충북대학교SW중심대학산업단

C언어 프로그래밍 기초 및 실습

박상수 강사

2020년 8월 13일

강의 목차

- **제어문**
  - 반복문 (For Loop, While)
- **C언어의 아파트**
  - 배열 (숫자, 문자)과 포인터
- **함수 및 연습문제**
  - 함수 및 main 함수, 행렬 곱셈

반복문 사용이 필요한 경우 #1

- 프로그래밍 언어를 배우는 이유
  - 1부터 100까지 숫자의 합을 출력하는 프로그램을 만든다면 ?

```
#include <stdio.h>
int main() {
  printf("%d", 1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 10 + 11 + 12 + 13 + 14 + 15 +
              16 + 17 + 18 + 19 + 20 + 21 + 22 + 23 + 24 + 25 + 26 + 27 +
              28 + 29 + 30 + 31 + 32 + 33 + 34 + 35 + 36 + 37 + 38 + 39 +
              40 + 41 + 42 + 43 + 44 + 45 + 46 + 47 + 48 + 49 + 50 + 51 +
              52 + 53 + 54 + 55 + 56 + 57 + 58 + 59 + 60 + 61 + 62 + 63 +
              64 + 65 + 66 + 67 + 68 + 69 + 70 + 71 + 72 + 73 + 74 + 75 +
              76 + 77 + 78 + 79 + 80 + 81 + 82 + 83 + 84 + 85 + 86 + 87 +
              88 + 89 + 90 + 91 + 92 + 93 + 94 + 95 + 96 + 97 + 98 + 99 +
              100);
  return 0;
}
```

반복문이 필요한 경우

반복문 사용이 필요한 경우 #2

- 컴퓨터가 생겨난 이유는 ?
  - 반복적인 계산을 처리하기 위한 기계 (Machine)
  - 반복문은 컴퓨터에서 많이 사용되는 프로그램의 루틴

```c
#include <stdio.h>
int main() {
  int i;
  for (i = 0; i < 20; i++) {
    printf("숫자 : %d \n", i);
  }

  return 0;
}
```

```
숫자 : 0
숫자 : 1
숫자 : 2
.
.
.
숫자 : 17
숫자 : 18
숫자 : 19
```

반복문을 사용한 숫자 출력

반복문: For 문 #1

- For 문 구조
  - 특정 변수를 사용하여 반복의 조건을 결정 및 확인하여 실행
    - 반복문은 얼마나 반복을 해야 될지를 알아야 함
    - 제어 변수가 특정한 조건을 만족할 때에만 반복
  - `for (i=0; i<20; i++)`
    - 초기식 (제어 변수의 초기화): 처음 `i`의 값을 0으로 설정
    - 조건식 (특정 조건을 만족할 때만 반복문 반복): `i`의 값이 20보다 작은 경우에 조건 만족
    - 증감식 (1회 실행 시 제어 변수의 값을 어떻게 변화): `i`의 값을 1씩 증가

```
for (/* 초기식 */; /* 조건식 */; /* 증감식 */) {
  // 명령1;
  // 명령2;
  // ....
}
```

For 문의 포맷

반복문: For 문 #2

- For 문을 사용하는 프로그램의 예
  - 초기식: i의 값을 0으로 초기화
  - 조건식: i<20이 맞나 ?
    - 조건식 충족, 중괄호 속의 내용을 실행, 0 출력
  - 증감식: i++ (i=i+1)을 되어 있으므로
    - i의 값을 1증가
  - 조건식: i<20이 맞나 ?
    - i의 값이 20이므로 조건식을 충족하지 않음 따라서 For문 탈출 (숫자 20 출력 X)

초기식 조건식 증감식

```
for (i = 0; i < 20; i++) {
  printf("숫자 : %d \n", i);
}
```

For 문의 포맷: 초기식/조건식/증감식

반복문: For 문 실습

- For 문을 사용해 1부터 19까지의 합을 구하는 프로그램
  - For 문은 {} 안에 작업들이 조건식이 성립할 동안 반복
    - 반복과 함께 매 반복바다 증감식을 실행
    - 이러한 For 문의 특징을 고려하여 아래의 프로그램의 조건식과 증감식 부분 작성 !

```c
/* 1 부터 19 까지의 합*/
#include <stdio.h>
int main() {
  int i, sum = 0;
  for (i=0;___;___) {
    sum = sum + i;
  }
  printf("1 부터 19 까지의 합 : %d", sum);

  return 0;
}
```

```
1 부터 19 까지의 합 : 190
```

For 문의 예: 숫자의 합을 계산하는 프로그램

반복문: While 문 #1

- For 문과 유사한 반복문
  - For 문 처럼 '조건식 '에는 While 문을 계속 반복할 조건
    - **시작부터 조건식을 검사 (While 문의 특징)**
      - i의 값이 100 이하인지 검사 (i<=100)
      - sum에 i의 값을 더하는 과정 (sum+=i)
      - i의 값을 증감하고 (i++)
      - 다시 처음으로 돌아가서 조건식 검사

```
while (/* 조건식 */) {
  // 명령1;
  // 명령2;
  // ...
}
```

```
while (i<=100) {
  sum+=i;
  i++;
}
```

While 문의 포맷

반복문: While 문 #2

- While 문을 사용한 구구단 프로그램
  - While 문은 조건식이 충족된다면 실행문을 실행
    - {} 안에 증감에 대한 코드가 없더라도 동작 가능
    - 하지만 While 문을 나오는 구문이 없다면 **무한루프**로 동작
    - 다음의 코드를 실행하면 ?

```c
#include <stdio.h>
int main(){
    int i=1;

    while(i<10){
        // int j=1;
        while(j<10){
            printf("%d * %d = %d\n",i,j,i*j);
            //j++;
        }
        i++;
        printf("\n");
    }
}
```

While 문을 사용한 프로그램의 예: 구구단

반복문: While 문 #3

- 조건문을 사용한 반복문의 탈출
  - **무한루프**: 프로그램이 일련의 명령을 무한히 반복하는 경우 (실수/의도적으로 발생)
    - While 문은 무한루프를 만들기 쉬움
    - 무한루프를 만들고 어떤 특정 입력이나 조건이 성립되었을 때 무한 루프를 탈출 !
      - For 문 보다는 While 문을 사용하여 구현하는 것이 유용
      - 루프의 탈출을 위해서는 break 사용

```c
#include <stdio.h>
int main(){
    while(1){
        printf("while문 실행\n");
        break;
        printf("bareak문 뒷문장은 실행되지 않습니다.\n");
    }
    printf("프로그램을 종료합니다.");
}
```

While 문을 사용한 프로그램의 예: 무한루프

반복문: Do-While 문

- 조건식을 나중에 점검하는 반복문
  - While/For 문: 조건식을 먼저 점검하고, 조건에 따라 반복문의 실행 여부 결정
    - 따라서, 처음부터 참이 아니라면 실행 While 문 안의 내용은 실행 불가능
  - Do-While 문: 명령을 실행 한 후 조건문을 검사
    - 최소한 한 번은 실행
    - 조건식 (i<1)을 나중에 검사하기 때문에 sum의 값 1 출력

```c
#include <stdio.h>
int main() {
  int i = 1, sum = 0;

  do {
    sum += i;
    i++;
  } while (i < 1);

  printf(" sum : %d \n", sum);
  return 0;
}
```

Do-While 문의 예

반복문: 실습

- 반복문을 사용하여 별을 출력하는 프로그램
  - 변수 i는 가로 줄 선택, j는 선택된 가로 줄에 몇 개의 별을 출력할 것인지 결정
    - 사각형: 가로 한 줄에 5개의 별을 출력, 5개의 가로 줄 출력
    - 직각사각형: 가로 한 줄에 "**가로 줄 숫자/5-가로 줄 숫자**"를 사용하여 별 출력

```c
#include <stdio.h>

int main() {
    for(int i=0;i<5;i++){
        for(int j=0;j<5;j++){
          printf("*");
        }
        printf("\n");
    }
}
```

```
*****
*****
*****
*****
*****
```

```c
#include <stdio.h>

int main() {
    for(int i=0;i<5;i++){
        for(int j=0;j<=i;j++){
          printf("*");
        }
        printf("\n");
    }
}
```

```
i/j
0/0
0/1
0/2
.
.
.
```

```
*
**
***
****
*****
```

```c
#include <stdio.h>

int main() {
    for(int i=0;i<5;i++){
        for(int j=0;j<____;j++){
          printf("*");
        }
        printf("\n");
    }
}
```

```
i/j
0/5
0/4
0/3
.
.
.
```

```
*****
****
**
*
*
```

반복문을 사용한 별 출력의 예

강의 목차

- 제어문
  - 반복문 (For Loop, While)
- **C언어의 아파트**
  - 배열 (숫자, 문자)과 포인터
- 함수 및 연습문제
  - 함수 및 main 함수, 행렬 곱셈

배열 (Array) #1

- 많은 수의 데이터를 관리할 때 편리한 방법
  - 100개의 수를 모두 저장하기 위해서는 ?
    - 값을 기억할 변수를 선언해야 하며, 100개의 변수 선언 필요

```
int num1; int num2; int num3; ... ... int num99; int num100;
```

    - 만약 전국의 모든 학생수에 해당하는 프로그램을 작성한다면 많은 변수가 필요
    - 'Ctrl+C', 'Ctrl+V' 방법으로는 코딩하는 것이 불가능
    - **C언어에는 배열 (Array)라는 변수들의 집합 처럼 사용가능한 기능 존재**

```c
/* 배열 기초 */
#include <stdio.h>
int main() {
  int arr[10] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};

  printf("Array 3 번째 원소 : %d \n ", arr[2]);
  return 0;
}
```

배열 사용의 예

배열 (Array) #2

- 특정한 형 (Type)의 변수들의 집합
  - 배열을 선언하는 방법

```
Array_type Array_name [# of Array];  // 배열의_타입 배열의_이름 원소_개수 순으로 입력
```

    - Array_type: 배열의 자료형 (int, double, char 등)
    - Array_name: 배열의 이름
    - # of Array: 배열의 크기
  - 배열의 데이터를 사용하는 방법
    - 배열의 n번째 원소에 접근하기 위해서는 array[n-1] 로 사용하면 가능
    - n번째 원소에 접근하는 경우 (array[n])는 오류 발생: 존재하지 않는 원소에 접근

```c
/* 배열 출력하기 */
#include <stdio.h>
int main() {
  int arr[10] = {2, 10, 30, 21, 34, 23, 53, 21, 9, 1};
  int i;
  for (i = 0; i < 10; i++) {
    printf("배열의 %d 번째 원소 : %d \n", i + 1, arr[i]);
  }
  return 0;
}
```

배열의 데이터를 사용하는 예

배열 (Array) 실습

- 학생들의 점수를 입력 받아 평균을 계산하는 프로그램
  - 배열의 n번째 원소에 접근하기 위해서는 array[n-1] 로 사용
    - 5명의 평균을 계산하기 위해서는 특정 변수에 학생들이 획득한 값을 누적
    - 누적된 값을 학생 수와 동일한 값으로 나눠 평균을 계산

```c
/* 평균 구하기*/
#include <stdio.h>
int main() {
  int arr[5]={100, 100, 80, 90, 0};  // 성적을 저장하는 배열
  int i, ave = 0;

  for (i = 0; i < 5; i++)  // 학생들의 성적을 입력받는 부분
  {
    printf("%d 번째 학생의 성적은? ", i + 1);
  }

  for (i = 0; i < 5; i++)  // 전체 학생 성적의 합을 구하는 부분
  {
    ave = ave + ____;
  }

  // 평균이므로 5 로 나누어 준다.
  printf("전체 학생의 평균은 : %d \n",  ave / 5);
  return 0;
}
```

```
1 번째 학생의 성적은? 100
2 번째 학생의 성적은? 100
3 번째 학생의 성적은? 80
4 번째 학생의 성적은? 90
5 번째 학생의 성적은? 0
전체 학생의 평균은 : 74
```

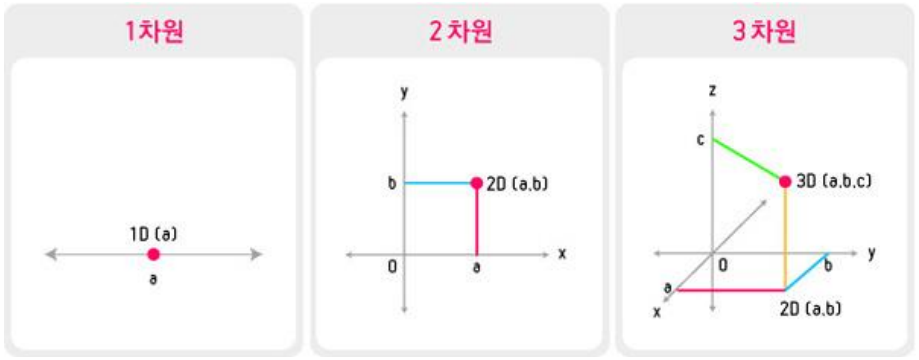
배열을 사용해 평균을 계산하는 프로그램

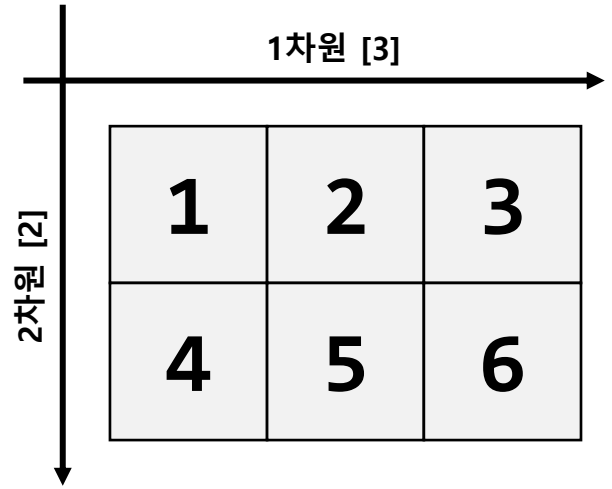
고차원 배열 (Array) #1

- 2차원 이상의 차원으로 구성된 배열
  - 행렬 연산, 그래픽과 같은 작업을 위해서는 고차원의 배열 필요

```
Array_type Array_name [# of Array][# of Array];    // 배열의_타입 배열의_이름 원소_개수 (차원) 순으로 입력
```

  - Array_type: 배열의 자료형 (int, double, char 등)
  - Array_name: 배열의 이름
  - # of Array: 배열의 크기 (3차원, 2차원, 1차원)






```
int array [2][3]={{1,2,3},{4,5,6}}
```

고차원의 행렬의 예 (좌), 행렬에 저장된 값 (우)

고차원 배열 (Array) #2

- 2차원 배열을 사용하여 학생의 점수를 입력 받는 프로그램
  - Scanf(자료형 타입(format), 변수): 데이터를 읽어와 형식에 따라 변수에 저장하는 함수
  - int score[3][2]: 3행, 2열 크기를 갖는 2차원 배열을 선언
  - 배열에 저장된 값을 출력하고 값이 생각한 것 처럼 저장이 되었는지 확인하세요

```c
/* 학생 점수 입력 받기 */
#include <stdio.h>
int main() {
  int score[3][2];
  int i, j;

  for (i = 0; i < 3; i++)  // 총 3 명의 학생의 데이터를 받는다
  {
    for (j = 0; j < 2; j++) {
      if (j == 0) {
        printf("%d 번째 학생의 국어 점수 : ", i + 1);
        scanf("%d", &score[i][j]);
      } else if (j == 1) {
        printf("%d 번째 학생의 수학 점수 : ", i + 1);
        scanf("%d", &score[i][j]);
      }
    }
  }

  for (i = 0; i < 3; i++) {
    printf("%d 번째 학생의 국어 점수 : %d, 수학 점수 : %d \n", i + 1,
           score[i][0], score[i][1]);
  }

  return 0;
}
```

	국어	수학
학생1	Score[0][0]	Score[0][1]
학생2	Score[1][0]	Score[1][1]
학생3	Score[2][0]	Score[2][1]

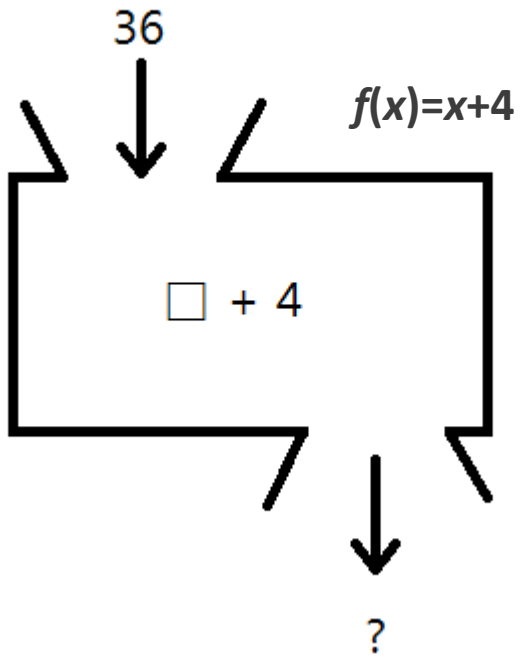
고차원 행렬을 사용하여 학생의 점수를 저장하는 프로그램

강의 목차

- 제어문
  - 반복문 (For Loop, While)
- C언어의 아파트
  - 배열 (숫자, 문자)과 포인터
- **함수 및 연습문제**
  - 함수 및 main 함수, 행렬 곱셈

함수 (Function) #1

- 첫 번째 집합의 임의의 원수를 두 번째 집합의 오직 한 원소에 대응하는 관계 (사전적 정의)
  - 수학에서의 함수는 마술 상자와 비슷한 개념
    - 특별한 값 (x)를 입력 받아 내부에서 연산을 취하여 결과를 출력하는 것을 함수라 정의
    - 아래의 그림에서 출력의 값은 어떤 값 ?



함수: 블랙박스

함수 (Function) #2

- `int print_hello()`
  - 정의 (definition): 함수 이름 (`print_hello`) + Return 타입 (`int`)
    - Return: 특정 연산을 수행했을 때의 결과값 (예: 36을 마술상자에 넣는다면 4를 더해서 40을 출력)
    - `()`는 함수의 이름에 포함되지는 않지만 함수라는 것을 나타내는 표현
    - {} 내부에 함수가 수행하는 연산을 정의

```c
#include <stdio.h>
/* 보통 C 언어에서, 좋은 함수의 이름은 그 함수가
무슨 작업을 하는지 명확히 하는 것이다. 수학에서는
f(x), g(x) 로 막 정하지만, C 언어에서는 그 함수가 하는
작업을 설명해주는 이름을 정하는 것이 좋다. */

int print_hello() {
  printf("Hello!! \n");
  return 0;
}

int main() {
  printf("함수를 불러보자 : ");
  print_hello();

  printf("또 부를까? ");
  print_hello();
  return 0;
}
```

함수의 예: print_hello ()

함수 (Function) #3

- `int main()`
  - 프로그램을 실행할 때 컴퓨터가 처음 시작하는 함수
    - `main()` 함수를 가장 먼저 호출
    - 모든 C언어로 작성된 프로그램은 반드시 포함

- 함수의 인자 (Argument)
  - `int slave(int master_money)`
    - 함수는 return을 사용하여 결과값을 전달, 인자 (`master_money`)를 사용하여 연산에 사용 값을 받음
    - 인자 값은 반드시 데이터 타입이 명시되어야 함

```c
#include <stdio.h>
int slave(int master_money) {
  master_money += 10000;
  return master_money;
}

int main() {
  int my_money = 100000;
  printf("2020.12.12 재산 : $%d \n", slave(my_money));

  return 0;
}
```

```
2020.12.12 재산 : $110000
my_money : 100000
```

함수의 인자를 전달하는 경우의 예

함수 (Function) 실습

- 학생의 성적을 입력 받고, 평균을 계산하는 함수를 포함하는 프로그램
  - `int average_calculation (int number[], int count)`
    - 배열을 인자 (`int number[]`)로 받을 때는 "변수_타입 변수 이름 []" 으로 명시
    - `int count`: 각 과목당 학생의 숫자

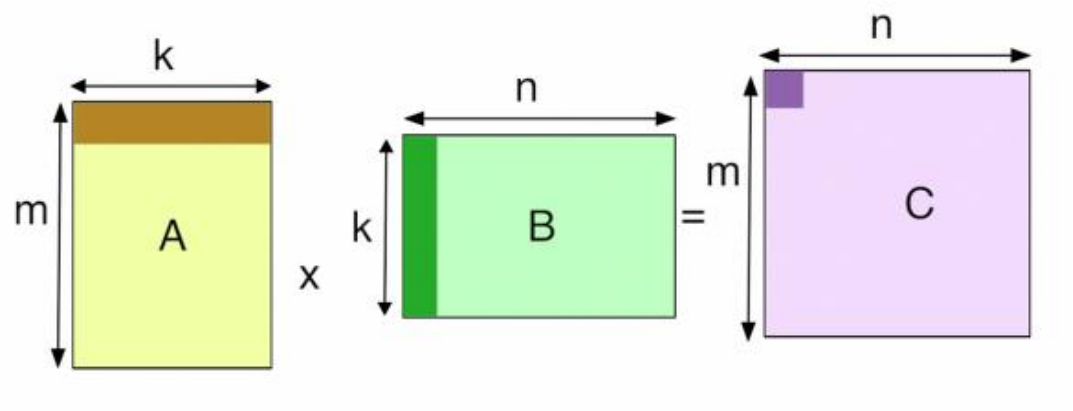
```
int average_calculation (int number[], int count)
{
  int i;
  int sum=0;
  // For 문을 사용하여 각 과목의 합을 계산 (변수 i를 사용하며, i는 학생의 숫자보다 작은 값만큼 동작)
  for (_____________)
    sum += number[i]; // sum 변수에 학생들의 점수를 누적하여 점수 합산

  return ____; // 계산된 값을 학생의 숫자만큼 나눠서 return
}

/* 학생 점수 입력 받기 */
#include <stdio.h>
int main(){

 int kor_score[3]={80,100,90};
 int math_score[3]={100,100,50};
 int i, j;

 printf("학생들의 국어점수의 평균은 %d 입니다 \n", average_calculation(kor_score, 3));
 printf("학생들의 수학점수의 평균은 %d 입니다 \n", average_calculation(math_score, 3));

 return 0;
}
```

```
학생들의 국어점수의 평균은 270 입니다
학생들의 수학점수의 평균은 250 입니다
```

배열을 인자로 받는 함수

연습문제

- 행렬 곱셈 프로그램
  - 인공지능, 수치해석 등 다양한 분야에서 가장 널리 사용되는 수학 연산
  - A행렬의 행과 B행렬의 열을 곱하고 누적하여 계산
    - 예) A행렬의 행 [1,2], B행렬의 열 [5,7], 1*5+2*7=19



$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \times \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix} = \begin{bmatrix} 1\times5 + 2\times7 & 1\times6 + 2\times8 \\ 3\times5 + 4\times7 & 3\times6 + 4\times8 \end{bmatrix}$$

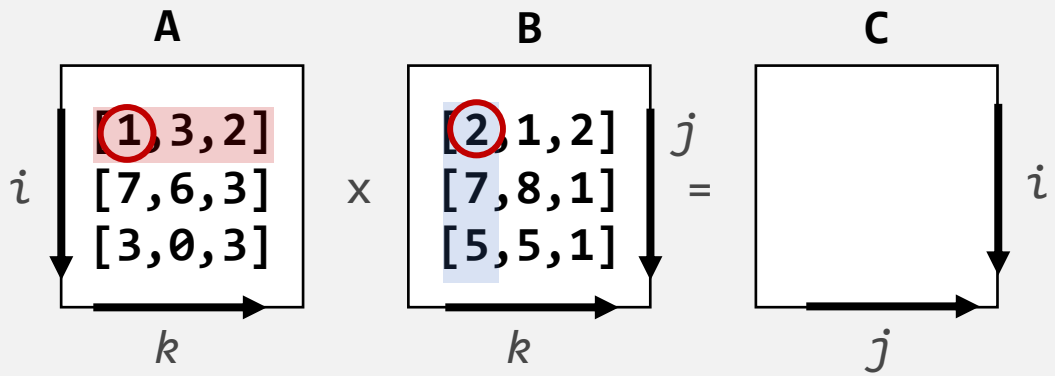
$$= \begin{bmatrix} 5+14 & 6+16 \\ 15+28 & 18+32 \end{bmatrix} = \begin{bmatrix} 19 & 22 \\ 43 & 50 \end{bmatrix}$$

연습문제

행렬 곱셈 프로그램

- 반복문 (for 문)을 사용하여 행렬 곱셈 !

```c
#include <stdio.h>
#define l 3 #define m 3 #define n 3

int main(void) {

    /* x[a][b] has a rows, b columns */
    int A[l][m] = { { 1,3,2 }, { 7,6,3 }, { 3,0,3 } };
    int B[m][n] = { { 2,1,2 }, { 7,8,1 }, { 5,5,1 } };
    int C[l][n];    // C = A x B
    int i, j, k;

    for(i = 0; i < l; i++)
        for(j = 0; j < n; j++){
            C[i][j] = 0;

            for(k = 0; k < m; k++)
                C[i][j] += _____________;
        }

    for(i = 0; i < 3; i++){
        for(j = 0; j < n; j++)
            printf("%d\t", C[i][j]);
        printf("\n");
    }

    return 0;
}
```



(1) A행렬의 행 선택 (A[i][ ])
(2) B행렬의 열 선택 (B[ ][j])
(3) A행렬의 열과 B행렬의 행 선택 (A[i][k])* B[k][j]))

```
33    35    7
71    70    23
21    18    9
```

반복문을 사용하는 행렬 곱셈 프로그램

감사합니다