|  |  |
| --- | --- |
| 이름 | 2171262 박소희 |
| 주제 | Chap 03. 배열, 구조체, 포인터 |
| 일시 | 2023 03.30 18:00 ~ 19:30 디스코드 |

이번 시간에는 chap 03을 공부하고 연습문제를 각자 풀어보기로 하였다.

1. 크기가 10인 배열 two[]를 선언하고 여기에 2의 제곱값들을 저장해보자. 배열의 첫 번째 요소에는 2의 0승을 저장하고 마지막 요소에는 2의 9승을 저장한다.

#include <stdio.h>

int power(int num, int k)

{

if(k == 0) return 1;

else return (num \* power(num, k-1));

}

void insert\_array(int\* arr, int size)

{

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

{

arr[i] = power(2, i);

}

}

void print\_array(int \* arr, int size)

{

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

{

printf("%d ",arr[i]);

}

printf("\n");

}

int main(void)

{

int two[10];

int size = sizeof(two)/sizeof(int);

insert\_array(two, size);

print\_array(two, size);

return 0;

}

위와 같은 방식으로 배열의 크기를 구하고 insert\_array 함수와 print\_array 함수를 호출해주었다. 특히 insert\_array 함수에서는 int형을 반환하는 power함수를 호출해주어 2의 거듭제곱이 반환되도록 하였다. power함수에서는 앞서 학습했던 재귀의 형식을 사용하였다. 만약 반복문의 형식을 사용하면 다음과 같다.

int power(int num, int k)

{

int val = 1;

for(int i = 0; i < k; i++)

{

val = num \* val;

}

return val;

}

단순히 2의 거듭제곱 수식을 반복문으로써 구현해내면 된다.

텍스트이(가) 표시된 사진

자동 생성된 설명

1. 전치행렬을 계산해보자. 필자는 주어진 배열값을 전치계산하여 또 다른 배열의 원소로 값을 대입하는 방법과(방법 1) 주어진 배열내에서 전치계산하여 그 배열을 그대로 리턴하는 방식(방법 2)를 모두 구현해보았다.

방법 1 코드)

#include <stdio.h>

#define ROWS 3

#define COLS 3

void matrix\_transpose(int a[ROWS][COLS], int b[ROWS][COLS])

{

for(int r = 0; r < ROWS; r++)

{

for(int c = 0; c < COLS; c++)

{

b[c][r] = a[r][c]; //단순히 행과 열을 바꾸어 값 대입

}

}

}

void matrix\_print(int a[ROWS][COLS])

{

printf("==========================\n");

for(int r = 0; r < ROWS; r++)

{

for(int c = 0; c < COLS; c++)

{

printf("%d ",a[r][c]);

}

printf("\n");

}

printf("==========================\n");

}

int main (void)

{

int arr1[ROWS][COLS] = {{2, 3, 0},

{8, 9, 1},

{7, 0, 5}};

int arr2[ROWS][COLS];

matrix\_transpose(arr1, arr2);

matrix\_print(arr1);

matrix\_print(arr2);

return 0;

}

방법 2) 배열 내에서 인덱스 값 옮기기

#include <stdio.h>

#define ROWS 3

#define COLS 3

//전치 행렬로 변환하는 과정

void matrix\_transpose(int a[ROWS][COLS])

{

for(int r = 0; r < ROWS; r++)

{

for(int c = 0; c < r; c++)

{

a[r][c] ^= a[c][r]; //xor 연산으로 전치 구현

a[c][r] ^= a[r][c];

a[r][c] ^= a[c][r];

}

}

}

void matrix\_print(int a[ROWS][COLS])

{

printf("==========================\n");

for(int r = 0; r < ROWS; r++)

{

for(int c = 0; c < COLS; c++)

{

printf("%d ",a[r][c]);

}

printf("\n");

}

printf("==========================\n");

}

int main (void)

{

int arr1[ROWS][COLS] = {{2, 3, 0},

{8, 9, 1},

{7, 0, 5}};

//또 다른 배열이 아닌 arr1에 직접 전치행렬 구현

matrix\_print(arr1);

matrix\_transpose(arr1);

matrix\_print(arr1);

return 0;

}

위와 같이 구현할 수 있다. 방법 1은 행과 열만큼 loop를 돌려서 단순히 행과 열을 나타내는 인덱스를 바꾸어서 또 다른 배열에 대입을 하면 되지만 방법 2는 자기자신을 전치한 내용을 그대로 리턴하는 방식이기 때문에 바깥 루프는 행만큼, 안쪽 루프는 해당 row까지만 즉 전체 1/2만 루프하여 값을 교환해준다.

여기서 temp 임시 변수를 통해 값을 바꾸어 줄 수도 있지만 XOR 연산을 통해 교환해주면 레지스터에 값을 load 및 store 할 필요 없이 비트 연산으로 수행되기 때문에 속도가 더 빨라진다는 장점이 있다.

텍스트이(가) 표시된 사진

자동 생성된 설명

1. 연습문제 8번 ) 크기가 n인 배열 array에서 임의의 위치 loc에 정수 value를 삽입하는 함수 insert를 작성해보자. 정수가 삽입되면 그 뒤의 정수들은 모두 한 칸 씩 밀려나야 한다.

#include <stdio.h>

#include <stdlib.h>

void insert(int array[], int loc, int value, int len)

{

for(int i = 0; i < len-loc-1; i++)

{

array[len-1-i] = array[len-1-(i+1)];

}

array[loc] = value;

}

void print\_array(int array[], int len)

{

for(int i = 0; i < len; i++)

{

printf("%d ", array[i]);

}

printf("\n");

}

int main(void)

{

int arr[10] = {0, };

int loc = 0, value = 0;

int len = sizeof(arr)/sizeof(int);

printf("배열 요소 input : ");

for(int i = 0; i < len; i++)

{

scanf("%d",&arr[i]);

}

printf("인덱스 / 위치 각각 input : ");

scanf("%d %d",&loc, &value);

insert(arr, loc, value, len);

print\_array(arr, len);

return 0;

}

다음과 같이 배열 요소를 입력받고 insert함수와 print\_array함수를 각각 호출한다. insert함수에서는 (배열의 크기-인덱스 위치) 만큼 loop하여 뒤로 한 칸씩 이동하도록 한다. 반복문이 끝나면 loc인덱스 위치에 value 값을 삽입한다.

텍스트이(가) 표시된 사진

자동 생성된 설명