2021 2학기
C프로그래밍 (CSE2035) Project

Sungwon Jung, Ph.D.

Dept. of Computer Science and Engineering

Sogang University

Seoul, Korea

Tel: +82-2-705-8930

Email : jungsung@sogang.ac.kr

Outline

- 문제 정의
  - ✓ 설계 목표 및 제한 조건
- 문제 소개
- 관련 지식
  - ✓ $n \times n$ 행렬의 역행렬 구하는 방법
- 문제 설계
- 함수 명세
- 평가 기준 및 제출 마감

문제 정의 – 설계 목표 및 제한 조건

설계 목표

- ✓ 주어진 $n \times n$ 행렬의 역행렬을 구하는 프로그램 제작

제한 조건

- ✓ 구현 환경 : Linux (cspro 서버) 기준
- ✓ Pointer를 사용하여 주어진 문제를 해결
- ✓ <stdio.h>, <stdlib.h> library 만 사용 가능
- ✓ 정적 배열을 선언할 수 없음
- ✓ 전역 변수 사용 불가능
- ✓ 배열의 원소에 접근할 때는 반드시 포인터 연산자를 사용

문제 소개

- Input : $n \times n$ 행렬 $(n \leq 5)$

- 주어진 행렬의 역행렬 존재 여부를 판단.
  역행렬이 있으면 invertible(또는 non-singular), 없으면 non-invertible(또는 singular)라 한다.

- 행렬식(determinant)은 정사각행렬에 수를 대응시키는 함수이다.
  행렬 A의 determinant는 det(A)로 나타낸다.
  det(A)=0 이면 주어진 행렬은 singular이고, det(A)≠0 이면 non-singular이다. singular일 경우에는 역행렬이 존재하지 않는다 따라서 non-singular인 행렬에 대해서만 역행렬을 계산한다.

Example:

$A = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$일 때, $det(A) = 1 \times 4 - 2 \times 3 = -2 \neq 0$이므로 A는 non-singular이다.

따라서 역행렬이 존재한다.

관련 지식(1)

일반적인 n×n 행렬 A에 대하여 det(A)를 구하는 법은 다음과 같다.

[Theorem 1]

$A = \begin{pmatrix} a_{11} & a_{12} \cdots & a_{1n} \\ a_{21} & a_{22} \cdots & a_{2n} \\ \vdots & \vdots \ddots & \vdots \\ a_{n1} & a_{n2} \cdots & a_{nn} \end{pmatrix}$ 때,

$det(A)$ 는 임의의 i에 대하여 $det(A) = \sum_{j=1}^{n} (-1)^{i+j} \times a_{ij} \times det(M_{ij})$이다.

또한, $C_{ij} = (-1)^{i+j} \times det(M_{ij})$를 Cofactor of entry a_{ij}라 한다.

여기서 $M_{1j} = \begin{pmatrix} a_{11} & a_{12} \cdots & a_{1j} \cdots & a_{1n} \\ a_{21} & a_{22} \cdots & a_{2j} \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} \cdots & a_{nj} \cdots & a_{nn} \end{pmatrix}$, 1행과 j열을 제외한 (n-1)×(n-1) 부분행렬을 나타낸다.

관련 지식(2)

Example1:

$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$일 때, i를 1이라 하면

$$det(A) = \sum_{j=1}^{3}(-1)^{1+j} \times a_{1j} \times det(M_{1j})$$

$$= \underbrace{(-1)^{1+1} \times a_{11} \times det(M_{11})}_{c_{11}} + \underbrace{(-1)^{1+2} \times a_{12} \times det(M_{12})}_{c_{12}} + \underbrace{(-1)^{1+3} \times a_{13} \times det(M_{13})}_{c_{13}}$$

$$= 1 \times det\begin{pmatrix} 5 & 6 \\ 8 & 9 \end{pmatrix} + (-1) \times 2 \times det\begin{pmatrix} 4 & 6 \\ 7 & 9 \end{pmatrix} + 3 \times det\begin{pmatrix} 4 & 5 \\ 7 & 8 \end{pmatrix}$$

$$= 1 \times (5 \times 9 - 6 \times 8) + (-1) \times 2 \times (4 \times 9 - 6 \times 7) + 3 \times (4 \times 8 - 5 \times 7)$$

$$= (-3) + 12 - 9 = 0$$

$\therefore$ **A is singular, because of** $\det(A) = 0$

[Note] 나머지도 구해보면 $C_{21} = (-1)^{2+1} \times det(M_{21}) = -6$, $C_{22} = -12$, $C_{23} = 6$, $C_{31} = -3$, $C_{32} = -6$, $C_{33} = -3$이다.

이번 프로젝트에서는 $det(A) = \sum_{j=1}^{n}(-1)^{1+j} \times a_{1j} \times det(M_{1j})$ 라는 전제하에 진행한다. 따라서 n×n 행렬에 대해서 $c_{11} + c_{12} + \cdots + c_{1n}$ 을 계산한 다음 $det(A) = 0$이면 프로그램을 종료하고, $det(A) \neq 0$ 이면 역행렬을 계산한다.

관련 지식(3)

[Theorem 2]

A를 $n \times n$ 행렬, C_{ij}를 Cofactor of entry a_{ij}, $det(A) \neq 0$ 이라고 가정하면

$C = \begin{pmatrix} C_{11} & \cdots & C_{1n} \\ \vdots & \ddots & \vdots \\ C_{n1} & \cdots & C_{nn} \end{pmatrix}$**를 Cofactor Matrix라 하자. 그러면** $A^{-1} = \frac{1}{det(A)} C^T$**이다.**

C^T 는 전치 행렬($[\mathbf{A}^{\mathrm{T}}]_{ij} = [\mathbf{A}]_{ji}$)이다.

Example2: 3×3 Matrix에 대해서 다음과 같은 공식이 성립한다.

$$\mathbf{A}^{-1} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}^{-1} = \frac{1}{\det(\mathbf{A})} \begin{bmatrix} A & B & C \\ D & E & F \\ G & H & I \end{bmatrix}^{T} = \frac{1}{\det(\mathbf{A})} \begin{bmatrix} A & D & G \\ B & E & H \\ C & F & I \end{bmatrix}$$

여기서 $\det(\mathbf{A}) = a(ei - fh) - b(id - fg) + c(dh - eg)$ 이고,

$$\begin{array}{lll} A = (ei - fh) & D = -(bi - ch) & G = (bf - ce) \\ B = -(di - fg) & E = (ai - cg) & H = -(af - cd) \\ C = (dh - eg) & F = -(ah - bg) & I = (ae - bd) \end{array}$$

이다.

문제 설계(1) – 기본 구현 & 추가 구현

- 구현
  - ✓ **matrix.txt 파일을 불러와 이차원배열에 저장하는 함수를 작성한다. matrix.txt 파일에 첫번째 줄에는 행렬의 크기를 나타내는 N이, 두번째 줄 부터는 행렬의 원소에 해당하는 값이 들어있다.**
  - ✓ **$n \times n$ 행렬이 주어졌을 때 행렬의 determinant를 계산하여 0이면 프로그램을 종료하고 0이 아니라면 inverse matrix를 구하는 프로그램을 작성한다. (관련 지식 1, 2, 3 참고)**

문제 설계(2) – 프로그램 흐름

- 프로그램의 기본 흐름 (1)

  - ✓ 파일 입출력을 통해 **matrix.txt** 를 불러와 첫번째 줄에 들어있는 값이 **0**보다 작다면 프로그램을 종료한다.

  - ✓ 만약 **0**보다 크다면 **matrix.txt** 파일의 두번째 줄 부터 저장된 값들을 순서대로 입력 받아 이차원배열에 저장한다.

  - ✓ 프로젝트 진행 과정에서 배열을 선언할 때 반드시 동적 할당을 통해 선언하여야 하고, 사용이 끝난 후엔 반드시 할당된 메모리를 해제해주여야 한다.

문제 설계(2) – 프로그램 흐름

- 프로그램의 기본 흐름 (2)

  - ✓ 이번 프로젝트에서는 $n \leq 5$ 인 경우로만 가정하고 프로젝트를 진행한다.
  - ✓ 이차원배열에 저장된 행렬의 **determinant**를 c_{1j}를 이용하여 계산하고 **singular** 하다면 프로그램을 종료하고, **non-singular** 하다면 앞서 구한 c_{1j}에 이어서 $c_{2j}, \cdots, c_{nj}$ 를 구하여 **cofactor matrix**를 완성한다.
  - ✓ **cofactor matrix**의 전치 행렬을 구하는 함수를 작성한다.
  - ✓ 앞서 구한 **determinant**와 **cofactor matrix**의 전치 행렬을 사용해 **inverse matrix**를 계산한다.
  - ✓ 최종적으로 주어진 **matrix**와 **inverse matrix**의 곱셈을 출력하여 역행렬을 정확하게 계산하였는지를 확인하는 함수를 작성한다.

함수 명세

함 수 이 름	역 할
`double detA(double** A, int n)`	$n \times n$ matrix A(n < 6)의 det (A) 값을 return 한다.
`double** cofactorMatrix(double** A, int n)`	$n \times n$ matrix A cofactor matrix를 return해준다.
`double** transposeMatrix(double** A, int n)`	$n \times n$ matrix A 전치행렬을 return해준다.
`double** inverseMatrix(double** A, double det, int n)`	$n \times n$ matrix A의 inverse matrix를 return해준다.
`double** inverseCheckMatrix(double** A, double** B, int n)`	$n \times n$ matrix A와 inverse matrix B를 순서대로 곱한 Matrix C를 return해준다. 이때, C가 항등 행렬이 아니라면 역행렬의 계산이 잘못되었음을 알 수 있다.
`void printMatrix(double** A, int n)`	$n \times n$ Matrix A를 출력해준다.

명시된 함수는 반드시 구현해야 하고, 이외의 함수를 추가로 작성하여 사용할 수 있음

추가된 함수는 반드시 주석을 달아 상세하게 설명하고, 보고서에도 상세하게 설명해야 한다.

평가 기준 및 제출 마감(1)

- 설계 보고서(30점), 소스 코드(60점; 기본 구현 50점, 입·출력 10점), 주석(5점), 형식(5점), **Late시 1일 당 -10점(최대 -50점)**

- 프로그램 작성
  - ✓ 프로그램 구동 불가시 0점
  - ✓ 테스트 도중 **Segmentation Fault가 발생할 경우** 무조건 0점
  - ✓ 기본 기능 미구현시 감점
  - ✓ 전역 변수 사용 금지 , recursion 함수는 사용을 금지한다.
  - ✓ <stdio.h>, <stdlib.h> 이외의 library는 사용을 금지한다.
  - ✓ n의 값은 정수, matrix의 각 원소는 -100<x< 100의 실수 값을 가진다고 가정한다.
  - ✓ 출력 시, 7칸에 맞춰 오른쪽 정렬, 소수점 셋째 자리에서 반올림하여 출력한다.
  - ✓ 모든 계산은 포인터 연산으로만 해결한다. 즉, A[i][j]와 같은 표현 사용시 함수 당 감점.
  - ✓ 자신의 코드를 다른 사람이 알기 쉽게 주석을 자세히 적는다.
  - ✓ COPY 발견 시, 이유를 불문하고 두 사람 모두 0점 처리
  - ✓ Warning 발생 시, Warning 한 건당 -1점

평가 기준 및 제출 마감(2)

- **제출 방법**

proj1_학번 이름의 폴더를 만들고, 이 폴더에 소스파일, Makefile, 설계보고서를 넣어서 폴더를 tar.gz로 압축하여 제출. (Appendix A 참고) a.out등 실행 파일은 모두 제거한 뒤 압축하여 보낼 것. 실행 파일이 포함되면 gmail로 부터 반송될 수 있음.

- **제출 형식**
  - ✓ **main()함수가 있는 파일 : proj1_학번.c (proj1_20210000.c)**
  - ✓ **압축파일 : proj1_학번.tar.gz (proj1_20210000.tar.gz)**
  - ✓ **과제 게시판에 업로드**

- **제출일 : 2021/11/21(일요일) 23:59:59 까지**

  Late - 2021/11/26(금) 23:59:59까지 (하루당 -10점)

- **기타 문의 : hyungsseop@u.sogang.ac.kr**

Appendix A – tar.gz

tar.gz 파일 압축 풀기

```
gr220200026@cspro1:~/proj1_2021$ ls
proj1_20210000  proj1_20210000.tar.gz
gr220200026@cspro1:~/proj1_2021$ tar -xvzf proj1_20210000.tar.gz
proj1_20210000/
proj1_20210000/makefile
proj1_20210000/20212000.h
proj1_20210000/20210000.c
gr220200026@cspro1:~/proj1_2021$ ls
proj1_20210000  proj1_20210000.tar.gz
gr220200026@cspro1:~/proj1_2021$ cd proj1_20210000
gr220200026@cspro1:~/proj1_2021/proj1_20210000$ ls
20210000.c  20212000.h  makefile
gr220200026@cspro1:~/proj1_2021/proj1_20210000$
```

tar.gz 압축 하기(폴더)

소스 파일이 포함된 directory

```
gr220200026@cspro1:~/proj1_2021$ ls
proj1_20210000
gr220200026@cspro1:~/proj1_2021$ tar -cvzf proj1_20210000.tar.gz proj1_20210000/
proj1_20210000/
proj1_20210000/makefile
proj1_20210000/20212000.h
proj1_20210000/20210000.c
gr220200026@cspro1:~/proj1_2021$ ls
proj1_20210000  proj1_20210000.tar.gz
gr220200026@cspro1:~/proj1_2021$
```

압축 파일명

Appendix B - Makefile

```
gr220200026@cspro1:~/proj1_2021/proj1_20210000$ ls
20212000.c  20212000.h  makefile
gr220200026@cspro1:~/proj1_2021/proj1_20210000$ make
src > [93m20212000.c ...[0m
[96m[1mRelease build start...[0m
[96m[1mcomplete build...[0m
[96m[1mOutput file -> [92mtest[0m

gr220200026@cspro1:~/proj1_2021/proj1_20210000$ ls
20212000.c  20212000.h  20212000.o  makefile  test
gr220200026@cspro1:~/proj1_2021/proj1_20210000$ make clean
[91m Delete object files...[0m
gr220200026@cspro1:~/proj1_2021/proj1_20210000$ ls
20212000.c  20212000.h  makefile
gr220200026@cspro1:~/proj1_2021/proj1_20210000$
```

make를 입력하면 자동으로 Makefile의 내용을 바탕으로 컴파일 한다

Makefile 내의 [clean :] 규칙을 실행한다. 자세한 것은 첨부된 Makefile 참조

- ✓ 제출할 때에는 반드시 make clean 상태로 제출한다.