**Sprawozdanie**

z laboratoriów Podstaw Informatyki z dnia 6.11.2020

Daniel Ślusarczyk, grupa 3, numer albumu:311511

**Spis treści:**

1. Cel projektu 2
2. Teoria 2
3. Szczegóły implementacyjne 2
4. Sposób wywołania programu 3
5. Wnioski i spostrzeżenia 7
6. Cel Projektu:

Zadaniem laboratoryjnym było napisanie programu w języku C odpowiedzialnego za wczytywanie parametrów wybranego grafu z podanego pliku tekstowego i znajdowanie najkrótszej ścieżki od określonego początku i końca.

1. Teoria:

Przygotowany program interpretuje wczytywany graf za pomocą macierzy sąsiedztwa tworzonej na podstawie przygotowanego wcześniej pliku.

Szukanie najkrótszej ścieżki odbywa się za pośrednictwem przygotowanej przeze mnie funkcji wzorowanej na algorytmie Dijkstry. Algorytm ten jest to schemat postępowania w celu znalezienie najkrótszej ścieżki w grafie o nieujemnych wagach krawędzi. Znajduje on odległości od wierzchołka startowego do wszystkich pozostałych wierzchołków przechodząc po wszystkich i przypisując im odpowiednie koszty. Modyfikacja tego schematu w przypadku tego programu polegała na wprowadzaniu dodatkowych tablic sprawdzających zależności, które pozwalają na implementację tego rozwiązania w języku C.

1. Szczegóły implementacyjne:

Przedstawienie grafu w formie możliwej do wczytania przez program zostało rozwiązane za pomocą pliku tekstowego zawierającego:

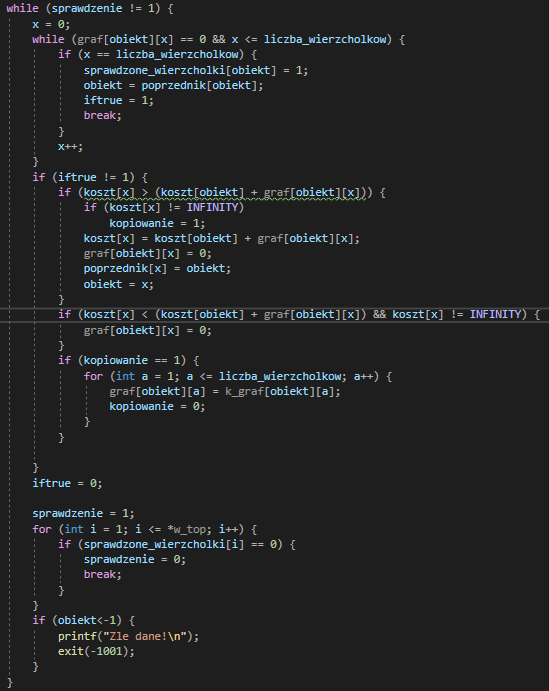
* W pierwszej linii liczbę wierzchołków i liczbę zadeklarowanych krawędzi (pozwala sprawdzić, czy zadeklarowana liczba krawędzi zgadza z liczbą wczytanych przez program):

**W=<liczba wierzchołków>;E=<liczba krawędzi>**

* W kolejnych linia poszczególne krawędzie między wierzchołkami

**e<nr krawędzi>=(V<nr wierzchołka zaczynającego krawędź>,V<nr wierzchołka kończącego>)**

Przygotowany program rozwiązuje problem wczytywania pliku za pomocą funkcji zaimplementowanych w języku C (fopen, fget, fscanf). Kończy wykonywanie programu z określonym błędem jeśli jakikolwiek znak został błędnie wpisany. Następnie tworzy macierz sąsiedztwa (oraz jej kopię) i przypisuje każdemu wierzchołkowi „losową” wartość zmiennoprzecinkową. Każda wprowadzona krawędź zostaje zaznaczona w macierzy sąsiedztwa za pomocą średniej arytmetycznej obu wierzchołków, która stanowi jej wagę. Następnie program przechodzi do funkcji odpowiedzialnej za znajdowanie najkrótszej ścieżki grafu.

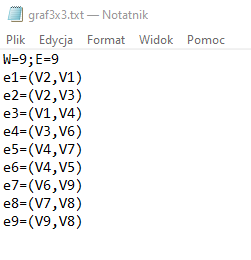
Przedstawiony najważniejszy fragment kodu odpowiada za:

* przeanalizowanie całego grafu
* wpisanie w tablice „koszt[]” sumę wag potrzebnych do dojścia do danego wierzchołka
* wpisanie w tablicę „poprzednik[]” wierzchołka, przez który trzeba przejść, aby dostrzec najkrótszą drogą do tego analizowanego.
* sprawdzanie czy koszt aktualnie analizowanego wierzchołka (obiektu) jest mniejszy od wcześniej znalezionego (początkowo wszystkie koszty mają absurdalnie wielką wartość symbolizującą nieskończoność)
* zakończenie analizowania grafu w momencie, gdy wszystkie obiekty zostaną sprawdzone- tablica „sprawdzone\_wierzchołki[]”

Ostatnim etapem programu jest wypisanie najkrótszej ścieżki od końca do początku.

1. Sposób wywoływania programu:

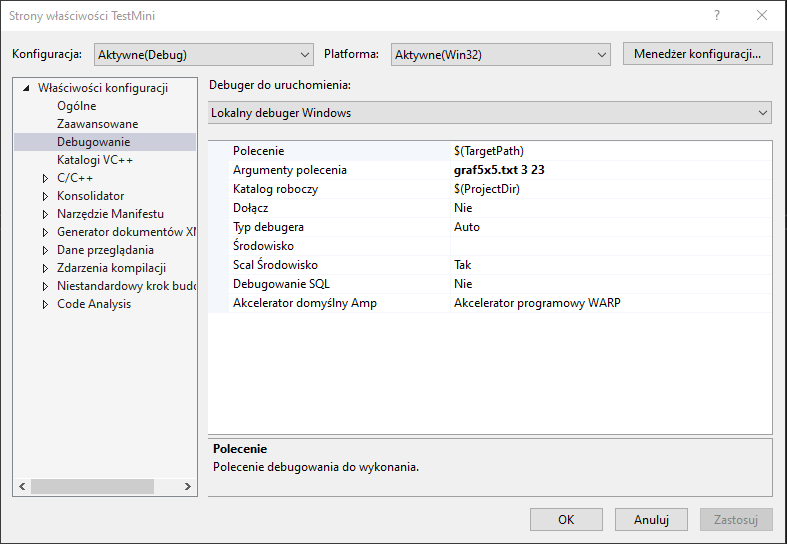
* Przygotowanie pliku tekstowego zawierającego w pierwszej linii parametry grafu, a w kolejnych wszystkie krawędzie. Przykładowe grafy przedstawione w prezentacji są załączone w pliku pod nazwami (graf10x10.txt, graf7x7.txt, graf 5x5,txt). Przykładowy graf3x3.txt:



* Wprowadzenie danych uruchomieniowych programu w polu:

Projekt->Właściwości->Debugowanie->Argumenty polecenia

Zgodnie ze wzorem:

<nazwa pliku do wczytania grafu> <wierzchołek startowy> <wierzchołek końcowy>

**W przypadku braku argumentów uruchomieniowych program analizuje z pliku „graf5x5.txt”.**

* Ograniczenia:

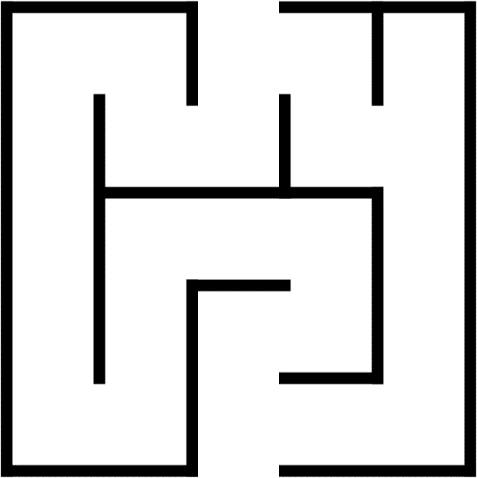
- Program nie działa poprawnie przy podaniu początku i końca w wierzchołkach, do których nie można przejść

- Program obsługuje maksymalnie grafy o 100 wierzchołkach

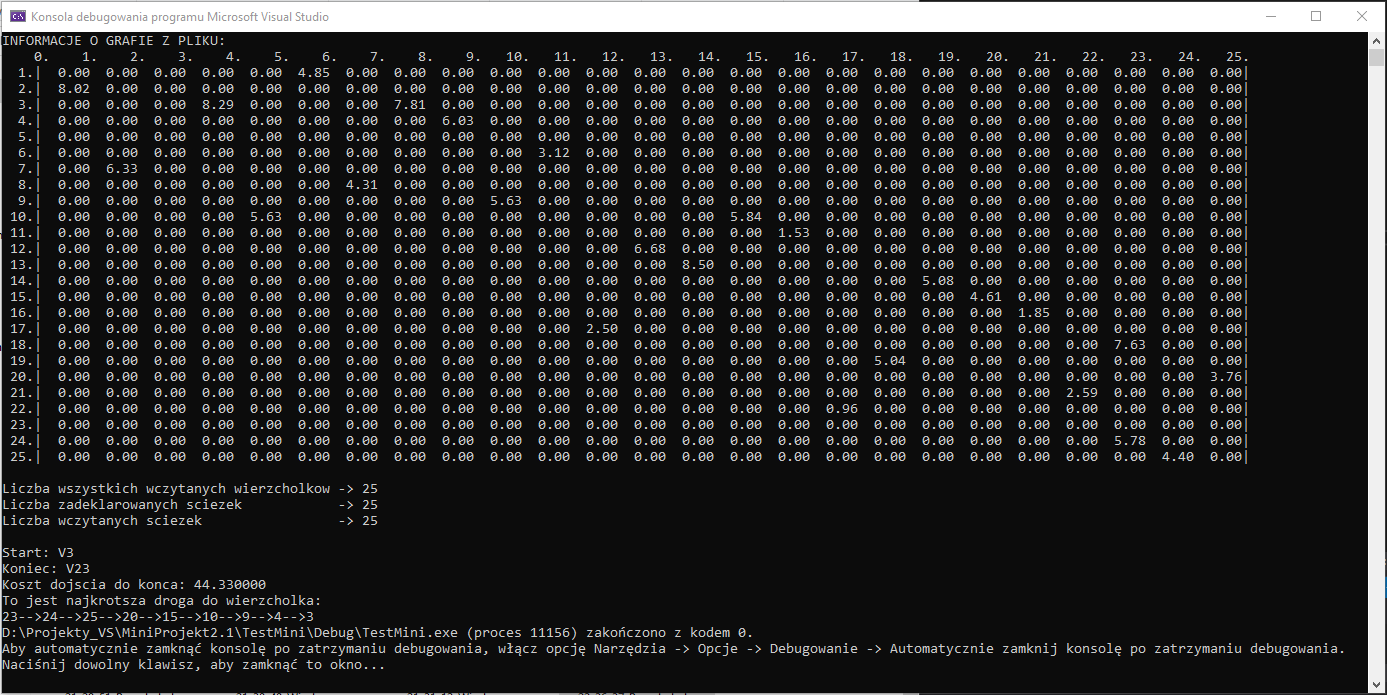
- Każdy podany wierzchołek musi mieć ścieżkę, która do niego prowadzi (inaczej nie można przeanalizować wszystkich wierzchołków)

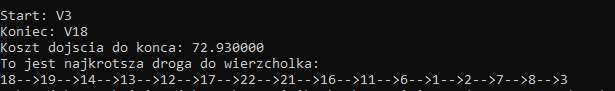
* Przykładowe załączone grafy i możliwe wyniki rozwiązań:

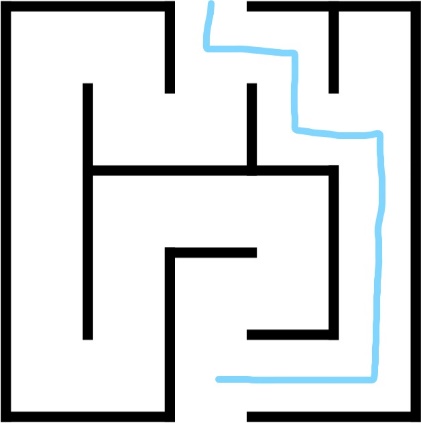
**Argument uruchomieniowy: graf5x5.txt 3 23**



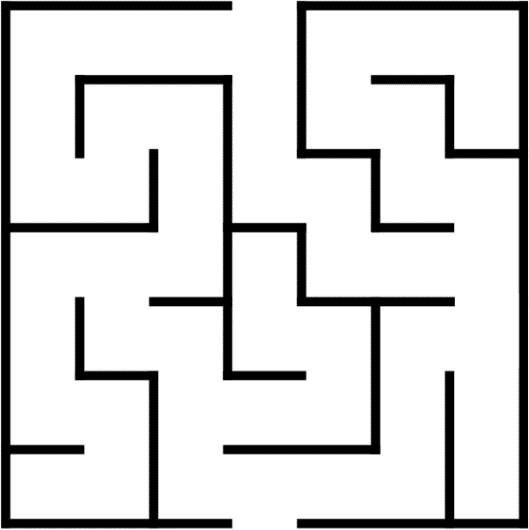
Przykładowy wynik dla wylosowanych danych:



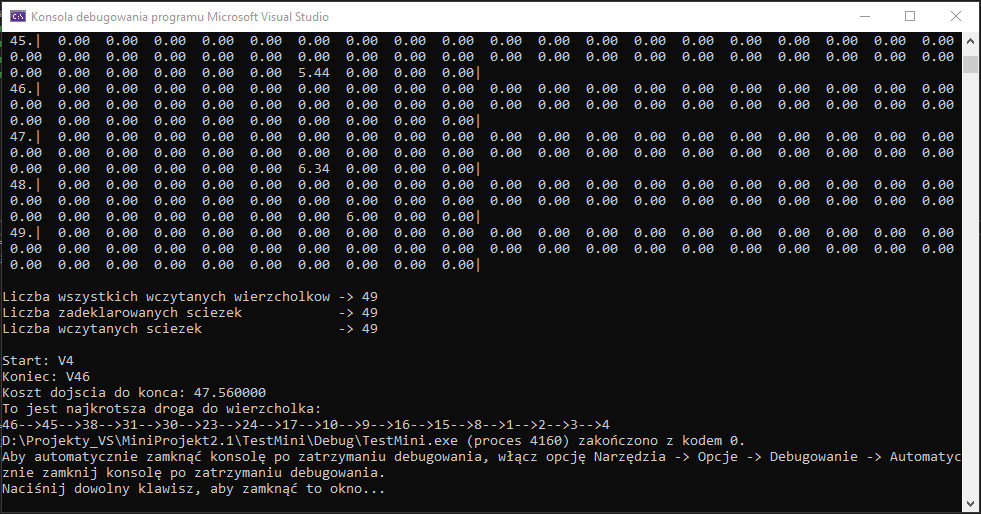
Co reprezentuje ścieżkę:

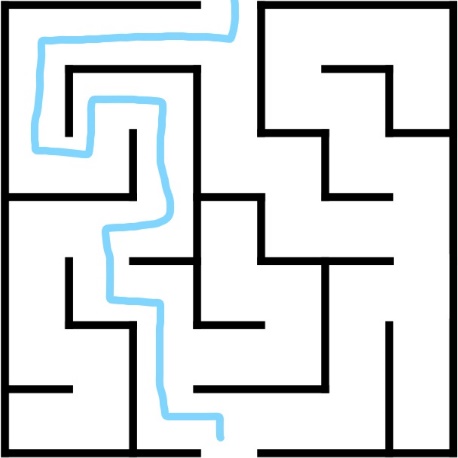


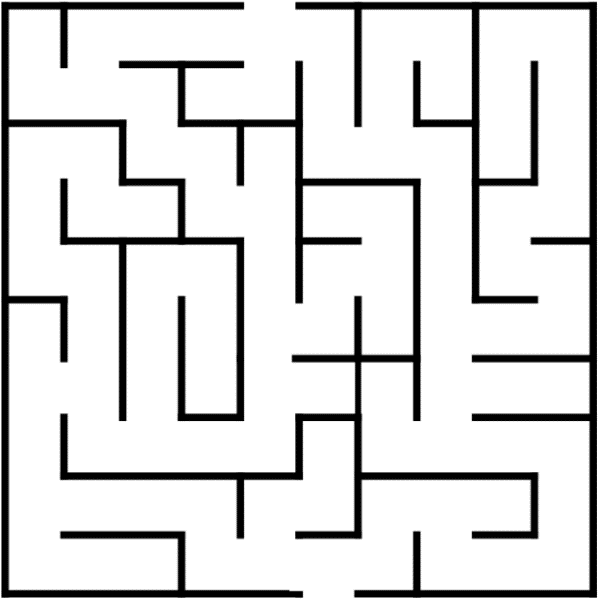
**Argument uruchomieniowy: graf7x7.txt 4 46**



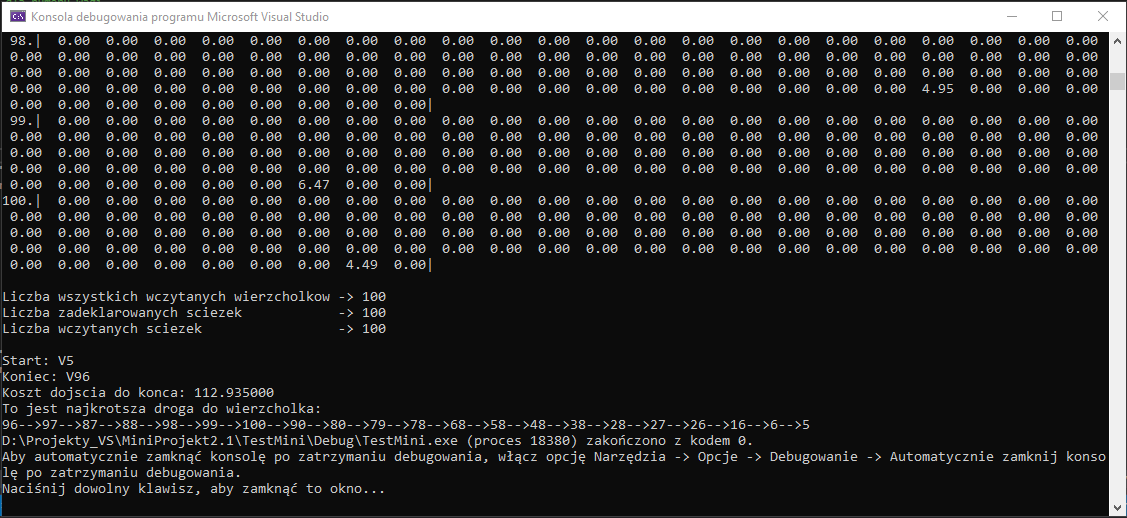
Przykładowy wynik dla wylosowanych danych:

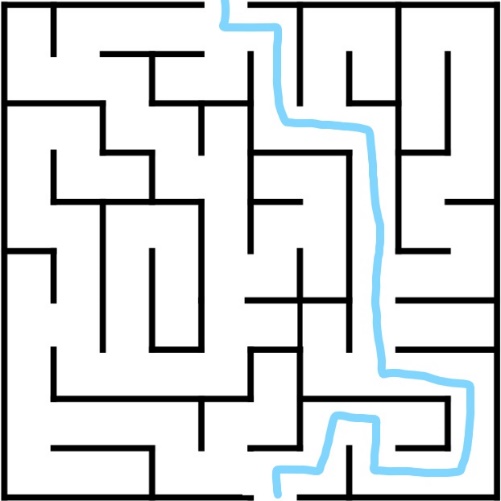


Co reprezentuje ścieżkę:

**Argument uruchomieniowy graf10x10.txt 5 96**

Przykładowy wynik dla wylosowanych danych:



Co reprezentuje ścieżkę:

1. Wnioski i spostrzeżenia

Głównym problemem w przypadku przedstawionego programu było przewidzenie wszystkich możliwości jakie mogę prowadzić do znalezienia najkrótszej ścieżki w grafie. Wiele sytuacji prowadzących do niewłaściwego działania programu wynika z błędnego wprowadzenia informacji, które uniemożliwiają analizę grafu. W takim przypadku często ciężko rozstrzygnąć, czy błędu należy szukać w kodzie, przygotowanym grafie, czy pliku. Program korzysta jednak z oryginalnego kodu, który został w całości przygotowany przeze mnie, a jedynie niektóre elementy działania zostały zaczerpnięta z algorytmu Dijkstry.