강의명: 프로그래밍
실습 번호: 6
실습 제목: Pointers and arrays(포인터 및 배열)
학생 이름: 임지빈
학번: 202211051

1. 함수의 배열 인수
1.1
```
void init(int a[][6], int n)
{
 int i,j;
 for (i=0; i<n; i++)
   for (j=0; j<6; j++)
     a[i][j]=i+j;
}
```
1.2
```
void print(int a[][6], int n)
{
 int i, j;
 printf("{");
 for (i=0;i<n;i++){
   printf("{");
   for (j=0; j<6; j++)
     printf("%d%s", a[i][j],j==5 ? " " :
","  );
   printf("}%s", i==3 ? "" : ",");
 }
 printf("};\n");
}
```
시작과 끝에 괄호를 출력한다.

J가 0,1,2,3,4일때는 배열이 끝나지 않았기 때문에 “,”를 써줘야하고 5이면 괄호를 닫기 때문에 조건 연산자를 사용하여 출력한다.
I도 마찬가지로 0,1,2일때는 “,”를 써줘야하고 4이면 배열이 끝나기 때문에 조건 연산자를 사용하여 출력한다.

1.3

```
s2211051@oak:lab06$ gcc array-para.c -o array-para
s2211051@oak:lab06$ ./array-para
matrix[4][6]={{0,1,2,3,4,5 },{1,2,3,4,5,6 },{2,3,4,5,6,7 },{3,4,5,6,7,8 }};
```

2. 변수의 주소

2.1

```
int main(void)
{
 int i;
 float f;
 int m[4][6];

 printf("address of i=%p\n", &i);
 printf("address of f=%p\n", &f);
 printf("address of m=%p\n", &m);
 printf("address of m[0][2]=%p\n",&m[0]
[2]);

 return EXIT_SUCCESS;
}
```

변수의 주소를 출력하려면 주소 연산자 &를 사용하여 표현한다.

2.2

```
s2211051@oak:lab06$ gcc var-addr.c -o var-addr
s2211051@oak:lab06$ ./var-addr
address of i=0x7ffdde4b2b68
address of f=0x7ffdde4b2b6c
address of m=0x7ffdde4b2b70
address of m[0][2]=0x7ffdde4b2b78
```

3. 함수의 포인터 인수

3.1

Int x, int y는 바뀌지만 변수 a, b 값은 사라져버리기 때문에 swap하지못한다.

3.2

```
void swap2(int *px, int *py)
{
 int temp;

 temp=*px;
 *px=*py;
 *py=temp;
}
```

포인터에 가리키는 주소의 값을 나타내는 참조 연산자 *를 사용하였다.

3.3

```
s2211051@oak:lab06$ gcc swap2.c -o swap2
s2211051@oak:lab06$ ./swap2
swap2: a=10, b=20 => a=20, b=10
s2211051@oak:lab06$ gcc swap1.c -o swap1
s2211051@oak:lab06$ ./swap1
swap1: a=10, b=20 => a=10, b=20
```

4. 주소 연산

4.1

```
s2211051@oak:lab06$ gcc addr-arith.c -o addr-arith
s2211051@oak:lab06$ ./addr-arith
&a[0][0]=0x601060, &a[0][1]=0x601064, &a[0][1]-&a[0][0]
=1
&a[0][0]=0x601060, &a[1][0]=0x601070, &a[1][0]-&a[0][0]
=4
a+1=0x601070, a+2=0x601080, (a+2)-(a+1)=1
```

&a[0][0]의 주소를 출력하고, &a[0][1]의 주소를 출력했고, 둘이 뺄셈을 하게 되면 2-1 으로 1이다.

&a[0][0]의 주소를 출력하고, &a[1][0]의 주소를 출력했고, 둘이 뺄셈을 하게 되면 5-1 으로 4이다.

a+1의 주소는 두번째 element의 주소이고, a+2의 주소는 3번째 element의 주소이다. 두개를 빼게 되면 integer 4개 있는 array가 한

개가 있기 때문에 1이 나온다.

5. 배열의 sizeof 연산

5.1

```
s2211051@oak:lab06$ gcc sizeof-array.c -o sizeof-array
s2211051@oak:lab06$ ./sizeof-array
sizeof(a[2][3])=4
sizeof(a[1])=16
sizeof(a)=48
```

a[2][3]는 int형이므로 4이다.

a[1]는 int형 변수 4개이므로 4*4=16이다.

a의 크기는 int형 변수 12개이므로 4*12=48이다.

6. 스트링의 길이

6.1

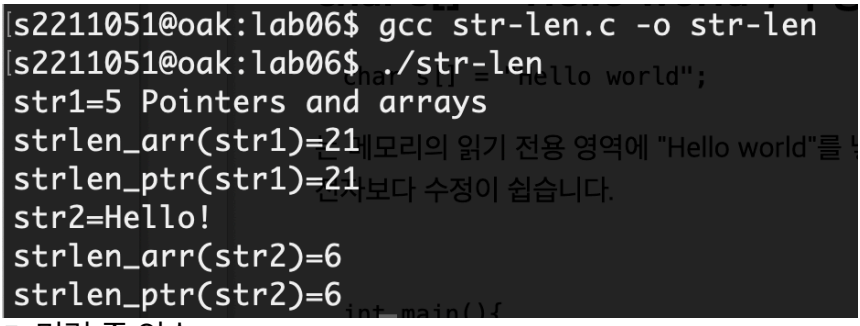
```
int strlen_arr(char s[])
{
 int a;
 for (a=0; s[a] != '\0'; a++){
 }
 return a;
}
```

A를 0부터 시작하여S[a]을 null 전까지만 표시하도록 출력한다.

6.2

```
int strlen_ptr(char *s)
{
 int n;
 for (n=0; *s != '\0'; s++)
   n++;
 return n;
}
```

6.3



7. 명령 줄 인수

7.1

```
int main(int argc, char *argv[])
{
 if (argc != 3)
 {
   printf("arguments error!\n");
 } else

 printf("%s+%s=%d\n",argv[1],argv[2],atoi(a
rgv[1])+\
atoi(argv[2]));

 return EXIT_SUCCESS;
}
```

Argc 메인함수에 전달되는 정보의 갯수가 3개가 아니라면
arguments error!을 출력한다.
그것이 아니라면 주어진 인수들을 더하는 값을 출력한다.
Char*을 int로 바꿔서 계산해야하기 때문에
atoi(argv[1])+atoi(argv[2])를 출력한다.

7.2

```
[s2211051@oak:lab06$ gcc plus.c -o plus
[s2211051@oak:lab06$ ./plus
arguments error!
[s2211051@oak:lab06$ ./plus 10 20
10+20=30
[s2211051@oak:lab06$ ./plus 20 30
20+30=50
[s2211051@oak:lab06$ ./plus 10 20 30
arguments error!
```