20180226
4 일차
신민철

static 을 사용하면 배열을 리턴할 수 있다.

static 변수는 전역 변수와 마찬가지로 data 영역에
지역 변수를 static 으로 선언했다면
이 변수를 선언한 함수내에서만 접근 가능함
정적변수가 되기 때문.
*.c 모든이름의 c 파일 cp -r 디렉토리 복사 *은 모든파일`:

continue 는 밑에라인을 무시하고 함수다음것을 실행함.
break 은 함수탈출
dowhile 은 kernal 매크로에 사용.

scope 란 { } 영역이다

```
int main(void)
{
        int i, result;

        for(i = 0, result = 'A'; i < 10; i++, result++)
        이렇게도 사용이 가능함.


}
int main(void)
{
int i, result = 'A';
for( ; ; )  //for 세미콜론 가운데부분만 공백이라면 무한루프가 형성됨.
{
        printf("%c\n",result);
}
return 0 ;
}
```

goto 는 if 문을 많이 사용하면 CPU 에 치명적이기 때문에 그것을 감소시키기 위해서
goto 로 한번에 빠져나간다

```
#include<stdio.h>

int main(void)
{
      int i, j, k, flag = 0;
      for(i = 0; i < 10; i++)
```

```
	{
		for(j = 0; j < 10; j++)
		{
			for(k = 0 ; k < 10; k++)
			{
				if((i==2) && (j ==2) &&(k == 2))
				{
					printf("Error!!!\n");
					flag = 1;
				}
				else
				{
					printf("Data\n");
				}
				if(flag)
				{
					break;
				}
			}
			if(flag)
			{
				break;
			}
		}
		if(flag)
		{
			break;
		}
	}
}
```

이렇게 쓰면 if 와 break 을 많이쓰기 때문에 cpu 에 좋지 않으므로

```
#include<stdio.h>

int main(void)
{
	int i, j, k, flag = 0;
	for(i = 0; i < 10; i++)
	{
		for(j = 0; j < 10; j++)
		{
			for(k = 0 ; k < 10; k++)
			{
				if((i==2) && (j ==2) &&(k == 2))
				{
					printf("Error!!!\n");
					goto err_handler;
				}
				else
				{
					printf("Data\n");
```

```
                        }
                    }
                }
        }
        return 0;
err_handler:
        printf("Goto Zzang!\n");
        return -1;
}
```

이렇게 goto 를 사용해서 한방에 에러가나면 탈출할 수 있다.

파이프라인은 기본적으로 3 개 동작하는데
첫째로 fetch 둘째로 decode 셋째로 excute 한다.
fetch 는 명령어를 가져오는거고 decode 는 명령어를 해석하는것이고 excutie 는 실행하는것이다
3 단계는 예를들어 5 개의명령어가 있으면
첫번째 단계에 fetch 는 첫번째 명령어만 하고 두번째에 첫번째 fetch 한것이 decode 하고
두번째 명령어에 나오는 파이프가 fetch 한다 그리고 세번째에 첫번째파이프가 excute 하고
두번째 명령어가 decode 를 하게되며 세번째 명령어가 fetch 된다째
그리고 네번째에 나오는 명령어가 fetch 하게되고 두번째가 excute 한다.
세번째명령어는 decode 를하게된다. 그리고 다섯번째 명령어가 fetch 하게되면
네번째 명령어는 decode 를하게되고 세번째 명령어는 excute 를 하게된다.
그리고 나머지 명령어들이 순차적으로 fetch decode excute 한다.

피보나치 재귀함수 호출방식은
먼저 첫번째로 왼쪽부터 있는 함수들을 호출하고 그 함수들이 리턴을 받으면 리턴받은 함수 바로 위에 있는 오른쪽 함수를 실행한다. 거기서 또 왼쪽에 있는 함수부터 실행하고 리턴하면 오른쪽 함수를 실행하고 그 함수가 리턴하면 왼쪽과 오른쪽의 함수 연산을 리턴하고 그 위에있는 오른쪽 함수를 실행한다. 그림으로 나타내자면 다음과 같다.

Main Res 6 return 8	Fin 6	
Fib (6) fib(5)+fib(4) return 5+3=8		
Fib (5) fib(4)+fib(3) return 3+2=5	Fib (4) fib(3)+fib(2) return 2+1=3	
Fib (4) fib(3)+fib(2) return 2+1= 3	Fib (3) fib(2) + fib (1) return 1+1=2	Fib (2) return 1
Fib (3) fib(2)+fib(1) return1+1=2	Fib (2) return 1	Fib (1) return 1
Fib (2) return 1	Fib (1) return 1	