|  |  |
| --- | --- |
|  | 공과대학  컴퓨터공학부 |

**(C 프로그래밍) 실습보고서**

|  |  |  |  |
| --- | --- | --- | --- |
| **분반** | 0438 | **수행일자** | **2023. 5 . 10 . ( 10 주차)** |
| **학번** | 202311254 | **이름** | 권동희 |

**1. 실습 제목: Programming practice1**

**2. 소스 코드**

#include <stdio.h>

void up\_and\_down(int n);

int main(void) {

up\_and\_down(1);

return 0;

}

void up\_and\_down(int n)

{

printf("Level %d before a recursion\n", n);

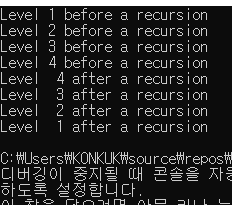
if (n < 4)

up\_and\_down(n + 1);

printf("Level % d after a recursion\n", n);

}

**3. 수행 결과 캡쳐**

****

**4. 의견**

up\_and\_down 함수가 호출된 후, 1,2,3,4 순서대로 함수에 들어간 뒤, 4,3,2,1 순서로 종료가 된다.

재귀호출이 되었을 때 함수가 종료되는 것이 아닌 대기 상태인 것과, 함수 호출 순서가 다른 것에 유의해야 할 것 같다.

**1. 실습 제목: Programming practice2**

**2. 소스**

#include <stdio.h>

#include <time.h>

int fib(int n);

int fib\_iter(int n);

int fib(int n)

{

if (n == 0)

return 0;

if (n == 1)

return 1;

return (fib(n - 1) + fib(n - 2));

}

int fib\_iter(int n)

{

if (n < 2)

return n;

else

{

int i, tmp, f1 = 1, f0 = 0;

for (i = 2; i <= n; i++)

{

tmp = f1;

f1 += f0;

f0 = tmp;

}

return f1;

}

}

#include"my.h"

int main(void)

{

int n;

time\_t t1, t2;

printf("정수 입력하세요: ");

scanf\_s("%d", &n);

t1 = time(NULL);

printf("피보나치수: %d\n", fib\_iter(n));

t2 = time(NULL);

printf("반복문 시간: %d\n", t2 - t1);

t1 = time(NULL);

printf("피보나치수: %d\n", fib(n));

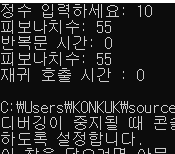
t2 = time(NULL);

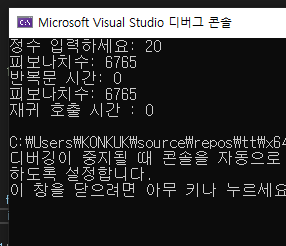
printf("재귀 호출 시간 : %d\n", t2 - t1);

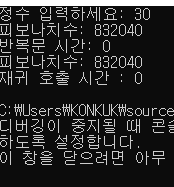
return 0;

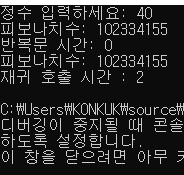
}

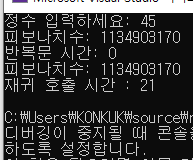
**3. 수행 결과 캡쳐**

****

****

****

****

****

**4. 의견**

수행시간이 차이나는 이유:

재귀함수 안에서 재귀함수 호출이 반복적으로 일어나고 있다. 재귀함수의 시간 복잡도와, 계산 비용이 증가하게 된다. 그렇기 때문에 위 코드에서 반복문보다 재귀함수를 사용했을 때 수행시간이 늘어나게 된다.

**1. 실습 제목: Lab Assignment1**

**2. 소스 코드**

#include<stdio.h>

void reverse();

int main()

{

printf("String:");

reverse();

return 0;

}

void reverse()

{

char ch;

ch = getchar();

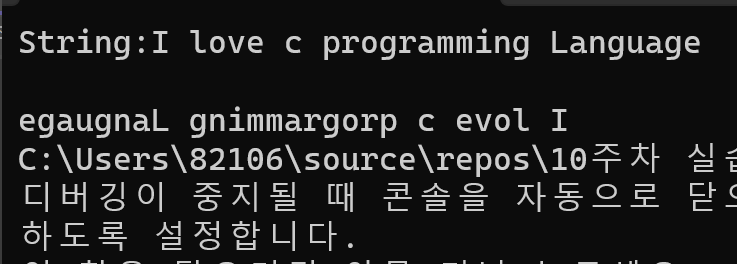
if (ch != '\n')

reverse();

printf("%c", ch);

}

**3. 수행 결과 캡쳐**

****

**4. 의견**

**enter가 입력되기 전까지 재귀함수를 사용해 계속 입력을 받다가 enter가 입력된 이후, 함수 호출 순서와 반대되어 함수가 종료되면서 거꾸로 출력된다.**

**1. 실습 제목: Lab Assignment2**

**2. 소스 코드**

#include<stdio.h>

#include<time.h>

#include<stdlib.h>

int main()

{

srand((unsigned)time(NULL));

int i, j, x;

int s[5][5];

int sum , max, min;

for (j = 0; j < 5; j++) //s[j][i]

{

for (i = 0; i < 5; i++)

{

x = rand() % 100;

s[j][i] = x;

printf("%d ", s[j][i]);

}

printf("\n");

}

for (j = 0; j < 5; j++)

{

printf("[%d] 행, ", j);

sum = 0;

max = s[j][0];

min = s[j][0];

for (i = 0; i < 5; i++)

{

sum = sum + s[j][i];

if (max < s[j][i])

max = s[j][i];

if (min > s[j][i])

min = s[j][i];

}

printf("sum:%3d, ", sum);

printf("max:%3d, ", max);

printf("min:%3d, ", min);

printf("average:%5.1f\n", (double)sum / 5);

}

printf("\n");

for (i = 0; i < 5; i++)

{

printf("[%d]열, ", i);

sum = 0;

max = s[0][i];

min = s[0][i];

for (j = 0; j < 5; j++)

{

sum = sum + s[j][i];

if (max < s[j][i])

max = s[j][i];

if (min > s[j][i])

min = s[j][i];

}

printf("sum:%3d, ", sum);

printf("max:%3d, ", max);

printf("min:%3d, ", min);

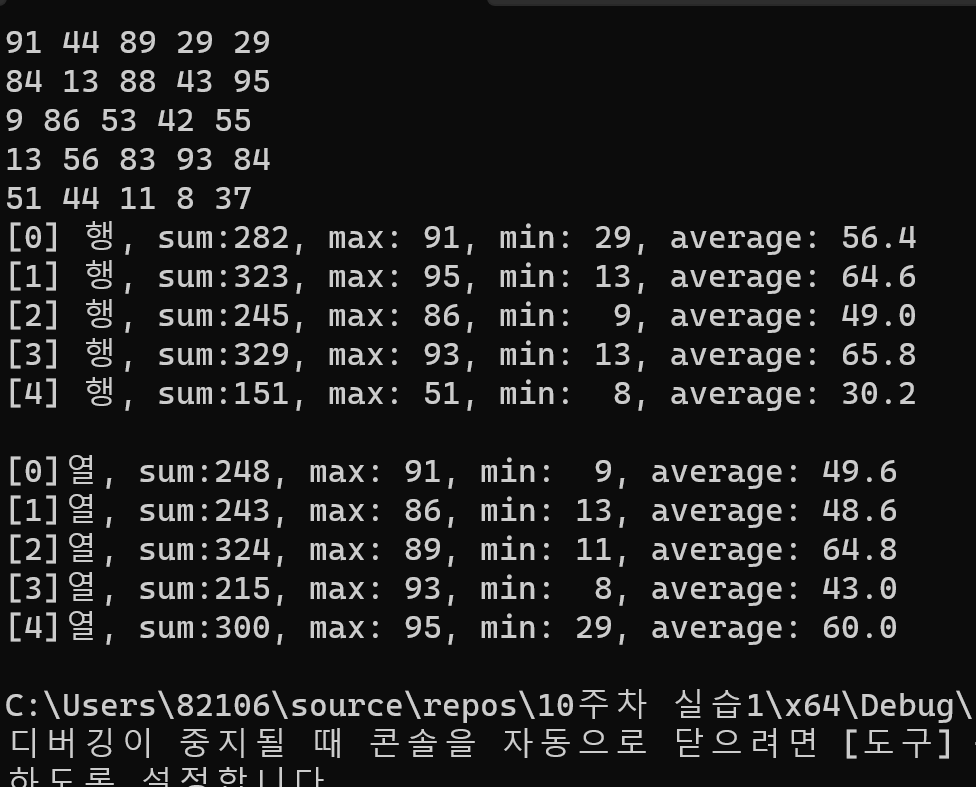
printf("average:%5.1f\n", (double)sum / 5);

}

return 0;

}

**3. 수행 결과 캡쳐**



**4. 의견**

중첩 for문과 rand()%100을 사용하여 0~99사이 정수를 랜덤하게 생성한 뒤 2차원 배열에 넣어주고, 중첩 for문을 사용하여 sum,max,min을 계산해준다. 평균을 계산하기 위해 sum을 형변환해서 나머지 첫 째 자리까지 출력해준다.