

## My Project

Generated by Doxygen 1.15.0



---

<b>1 Class Index</b>	<b>1</b>
1.1 Class List . . . . .	1
<b>2 File Index</b>	<b>3</b>
2.1 File List . . . . .	3
<b>3 Class Documentation</b>	<b>5</b>
3.1 PiCalculator Class Reference . . . . .	5
3.1.1 Detailed Description . . . . .	5
3.1.2 Member Function Documentation . . . . .	5
3.1.2.1 calculate() . . . . .	5
<b>4 File Documentation</b>	<b>7</b>
4.1 PiCalculator.cpp File Reference . . . . .	7
4.1.1 Detailed Description . . . . .	7
4.2 PiCalculator.h File Reference . . . . .	7
4.2.1 Detailed Description . . . . .	7
4.3 PiCalculator.h . . . . .	8
4.4 Projekt05.cpp File Reference . . . . .	8
4.4.1 Detailed Description . . . . .	8
4.4.2 Function Documentation . . . . .	8
4.4.2.1 main() . . . . .	8
<b>Index</b>	<b>9</b>



# Chapter 1

## Class Index

### 1.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

<a href="#">PiCalculator</a>	Klasa odpowiedzialna za obliczenia matematyczne liczby PI . . . . .	<a href="#">5</a>
------------------------------	---	-------------------



## Chapter 2

# File Index

### 2.1 File List

Here is a list of all documented files with brief descriptions:

<a href="#">PiCalculator.cpp</a>	
Implementacja metod klasy <a href="#">PiCalculator</a> . . . . .	7
<a href="#">PiCalculator.h</a>	
Definicja klasy <a href="#">PiCalculator</a> . . . . .	7
<a href="#">Projekt05.cpp</a>	
Główny modul programu wykonawczego . . . . .	8





## Chapter 3

# Class Documentation

### 3.1 PiCalculator Class Reference

Klasa odpowiedzialna za obliczenia matematyczne liczby PI.

```
#include <PiCalculator.h>
```

#### Public Member Functions

- double [calculate](#) (long long num\_steps, int num\_threads)  
*Metoda obliczajaca PI.*

#### 3.1.1 Detailed Description

Klasa odpowiedzialna za obliczenia matematyczne liczby PI.

- Klasa udostepnia metody do rownoleglego obliczania wartosci PI z wykorzystaniem biblioteki standardowej watkow C++11.

#### 3.1.2 Member Function Documentation

##### 3.1.2.1 calculate()

```
double PiCalculator::calculate (  
    long long num_steps,  
    int num_threads)
```

Metoda obliczajaca PI.

#### Parameters

---

<i>num_steps</i>	Liczba iteracji algorytmu.
<i>num_threads</i>	Liczba wątków roboczych.

### Returns

double Wynik obliczen.

Wektor przechowujący wyniki czastkowe z kazdego watku Uzywamy wektora, aby uniknac wyscigu danych (data race) przy sumowaniu

Obliczamy ile krokow petli przypada na jeden watek

Uruchamiamy watek z wyrazeniem lambda Przekazujemy zmienne przez wartosc [=] oraz wektor wynikow przez referencje [&partial\_sums]

Ostatni watek bierze na siebie ewentualna reszte z dzielenia

Petla calkowania numerycznego (metoda prostokatow z punktem srodkowym)

Zapisz wynik czastkowy w odpowiednim miejscu wektora

Czekamy na zakonczenie wszystkich watkow (join)

Sumujemy wyniki czastkowe

The documentation for this class was generated from the following files:

- [PiCalculator.h](#)
- [PiCalculator.cpp](#)

# Chapter 4

## File Documentation

### 4.1 PiCalculator.cpp File Reference

Implementacja metod klasy [PiCalculator](#).

```
#include "PiCalculator.h"  
#include <thread>  
#include <vector>
```

#### 4.1.1 Detailed Description

Implementacja metod klasy [PiCalculator](#).

### 4.2 PiCalculator.h File Reference

Definicja klasy [PiCalculator](#).

#### Classes

- class [PiCalculator](#)  
*Klasa odpowiedzialna za obliczenia matematyczne liczby PI.*

#### 4.2.1 Detailed Description

Definicja klasy [PiCalculator](#).

## 4.3 PiCalculator.h

[Go to the documentation of this file.](#)

```
00001
00005
00006 #ifndef PICALCULATOR_H
00007 #define PICALCULATOR_H
00008
00015 class PiCalculator {
00016 public:
00023     double calculate(long long num_steps, int num_threads);
00024 };
00025
00026 #endif
```

## 4.4 Projekt05.cpp File Reference

Główny moduł programu wykonawczego.

```
#include <iostream>
#include <chrono>
#include <iomanip>
#include <string>
#include "PiCalculator.h"
```

### Functions

- int [main](#) (int argc, char \*argv[])

### 4.4.1 Detailed Description

Główny moduł programu wykonawczego.

- Odpowiada za:
- Parsowanie argumentów linii komend.
- Inicjalizację procesu obliczeniowego.
- Precyzyjny pomiar czasu rzeczywistego wykonania zadania.

### 4.4.2 Function Documentation

#### 4.4.2.1 main()

```
int main (
    int argc,
    char * argv[])
```

Sprawdzamy, czy podano odpowiednią liczbę argumentów. Oczekujemy: ./Projekt05 <liczba\_kroków> <liczba\_wątków>

Pomiar czasu START

Obliczenia

Pomiar czasu STOP

Wypisujemy TYLKO czas na końcu, aby Python łatwo to odczytał. Python będzie czytał ostatnią linię wyjścia.

# Index

- calculate
  - PiCalculator, [5](#)
- main
  - Projekt05.cpp, [8](#)
- PiCalculator, [5](#)
  - calculate, [5](#)
- PiCalculator.cpp, [7](#)
- PiCalculator.h, [7](#)
- Projekt05.cpp, [8](#)
  - main, [8](#)