Algorytmy i struktury danych

Sprawozdanie – projekt 3.

Zadanie

Graf skierowany reprezentowany przy pomocy listy krawędzi

Adrian Jakubowski

Inżynieria i analiza danych, 1. rok, grupa 3.

1. Wstęp

1.1. Temat

Treść zadania jest następująca:

Napisz program, który dla zadanego grafu skierowanego reprezentowanego przy pomocy listy krawędzi wyznaczy i wypisze następujące informacje:

1) wszystkich sąsiadów dla każdego wierzchołka grafu (sąsiad danego w_i to ten wierzchołek, do którego prowadzi krawędź z w_i)
2) wszystkie wierzchołki, które są sąsiadami danego wierzchołka
3) stopnie wychodzące wszystkich wierzchołków
4) stopnie wchodzące wszystkich wierzchołków
5) wszystkie wierzchołki izolowane
6) wszystkie pętle
7) wszystkie krawędzie dwukierunkowe

Każdy z powyższych podpunktów powinien być realizowany jako oddzielna funkcja. W funkcji main() należy przedstawić działanie napisanej przez siebie biblioteki na reprezentatywnym przykładzie. Kod powinien być opatrzony stosownymi komentarzami.

1.2. Opis problemu

Powyższe zadanie polega na zdefiniowaniu grafu, następnie utworzenia tablicy list, a ostatecznie wyznaczeniu i wypisaniu do pliku tekstowego informacji o tym grafie. Informacje, które program ma przeanalizować znajdują się w podpunktach zadania.

2. Analiza i projektowanie

2.1. Opis podstaw teoretycznych zagadnienia

Rozwiązanie zadania polegało przede wszystkim na wprowadzeniu definicji grafu, zaimplementowaniu tablicy list, a następnie w poszczególnych funkcjach na przeglądaniu jej za pomocą pętli for w celu wyznaczenia poszczególnych informacji. O czym tak naprawdę mowa?

Graf - jest strukturą danych składającą się z dwóch zbiorów: zbioru wierzchołków i zbioru krawędzi, co matematycznie zapisujemy w postaci uporządkowanej pary (tzn. takiej, gdzie istotna jest kolejność elementów tworzących tę parę)

Sąsiad wierzchołka grafu – sąsiad wierzchołka w_i, to wierzchołek, do którego prowadzi krawędź z wierzchołka w_i

Stopień wychodzący wierzchołka - liczba krawędzi wychodzących z wierzchołka.

Stopień wchodzący wierzchołka - liczba krawędzi wchodzących do wierzchołka.

Wierzchołek izolowany - wierzchołek nie połączony krawędzią z żadnym innym wierzchołkiem grafu

Pętla – pętlę otrzymujemy wtedy, gdy wierzchołek jest połączony krawędzią ze samym sobą

Krawędź dwukierunkowa – występuje wtedy, gdy prowadzi jednocześnie z wierzchołka „i” do wierzchołka „j” i na odwrót.

2.2. Opis szczegółów implementacji problemu

2.2.1. Biblioteki

iostream	Biblioteka we-wyjścia. Deklaruje obiekty, które kontrolują odczytywanie ze strumieni standardowych i zapisywanie ich w tych strumieniach. Jest to często jedyny nagłówek potrzebny do wprowadzania danych i danych wyjściowych.
fstream	Dostarcza funkcji pozwalających nam zarówno zapisywać pliki jak i je odczytywać.
opracujgraf.h	Własna biblioteka, w której zawarte są zapowiedzi funkcji znajdujących się w pliku opracujgraf.cpp

2.2.2.Zmienne

A[n]	Tablica list posiadająca rozmiar równy ilości wierzchołków (każdy wierzchołek posiada swoją listę)
p	zmienna pomocnicza, która pełni rolę listy; za pomocą tej zmiennej możemy poruszać się po liście analizując poszczególne, a także na początku pomaga wpisać definicję grafu do list;
n	zmienna typu int; odpowiada ilości wierzchołków
m	zmienna typu int; odpowiada ilości krawędzi
v1	zmienna typu int; za jej pomocą wprowadzamy do grafu wierzchołki startowe;
v2	zmienna typu int; za jej pomocą wprowadzamy do grafu wierzchołki końcowe;
plik	zmienna globalna, plikowa typu ofstream. Służy do zapisywania danych do pliku
i,j	iteratory pętli for
istnieje, izolowany, warunek1, warunek2	zmienne typu bool;

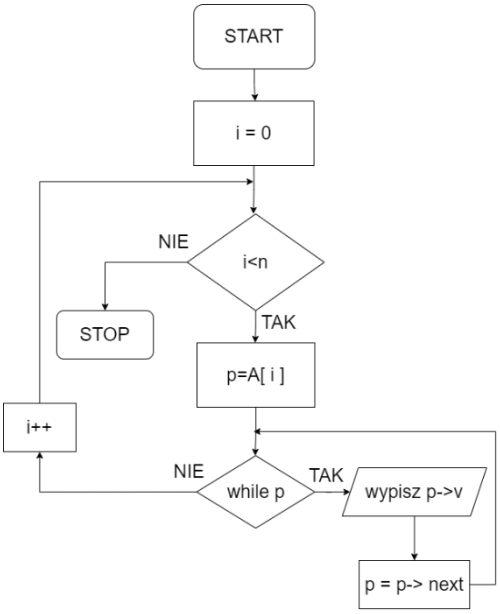
2.2.3. Funkcje

main	główna funkcja programu, zawiera inicjalizacje i deklaracje zmiennych; w niej wywoływane są pozostałe funkcje
sasiedzi	za pomocą pętli for wyświetla kolejne elementy poszczególnych list, odpowiadających odpowiednim wierzchołkom; wszystkie elementy, które są wywołane dla tablicy A[i], są sąsiadami wierzchołka „i"; wyświetla wynik
sasiedziKazdego Wierzcholka	za pomocą zagnieżdżonej pętli for, funkcja zlicza w ilu listach znajduję się i-ty element; jeśli ten licznik jest równy ilości wierzchołków to znaczy, że i-ty wierzchołek jest sąsiadem każdego wierzchołka
stopnieWychodzace	za pomocą pętli zlicza ile istnieje krawędzi wychodzących od i-tego wierzchołka; licznik ten oznacza stopień wychodzący wierzchołka; ostatecznie wypisuje wierzchołki i ich stopnie wychodzące
stopnieWchodzace	za pomocą zagnieżdżonej pętli for zlicza ile istnieje krawędzi (we wszystkich listach) wchodzących do i-tego wierzchołka; licznik ten oznacza stopień wchodzący wierzchołka; ostatecznie wypisuje wierzchołki i ich stopnie wychodzące
wierzcholkiIzolowane	za pomocą pętli for i zmiennych typu bool funkcja sprawdza czy istnieje jakakolwiek krawędź między i-tym wierzchołkiem; najpierw sprawdza ilość elementów na liście wierzchołka i; jeśli ilość ta jest równa zero, funkcja sprawdza czy wierzchołek i znajduje się na innych listach; jeśli nie – jest wierzchołkiem izolowanym; wypisuje wierzchołki izolowane
petle	za pomocą pętli i zmiennych bool sprawdza, czy istnieją takie krawędzie, które przechodzą od i-tego wierzchołka do tego samego wierzchołka (czy na liście i-tego wierzchołka istnieje i-ty element); ostatecznie wypisuje wierzchołek, który ma pętle
krawedz Dwukierunkowa	za pomocą pętli i zmiennych bool sprawdza, czy istnieją takie krawędzie, które przechodzą zarówno od i-tego do j-tego wierzchołka, jak i w drugą stronę (czy na liście i-tego wierzchołka jest j-ty element i odwrotnie); wypisuje między jakimi wierzchołkami istnieje krawędź dwukierunkowe
struct lista	zawiera definicje listy o nazwie „lista" oraz określa typ danych „v" w niej zawarty jako int

2.3. Schematy blokowe i pseudokody

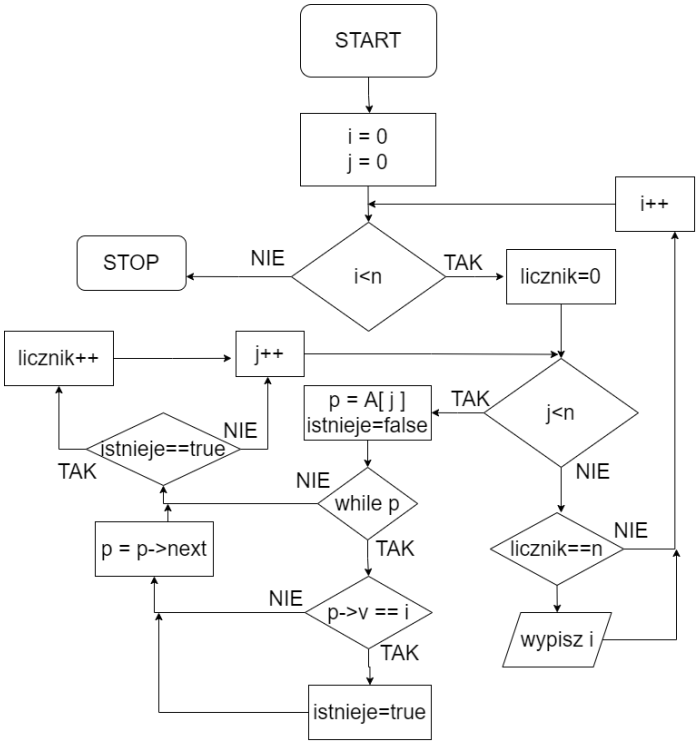
2.3.1. Sąsiedzi dla każdego wierzchołka grafu

```
i <- 0
dla i<-0 do n wykonuj
p <- A[ i ]
      dopóki p wykonuj
      wypisz p->v
      p->next
i++
```



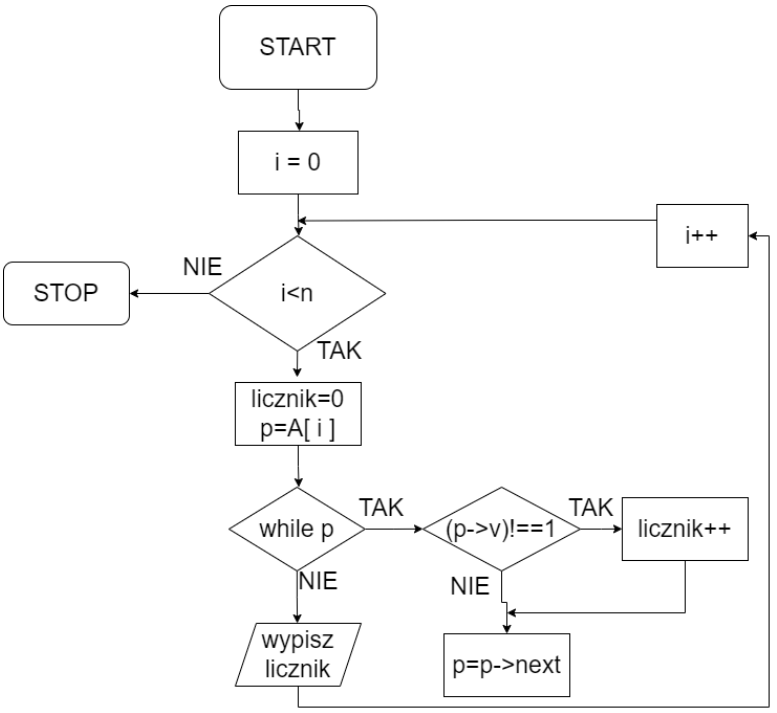
2.3.2. Wierzchołki, które są sąsiadami każdego wierzchołka

```
i,j <- 0
dla i<-0 do n wykonuj
licznik=0
      dla j<-0 do n wykonuj
      p<-A[ j ]
      istnieje=false
            dopoki p wykonuj
                  if (p->v ==i) wykonuj
                  istnieje=true;
            p= p->next
      if(istnieje==true)wykonuj
      licznik++
if(licznik==n)
wypisz i
i++
```



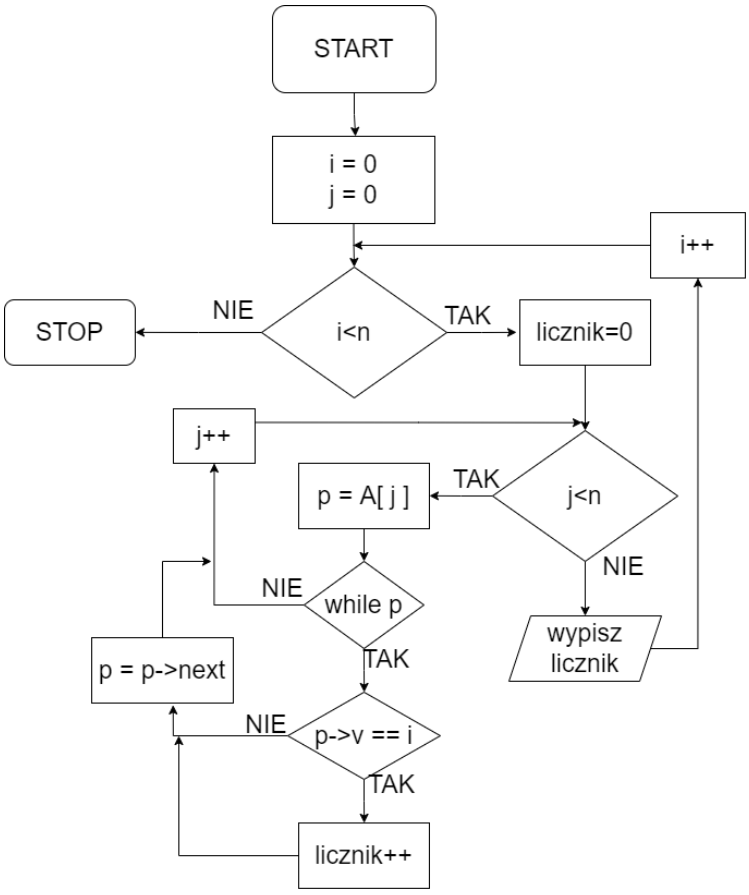
2.3.3. Stopnie wychodzące wierzchołków

```
i <- 0
dla i<-0 do n wykonuj
p <- A[ i ]
licznik=0
      dopóki p wykonuj
            if(p->v!=-1) wykonuj
            licznik++
      p=p->next
wypisz licznik
i++
```



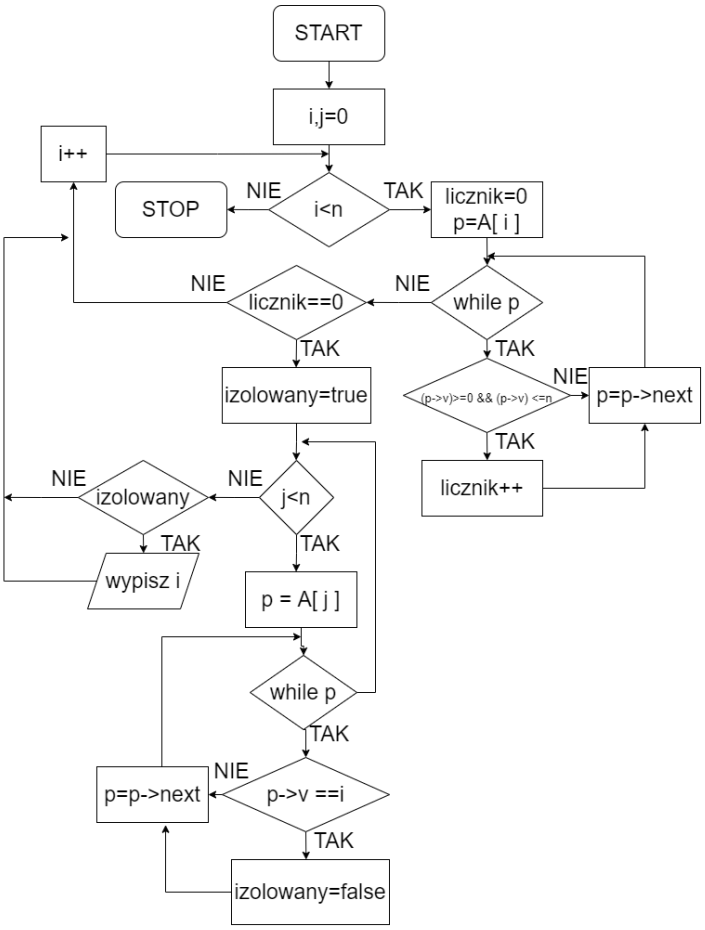
2.3.4. Stopnie wchodzące wierzchołków

```
i, j <- 0
dla i<-0 do n wykonuj
licznik<-0
      dla j<-0 do n wykonuj
      p <- A[ j ]
      dopóki p wykonuj
            if(p->v==i) wykonuj
            p=p->next
wypisz licznik
i++
```



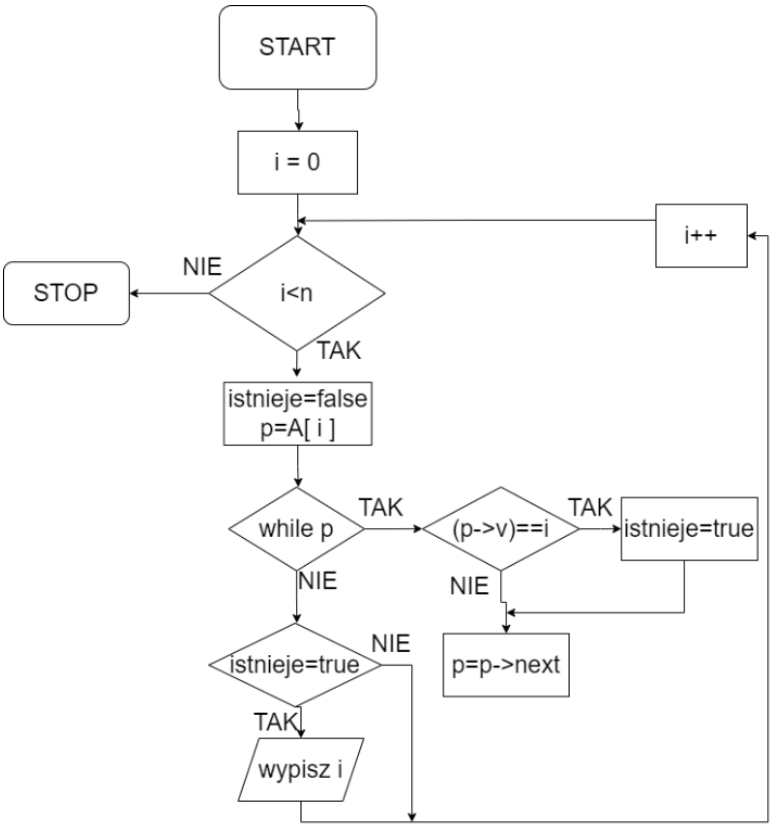
2.3.5. Wierzchołki izolowane

```
i, j <- 0
dla i<-0 do n wykonuj
licznik<-0
p <- A[ i ]
      dopóki p wykonuj
            if(p->v)>=0 &&(p->v)<=n
            wykonuj
            licznik++
      p=p->next
if licznik==0
izolowany<- true
      dla j<-0 do n wykonuj
      p<-A[ j ]
            dopoki p wykonuj
                  if(p->v == i)
wykonuj
                  izolowany=false
            p=p->next
            j++
if izolowany
      wypisz i
i++
```



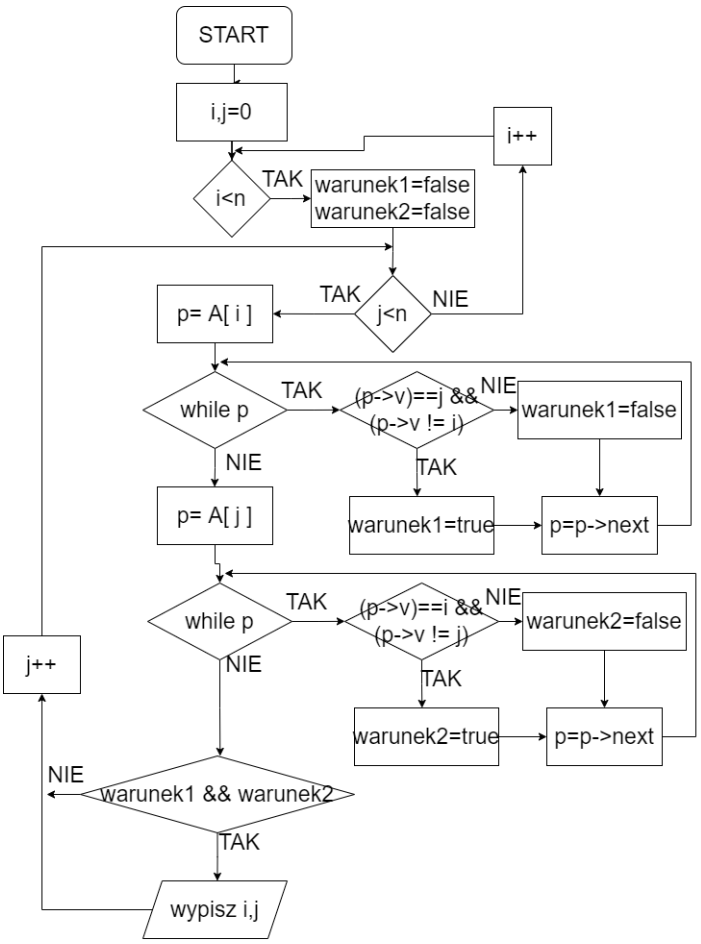
2.3.6.Pętle

```
i <- 0
dla i<-0 do n wykonuj
p <- A[ i ]
istnieje <- false
      dopóki p wykonuj
            if(p->v==i) wykonuj
            istnieje <-true
      p=p->next
if istnieje==true wykonuj
      wypisz i
i++
```



2.3.7.Krawędzie dwukierunkowe

```
i,j <- 0
dla i<-0 do n wykonuj
warunek1,warunek2 <- false
      dla j<-0 do n wykonuj
      p<- A[ i ]
      dopoki p wykonuj
            if(p->v==j && p->v!=i)
            wykonuj
            warunek1<-true
            else warunek1<-false
            p=p->next
      p<-A[ j ]
      dopoki p wykonuj
            if(p->v==i && p->v!=j)
            wykonuj
            warunek2<-true
            else warunek2<-false
            p=p->next
      if(warunek1&&warunek2)wykonuj
      wypisz i,j
      j++
i++
```



3. Wyniki

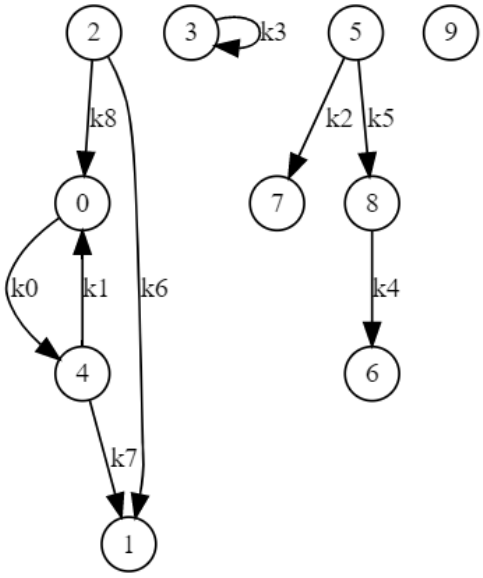
3.1. Przykłady działania programu

Poniżej przedstawione przykłady zostały skopiowane z pliku „wyniki.txt", do którego program zapisuje rezultaty swoich działań.

3.1.1. Pierwszy graf

Wejście:

Podaj liczbe wierzcholkow
10
Podaj liczbe krawedzi
9
Wprowadz graf za pomoca listy krawedzi (wierzcholek startowy, wierzcholek koncowy)
0 4
4 0
5 7
3 3
8 6
5 8
2 1
4 1
2 0



Wyjście:

1. Sasiedzi kazdego wierzcholka grafu:
W0 - sasiedzi: W4
W1 - sasiedzi:
W2 - sasiedzi: W0 W1
W3 - sasiedzi: W3
W4 - sasiedzi: W1 W0
W5 - sasiedzi: W8 W7
W6 - sasiedzi:
W7 - sasiedzi:
W8 - sasiedzi: W6
W9 - sasiedzi:

2. Wierzcholki, ktore sa sasiadami kazdego wierzcholka

3. Stopnie wychodzace wierzcholkow
W0 - stopien wychodzacy: 1
W1 - stopien wychodzacy: 0
W2 - stopien wychodzacy: 2
W3 - stopien wychodzacy: 1
W4 - stopien wychodzacy: 2
W5 - stopien wychodzacy: 2
W6 - stopien wychodzacy: 0

W7 - stopien wychodzacy: 0
W8 - stopien wychodzacy: 1
W9 - stopien wychodzacy:
4. Stopnie wchodzace wierzcholkow
W0 - stopien wchodzacy: 2
W1 - stopien wchodzacy: 2
W2 - stopien wchodzacy: 0
W3 - stopien wchodzacy: 1
W4 - stopien wchodzacy: 1
W5 - stopien wchodzacy: 0
W6 - stopien wchodzacy: 1
W7 - stopien wchodzacy: 1
W8 - stopien wchodzacy: 1
W9 - stopien wchodzacy: 0

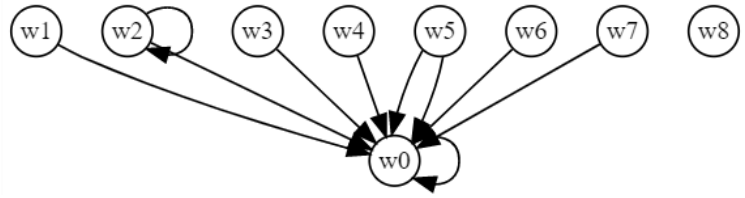
5. Wierzcholki izolowane
W9 - jest wierzcholkiem izolowanym

6. Petle
W3 - posiada petle

7. Krawedzie dwukierunkowe
Krawedz dwukierunkowa laczy wierzcholek W0 z wierzcholkiem W4
Krawedz dwukierunkowa laczy wierzcholek W4 z wierzcholkiem W0

3.1.2. Drugi graf

Wejście:
Podaj liczbe wierzcholkow
9
Podaj liczbe krawedzi
10
Wprowadz graf za pomoca listy krawedzi (wierzcholek startowy, wierzcholek koncowy)
1 0
5 0
6 0
0 0
2 0
3 0
4 0
7 0
2 2
8 0



Wyjście:
1. Sasiedzi kazdego wierzcholka grafu:
W0 - sasiedzi: W0
W1 - sasiedzi: W0
W2 - sasiedzi: W2 W0
W3 - sasiedzi: W0
W4 - sasiedzi: W0
W5 - sasiedzi: W0
W6 - sasiedzi: W0
W7 - sasiedzi: W0

W8 - sasiedzi: W0
2. Wierzcholki, ktore sa sasiadami kazdego wierzcholka
W0 - jest sasiadem kazdego wierzcholka

3. Stopnie wychodzace wierzcholkow
W0 - stopien wychodzacy: 1
W1 - stopien wychodzacy: 1
W2 - stopien wychodzacy: 2
W3 - stopien wychodzacy: 1
W4 - stopien wychodzacy: 1
W5 - stopien wychodzacy: 1
W6 - stopien wychodzacy: 1
W7 - stopien wychodzacy: 1
W8 - stopien wychodzacy: 1

4. Stopnie wchodzace wierzcholkow
W0 - stopien wchodzacy: 9
W1 - stopien wchodzacy: 0
W2 - stopien wchodzacy: 1
W3 - stopien wchodzacy: 0
W4 - stopien wchodzacy: 0
W5 - stopien wchodzacy: 0
W6 - stopien wchodzacy: 0
W7 - stopien wchodzacy: 0
W8 - stopien wchodzacy: 0

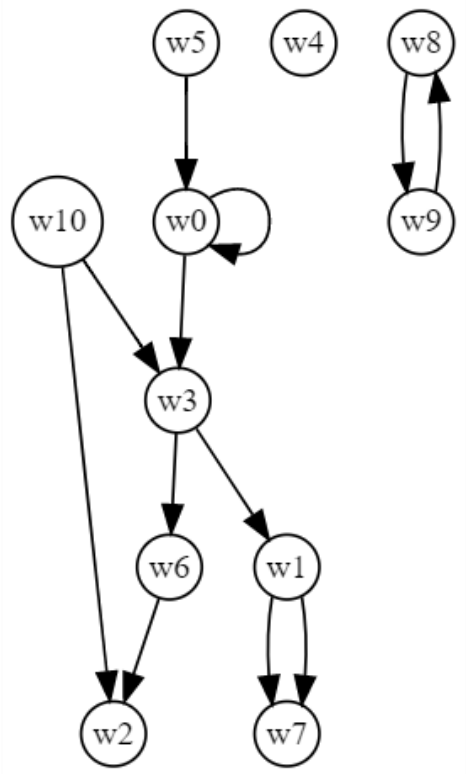
5. Wierzcholki izolowane

6. Petle
W0 - posiada petle
W2 - posiada petle

7. Krawedzie dwukierunkowe

3.1.3. Trzeci graf

Wejście:
Podaj liczbe wierzcholkow
11
Podaj liczbe krawedzi
12
Wprowadz graf za pomoca listy krawedzi (wierzcholek startowy, wierzcholek koncowy)
5 0
6 2
1 7
8 9
10 3
10 2
3 1
9 8
1 7
0 3
3 6
0 0



Wyjście:

1. Sasiedzi kazdego wierzcholka grafu:
W0 - sasiedzi: W0 W3
W1 - sasiedzi: W7 W7
W2 - sasiedzi:
W3 - sasiedzi: W6 W1
W4 - sasiedzi:
W5 - sasiedzi: W0
W6 - sasiedzi: W2
W7 - sasiedzi:
W8 - sasiedzi: W9
W9 - sasiedzi: W8
W10 - sasiedzi: W2 W3

2. Wierzcholki, ktore sa sasiadami kazdego wierzcholka

3. Stopnie wychodzace wierzcholkow
W0 - stopien wychodzacy: 2
W1 - stopien wychodzacy: 2
W2 - stopien wychodzacy: 0
W3 - stopien wychodzacy: 2
W4 - stopien wychodzacy: 0
W5 - stopien wychodzacy: 1
W6 - stopien wychodzacy: 1
W7 - stopien wychodzacy: 0
W8 - stopien wychodzacy: 1
W9 - stopien wychodzacy: 1
W10 - stopien wychodzacy: 2

4. Stopnie wchodzace wierzcholkow
W0 - stopien wchodzacy: 2
W1 - stopien wchodzacy: 1
W2 - stopien wchodzacy: 2
W3 - stopien wchodzacy: 2
W4 - stopien wchodzacy: 0
W5 - stopien wchodzacy: 0
W6 - stopien wchodzacy: 1
W7 - stopien wchodzacy: 2
W8 - stopien wchodzacy: 1
W9 - stopien wchodzacy: 1
W10 - stopien wchodzacy: 0

5. Wierzcholki izolowane
W4 - jest wierzcholkiem izolowanym

6. Petle
W0 - posiada petle

7. Krawedzie dwukierunkowe
Krawedz dwukierunkowa laczy wierzcholek W8 z wierzcholkiem W9
Krawedz dwukierunkowa laczy wierzcholek W9 z wierzcholkiem W8

4.Podsumowanie

4.1. Wnioski

Podsumowując, można stwierdzić, że program poprawnie spełnia swoją funkcję. Nie jest żadną tajemnicą, że istnieje bardziej optymalna wersja tego programu, jednak brak obycia w owym temacie i wiedzy z zakresu programowania nie pozwoliły mi, abym napisał program w bardziej optymalnej wersji.

4.2.Źródła i pomoce

- dokładne opisy i definicje bibliotek - https://docs.microsoft.com/pl-pl/cpp/standard-library/cpp-standard-library-header-files?view=msvc-160 oraz https://cpp0x.pl/kursy/Kurs-C++/Dodatkowe-materialy/Obsluga-plikow/305

- tworzenie definicji grafów, grafy przedstawiane graficznie - http://www.algorytm.org/narzedzia/edytor-grafow.html

- tworzenie schematów blokowych - https://app.diagrams.net

- początkowy zarys kodu programu, implementacja grafu - https://eduinf.waw.pl/inf/alg/001_search/0124.php

- definicje dotyczące grafów i informacji na ich temat – https://eduinf.waw.pl/inf/alg/001_search/0123.php

- informacje dotyczące list – https://eduinf.waw.pl/inf/alg/001_search/0085.php