19.05.2025

PROGRAM DO OBLICZANIA PÓL I OBWODÓW FIGUR GEOMETRYCZNYCH

Autor: Franciszek Łabuz kl.1t

SPIS TREŚCI:

1. CEL PROJEKTU

1.1. Opis funkcjonalności aplikacji

Aplikacja jest interaktywnym **kalkulatorem geometrycznym**, który umożliwia użytkownikowi **obliczanie pola powierzchni, obwodu oraz objętości** różnych figur płaskich i brył przestrzennych. Program działa w trybie tekstowym, z prostym i czytelnym menu, które prowadzi użytkownika krok po kroku przez proces obliczeniowy.

Główne funkcje:

- **Wybór figury geometrycznej** z dostępnych opcji:
  - Figury płaskie:
    - Kwadrat
    - Prostokąt
    - Trójkąt
    - Równoległobok
    - Romb
    - Trapez
  - Bryły przestrzenne:
    - Ostrosłup
    - Prostopadłościan
- **Obliczenia:**
  - Dla figur płaskich: możliwość obliczenia **pola powierzchni** lub **obwodu**.
  - Dla brył przestrzennych: możliwość obliczenia **pola całkowitego powierzchni** lub **objętości**.
- **Wprowadzanie danych**: użytkownik podaje odpowiednie wymiary (np. długości boków, wysokość, pole podstawy), które są sprawdzane pod kątem poprawności (np. czy są liczbami dodatnimi).

- **Jednostki miary**: przed każdym obliczeniem użytkownik wybiera jednostkę długości – **mm, cm lub m**, która jest

następnie dołączana do wyniku wraz z odpowiednim potęgowaniem (np. cm^2, cm^3).

- **Obsługa błędów**:
  - Sprawdzenie, czy dane wejściowe są liczbami.
  - Dla trójkąta – dodatkowe sprawdzenie, czy z podanych boków można utworzyć trójkąt (nierówność trójkąta).
- **Opcja ponawiania obliczeń**: po zakończeniu jednego obliczenia użytkownik może zdecydować, czy chce wykonać kolejne obliczenia dla tej samej figury, bez konieczności powrotu do menu głównego.
- **Intuicyjny interfejs tekstowy**: użytkownik porusza się po aplikacji wybierając opcje za pomocą cyfr i liter (T/N).

Przykładowy scenariusz użycia:

1. Użytkownik uruchamia program i z menu wybiera „Prostokąt”.
2. Wybiera, czy chce obliczyć pole, czy obwód.
3. Wprowadza wymagane wymiary (np. długości boków).
4. Program oblicza i wyświetla wynik wraz z jednostką.
5. Program pyta, czy użytkownik chce ponowić obliczenia.
6. Po zakończeniu użytkownik wraca do menu lub wybiera opcję zakończenia programu.

2. Wymagania funkcjonalne

- Program umożliwia **obliczanie pola i obwodu** figur płaskich (kwadrat, prostokąt, trójkąt, równoległobok, romb) oraz **pola całkowitego i objętości** brył przestrzennych (ostrosłup, prostopadłościan).
- Użytkownik wybiera **jednostkę długości** spośród mm, cm lub m, a wyniki wyświetlane są z odpowiednimi potęgami jednostek (² dla pola, ³ dla objętości).

- Aplikacja posiada **interaktywny, tekstowy interfejs** prowadzący użytkownika przez kolejne kroki obliczeń.
- Po każdym obliczeniu można **powtórzyć działanie bez konieczności restartu programu**.
- Program wykonuje **walidację danych wejściowych**, sprawdzając poprawność i zgodność z zasadami geometrii, oraz informuje o błędach, zabezpieczając przed nieprawidłowymi danymi.

3. Wymagania niefunkcjonalne

- Kod jest napisany w sposób **czytelny i przejrzysty**, z wyraźnym podziałem na funkcje odpowiedzialne za konkretne obliczenia i zadania, co ułatwia jego zrozumienie i utrzymanie.
  (poniżej przykład jednej z funkcji obliczającej)

```
void PoleKwadratu() {
    do {
        system("cls");
        kreski();
        cout << "POLE KWADRATU\n";
        string jednostka = wybierzJednostke();
        int a;
        cout << "Podaj dlugosc boku (" << jednostka << "): ";
        if (!wczytajLiczbe(a)) {
            cout << "Niepoprawna wartosc!\n";
            return;
        }
        cout << "Pole kwadratu: " << a * a << " " << jednostka << "^2\n";
        kreski();
    } while (DalszyWybor());
}
```

(W powyższym kodzie zauważamy, że wszystkie nawiasy są na swoim miejscu i dobrze ustawione, przez co przejrzystość i czytelność kodu jest lepsza)

- Struktura programu jest **skalowalna** — dzięki wykorzystaniu funkcji wywoływanych przez wskaźniki istnieje możliwość łatwego dodania nowych figur geometrycznych i ich obliczeń bez większych zmian w kodzie. (wystarczy skopiować jedną z funkcji dla pola i obwodu i zmienić tylko zmienne i wzory)

```
void PoleKwadratu() {
    do {
        system("cls");
        kreski();
        cout << "POLE KWADRATU\n";
        string jednostka = wybierzJednostke();
        int a;
        cout << "Podaj dlugosc boku (" << jednostka << "): ";
        if (!wczytajLiczbe(a)) {
            cout << "Niepoprawna wartosc!\n";
            return;
        }
        cout << "Pole kwadratu: " << a * a << " " << jednostka << "^2\n";
        kreski();
    } while (DalszyWybor());
}
```

(W powyższym przykładzie, gdy np. chcielibyśmy dodać funkcje obliczająca pole koła, wystarczy zmienić wzór i treść tekstową i kolejna figura gotowa ☺

- Program jest **wydajny obliczeniowo**, ponieważ wykonuje proste operacje matematyczne i działa w trybie tekstowym, co minimalizuje wymagania sprzętowe.
- Interfejs użytkownika jest **przejrzysty i intuicyjny**, oparty na menu tekstowym, które prowadzi użytkownika krok po kroku, minimalizując ryzyko błędów i ułatwiając nawigację po programie.

4. Opis technolgi

- **Język programowania:** C++ — popularny, szybki język o dużych możliwościach, idealny do tworzenia aplikacji konsolowych i prostych narzędzi matematycznych. Program został napisany w środowisku programistycznym DEV- C++
- **Środowisko uruchomieniowe:** Konsola systemu Windows — aplikacja działa w trybie tekstowym, co pozwala na szybkie wprowadzanie danych i wyświetlanie wyników bez potrzeby tworzenia graficznego interfejsu użytkownika.
- **Biblioteki standardowe:**

iostream — podstawowa biblioteka do obsługi wejścia i wyjścia (czytanie danych od użytkownika i wyświetlanie wyników).

clocale — pozwala na ustawienie lokalizacji, dzięki czemu program poprawnie wyświetla polskie znaki diakrytyczne (np. ą, ć, ę).

- **Brak dodatkowych zewnętrznych bibliotek:** Projekt korzysta wyłącznie z bibliotek standardowych, co zwiększa jego przenośność i ułatwia kompilację na różnych platformach.
- **Narzędzia:** Kod można kompilować przy użyciu popularnych kompilatorów C++ takich jak GCC, Clang czy MSVC. Do uruchamiania i testowania wykorzystywana jest konsola systemowa.
- **Obsługa systemowa:** Funkcja system("cls") jest wykorzystywana do czyszczenia ekranu konsoli, co poprawia czytelność interfejsu użytkownika.

5. Struktura kodu

🕐 **Główne funkcje programu:**
Program składa się z kilku kluczowych funkcji, które dzielą się na:

- Funkcje obsługujące menu i wybór figur (menu(), Figury-Plaskie(), FiguryPrzestrzenne()),

OPIS NAJWAZNIEJSZYCH FUNKCJI

OPIS FUNKCJI MENU()

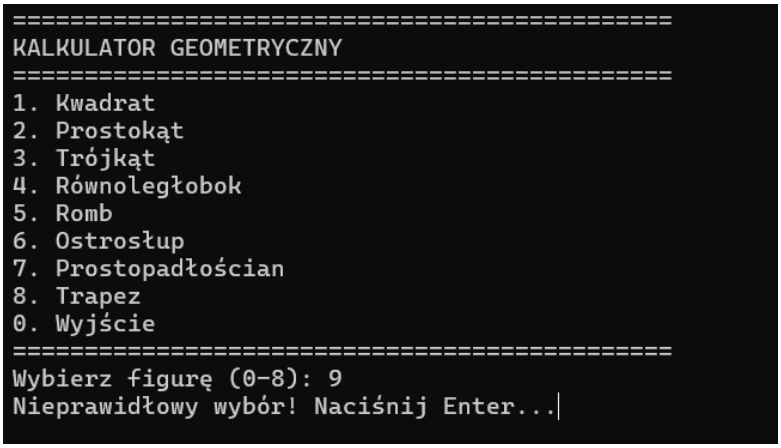
```
1  void menu() {
2      while (true) {
3          system("cls");
4          kreski();
5          cout << "KALKULATOR GEOMETRYCZNY\n";
6          kreski();
7          cout << "1. Kwadrat\n";
8          cout << "2. Prostok�t\n";
9          cout << "3. Tr�jk�t\n";
10         cout << "4. R�wnoleg�obok\n";
11         cout << "5. Romb\n";
12         cout << "6. Ostros�up\n";
13         cout << "7. Prostopad�o�cian\n";
14         cout << "8. Trapez\n";
15         cout << "0. Wyj�cie\n";
16         kreski();
```

Tutaj mamy wybór 8 figur (1-8) i opcje wyjścia (0), która konczy program)

```
1  switch (wybor) {
2          case '1': {
3              FiguryPlaskie("KWADRAT", PoleKwadratu, ObwodKwadratu);
4              break;
5          }
```

Reszta kodu do switch zmiennej „char wybor”, który ma 8 możliwości czyli wyboru figur.

A po wpisaniu innej liczby niż w zakresie od 1-8 wyskakuje:



Po wpisaniu niewłaściwej możliwości, wyskakuje nam określony komunikat i po naciśnięciu przycisku Enter przenosi nam jeszcze raz do menu.

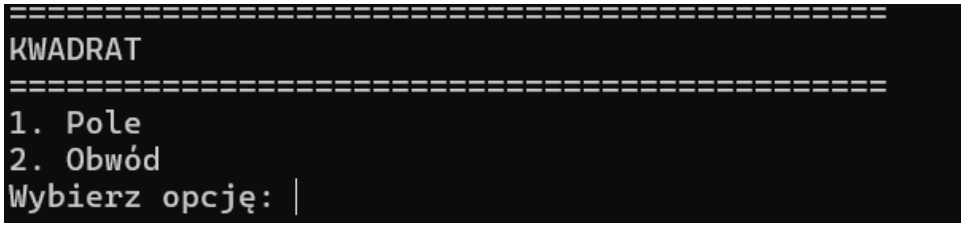
W switchu zauważamy funkcje „FiguryPlaskie()”, która jest kluczowa w naszym programie,

OPIS FUNKCJI FiguryPlaskie()

```
1  void FiguryPlaskie(const string& nazwaFigury, void (*poleFunkcja)(), void (*obwodFunkcja)()) {
2      int wybor;
3      system("cls");
4      kreski();
5      cout << nazwaFigury << "\n";
6      kreski();
7      cout << "1. Pole\n2. Obw\d\n";
8      cout << "Wybierz opcj: ";
9      cin >> wybor;
10
11     if (wybor == 1) {
12         poleFunkcja();
13     } else if (wybor == 2) {
14         obwodFunkcja();
15     } else {
16         cout << "Nieprawid owy wyb r!\n";
17     }
18 }
19
```

Funkcja ta zapewnia, że kod, nie jest tak powtarzalny i czytelny, funkcja ta przyjmuje 4 parametry(nazwę figury, nazwe funkcji obliczającej pole, nazwę funkcji obliczającej obwód

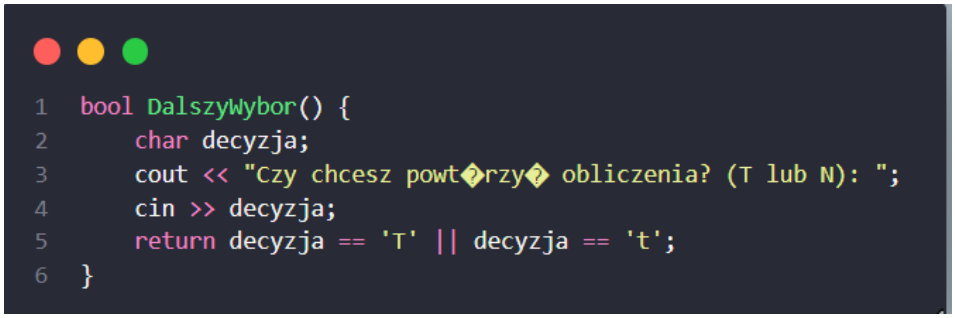
Po wybraniu w menu 1, czyli kwadrat uruchamia się ta funkcja



W switchu w funkcji menu przypisuje się parametry dla tej funkcji i wykonuje się ona.

Funkcja „FiguryPrzestrzenne()" ma indentyczne działania jak funkcja „FiguryPlaskie()"

OPIS FUNKCJI „DalszyWybor()"



```
1  bool DalszyWybor() {
2      char decyzja;
3      cout << "Czy chcesz powt\ufffdrzy\ufffd obliczenia? (T lub N): ";
4      cin >> decyzja;
5      return decyzja == 'T' || decyzja == 't';
6  }
```

Funkcja ta, sprawia, że po zakończonych obliczeniach, mamy wybor, czy powtórzyć program(T), albo zakończyć(N)

(Przykład działania)

```
==========================================
POLE KWADRATU
Wybierz jednostkę:
1. mm
2. cm
3. m
Twój wybór: 2
Podaj długość boku (cm): 34
Pole kwadratu: 1156 cm^2
==========================================
Czy chcesz powtórzyć obliczenia? (T lub N): N

Naciśnij Enter, aby wrócić do menu...
```

```
==============================
OBWÓD KWADRATU
Wybierz jednostkę:
1. mm
2. cm
3. m
Twój wybór:
```

To się dzieje, gdy wybierzemy „T”
(Powtarzają się całe obliczenia danej figury)

- Funkcje obliczające pole i obwód poszczególnych figur (np. PoleKwadratu(), ObwodKwadratu(), PoleProstokata(), itd.),
- Funkcje pomocnicze, takie jak wczytajLiczbe() do walidacji danych, kreski() do wyświetlania linii oddzielających, czy wybierzJednostke() do wyboru jednostek.

OPIS FUNKCJI „Wczytajliczbe()”

```cpp
bool wczytajLiczbe(int &liczba) {
    cin >> liczba;
    if (cin.fail() || liczba <= 0) {
        cin.clear();
        cin.ignore(100, '\n');
        return false;
    }
    return true;
}
```

Funkcja ta przyjmuje jeden parametr „liczba”, funkcja sprawdza czy wprowadzana zmienna w dalszej części kodu jest liczba! i większą od 0!

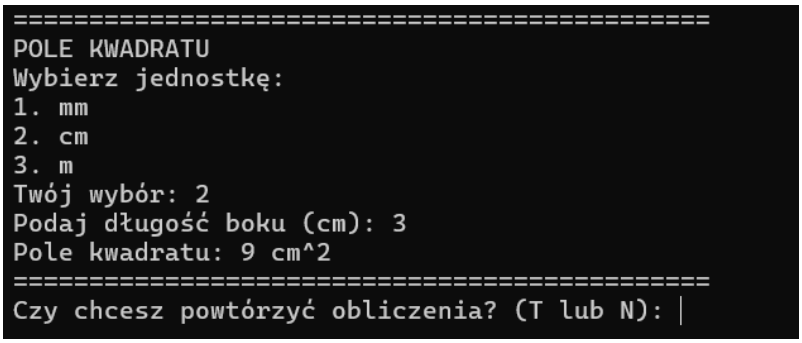
W instrukcji warunkowej sprawdzamy za pomocą „cin.fail” czy dana, którą wpiszemy jest zgodna z typem danych którą zadeklarowaliśmy (int) i czy jest większa od 0, jeżeli wpiszemy nieodpowiednią dane, funkcja zwraca False, jednak gdy wpiszemy dobrze, zwraca wartość True. Instrukcja cin.clear() usuwa błąd wejścia, żeby cin znów działał. cin.ignore(100, '\n') pomija błędne znaki, które zostały w buforze, np. literki po wpisaniu złych danych. Dzięki temu program może znów poprawnie wczytywać dane.

OPIS FUNKCJI „WybierzJednostke()”

```
string wybierzJednostke() {
    int wybor;
    cout << "Wybierz jednostk�:\n";
    cout << "1. mm\n2. cm\n3. m\n";
    cout << "Tw�j wyb�r: ";
    cin >> wybor;

    if (cin.fail()) {
        cin.clear();
        cin.ignore(100, '\n');
        cout << "Nieprawid�owy wyb�r. Ustawiono domy�lnie cm.\n";
        return "cm";
    }

    switch (wybor) {
        case 1: return "mm";
        case 2: return "cm";
        case 3: return "m";
        default:
            cout << "Nieprawid�owy wyb�r. Ustawiono domy�lnie cm.\n";
            return "cm";
    }
}

```

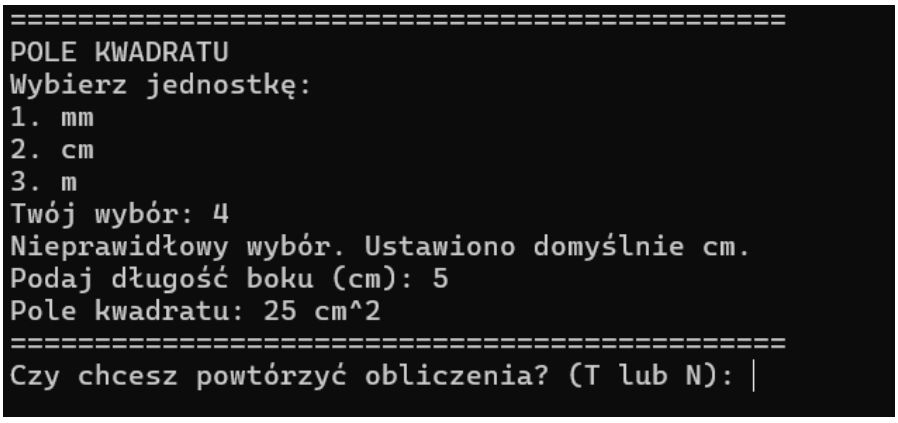
Funkcja ta wybiera jednostkę która wybiera jednostkę miary tutaj też wybieramy liczby od (1-3), które oznaczają jednostke (mm,cm,m), tutaj też zauważamy cin.fail(), który sprawdza czy wprowadzona dana jest zgodna z typem danych, który określiliśmy,

(Poniżej przykład działania tej funkcji gdy wybierzemy już funkcję)



Wybieramy pożądaną jednostkę i potem otrzymujemy wynik w naszej wybranej jednostce. Gdy jednak w wyborze jednostki wpiszemy cokolwiek innego niż 1-3 domyślnie ustawi nam się jednostka cm.

(Przykład)



🕐 **Algorytmy obliczające pola i obwody:**
Każda figura ma przypisane własne algorytmy obliczeniowe bazujące na wzorach matematycznych.

- Pole i obwód kwadratu to odpowiednio a*a i 4*a, gdzie a to długość boku.
- Prostokąt ma pole a*b i obwód 2*(a+b).
- Trójkąt pole liczone jest jako 0.5 * a * h, a obwód jako suma trzech boków z dodatkową kontrolą poprawności danych (sprawdzenie warunku trójkąta).
- Dla brył przestrzennych (ostrosłup, prostopadłościan) obliczane są pola powierzchni oraz objętości zgodnie z odpowiednimi wzorami.

🕐 **Obsługa błędów i walidacja danych wejściowych:**

- Funkcja wczytajLiczbe() zapewnia, że użytkownik wprowadza tylko liczby całkowite dodatnie.
- W przypadku błędnego wprowadzenia dane wejściowe są czyszczone i użytkownik jest o tym informowany.
- Dla trójkąta dodatkowo sprawdzany jest warunek, czy podane boki mogą tworzyć rzeczywistą figurę (warunek nierówności trójkąta).
- W przypadku niepoprawnego wyboru opcji w menu lub wyboru jednostek program wyświetla odpowiednie komunikaty i ustawia wartości domyślne.

6. INTERFEJS UŻYTKOWNIKA

Program działa w **konsoli** i umożliwia użytkownikowi **prostą obsługę poprzez tekstowy interfejs**. Po uruchomieniu wyświetlane jest menu, z którego można wybrać jedną z dostępnych figur geometrycznych, takich **jak kwadrat, prostokąt, trójkąt, romb, trapez, równoległobok, ostrosłup czy prostopadłościan.**

Po wyborze figury użytkownik decyduje**, czy chce obliczyć pole, obwód, objętość lub pole całkowite.** Następnie program prosi o podanie potrzebnych danych **(np. długości boków, wysokości) oraz wybór jednostki miary: mm, cm lub m.**

Wprowadzone dane są sprawdzane – jeśli są błędne, użytkownik otrzymuje komunikat i może spróbować ponownie. Po podaniu poprawnych wartości program oblicza wynik i wyświetla go z odpowiednią jednostką. **Na końcu użytkownik może powtórzyć obliczenia lub wrócić do menu głównego.**

Interfejs jest przejrzysty, a **obsługa programu intuicyjna i uporządkowana.**

7. TESTOWANIE I WALIDACJA

W programie zastosowano różne **metody testowania** w celu zapewnienia **poprawności obliczeń**. Każda funkcja obliczeniowa była sprawdzana na podstawie znanych wzorów matematycznych oraz przykładowych danych, aby upewnić się, że wyniki są zgodne z oczekiwaniami. **(Funkcja WprowadzLiczbe();)**

Program zawiera **zabezpieczenia** przed wprowadzeniem **niepoprawnych danych**. Mechanizmy te obejmują sprawdzanie, czy dane są liczbami dodatnimi oraz obsługę błędów wejścia (np. czyszczenie i ignorowanie błędnych wartości w strumieniu wejściowym). W przypadku wykrycia błędów użytkownik jest informowany i ma możliwość ponownego wprowadzenia danych. **(Funkcja WprowadzLiczbe();)**

Przeprowadzono również **testy interfejsu użytkownika**, które sprawdzają czy program poprawnie reaguje na wybory użytkownika, czy menu jest czytelne i czy wszystkie opcje działają zgodnie z założeniami. Testy te mają na celu zapewnienie **intuicyjnej i płynnej obsługi** programu.

8. PODSUMOWANIE

Kluczowe cechy aplikacji:

- Intuicyjny i czytelny interfejs tekstowy.
- Możliwość obliczania pola, obwodu i objętości dla różnych figur płaskich i brył przestrzennych.
- Walidacja danych wejściowych zabezpieczająca przed błędami. (Funkcje, które opisałem wcześniej)
- Opcja wyboru jednostek miar (mm, cm, m).
- Powtarzalność obliczeń bez konieczności ponownego uruchamiania programu.

Możliwe przyszłe rozszerzenia i ulepszenia:

- Dodanie nowych figur geometrycznych, np. koło, elipsa.
- Obsługa figur nieregularnych i bardziej złożonych kształtów.
- Implementacja graficznego interfejsu użytkownika (GUI) dla łatwiejszej obsługi.
- Możliwość zapisu wyników do pliku tekstowego lub CSV.
- Dodanie funkcji konwersji między jednostkami miar.
- Wprowadzenie opcji rysowania figur na ekranie.
- Rozbudowa o funkcje edukacyjne, np. wyjaśnienia wzorów i przykładów obliczeń.