**Lab 07**

|  |
| --- |
| 그림입니다. 원본 그림의 이름: YU_UI_RGB-10.png 원본 그림의 크기: 가로 2256pixel, 세로 3047pixel 프로그램 이름 : Adobe ImageReady |

|  |  |
| --- | --- |
| 과목명 | 프로그래밍 언어 |
| 교수님 | 김영탁 교수님 |
| 이 름 | 김주환 |
| 학 번 | 21812158 |
| 일 자 | 2021.04.15.목 |

**행렬 연산을 위한 2차원 배열의 동적 생성, 파일 입출력**

**1. 행렬 연산을 위한 2차원 배열의 동적 생성, 주소 확인 및 삭제**

1) 지정된 크기의 2차원 double 자료형 배열을 동적으로 생성하는 함수를 작성하라

double\*\* create\_double\_mtrx(int row\_size, int col\_size) {

double\*\* ppdm = NULL;

ppdm = (double\*\*)calloc(row\_size, sizeof(double\*));

for (int i = 0; i < row\_size; i++) {

ppdm[i] = (double\*)calloc(col\_size, sizeof(double));

}

return ppdm;

}

void check\_address\_2dim\_array\_for\_matrix() {

int m[3][3] = {

10, 20, 30, 40, 50, 60, 70, 80, 90

};

printf("m = %p\n", m);

printf("m[0] = %p\n", m[0]);

printf("m[1] = %p\n", m[1]);

printf("m[2] = %p\n", m[2]);

printf("&m[0][0] = %p\n", &m[0][0]);

printf("&m[1][0] = %p\n", &m[1][0]);

printf("&m[2][0] = %p\n", &m[2][0]);

printf("\n");

}

2) 2차원 배열의 이름, 첫번째 행 주소, 첫번째 원소 주소를 출력하는 함수를 작성하라

void delete\_double\_mtrx(double\*\* dm, int row\_size, int col\_size) {

for (int i = 0; i < row\_size; i++) free(dm[i]);

free(dm);

}

3) 지정된 크기의 2차원 double 자료형 배열을 삭제하는 함수를 작성하라

**2. 행렬 연산을 위한 double 자료형의 파일 입력**

1) 지정된 크기의 2차원 double 자료형 배열을 동적으로 생성하고, 입력 데이터 파일로부터 지정된 크기만큼의 데이터를 fscanf()함수를 사용하여 읽고, 동적으로 생성한 2차원 배열에 저장하는 기능을 구현하라.

double\*\* fget\_double\_mtrx(FILE\* fin, double\*\* dm, int row\_size, int col\_size) {

double d = 0.0;

dm = create\_double\_mtrx(row\_size, col\_size);

#ifdef AM

row\_size--;

col\_size--;

#endif

for (int i = 0; i < row\_size; i++) {

for (int j = 0; j < col\_size; j++) {

if (fscanf(fin, "%lf", &d) != EOF) dm[i][j] = d;

}

}

return dm;

}

**3. 행렬의 화면/파일 출력**

1) 행렬을 화면에 출력하는 함수를 작성하라. 각 값은 최소 8자리, 소수점 이하 2자리, 오른쪽 정렬, 대괄호로 감싸기 위해 확장 완성형 코드를 사용

void print\_mtrx(double\*\* dm, int row\_size, int col\_size) {

unsigned char a6 = 0xA6, a1 = 0xA1, a2 = 0xA2;

unsigned char a3 = 0xA3, a4 = 0xA4, a5 = 0xA5;

for (int i = 0; i < row\_size; i++) {

for (int j = 0; j < col\_size; j++) {

if ((i == 0) && (j == 0)) {

printf("%c%c%7.2lf", a6, a3, dm[i][j]);

}

else if ((i == 0) && j == (col\_size - 1)) {

printf("%7.2lf%c%c", dm[i][j], a6, a4);

}

else if ((i > 0) && (i < row\_size - 1) && (j == 0)) {

printf("%c%c%7.2lf", a6, a2, dm[i][j]);

}

else if ((i > 0) && (i < row\_size - 1) && (j == (col\_size - 1))) {

printf("%7.2lf%c%c", dm[i][j], a6, a2);

}

else if ((i == row\_size - 1) && (j == 0)) {

printf("%c%c%7.2lf", a6, a6, dm[i][j]);

}

else if ((i == row\_size - 1) && (j == (col\_size - 1))) {

printf("%7.2lf%c%c", dm[i][j], a6, a5);

}

else {

printf("%7.2lf", dm[i][j]);

}

}

printf("\n");

}

}

void fprint\_mtrx(FILE\* fout, int cal, double\*\* dm, int row\_size, int col\_size) {

unsigned char a6 = 0xA6, a1 = 0xA1, a2 = 0xA2;

unsigned char a3 = 0xA3, a4 = 0xA4, a5 = 0xA5;

if (cal == 0) {

fout = fopen(MTRXNXNINPUTDATA, "w");

if (fout == NULL) {

printf("Error");

return;

}

}

else if (cal == 1) {

fout = fopen(MTRXINOUTDATA, "w");

if (fout == NULL) {

printf("Error");

return;

}

}

for (int i = 0; i < row\_size; i++) {

for (int j = 0; j < col\_size; j++) {

if ((i == 0) && (j == 0)) {

fprintf(fout, "%c%c%7.2lf", a6, a3, dm[i][j]);

}

else if ((i == 0) && j == (col\_size - 1)) {

fprintf(fout, "%7.2lf%c%c", dm[i][j], a6, a4);

}

else if ((i > 0) && (i < row\_size - 1) && (j == 0)) {

fprintf(fout, "%c%c%7.2lf", a6, a2, dm[i][j]);

}

else if ((i > 0) && (i < row\_size - 1) && (j == (col\_size - 1))) {

fprintf(fout, "%7.2lf%c%c", dm[i][j], a6, a2);

}

else if ((i == row\_size - 1) && (j == 0)) {

fprintf(fout, "%c%c%7.2lf", a6, a6, dm[i][j]);

}

else if ((i == row\_size - 1) && (j == (col\_size - 1))) {

fprintf(fout, "%7.2lf%c%c", dm[i][j], a6, a5);

}

else {

fprintf(fout, "%7.2lf", dm[i][j]);

}

}

fprintf(fout, "\n");

}

fclose(fout);

}

2) 행렬을 파일로 출력하는 함수를 작성하라

**행렬의 연산**

**1. 행렬의 덧셈, 뺄셈, 곱셈 연산**

void add\_mtrx(double\*\* dm1, double\*\* dm2, double\*\* dm3, int row\_size, int col\_size) {

for (int i = 0; i < row\_size; i++) {

for (int j = 0; j < col\_size; j++) {

dm3[i][j] = dm1[i][j] + dm2[i][j];

}

}

}

1) 2개의 행렬의 덧셈을 계산하고 결과행렬에 저장하는 함수를 작성하라.

void sub\_mtrx(double\*\* dm1, double\*\* dm2, double\*\* dm3, int row\_size, int col\_size) {

for (int i = 0; i < row\_size; i++) {

for (int j = 0; j < col\_size; j++) {

dm3[i][j] = dm1[i][j] - dm2[i][j];

}

}

}

2) 2개의 행렬의 뺄셈을 계산하여 그 결과를 주어진 행렬에 저장하는 함수를 작성하라.

void multiply\_mtrx(double\*\* dm1, double\*\* dm2, double\*\* dm3, int row\_size, int col\_size, int temp) {

for (int i = 0; i < row\_size; i++) {

for (int j = 0; j < col\_size; j++) {

for (int k = 0; k < temp; k++) {

dm3[i][j] += dm1[i][k] \* dm2[k][j];

}

}

}

}

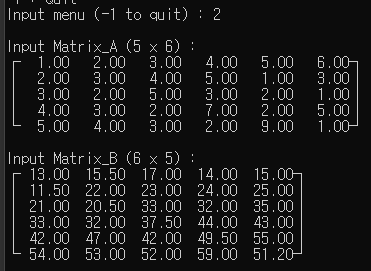
3) 2개의 크기가 다른 행렬의 곱을 계산하여 이를 결과행렬에 저장하는 함수를 작성하라.

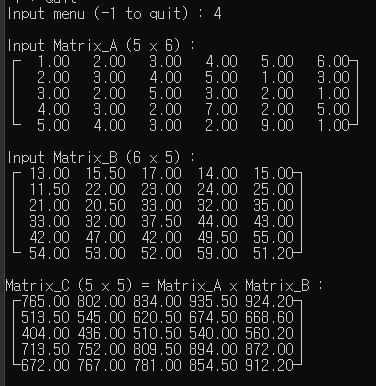
**main 프로그램과 실행 결과**

**1. main() 함수**

텍스트이(가) 표시된 사진

자동 생성된 설명

테이블이(가) 표시된 사진

자동 생성된 설명

Oral Test

|  |
| --- |
| **(1) 포인터 연산과 그 포인터의 자료형이 어떤 관계에 있는지 설명하라.**   * 주소 연산자 & 변수의 주소를 찾아낼 때 사용 * 간접 참조 연산자 \* 포인터가 가르키고 있는 곳의 값을 읽거나 쓸 때 사용 * [] 포인터를 배열의 이름처럼 사용하여, 동적 배열로 사용할 때 * ++/-- 자료형에 따라 포인터 값이 자료형의 크기만큼 증가 혹은 감소된다. char : 1, short : 2, int : 4, float : 4, double : 8 \*p++ : 값을 가져온 후 p 증가 (\*p)++ : 해당 위치의 값을 증가 |

|  |
| --- |
| **(2) 함수호출에서의 인수 전달 방식 중 call-by-value와 call-by-pointer의 차이점을 예를 들어 설명하고, call-by-pointer의 장점에 대하여 설명하라.**  값에 의한 호출은 데이터를 복사해서 사용하는 반면에 포인터에 의한 호출은 데이터의 주소를 넘겨받아서 해당 값 자체를 사용한다. 포인터에 의한 호출은 해당 값 자체를 사용하기 때문에 여러 개의 값을 반환할 수 있다.  포인터로 결과값을 반환할 경우에는 함수가 종료되었을 경우에도 사라지지 않는 변수를 반환해야 한다. |

|  |
| --- |
| **(3) 동적 메모리 할당 방법을 사용하여, 행렬 계산을 위한 2차원 배열을 동적으로 생성하는 절차와 삭제하는 절차에 대하여 각 단계별로 상세하게 설명하라.**  - 생성  1. 열의 크기 만큼 double의 2중 포인터를 자료형으로 가지는 동적 할당을 실행  2. 각 열의 원소에 대응하는 행의 크기만큼 반복문을 통해 동적 할당  3. 이전에 생성한 열에 대한 동적 할당과 행에 대한 동적할당을 합쳐서 2차원 배열로 사용한다.   * 삭제   1. 생성의 반대로 실행  2. 각 열의 원소에 대응하는 동적 할당들을 반복문을 통해 해제  3. 초기에 열의 크기만큼 생성한 동적할당을 해제 |

|  |
| --- |
| **(4) 지정된 파일로부터 행렬의 크기를 읽고, 이 행렬의 크기에 따라 double 자료형 2차원 배열을 동적으로 구성한 후, “행 x 열”개의 double 자료형 행렬 데이터를 파일로부터 읽어 동적으로 생성된 2차원 배열에 저장하는 절차에 대하여 각 단계별로 상세하게 설명하라.**  1. 지정된 파일로부터 행렬의 크기를 읽기  fscanf를 통해 txt파일의 행렬 상단에 기록되어 있는 행렬의 크기를 읽어낸다.  2. 행렬의 크기에 따라 double 자료형 2차원 배열을 동적으로 구성  문제 3과 같은 구성으로 배열을 동적으로 구성한다.  3. 행렬의 크기에 맞는 데이터를 파일로부터 읽어서 동적으로 생성된 2차원 배열에 저장하라  위에서 행렬의 크기를 읽었다면 파일의 커서는 데이터를 읽을 수 있는 위치에 있다. 그 지점부터 반복문을 통해 주어진 크기에 맞는 행렬 데이터를 동적으로 생성한 2차원 배열에 대입한다. |