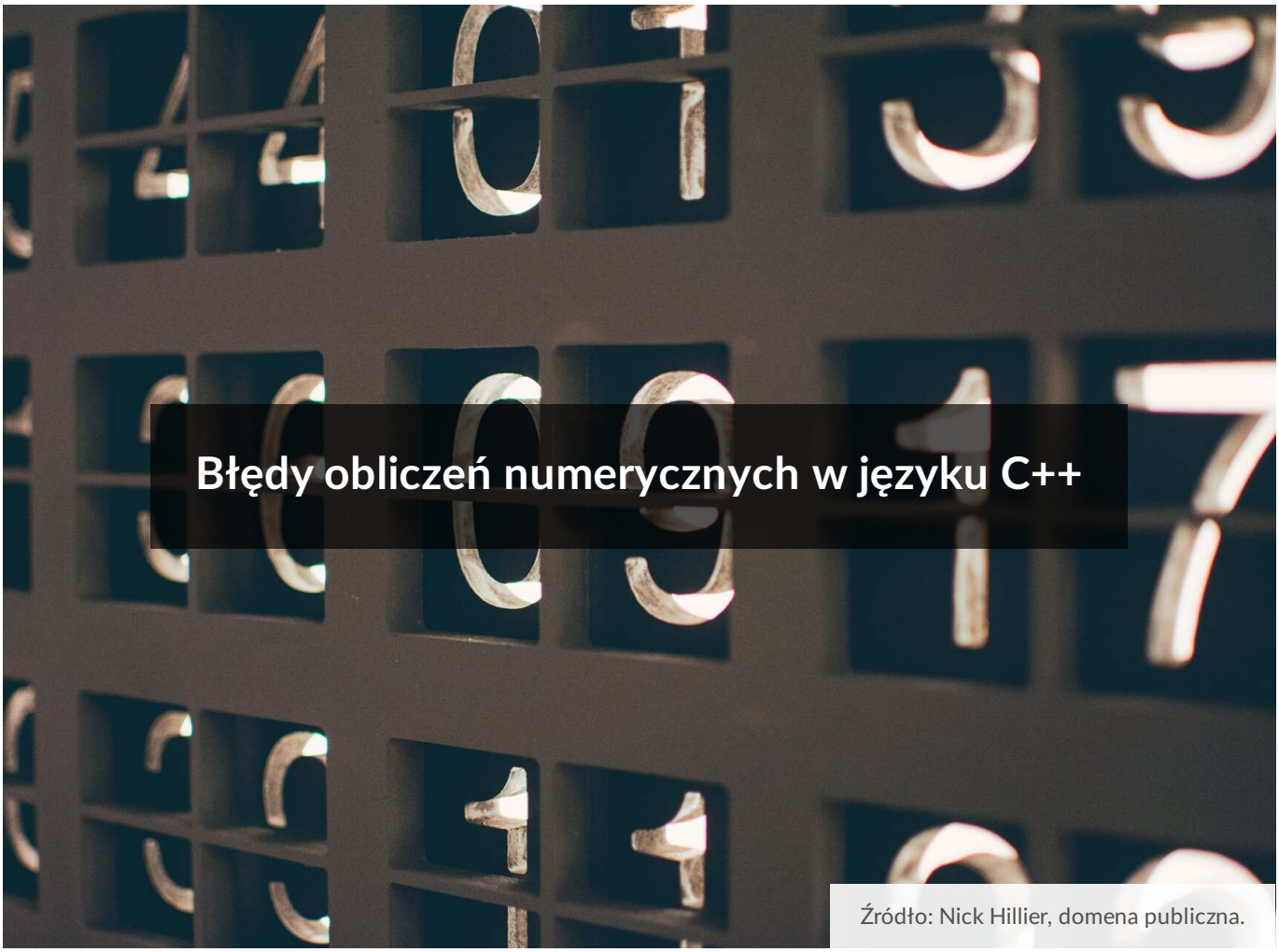
# Błędy obliczeń numerycznych w języku C++


- Wprowadzenie
- Film samouczek
- Przeczytaj
- Sprawdź się
- Dla nauczyciela




Źródło: Nick Hillier, domena publiczna.

W e-materiale Błędy obliczeń numerycznych poznaliśmy najważniejsze informacje dotyczące błędów obliczeń.

W tym e-materiale przyjrzymy się tym zagadnieniom w języku C++.

Ciekawi cię, jak wyglądają omawiane problemy w innych językach programowania? Możesz się z nimi zapoznać w dwóch pozostałych e-materiałach z tej serii:

- Błędy obliczeń numerycznych w języku Java,
- Błędy obliczeń numerycznych w języku Python.

Więcej zadań? Błędy obliczeń numerycznych – zadania maturalne.

> **Twoje cele**
>
> - Powtórzysz wiadomości dotyczące błędów względnych i bezwzględnych oraz ich obliczania.
> - Zaimplementujesz program, który metodą siecznych wyznaczy pierwiastek dla zadanej funkcji.
> - Wykonasz ćwiczenia sprawdzające twoją wiedzę dotyczącą błędów obliczeń.

# Film samouczek

**Ważne!**

Metoda siecznych służy do wyznaczenia przybliżonej wartości pierwiastka funkcji $f$ w badanym przedziale $[a, b]$. W algorytmie metody siecznych cyklicznie wyznacza się wartości funkcji w kolejnych punktach $x_1$ oraz $x_2$. Następnie prowadzi się sieczną (prostą łączącą wartości w punktach) łączącą wartości funkcji w punktach $x_1$ oraz $x_2$. Punkt przecięcia siecznej z osią $OX$ to punkt $x_0$. Jeśli dokładność wyniku jest odpowiednia, punkt ten jest miejscem zerowym. Jeśli chcemy uzyskać dokładniejszy wynik, zapisujemy do punktu $x_2$ wartość $x_1$, a do punktu $x_1$ wartość $x_0$. Wykonujemy kolejne powtórzenie omawianych kroków aż do uzyskania odpowiednio dokładnego wyniku.

Wzór na obliczenie $x_0$:

$$x_0 = x_1 - f\left(x_1\right) \cdot \frac{x_1 - x_2}{f(x_1) - f(x_2)}$$

Aby zastosować metodę siecznych, funkcja musi:

- być ciągła,
- być określona,
- na końcach zadanych przedziałów przyjmować różne znaki.

Metoda siecznych należy do **metod numerycznych** oraz określa **przybliżoną** wartość pierwiastka funkcji, obarczona jest zatem potencjalnymi **błędami obliczeń numerycznych**.

**Problem 1**

Napisz program, który metodą siecznych wyznaczy pierwiastek $x_0$ dla zadanej funkcji. Przyjmij, że pierwiastek wyznaczany jest dla przedziału z początkiem w punkcie `punktStartowy` oraz z końcem w punkcie `punktKoncowy`, a także kolejnych, coraz mniejszych przedziałów, aż do spełnienia warunków:

- | `punktStartowy` – `punktKoncowy` | ≤ `epsilon`
- | $f(x_0)$ | < `epsilon0`

gdzie zmienne `epsilon` i `epsilon0` określają dokładność wyznaczania pierwiastka.

Za początek każdego nowego przedziału (przechowywanego w zmiennej `punktStartowy`) przyjmujemy punkt końcowy poprzedniego przedziału, a za koniec (przechowywany w zmiennej `punktKoncowy`) ostatni wyznaczony pierwiastek funkcji.

Swój program przetestuj dla funkcji $f\left(x\right)=x^2-2$, wartości zmiennych `epsilon` oraz `epsilon0` równych odpowiednio 0,01 i 0,0001, a także przedziału liczbowego $\left[\ 1,\ 1{,}5\ \right]$.

**Specyfikacja:**

*Dane:*

- `punktStartowy` - początek badanego przedziału; liczba rzeczywista
- `punktKoncowy` - koniec badanego przedziału; liczba rzeczywista
- `epsilon` - dokładność wyznaczania pierwiastka; liczba rzeczywista
- `epsilon0` - dokładność porównania wartości funkcji dla wytypowanego pierwiastka z zerem; liczba rzeczywista

*Wynik:*

Program na standardowym wyjściu zwraca wyznaczony metodą siecznych pierwiastek $x_0$ zadanej funkcji.

```cpp
#include <iostream>
#include <cmath>
using namespace std;

// Miejsce na wprowadzenie kodu

int main() {
    double punktStartowy = 1;
    double punktKoncowy = 1.5;
    double epsilon = 0.01;
    double epsilon0 = 0.0001;
    // Miejsce na wprowadzenie kodu
}
```

```
1
```

**Polecenie 1**

Porównaj swoje rozwiązanie z filmem przedstawiającym przykładową implementację algorytmu wyznaczania pierwiastka funkcji za pomocą metody siecznych w języku C++.


Film dostępny pod adresem /preview/resource/R1G6LLgS719NX



Film przedstawiający prezentację dotyczącą metod numerycznych oraz implementacji algorytmu pierwiastka kwadratowego.

**Polecenie 2**

Opracuj notatkę podsumowującą najważniejsze informacje przedstawione w filmie.

# Przeczytaj

## Zaokrąglanie liczby rzeczywistej w granicy błędu względnego

Spróbujemy stworzyć program obliczający kolejne zaokrąglenia podanej liczby, za każdym razem do mniejszej liczby miejsc po przecinku. Warunkiem zakończenia operacji będzie przekroczenie ustalonej wartości błędu względnego. W ten sposób otrzymamy największe możliwe zaokrąglenie przy ustalonych warunkach. Takie rozwiązanie jest pożądane w sytuacji, gdy operujemy mniej dokładnymi liczbami, jednak chcemy zminimalizować błąd do ustalonego stopnia.

**Specyfikacja:**

*Dane:*

- `x` – wartość przekazana do zaokrąglenia przez użytkownika; liczba rzeczywista z przedziału (0, 1), posiadająca 7 cyfr po przecinku
- `bladKrytyczny` – wartość błędu względnego, po której osiągnięciu proces zaokrąglania zostanie przerwany; liczba rzeczywista z przedziału (0, 100)

*Wynik:*

Program na standardowym wyjściu wypisuje wyznaczone zaokrąglenie podanej liczby.

Rozważmy próg błędu względnego równy 2%. Liczba 0,511 może zostać zaokrąglona do 0,51 – przez co uzyskamy błąd względny 0,1957%.

$$\frac{|0{,}511 - 0{,}51|}{0{,}511} \times 100\% = \frac{0{,}001}{0{,}511} \times 100\% = 0,001957 \times 100\% = 0,1957\%$$

> **Ważne!**
>
> Zaokrągliliśmy wynik dzielenia.

Liczbę 0,51 możemy dalej zaokrąglić do 0,5, co w stosunku do oryginalnej wartości daje nam błąd 2,153%, który przekracza wartość krytyczną. Z czego wynika, że zaokrąglenie mieszczące się w ustalonym błędzie względnym wynosi 0,51.

$$\frac{|0{,}511 - 0{,}5|}{0{,}511} \times 100\% = \frac{0{,}011}{0{,}511} \times 100\% = 0,02153 \times 100\% = 2,153\%$$

Dobierzmy odpowiednie biblioteki. Wykorzystamy podstawowe operacje matematyczne, jak zaokrąglanie i wyliczanie wartości bezwzględnej, dlatego niezbędna będzie biblioteka `<cmath>`. Będziemy pracować na niewielkich liczbach z przedziału (0, 1) mających siedem cyfr po przecinku. Wyniki należy wyświetlać z odpowiednią precyzją, do czego posłuży

nam funkcja `setprecision` z biblioteki `<iomanip>`. Pełny zbiór bibliotek wygląda następująco:

```
#include <iostream>
#include <cmath>
#include <iomanip>
```

Skorzystamy również z przestrzeni nazw `std`.

```
using namespace std;
```

Kolejnym krokiem jest napisanie funkcji odpowiedzialnej za zaokrąglanie podanej liczby. Będzie ona zawierać dwa argumenty. Jeden to wspomniana liczba, drugi – wartość określająca, do ilu miejsc po przecinku będziemy zaokrąglać. Wykorzystamy funkcję `pow` podnoszącą wartość do podanej potęgi. Jako wykładnik posłuży nam pożądana liczba cyfr po przecinku po operacji zaokrąglenia. Jeżeli chcemy uzyskać jedną cyfrę po przecinku, wówczas niezbędne jest pomnożenie pierwotnej wartości przez $10^1$. Po przeprowadzeniu tego mnożenia, zaokrąglamy otrzymaną liczbę, a następnie dzielimy ją przez $10^1$. Rezultatem tej operacji będzie gotowa, zaokrąglona liczba z jedną cyfrą po przecinku.

```
double zaokraglenie(double liczba, int poPrzecinku) {
    return round(liczba*(pow(10,poPrzecinku)))/(pow(10,poPrzecink
}
```

Zadeklarujmy teraz niezbędne zmienne. Należą do nich opisane w specyfikacji `x` oraz `bladKrytyczny`, a także:

- `xZaokraglony` – zaokrąglona liczba,
- `bladBezwzgledny` – wyliczona wartość błędu bezwzględnego,
- `bladWzgledny` – wyliczona wartość błędu względnego.

```
int main() {
    double x, xZaokraglony, bladKrytyczny, bladBezwzgledny, bladW
}
```

Zmienne `x` oraz `bladKrytyczny` pobierzemy od użytkownika. Pamiętajmy o warunkach, które powinna spełnić liczba przekazana do zaokrąglenia — ma to być liczba rzeczywista z przedziału (0, 1), posiadająca 7 cyfr po przecinku.

```
int main() {
    double x, xZaokraglony, bladKrytyczny, bladBezwzgledny, bladW

    cout << "Podaj wartość liczby do zaokrąglenia z przedziału (0
    cin >> x;
    cout << endl << "Podaj krytyczną wartość błędu: ";
    cin >> bladKrytyczny;
    cout << endl;
}
```

Po pobraniu niezbędnych informacji, możemy przejść do zaokrąglania liczby. Cały proces zrealizujemy za pomocą pętli `for`. W każdym kolejnym wykonaniu pętli zaokrąglona liczba będzie miała jedną cyfrę po przecinku mniej. Aby tego dokonać, jako drugi argument funkcji `zaokraglenie` wykorzystamy wartość `7-i`. Następnie obliczymy wartości błędu bezwzględnego i względnego. Przydatna będzie tu funkcja `fabs` zwracająca wartość bezwzględną liczby.

```
for(int i=1; i<=7; i++) {
    xZaokraglony = zaokraglenie(x, 7-i);
    bladBezwzgledny = fabs(x-xZaokraglony);
    bladWzgledny = (bladBezwzgledny/x)*100;
}
```

Wciąż jednak brakuje nam warunku zatrzymania pętli w związku z przekroczeniem ustalonej wartości błędu. Zrealizujemy go za pomocą instrukcji `if`. Pamiętajmy, że na końcu działania programu wypisana ma zostać ostatnia wartość zaokrąglenia, dla której nie przekroczono warunku krytycznego. Dlatego w wypadku spełnienia warunku pętli nadpiszemy aktualną wartość zaokrąglenia poprzednią wartością.

```
for (int i=1; i<=7; i++) {
    xZaokraglony = zaokraglenie(x, 7-i);
    bladBezwzgledny = fabs(x-xZaokraglony);
    bladWzgledny = (bladBezwzgledny/x)*100;

    if (bladWzgledny > bladKrytyczny) {
        xZaokraglony = zaokraglenie(x, 8-i);
        break;
    }
}
```

Teraz pozostaje już wyłącznie wypisać końcowy wynik, pamiętając o wymogu odpowiedniej precyzji. Cały program prezentuje się następująco:

```
#include <iostream>
#include <cmath>
#include <iomanip>
using namespace std;

double zaokraglenie(double liczba, int poPrzecinku) {
    return round(liczba*(pow(10,poPrzecinku)))/(pow(10,poPrzecink
}

int main() {
    double x, xZaokraglony, bladKrytyczny, bladBezwzgledny, bladW

    cout << "Podaj wartość liczby do zaokrąglenia z przedziału (0
    cin >> x;
    cout << endl << "Podaj krytyczną wartość błędu: ";
    cin >> bladKrytyczny;
    cout << endl;

    for (int i=1; i<=7; i++) {
        xZaokraglony = zaokraglenie(x, 7-i);
        bladBezwzgledny = fabs(x-xZaokraglony);
        bladWzgledny = (bladBezwzgledny/x)*100;

        if (bladWzgledny > bladKrytyczny) {
            xZaokraglony = zaokraglenie(x, 8-i);
            break;
        }
    }

    cout << setprecision(8) << "Ustalona wartość błędu została pr

    return 0;
}
```

Dla danych z omówionego wcześniej przykładu (x wynoszące 0,511, `bladKrytyczny` równy 2%) otrzymujemy wynik 0.51, czyli taki, jakiego się spodziewaliśmy.

Dla danych: `x` równe 0,1234567, `bladKrytyczny` wynoszący 3%, otrzymujemy wynik 0,12 – jest to zaokrąglenie z błędem względnym wynoszącym około 2.7%. Gdybyśmy liczbę zaokrąglili do 0,1, próg błędu zostałby przekroczony.

## Słownik

**błąd bezwzględny**

wyraża bezwzględną różnicę pomiędzy wartością zmierzoną a dokładną; obliczamy go zgodnie ze wzorem:

$$\Delta x = |x - x_0|$$

gdzie $x$ to dokładna wartość, $x_0$ oznacza zmierzoną wartość, a $\Delta x$ – błąd bezwzględny

**błąd względny**

informuje, o ile procent różni się wartość zmierzona od dokładnej; obliczamy go zgodnie ze wzorem:

$$\delta = \frac{|x - x_0|}{x} \cdot 100\% = \frac{\Delta x}{x} \cdot 100\%$$

gdzie $x$ to dokładna wartość, $x_0$ oznacza zmierzoną wartość, $\Delta x$ – błąd bezwzględny, a za pomocą $\delta$ określamy błąd względny

# Sprawdź się

Pokaż ćwiczenia: 

**Ćwiczenie 1**

Napisz program zaokrąglający podaną liczbę `x` tak, aby błąd względny nie przekroczył ustalonej wartości krytycznej `bladKrytyczny`.

Przetestuj działanie programu zaokrąglając liczbę 0,054256, dopóki błąd względny nie przekroczy wartości 5%.

**Specyfikacja:**

*Dane:*

- `x` - zaokrąglana wartość; liczba rzeczywista z przedziału [0, 1]
- `bladKrytyczny` - wartość krytyczna; liczba rzeczywista z przedziału [0, 100]

*Wynik:*

Program zwraca na standardowe wyjście zaokrągloną liczbę.

## Twoje zadania

1. Program ma zaokrąglać podaną liczbę, dopóki błąd względny nie przekroczy ustalonej wartości.

**Ćwiczenie 2**

Korzystając ze zdefiniowanej w programie stałej `PIERWIASTEK_Z_2`, sprawdź, z dokładnością do ilu miejsc po przecinku należy wypisać jej przybliżenie, aby błąd względny między wartością pierwotną a przybliżoną wynosił mniej niż 0,01%. Wypisz wyznaczoną liczbę cyfr.

**Przykład:**

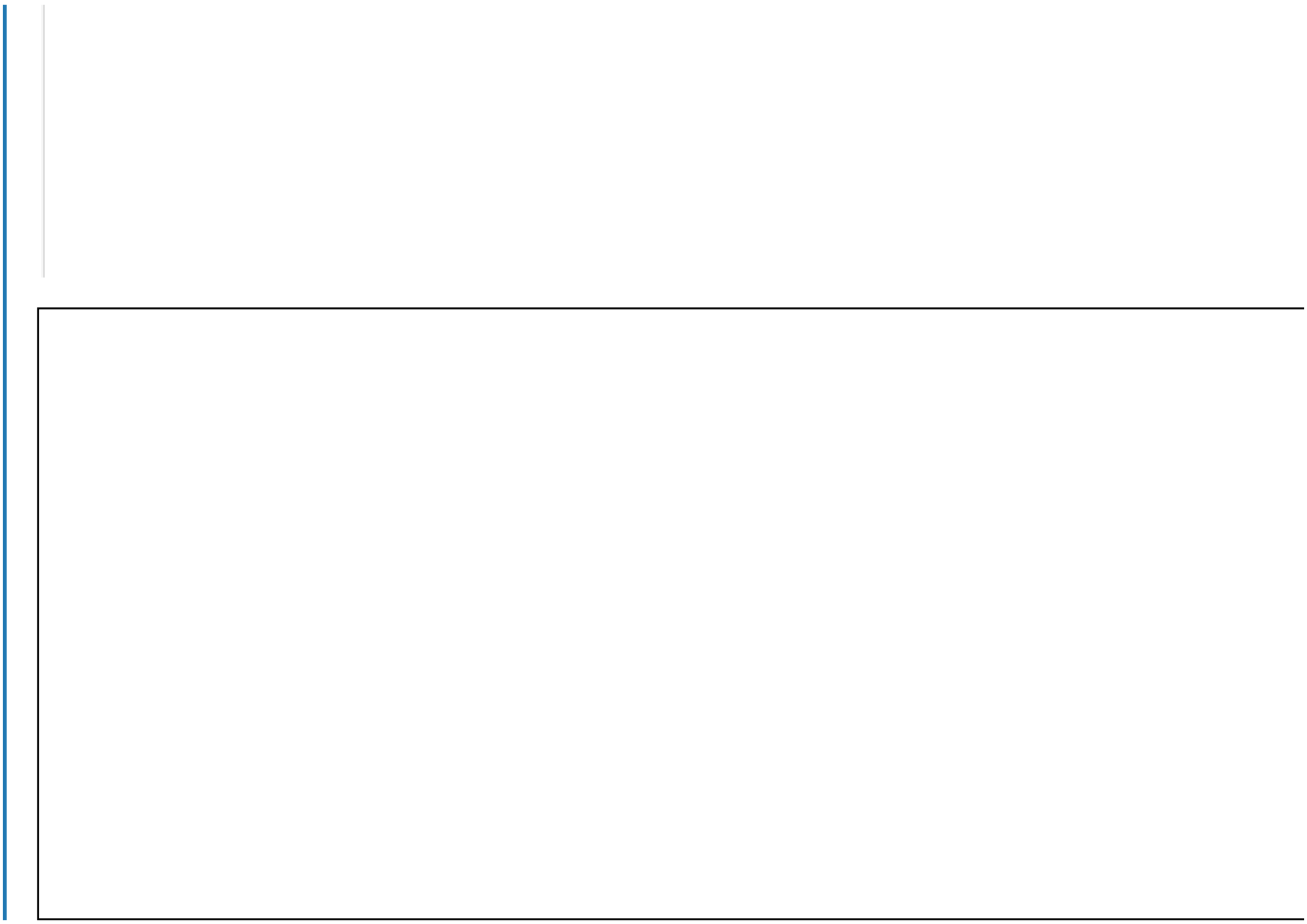
Błąd względny między przybliżeniem pierwiastka z liczby 2 do dwóch cyfr po przecinku a wartością 1,41421356237 wynosi:

$$\frac{1,41421356237 - 1,41}{1,41421356237} \cdot 100\% = 0,29794385247\%$$

Potrzeba zatem przybliżenia do dwóch cyfr po przecinku, aby błąd względny wyniósł mniej niż 0,3%.

## Twoje zadania

1. Program sprawdza, ile cyfr po przecinku potrzeba, aby błąd względny pomiędzy wartością stałej `PIERWIASTEK_Z_2` wynosił mniej niż 0,01%.



**Ćwiczenie 3**

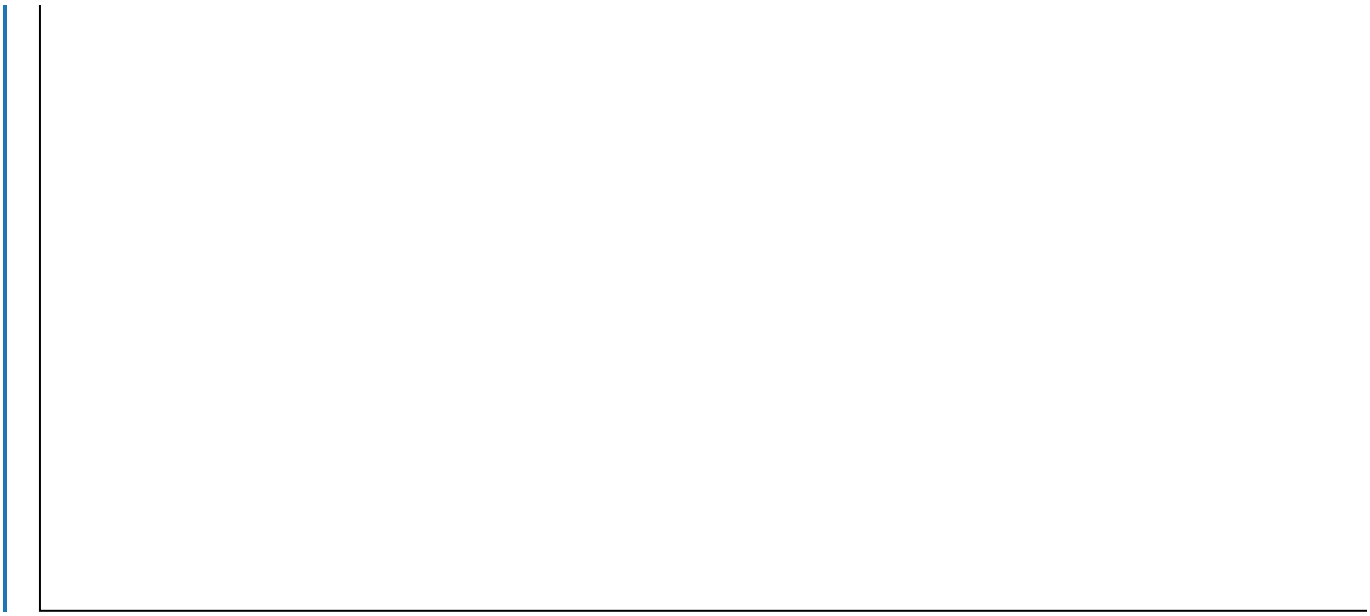
Liczba Eulera może być zdefiniowana przez sumę następującego szeregu:

$$e = \sum_{n=0}^{\infty} \frac{1}{n!} = \frac{1}{0!} + \frac{1}{1!} + \frac{1}{2!} + \dots$$

Sprawdź, dla jakiego $n$ błąd względny wyznaczonego przybliżenia liczby $e$ będzie mniejszy niż 0,0000005. Jako wartość dokładną przyjmij stałą matematyczną M_E. Wypisz minimalną wartość $n$.

## Twoje zadania

1. Program sprawdza, ile wyrazów szeregu trzeba zsumować, aby otrzymać podstawę logarytmu naturalnego na poziomie błędu względnego 0,0000005.



# Dla nauczyciela

**Autor:** Maurycy Gast
**Przedmiot:** Informatyka

**Temat: Błędy obliczeń numerycznych w języku C++**

**Grupa docelowa:**

Szkoła ponadpodstawowa, liceum ogólnokształcące, technikum, zakres rozszerzony

**Podstawa programowa:**

Cele kształcenia – wymagania ogólne

I. Rozumienie, analizowanie i rozwiązywanie problemów na bazie logicznego i abstrakcyjnego myślenia, myślenia algorytmicznego i sposobów reprezentowania informacji.

II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera oraz innych urządzeń cyfrowych: układanie i programowanie algorytmów, organizowanie, wyszukiwanie i udostępnianie informacji, posługiwanie się aplikacjami komputerowymi.

Treści nauczania – wymagania szczegółowe

I. Rozumienie, analizowanie i rozwiązywanie problemów.

Zakres rozszerzony. Uczeń spełnia wymagania określone dla zakresu podstawowego, a ponadto:

7) wyjaśnia, jakie może być źródło błędów pojawiających się w obliczeniach komputerowych: błąd zaokrąglenia, błąd przybliżenia;

II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera i innych urządzeń cyfrowych.

Zakres rozszerzony. Uczeń spełnia wymagania określone dla zakresu podstawowego, a ponadto:

3) sprawnie posługuje się zintegrowanym środowiskiem programistycznym przy pisaniu, uruchamianiu i testowaniu programów;

**Kształtowane kompetencje kluczowe:**

- kompetencje cyfrowe;

- kompetencje osobiste, społeczne i w zakresie umiejętności uczenia się;
- kompetencje matematyczne oraz kompetencje w zakresie nauk przyrodniczych, technologii i inżynierii.

**Cele operacyjne (językiem ucznia):**

- Powtórzysz wiadomości dotyczące błędów względnych i bezwzględnych oraz ich obliczania.
- Zaimplementujesz program, który metodą siecznych wyznaczy pierwiastek dla zadanej funkcji.
- Wykonasz ćwiczenia sprawdzające twoją wiedzę dotyczącą błędów obliczeń.

**Strategie nauczania:**

- konstruktywizm;
- konektywizm.

**Metody i techniki nauczania:**

- dyskusja;
- rozmowa nauczająca z wykorzystaniem multimedium i ćwiczeń interaktywnych;
- ćwiczenia praktyczne.

**Formy pracy:**

- praca indywidualna;
- praca w parach;
- praca w grupach;
- praca całego zespołu klasowego.

**Środki dydaktyczne:**

- komputery z głośnikami, słuchawkami i dostępem do internetu;
- zasoby multimedialne zawarte w e-materiale;
- tablica interaktywna/tablica, pisak/kreda;
- oprogramowanie dla języka C++, w tym kompilator GCC/G++ 4.5 (lub nowszej wersji) i Code::Blocks 16.01 (lub nowszej wersji), Orwell Dev-C++ 5.11 (lub nowszej wersji) lub Microsoft Visual Studio.

**Przebieg lekcji**

**Przed lekcją:**

1. **Przygotowanie do zajęć.** Nauczyciel loguje się na platformie i udostępnia e-materiał „Błędy obliczeń numerycznych w języku C++”. Nauczyciel prosi uczniów o zapoznanie się z treściami w sekcji „Przeczytaj” dotyczącymi programowania.

**Faza wstępna:**

1. Nauczyciel wyświetla uczniom temat, wskazuje cele zajęć oraz ustala z uczestnikami zajęć kryteria sukcesu.
2. **Rozpoznanie wiedzy uczniów.** Nauczyciel zadaje uczniom pytania dotyczące ich aktualnego stanu wiedzy w obszarze poruszanego tematu, opartego o programowanie.

**Faza realizacyjna:**

1. **Praca z multimedium.** Nauczyciel wyświetla zawartość sekcji „Film samouczek". Uczniowie rozwiązują problem nr 1. Następnie wykonują polecenie nr 1.
2. **Praca z tekstem.** Odnosząc się do treści zawartej w sekcji „Przeczytaj", uczniowie przygotowują mapę myśli podsumowującą najważniejsze informacje.
3. **Ćwiczenie umiejętności.** W kolejnym etapie uczniowie dobierają się w pary i wykonują ćwiczenia 1–3 z sekcji „Sprawdź się".

**Faza podsumowująca:**

1. Nauczyciel ponownie wyświetla na tablicy temat i cele lekcji zawarte w sekcji „Wprowadzenie". W kontekście ich realizacji następuje omówienie ewentualnych problemów z rozwiązaniem ćwiczenia z sekcji „Sprawdź się".
2. Wybrany uczeń podsumowuje zajęcia z programowania w języku C++, zwracając uwagę na nabyte umiejętności.

**Praca domowa:**

1. Uczniowie wykonują polecenie nr 2 z sekcji „Film samouczek".

**Materiały pomocnicze:**

- Oficjalna dokumentacja techniczna dla języka C++.
- Oficjalna dokumentacja techniczna dla kompilatora GCC/G++ 4.5 (lub nowszej wersji).
- Oficjalna dokumentacja techniczna dla oprogramowania Code::Blocks 16.01 (lub nowszej wersji), Orwell Dev-C++ 5.11 (lub nowszej wersji) lub Microsoft Visual Studio.

**Wskazówki metodyczne:**

- Treści w sekcji „Film samouczek" można wykorzystać jako podsumowanie i utrwalenie wiedzy uczniów.