**Politechnika Wrocławska**

**Wydział Elektroniki**

PROJEKT Z ZARZĄDZANIA W SYSTEMACH I SIECIACH KOMPUTEROWYCH

Zastosowanie algorytmów wielowątkowych do rozwiązywania Symetrycznego Problemu Komiwojażera

|  |  |
| --- | --- |
| Autor:  Katarzyna Brzezińska, 235879  Eliza Kalata, 235543 | Prowadzący zajęcia:  Dr inż. Robert Wójcik, W4/K-9 |
|  | Ocena pracy: |

Spis treści

[Spis rysunków 3](#_Toc26206778)

[Spis tabel 4](#_Toc26206779)

[Spis listingów 5](#_Toc26206780)

[1. Wstęp 6](#_Toc26206781)

[1.1. Cel projektu 6](#_Toc26206782)

[1.2. Zakres projektu 6](#_Toc26206783)

[2. Sformułowanie problemu 7](#_Toc26206784)

[2.1. Podstawowe założenia 7](#_Toc26206785)

[2.2. Zastosowany algorytm 7](#_Toc26206786)

[2.3. Analiza złożoności obliczeniowej problemu 7](#_Toc26206787)

[2.4. Metoda i algorytmy rozwiązywania problemu 7](#_Toc26206788)

[2.5. Przykład praktyczny algorytmu wykorzysującego programowanie dynamiczne: 7](#_Toc26206789)

[2.6. Technologie i narzędzia implementacji 8](#_Toc26206790)

[2.7. Harmonogram prac 8](#_Toc26206791)

[3. Projekt 10](#_Toc26206792)

[3.1. Zastosowane technologie i narzędzia projektowe 10](#_Toc26206793)

[3.2. Struktura programu 10](#_Toc26206794)

[3.3. Koncepcja działania algorytmu 10](#_Toc26206795)

[3.4. Diagram klas 10](#_Toc26206796)

[3.5. Struktura danych wejściowych 10](#_Toc26206797)

[3.6. Struktura wyników 10](#_Toc26206798)

[4. Implementacja systemu 11](#_Toc26206799)

[4.1. Wybrane klasy 11](#_Toc26206800)

[4.2. Realizacja algorytmu 11](#_Toc26206801)

[4.3. Mechanizm odczytu danych 11](#_Toc26206802)

[4.4. Zapis i prezentacja wyników 11](#_Toc26206803)

[5. Testowanie i ocena wydajności 12](#_Toc26206804)

[5.1. Poprawność 12](#_Toc26206805)

[5.2. Analiza czasów 12](#_Toc26206806)

[5.2.1. Porównanie czasów 12](#_Toc26206807)

[5.2.2. Weryfikacja z najlepszym uzyskanym wynikiem 12](#_Toc26206808)

[5.3. Wnioski z testów 12](#_Toc26206809)

[6. Podsumowanie 13](#_Toc26206810)

[Literatura 14](#_Toc26206811)

# Spis rysunków

[Rysunek 1Macierz kosztów z pliku tekstowego 10](#_Toc26300821)

[Rysunek 2.Wykres zależności czasu od liczby wątków dla 17 miast 12](#_Toc26300822)

[Rysunek 3.Wykres zależności czasu od liczby wątków dla 20 miast 13](#_Toc26300823)

# Spis tabel

[Tabela 1.Zależność czasu od liczby wątków 12](#_Toc26300847)

[Tabela 2.Zależność czasu od liczby wątków dla 20 miast 13](#_Toc26300848)

# Spis listingów

# Wstęp

## Cel projektu

Celem projektu było stworzenie oraz implementacja systemu realizującego obliczenia i znajdującego rozwiązanie problemu komiwojażera [1] przy użyciu wątków. Kryterium oceny jakości trasy jest suma wag krawędzi, tworzących w grafie pełen cykl Hamiltona [2].

## Zakres projektu

Swoim zakresem projekt obejmuje zarządzanie użycie wątków w celu przyspieszenia obliczeń oraz otrzymania poprawnego wyniku, który zostaną porównany i zweryfikowany zgodnie z przyjętymi założeniami.

# Sformułowanie problemu

## Podstawowe założenia

Problem optymalizacyjny: Jak powinien zaplanować trasę podróży komiwojażer, aby w sumie przebył możliwie najkrótszą drogę?

Problem komiwojażera jest problemem NP – zupełnym. Mając pełny graf skierowany G=(V,E;w) ,gdzie |V| =n , a w(e) wyznacza wagę krawędzi e. Problem polega na znalezieniu cyklu Hamiltona o najmniejszej wadze w G.

Symetryczny Problem Komiwojażera będzie rozwiązywany przy uwzględnieniu następujących założeń [3]:

• zbudowanie ważnego, skierowanego grafu, którego wierzchołki są miastami;

• połączenie pary wierzchołków krawędziami – odcinkiem trasy, który będzie odzwierciedlał odległość pomiędzy dwoma miastami;

• przyjęcie za punkt początkowy trasy wierzchołek (miasto) o numerze 1;

• krawędzie opatrzone są dodatnimi wagami;

• odwiedzenie wszystkich miast odpowiada cyklowi, który przechodzi przez każdy wierzchołek grafu dokładnie raz, tworząc pełny cykl Hamiltona o minimalnej sumie wag krawędzi;

## Zastosowany algorytm

Sformułowanie problemu zarządzania zasobami w systemach i sieciach komputerowych.

## Analiza złożoności obliczeniowej problemu

Zastosowaną metodą rozwiązania problemu jest wykorzystanie Programowania Dynamicznego [1] do implementacji algorytmu. Programowanie Dynamiczne jest metodą rozwiązywania zadań optymalizacyjnych, które na mocy pewnych własności, można sformułować jako poszukiwanie ciągu decyzji. Zostanie zastosowany algorytm Bellmana - Helda-Karpa ma złożoność czasową O(n^2 \*2^n) [5].

## Metoda i algorytmy rozwiązywania problemu

Problem Komiwojażera będzie rozwiązywany przy pomocy algorytmu Programowania Dynamicznego. Dzięki zastosowaniu wątków

## Przykład praktyczny algorytmu wykorzysującego programowanie dynamiczne:

matrix(0,{a,b})-zaczynamy trasę od 0 i może przejść przez a do b lub przez b do a.

Dodaje wagę drogi powrotnej do wierzchołka początkowego w naszym przypadku jest nim 0

matrix ( 2, {3} ) =  (2,3) + matrix (3, 0 )     4+10=14

matrix ( 3, {2} ) =  (3,2) + matrix (2, 0 )     1+14=15

matrix ( 1, {3} ) =  (1,3) + matrix (3, 0 )     6+10=16

matrix ( 3, {1} ) =  (3,1) + matrix (1, 0 )     28+26 = 54

matrix ( 1, {2} ) =  (1,2) + matrix (2, 0 )     25+14 = 39

matrix ( 2, {1} ) =  (2,1) + matrix (1, 0 )     37+26=63

matrix (1, {2,3} ) = minimum of

= { (1,2) + matrix (2, {3} ) 25+14=39

= { (1,3) + matrix {3, {2} ) 6+15=21

matrix (2, {1,3} ) = minimum of

= { (2,1) + matrix (1, {3} ) 37+16=53

= { (2,3) + v {3, {1} ) 4+54=58

v (3, {1,2} ) = minimum of

= { (3,1) + matrix (1, {2} ) 28+39=67

= { (3,2) + matrix {3, {1} ) 1+54=55

matrix( 0, {1,2,3} ) = minimum of

= { (0,1) + matrix (1, {2,3} ) 33+21=54

= { 0,2) + matrix (2, {1,3} ) 6+53=59

= { (0,3) + matrix (3, {1,2} ) 15+55=70

Tras:0->1->3->2->0, waga: 54

## Technologie i narzędzia implementacji

Program realizujący algorytm zostanie napisany w języku C++, przy wykorzystaniu bibliotek STL [6]. Projekt będzie realizowany w środowisku Microsoft Visual Studio 2013 [7].

## Harmonogram prac

|  |  |
| --- | --- |
| Tydzień: | Zakres prac |
| 1 | Opis założeń projektu |
| 2 | Implementacja algorytmów |
| 3 | Zrobienie testów i przedstawienie wyników |
| 4 | Napisanie sprawozdania |

# Projekt

## Zastosowane technologie i narzędzia projektowe

Projekt został wykonany w środowisku programistycznym Visual Studio 13

## Struktura programu

## Koncepcja działania algorytmu

Algorytm wykorzystuje maske bitów oraz rekurencje do znalezienia optymalnego rozwiązania, co w naszym przypadku jest drogą o jak najmniejszym koszcie przebycia.

## Diagram klas

Obraz zawierający zrzut ekranu

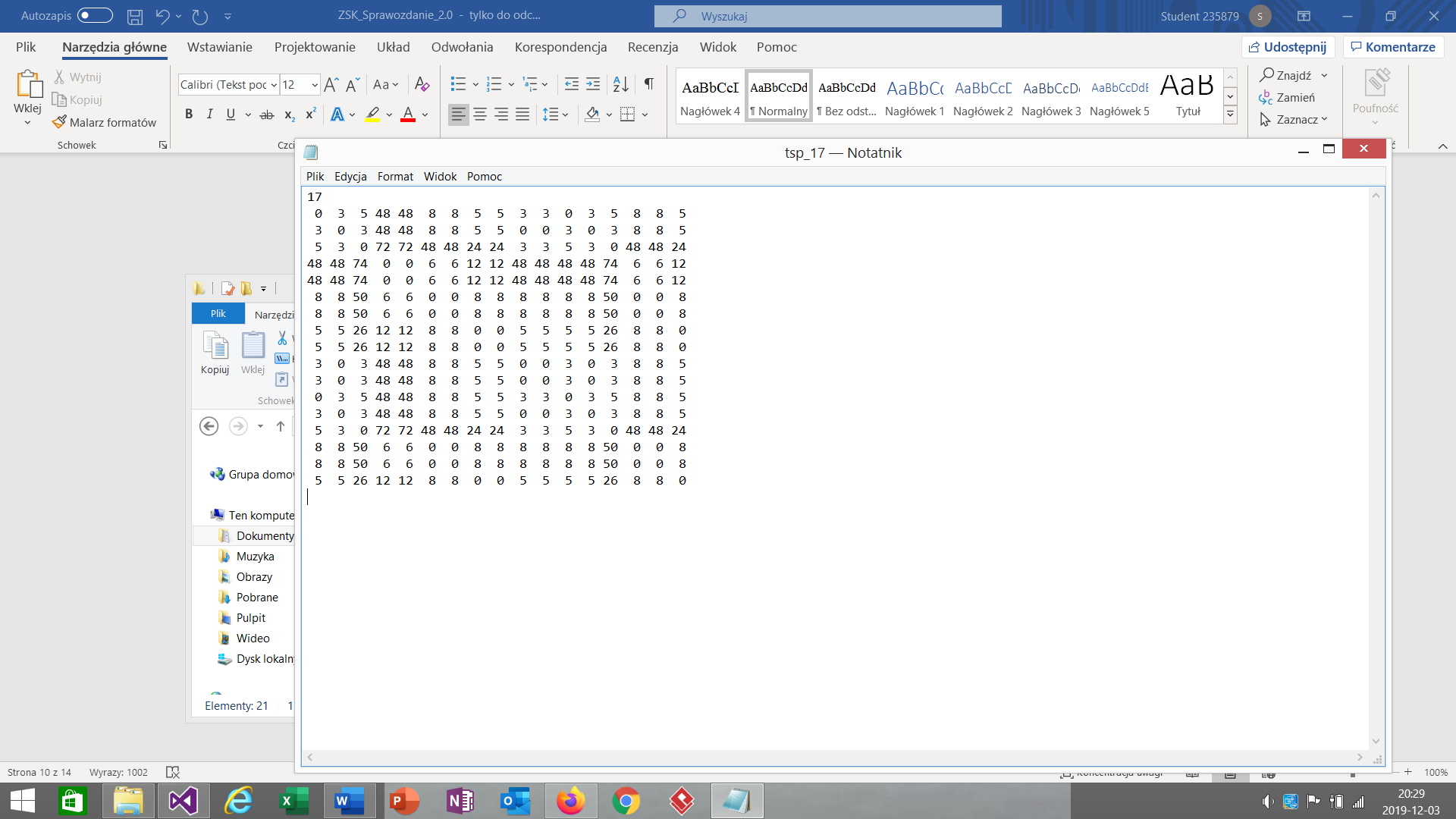
Opis wygenerowany automatycznie

Rysunek Diagram Klas

## Struktura danych wejściowych

Dane wejściowe są pobierane z plików tekstowych, gdzie pierwszy wiersz zawiera tylko liczbę miast, w kolejnych wierszach przedstawiona jest macierz kosztów, dane na przekątnej mają wartość 0.

Dane wczytywane są do zmiennej *std::vector<vector<int>> matrix* w funkcji *pobierz*.



Rysunek Macierz kosztów z pliku tekstowego

## Struktura wyników

Dane wyjściowe, czyli miasta w odpowiedniej kolejności są w wyniku pracy funkcji *dynamic* umieszczane w zmiennej *std::vector<vector<int>> k* , a później w wyniku działania funkcj *d* wyświetlane w oknie konsoli.

# Implementacja systemu

Klasa Pr implementuje wykorzystanie wątków, które jako parametr przyjmują klasę K wraz z naszą funkcją docelową - dynami. Liczba wątków została ustawiona w sposób stały, przy użyciu predyrektywy define, w celu zmiany liczby wątków należy zmienić ja z poziomu kodu.

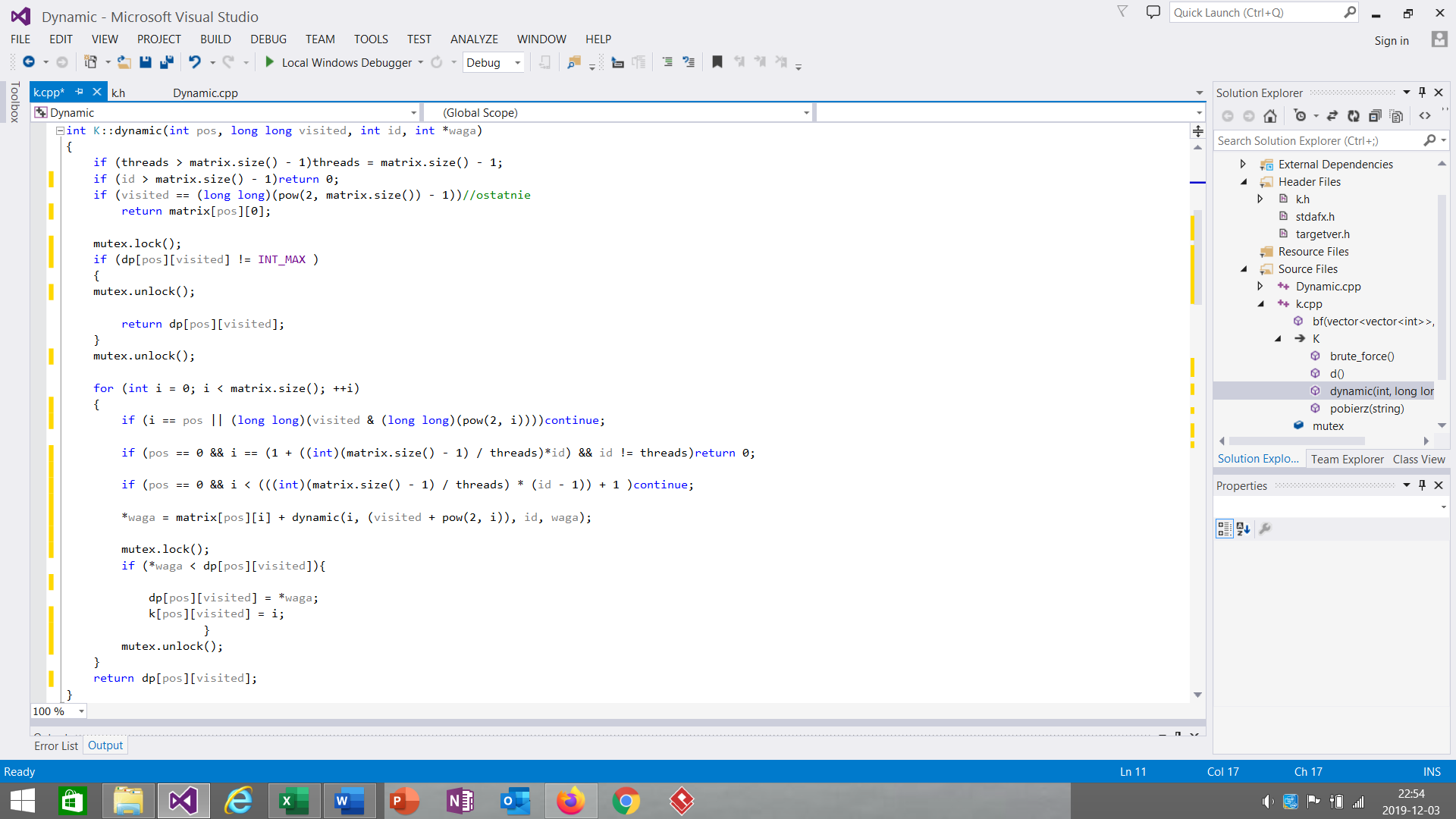
Funkcja std::chrono z biblioteka STL odpowiada za wykonywanie pomiarów czasowych dla zadanych parametrów

## Wybrane klasy

## Realizacja algorytmu

Implementacja algorytmu programowania dynamicznego znajduje się w funkcji dynamic wywoływane jest dla stałego wybranego wierzchołek o numerze 1.

Warto zaznaczyć problem współdzielenia zasobów występujący w tej funkcji, z tego powodu wykorzystujemy mechanizm mutex, pozwalający na dostęp do zasobu tylko jednemu z wątków, gdy inne stoją w kolejce

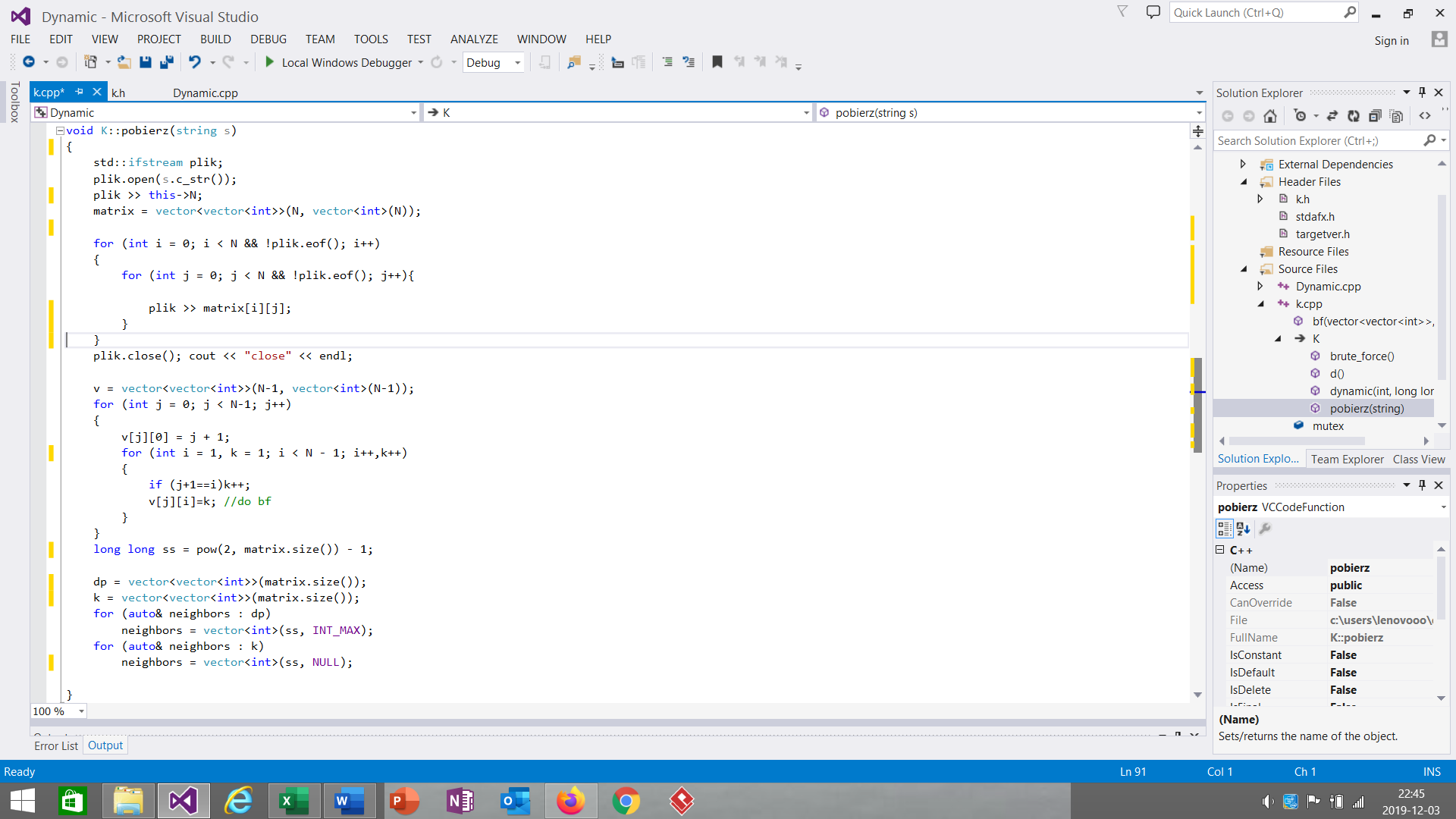


Rysunek Funkcj dynamic implementująca algorytm wykorzystujące programowanie dynamiczne

## Mechanizm odczytu danych

Pliki z informacjami o liczbie miast oraz wagach pomiędzy nimi są wczytywane z pliku tekstowego przy użyciu funkcji *pobierz* będącej metodą klasy *Dynamic*.

Dane wczytywane są do zmiennej matrix, która jest wektorem wektorów przechowujących zmienne typu *int (std:: vector<vector<int>>).*



Rysunek Metoda pobierz klasy K

## Zapis i prezentacja wyników

# Testowanie i ocena wydajności

## Poprawność

## Analiza czasów

### Porównanie czasów

Tabela .Zależność czasu od liczby wątków



Rysunek .Wykres zależności czasu od liczby wątków dla 17 miast

Tabela .Zależność czasu od liczby wątków dla 20 miast



Rysunek .Wykres zależności czasu od liczby wątków dla 20 miast

### Weryfikacja z najlepszym uzyskanym wynikiem

## Wnioski z testów

Skonfigurowany system został zweryfikowany pod kątem poprawności i efektywności działania. Porównywanie czasu działania algorytmów zostało zrealizowane przy użyciu biblioteki „chrono” STL [9].

Pomiary: (tabele)

Diagramy

Porównanie

# Podsumowanie

# Literatura

[1] Cormen T., Leiseron C., Rivest R., Wprowadzenie do algorytmów, WNT, Warszawa, 2001.

[2] https://eduinf.waw.pl/inf/alg/001\_search/0136.php

[3] http://www.mini.pw.edu.pl/MiNIwyklady/grafy/prob-komiw.html

[4] http://wazniak.mimuw.edu.pl/index.php?title=Z%C5%82o%C5%BCono%C5%9B%C4%87\_obliczeniowa/Wyk%C5%82ad\_5:\_Problemy\_NP-zupe%C5%82ne

[5] http://algorytmy.ency.pl/artykul/algorytm\_helda\_karpa

[6] http://cpp0x.pl/kursy/Kurs-STL-C++/Wstep-podstawowe-informacje-o-STL-u/115

[7] https://docs.microsoft.com/pl-pl/visualstudio/productinfo/vs2013-sysrequirements-vs

[8] http://cpp0x.pl/kursy/Kurs-New-Curses-C++/19

[9] https://en.cppreference.com/w/cpp/chrono