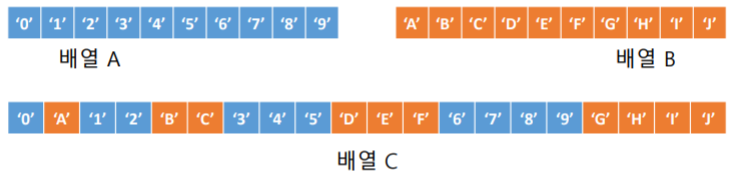
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 11주차 과제 | | | | | |
| 학년 | 2 | 학번 | 2017707007 | 이름 | 강민준 |

1. 과제 설명
   1. 과제 1번

길이가 x인 배열 A, B를 조합하여 길이가 2X인 배열 C를 만듦.



길이가 x인 char형 배열A, B를 만들고 (이 떄, x는 입력받음)

길이가 2x인 char형 배열 C를 만들어 값을 할당.

* 1. 과제 2번

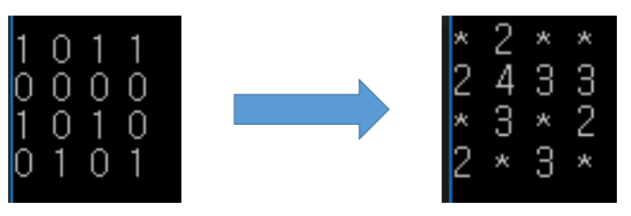
동적 할당을 이용하여, 배열 크기 할당 후, 문장을 입력하고 다시 출력함



* 1. 과제 3번

지뢰 찾기 게임을 구현한다.

동적 할당 배열에 0또는 1(지뢰)로 입력 받고 , 지뢰는 \*, 지뢰가 아닌 경우, 주변 지뢰 개수출력.



1. 과제 이론
   1. 과제 1번

길이가 x인 char형 배열 A, B라고 명시했으니, 동적 할당을 사용하여, 배열 길이를 결정하고, 특정한 수식을 사용해서 배열 C를 만듦. Sorce Code

#include<stdio.h>

#include<stdlib.h>

int main()

{

int ARRAY\_SIZE;

do

{

printf("배열 갯수 입력 [1~10]: ");

scanf("%d", &ARRAY\_SIZE);

} while (ARRAY\_SIZE < 1 || ARRAY\_SIZE>10);

char \*cp\_num, \*cp\_char, \*cp\_sum;

cp\_num = (char \*)calloc(ARRAY\_SIZE \* sizeof(char) + 1, sizeof(char));

cp\_char = (char \*)calloc(ARRAY\_SIZE \* sizeof(char) + 1, sizeof(char));

cp\_sum = (char \*)calloc(2\*ARRAY\_SIZE \* sizeof(char) + 4, sizeof(char));

//malloc이 아니라 calloc으로 한이유는 초기값을 0으로 주어 출력할 때 편하기 위함이다.

if (!cp\_num || !cp\_char || !cp\_sum) { //memory공간이 부족하여 NULL 을 RETURN 할 때,

printf("메모리 부족\n");

return 1;

}

for (int i = 0; i < ARRAY\_SIZE; i++) { //값 대입

//itoa(i, &cp\_num[i], 10); 변환 대상, 값을 넣을 배열, 10진수 //이걸 사용한 이유는 접근을 잘못했다.

cp\_num[i] = i + 48;

cp\_char[i] = i + 65;

}

int i = 0, k = 0, l = 0;

while (i < ARRAY\_SIZE\*2) { //i가 SIZE\*2 보다 작을 때

for (int j = 0; j <= k; j++) { //for 문 2개는 전부 문장을 만들기 위함.

cp\_sum[i] = cp\_num[j + l];

i++;

}

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

cp\_sum[i] = cp\_char[j + l];

i++;

}

k++;

l = 2 \* k - 1;

if (i == 12)l++; // l에 대한 수식을 2k-1로 정의했는데, i가 12일때 6이필요하지만 수식을 생각못해 조건문 추가.

}

printf("Array A : %s\n", cp\_num); //출력

printf("Array B : %s\n", cp\_char);

printf("Array C : %s\n", cp\_sum);

free(cp\_num); //할당받은 메모리 해제

free(cp\_char);

free(cp\_sum);

}

* 1. 과제 2번

동적 할당을 이용하여 배열을 만들고 값을 입력하고, 출력하는 것.

Sorce Code

#include<stdio.h>

#include<stdlib.h>

int main()

{

char \*cp;

char buffer;

int array\_size;

printf("할당하려는 크기 : ");

scanf("%d", &array\_size);

while (buffer = getchar() != '\n' && buffer != EOF);

cp = (char\*)calloc(array\_size+1, sizeof(char));

printf("출력히려는 문장 : ");

scanf("%[^\n]s", cp);

printf("출력확인 : %s\n",cp);

free(cp);

}

* 1. 과제 3번

이 때 1차정리를 이용하면, 지뢰는 1로 표현하고 지뢰가 아닐 경우 0으로 표현함.

1차정리에 가진 식으로 2차정리를 이용해서 지뢰가 아닌 경우 주변 8칸에 지뢰 수를 출력.

Sorce Code

#include<stdio.h>

#include<stdlib.h>//random, malloc

#include<time.h>//random

void print(int \*ip, int array\_size);

void change\_star(int \*ip, int array\_size);

int select\_num(int \*ip, int array\_size);

int main()

{

srand((unsigned int)time(NULL));

int array\_size;

printf("n\*n개의 지뢰찾기입니다.\nn개를 입력해주세요 : ");

scanf("%d", &array\_size);

int \*ip; // 지뢰배열 동적할당

ip = (int \*)malloc(array\_size\*array\_size \* sizeof(int));

for (int i = 0; i < array\_size\*array\_size; i++) //지뢰 값을 랜덤으로 넣어줌.

{

ip[i] = rand() % 2;

}

printf("\n\n");

print(ip, array\_size);

printf("=== 변환 후 ===\n\n\n");

change\_star(ip, array\_size); // 0과 1로 이루어진 배열을 1은 \*로 바꾸는 작업을 거침.

select\_num(ip, array\_size); //주변에 지뢰갯수가 몇개인지 count

print(ip, array\_size);

free(ip);

}

void print(int \*ip, int array\_size) //출력전용 함수.

{

int a = 1;

for (int i = 0; i < array\_size\*array\_size; i++)

{

if (ip[i] < 10) //별이 아니라면, ASCII code 이용

{

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

}

else {

printf("%c ", ip[i]);

}

if (i == a \* array\_size - 1) {

a++;

printf("\n");

}

}

printf("\n\n");

}

void change\_star(int \*ip, int array\_size) //1이면 별로바꾸는 함수

{

for (int i = 0; i < array\_size\*array\_size; i++)

{

if (ip[i])

{

ip[i] = '\*';

}

}

}

int select\_num(int \*ip, int n) //지뢰가 아닌경우 주변 cnt++ 해주는 함수.

{

int \*\*ip2;

int cnt = 0;

ip2 = (int \*)malloc(n \* sizeof(int\*));

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

{

ip2[i] = malloc(sizeof(int) \* n);

}

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

{

for (int j = 0; j < n; j++)

{

ip2[i][j] = ip[k];

k++;

}

}

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

{

for (int j = 0; j < n; j++)

{

if (!(ip2[i][j])) //별이 아니라면,

{

cnt = 0;

int flag[4] = { 1,1,1,1 }; //[0]x\_left, [1]x\_right, [2]y\_up, [3]y\_down

//flag를 두어, 모서리인지 맨윗줄인지 등을 검사

if (i - 1 < 0)

{

flag[2] = 0;

}

if (i + 1 >= n)

{

flag[0] = 0;

}

if (j - 1 < 0)

{

flag[3] = 0;

}

if (j + 1 >= n)

{

flag[1] = 0;

}

// 주변 8개 모두를 검사하고, cnt++을 한다

if (flag[2] && (ip2[i - 1][j] == '\*'))

cnt++;

if (flag[0] && (ip2[i + 1][j] == '\*'))

cnt++;

if (flag[3] && (ip2[i][j - 1] == '\*'))

cnt++;

if (flag[1] && (ip2[i][j + 1] == '\*'))

cnt++;

if (flag[0] && flag[3] && (ip2[i + 1][j - 1] == '\*'))

cnt++;

if (flag[2] && flag[1] && (ip2[i - 1][j + 1] == '\*'))

cnt++;

if (flag[2] && flag[3] && (ip2[i - 1][j - 1] == '\*'))

cnt++;

if (flag[0] && flag[1] && (ip2[i + 1][j + 1] == '\*'))

cnt++;

//ip[k]에 바로 값을 넣어준다.

ip[k] = cnt;

}

k++;

}

}

free(ip2); // 이차원배열로 보기 편하게 사용했던 ip2는 해제한다.

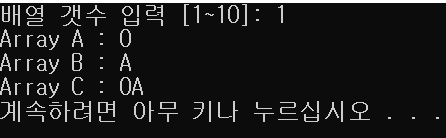
}

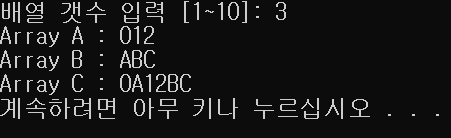
1. 주요 소스 설명

|  |
| --- |
| #과제 1번    for (int i = 0; i < ARRAY\_SIZE; i++) { //값 대입  //itoa(i, &cp\_num[i], 10); 변환 대상, 값을 넣을 배열, 10진수  cp\_num[i] = i + 48;  cp\_char[i] = i + 65;  }  int i = 0, k = 0, l = 0;  while (i < ARRAY\_SIZE\*2) { //i가 SIZE\*2 보다 작을 때  for (int j = 0; j <= k; j++) { //for 문 2개는 전부 문장을 만들기 위함.  cp\_sum[i] = cp\_num[j + l];  i++;  }  for (int j = 0; j <= k; j++) {  cp\_sum[i] = cp\_char[j + l];  i++;  }  k++;  l = 2 \* k - 1;  if (i == 12)l++; // l에 대한 수식을 2k-1로 정의했는데, i가 12일때 6이필요하지만 수식을 생각못해 조건문 추가.  }  Char형 배열로 선언하여, for문으로 i+48, i+68과 같이 값을 대입하였고, 밑 while문으로 값을 원하는 만큼 배열 C를 만들어 출력함. |
| #과제 2번  int main()  {  char \*cp;  char buffer;  int array\_size;  printf("할당하려는 크기 : ");  scanf("%d", &array\_size);  while (buffer = getchar() != '\n' && buffer != EOF);  cp = (char\*)calloc(array\_size +1, sizeof(char));  printf("출력히려는 문장 : ");  scanf("%[^\n]s", cp);  printf("출력확인 : %s\n",cp);  free(cp);  }  Char형 배열로 만들었고, 배열크기를 입력받고, 입력된 수만큼 배열을 생성하여 값을 출력.  Array\_size +1 만큼 할당한 것은 널문자를 고려하기 위함이고, calloc으로 배열을 초기화하며 만든 이유도 위와 같은 이유이다. |

|  |
| --- |
| #과제 3번  int select\_num(int \*ip, int n) //지뢰가 아닌경우 주변 cnt++ 해주는 함수.  {  int \*\*ip2;  int cnt = 0;  ip2 = (int \*)malloc(n \* sizeof(int\*));  for (int i = 0; i < n; i++)  {  ip2[i] = malloc(sizeof(int) \* n);  }  for (int k = 0, i = 0; i < n; i++)  {  for (int j = 0; j < n; j++)  {  ip2[i][j] = ip[k];  k++;  }  }  for (int k = 0, i = 0; i < n; i++)  {  for (int j = 0; j < n; j++)  {  if (!(ip2[i][j])) //별이 아니라면,  {  cnt = 0;  int flag[4] = { 1,1,1,1 }; //[0]x\_left, [1]x\_right, [2]y\_up, [3]y\_down  //flag를 두어, 모서리인지 맨윗줄인지 등을 검사  if (i - 1 < 0)  {  flag[2] = 0;  }  if (i + 1 >= n)  {  flag[0] = 0;  }  if (j - 1 < 0)  {  flag[3] = 0;  }  if (j + 1 >= n)  {  flag[1] = 0;  }  // 주변 8개 모두를 검사하고, cnt++을 한다  if (flag[2] && (ip2[i - 1][j] == '\*'))  cnt++;  if (flag[0] && (ip2[i + 1][j] == '\*'))  cnt++;  if (flag[3] && (ip2[i][j - 1] == '\*'))  cnt++;  if (flag[1] && (ip2[i][j + 1] == '\*'))  cnt++;  if (flag[0] && flag[3] && (ip2[i + 1][j - 1] == '\*'))  cnt++;  if (flag[2] && flag[1] && (ip2[i - 1][j + 1] == '\*'))  cnt++;  if (flag[2] && flag[3] && (ip2[i - 1][j - 1] == '\*'))  cnt++;  if (flag[0] && flag[1] && (ip2[i + 1][j + 1] == '\*'))  cnt++;  //ip[k]에 바로 값을 넣어준다.  ip[k] = cnt;  }  k++;  }  }  free(ip2); // 이차원배열로 보기 편하게 사용했던 ip2는 해제한다.  }  Flag 배열를 만들고, 2차원 배열을 새롭게 할당 받았으며, flag에 따라 각각에 해당하는 if문이 달라짐으로 써, 주변에 지뢰가 몇 개인지 값을 정하는 형태.  random값으로 모든 지뢰를 자동으로 들어가게 해놓았다. |

1. 실행화면
   1. 과제 1번

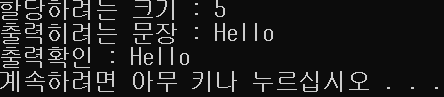


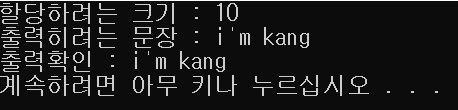




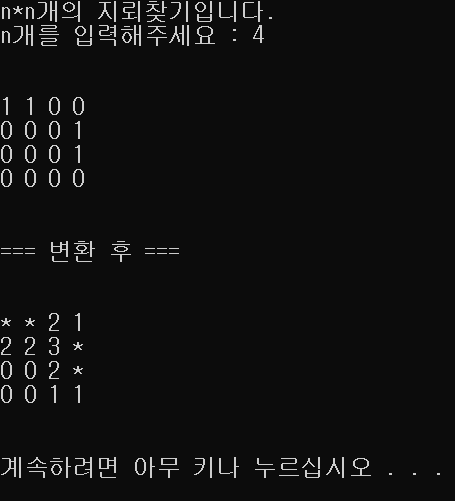


* 1. 과제 2번





* 1. 과제 3번





1. 고찰

1번 과제는 예전에 비슷한 과제로 했었던 코드에서 동적할당으로 배열을 선언한다라는 차이점이 있었고, 2번 과제는 동적할당으로 배열만 만들어주면 되는 거라 수월했음. 단 널 문자를 고려해주지 않아 보고서를 수정하는 사태를 겪음. 가장 어렵다고 느낀 지뢰찾기의 경우 일일히 지뢰를 입력받으면 게임하는 느낌이 없다는 점과, 번거로움이 발생하여 랜덤값을 주어, 프로그램이 실행할 때마다 값이 바뀌게 설정함. 그렇게해서 지뢰를 출력하는 데에서는 큰 문제가 없었지만 if문을 여러 개사용하여 지뢰주변에 숫자를 나타내는 점에서 if문을 여러 개 사용하고 싶지가 않아서 반복문으로 구성을 하려다가 결국 실패함. if문을 12개 정도사용하니 연산이 많이 들어가는 프로그램으로써, 좋지 않은 효율을 가지고 있고, 코드의 흐름을 한눈에 보기에 불편한 점이 있다고 자각. 이번 과제에 자가채점을 매겨본다면, 만족스럽지 못한 코딩이였다고 판단.