

AKADEMIA NAUK STOSOWANYCH W NOWYM SĄCZU

Wydział Nauk Inżynierskich
Katedra Informatyki

DOKUMENTACJA PROJEKTOWA ZAAWANSOWANE PROGRAMOWANIE

Wiliczenie przybliżonej liczby PI metodą całkowania numerycznego przy pomocy czatu GPT

Autor:
Grzegorz Fiejtek

Prowadzący:
mgr inż. Dawid Kotlarski

Nowy Sącz 2025

Spis treści

1. Ogólne określenie wymagań	4
2. Analiza problemu	5
2.1. Co to algorytm sprawdzania podzielności	5
2.2. Zastosowanie	5
2.3. Sposób działania	6
3. Projektowanie	8
3.0.1. Co to jest ChatGPT?	8
3.0.2. zastosowanie	8
3.0.3. Sposób działania	9
3.0.4. Wykorzystanie w projekcie	10
3.1. c++	10
3.1.1. Co to c++	10
3.1.2. zastosowanie	10
3.1.3. Sposób działania	11
3.2. visual studio	11
3.2.1. Co to Visual Studio	11
3.2.2. Zastosowanie	12
3.2.3. sposób działania	12
3.3. doxygen	14
3.3.1. Co to doxygen?	14
3.3.2. zastosowanie	14
3.3.3. sposób działania	15
4. Implementacja	16
4.1. Program	16
4.2. Porównanie wyników	20
5. Wnioski	21
Literatura	23

Spis rysunków	24
Spis listingów	25

1. Ogólne określenie wymagań

Projekt polega na napisaniu wielowątkowego programu, który wyszukuje liczby pierwsze w zadanym przedziale metodą sprawdzania podzielności. Głównym celem jest:

1. Wydzielenie zadanego przedziału liczb (np. od 2 do 2 mld) na mniejsze segmenty i przypisanie ich do określonej liczby wątków (od 1 do nawet 50).
2. Porównanie czasów wykonania dla różnych wielkości zakresów oraz różnych liczb wątków.
3. Zebranie wyników w formie tabelarycznej (arkusz kalkulacyjny), a następnie utworzenie wykresów zależności czasu od liczby wątków. Sprawdzenie poprawności działania programu za pomocą testów jednostkowych (np. Google Test).
4. Zbadanie wydajności i wykorzystania zasobów sprzętowych (obciążenie procesora, liczba wątków, częstotliwość taktowania, użycie pamięci) w różnych wariantach uruchomienia.

Aby sprawdzić, czy dana liczba n jest pierwsza, wykorzystano prostą metodę sprawdzania podzielności:

- Jeżeli $n < 2$, to nie jest ona liczbą pierwszą.
- Jeżeli istnieje jakikolwiek dzielnik d w zakresie od 2 do \sqrt{n} , taki że $n \bmod d = 0$, wówczas n nie jest liczbą pierwszą.
- W przeciwnym wypadku n jest liczbą pierwszą.

Wielowątkowość osiągnięto poprzez równomierne dzielenie zakresu liczb pomiędzy wątki. Każdy wątek sprawdza przydzielony przedział i zlicza, ile liczb pierwszych się w nim znajduje. Pod koniec obliczeń sumujemy wyniki.

2. Analiza problemu

2.1. Co to algorytm sprawdzania podzielności

Algorytm *sprawdzania podzielności* (*trial division*) jest jedną z najprostszych metod umożliwiającą weryfikację, czy dana liczba n jest liczbą pierwszą. Opiera się na następujących krokach:

1. **Sprawdzenie progu minimalnego:** jeżeli $n < 2$, to z definicji nie jest ona liczbą pierwszą i od razu kończymy procedurę.
2. **Poszukiwanie dzielnika w przedziale $[2, \sqrt{n}]$:** dla każdej liczby całkowitej d z przedziału $[2, \lfloor \sqrt{n} \rfloor]$ sprawdzamy, czy dzieli ona liczbę n bez reszty ($n \bmod d = 0$). Jeśli znajdziemy taką liczbę d , to n nie jest liczbą pierwszą, co oznacza, że możemy zakończyć dalsze badanie.
3. **Wniosek o pierwszości:** jeśli w powyższym przedziale nie wystąpił żaden dzielnik, wówczas liczba n jest liczbą pierwszą.

Złożoność obliczeniowa tej metody, rozpatrując pojedynczą liczbę n , wynosi w najgorszym przypadku $\mathcal{O}(\sqrt{n})$. Dla bardzo dużych liczb może to być nieefektywne, dlatego w zastosowaniach wymagających szybkich testów pierwszości (np. w kryptografii) korzysta się z bardziej zaawansowanych i szybszych metod, takich jak Sito Eratostenesa, test Millera–Rabina czy test AKS. Niemniej metoda *trial division* jest intuicyjna i często stosowana w celach edukacyjnych do prostych przykładów.

2.2. Zastosowanie

Algorytm *trial division* może być z powodzeniem stosowany w mniejszych lub edukacyjnych projektach, gdzie zależy nam przede wszystkim na prostocie implementacji i zrozumiałości kodu. Przykładowe obszary zastosowań:

- **Proste programy wyznaczające liczby pierwsze:**

Jeśli zakres badanych liczb nie jest bardzo duży (np. do kilku milionów), algorytm sprawdzania podzielności może wystarczyć do wygenerowania listy liczb pierwszych na potrzeby zadań szkolnych czy akademickich.

- **Weryfikacja poprawności implementacji innych metod:**

Często *trial division* wykorzystuje się jako punkt odniesienia w testach. Można porównać jego wyniki z wynikami bardziej zaawansowanych algorytmów (np.

Sit Eratostenesa, testu Millera–Rabina), aby upewnić się, że nowa metoda nie zwraca błędnych rezultatów.

- **Aplikacje edukacyjne:**

Dzięki przejrzystości i niewielkiej liczbie kroków sprawdzanie podzielności jest idealne do przedstawiania zagadnienia liczb pierwszych w projektach dydaktycznych. Pozwala pokazać etapy pętli sprawdzającej kolejne dzielniki aż do \sqrt{n} .

- **Początkowe etapy wyszukiwania liczb pierwszych w dużym zakresie:**

W niektórych algorytmach można użyć *trial division* do odsiania prostych przypadków (np. dzielenie przez liczby pierwsze z małej puli) zanim zastosuje się kosztowniejsze, zaawansowane testy pierwszości do większych liczb.

W przypadku dużych zakresów lub potrzeby szybkiej weryfikacji (np. w zastosowaniach kryptograficznych) metoda *trial division* nie jest wydajna, dlatego najczęściej korzysta się z bardziej efektywnych algorytmów, takich jak:

- **Sito Eratostenesa** – doskonałe do wyznaczania wszystkich liczb pierwszych w określonym przedziale (szczególnie gdy górna granica przedziału jest znana i nieprzesadnie duża).
- **Test Millera–Rabina** – algorytm probabilistyczny, znacznie szybszy od sprawdzania podzielności, często stosowany w praktyce.
- **Test AKS** – deterministyczny test pierwszości o złożoności wielomianowej, choć w praktyce rzadziej wykorzystywany niż testy probabilistyczne.

Pomimo ograniczeń wydajnościowych, metoda *trial division* bywa więc przydatna wszędzie tam, gdzie kluczowa jest łatwość implementacji i przejrzystość idei.

2.3. Sposób działania

Metoda *trial division* opiera się na prostym założeniu, że jeśli liczba n ma dzielnik mniejszy lub równy \sqrt{n} , to nie jest liczbą pierwszą. Główne kroki algorytmu przedstawiają się następująco:

1. **Warunek początkowy:**

Na początku sprawdzamy, czy $n < 2$. Jeżeli tak, to z definicji n nie jest liczbą pierwszą i kończymy działanie algorytmu.

2. Wyznaczenie górnej granicy sprawdzania:

Obliczamy $k = \lfloor \sqrt{n} \rfloor$. Dalsze sprawdzenia wykonujemy w pętli dla d od 2 do k .

3. Sprawdzanie podzielności:

Wewnątrz pętli sprawdzamy, czy $n \bmod d = 0$. Jeśli tak, to oznacza to, że d jest dzielnikiem n , co natychmiast dowodzi, że n *nie* jest liczbą pierwszą.

4. Decyzja o wyniku:

- Jeżeli w pętli znajdziemy jakikolwiek dzielnik d , który dzieli n bez reszty, algorytm zwraca wynik, że n jest liczbą złożoną.
- Jeżeli żadna z liczb w przedziale $[2, k]$ nie podzieli n bez reszty, uznajemy n za liczbę pierwszą.

Przeanalizujmy to na przykładzie liczby $n = 37$:

- Najpierw obliczamy $\lfloor \sqrt{37} \rfloor = 6$.
- Następnie sprawdzamy kolejne $d \in \{2, 3, 4, 5, 6\}$.
- Ani 2, ani 3, 4, 5 czy 6 nie dzielą 37 bez reszty, więc wniosek jest taki, że 37 jest liczbą pierwszą.

Złożoność czasowa algorytmu w najgorszym przypadku wynosi $\mathcal{O}(\sqrt{n})$, co oznacza, że dla każdej liczby potrzebujemy do \sqrt{n} prób sprawdzenia podzielności. Dlatego właśnie *trial division* jest uznawana za algorytm intuicyjny, ale niezbyt wydajny przy badaniu bardzo dużych liczb.

3. Projektowanie

Chat GPT

3.0.1. Co to jest ChatGPT?

ChatGPT to duży model językowy (z rodziny *Generative Pre-trained Transformer*), opracowany przez firmę OpenAI. Dzięki wykorzystaniu technik głębokiego uczenia maszynowego oraz trenowaniu na ogromnych zbiorach danych pochodzących z Internetu, jest w stanie generować odpowiedzi w wielu językach, w tym po polsku.

3.0.2. zastosowanie

ChatGPT może być wykorzystywany w wielu dziedzinach i scenariuszach. Kilka przykładowych zastosowań to:

- **Tworzenie i redagowanie tekstów** – generowanie artykułów, esejów, opisów produktów czy streszczeń może być wspomagane przez model językowy. Dodatkowo ChatGPT może służyć do korekty i optymalizacji już istniejących treści.
- **Pomoc w programowaniu** – ChatGPT ułatwia tworzenie kodu, wskazuje potencjalne błędy, a także pomaga w wyjaśnianiu trudnych zagadnień dotyczących różnych języków i bibliotek programistycznych.
- **Wsparcie w obsłudze klienta** – w postaci chatbotów w serwisach internetowych lub systemach call center, gdzie jest w stanie odpowiadać na najczęściej zadawane pytania i rozwiązywać typowe problemy klientów.
- **Generowanie pomysłów i inspiracji** – model może służyć do burzy mózgów, sugerowania koncepcji czy tworzenia scenariuszy w obszarach kreatywnych, takich jak pisanie książek, scenariuszy filmowych czy projektów gier.
- **Tłumaczenia językowe** – choć ChatGPT nie jest specjalistycznym tłumaczem, w wielu przypadkach potrafi pomóc w zrozumieniu lub przeformułowaniu tekstu między różnymi językami.
- **Edukacja i nauka** – może pełnić rolę wirtualnego asystenta w rozwiązywaniu zadań, wyjaśnianiu zagadnień i przy nauce języków obcych, prezentując przykłady i wyjaśnienia.

Należy jednak pamiętać, że ChatGPT, jako model statystyczny, może niekiedy popełniać błędy i wymaga weryfikacji udzielanych odpowiedzi, zwłaszcza w krytycznych zastosowaniach zawodowych czy naukowych.

3.0.3. Sposób działania

ChatGPT jest dużym modelem językowym (z rodziny Generative Pre-trained Transformer) stworzonym przez firmę OpenAI. W swojej budowie i działaniu opiera się na kilku kluczowych etapach:

1. Architektura transformera:

Model ChatGPT wykorzystuje *architekturę transformera*, która umożliwia równoległe przetwarzanie sekwencji tekstu. W przeciwieństwie do starszych rozwiązań RNN (ang. *Recurrent Neural Network*), w transformatorach istotną rolę odgrywa mechanizm *self-attention*, pozwalający modelowi wychwytywać zależności między słowami (tokenami) niezależnie od ich wzajemnego położenia w zdaniu.

2. Proces trenowania:

ChatGPT jest *wstępnie wytrenowany* na ogromnych zbiorach danych tekstowych pochodzących z Internetu. W trakcie uczenia model uczy się przewidywać kolejne słowa (tokeny) w sekwencji, co oznacza, że otrzymuje fragment tekstu i próbuje dokończyć go w najbardziej prawdopodobny sposób. Dzięki temu buduje wewnętrzną reprezentację struktur językowych.

3. Dopasowywanie odpowiedzi:

Po zakończeniu wstępnego treningu model przechodzi fazę *fine-tuning* z nadzorem (instrukcjami), co pozwala dostosować go do konkretnych zadań, takich jak konwersacja czy odpowiadanie na pytania. Dodatkowo wykorzystywane są techniki uczenia ze wzmocnieniem z udziałem człowieka (ang. *Reinforcement Learning from Human Feedback*, RLHF), aby zwiększyć trafność i zrozumiałość generowanych odpowiedzi.

4. Generowanie tekstu (inferencja):

W momencie, gdy użytkownik zadaje pytanie lub wprowadza polecenie, ChatGPT analizuje wprowadzone dane (kontekst) i przewiduje najbardziej prawdopodobną sekwencję słów (tokenów), którą powinien wygenerować jako odpowiedź. Proces ten przebiega *autoregresywnie*, co oznacza, że model dobiera kolejne słowa na podstawie poprzednio wygenerowanych i kontekstu wejściowego.

5. Rozumienie i kontekst:

Dzięki *mechanizmowi self-attention* ChatGPT może odnosić się do wcześniejszych słów czy zdań i uchwycić długodystansowe zależności językowe. W praktyce oznacza to, że może pamiętać treść konwersacji czy dłuższych tekstów i na tej podstawie formułować spójniejsze odpowiedzi.

6. Ograniczenia:

ChatGPT, będąc modelem statystycznym, może czasem generować odpowiedzi błędne lub niepełne. Nie posiada także wiedzy o zdarzeniach, których nie uwzględniono w danych treningowych (np. wydarzenia najnowsze), a jego wiedza jest ograniczona do czasu zakończenia zbierania danych. Nie jest też *świadomy* w ludzkim rozumieniu tego słowa — jego działanie sprowadza się do wykorzystywania wzorców nauczonych w procesie trenowania.

3.0.4. Wykorzystanie w projekcie

Chat GPT został wykorzystany w celu stworzenia gotowego kodu ??[1]

3.1. c++

3.1.1. Co to c++

C++ to język programowania ogólnego przeznaczenia, który rozszerza język C o mechanizmy programowania obiektowego, a także inne zaawansowane funkcje. Jego ikona znajduje się na rysunku .3.1

3.1.2. zastosowanie

C++ jest używany w wielu dziedzinach, w tym:

Tworzenie systemów operacyjnych: np. fragmenty systemów takich jak Windows czy Linux.

Aplikacje desktopowe i mobilne: edytory tekstu, programy graficzne, oprogramowanie inżynierskie.

Programowanie gier: ze względu na wydajność i możliwość pracy z grafiką oraz pamięcią.

Systemy wbudowane: np. oprogramowanie dla mikrokontrolerów. Przetwarzanie wysokowydajne



Rys. 3.1. ikona c++

3.1.3. Sposób działania

C++ umożliwia programowanie zarówno proceduralne, jak i obiektowe. Jego cechy obejmują:

Programowanie obiektowe: C++ wprowadza klasy i obiekty, co pozwala na grupowanie danych oraz funkcji w struktury, co ułatwia zarządzanie złożonością kodu.

Dziedziczenie: Możliwość tworzenia nowych klas na podstawie istniejących, co wspiera ponowne używanie kodu.

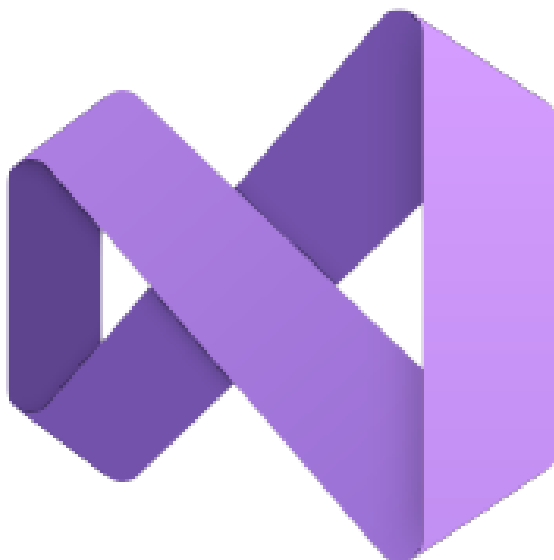
Polimorfizm: Pozwala funkcjom i obiektom przyjmować różne formy w zależności od kontekstu.

Zarządzanie pamięcią: W C++ mamy pełną kontrolę nad pamięcią, co pozwala na dynamiczne alokowanie i zwalnianie zasobów.[\[2\]](#)

3.2. visual studio

3.2.1. Co to Visual Studio

Visual Studio to zintegrowane środowisko programistyczne stworzone przez firmę Microsoft. Jest to narzędzie używane do tworzenia, testowania i debugowania aplikacji w różnych językach programowania, takich jak C++, C#, Python, JavaScript, i wielu innych. Jego ikona znajduje się na rysunku 3.2



Rys. 3.2. Ikona VS

3.2.2. Zastosowanie

Visual Studio jest używane przez programistów do tworzenia różnych typów oprogramowania:

- Aplikacje desktopowe (Windows Forms, WPF),
- Aplikacje webowe (ASP.NET, HTML, CSS, JavaScript),
- Aplikacje mobilne (Xamarin, .NET MAUI),
- Gry (Unity, Unreal Engine),
- Usługi chmurowe (Microsoft Azure).

3.2.3. sposób działania

Visual Studio ułatwia tworzenie oprogramowania, oferując szereg funkcji, które wspierają proces tworzenia kodu:

Edytor kodu: Oferuje wsparcie dla składni wielu języków, podświetlanie kodu, autouzupełnianie, a także sugestie podczas pisania.

Kompilator i debugger: Visual Studio pozwala kompilować kod do plików wykonywalnych i automatycznie wykrywać oraz naprawiać błędy w kodzie.

IntelliSense: Zaawansowana funkcja podpowiedzi podczas pisania kodu, która analizuje kod i proponuje odpowiednie funkcje, metody czy zmienne.

Integracja z systemami kontroli wersji: Visual Studio obsługuje Git, co po-

zwala zarządzać kodem, śledzić zmiany i współpracować z zespołem bezpośrednio z poziomu środowiska IDE.

Debugowanie i profilowanie: Narzędzia do śledzenia błędów oraz wydajności kodu, które pomagają analizować i optymalizować aplikacje.

Szablony projektów: Visual Studio oferuje szereg gotowych szablonów projektów, które ułatwiają rozpoczęcie pracy nad aplikacjami webowymi, desktopowymi czy mobilnymi.[\[3\]](#)

3.3. doxygen

3.3.1. Co to doxygen?

Doxygen to narzędzie do automatycznego generowania dokumentacji technicznej z kodu źródłowego. Obsługuje wiele języków programowania, takich jak C, C++, Java, Python, Fortran, PHP i inne. Doxygen skanuje kod źródłowy i na podstawie specjalnych komentarzy generuje szczegółową dokumentację w formacie HTML, PDF, LaTeX, czy RTF. Jego ikona znajduje się na rysunku 3.3



Rys. 3.3. Ikona doxygen

3.3.2. zastosowanie

Doxygen jest szeroko stosowany w projektach programistycznych, aby generować dokumentację bezpośrednio na podstawie kodu. Jest szczególnie popularny w zespołach pracujących z językami C i C++, ale także w innych środowiskach, które wymagają dokumentacji technicznej. Przykładowe zastosowania:

Dokumentacja bibliotek i API: Umożliwia automatyczne tworzenie dokumentacji dla publicznych interfejsów, funkcji, klas, metod i zmiennych.

Projekty open-source: Doxygen jest popularny w projektach open-source, ponieważ pozwala na łatwe udostępnienie dokumentacji użytkownikom i programistom pracującym nad kodem.

Zarządzanie dużymi projektami: Dzięki automatycznemu generowaniu dokumentacji, Doxygen pomaga programistom zrozumieć struktury dużych i złożonych projektów.

3.3.3. sposób działania

Doxygen działa poprzez analizowanie specjalnych komentarzy w kodzie, które są oznaczane za pomocą znaczników podobnych do tych w języku HTML. Po uruchomieniu Doxygen generuje dokumentację na podstawie tych komentarzy oraz samego kodu, opisując struktury takie jak klasy, funkcje, zmienne i ich relacje.[\[4\]](#)

4. Implementacja

4.1. Program

```

1  /**
2   * @file main.cpp
3   * @brief PrzykŁad caŁkowania funkcji \f$f(x) = \sqrt{1 - x^2}\f$
4   *       metod prostokŁt w
5   *       oraz wyliczania przybli enia liczby \f$\pi\f$ z
6   *       wykorzystaniem wielow Łtkowo Źci.
7   *
8   * Program tworzy T w Łtk w , z kt rych ka dy oblicza cz Ź Ź
9   * sumy sk Ładowej ca Łki na
10  * wybranym przedziale. Nast Źpnie wyniki te s sumowane, aby
11  * otrzyma warto Ź ca Łki.
12  * Ca Łkowany jest fragment okr Źgu jednostkowego zdefiniowany
13  * przez r wnanie
14  * \f$y = \sqrt{1 - x^2}\f$ w przedziale [0, 1].
15  * Przyk Ładowe u ycie:
16  * @code
17  *   ./program
18  *   Podaj liczbe podprzedzialow (N): 100000000
19  *   Podaj liczbe WŁtkow (N): 4
20  * @endcode
21  */
22
23
24 #include <iostream>
25 #include <thread>
26 #include <vector>
27 #include <cmath>
28 #include <chrono>
29
30 /**
31  * @brief Funkcja, kt r ca Łkujemy: \f$f(x) = \sqrt{1 - x^2}\f$
32  *
33  *
34  * @param[in] x Argument funkcji.
35  * @return Warto Ź \f$\sqrt{1 - x^2}\f$.
36  */
37
38 inline double f(double x) {
39     return std::sqrt(1.0 - x * x);
40 }
41
42 /**
43  * @struct ThreadData

```



```

36  * @brief Struktura przechowująca dane dla wtku.
37  *
38  * Struktura zawiera informacje o przedziale pracy danego wtku:
39  * - start i end, czyli indeks początkowy i końcowy
    przedziału,
40  * - sum, w którym gromadzona jest częściowa suma,
41  * - step, czyli szerokość jednego prostokąta do metody
    prostokątów.
42  */
43  struct ThreadData {
44      unsigned long long start; ///< Indeks początkowy przedziału
    do całkowania.
45      unsigned long long end;    ///< Indeks końcowy przedziału do
    całkowania.
46      double sum;                ///< Częściowa suma wyliczona przez
    wtek.
47      double step;              ///< Szerokość jednego prostokąta.
48  };
49
50  /**
51  * @brief Funkcja uruchamiana w wtku. Oblicza częściową sumę
    dla fragmentu całki.
52  *
53  * Metoda prostokątów bazuje na pobraniu wartości funkcji w
    źródłu przedziału:
54  * 
$$x_i = (i + 0.5) \cdot \text{step}$$

55  *
56  * @param[in,out] data Struktura ThreadData z danymi
    wejściowymi (zakres przedziału)
57  *                  i wyjściowymi (suma częściowa).
58  */
59  void threadFunction(ThreadData& data) {
60      double partialSum = 0.0;
61      for (unsigned long long i = data.start; i < data.end; ++i) {
62          double x = (static_cast<double>(i) + 0.5) * data.step;
63          partialSum += f(x);
64      }
65      data.sum = partialSum;
66  }
67
68  /**
69  * @brief Funkcja główna programu.
70  *
71  * Wczytuje liczbę przedziału N i liczbę wtków T.
72  * Następnie dzieli pracę między wtki, tworzy je, uruchamia

```

```
73 * i na koniec sumuje wyniki.
74 * Zmierzone zostaje również całkowite zużycie czasu obliczeń.
75 *
76 * @return Kod wyjścia programu (0 jeżeli zakończy się sukcesem)
77 *
78 int main() {
79     // Wczytanie parametrów: liczba podprzedziałów (N) i liczba
80     // wątków (T)
81     unsigned long long N;
82     int T;
83     std::cout << "Podaj liczbę podprzedziałów (N): ";
84     std::cin >> N;
85     std::cout << "Podaj liczbę wątków (N): ";
86     std::cin >> T;
87
88     // Rozpoczęcie pomiaru czasu
89     auto startTime = std::chrono::high_resolution_clock::now();
90
91     double step = 1.0 / static_cast<double>(N); // szerokość
92     // jednego prostokąta
93
94     // Utworzenie wątków i wektorów do przechowywania danych
95     std::vector<std::thread> threads(T);
96     std::vector<ThreadData> threadData(T);
97
98     // Podział przedziałów na wątki
99     unsigned long long chunk = N / T; // ilość pracy dla
100     // każdego wątku
101     unsigned long long remainder = N % T; // reszta do
102     // rozdysponowania
103
104     unsigned long long currentStart = 0;
105
106     for (int i = 0; i < T; ++i) {
107         /**
108          * @note Rozdzielamy resztę w taki sposób, aby wątki w
109          * pierwszej kolejności
110          * dostawały po jednym dodatkowym podprzedziale,
111          * jeżeli coś zostało.
112          */
113         unsigned long long currentEnd = currentStart + chunk;
114         if (i < remainder) {
115             currentEnd += 1;
116         }
117     }
```

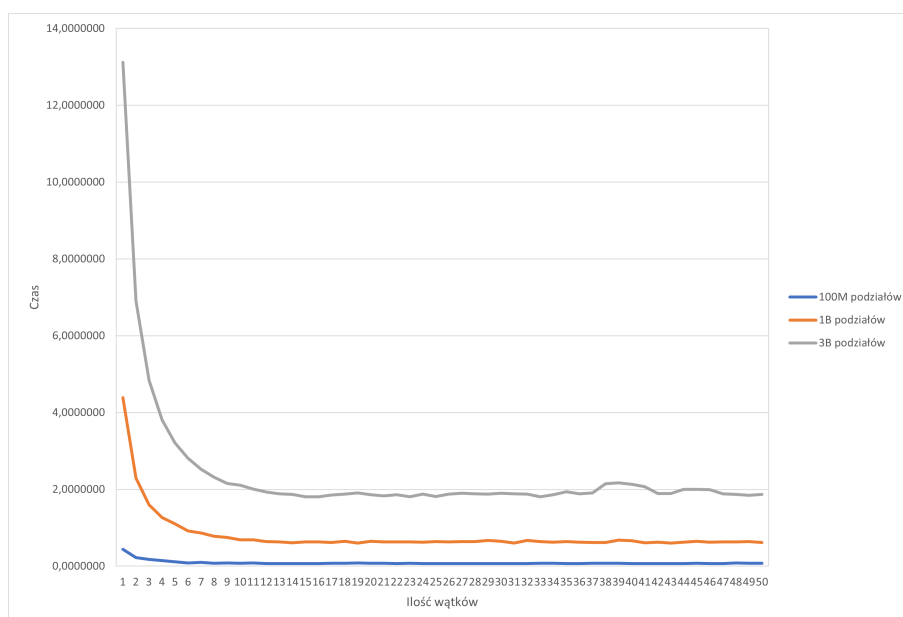
```

111
112     threadData[i].start = currentStart;
113     threadData[i].end = currentEnd;
114     threadData[i].step = step;
115     threadData[i].sum = 0.0;
116
117     // Uruchomienie w tku
118     threads[i] = std::thread(threadFunction, std::ref(
119 threadData[i]));
120
121     currentStart = currentEnd;
122 }
123
124 // Czekamy na zakończenie wszystkich w tk w i sumujemy
125 wyniki
126 double totalSum = 0.0;
127 for (int i = 0; i < T; ++i) {
128     threads[i].join();
129     totalSum += threadData[i].sum;
130 }
131
132 // Całka z sqrt(1 - x^2) w przedziale [0,1] to pi/4, st d
133 przybliżenie pi
134 double integralValue = step * totalSum;
135 double pi_approx = 4.0 * integralValue;
136
137 // Koniec pomiaru czasu
138 auto endTime = std::chrono::high_resolution_clock::now();
139 std::chrono::duration<double> elapsed = endTime - startTime;
140
141 // Wyświetlenie wyniku i czasu wykonania
142 std::cout << "Oszacowana wartosc pi: " << pi_approx << std::
143 endl;
144 std::cout << "Czas obliczen: " << elapsed.count() << " s" <<
145 std::endl;
146
147 return 0;
148 }

```

Listing 1. Program main

Program został opisany w formie komentarzy ??

**Rys. 4.1.** Porównanie wyników

4.2. Porównanie wyników

Rysunek 4.1 przedstawia porównanie czasów wykonania w porównaniu do ilości wykorzystanych wątków przy odpowiednich ilościach podziałów (Użyty procesor to ryzen 7 9700x posiadający 8 rdzeni i 16 wątków).

5. Wnioski

Projekt polegający na implementacji wielowątkowego programu sortującego liczby metodą scalania oraz wyliczania przybliżonej wartości liczby metodą całkowania numerycznego okazał się sukcesem pod wieloma względami.

Dzięki Google Test proces tworzenia testów jednostkowych stał się niezwykle prosty. Chociaż algorytm sortowania przez scalanie jest stosunkowo skomplikowany, Google Test umożliwił mi łatwe i szybkie stworzenie testów sprawdzających różne przypadki, takie jak sortowanie już posortowanych danych, danych w odwrotnej kolejności, a także przypadków brzegowych z małą liczbą elementów.

Implementacja wielowątkowości za pomocą biblioteki `std::thread` pozwoliła na znaczące skrócenie czasu wykonania operacji sortowania oraz obliczeń całki w porównaniu do wersji jednowątkowej. Testy przeprowadzone z różną liczbą wątków (od 1 do 50) wykazały, że wzrost liczby wątków generalnie przekłada się na redukcję czasu obliczeń. Jednakże, po przekroczeniu liczby rdzeni fizycznych procesora, zaobserwowano spowolnienie wynikające z narzutów związanych z zarządzaniem wątkami oraz przełączaniem kontekstu. Oznacza to, że optymalna liczba wątków jest zazwyczaj równa liczbie dostępnych rdzeni CPU, co maksymalizuje efektywność wykorzystania zasobów sprzętowych.

Chat GPT poradził sobie bez problemu z napisaniem kodu, jednakże AI może generować kod, który wygląda poprawnie, ale zawiera błędy logiczne lub składniowe. Nie zawsze jest w stanie zrozumieć specyficzne wymagania projektu, co może prowadzić do wprowadzenia błędów.

Nadmierne poleganie na AI może prowadzić do braku zrozumienia podstawowych koncepcji programistycznych. Programista może stać się mniej samodzielny w rozwiązywaniu problemów bez wsparcia AI. AI może nieświadomie generować kod podatny na ataki lub zawierający luki bezpieczeństwa, zwłaszcza jeśli jest używany do tworzenia aplikacji o krytycznym znaczeniu.

Istnieją obawy dotyczące praw autorskich w przypadku kodu generowanego przez AI, zwłaszcza jeśli opiera się on na fragmentach istniejących projektów bez odpowiedniego licencjonowania. AI działa na podstawie danych treningowych i może nie mieć pełnego kontekstu projektu, co może prowadzić do nieoptymalnych rozwiązań.

Tak, z pewnymi zastrzeżeniami. AI może znacząco zwiększyć efektywność procesu programowania, zwłaszcza w przypadku rutynowych zadań, szybkiego prototypowania czy generowania szkieletów kodu. Jednak kluczowe jest, aby programista

posiadał podstawową wiedzę i umiejętności, aby móc ocenić i poprawić wygenerowany kod. AI powinno być traktowane jako narzędzie wspierające, a nie zastępujące programistę.

Podsumowując chat GPT w dużym stopniu pomógł przy stworzeniu tego projektu oraz był bardzo pomocny, jednakże trzeba być świadomym że stworzone odpowiedzi mogą zawierać błędy.

Bibliografia

- [1] *Chat GPT*. URL: <https://openai.com/index/chatgpt/>.
- [2] *Wikipedia c++*. URL: <https://pl.wikipedia.org/wiki/C%2B%2B>.
- [3] *Strona internetowa Visual studio*. URL: <https://visualstudio.microsoft.com/pl/>.
- [4] *Strona internetowa Doxygen'a*. URL: <https://doxygen.nl/>.

Spis rysunków

3.1. ikona c++	11
3.2. Ikona VS	12
3.3. Ikona doxygen	14
4.1. Porównanie wyników	20

Spis listingów

1.	Program main	16
----	------------------------	----