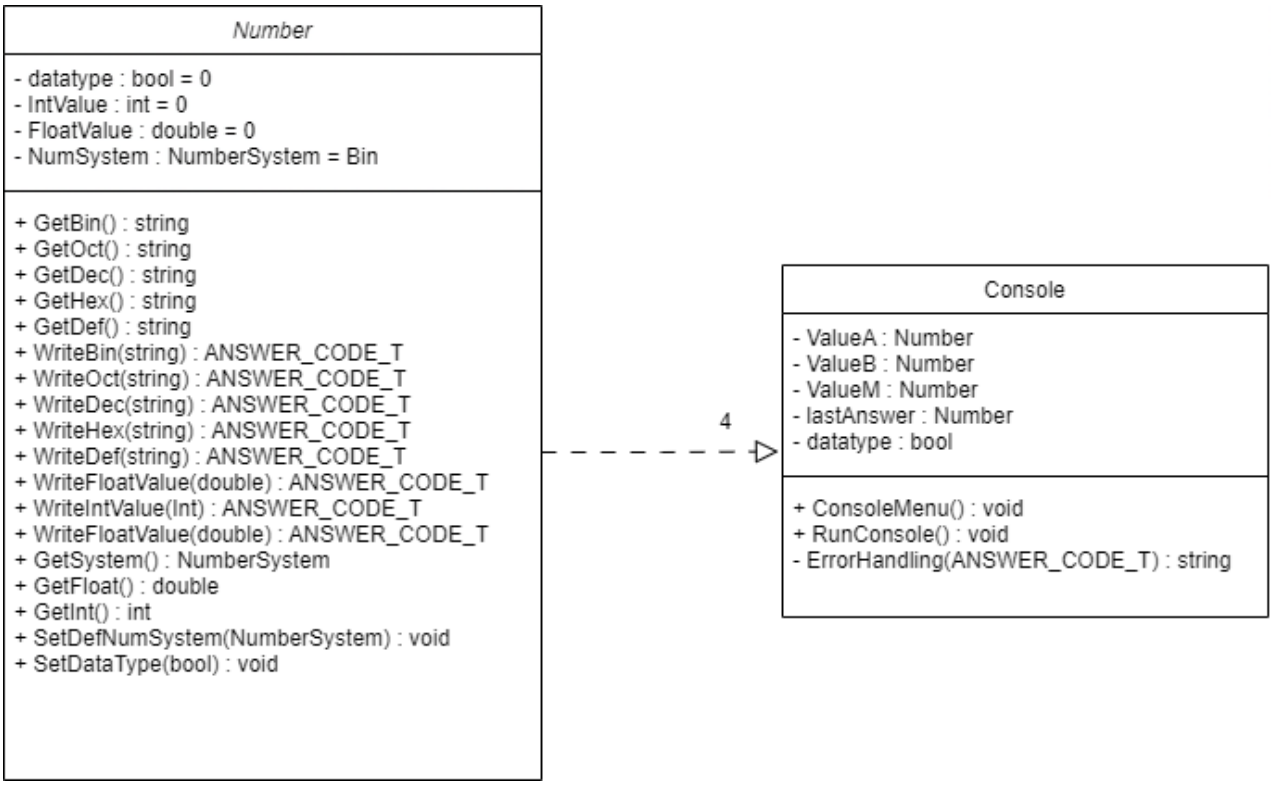
Obsługa podstawowych funkcji dodawanie, odejmowanie, mnożenie, dzielenie dwóch zmiennych A i B. Dodatkowo wspierane przez funkcję Memory oraz możliwość szybkiego przypisania ostatniego wyniku działań do A i B

2.3 Interfejs użytkownika

Prosty dialog konsolowy z wyświetlanym Menu głównym po uruchomieniu programu. Intuicyjne wpisywanie zmiennych (format: A=999), szybkie wykonywanie konwersji między systemami liczbowymi poprzez jednoliterowe komendy:
B – Binarny ; O – Ósemkowy ; D – Dziesiętny ; H – szesnastkowy.

Obsługa ww.. funkcji w możliwie prosty i intuicyjny sposób.

3. Projekt techniczny



Wykorzystanie klas w projekcie.

4. Opis realizacji

4.1 Klasa Number

Wykorzystane biblioteki:

```
#include <string>
#include <stdio.h>
#include <sstream>
#include <format>
```

Typy pomocnicze:

```
typedef enum
{
	ANSWER_CODE_OK,
	ANSWER_CODE_NAN,
	ANSWER_CODE_RANGE,
	ANSWER_CODE_NIN // Not Integer Number
} ANSWER_CODE_T;

enum NumberSystem
{
	Bin = 0,
	Oct,
	Dec,
	Hex
};
```

Zmienne:

```
bool datatype = 0; // 0 - Int ; 1 - Float
int IntValue = 0;
double FloatValue = 0.0;
NumberSystem NumSystem = Bin;
```

Klasa zawiera metody umożliwiające szybką i wygodną pracę nad wartościami liczbowymi z reprezentacjami w różnych systemach liczbowych.

Metody zaimplementowane w klasie pozwalające na wyświetlanie reprezentacji liczby w danym systemie liczbowym konwertują wartość przechowywaną w liczbowym typie danej (double lub int) na string. Pozostałe metody umożliwiające bezpośredni zapis/odczyt wartości lub konwersję wartości podanej w stringu w jednym z podanych systemów liczbowych.

Dodatkowo podczas rozwoju interfejsu użytkownika został dodany atrybut NumSystem – pozwalający na redukcję kodu, przerzucając obsługę systemu liczbowego z poziomu interfejsu – do tej klasy. Za czym stoją następne metody umożliwiające na zapis/odczyt wartości w wcześniej ustalonym systemie liczbowym.

4.2 Metody klasy numer

```
        ////////////////////////////////////////////////////////
        // Get a string of number in Binary system
        ////////////////////////////////////////////////////////
        // OUTPUT:
        //          String contains an number in Binary system
        string GetBin();

        ////////////////////////////////////////////////////////
        //  Converts and save binary number string to value
        ////////////////////////////////////////////////////////
        //
        // INPUT:
        //          input - string containing binnary number
        // OUTPUT:
        //          error codes or OK
        ANSWER_CODE_T WriteBin(string input);

        ////////////////////////////////////////////////////////
        //  Save given floating point value
        ////////////////////////////////////////////////////////
        //
        // INPUT:
        //          Value - floating point value
        // OUTPUT:
        //          error codes or OK
        ANSWER_CODE_T WriteFloatValue(double Value);
```

```
//////////////////////////////////////////////////////////
//  Save given integer value
//////////////////////////////////////////////////////////
//
// INPUT:
//           Value - integer value
// OUTPUT:
//           error codes or OK
ANSWER_CODE_T WriteIntValue(int Value);

//////////////////////////////////////////////////////////
//  Get current selected number system
//////////////////////////////////////////////////////////
//
// OUTPUT:
//           Number system
NumberSystem GetSystem();

//////////////////////////////////////////////////////////
//  Get floating point value
//////////////////////////////////////////////////////////
//
// OUTPUT:
//           Floating point value
double GetFloat();

//////////////////////////////////////////////////////////
//  Get integer value
//////////////////////////////////////////////////////////
//
// OUTPUT:
//           integer value
int GetInt();

//////////////////////////////////////////////////////////
// Set current number system
//////////////////////////////////////////////////////////
//
// INPUT:
//           NSys - name of number system as NumberSystem enum
// OUTPUT:
//           void
void SetDefNumSystem(NumberSystem NSys);

//////////////////////////////////////////////////////////
// Set number types - floating point or integer
//////////////////////////////////////////////////////////
//
// INPUT:
//           n - 0 (integer) or 1 (floating point)
// OUTPUT:
//           void
void SetDataType(bool n);
```

4.3 Konsolowy interfejs użytkownika

Wykorzystując klasę Number w interfejsie znacząco upraszczają się operacje in/out dla użytkownika, dodatkowo umożliwiając szybką konwersje typów między string a liczbowymi i odwrotnie, wykonywanie obliczeń na danych podanych przez użytkownika jest dodatkowo uproszczone.

Interfejs użytkownika jest intuicyjny i szybki.

```
****************************************
*     Calculator for electronics       *
****************************************
* Choose number system:
   'B' - BIN   'O' - OCT
   'D' - DEC   'H' - HEX
* Choose data type: 'F' - Float 'I' - Integers (separete A,B,M)
* Write/read value A or B:
   'A=' or 'B=' ('Number' or 'M'(emory) or 'ANS'(wer))
* Save in M(emory):
   'M+' or 'M-'
* Read/Clear M(emory): 'M' / 'MCLR'
* Math:
   '1' A+B   '2' A-B
   '3' A*B   '4' A/B
```

Obsługa odbywa się poprzez skrócone formy komend.
Kluczową kwestią jest wpisywanie wartości, odbywa się w za pomocą wpisania komendy ‘A=(wartość liczbowa)(enter)” analogicznie dla drugiej zmiennej (w wybranym systemie liczbowym), nie ma potrzeby wpisywania „A= (enter) (wartość) (enter)”. Dodatkowo zamiast wartości możemy skorzystać z pamięci (M) lub przepisania wartości ostatniej odpowiedzi.

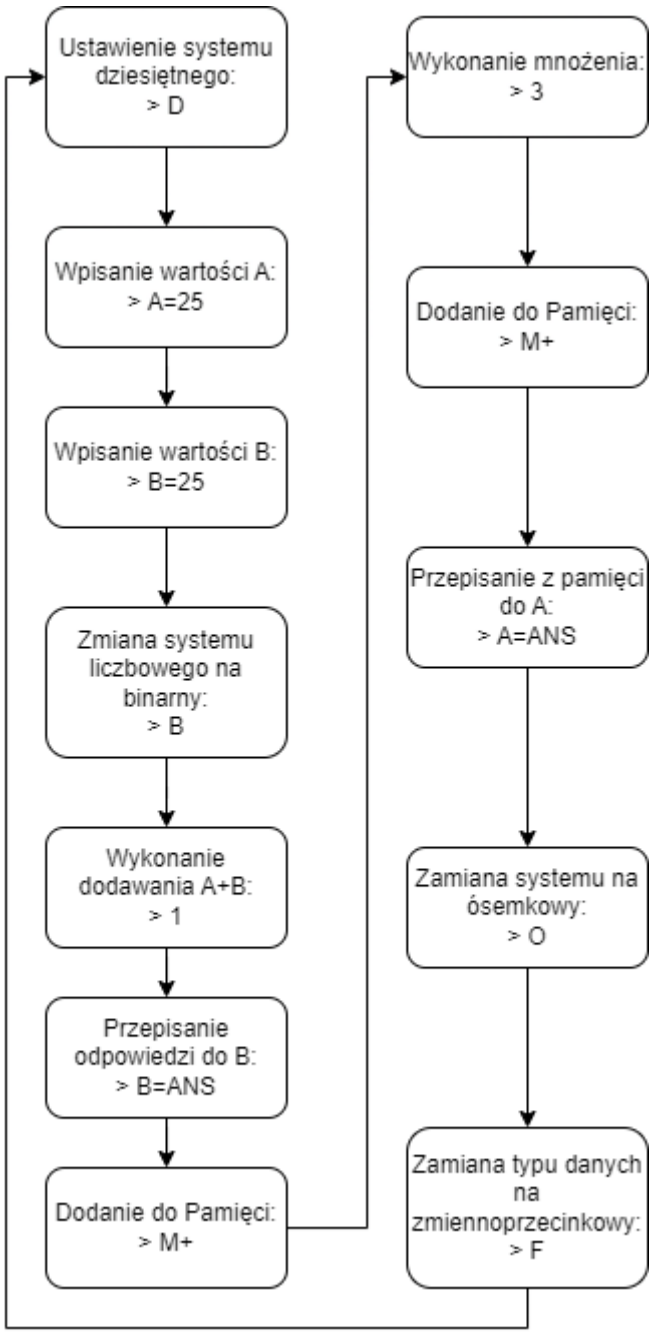
Interfejs również posiada obsługę błędów:

- `NotAnNumber (NAN)` – Wskazujący na wpisanie niepoprawnej wartości lub użycie złego formatowania.
- `NotIntegerNumber (NIN)` – Ukazujący się trybu pracy na liczbach całkowitych, przy wpisaniu liczby zmiennoprzecinkowej
- `OverRange` – Przy przekroczeniu zakresu kalkulatora
- `Unknown Command` – sygnalizujący błędnie wpisaną komendę

Uruchomienie interfejsu (z poziomu kodu) ogranicza się do wywołania funkcji obsługi interfejsu użytkownika.

```
//////////////////////////////////////////////////////
//  Runs console-type UI
//////////////////////////////////////////////////////
void  RunConsole();
```

4.4 Przykładowy ciąg operacji wykonanych na kalkulatorze



5. Bibliografia

1) https://en.cppreference.com/w/
2) https://github.com/BogCyg/BookCPP_PL
3) Prof. Bogusław Cyganek - Programowanie w języku C++. Wprowadzenie dla inżynierów 2023