Politechnika Śląska w Gliwicach
Wydział Automatyki, Elektroniki i Informatyki

Podstawy Programowania Komputerów

Cykl

autor	Dawid Kaczor
prowadzący	dr inż. Krzysztof Simiński
rok akademicki	2016/2017
kierunek	informatyka
rodzaj studiów	SSI
semestr	1
termin laboratorium	wtorek, 12:00-13:30
grupa	6
sekcja	2
termin oddania sprawozdania	2017-01-26
data oddania sprawozdania	2017-01-29

1 Treść zadania

Napisać program do wyznaczania cykli w grafie skierowanym. W pliku wejściowym podane są krawędzie pewnego grafu. Są one zapisane w następujący sposób:

```
<wierzchołek początkowy> -> <wierzchołek końcowy>
```

Poszczególne łuki są rozdzielone przecinkami. Przykładowy plik wejściowy:

```
2 -> 3, 1 -> 3, 3

     ->


          3,3
->3
```

W pliku wynikowym zostają zapisane znalezione cykle (każdy cykl w osobnej linii) lub informacja, że w grafie nie występują cykle.

Program uruchamiany jest z linii poleceń z wykorzystaniem następujących przełączników:

- `-g` plik wejściowy z grafem
- `-c` plik wyjściowy ze znalezionymi cyklami

2 Analiza zadania

Zagadnienie przedstawia problem wyszukiwania cyklu w grafie zapisanym w pliku.

2.1 Struktury

W programie wykorzystano listę podwieszaną jednokierunkową. Główna lista przechowuje numery wierzchołków, informację czy dany wierzchołek został już odwiedzony (przydatne przy wyszukiwaniu cyklu) oraz wskaźnik na pierwszy element jednokierunkowej listy podrzędnej. Lista ta symbolizuje krawędzie i przechowuje wskaźniki na wierzchołki z którymi połączony jest wierzchołek z listy nadrzędnej. Wykorzystano również listę jednokierunkową symbolizującą stos, na który odkładane są numery wierzchołków które zostały odwiedzone w trakcie wyszukiwania cyklu

2.2 Algorytmy

Program zapisuje dane z pliku do programu poprzez sprawdzenie czy na liście istnieje każdy z podanych wierzchołków i dodanie brakujących, następnie dodanie połączeń w podliście. Wyszukiwanie cyklu polega na przejściu przez każdy z wierzchołków w głównej liście, w każdym kroku następuje przejście do listy krawędzi i porównanie z wierzchołkiem. Jeśli są one sobie równe, znaleziono cykl, w przeciwnym wypadku następuje kolejne wyszukanie, tym razem dla kolejnej krawędzi.

3 Specyfikacja zewnętrzna

Program jest uruchamiany z linii poleceń. Należy przekazać do programu nazwy plików: wejściowego i wyjściowego po odpowiednich przełącznikach (odpowiednio: `-g` dla pliku wejściowego i `-c` dla pliku wyjściowego), np.

```
program -g wejscie.txt -c wyjscie.txt
program -c wyjscie.txt -g wejscie.txt
```

Przełączniki mogą być podane w dowolnej kolejności. Uruchomienie programu bez żadnego parametru lub z parametrem `-h`

```
program
program -h
```

jak i uruchomienie programu z nieprawidłowymi parametrami powoduje wyświetlenie komunikatu

```
Witam w programie wyszukującym cykle w grafie
skierowanym

Jak uruchomić program?
W wierszu poleceń należy wpisać nazwę programu z
rozszerzeniem .exe, następnie po przełączniku -g
nazwę pliku tekstowego z danymi wejściowymi i po
przełączniku -c nazwę pliku tekstowego w którym mają
zapisane zostać dane wyjściowe.

Jak mają wyglądać dane wejściowe?
Należy podać krawędzie w następujący sposób:
<wierzchołek początkowy> -> <wierzchołek końcowy>,
gdzie wierzchołki są liczbami naturalnymi a każda
krawędź jest zakończona przecinkiem, na przykład:
1->2, 2->3, 3->1,
```

4 Specyfikacja wewnętrzna

W programie funkcje operujące na plikach i zmiennych zostały oddzielone od funkcji głównej.

4.1 Typy zdefiniowane w programie

W programie zdefiniowano następujące typy

```
struct wierzcholek
{
     int numer;
     bool odwiedziny;
     wierzcholek *pNext;
     krawedz *pKrawedzie;
};
```

Typ ten służy do przechowywania wierzchołków.

```
struct krawedz
{
     wierzcholek *pEdge;
     krawedz *pDown;
};
```

Typ ten służy do przechowywania krawędzi

```
struct stos
{
     int number;
     stos *pNext;
};
```

Typ ten służy do przechowywania numerów odwiedzonych już wierzchołków.

4.2 Funkcje

```
bool odczyt (int ile, char ** argumenty, string
&szinput, string &szoutput)
```

Funkcja wczytuje parametry wywołania programu i sprawdza ich poprawność, są one podawane w linii poleceń jako przełączniki w dowolnej kolejności

```
-g input
-c output
-h
```

Funkcja pobiera następujące parametry

`ile`	liczba parametrów podanych przy uruchomieniu programu
`argumenty`	tablica wskaźników na łańcuchy użytych przy uruchomieniu
`szinput`	parametr wyjściowy, do którego zostanie zapisana odczytana nazwa pliku wejściowego
`szoutput`	parametr wyjściowy, do którego zostanie zapisana odczytana nazwa pliku wyjściowego

Funkcja zwraca następujące wartości

`true`	podano prawidłowe parametry wywołania programu, zostały one odczytane i zapisane do `szinput` i `szoutput`
`false`	nie udało się odczytać wartości parametrów wywołania programu, wartości zmiennych `szinput` i `szoutput` mogą nie zawierać istotnych wartości

Funkcja `zwroc_adres_wierzcholka` wyszukuje w pamięci wierzchołek o podanym numerze

```
wierzcholek *zwroc_adres_wierzcholka (wierzcholek *&pHead, int szukana)
```

Funkcja przyjmuje następujące parametry

`*&pHead`	adres pierwszego elementu listy wierzchołków
`szukana`	numer szukanego wierzchołka

Funkcja zwraca adres znalezionego wierzchołka.

Procedura `dodaj_krawedz` dodaje do podlisty pierwszego wierzchołka wskaźnik na drugi wierzchołek połączony krawędzią

```
void dodaj_krawedz (wierzcholek *&pHead, int pierwszy, int drugi)
```

Procedura przyjmuje następujące parametry

`*&pHead`	adres pierwszego elementu listy wierzchołków
`pierwszy`	wierzchołek początkowy
`drugi`	wierzchołek końcowy

Funkcja `usun_strzalke` zamienia w łańcuchu znaki pomiędzy liczbami na znaki białe

```
string usun_strzalke (string &linia)
```

Funkcja przyjmuje następujące parametry

`&linia`	referencja na łańcuch w którym należy dokonać zamiany

Funkcja zwraca łańcuch zawierający tylko cyfry i znaki białe.

Procedura `usun_szczyt_stosu` usuwa z pamięci pierwszy element stosu

```
void usun_szczyt_stosu (stos *&pHead)
```

Procedura przyjmuje następujące parametry

`*&pHead`	adres elementu do usunięcia

Procedura `usun_caly_stos` usuwa cały stos i zwalnia po nim pamięć

```
void usun_caly_stos (stos *&pHead)
```

Procedura przyjmuje następujące parametry

`*&pHead`	adres pierwszego elementu stosu

Procedura `dodaj_na_stos` dodaje element na szczyt stosu

```
void dodaj_na_stos (stos *&pHead, int numer)
```

Procedura przyjmuje następujące parametry

`*&pHead`	adres pierwszego elementu stosu
`numer`	liczba, którą należy dodać na stos

Procedura `wypisz_stos` wypisuje do pliku wszystkie elementy stosu

```
void wypisz_stos(stos *pHead, ofstream &plik)
```

Procedura przyjmuje następujące parametry

`*&pHead`	adres pierwszego elementu stosu
`&plik`	plik do którego należy wypisać elementy stosu

Funkcja `szukaj_cyklu` wyszukuje cykl w grafie

```
bool szukaj_cyklu (wierzcholek *glowa, int szukana, stos *&peak)
```

Funkcja przyjmuje następujące parametry

`*glowa`	adres wierzchołka dla którego wyszukiwany jest cykl
`szukana`	liczba której szuka funkcja
`*&peak`	adres stosu na który odkładane są liczby

Funkcja zwraca następujące wartości

`True`	cykl został znaleziony
`False`	nie znaleziono cyklu

Procedura `zeruj_odwiedziny` przypisuje całej liście wierzchołków w polu `odwiedziny` wartość `false`

```
void zeruj_odwiedziny(wierzcholek *&pHead)
```

Procedura przyjmuje następujące parametry

`*&pHead`	adres pierwszego elementu listy

Procedura `usun_liste_krawedzi` usuwa podlistę danego wierzchołka i zwalnia po niej pamięć

```
void usun_liste_krawedzi (krawedz *&pHead)
```

Procedura przyjmuje następujące parametry

`*&pHead`	adres pierwszego elementu podlisty

Procedura usun_liste_wierzcholkow usuwa główną listę i zwalnia po niej pamięć

```
void usun_liste_wierzcholkow (wierzcholek *&pHead)
```

Procedura przyjmuje następujące parametry

`*&pHead`	adres pierwszego elementu listy

5 Testowanie

Program został przetestowany na różnego rodzaju danych wejściowych. Przy prawidłowych parametrach nie zaobserwowano żadnych błędnych zapisów. Pusta linia powoduje przejście do następnej linii w pliku wejściowym. Program został sprawdzony pod kątem wycieków pamięci

6 Wnioski

Program wyszukujący cykl w grafie okazał się dla mnie sporym wyzwaniem. Jego implementacja wymagała wiedzy zarówno z zakresu programowania jak i teorii grafów. Projekt ten pomógł mi przećwiczyć operacje na strukturach danych i wskaźnikach. Zadanie było z pewnością ciekawe i rozwijające.