**TRƯỜNG ĐẠI HỌC BÁCH KHOA**

**KHOA CÔNG NGHỆ THÔNG TIN**



**ĐỒ ÁN CƠ SỞ**

***ĐỀ TÀI 309:***

***ĐỌC GHI FILE VÀ GIẢI HỆ PHƯƠNG TRÌNH BẰNG PHƯƠNG PHÁP LẶP ĐƠN***

Người hướng dẫn**: TS.PHẠM CÔNG THẮNG**

Sinh viên thực hiện**: LÊ ĐẶNG TRƯỜNG ĐẠT**

**LỚP: 18TCLC\_DT3**

**NHÓM: 10B**

**Đà Nẵng, 06/2020**

MỤC LỤC

[MỤC LỤC i](#_Toc43038055)

[DANH MỤC HÌNH VẼ VÀ BẢNG iii](#_Toc43038056)

[LỜI MỞ ĐẦU 1](#_Toc43038057)

[1. TỔNG QUAN ĐỀ TÀI 3](#_Toc43038058)

[2. CƠ SỞ LÝ THUYẾT 4](#_Toc43038059)

[2.1. Ý tưởng: 4](#_Toc43038060)

[2.2. Cơ sở lý thuyết: 4](#_Toc43038061)

[3. TỔ CHỨC CẤU TRÚC DỮ LIỆU VÀ THUẬT TOÁN 9](#_Toc43038062)

[3.1. Phát biểu bài toán 9](#_Toc43038063)

[3.2. Cấu trúc dữ liệu 10](#_Toc43038064)

[3.3. Thuật toán 10](#_Toc43038065)

[4. CHƯƠNG TRÌNH VÀ KẾT QUẢ 11](#_Toc43038066)

[4.1. Tổ chức chương trình 11](#_Toc43038067)

[4.2. Ngôn ngữ cài đặt 11](#_Toc43038068)

[4.3. Kết quả 11](#_Toc43038069)

[4.3.1. Giao diện chính của chương trình 11](#_Toc43038070)

[4.3.2. Kết quả thực thi của chương trình 11](#_Toc43038071)

[4.3.3. Nhận xét đánh giá 14](#_Toc43038072)

[5. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 15](#_Toc43038073)

[5.1. Kết luận 15](#_Toc43038074)

[5.2. Hướng phát triển 15](#_Toc43038075)

[TÀI LIỆU THAM KHẢO 16](#_Toc43038076)

[[1] Đỗ Thị Tuyết Hoa, Bài giảng môn Phương pháp tính, 2007 16](#_Toc43038077)

[[2] Nguyễn Văn Hiếu, Đọc ghi file trong C, https://nguyenvanhieu.vn/doc-ghi-file-trong-c/ , ngày truy cập: 5/4/2020 16](#_Toc43038078)

[[3] Lê Minh Hoàng, Giải thuật và lập trình, 1999 – 2002 16](#_Toc43038079)

[PHỤ LỤC 17](#_Toc43038080)

DANH MỤC HÌNH VẼ VÀ BẢNG

[Bảng 1: Bảng các tham số mode 5](#_Toc41905695)

[Hình 1: Mô phỏng nút trong danh sách liên kết kép [3] 6](#_Toc43038009)

[Hình 2: Mô phỏng sự liên kết giữa các nút trong danh sách liên kết kép [3] 6](#_Toc43038010)

[Hình 3: Hình minh họa dữ liệu đầu vào (Input) 9](#_Toc43038011)

[Hình 4: Hình minh họa dữ liệu đầu vào (2) (Input) 9](#_Toc43038012)

[Hình 5: Hình minh họa dữ liệu đầu vào (3) (Input) 9](#_Toc43038013)

[Hình 6: Hình minh họa dữ liệu đầu ra trong file RESULT1.OUT (Output) 9](#_Toc43038014)

[Hình 7: Giao diện chính của chương trình 11](#_Toc43038015)

[Hình 8: Dãy số được đọc từ file 12](#_Toc43038016)

[Hình 9: Nhập các số để chèn, các dãy số sau khi được chèn (Console) 12](#_Toc43038017)

[Hình 10: Kết quả trong file RESULT1.OUT sau thi chèn số vào các dãy số 12](#_Toc43038018)

[Hình 11: Ma trân B và hệ số của hệ phương trình 13](#_Toc43038019)

[Hình 12: Mô tả cách thức thực hiện của chương trình với nghiệm ban đầu 13](#_Toc43038020)

[Hình 13: Chương trình tự động tính toán nghiệm xấp xỉ ban đầu 13](#_Toc43038021)

[Hình 14: Mô phỏng cách bạn tự nhập nghiệm xấp xỉ ban đầu 13](#_Toc43038022)

[Hình 15: Kết quả trên màn hình Console 14](#_Toc43038023)

[Hình 16: Kết quả trên file RESULT2.OUT 14](#_Toc43038024)

[Hình 17: Sai số của mỗi phương trình 14](#_Toc43038025)

[Hình 18: Mô phỏng chương trình hỏi bạn “có muốn tiếp tục không” 14](#_Toc43038026)

LỜI MỞ ĐẦU

Đồ án cơ sở là một trong những đồ án quan trọng giúp cho sinh viên có thể sử dụng các kiến thức đã được học ( bao gồm: kĩ thuật lập trình, phương pháp tính, cấu trúc dữ liệu) để áp dụng giải quyết một số vấn đề trong cuộc sống.

Được giao đề tài số 309 và được Thầy Phạm Công Thắng hướng dẫn, em rất vui và đã hoàn thành được những nhiệm vụ được giao.

1. Mục đích thực hiện đề tài:

* Giúp cho sinh viên thấy được ứng dụng của các bộ môn đã được học (kĩ thuật lập trình, phương pháp tính, cấu trúc dữ liệu).
* Giúp sinh viên rèn luyện thuật toán (Algorithm).
* Giúp rèn luyện kĩ năng giải quyết vấn đề cho sinh viên.
* Giúp cho sinh viên giải được một số ma trận bằng phương pháp lập trình.

1. Mục tiêu đề tài:

* Sinh viên sử dụng thành thạo ngôn ngữ C, nắm rõ thuật toán về danh sách liên kết kép cũng như phương pháp lặp đơn để giải ma trận.
* Sinh viên có tính tự giác hơn trong việc tìm tòi các kiến thức mới (cách đọc file, xuất dữ liệu ra file).
* Sinh viên biết cách thêm phần tử vào danh sách liên kết kép.
* Giải quyết được các yêu cầu do giảng viên hướng dẫn giao.
* Đọc được dữ liệu từ file và ghi được dữ liệu ra file.

1. Phạm vi và đối tượng nghiên cứu:

* Danh sách liên kết kép.
* Thuật toán thêm một phần tử vào danh sách liên kết kép.
* Phương pháp lặp đơn để giải hệ phương trình.
* Mảng 1 chiều, mảng 2 chiều.
* Mở file, đọc file, ghi file, đóng file.
* Hàm, biến cục bộ, biến toàn cục, con trỏ, câu lệnh điều kiện, vòng lặp, …

1. Phương pháp nghiên cứu:

* Tổng hợp các kiến thức liên môn để giải quyết vấn đề trong bài toán.
* Tham khảo sách giáo trình và tìm tài liệu từ internet để đọc thêm.
* Hỏi giảng viên hướng dẫn về những thắc mắc trong quá trình nghiên cứu.

1. Cấu trúc của đồ án môn học:

* Gồm 5 phần chính:
* Tổng quan đề tài.
* Cơ sở lý thuyết.
* Tổ chức cấu trúc dữ liệu và thuật toán.
* Chương trình và kết quả.
* Kết luận và hướng phát triển.

# TỔNG QUAN ĐỀ TÀI

Viết chương trình bằng ngôn ngữ C thực hiện các công việc sau:

a) Đọc vào N danh sách từ file DAYSO.IN (Gợi ý: trong file DAYSO.IN bao gồm hàng , mỗi hàng gồm M phần tử số, M=N-1) . Mỗi hàng này tương ứng là một danh sách liên kết.

b) Thực hiện thêm vào mỗi danh sách liên kết một phần tử trước phần tử thứ (làm tròn nếu lẻ) , hiển thị kết quả và lưu kết quả ra file *RESULT1.OUT* .

c) Kết quả sau khi thực hiện ở câu b), chúng ta nhận được danh sách liên kết, mỗi danh sách có phần tử. Biết rằng các phần tử này là các giá trị được biểu diễn theo ma trận sau (mỗi danh sách tương ứng mỗi hàng của ma trận):

Hãy tìm nghiệm của hệ phương trình bằng ***phương pháp lặp đơn***, với (B là mảng một chiều bất kỳ nhập vào), hiển thị kết quả và lưu kết quả ra file *RESULT2.OUT* .

**Yêu cầu**: Sử dụng ***danh sách liên kết kép*** để thực hiện các công việc trên.

**Dữ liệu đầu vào**: file DAYSO.IN.

**Dữ liệu đầu ra**: kết quả lưu ở 2 file *RESULT1.OUT, RESULT2.OUT*.

# CƠ SỞ LÝ THUYẾT

## Ý tưởng:

* **Giải quyết câu a:**
* Dùng một con trỏ \*file để đọc N từ file DAYSO.IN và gán giá trị đó cho biến N. Từ đó rút ra M.
* Dùng con trỏ \*file để đọc các giá trị được lưu trong file DAYSO.IN (Các giá trị trong cùng một hàng được liên kết với nhau theo danh sách liên kết kép và các hàng ấy cũng được liên kết với nhau theo một danh sách liên kết kép).
* **Giải quyết câu b:**
* Tính toán giá trị M/2 (làm tròn nếu M lẻ) và tính toán vị trí liền trước M/2. Sau đó ta sử dụng thuật toán thêm một phần tử (số) vào danh sách liên kết kép và sử dụng con trỏ \*file2 để ghi kết quả ra file RESULT1.OUT.
* **Giải quyết câu c:**
* Ta đọc các giá trị từ file RESULT1.OUT và sao chép chúng vào một mảng 2 chiều HS (mảng hệ số gồm N hàng và N cột).
* Cho người dùng nhập vào một mảng B sau đó sao chép các giá trị của mảng B vào cột thứ N+1 của mảng 2 chiều HS (mảng hệ số).
* Như vậy ta có một ma trận hệ số gồm N hàng và N+1 cột.
* Áp dụng phương pháp lặp đơn kết hợp với điều kiện để giải ma trận gồm N hàng và N+1 cột này.

## Cơ sở lý thuyết:

1. **Về kỹ thuật lập trình [2]:**

* Trong ngôn ngữ lập trình C, có một số thao tác chính khi làm việc với file:
* Tạo mới một file.
* Mở một file đã có.
* Đóng file đang mở.
* Đọc thông tin từ file, ghi thông tin ra file.
* Thao tác với file trên ngôn ngữ C:
* Khi làm việc với file, ta cần khai báo 1 con trỏ kiểu FILE. Việc khai báo này là cần thiết để có sự kết nối giữa chương trình của ta và tập tin mà ta cần thao tác.

VD: FILE \*file;

* Để đọc ghi file trong C cũng như trong mọi ngôn ngữ lập trình, việc đầu tiên bạn cần làm là mở file mà bạn muốn làm việc. Trong ngôn ngữ lập trình C, chúng ta có thể mở file bằng cách sử dụng hàm fopen(“file\_name”, ”mode”) trong thư viện stdio.h. Ý nghĩa các tham số mode được liệt kê trong bảng 1.

VD: file = fopen(“DAYSO.IN”,”r”);

|  |  |  |
| --- | --- | --- |
| Mode | Ý nghĩa | Nếu file không tồn tại |
| r | Mở file chỉ cho phép đọc. | Nếu file không tồn tại, fopen() trả về NULL. |
| w | Mở file chỉ cho phép ghi. | Nếu file đã tồn tại, nội dung sẽ bị ghi đè. Nếu file không tồn tại, nó sẽ được tạo tự động. |
| a | Mở file ở chế độ ghi “append”. Tức là sẽ ghi vào cuối của nội dung đã có. | Nếu file không tồn tại, nó sẽ được tạo tự động. |
| r+ | Mở file cho phép cả đọc và ghi. | Nếu file không tồn tại, fopen() trả về NULL. |
| w+ | Mở file cho phép cả đọc và ghi. | Nếu file đã tồn tại, nội dung sẽ bị ghi đè. Nếu file không tồn tại, nó sẽ được tạo tự động. |
| a+ | Mở file cho phép đọc và ghi “append”. | Nếu file không tồn tại, nó sẽ được tạo tự động. |

Bảng 1: Bảng các tham số mode [2]

* Để làm việc với file văn bản, chúng ta sẽ sử dụng fprintf() và fscanf() lần lượt để ghi file và đọc file.

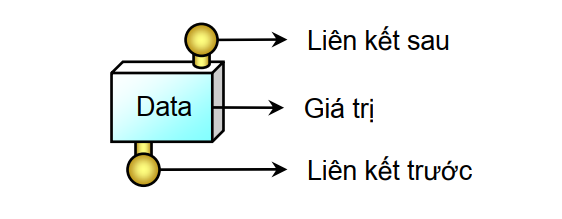
VD về ghi file: fprintf(file4,"We cannot solve this system of equations");

VD về đọc file: fscanf(file3,"%f",&HS[i][j]);

* Khi làm việc với tập tin hoàn tất, ta cần đóng file sau khi làm việc với nó xong. Việc đóng file đang mở có thể được thực hiện bằng cách dùng hàm fclose(). VD: fclose(file);

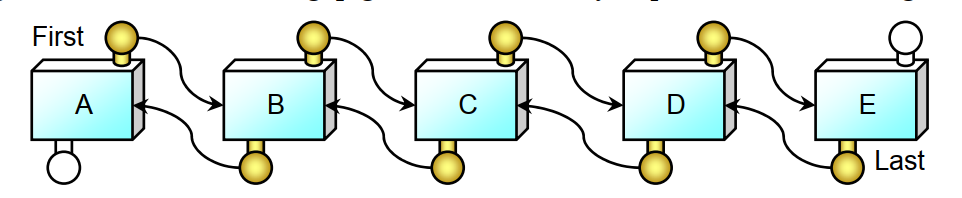
1. **Về danh sách liên kết kép [3]:**

* Định nghĩa: danh sách liên kết kép là danh sách mà mỗi phần tử trong danh sách có kết nối với phần tử đứng ngay trước và với phần tử đứng ngay sau nó.
* Danh sách liên kết kép gồm các nút được nối với nhau theo hai chiều. Hình 1 mô tả một nút trong danh sách liên kết kép. Mỗi nút là một bản ghi (record) gồm 3 trường:
* Trường thứ nhất chứa giá trị lưu trong nút đó.
* Trường thứ hai (Next) chứa liên kết (con trỏ) tới nút kế tiếp, tức là chứa một thông tin đủ để biết nút kế tiếp nút đó là nút nào, trong trường hợp là nút cuối cùng ( không có nút kế tiếp ), trường liên kết này được gán một giá trị đặc biệt.
* Trường thứ ba (Prev) chứa liên kết (con trỏ) tới nút liền trước, tức là chứa một thông tin đủ để biết nút đứng trước nút đo trong danh sách là nút nào, trong trường hợp là nút đầu tiên (không có nút liền trước), trường này được gán một giá trị đặc biệt.



Hình 1: Mô phỏng nút trong danh sách liên kết kép [3]

* Khác với danh sách nối đơn, danh sách nối kép có hai chốt: nút đầu tiên trong danh sách được gọi là First ( or Head ), nút cuối cùng trong danh sách được gọi là Last ( or Tail ). Để duyệt danh sách nối kép, ta có hai cách: hoặc bắt đầu từ First, dựa vào liên kết Next để đi sang nút kế tiếp, đến khi gặp giá trị đặc biệt (duyệt qua nút cuối) thì dừng lại; hoặc bắt đầu từ Last, dựa vào liên kết Prev để đi sang nút liền trước, đến khi gặp giá trị đặc biệt (duyệt qua nút đầu) thì dừng lại. Hình 2 mô phỏng sự liên kết giữa các nút trong danh sách liên kết kép.



Hình 2: Mô phỏng sự liên kết giữa các nút trong danh sách liên kết kép [3]

VD:

typedef struct Node

{

float data;

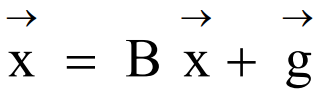
Node \*next; // trỏ đến phần tử đứng sau

Node \*previous; // trỏ đến phần tử đứng trước

} Node;

1. **Về phương pháp lặp đơn [1]:**

* Là một phương pháp tìm nghiệm gần đúng. Thông thường, ta cho ẩn số một giá trị ban đầu, từ giá trị này tính giá trị nghiệm gần đúng tốt hơn theo một quy tắc nào đó. Quá trình này được lặp lại nhiều lần và với một số điều kiện nhất định, ta nhận được nghiệm gần đúng.
* Biến đổi hệ phương trình về dạng:

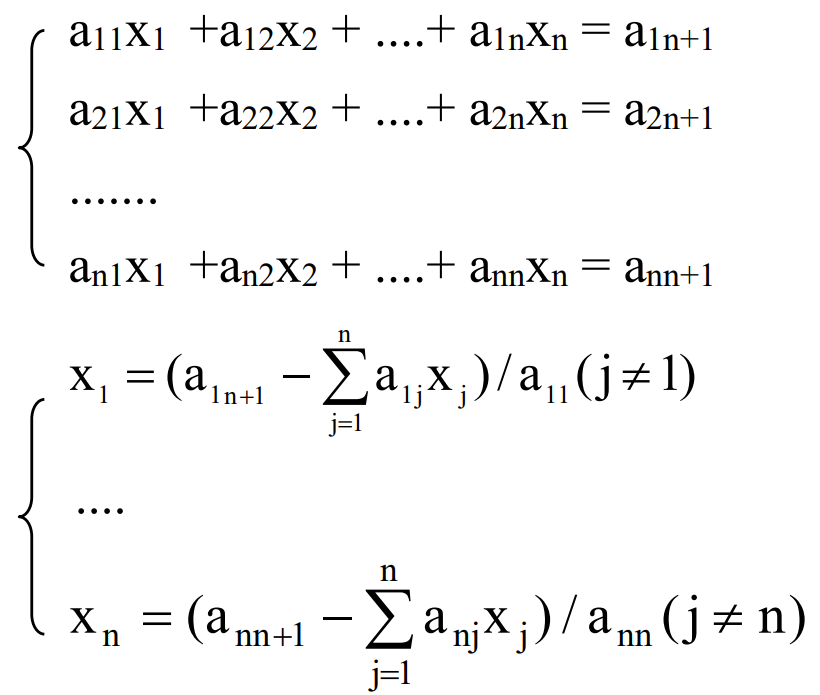


* Trong đó:

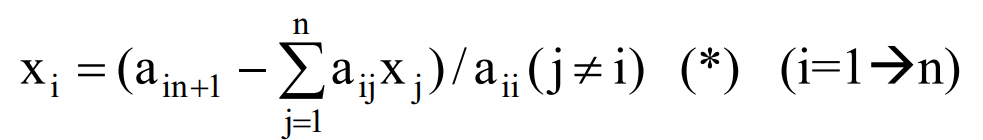
→ →

x = (x1,x2,…,xn);g = (g1,g2,…,gn);B={bij}n

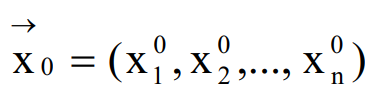
* Cách biến đổi:

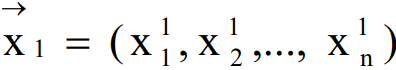


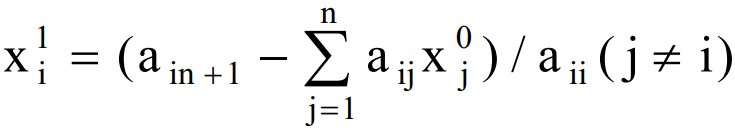
* Tổng quát:



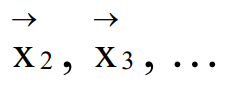
* Cho hệ phương trình xấp xỉ nghiệm ban đầu:



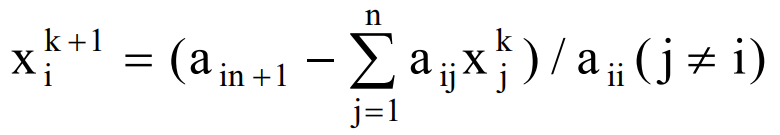
* Thay  vào (\*) để tính: 



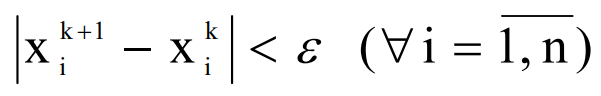
* Tương tự, tính:

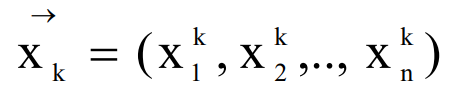


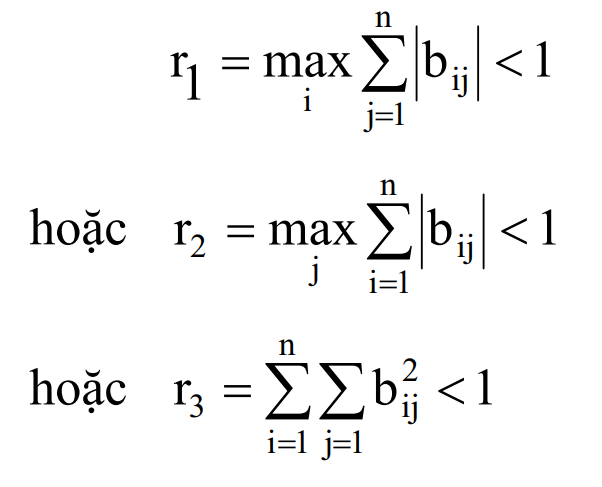
* Tổng quát:



* Quá trình lặp sẽ dừng khi thỏa mãn tiêu chuẩn hội tụ tuyệt đối:



* Khi đó­­ là nghiệm gần đúng của hệ phương trình.
* Điều kiện hội tụ: hệ phương trình có ma trận lặp B thỏa mãn:

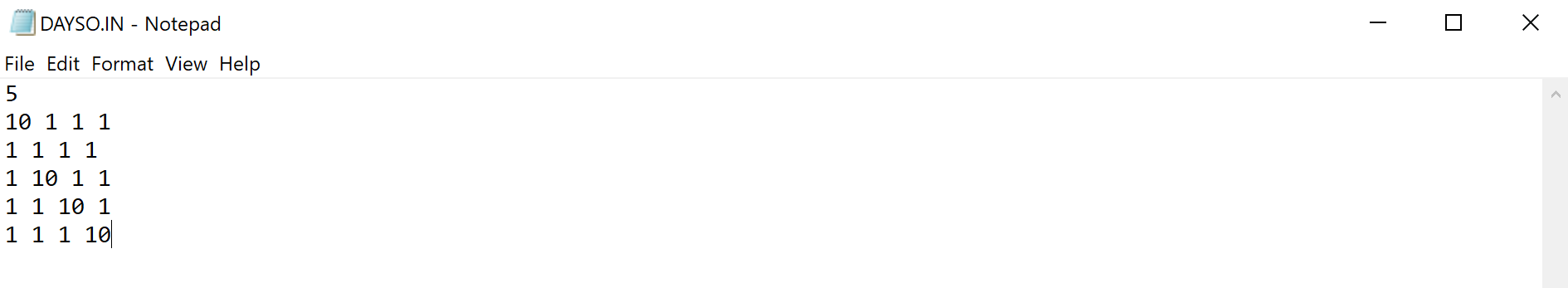


thì quá trình sẽ hội tụ đến nghiệm.

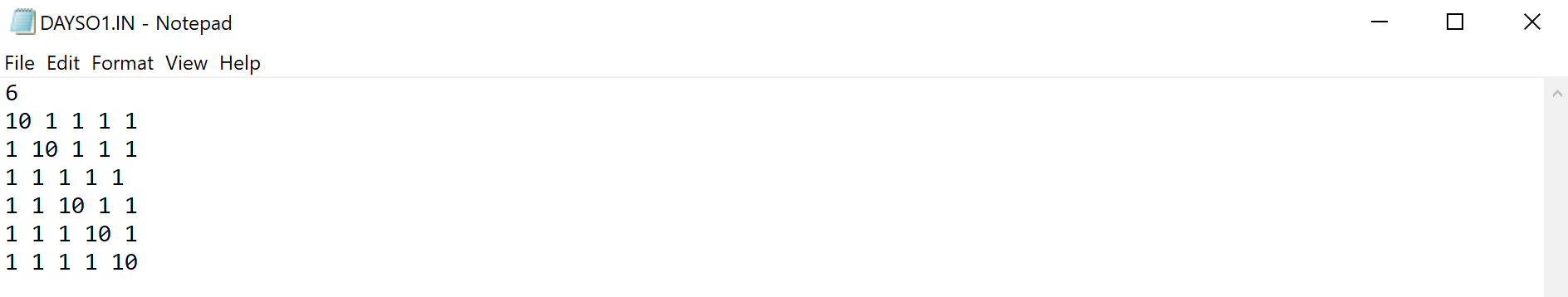
# TỔ CHỨC CẤU TRÚC DỮ LIỆU VÀ THUẬT TOÁN

## Phát biểu bài toán

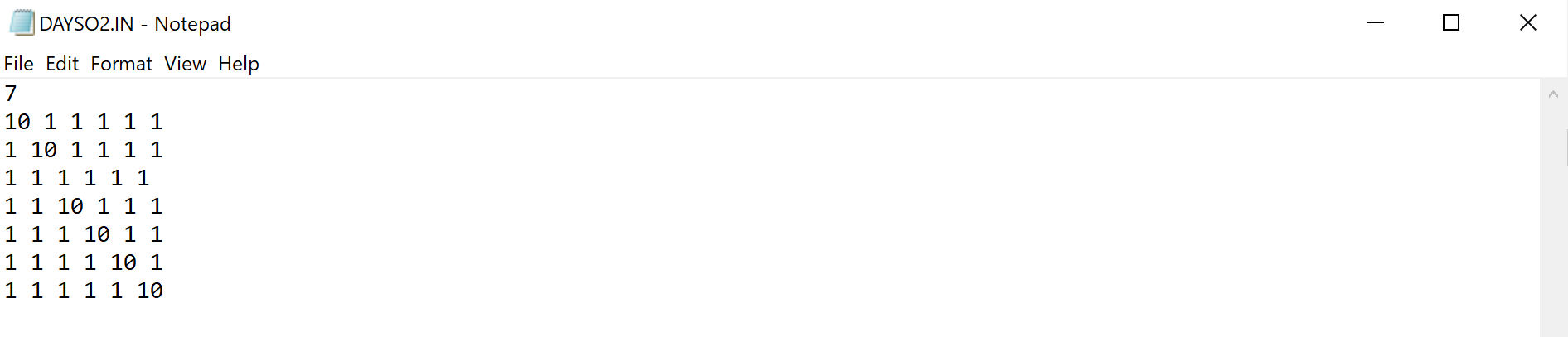
* Dữ liệu đầu vào ở đây bao gồm giá trị N và N hàng số, mỗi hàng số có N-1 số được lưu trong file DAYSO.IN. Hình 3, 4, 5 mô tả dữ liệu đầu vào.



Hình 3: Hình minh họa dữ liệu đầu vào (Input)

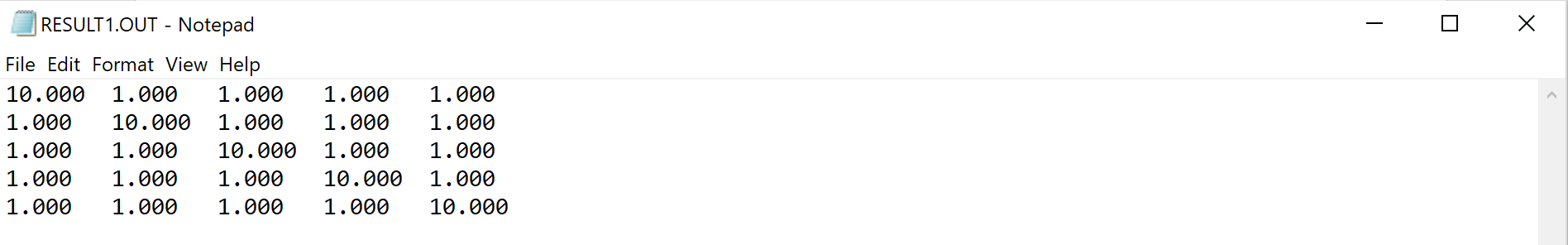


Hình 4: Hình minh họa dữ liệu đầu vào (2) (Input)



Hình 5: Hình minh họa dữ liệu đầu vào (3) (Input)

* Dữ liệu đầu ra sẽ đi vào 2 file là: RESULT1.OUT, RESULT2.OUT.
* Trong file RESULT1.OUT: lưu trữ N dãy số, mỗi dãy số gồm N số. Các dãy số này có được sau khi ta đã thêm vào trước vị trí thứ M/2 (làm tròn nếu M lẻ) của mỗi dãy số một phần tử số bất kì (do người dùng nhập). Hình 6 mô tả dữ liệu đầu ra.



Hình 6: Hình minh họa dữ liệu đầu ra trong file RESULT1.OUT (Output)

* Trong file RESULT2.OUT: lưu nghiệm của hệ phương trình trong trường hợp thỏa mãn điều kiện hội tụ. Trong trường hợp hệ phương trình bạn cho không thể giải được hoặc số lần lặp đến nghiệm vượt quá 100, file RESULT2.OUT sẽ in ra dòng chữ “We cannot solve this system of equations”.

## Cấu trúc dữ liệu

* Các con trỏ file, file2, file3, file4 dùng để mở file cho việc đọc và ghi.
* Các biến N, M để lưu trữ dữ liệu đọc từ file DAYSO.IN.
* Mảng a[] trong hàm void input( FILE \*file ) dùng để lưu các dãy số trong file DAYSO.IN từ đó tạo các các phần tử số ( các Node ).
* Biến position để lưu trữ vị trí thêm phần tử cho các dãy số.
* Mảng B[] trong hàm SolvingSystemOfEquations(FILE \*file3,FILE \*file4,int N) để lưu trữ dữ liệu của ma trận hệ số tự do.
* Mảng X[] trong hàm SolvingSystemOfEquations(FILE \*file3,FILE \*file4,int N) để lưu trữ các nghiệm xấp xỉ ban đầu cũng như nghiệm cuối cùng của hệ phương trình.
* Mảng XNext[] trong hàm SolvingSystemOfEquations(FILE \*file3,FILE \*file4,int N) để lưu trữ giá trị tạm, từ đó tính giá trị nghiệm xấp xỉ tiếp theo.
* Mảng HS[][] trong hàm SolvingSystemOfEquations(FILE \*file3,FILE \*file4,int N) để lưu trữ dữ liệu của ma trận hệ số và ma trận hệ số tự do.
* Biến count dùng để lưu trữ số lần lặp trong phương pháp lặp đơn ( giúp ích cho việc dừng chương trình khi hệ phương trình không thỏa mãn điều kiện hội tụ ).
* Mảng Y[] trong hàm SolvingSystemOfEquations(FILE \*file3,FILE \*file4,int N) để lưu trữ chính xác nghiệm được in ra.

## Thuật toán

* Thuật toán thêm phần tử vào danh sách liên kết kép:
* Bước 1: tính toán vị trí để thêm phần tử: position = ceil (M/2) – 1.
* Bước 2: dùng con trỏ đã lưu địa chỉ của hàng đầu tiên để duyệt qua các hàng. Trong lúc duyệt qua các hàng ta tạo ra con trỏ phụ gán bằng con trỏ lưu địa chỉ của phần tử số đầu tiên trong hàng để duyệt qua các phần tử số.
* Bước 3: tạo một biến đếm, cho vòng lặp và nếu biến đếm bằng đúng vị trí cần thêm phần tử thì :
  + Tạo một phần tử số x
  + Lấy giá trị người dùng nhập vào gán cho x🡪data
  + x🡪prev = tmp2🡪prev
  + x🡪prev🡪next = x
  + x🡪next = tmp2
  + tmp2🡪prev = x
* Thuật toán giải hệ phương trình bằng phương pháp lặp đơn: [1]
* Bước 1: Dựa vào giá trị n đã được đọc từ file, sao chép các hệ số ( bao gồm hệ số của hệ phương trình và các hệ số tự do ) vào mảng HSij ( i = 1 ­­🡪 n, j = 1 🡪 ­n+1 ).
* Bước 2: Tính toán nghiệm xấp xỉ ban đầu đưa vào mảng Xi ( i = 0 🡪 n-1 ).
* Bước 3: Lặp:
* loop = false /\* để thoát \*/ , count = 0 /\* đếm số lần lặp \*/
* lặp i = 1 🡪 n

{

s = 0

lặp j = 1 🡪 n

if (j != i ) s = s + HSij \* Xj-1

XNexti-1 = HSij – s

if(HSii != 0 ) XNexti-1 = XNexti-1/HSii

else thoát

if ( | Xi-1 – XNexti-1 | >= eps && count <100 ) loop = true

}

Xi-1 = XNexti-1 ( i = 1 ­­🡪 n )

Trong khi ( loop )

* Xuất nghiệm: Xi ( i = 0 ­­🡪 n-1 )
* Thuật toán lấy chính xác nghiệm được in ra để thử lại trên từng phương trình:
* Bước 1: khai báo mảng Yn ( n = 0 ­🡪 n-1 ).
* Bước 2: lặp i = 0 🡪 n-1
* Yi = làm tròn trên ( Xi \* 1000 + 0.5 )
* Yi = Yi/1000
* Nhìn chung thuật toán giải hệ phương trình bằng phương pháp lặp đơn tương đối phức tạp còn thuật toán thêm phần tử vào danh sách liên kết kép và thuật toán lấy chính xác nghiệm được in ra để thử lại trên từng phương trình thì tương đối đơn giản.

# CHƯƠNG TRÌNH VÀ KẾT QUẢ

## Tổ chức chương trình

* Chương trình bao gồm hàm main () và các hàm con thức hiện các chức năng riêng biệt. Các hàm con được khai báo nguyên mẫu hàm tiện cho việc quản lý và sử dụng.
* Ta khai báo các struct Node, row, để mỗi phần tử trong dãy số sẽ liên kết với nhau theo danh sách liên kết kép và để mỗi hàng số cũng được liên kết với nhau theo danh sách liên kết kép.
* Hầu hết chương trình tự động tính toán, người dùng chỉ việc nhập dữ liệu vào file DAYSO.IN.
* Tuy nhiên người dùng vẫn có thể nhập nghiệm xấp xỉ ban đầu theo ý người dùng. Gõ số 1 nếu người dùng muốn nhập bằng tay, gõ bất kì số nào nếu người dùng muốn chương trình tính toán một cách tự động.
* Sau khi sử dụng chương trình, người dùng có thể tiếp tục chương trình bằng các thao tác đơn giản. Đó là bấm chữ ‘y’ (yes) nếu muốn tiếp tục, chữ bất kì nếu muốn dừng lại.

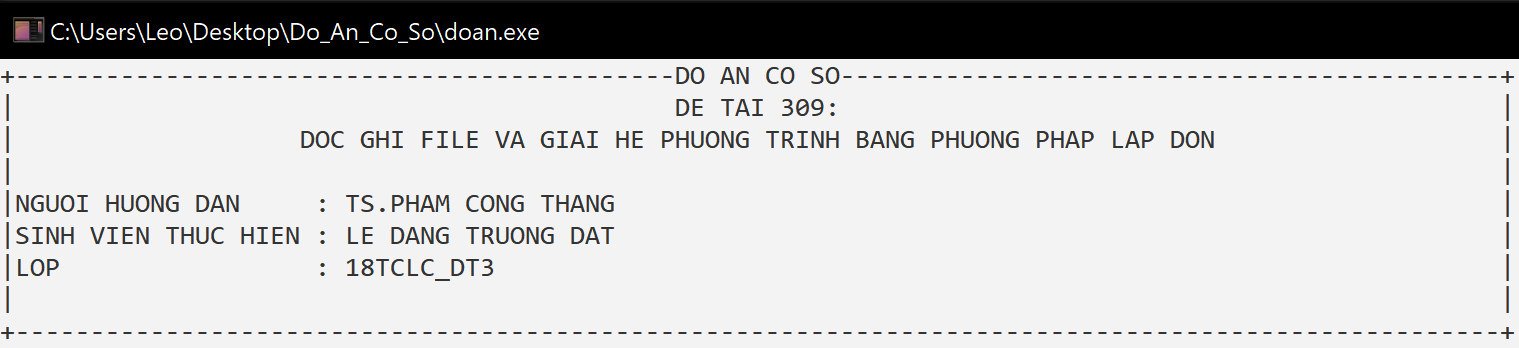
## Ngôn ngữ cài đặt

* Chương trình được viết bằng ngôn ngữ C trên ứng dụng devC++.

## Kết quả

### Giao diện chính của chương trình

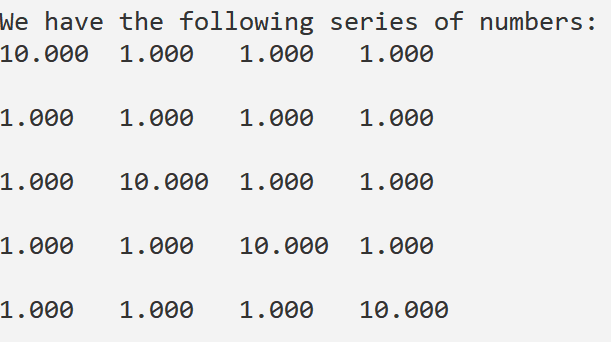
* Hình 7 mô tả giao diện chính của chương trình.



Hình 7: Giao diện chính của chương trình

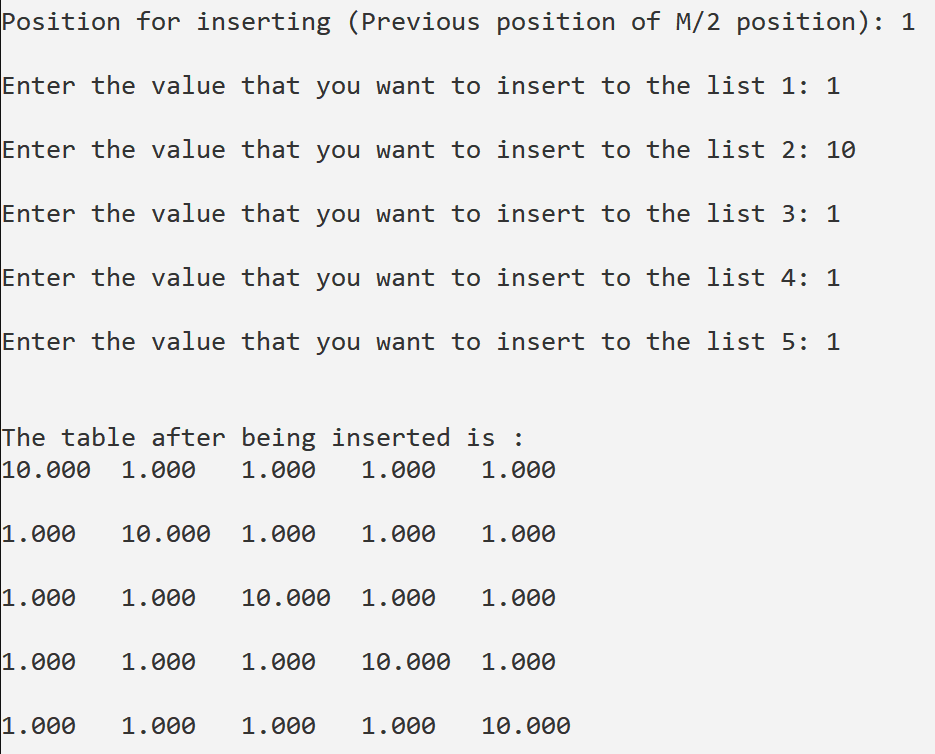
### Kết quả thực thi của chương trình

* Quá trình đi đến kết quả:
* Khi chương trình bắt đầu, chương trình sẽ đọc file DAYSO.IN, tạo danh sách liên kết và xuất ra màn hình Console như hình 8.

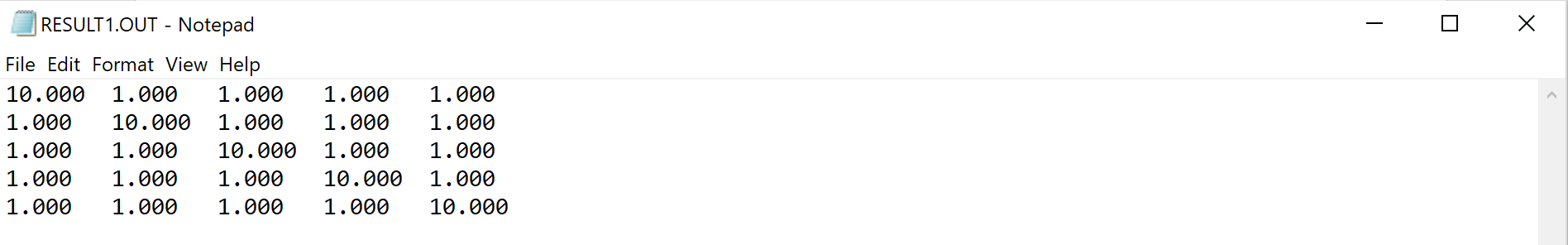


Hình 8: Dãy số được đọc từ file

* Tiếp theo, chương trình yêu cầu bạn nhập các số muốn chèn, và in ra màn hình Console cũng như file RESULT1.OUT các dãy số sau khi được chèn vào. ( Xem hình 9,10 ).

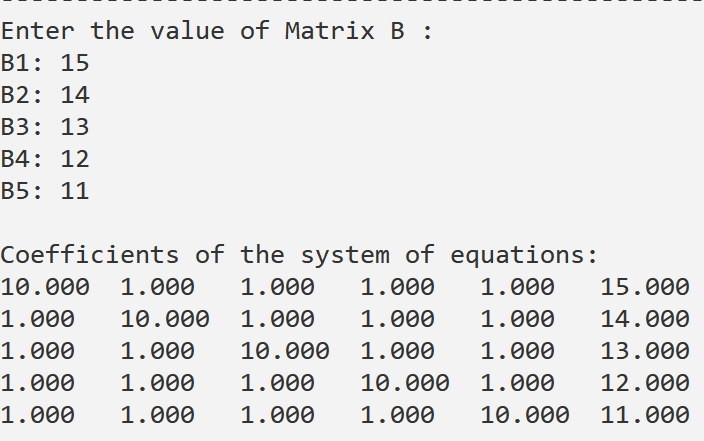


Hình 9: Nhập các số để chèn, các dãy số sau khi được chèn (Console)



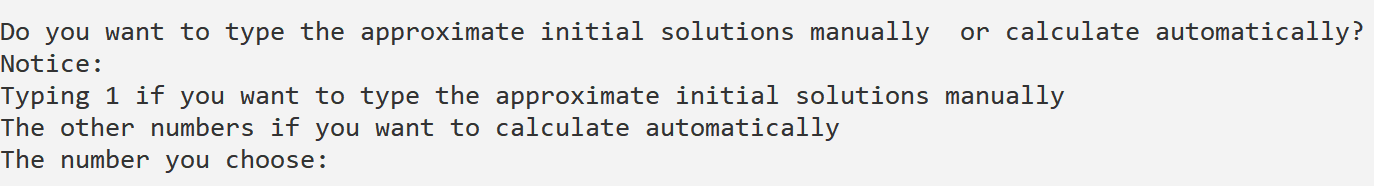
Hình 10: Kết quả trong file RESULT1.OUT sau thi chèn số vào các dãy số

* Tiếp theo, chương trình yêu cầu người dùng nhập ma trận B (ma trận hệ số tự do), sau đó chương trình sẽ tính toán và in ra toàn bộ các hệ số của hệ phương trình. ( Xem hình 11 )



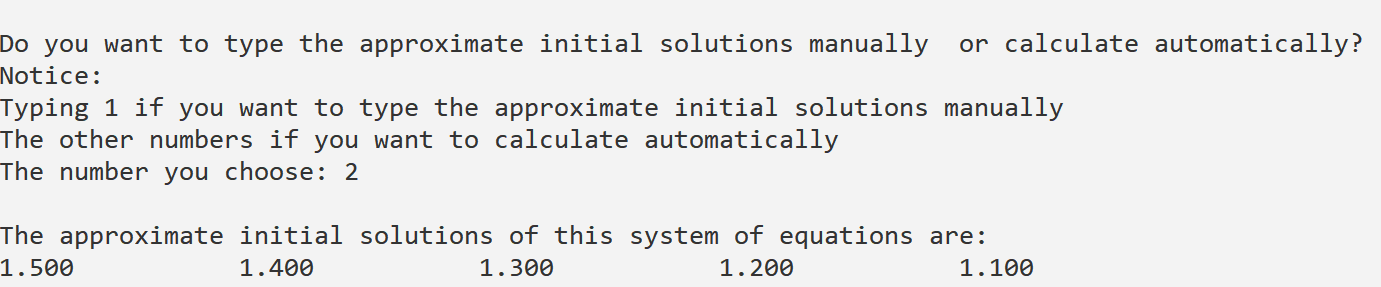
Hình 11: Ma trân B và hệ số của hệ phương trình

* Tiếp theo, chương trình sẽ hỏi bạn muốn tự nhập các giá trị nghiệm xấp xỉ ban đầu hay để chương trình tự động tính toán. ( Xem hình 12 )



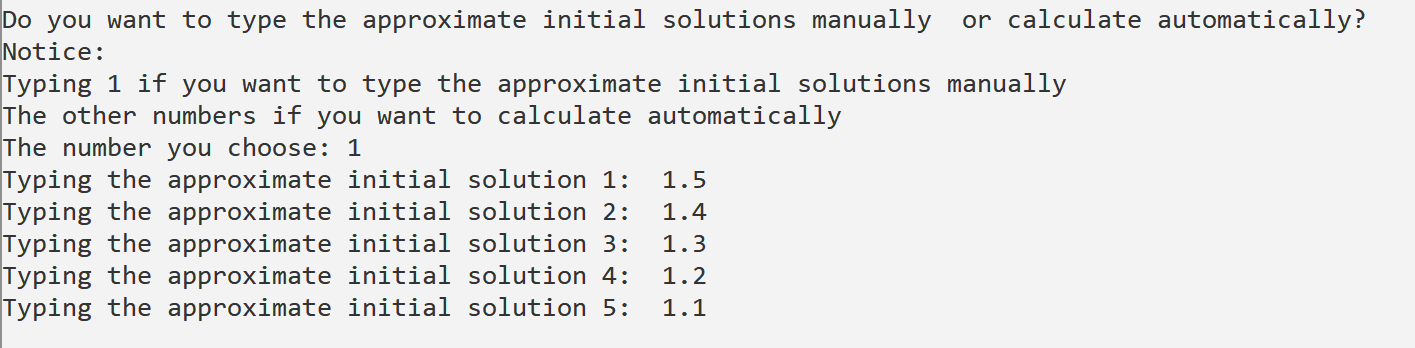
Hình 12: Mô tả cách thức thực hiện của chương trình với nghiệm ban đầu

* Nếu bạn muốn tính toán tự động: ( Xem hình 13 )



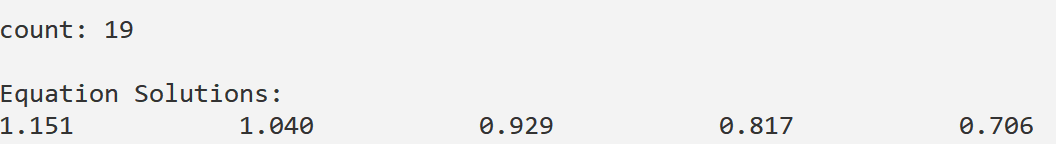
Hình 13: Chương trình tự động tính toán nghiệm xấp xỉ ban đầu

* Nếu bạn muốn tự nhập dữ liệu cho nghiệm xấp xĩ ban đầu: ( Xem hình 14 )

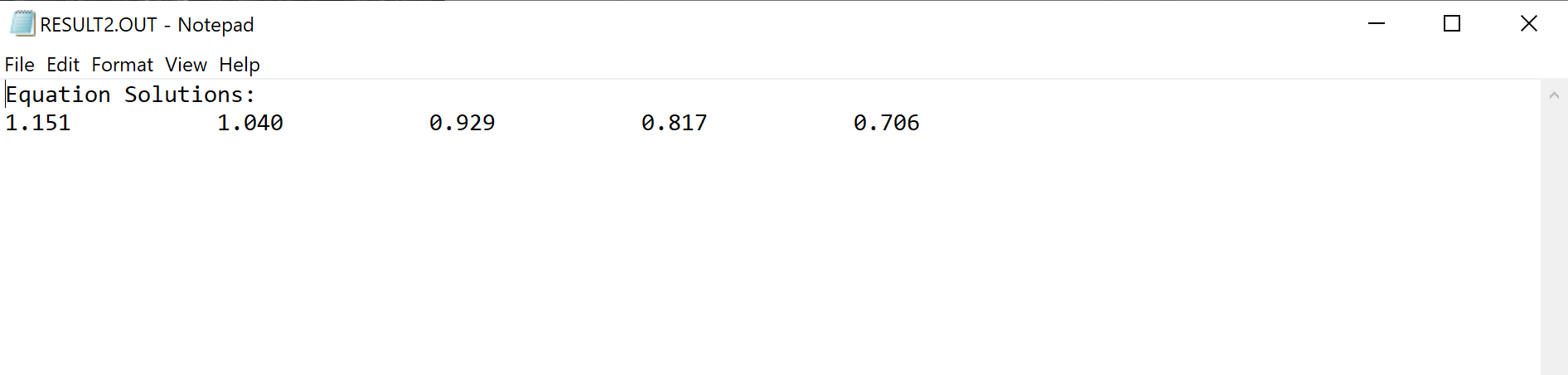


Hình 14: Mô phỏng cách bạn tự nhập nghiệm xấp xỉ ban đầu

* Kết quả của chương trình: ( Xem hình 15, 16 )

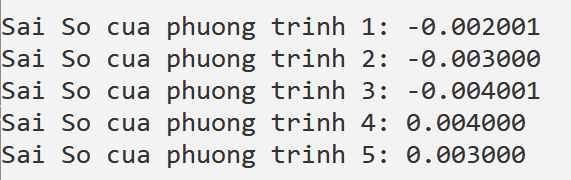


Hình 15: Kết quả trên màn hình Console



Hình 16: Kết quả trên file RESULT2.OUT

* Sau khi thực hiện chương trình, chương trình sẽ thay tất cả nghiệm tìm được vào mỗi phương trình để tìm sai số của mỗi phương trình.



Hình 17: Sai số của mỗi phương trình

* Tiếp theo, chương trình sẽ hỏi bạn có muốn tiếp tục chương trình không, bấm ‘y’ (yes) nếu muốn, bấm bất kì phím nào nếu không muốn.



Hình 18: Mô phỏng chương trình hỏi bạn “có muốn tiếp tục không”

### Nhận xét đánh giá

* Kết quả của chương trình chỉ có thể xấp xĩ với nghiệm chính xác (tuy nhiên sai số ở đây là rất nhỏ).
* Chương trình đã đáp ứng các yêu cầu đưa ra.
* Khả năng giải hệ phương trình còn hạn chế (không thể giải được một số hệ phương trình không thỏa mãn điều kiện hội tụ cũng như số lần tính toán lớn hơn 100).

# KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

## Kết luận

* Chương trình đã mô tả được kết quả theo yêu cầu. Đó là việc xuất nghiệm ra file RESULT2.OUT cũng như việc xuất các dãy số sau khi chèn thêm số vào các dãy số ra file RESULT1.OUT.
* Chương trình có khả năng giải được các hệ phương trình thỏa mãn điều kiện hội tụ.
* Chương trình đã đáp ứng được các yêu cầu của giảng viên hướng dẫn.
* Thông qua đồ án, em đã:
* Cải thiện được kĩ năng lập trình.
* Ứng dụng danh sách liên kết kép và phương pháp lặp đơn để giải quyết vấn đề trong bài toán.
* Rèn luyện kỹ năng chủ động tìm hiểu kiến thức mới như cách đọc ghi file trong C.
* Ôn lại các kiến thức đã được học.

## Hướng phát triển

* Tuy bài toán đã giải quyết được một số hệ phương trình thỏa điều kiện hội tụ, trong tương lai, em sẽ nghiên cứu phát triển nó theo nhiều hướng mới:
* Tăng khả năng giải quyết bài toán hệ phương trình: em sẽ nghiên cứu để mở rộng giới hạn của thuật toán này sao cho nó có thể giải được nhiều hệ phương trình hơn bao gồm những hệ phương trình không thỏa điều kiện hội tụ.
* Tìm cách tối ưu hóa thuật toán: em sẽ nghiên cứu để tìm ra những thuật toán hay hơn giúp cho việc thực hiện chương trình có thể nhanh hơn, nghiệm gần đúng chuẩn hơn.
* Nếu có cơ hội em sẽ phát triển ứng dụng này thành một ứng dụng đa ngôn ngữ, tức là người dùng có thể chọn ngôn ngữ mà họ muốn dùng (trong bài em đã sử dụng tiếng anh để ứng dụng có thể đáp ứng cho nhu cầu của nhiều người hơn thay vì sử dụng tiếng việt).
* Trong tương lai, em thấy ứng dụng này cần phải cải thiện nhiều hơn về giao diện giúp cho người dùng thích thú hơn với việc sử dụng ứng dụng.
* Kết hợp với các loại ứng dụng khác để tạo ra một ứng dụng đa chức năng (ví dụ như kết hợp với các ứng dụng giải phương trình bậc nhất, giải phương trình bậc 2,…).

TÀI LIỆU THAM KHẢO

[1] Đỗ Thị Tuyết Hoa, Bài giảng môn Phương pháp tính, 2007

[2] Nguyễn Văn Hiếu, Đọc ghi file trong C, <https://nguyenvanhieu.vn/doc-ghi-file-trong-c/> , ngày truy cập: 5/4/2020

[3] Lê Minh Hoàng, Giải thuật và lập trình, 1999 – 2002

PHỤ LỤC

Source Code:

// Tạo khung giới thiệu

void draw ()

{

for (int i=0;i<=90;i++){

switch (i)

{

case 0: printf("+");break;

case 45: printf("DO AN CO SO");break;

case 90: printf("+\n");break;

default: printf("-");

}

}

for (int i=0;i<=90;i++){

switch (i)

{

case 0: printf("|");break;

case 45: printf("DE TAI 309:");break;

case 90: printf("|\n");break;

default: printf(" ");

}

}

printf("| DOC GHI FILE VA GIAI HE PHUONG TRINH BANG PHUONG PHAP LAP DON |\n");

printf("| |\n");

printf("|NGUOI HUONG DAN : PHAM CONG THANG |\n");

printf("|SINH VIEN THUC HIEN : LE DANG TRUONG DAT |\n");

printf("|LOP : 18TCLC\_DT3 |\n");

printf("| |\n");

for (int i=0;i<=100;i++){

switch (i)

{

case 0: printf("+");break;

case 100: printf("+\n"); break;

default: printf("-");

}

}

}

// Đường phân chia

void devide()

{

for (int i=0;i<=100;i++){

if(i==0||i==100) printf("\n");

else printf("-");

}

}

// Kiểm tra sự tồn tại của File DAYSO.IN

void check (FILE \*file)

{

if (file == NULL)

{

printf ("We cannot open this file");

exit (0);

}

}

// Hàm tạo node

Node \*CreateNode (float x)

{

Node \*Np = new Node;

Np->data = x;

Np->next = NULL;

Np->prev = NULL;

return Np;

}

// Hàm tạo row ( trong các hàng sẽ tạo ra các node )

row \*CreateRow (float a [], int m)

{

int i = 0;

row \*Rp = new row;

Node \*Nhead = NULL;

Node \*Nlast = NULL;

Node \*Ncurrent = NULL;

while (i < m)

{

Node \*p = CreateNode(a[i]);

if (Nhead == NULL)

{

Nhead = p;

}

else

{

Ncurrent = Nhead;

while (Ncurrent->next != NULL)

Ncurrent = Ncurrent->next;

Ncurrent->next = p;

p->prev = Ncurrent;

}

i++;

}

Rp->data\_r = Nhead;

Rp->nextrow = NULL;

Rp->prevrow = NULL;

return Rp;

}

// Hàm đọc dữ liệu từ file DAYSO.IN và tạo row

void input (FILE \*file)

{

printf("We have the following series of numbers:\n");

// printf("%d %d\n", N, M);

// khai bao cac con tro thuoc danh sach lien ket hang

row \*Rhead = NULL;

row \*Rcurrent = NULL;

row \*Rlast = NULL;

while (!feof(file))

{

float a[M];

for (int j = 0; j < M; j++)

{

fscanf(file, "%f", &a[j]);

}

row \*r = CreateRow(a, M);

if (Rhead == NULL)

{

Rhead = r;

}

else

{

Rcurrent = Rhead;

while (Rcurrent->nextrow != NULL)

Rcurrent = Rcurrent->nextrow;

Rcurrent->nextrow = r;

r->prevrow = Rcurrent;

}

}

table = Rhead;

fclose(file);

}

// Hàm xuất ra Console

void outputForConsole()

{

row \*var\_run1 = table;

while (var\_run1 != NULL)

{

Node \*var\_run2 = var\_run1->data\_r;

while (var\_run2 != NULL)

{

printf("%.3f\t", var\_run2->data);

var\_run2 = var\_run2->next;

}

printf("\n\n");

var\_run1 = var\_run1->nextrow;

}

}

// Hàm để chèn thêm phần tử vào vị trí trước vị trí M/2 ( làm tròn nếu M lẻ )

void addBefore(int num)

{

int position = ceil(num/2.0) -1;

printf("Position for inserting (Previous position of M/2 position): %d \n",position);

row \*tmp1= table;

int varInPrint= 1;

while (tmp1 !=NULL){

Node \*tmp2 = tmp1->data\_r;

int dem=0;

while (tmp2 !=NULL){

if (dem == position) {

Node \*x = new Node;

printf("\nEnter the value that you want to insert to the list %d: " , varInPrint);

float ValueInserted;

scanf("%f",&ValueInserted);

x->data = ValueInserted;

x->prev = tmp2->prev;

x->prev->next = x;

x->next = tmp2;

tmp2->prev = x;

}

dem++;

tmp2 = tmp2->next;

}

tmp1 = tmp1->nextrow;

varInPrint++;

}

}

// Hàm xuất ra file và console

void outputForConsoleAndFile(FILE \*file2)

{

row \*var\_run1 = table;

while (var\_run1 != NULL)

{

Node \*var\_run2 = var\_run1->data\_r;

while (var\_run2 != NULL)

{

printf("%.3f\t", var\_run2->data);

fprintf(file2,"%.3f\t",var\_run2->data);

var\_run2 = var\_run2->next;

}

printf("\n\n");

fprintf(file2,"\n");

var\_run1 = var\_run1->nextrow;

}

fclose(file2);

}

// Hàm tìm nghiệm xấp xỉ ban đầu

void CalInitialSolution(float HS[MAX][MAX],float X[],int N)

{

for (int i=0; i<N;i++){

X[i]= HS[i+1][N+1]/HS[i+1][i+1];

}

printf("\nThe approximate initial solutions of this system of equations are:\n");

for (int i=0;i<N;i++){

printf("%.3f\t\t",X[i]);

}

}

// Hàm kiểm tra sai số kết quả

void checkSaiSo(float HS[MAX][MAX],float X[],int N){

for (int i=1;i<=N;i++){

float s=0;

float saiso;

for (int j=0;j<N;j++){

s= s + X[j]\*HS[i][j+1];

}

// printf("\n%f\n",HS[i][N+1]);

saiso = HS[i][N+1]-s;

printf("\nSai So cua phuong trinh %d: %f ",i,saiso);

}

}

// Hàm giải hệ phương trình bằng phương pháp lập đơn

void SolvingSystemOfEquations(FILE \*file3,FILE \*file4,int N)

{

float B[N];

float X[N];

float XNext[N];

printf("Enter the value of Matrix B : \n");

for (int i = 0; i < N ; i ++ ){

printf("B%d: ",i+1);

scanf("%f",&B[i]);

}

float HS[MAX][MAX];

for (int i=1;i<=N;i++){

for (int j=1;j<=N;j++) {

fscanf(file3,"%f",&HS[i][j]);

}

}

for (int i=1;i<=N;i++){

HS[i][N+1] = B[i-1];

}

printf("\nCoefficients of the system of equations:\n");

for (int i = 1;i<=N;i++){

for(int j=1;j<=N+1;j++){

printf("%.3f\t",HS[i][j]);

}

printf("\n");

}

devide();

int number;

char con;

while(1){

printf("\nDo you want to type the approximate initial solutions manually or calculate automatically?\n");

printf("Notice: ");

printf("\nTyping 1 if you want to type the approximate initial solutions manually");

printf("\nThe other numbers if you want to calculate automatically");

printf("\nThe number you choose: ");

scanf("%d",&number);

switch(number){

case 1: {

for (int i =0 ; i<N;i++){

printf("Typing the approximate initial solution %d: ",i+1);

scanf("%f",&X[i]);

}

break;

}

default: CalInitialSolution(HS,X,N);

}

devide();

bool loop; // de chay vong lap

int count=0;// dem so vong lap

do {

loop=false;count++;

// vong lap tinh x1,x2,x3,...

for(int i=1;i<=N;i++){

float s=0;

for(int j=1;j<=N;j++) if(j!=i) s+= HS[i][j]\*X[j-1];

XNext[i-1] = HS[i][N+1] - s;

// tinh truoc s roi moi chia cho HS[i][i];

if(HS[i][i]!=0) XNext[i-1]=XNext[i-1]/HS[i][i];

else exit(0);

if(fabs(X[i-1]-XNext[i-1])>eps && count < 100) loop = true;

for(int i=1;i<=N;i++) X[i-1] = XNext[i-1];

}

} while(loop);

// xuat ket qua va in file "RESULT2.OUT"

printf("\ncount: %d\n",count);

if(count<100){

printf("\nEquation Solutions:\n");

fprintf(file4,"Equation Solutions:\n");

for (int i=0;i<N;i++){

printf("%.3f\t\t",X[i]);

fprintf(file4,"%.3f\t\t",X[i]);

}

}

else {

printf("\nWe cannot solve this system of equations");

fprintf(file4,"We cannot solve this system of equations");

}

devide();

// lay chinh xac nghiem duoc in ra;

float Y[N];

for (int i=0;i<N;i++){

Y[i] = floor(X[i]\*1000+0.5);

Y[i]/= 1000;

}

checkSaiSo(HS,Y,N);

devide();

printf("\nDo you want to continue (Y/N) ?\n");

con = getch();

if(con!='y') break;

devide();

}

fclose(file3);

fclose(file4);

}

// Hàm main ()

int main()

{

draw();

// cau a:

// con tro tro den file DAYSO.IN.

FILE \*file;

file = fopen("DAYSO.IN","r");

// kiem tra xem file da co chua, neu chua xuat ra man hinh dong chu "we cannot open this file".

check(file);

// doc N tu file DAYSO.in

fscanf(file, "%d", &N);

M=N-1;

//printf("%d\t%d\n",N,M);

input(file);

outputForConsole();

devide();

// cau b:

FILE \*file2;

file2 = fopen("RESULT1.OUT","w");

addBefore(M);

printf("\n\nThe table after being inserted is : \n");

outputForConsoleAndFile(file2);

devide();

// cau c:

FILE \*file3,\*file4;

file3 = fopen("RESULT1.OUT","r");

file4 = fopen("RESULT2.OUT","w");

SolvingSystemOfEquations(file3,file4,N);

}

// Khai báo các struct và các biến toàn cục

// struct node

typedef struct Node

{

float data;

Node \*next;

Node \*prev;

}Node;

// struct row

typedef struct row

{

Node \*data\_r;

row \*nextrow;

row \*prevrow;

} row;

// khai bao 2 bien toan cuc N va M.

int N,M;

row \*table = NULL;

// Khai báo thư viện và một số hằng số

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

#include <conio.h>

#define eps 1e-3

#define MAX 10

#define NFM 1000

// Khai báo nguyên mẫu hàm

void draw();

void devide();

void check(FILE \*file);

Node \*CreateNode(float x);

row \*CreateRow(float a[], int m);

void input(FILE \*file);

void outputForConsole();

void addBefore(int num);

void outputForConsoleAndFile(FILE \*file2);

void CalInitialSolution(float HS[MAX][MAX],float X[],int N);

void checkSaiSo(float HS[MAX][MAX],float X[],int N);

void SolvingSystemOfEquations(FILE \*file3,FILE \*file4,int N);