**Lab 03**

|  |
| --- |
| 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 | 2021.03.18.목 |

1. **지정된 연월일이 서기 1년 1월 1일로부터 몇 번째 날인지, 그리고 무슨 요일인지를 계산하여 출력하는 프로그램 작성**
   1. 연월일 3개 정수로 지정된 날짜를 입력 받아, 서기 (AD)1년 1월 1일로부터 몇 번째 날이며, 무슨 요일인가를 계산하여 출력하기 위한 함수 bool isLeapYear(int y)와 int getDaysFromJan01AD01(int year, int month, int day)를 작성하라.

참고로 서기 1년 1월 1일은 월요일이다.

int getDaysFromJan01AD01(int year, int month, int day) {

int days\_from\_0 = 0;

int month\_days[13] = { 0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 };

for (int y = 1; y < year; y++) {

days\_from\_0 += isLeapYear(y) ? 366 : 365;

}

if (isLeapYear(year)) {

month\_days[2] = 29;

}

for (int m = 1; m < month; m++) {

days\_from\_0 += month\_days[m];

}

days\_from\_0 += day;

return days\_from\_0;

}

bool isLeapYear(int y) {

if ((y % 4 == 0 && y % 100 != 0) || y % 400 == 0) {

return true;

}

else {

return false;

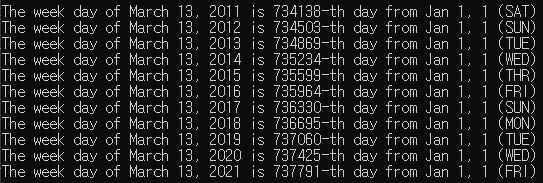
}

}

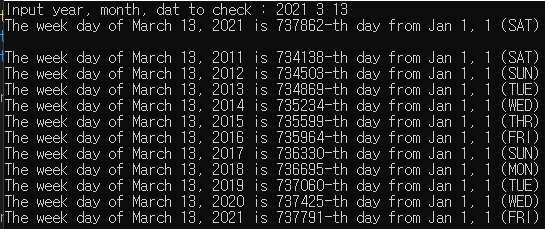
* 1. 오늘의 연 월 일 3개 정수를 입력 받아 서기 (AD)1년 1월 1일로부터 몇 번째 날이며, 무슨 요일인가를 계산하여 출력하라, 달에 해당하는 이름을 출력하고, 요일에 해당하는 이름을 각각 출력할 것.



* 1. 2011년 이후 올해까지의 매년 1월 1일이 서기 1년 1일로부터 몇 번째 날인지, 그리고 무슨 요일인가를 출력하라.



* 1. 실행 예제:



1. **5 x 5 크기의 행렬 A, B에 대한 덧셈, 뺄셈, 곱셈 연산**
   1. 2개의 5 x 5 크기의 행렬 A, B의 덧셈, 뺄셈, 곱셈을 계산하여 그 결과를 C, D, E에 저장하는 행렬 연산 함수 3개(addMtrx(), subMtrx(), mulMtrx())를 작성하라.

void mulMtrx(double A[][SIZE], double B[][SIZE], double X[][SIZE], int size) {

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

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

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

X[i][j] += A[i][k] \* B[k][j];

}

}

}

printMtrx(X, size);

}

void subMtrx(double A[][SIZE], double B[][SIZE], double X[][SIZE], int size) {

for (int row = 0; row < ROW; row++) {

for (int col = 0; col < COL; col++) {

X[row][col] = A[row][col] - B[row][col];

}

}

printMtrx(X, size);

}

void addMtrx(double A[][SIZE], double B[][SIZE], double X[][SIZE], int size) {

for (int row = 0; row < ROW; row++) {

for (int col = 0; col < COL; col++) {

X[row][col] = A[row][col] + B[row][col];

}

}

printMtrx(X, size);

}

* 1. 2차원 배열 A와 B를 각각 초기화하라, 초기화 데이터는 다음과 같이 설정할 것.

double A[ROW][COL] = { 0 };

double B[ROW][COL] = { 0 };

double X[ROW][COL] = { 0 };

int num = 1;

for (int row = 0; row < ROW; row++) {

for (int col = 0; col < COL; col++) {

if (row == col) {

B[row][col] = 1.0;

}

A[row][col] = num;

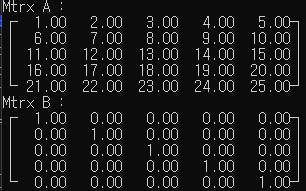
num += 1;

}

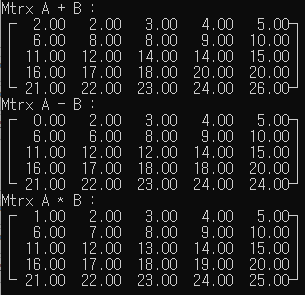
}

* 1. 초기화 된 2개의 행렬을 printMtrx(double M[][SIZE], int size)함수를 사용하여 출력할 것.

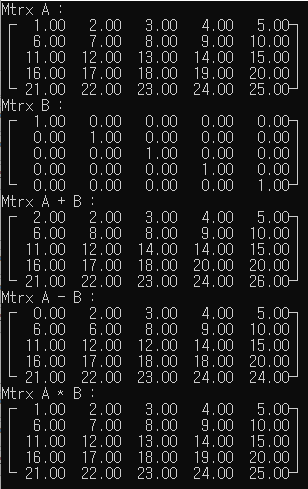
이 때, 행렬을 표시하기 위하여, 확장 완성형 코드를 사용할 것.



* 1. 행렬 A, B의 덧셈, 뺄셈, 곱셈을 위 1)에서 작성한 함수들을 사용하여 실행하라.
  2. 행렬 연산을 총괄하는 main() 함수에서 2개의 행렬 초기화를 실행하고, printMtrx(), addMtrx(), subMtrx(), mulMtrx() 함수를 호출하며, 행렬 준비, 행렬 연산 및 결과 출력을 실행하고, 그 결과를 확인 할 것.



* 1. 실행 예제:



Oral Test

|  |
| --- |
| **(1) enum을 사용하는 방법에 대하여 예를 들어 설명하라. enum으로 선언되는 기호 상수를 사용하여 요일의 이름과 달의 이름을 출력하는 방법에 대하여 설명하라.**  enum MONTH { January = 1, Fabruary, March, April, May, June, July, August, September, October, November, December };  enum WEEKDAY { SUN, MON, TUE, WED, THR, FRI, SAT };  위의 코드를 통해 열거형을 정의할 수 있다. 하지만, 정의만으로는 사용할 수 없고, 따로 변수를 선언 해 줘야 사용 가능하다.  열거형의 값은 처음에만 할당해주면 이 후의 값은 1씩 증가하면서 자동으로 할당됩니다.  따로 할당해주지 않으면 자동으로 0으로 할당됩니다.  for (int m = January; m <= December; m++) {  printf("%s\n", month\_name[m]);  for (int w = SUN; w <= SAT; w++) {  printf("%s ", week\_day\_name[w]);  }  printf("\n\n");  }  const char\* month\_name[NUM\_MONTHS] = { "", "January", "Fabruary", "March", "April", "May", "June", "July", "August", "September", "October", "November", "December" };  const char\* week\_day\_name[DAYS\_PER\_WEEK] = { "SUN", "MON", "TUE", "WED", "THR", "FRI", "SAT" };  위와 같이, 열거한 요일의 이름과 달의 이름을 문자열 배열로 선언한다.  위와 같이 반복문을 사용하여 요일의 이름과 달의 이름을 출력한다.  열거형을 사용하면 상수를 편리하게 정의할 수 있으며 가독성을 높일 수 있다. |

|  |
| --- |
| **(2) 반복문의 실행에서 continue와 break가 실행되면 어떤 결과가 나타나는지에 대하여 설명하라.**   * 반복문에 continue가 포함된 경우   continue 이후의 동작을 전부 무시하고 반복문의 조건식으로 돌아간다.     * 반복문에 break가 포함된 경우   해당 반복문을 탈출한다. |

|  |
| --- |
| **(3) 함수호출에서 2차원 배열 2개 (A, B)를 인수로 전달하고, 그 배열들에 대한 지정된 연산(+, -, \*)을 수행하여 또 다른 2차원 배열 X로 반환하기 위한 함수 matrixOperation() 구현 방법에 대하여 설명하라.**   * add\_matrix\_operation()   2차원 배열 A와 B의 각 요소들을 2중 반복문을 변수 i, j를 인덱스로 사용하여 순차적으로 덧셈 연산을 한 후 2차원 배열 X로 반환한다.  void add\_matrix\_operation(double A[][SIZE], double B[][SIZE], double X[][SIZE]) {  for (int row = 0; row < ROW; row++) {  for (int col = 0; col < COL; col++) {  X[row][col] = A[row][col] + B[row][col];  }  }  }   * sub\_matrix\_operation()   2차원 배열 A와 B의 각 요소들을 2중 반복문을 변수 i, j를 인덱스로 사용하여 순차적으로 뺄셈 연산을 한 후 2차원 배열 X로 반환한다.  void sub\_matrix\_operation(double A[][SIZE], double B[][SIZE], double X[][SIZE]) {  for (int row = 0; row < ROW; row++) {  for (int col = 0; col < COL; col++) {  X[row][col] = A[row][col] - B[row][col];  }  }  }   * mul\_matrix\_operation()   2차원 배열 A와 B의 각 요소들을 3중 반복문을 통해 연산한다.  A는 변수 i를 인덱스로 사용해 행을바꿔가며, B는 변수 j를 인덱스로 사용해 열을바꿔간다.  변수 k는 행과 열이 바뀔 때 배열의 size만큼 해당 요소를 이동한다.  이때, 대응하는 요소들의 곱을 합해서 해당 행과 열의 교차점에 값을 대입하여 2차원 배열 X로 반환한다.  void mul\_matrix\_operation(double A[][SIZE], double B[][SIZE], double X[][SIZE]) {  for (int i = 0; i < size; i++) {  for (int j = 0; j < size; j++) {  for (int k = 0; k < size; k++) {  X[i][j] += A[i][k] \* B[k][j];  }  }  }  } |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **(4) N x K 행렬 A와 K x M 행렬 B의 곱셈 계산을 하여 N x M 행렬 C에 저장하는 반복문의 구성에 대하여 설명하고, 각 항목의 계산이 어떤 순서로 실행되는가에 대하여 설명하고, 각 항목에 계산이 어떤 순서로 실행되는가에 대하여 그림으로 표현하고, 이를 설명하라.**  N x K 행렬 A와 K x M 행렬 B의 각 요소들을 3중 반복문을 통해 연산한다.  A는 변수 i를 인덱스로 사용해 행을바꿔가며, B는 변수 j를 인덱스로 사용해 열을바꿔간다.  변수 k는 행과 열이 바뀔 때 배열의 size만큼 해당 요소를 이동한다.  (행렬의 형태는 다르지만 행렬 A의 열과 행렬 B의 행의 크기가 같아서 곱셈 연산이 가능하다.)  이때, 대응하는 요소들의 곱을 합해서 해당 행과 열의 교차점에 값을 대입하여 2차원 배열 X로 반환한다.    void multiMtrx(int A[][K], int B[][M], int X[][M]) {  for (int i = 0; i < N; i++) {  for (int j = 0; j < M; j++) {  for (int k = 0; k < K; k++) {  X[i][j] += A[i][k] \* B[k][j];  }  }  }  }   |  |  |  |  | | --- | --- | --- | --- | |  | k = 1 | k = 2 | k = 3 | | X[0][0] | 0x0091f670 {1, 0, 0, 0} | 0x0091f670 {11, 0, 0, 0} | 0x0091f670 {38, 0, 0, 0} | | X[0][1] | 0x0091f670 {38, 2, 0, 0} | 0x0091f670 {38, 14, 0, 0} | 0x0091f670 {38, 44, 0, 0} | | X[0][2] | 0x0091f670 {38, 44, 3, 0} | 0x0091f670 {38, 44, 17, 0} | 0x0091f670 {38, 44, 50, 0} | | X[0][3] | 0x0091f670 {38, 44, 50, 4} | 0x0091f670 {38, 44, 50, 20} | 0x0091f670 {38, 44, 50, 56} | | X[1][0] | 0x0091f680 {4, 0, 0, 0} | 0x0091f680 {29, 0, 0, 0} | 0x0091f680 {83, 0, 0, 0} | | X[1][1] | 0x0091f680 {83, 8, 0, 0} | 0x0091f680 {83, 38, 0, 0} | 0x0091f680 {83, 98, 0, 0} | | X[1][2] | 0x0091f680 {83, 98, 12, 0} | 0x0091f680 {83, 98, 47, 0} | 0x0091f680 {83, 98, 113, 0} | | X[1][3] | 0x0091f680 {83, 98, 113, 16} | 0x0091f680 {83, 98, 113, 56} | 0x0091f680 {83, 98, 113, 128} | |