

11주차 과제					
학년	2	학번	2016707079	이름	하상천

1. 과제 설명

원하는 크기의 홀수 n 을 입력 받은 후 동적 할당을 사용해서 $n \times n$ 배열을 만든 후에 아래의 그림처럼 1부터 n^2 까지 숫자를 규칙에 맞게 차례대로 채우는 코드를 만든다.

```
원하는 크기를 입력 하시오
7
34 35 36 37 38 39 4
33 46 47 48 40 3 5
32 45 49 41 2 19 6
31 44 42 1 18 20 7
30 43 26 17 25 21 8
29 27 16 24 23 22 9
28 15 14 13 12 11 10
```

2. 과제 이론

If문

조건을 주어 그 조건을 만족할 때에 실행하도록 한다.

Ex) if($i==1$) 이면 i 의 값이 1일 때만 실행하도록 한다.

그 외에는 다른 실행을 하고 싶다면 똑같은 형식으로 else if 또는 else로 조건을 주면 된다.

For문

일정한 조건을 주어서 원하는 만큼 반복할 수 있게 해주는 반복문

Ex) for($i=0; i<10; i++$); ---> $i=0$ 으로 시작하고 한번 반복 할 때마다 i 값이 1씩 증가한다.

그리고 $i<10$ 이라는 조건이 만족 할 때 까지만 반복한다. 즉 $i=10$ 이되면 반복하지 않는다.

동적 할당함수(malloc, free)

메모리를 동적으로 할당하기 위해서는 함수를 호출해야 함.

```
Void *malloc(unsigned int);
```

Int형 변수로 사용할 기억공간을 할당 받는 경우 int *ip //할당 받은 기억공간을 가리킬 포인터 변수

<stdlib.h>라이브러리가 필요하다.

Break문

Break를 만나면 가까운 반복문을 벗어난다.

3. 주요 소스 설명

```
#include<stdio.h>
```

```
#include<stdlib.h>//malloc 함수 사용
```

```
int main()
```

```
{
```

```
    int num, i, j;
```

```
    int x, y = 0;
```

```
    printf("원하는 크기를 입력하세요(홀수):");
```

```
    scanf("%d", &num);
```

```
    int **a; //이중포인터 선언
```

```
    a = (int**)malloc(num * sizeof(int*)); //기억공간을 동적으로 할당
```

```
    for (int i = 0; i < num; i++) {
```

```
        *(a + i) = (int *)malloc(num * sizeof(int));
```

```
    }
```

```
    int c = 1; //초기값 1
```

```
    for (i = num / 2 , j = num / 2 ; j < num; i--, j++) //배열의 가운데부터 대입하다가
```

j가 num보다 커지거나 같으면 반복문을 마무리한다.

```
{  
  
    a[i][j] = c;  
  
    c++;  
  
}
```

i++; // 위에 for문에서 마지막으로 i--, j++을 하고 for문을 끝내기 때문에 원래 상태로 되돌리기 위해 i++, j-- 를 한다.

```
j--;
```

for (x = 0, y = num; ; x += 2, y--) //한쪽 방향은 1씩 감소, 다른쪽 방향은 2씩 감소하기 때문

```
{  
  
    i++; //한칸 내리고 또 대입한다.
```

for (; i < y; i++) // 한칸씩 내리면서 대입하고 i가 y보다 크거나 같으면 반복문을 마무리한다.

```
{  
  
    a[i][j] = c;  
  
    c++;
```

if (c == (num*num) / 2 + 2) // c가 (num*num) / 2 + 2이면 가까운 반복문을 탈출한다.

```
{  
  
    break;  
  
}
```

```
}
```

if (c == (num*num) / 2 + 2) //c가 (num*num) / 2 + 2이면 가까운 반복문을 탈출한다.

```
{  
  
    break;  
  
}
```

i--; //원래상태로 복구

j--; //한칸 왼쪽으로 이동하여 대입한다.

for (; x < j; j--)// 한칸씩 왼쪽으로 이동하면서 대입하고 x가 j보다 크거나 같으면 반복문을 마무리한다.

```
{  
  
    a[i][j] = c;  
  
    c++;
```

if (c == (num*num) / 2 + 2) //c가 (num*num) / 2 + 2이면 가까운 반복문을 탈출한다.

```
{  
  
    break;  
  
}
```

```
}
```

if (c == (num*num) / 2 + 2)//c가 (num*num) / 2 + 2이면 가까운 반복문을 탈출한다.

```
{  
  
    break;
```

j += 2; //원래상태로 복구하고 한칸 오른쪽으로 이동하고 한칸 위로 이동하여 대입한다.

i--;

for (; j < y - 1; i--, j++) // 오른쪽으로 한칸씩 이동하고 위로 한칸 씩 이동하여 대입하고 j가 y-1보다 크거나 같으면 반복문을 마무리한다.

{

a[i][j] = c;

c++;

if (c == (num*num) / 2 + 2) //c가 (num*num) / 2 + 2이면 가까운 반복문을 탈출한다.

{

break;

}

}

if (c == (num*num) / 2 + 2) //c가 (num*num) / 2 + 2이면 가까운 반복문을 탈출한다.

{

break;

}

i++; //원래상태로 복구하고 한칸 아래쪽으로 이동하여 대입한다.

j--;

}

for (i = num / 2 + 1, j = num / 2 - 1; j >= 0; i++, j--) // 1을 쓴 칸에서 아래로 한칸, 왼쪽으로 한칸가서 대입하기 시작한다.

{

a[i][j] = c;

c++;

```
}
```

i--; //i++, j--를 하고 반복문을 마무리 하기 때문에 원래 상태로 복구하기 위하여 i--, j++을 해준다.

```
j++;
```

for (x = 0, y = num; ; x++, y -= 2) //한쪽은 한칸씩 감소하고, 다른 한쪽은 두칸씩 감소한다.

```
{
```

```
    i--; // 한칸 위로 올려주고 대입을 시작한다.
```

for (; i >= x; i--) //위로 한칸씩 올려주면서 대입하고 i가 x보다 작아질 때 까지 반복한다.

```
{
```

```
    a[i][j] = c;
```

```
    c++;
```

if (c == num * num + 1) //마지막 num * num 까지 배열에 대입 해주고 c++을 했으므로 c가 num * num + 1 이면 반복문을 마무리한다.

```
{
```

```
    break;
```

```
}
```

} i++; //i--하고 반복문을 마무리 했으므로 원래 상태로 복구해주고 오른쪽으로 한칸 이동해준다.

```
j++;
```

if (c == num * num + 1) //마지막 num * num 까지 배열에 대입해주고 c++을 했으므로 c가 num * num + 1 이면 반복문을 마무리한다.

```
{
```

```
    break;
```

```
}
```

for (; j < y - 1; j++) //오른쪽으로 한칸 씩 이동하면서 대입해주고 j가 y-1과 같거나 커질 때까지 반복한다.

```
{
```

```
    a[i][j] = c;
```

```
    c++;
```

```
    if (c == num * num + 1) {
```

```
        break;
```

```
    }
```

```
}
```

```
if (c == num * num + 1) {
```

```
    break;
```

} j -= 2; //원래상태로 복구해주고 아래로 한칸, 왼쪽으로 한칸 가주고 대입을 시작한다.

```
i++;
```

for (; i < y - 2; i++, j--) //아래로 한칸, 왼쪽으로 한칸씩 이동하면서 대입하고 i가 y-2와 같거나 커질 때까지 반복한다.

```
{
```

```
    a[i][j] = c;
```

```
    c++;
```

```
    if (c == num * num + 1) {
```

```
        break;
```

```
    }
```

```
}
```

```

        if (c == num * num + 1) {

            break;

        }i--; //원래상태로 복구해준다.

        j++;

    }

    for (int i = 0; i < num; i++) //배열 출력

    {

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

        {

            printf("%5d", a[i][j]); //여유있게 %5d로 5칸씩 주었다. a[i][j] 대
신에 (*(a+i)+j) 이렇게 표현해도 된다.

        }printf("\n");

    }

    for (int i = 0; i < num; i++) //할당 받은 메모리 반환

    {

        free(*(a + i)); //free(a[i]) 이렇게 표현해도 된다.

    }

    free(a);

}

```


4. 실행화면

Microsoft Visual Studio 디버그 콘솔

```
원하는 크기를 입력하세요(홀수):5
19 20 21 22 3
18 25 23 2 4
17 24 1 12 5
16 14 11 13 6
15 10 9 8 7

C:\Users\seomk\source\repos\11Homework\
이 창을 닫으려면 아무 키나 누르세요.
```

Microsoft Visual Studio 디버그 콘솔

```
원하는 크기를 입력하세요(홀수):7
34 35 36 37 38 39 4
33 46 47 48 40 3 5
32 45 49 41 2 19 6
31 44 42 1 18 20 7
30 43 26 17 25 21 8
29 27 16 24 23 22 9
28 15 14 13 12 11 10

C:\Users\seomk\source\repos\11Homework\Deb
이 창을 닫으려면 아무 키나 누르세요.
```

Microsoft Visual Studio 디버그 콘솔

```
원하는 크기를 입력하세요(홀수):21
251 252 253 254 255 256 257 258 259 260 261 262 263 264 265 266 267 268 269 270 11
250 305 306 307 308 309 310 311 312 313 314 315 316 317 318 319 320 321 271 10 12
249 304 350 351 352 353 354 355 356 357 358 359 360 361 362 363 322 272 9 68 13
248 303 349 386 387 388 389 390 391 392 393 394 395 396 364 323 273 8 67 69 14
247 302 348 385 413 414 415 416 417 418 419 420 397 365 324 274 7 66 116 70 15
246 301 347 384 412 431 432 433 434 435 421 398 366 325 275 6 65 115 117 71 16
245 300 346 383 411 430 440 441 436 422 399 367 326 276 5 64 114 155 118 72 17
244 299 345 382 410 429 439 437 423 400 368 327 277 4 63 113 154 156 119 73 18
243 298 344 381 409 428 438 424 401 369 328 278 3 62 112 153 185 157 120 74 19
242 297 343 380 408 427 425 402 370 329 279 2 61 111 152 184 186 158 121 75 20
241 296 342 379 407 426 403 371 330 280 1 60 110 151 183 206 187 159 122 76 21
240 295 341 378 406 404 372 331 281 222 59 109 150 182 205 207 188 160 123 77 22
239 294 340 377 405 373 332 282 223 58 108 149 181 204 218 208 189 161 124 78 23
238 293 339 376 374 333 283 224 57 107 148 180 203 217 219 209 190 162 125 79 24
237 292 338 375 334 284 225 56 106 147 179 202 216 221 220 210 191 163 126 80 25
236 291 337 335 285 226 55 105 146 178 201 215 214 213 212 211 192 164 127 81 26
235 290 336 286 227 54 104 145 177 200 199 198 197 196 195 194 193 165 128 82 27
234 289 287 228 53 103 144 176 175 174 173 172 171 170 169 168 167 166 129 83 28
233 288 229 52 102 143 142 141 140 139 138 137 136 135 134 133 132 131 130 84 29
232 230 51 101 100 99 98 97 96 95 94 93 92 91 90 89 88 87 86 85 30
231 50 49 48 47 46 45 44 43 42 41 40 39 38 37 36 35 34 33 32 31

C:\Users\seomk\source\repos\11Homework\Debug\11Homework.exe(4536 프로세스)이(가) 0 코드로 인해 종료되었습니다
```

5. 고찰

이번 과제기간을 2주동안 주셔서 곰곰이 많은 생각을 해보았다. 우선 이중 포인터를 선언하고 malloc 함수를 이용해서 메모리를 동적으로 할당 받았다. 그 다음 단일 포인터에 메모리를 동적으로 할당 받았다. 처음에는 여러가지 접근을 해보았다. 대각선으로 줄어드는 숫자의 크기,

가로로는 몇 개가 증가하고부터 숫자가 순서대로 진행되는지, 세로로 늘어나는 숫자의 크기 등 여러가지 많은 방법으로 하루동안 접근을 시도해보았지만 너무 복잡하고 정리하기가 어려워 다른 방법을 찾아 보았다. 숫자가 증가하는 방식을 쪼개어 보면 세가지 방법으로 나눌 수 있다. 대각선으로 증가하는 방법, 세로로 증가하는 방법, 가로로 증가하는 방법이다. 처음에 1부터 대각선으로 증가하는 것은 반만 증가하기 때문에 따로 for문을 통해 증가시켰고 그 이후에는 3가지 방법의 for문을 또 다른 for문으로 묶어서 증가 시켰다. 3가지 방법을 진행할 때마다 한 쪽은 1씩 감소하고 반대편 쪽은 2씩 감소하기 때문에 `for (x = 0, y = num; ; x += 2, y--)`를 사용하였다. 그리고 `for (; i < y; i++)`을 이용하여 세로로 증가하는 것을 대입시켰고, `for (; x < j; j--)`을 이용하여 가로로 증가하는 것을 대입시켰고, `for (; j < y - 1; i--, j++)`을 이용하여 대각선으로 증가하는 것을 대입시켰다. 그리고 `if (c == (num*num) / 2 + 2)` 이면 `break;` 을 이용하여 오른쪽 아래 삼각형을 마무리 하였다. 마찬가지로 방법으로 대각선 반만 증가하는 것은 따로 for문을 이용하였고 세가지 방법으로 나뉘는 것은 똑같이 for문 4개를 이용하여 표현하였다.

그리고 `num * num` 까지 배열에 대입해주고 `c++`을 했으므로 `if (c == num * num + 1)` 이면 `break;`를 통하여 반복문을 벗어난다. 배열 출력은 여유있게 `%5d`를 이용하여 5칸을 주었다. `a[i][j]`로 출력해도 되지만 `*(a+i+j)`로 출력해도 된다. 단일 포인터로 할당 받은 메모리는 `for (int i = 0; i < num; i++) { free(*(a + i)); }` 를 이용해서 반환 해준다. `free(*(a+i))`는 `free(a[i])` 이렇게 표현해도 된다. 마지막으로 `free(a)`를 이용하여 이중 포인터로 할당 받은 메모리를 반환한다. 처음에는 포인터에 대한 두려움도 있었고, 동적할당에 대한 두려움도 있었다. 하지만 이번 과제를 통해서 포인터와 배열은 같다는 것을 배우게 되었다. 그 결과 포인터에 대한 두려움이 많이 없어지고 배열처럼 친숙하게 느껴졌다. 그리고 `malloc`함수를 배우게 되어서 동적할당의 의미도 정확하게 알게 되었고, `free` 함수를 통해 동적할당 받은 메모리는 꼭 반환해주어야 된다는 것도 알게 되었다. 이중 포인터를 까먹지 않기 위해서 또 다른 문제를 풀어보고 복습을 해야겠다.