Politechnika Śląska
Wydział Automatyki, Elektroniki i Informatyki

# Podstawy Programowania Komputerów

Temat: ALOPS (1)

| | |
|---|---|
| Autor | Piotr Hasiec |
| Prowadzący | Dr inż. Krzysztof Simiński |
| Rok akademicki | 2019/2020 |
| Kierunek | Informatyka |
| Rodzaj studiów | SSI |
| Semestr | 1 |
| Termin laboratorium | Poniedziałek, 10:15 - 12:45 |
| Sekcja | 11 |
| Termin oddania sprawozdania | 2020-01-23 |

## 1. Treść zadania

Napisać program do analizy liniowych obwodów prądu stałego. Elementami, które mogą wystąpić w obwodzie, to: siła elektromotoryczna, siła prądowa, rezystor.

W pliku wynikowym zostają wypisane elementy obwodu, natężenie prądu, spadek napięcia na elemencie i moc wydzielona. Sporządzany jest także bilans mocy układu. Program uruchamiany jest z linii poleceń z wykorzystaniem następujących przełączników:
-i plik wejściowy z obwodem
-o plik wyjściowy z wyliczonymi wartościami

## 2. Analiza zadania

Zadanie porusza problem reprezentacji grafu jako struktury danych, przeszukiwania listy i zastosowania metod numerycznych do rozwiązywania liniowych układów równań liniowych.

### 2.1. Struktury danych

W programie wykorzystano reprezentację grafu jako listy sąsiedztwa. Pozwala to na korzystanie z zalet listy i względnie łatwy dostęp do każdego elementu obwodu. Elementy listy zwane później wierzchołkami przechowują numer, potencjał, wskaźnik na następny element listy i wskaźnik na listę elementów zwanych w dalej połączeniami. Połączenia zawierają:

- Nr wierzchołka w którym się kończą,
- typ (I dla SPM, E dla SEM, R dla rezystorów),
- wartość oporu, prądu wymuszenia, lub wzrostu napięcia (w zależności od typu),
- prąd płynący przez połączenie.
- Zaimplementowano dodatkową zmienną pozwalającą przy wypisywaniu rozróżnić gałęzie.

Reprezentacja ta wymaga pamięci proporcjonalnie do sumy liczby wierzchołków i liczby krawędzi grafu.

### 2.2. Algorytmy

Program ma dwie zasadnicze funkcje – utworzenie macierzy konduktancji i wymuszeń prądowych dla zadeklarowanego obwodu oraz rozwiązanie tak stworzonego układu równań.

W mojej implementacji pierwszej z procedur wymagane jest wypełnienie macierzy $VxV$ gdzie $V$ to ilość wierzchołków w grafie. Do wypełnienia każdej komórki należy przeszukać listę. Dostęp do poszczególnych krawędzi ma złożoność $O(n)$ zatem całość wypełnienia macierzy ma złożoność $O(V^2 n)$. Rozwiązywanie układu równań liniowych z kolei ma złożoność obliczeniową $V(n^3)$.

## 3. Specyfikacja zewnętrzna

Program uruchamiany z wiersza poleceń przyjmuje z parametrami wejściowymi nazwy (i rozszerzenia) plików: wyjściowego i wejściowego, podanych po przełącznikach. Koleino: wejściowego po przełączniku –i, a wyjściowego po przełączniku –o. Pliki te są pli-

kami tekstowymi o dowolnym rozszerzeniu. W przypadku braku jednego z parametrów program zapyta użytkownika o nazwę odpowiedniego pliku. W przypadku uruchomienia programu z błędnym parametrem parametr ten zostanie zignorowany. Uruchomienie programu z parametrem –h wyświetli krótką pomoc dla użytkownika.
Program przyjmuje dane w postaci:

<Typ> <Nr węzła początkowego> <Nr węzła końcowego> <Wartość>

Typ to:
R – dla opornika
I – dla źródła prądowego
E – Dla źródła napięciowego.

Strzałkowanie prądów i wzrostu napięć został przyjęte od węzła początkowego do węzła końcowego.
Numery węzłów mogą przyjmować tylko wartości całkowite. Wartość ma wymiar zależny od typu elementu. Nie należy podawać jednostki.

W przypadku problemu z otwarciem pliku program wyświetli komunikat:
„Błąd odczytu. Sprawdź czy podany plik istnieje i znajduje się w odpowiednim katalogu”

W przypadku podania wewnątrz pliką błędnych danych:
„Błędna wartość typu” lub „Zadeklarowano zerowy lub ujemny opór”

W przypadku gdy dany w pliku obwód jest nierozwiązywalny lub niespójny:
„Równanie sprzeczne lub tożsamościowe”

## 4. Specyfikacja wewnętrzna

Program został zrealizowany zgodnie z paradygmatem strukturalnym.

### 4.1. Ogólna struktura programu

W funkcji głównej znajduje się pętla przetwarzająca parametry wejściowe pliku. I w razie potrzeby zapisująca je do odpowiednich zmiennych typu string. Po przetworzeniu parametrów wywołana jest funkcja *download.* Pobiera ona dane z pliku którego nazwa została przekazana jako parametr funkcji głównej. W przypadku błędów w odczycie wyświetla stosowny komunikat i zwraca wartość false. W przypadku nieprawidłowego zadeklarowania danych wyświetla odpowiedni komunikat, przerywa wczytywanie danych do struktury i zwraca wartość false. W tym przypadku struktura jest usuwana i program się kończy, w przeciwnym program jest kontynuowany.

Następnie wywoływana jest funkcja *count* zwracająca liczbę wierzchołków w grafie. Jej wartość zapisywana jest do zmiennej *sizeofgrap.* Jest to parametr który będzie następnie przekazywany do funkcji oraz wykorzystywany przy zwalnianiu pamięci z tablicy będącej reprezentacją układu równań.

Kolejnym wywołaniem jest funkcja *makematrix* która alokuje dynamiczną tablice będącą reprezentacją układu równań liniowych opisujących zadany obwód. Macierz ta ma wymiary $(V-1)\ x\ V$

Gdzie $V$ to liczba wierzchołków.

Zwraca ona wskaźnik na tę właśnie tablicę lub, w przypadku gdy lista była pusta, nullptr.

W przypadku gdy zaalokowana tablica nie jest pusta wywoływana jest funkcja *gauss*. Jako parametry przyjmuje ona wskaźnik na tablicę oraz liczbę wierzchołków grafu z którego powstał układ równań. Rozwiązuje ona układ równań metodą przekształceń liniowych Gaussa. W przypadku równania tożsamościowego lub sprzecznego funkcja wyświetla stosowny komunikat i zwraca wartość false. W przypadku rozwiązywalności układu równań zwraca wartości potencjałów w ostatnim rzędzie tablicy. W przypadku poprawnego rozwiązania potencjały zapisywane są do struktury a następnie wywoływana jest funkcja *generate_currents.* Funkcja ta na podstawie analizy rozkładu potencjałów przypisuje połączeniom odpowiednie wartości prądów przez nie płynących. W tym momencie następuje wywołanie funkcji *save* - zapisującej do pliku (również wypisującej odpowiednie komunikaty w przypadku niepowodzenia). A następnie zwolnienie pamięci zajmowanej przez dynamicznie zaalokowaną tablicę oraz wywołana zostaje funkcje *delete_vertices_forend_r* zwalniająca pamięć zajmowaną przez graf.

### 4.2. Szczegółowy opis typów i funkcji

Szczegółowy opis typów i funkcji zawarty jest w załączniku.

## 5. Testowanie

Program został przetestowany na prawidłowych danych typowych - obwody zamknięte, spójne, niesprzeczne - i w ich przypadku daje prawidłowe wyniki. W przypadku obwodów nie zamkniętych (np. samo źródło napięciowe, samo źródło prądowe) ale spójnych również generuje poprawne wyniki. W przypadku zadeklarowania obwodu sprzecznego – nierozwiązywalnego lub więcej niż jednego obwodu wyświetla komunikat: „Zadeklarowano sprzeczny lub niespójny graf obwodu” i nie generuje pliku wyjściowego. W przypadku zadeklarowanie pustego obwodu program komunikuje błąd oraz również nie generuje pliku wyjściowego. Największa liczba zadeklarowanych elementów obwodu dla której udało się poprawnie uruchomić program wynosi 4927. W przypadku większej liczby gałęzi występuje problem przy zwalnianiu zaalokowanej pamięci.

## 6. Wnioski

ALOPS był najbardziej rozbudowanym projektem nad jakim pracowałem. Wymaga samodzielnego zarządzania pamięcią co w wielu przypadkach było kłopotliwe. W związku z brakiem znajomości destruktorów należało również pamiętać o ręcznym zwalnianiu pamięci w każdym przypadku prowadzącym do zamknięcia programu. Kolejnym angażującym i trudnym elementem była obsługa wszystkich wyjątków które mogły zaistnieć w trakcie pisania programu.

## 7. Źródła

Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest. Wprowadzenie do algorytmów. Wydawnictwa Naukowo-Techniczne, Warszawa, 2001.

Adam Macura. Teoria obwodów  Obwody prądu stałego.  Wydawnictwo Politechniki Śląskiej, Gliwice, 1997

http://www.cplusplus.com/

http://cpp0x.pl/