C++ - Kalkulator obiektowy

Ogólne

Program korzysta z klasy Kalkulator i umożliwia dokonanie prostych podstawowych działań na liczbach rzeczywistych. Wersja rozbudowana zawiera dodatkowo kolorowanie napisów oraz kontrolę błędnych danych.

Uwaga: Nigdy nie wolno kopiować kodu z PDF-a, gdyż zawiera on niewidoczne znaki końca linii i tzw. twarde odstępy. Kod należy przepisać ze zrozumieniem.

Kod - wersja podstawowa (*Windows*)

```
#include <iostream>

using namespace std;

class Kalkulator {
public:
   double dodaj(double a, double b) {
      return a + b;
   }

   double odejmij(double a, double b) {
      return a - b;
   }

   double pomnoz(double a, double b) {
      return a * b;
   }

   double podziel(double a, double b) {
      if (b != 0) {
         return a / b;
      }
       else {
         cout << "BŁĄD:" << endl;
         cout << "W matematyce formalnej nie można dzielić przez (na) zero." << endl;
         return 1;
      }
   }
}; // Koniec klasy
```

```
int main() {
    Kalkulator kalkulator;
    double liczba1, liczba2, wynik;
    char dzialanie;

    cout << "KALKULATOR" << endl;
    cout << "Podaj pierwszą liczbę: ";
    cin >> liczba1;

    cout << "Wybierz działanie (+, -, *, /): ";
    cin >> dzialanie;

    cout << "Podaj drugą liczbę: ";
    cin >> liczba2;

    switch (dzialanie) {
    case '+':
        wynik = kalkulator.dodaj(liczba1, liczba2);
        cout << "Suma liczb " << liczba1 << " oraz " << liczba2 << " wynosi " << wynik <<
        endl;
        break;
    case '-':
        wynik = kalkulator.odejmij(liczba1, liczba2);
        cout << "Różnica między " << liczba1 << " oraz " << liczba2 << " wynosi " << wynik
         << endl;
        break;
    case '*':
        wynik = kalkulator.pomnoz(liczba1, liczba2);
        cout << "Iloczyn liczby " << liczba1 << " oraz " << liczba2 << " wynosi " << wynik <<
         endl;
        break;
    case '/':
        wynik = kalkulator.podziel(liczba1, liczba2);
        cout << "Iloraz liczby " << liczba1 << " oraz " << liczba2 << " wynosi " << wynik <<
         endl;
        break;
    default:
        cout << "BŁĄD:" << endl;
        cout << "Nie wybrano prawidłowego działania." << endl;
        return 0.00000000;
    }
    cout << "Naciśnij ENTER, aby zakończyć..." << endl;
    system("pause > nul");
    return 0;
}
```

Kod - wersja rozbudowana (*Windows*)

```
#include <iostream> // Potrzebne do cout / cin
#include <algorithm> // Potrzebne do find()
#include <windows.h> // Potrzebne do kolorowania napisów; działa tylko w Windows.

using namespace std;

// Stworzenie klasy Kalkulator, która będzie posiadać publiczne metody "dodaj()", "odejmij()",
// "pomnoz()", "podziel()".
class Kalkulator {
public:
    double dodaj(double a, double b) {
        return a + b;
    }

    double odejmij(double a, double b) {
        return a - b;
    }

    double pomnoz(double a, double b) {
        return a * b;
    }

    double podziel(double a, double b) {
        if (b != 0) {
            return a / b;
        }
         else {
            cout << "\n\n\033[0;37;41mBŁĄD:\033[0m" << endl;
            cout << "\033[1;31;40mW matematyce formalnej nie można dzielić przez (na) zero.
            \033[0m" << endl;
            return 1;
        }
    }
};// Koniec klasy



int main() {
    // Jeśli Twój Windows obsługuje kolorowe znaki ANSI w Wierszu poleceń, możesz poniższy
    // blok kodu pominąć:
    // ============== KOLOROWE ZNAKI ANSI ====================
    // Na początek naprawiamy Windowsa, aby kolorował czcionki w konsoli.
    // Tworzymy tzw. uchwyt do tego, co będzie pojawiać się na konsoli (do bufora konsoli):
    HANDLE konsola = GetStdHandle(STD_OUTPUT_HANDLE);
    // Aktywujemy virtualny terminal i to, co będzie się na nim pojawiać:
    #ifndef ENABLE_VIRTUAL_TERMINAL_PROCESSING // Jeśli nie jest zdefiniowany, to:
```

```
#define ENABLE_VIRTUAL_TERMINAL_PROCESSING 0x0004
#endif

// Aby powyższe działało, musimy aktywować i to, co poniżej.
// Jeśli nie jest zdefiniowany, to:
#ifndef ENABLE_PROCESSED_OUTPUT
#define ENABLE_PROCESSED_OUTPUT 0x0001
#endif

// Wartość trybu (intup lub output). Słowo "dw" to skrót od "Display Window",
// jest to jednak nazwa zmiennej, i może być inna:
DWORD dwMode = 0;
dwMode |= ENABLE_PROCESSED_OUTPUT | ENABLE_VIRTUAL_TERMINAL_PROCESSING;
SetConsoleMode(konsola, dwMode);
// ====================== KONIEC BLOKU ====================

// Nazwa klasy "Kalkulator" to konstruktor, a "kalkulator" to obiekt klasy.
// Obiekt jest instancją klasy i w przeciwieństwie do niej, jest alokowany w pamięci.
// Nie mamy bezpośredniego dostępu do klasy, lecz jedynie do obiektu (instancji klasy).
Kalkulator kalkulator;
double liczba1, liczba2, wynik;
char dzialanie;
string wartoscUzytkownika;
bool wprowadzonoLiczbe1 = false; // W C++ zmienne logiczne nie mają domyślnej wartości
bool wprowadzonoLiczbe2 = false;
bool wprowadzonoDzialanie = false;

cout << "\n\n\033[1;36;40m========== KALKULATOR ==========\033[0m\n\n";

// Uwaga: Przypisanie wartości typu "string" do zmiennej "double", daje 0.
while(!wprowadzonoLiczbe1) {
   cout << "Podaj pierwszą liczbę: ";
   cin >> wartoscUzytkownika;
   try {
      // Próbujemy zamienieć stringa na liczbę i przypisać jej typ double.
      // Jeśli się to uda, to znaczy, że wprowadzona wartosc jest typu double.
      liczba1 = stod(wartoscUzytkownika); // stod = String-to-Double
      wprowadzonoLiczbe1 = true;
   } catch (const invalid_argument& exc) {
      cout << "\n\n\033[0;37;41mBŁĄD:\033[0m" << endl;
      cout << "\033[1;31;40mTo nie jest liczba! Spróbuj jeszcze raz.\033[0m\n\n" << endl;
   }
}

while(!wprowadzonoDzialanie) {
   // Obsłużymy dzielenie także wtedy, gdy użytkownik wpisze nieprawidłowy znak ukośnika
   // (piąta pozycja w tablicy).
   // Zauważ, że pozycja ta zawiera dodatkowy ukośnik - to jest tzw. znak "ucieczki".
```

```
    // Ostatnia zaś pozycja (\0) to inicjacja tablicy.
    char tablicaDzialan[6] = {'+', '-', '*', '/', '\\', '\0'};
    cout << "Wybierz działanie (+, -, *, /): ";
    cin >> dzialanie;
    bool jestObecne = find(begin(tablicaDzialan),end(tablicaDzialan),dzialanie) !=
    end(tablicaDzialan);
    if (jestObecne) {
        wprowadzonoDzialanie = true;
    }
     else {
        cout << "\n\n\033[0;37;41mBŁĄD:\033[0m" << endl;
        cout << "\033[1;31;40mTo nie jest poprawne działanie! Spróbuj jeszcze raz.
        \033[0m\n\n" << endl;
    }
}

// Uwaga: Przypisanie wartości typu "string" do zmiennej "double", daje 0.
while(!wprowadzonoLiczbe2) {
    cout << "Podaj drugą liczbę: ";
    cin >> wartoscUzytkownika;
    try {
        // Próbujemy zamienieć stringa na liczbę i przypisać jej typ double.
        // Jeśli się to uda, to znaczy, że wprowadzona wartosc jest typu double.
        liczba2 = stod(wartoscUzytkownika); // stod = String-to-Double
        wprowadzonoLiczbe2 = true;
    } catch (const invalid_argument& exc) {
        cout << "\n\n\033[0;37;41mBŁĄD:\033[0m" << endl;
        cout << "\033[1;31;40mTo nie jest liczba! Spróbuj jeszcze raz.\033[0m\n\n" << endl;
    }
}

// W przypadku, gdy zmienna "dzialanie" ma wartość:
switch (dzialanie) {
case '+':
    wynik = kalkulator.dodaj(liczba1, liczba2);
    cout << "\n\nSuma liczb " << liczba1 << " oraz " << liczba2 << " wynosi
     \033[1;32;40m" << wynik << "\033[0m.\n\n";
    break;
case '-':
    wynik = kalkulator.odejmij(liczba1, liczba2);
    cout << "\n\nRóżnica między " << liczba1 << " oraz " << liczba2 << " wynosi
     \033[1;32;40m" << wynik << "\033[0m.\n\n";
    break;
case '*':
    wynik = kalkulator.pomnoz(liczba1, liczba2);
    cout << "\n\nIloczyn liczby " << liczba1 << " oraz " << liczba2 << " wynosi
     \033[1;32;40m" << wynik << "\033[0m.\n\n";
    break;
```

```
    // Poniższą instrukcję należy traktować jako alternatywę:
    case '/':
    case '\\':
        if (liczba2 != 0) {
            wynik = kalkulator.podziel(liczba1, liczba2);
            cout << "\n\nIloraz liczby " << liczba1 << " oraz " << liczba2 << " wynosi
            \033[1;32;40m" << wynik << "\033[0m.\n\n";
        }
        else {
            cout << "\n\n\033[0;37;41mBŁĄD:\033[0m" << endl;
            cout << "\033[1;31;40mW matematyce formalnej nie można dzielić przez
            (na) 0.\033[0m\n\n" << endl;
        }
        break;
    // To się raczej nie wydarzy, bo pętle while wyłapują nieprawidłowe dane,
    // ale trzeba to zdefiniować:
    default:
        cout << "\n\n\033[0;37;41mBŁĄD:\033[0m" << endl;
        cout << "\033[1;31;40mNie wybrano prawidłowego działania.\033[0m\n\n" << endl;
        return 1;
    }
    cout << "Naciśnij ENTER, aby zakończyć..." << endl;
    system("pause > nul");
    return 0;
}
```

Kod - wersja rozbudowana (Linux)

```
#include <iostream> // Potrzebne do cout / cin
#include <algorithm> // Potrzebne do find()

using namespace std;

// Stworzenie klasy Kalkulator, która będzie posiadać publiczne metody "dodaj()", "odejmij()",
// "pomnoz()", "podziel()".
class Kalkulator {
public:
    double dodaj(double a, double b) {
        return a + b;
    }

    double odejmij(double a, double b) {
        return a - b;
    }

    double pomnoz(double a, double b) {
        return a * b;
```

```
    }

    double podziel(double a, double b) {
        if (b != 0) {
            return a / b;
        }
         else {
            cout << "\n\n\033[0;37;41mBŁĄD:\033[0m" << endl;
            cout << "\033[1;31;40mW matematyce formalnej nie można dzielić przez (na) zero.
            \033[0m" << endl;
            return 0;
        }
    }
};// Koniec klasy


int main() {
    // "Kalkulator" to konstruktor, a "kalkulator" to obiekt klasy.
    // Obiekt jest instancją klasy i w przeciwieństwie do niej, jest alokowany w pamięci.
    // Nie mamy bezpośredniego dostępu do klasy, lecz jedynie do obiektu (instancji klasy).
    Kalkulator kalkulator;
    double liczba1, liczba2, wynik;
    char dzialanie;
    string wartoscUzytkownika;
    bool wprowadzonoLiczbe1 = false; // W C++ zmienne logiczne nie mają domyślnej wartości
    bool wprowadzonoLiczbe2 = false;
    bool wprowadzonoDzialanie = false;

    cout << "\n\n\033[1;36;40m========== KALKULATOR ==========\033[0m\n\n";

    // Uwaga: Przypisanie wartości typu "string" do zmiennej "double", daje 0.
    while(!wprowadzonoLiczbe1) {
        cout << "Podaj pierwszą liczbę: ";
        cin >> wartoscUzytkownika;
        try {
            // Próbujemy zamienieć stringa na liczbę i przypisać jej typ double.
            // Jeśli się to uda, to znaczy, że wprowadzona wartosc jest typu double.
            liczba1 = stod(wartoscUzytkownika); // stod = String-to-Double
            wprowadzonoLiczbe1 = true;
        } catch (const invalid_argument& exc) {
            cout << "\n\n\033[0;37;41mBŁĄD:\033[0m" << endl;
            cout << "\033[1;31;40mTo nie jest liczba! Spróbuj jeszcze raz.\033[0m\n\n" << endl;
        }
    }

    while(!wprowadzonoDzialanie) {
        // Obsłużymy dzielenie także wtedy, gdy użytkownik wpisze nieprawidłowy znak ukośnika
        // (piąta pozycja w tablicy).
```

```
    // Zauważ, że pozycja ta zawiera dodatkowy ukośnik - to jest tzw. znak "ucieczki".
    // Ostatnia zaś pozycja (\0) to inicjacja tablicy.
    char tablicaDzialan[6] = {'+', '-', '*', '/', '\\', '\0'};
    cout << "Wybierz działanie (+, -, *, /): ";
    cin >> dzialanie;
    bool jestObecne = find(begin(tablicaDzialan),end(tablicaDzialan),dzialanie) !=
     end(tablicaDzialan);
    if (jestObecne) {
        wprowadzonoDzialanie = true;
    }
     else {
        cout << "\n\n\033[0;37;41mBŁĄD:\033[0m" << endl;
        cout << "\033[1;31;40mTo nie jest poprawne działanie! Spróbuj jeszcze raz.
        \033[0m\n\n" << endl;
    }
}


// Uwaga: Przypisanie wartości typu "string" do zmiennej "double", daje 0.
while(!wprowadzonoLiczbe2) {
    cout << "Podaj drugą liczbę: ";
    cin >> wartoscUzytkownika;
    try {
        // Próbujemy zamienieć stringa na liczbę i przypisać jej typ double.
        // Jeśli się to uda, to znaczy, że wprowadzona wartosc jest typu double.
        liczba2 = stod(wartoscUzytkownika); // stod = String-to-Double
        wprowadzonoLiczbe2 = true;
    } catch (const invalid_argument& exc) {
        cout << "\n\n\033[0;37;41mBŁĄD:\033[0m" << endl;
        cout << "\033[1;31;40mTo nie jest liczba! Spróbuj jeszcze raz.\033[0m\n\n" << endl;
    }
}

// W przypadku, gdy zmienna "dzialanie" ma wartość:
switch (dzialanie) {
case '+':
    wynik = kalkulator.dodaj(liczba1, liczba2);
    cout << "\n\nSuma liczb " << liczba1 << " oraz " << liczba2 << " wynosi
     \033[1;32;40m" << wynik << "\033[0m.\n\n";
    break;
case '-':
    wynik = kalkulator.odejmij(liczba1, liczba2);
    cout << "\n\nRóżnica między " << liczba1 << " oraz " << liczba2 << " wynosi
     \033[1;32;40m" << wynik << "\033[0m.\n\n";
    break;
case '*':
    wynik = kalkulator.pomnoz(liczba1, liczba2);
    cout << "\n\nIloczyn liczby " << liczba1 << " oraz " << liczba2 << " wynosi
```

```
        \033[1;32;40m" << wynik << "\033[0m.\n\n";
        break;
    // Poniższą instrukcję należy traktować jako alternatywę:
    case '/':
    case '\\':
        if (liczba2 != 0) {
            wynik = kalkulator.podziel(liczba1, liczba2);
            cout << "\n\nIloraz liczby " << liczba1 << " oraz " << liczba2 << " wynosi
            \033[1;32;40m" << wynik << "\033[0m.\n\n";
        }
        else {
            cout << "\n\n\033[0;37;41mBŁĄD:\033[0m" << endl;
            cout << "\033[1;31;40mW matematyce formalnej nie można dzielić przez
            (na) 0.\033[0m\n\n" << endl;
        }
        break;
    // To się raczej nie wydarzy, bo pętle while wyłapują nieprawidłowe dane,
    // ale trzeba to zdefiniować:
    default:
        cout << "\n\n\033[0;37;41mBŁĄD:\033[0m" << endl;
        cout << "\033[1;31;40mNie wybrano prawidłowego działania.\033[0m\n\n" << endl;
        return 1;
    }
    cout << "Naciśnij ENTER, aby zakończyć..." << endl;
    // Usuwa z bufora znak \n (w ilości 256) pozostawiony przez wcześniejsze "cin"
    // ("cin" domyślnie zostawia taki znak).
    cin.ignore(256,'\n');
    cin.get();
    // Zamiast cin.get można by użyć system("pause > nul"), ale jest to mniej uniwersalne
    // (działa tylko w Windows).
    return 0;
}
```

Ostatnia aktualizacja: 11 października 2023.