

객체지향 프로그래밍 1차 과제

2018202046 이준휘

1. Assignment 1

- 문제 설명

해당 문제는 1부터 50 사이의 수 n 를 입력을 받으면, n 번째 피보나치 수의 값을 출력하는 문제다.

- 해결 과정

Int 형 num 변수의 초기값을 -1로 두고 void Getnum(int& rnum) 함수를 통해 1~ 50 사이의 수를 입력을 받는다. 만약 입력을 받은 수가 해당 범위에 있는 수가 아닐 경우 system("cls") 명령어를 통해 화면을 지운 후 다시 입력을 받는다.


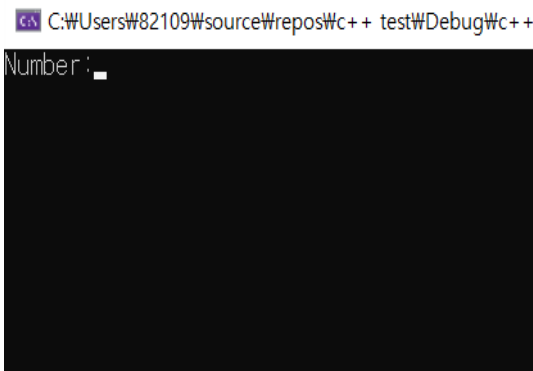
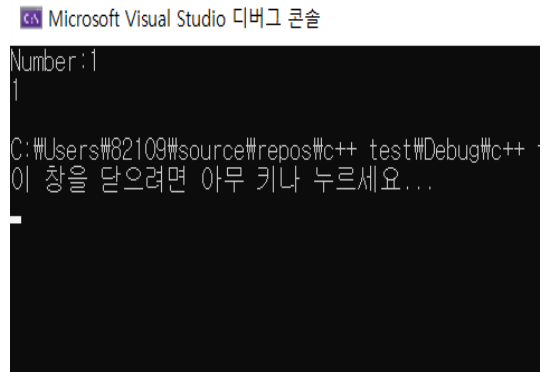
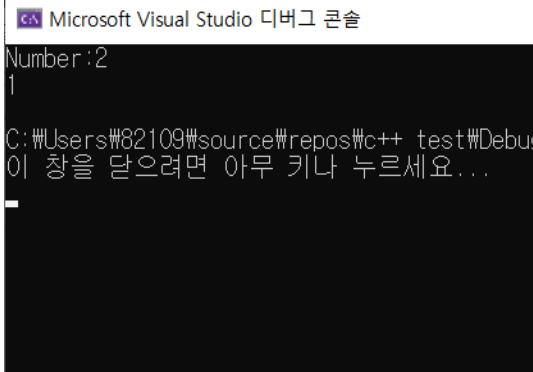
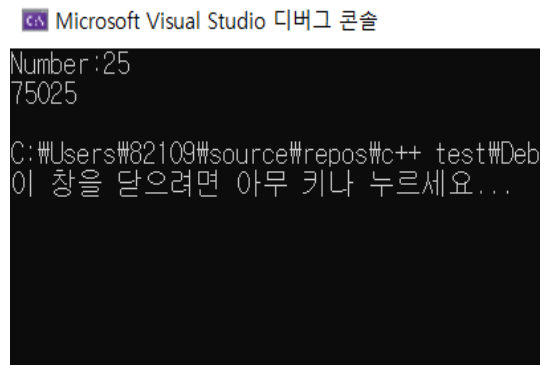
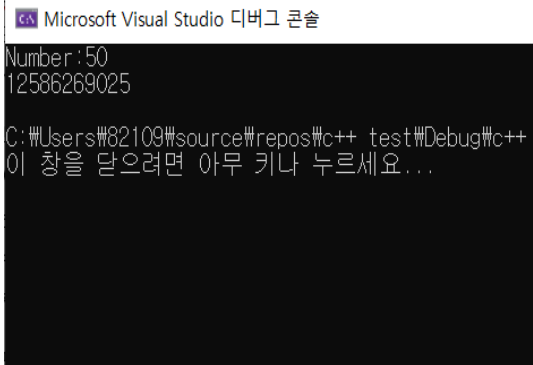
피보나치 수열은 long long int Fibo(int num) 함수를 통해 이루어진다. 해당 함수가 long long int 형으로 return 값을 갖는 이유는 int 형과 long int 형은 $2^{31} - 1$ 까지의 양수만을 표현할 수 있기 때문에 이보다 큰 값을 반환하기 위해서는 long long int 형으로 함수를 선언할 필요성이 있다.

피보나치 수열의 계산은 num 변수로 전달받은 입력값이 1 또는 2일 경우 1을 반환하고, 이 이상일 경우에는 long long int 형 변수 result, n1, n2로 계산을 실시한다. 변수 n1과 n2는 초기값 1을 갖고 있고 3번째 피보나치 수열부터 계산하는 것임으로 반복문에서 i 변수는 3부터 num과 같을 때까지 반복하게 된다. 반복되는 내용은 result의 값을 n1과 2의 합으로 저장하고 n2의 값을 n1으로 바꾸며, n1의 값을 result로 바꾼다.

해당 문제를 만약 함수를 재귀적으로 사용할 경우 비재귀적인 사용보다 중복된 연산이 더욱 많이 나오기 때문에 효율성이 떨어진다.

- 결과 화면

초기화면	잘못된 값을 입력
------	-----------

 <pre> C:\Users\82109\source\repos\c++ test\Debug\c++ te Number: </pre>	 <pre> C:\Users\82109\source\repos\c++ test\Debug\c++ Number: </pre>
Num = 1	Num = 2
 <pre> Microsoft Visual Studio 디버그 콘솔 Number:1 1 C:\Users\82109\source\repos\c++ test\Debug\c++ 이 창을 닫으려면 아무 키나 누르세요... </pre>	 <pre> Microsoft Visual Studio 디버그 콘솔 Number:2 1 C:\Users\82109\source\repos\c++ test\Debug\c++ 이 창을 닫으려면 아무 키나 누르세요... </pre>
Num = 25	Num = 50
 <pre> Microsoft Visual Studio 디버그 콘솔 Number:25 75025 C:\Users\82109\source\repos\c++ test\Debug\c++ 이 창을 닫으려면 아무 키나 누르세요... </pre>	 <pre> Microsoft Visual Studio 디버그 콘솔 Number:50 12586269025 C:\Users\82109\source\repos\c++ test\Debug\c++ 이 창을 닫으려면 아무 키나 누르세요... </pre>

- 고찰

초기 Fibo 함수를 int 형으로 반환값이 나오도록 설정하여 50을 입력하였을 때 출력되는 값이 제대로 나오지 않는 현상이 일어났다. 이 현상은 overflow 라고 불리며, 입력된 수가 자료형의 크기보다 클 경우 입력된 수의 일부분만 저장하면서 일어나는 현상이다. 필자는 이를 해결하기 위해 int 형보다 큰 값을 입력할 수 있는 long long int 형을 사용하였다. Long int 형을 사용하지 않은 이유는 int 형과 long int 형이 담을 수 있는 수의 크기가 같기 때문이다.

해당 문제를 풀면서 재귀적 용법과 비재귀적 용법을 사용할 때를 구분할 수 있는 기회였고, 피보나치 수열에 대해 복습할 수 있었던 과제였다.

2. Assignment 2

- 문제 설명

해당 문제는 1 ~ 9 사이의 사이즈를 입력을 받고 1로 초기값이 설정된 사이즈*사이즈 크기의 배열을 출력한다. 그리고 0 ~ 5 까지의 방향을 입력을 받으면 해당되는 방향으로 배열을 합하고 결과값을 출력한다.

- 해결 과정

우선 int 형 변수 size를 생성하고 초기값을 -1로 한다. 그리고 Getnum(int &num, int min, int max) 함수를 통하여 0보다 크고 10보다 작은 size를 받지 않을 경우 다시 수를 입력을 받도록 한다.

이후 정상적인 값이 입력받으면 int Makearr(int **rarr, int num)을 통하여 동적 할당을 통해 size*size 크기의 배열 arr를 만들고 각 배열의 값을 1로 초기화한다. 만약 동적할당이 제대로 동작하지 않을 경우 오류를 출력하고 main 함수를 종료한다.

그 후 2차원 배열을 출력하고, Getnum(int &num, int min, int max) 함수를 통해 0보다 크고 6보다 작은 수를 입력을 받는다. 입력받은 수는 int 형 변수 Movearr.dir 에 저장되고 해당 범위를 벗어나서 받을 경우 다시 수를 입력받도록 한다.

switch문을 통하여 Movearr.dir 의 값에 따라 아래와 같은 일을 수행한다.

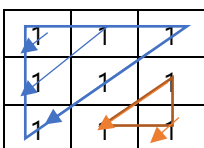
Movearr.dir 가 0일 경우 int Movearr.Up(int** &parr, int num) 함수를 수행한다. 해당 함수는 첫번째 행에 아래에 있는 행들의 값을 더하는 함수이다.

Movearr.dir 가 1일 경우 int Movearr.Down(int** &parr, int num) 함수를 수행한다. 해당 함수는 마지막 행에 위에 있는 행들의 값을 더하는 함수이다.

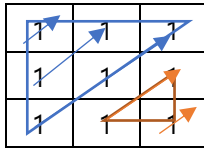
Movearr.dir 가 2일 경우 int Movearr.Right(int** &parr, int num) 함수를 수행한다. 해당 함수는 마지막 열에 좌측에 있는 열들의 값을 더하는 함수이다.

Movearr.dir 가 3일 경우 int Movearr.Left(int** &parr, int num) 함수를 수행한다. 해당 함수는 첫번째 열에 우측에 있는 열들의 값을 더하는 함수이다.

Movearr.dir 가 4일 경우 int Movearr.DownLeft(int** &parr, int num) 함수를 수행한다. 해당 함수는 아래 삼각형 형태로 나누어서 합산을 진행한다.



Movearr.dir 가 5일 경우 int Movearr.UpRight(int** &parr, int num) 함수를 수행한다. 해당 함수는 아래 삼각형 형태로 나누어서 합산을 진행한다.



해당 switch문을 완료한 후 바뀐 arr의 값을 void Printarr(int** parr, int num)를 통해 출력한다. 그 후 void Deletearr((int** parr, int num)를 통해 동적할당이 된 배열을 반환하는 것으로 프로그램이 마무리된다.

- 결과 화면

초기화면	Size 잘못된 값 입력
Size = 9	Movearr.dir 잘못된 값 입력
Movearr.dir = 0	Movearr.dir = 1

Movearr.dir = 2	Movearr.dir = 3
 <pre> Microsoft Visual Studio 디버그 콘솔 Array Size(0xNk10):8 1 Shift All Direction:2 0 C:\Users\#82109\source\repos\c++ test\Debug\c++ test.exe(프로세스 8856개)이(가) 이 창을 닫으려면 아무 키나 누르세요... </pre>	 <pre> Microsoft Visual Studio 디버그 콘솔 Array Size(0xNk10):9 1 Shift All Direction:3 0 C:\Users\#82109\source\repos\c++ test\Debug\c++ test.exe(프로세스 6632개)이(가) 이 창을 닫으려면 아무 키나 누르세요... </pre>
Movearr.dir = 4	Movearr.dir = 5
 <pre> Microsoft Visual Studio 디버그 콘솔 Array Size(0xNk10):7 1 Shift All Direction:4 0 7 6 5 4 3 2 1 C:\Users\#82109\source\repos\c++ test\Debug\c++ test.exe(프로세스 19064개)이(가) 이 창을 닫으려면 아무 키나 누르세요... </pre>	 <pre> Microsoft Visual Studio 디버그 콘솔 Array Size(0xNk10):6 1 Shift All Direction:5 0 C:\Users\#82109\source\repos\c++ test\Debug\c++ test.exe(프로세스 18976개) 이 창을 닫으려면 아무 키나 누르세요... </pre>

- 고찰

해당 과제를 진행할 때 필자는 공부하고 있던 클래스를 활용해보려고 하였다. 클래스 개념을 처음 적용해보았지만 구조체와 많이 다르지 않은 사용법에 금방 익힐 수 있었다. 함수를 만드는 과정에서 UpRight함수와 DownLeft함수를 어떻게 구현할 지 고민을 많이 했다. 그 과정에서 모든 부분을 한 번에 하지 않고 삼각형 2개로 분할하여 배열을 정리하는 법을 고안했다. 추후에 함수를 만들 때 이러한 분할을 잘 활용하면 유용할거라는 생각이 들었다.

3. Assignment 3

- 문제 설명

해당 문제는 1부터 10 사이의 정수 N을 받고 1~99 사이의 무작위 값을 가진 N*N 크기의 2차원 배열을 만든다. 그리고 이 배열을 정렬하여 출력을 하는 문제이다.

- 해결 과정

우선 int 형 변수 size를 호출하고, 해당 변수가 1~10 사이의 값을 받지 않으면 화면을

지우고 다시 입력을 받도록 한다.

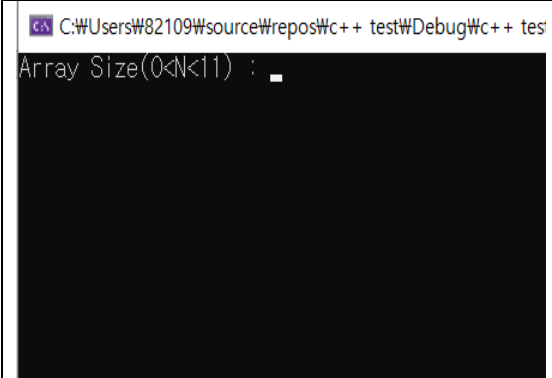
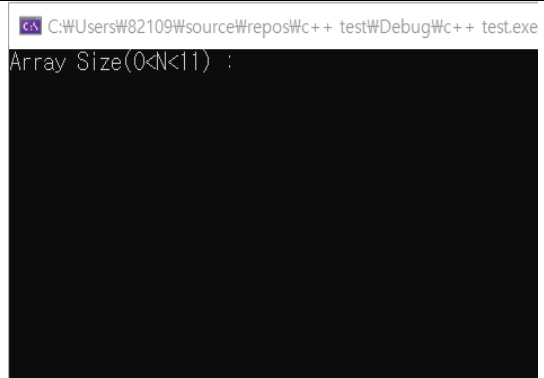
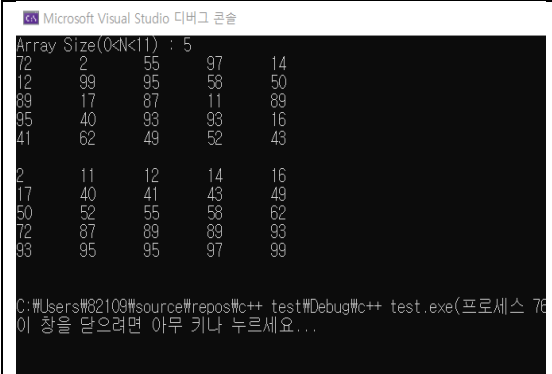
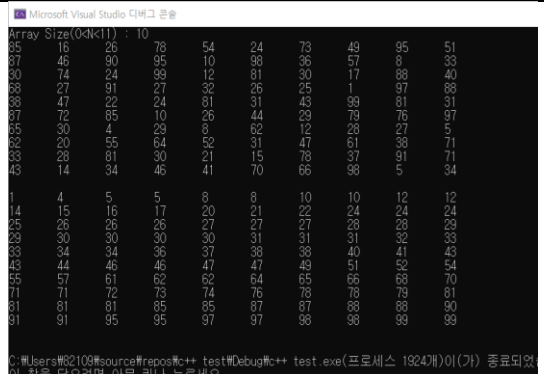
그 후 `int Makearr(int**& rarr, int num)` 함수를 통해 동적할당으로 `size*size` 크기의 2차원 배열을 만들고 1~99 사이의 무작위 난수로 초기화한다. 만약 동적 할당으로 배열이 정상적으로 생성되지 않은 경우 "메모리 할당 실패" 메시지 후 프로그램을 종료한다.

`void Printarr(parr, num)` 함수를 통하여 배열을 정해진 형식에 맞게 출력한다. 그리고 크기 `size*size`의 `int **` 형 `parr`, 1차원 포인터 배열을 만든다. 그리고 이 배열의 값은 순차적으로 2차원 배열 `arr`의 주소값으로 설정한다.

그 후 `void Sortarr(int**& rarr, int num)` 함수를 통하여 순차 정렬을 통해 배열을 정렬한다. 순차 정렬을 하는 과정에서 두 변수의 값을 바꿔야하는 경우에는 `int Change(int& a, int& b)` 함수를 통해 두 값을 바꾼다.

정렬을 완료한 후 다시 배열 출력 함수를 통하여 정렬된 배열을 출력한 후 프로그램을 종료한다.

- 결과 사진

초기화면	Size 잘못된 값 입력
	
Size = 5	Size = 10
	

- 고찰

해당 과제를 수행하는 과정에서 2차원 이상의 배열을 정렬할 때 1차원 배열과 포인터

를 이용하면 쉽게 할 수 있다는 사실을 알 수 있었다. 또한 배열을 정렬하면서 순차 정렬에 대해 복습할 수 있는 시간을 가질 수 있었다.

4. Assignment 4

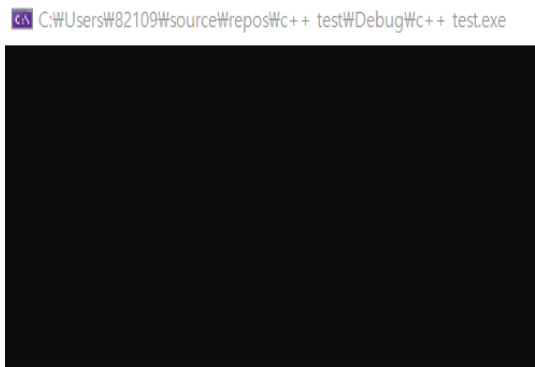
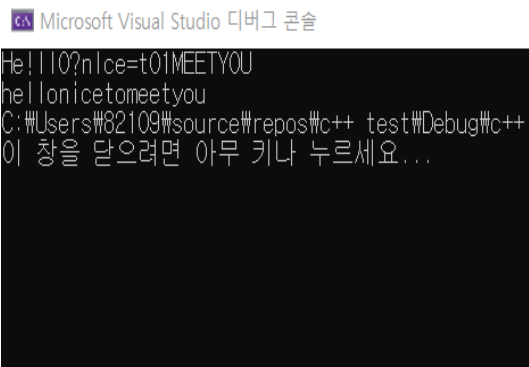
- 문제 설명

해당 과제는 공백이 없는 문자열을 입력을 받는다. 이 문자열에서 대문자는 소문자로 바꾸고, 알파벳 이외의 문자는 없앤 결과를 출력한다.

- 해결 과정

우선 string 형 문자열 변수 str를 입력받는다. 그 후 str의 첫번째부터 'w0'를 확인할 때까지 문자열을 출력하는데, 소문자와 대문자만 입력을 받도록 if문을 사용한다. 그리고 대문자일 경우에는 소문자로 바꾼 값(32를 더한 값)을 출력한다. 이외에 소문자는 그대로 출력한다.

- 결과 사진

초기화면	Str = He!!lO?nlce=tO1MEETYOU
	

- 고찰

해당 문제에서는 중요한 점은 아스키코드를 이용하여 특정 문자열을 가려서 받는 것이다. 이 과제를 통하여 알파벳의 아스키코드를 알 수 있었고, 문자열에 특정 값을 더하거나 뺌으로 인하여 다른 문자로 변환할 수 있는 방법을 알았다. 이 방법은 ctype 라이브러리에 tolower(), toupper(), isalpha()등을 활용하면 더욱 쉽게 문자를 바꾸고 비교할 수 있다.

5. Assignment 5

- 문제 설명

해당 문제는 문자열을 공백도 입력이 가능하도록 입력받는다. 그 후 입력된 문자열에서 반복되는 문자가 있을 경우 해당 문자와 반복된 횟수로 출력한다.

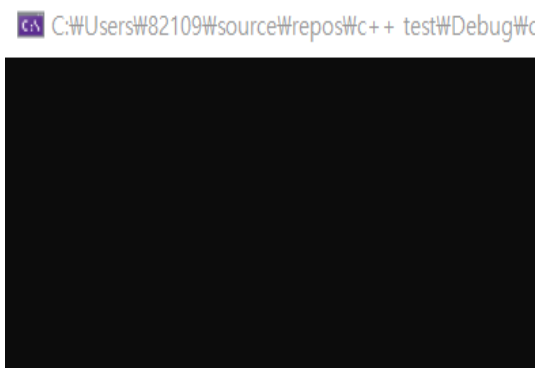
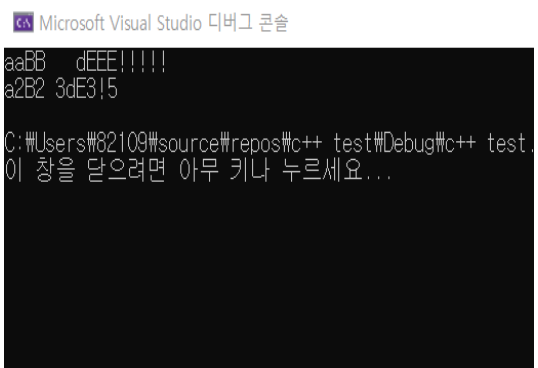
- 해결 과정

우선 문자열을 char형 변수 str1[1024]를 cin.getline(str1, sizeof(str1))을 통해 입력을 받는다. char str2[1024]는 가공할 문자열을 저장할 변수다. char temp를 통해 str1[0]의 값을 저장하고, 반복된 횟수를 저장할 변수 int count를 1로 초기화한다. 입력받는 문자열의 시작 주소값을 char* pstr1 변수에 저장하고, 가공할 문자열의 주소값을 저장할 변수 char* pstr2는 str2의 시작 주소값을 저장한다. pstr2 주소가 가리키는 값이 'W0'가 되도록 반복한다. 아래 내용을 반복한다.

우선 pstr1의 주소값을 한 칸 다음으로 옮기고 pstr1의 값과 temp를 통해 이전 값과 비교한다. 만약 반복되었을 경우에는 count를 1 더하고 마무리된다. 반복되지 않았을 경우 만약 count가 1일 경우 pstr2에 temp의 값만을 입력하고, count가 이 이상일 경우 문자와 count값을 순서대로 기록한 후 count를 다시 1로 바꾼다.

위의 반복문이 끝나면 pstr2에 'W0'를 추가한 후 str2를 출력한다.

- 결과 사진

초기화면	Str = aaBB dEEE!!!!
	

- 고찰

해당 과제를 하면서 공백 문자를 포함한 입력을 받을 때에는 cin.getline()을 통하여 입력받을 수 있다는 사실을 알았다. 그리고 _itoa_s를 통하여 숫자를 문자열에 더욱 쉽게 넣을 수 있는 방법도 알게되었다. 만약 변환한 문자열을 저장하지 않고 바로 출력한다면 _itoa_s를 쓰지 않아도 각각을 바로 출력함으로써 문제를 해결할 수 있다는 사실 또한 알았다.

6. Assignment 6

- 문제 설명

해당 문제는 3가지 함수를 구현하는 것을 목표로 한다. Void Sender(void) 함수는 숫자를 문자열로 입력받고 숫자 마지막에 각 자릿수의 합의 첫째자리 수를 추가하고 이를 출력한다. 그리고 해당 문자열을 void Transmission_Process(const char*) 함수로 전달한다.

Transmission_Process() 함수에서는 전달받은 문자열을 40%의 확률로 5자리의 수 중 임의의 1개의 자리의 숫자를 바꾼다. 그리고 이를 void Receiver(const char*) 함수로 전달한다.

Receiver() 함수에서는 문자열의 첫번째 수부터 4번째 수까지의 합의 첫째자리 수와 문자열의 5번째 수의 값이 일치하는지 확인하고 각각의 상황에 맞는 결과를 출력한다.

- 해결 과정

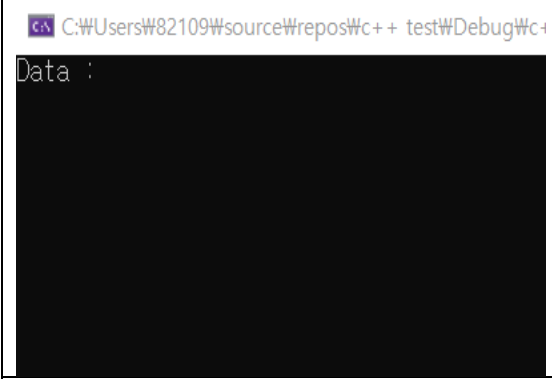
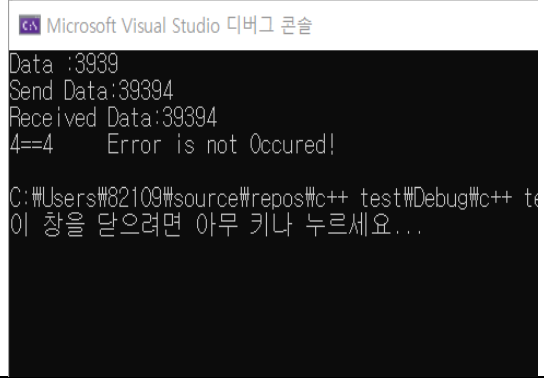
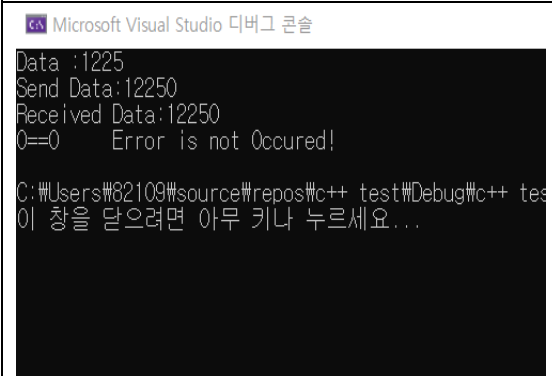
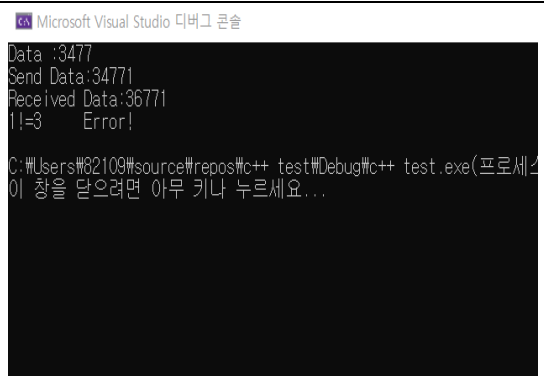
메인 함수에서는 Sender() 함수를 호출한다. Sender()에서는 입력을 받을 문자를 저장할 변수 char num[5]과 가공할 변수 char snum[6]를 만들고 num 변수에 값을 입력받는다. 그리고 입력받은 문자열을 atoi() 함수를 통하여 int intnum에 정수형태로 저장한다. 그 후 int 형 변수 sum에 각 자릿수의 값을 더한 값을 일의 자릿수를 저장하고 이를 _itoa_s() 함수를 통하여 마지막 자리에 추가한 수를 snum에 저장한 뒤 Transmission_Process() 함수를 호출한다.

int 형 변수 prob은 40%의 확률을 의미하게 된다. 만약 prob이 0,1일 경우에는 아래의 동작을 작동하게 하고 아닐 경우에는 Receiver에 Sender() 함수에서 전달받은 pnum을 그대로 전달한다. 40%의 확률로의 동작 내용은 prob2를 통해 랜덤한 자리를 우선 선택하게 된다. 그리고 해당되는 자리의 값이 0 ~ 9까지의 무작위 값을 갖는 change와 다를 때까지 change를 바꾸고 다를 경우 change의 값으로 해당 값을 바꾼다. 그리고 바꾼 값을 Receiver에 전달한다.

Receiver() 함수에서 우선 전달받은 문자열을 출력한다. 그 후 다시 int int_tnum 변수에 atoi() 함수를 통하여 전달받은 문자열을 정수로 변환한다. 그리고 전달받은 문자열의 0 ~ 4번째 자리의 수의 합산의 일의 자리를 5번째 자리의 수와 비교한다. 만약 다를 경우 값이 어떻게 다른지와 에러를 출력하고 같을 경우에는 같은 값을 출력하고 에러가 발생하지 않았다는 문구를 출력 후 프로그램이 종료된다.

- 결과 사진

초기화면	No Error 1
------	------------

	
No Error 2	Error
	

- 고찰

해당 과제를 수행하면서 string 라이브러리에 `_itoa_s()` 함수와 `atoi()` 함수, `strcpy_s()` 함수를 사용하는 것에 더욱 익숙해졌다. 그리고 `rand()`를 통하여 확률적으로 동작할 수 있도록 하는 방법을 알 수 있게 되었다. 해당 문제의 내용은 데이터 전달 시 전달된 데이터가 손실이 되어있는지 확인할 수 있는 방법으로 보인다.

7. Assignment 7

- 문제 설명

해당 문제는 x만을 변수로 하거나 0차식을 입력을 받아 해당 식이 올바른 식인지 검사하는 문제다. 해당 식에서 정상적인 식은 곱셈은 생략할 수 없도록 한다.

- 해결 과정

입력을 받을 char형 변수 `str[1024]`를 준비한다. 그리고 식을 확인하는데 쓰일 char 형 변수 `copy[1024]`를 준비한다. `str`에 식을 입력 받고 이를 `strcpy_s`를 통해 `copy`에 복사한다. 그 후 아래의 내용을 계속 반복한다.

가장 마지막에 입력한 '('의 위치를 저장할 int 형 변수 `start`를 -1로 초기화 후 선언하고,

start 위치 이후에 나올 가장 첫 번째 ')'의 위치를 저장할 int 형 변수 end를 선언한다. 그리고 bool PareCheck(const char* str, int& start, int& end) 함수를 통하여 아래 과정을 수행하고 false가 나올 경우 오류를 출력한다.

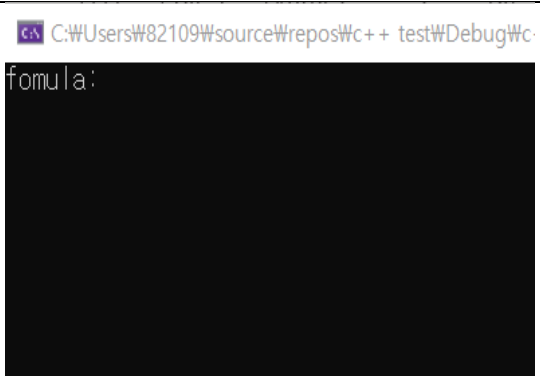
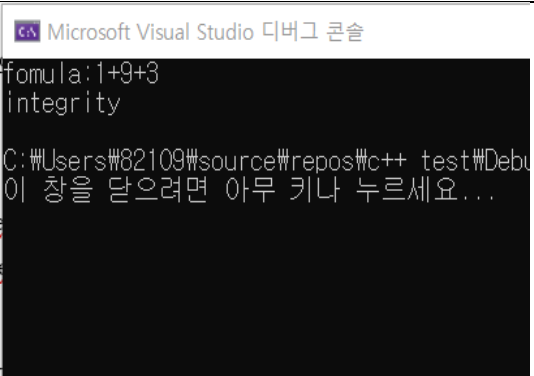
PareCheck() 함수는 '('의 마지막 위치를 확인하여 int& start에 저장한다. 만약 '('가 없을 경우 ')'가 있는지 확인하여 존재하면 false를 반환한다. 둘 다 없을 경우에는 int& end의 값을 'w0'의 위치로 바꾸고 truth를 반환한다. '('가 있을 경우 start 위치 이후로부터 ')'가 첫번째 위치의 값을 end에 저장한다. 검사를 마쳤을 때 ')'가 아직 초기값일 경우 false를 반환한다. 초기값이 아닐 경우에는 truth를 반환한다.




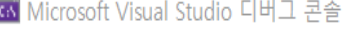
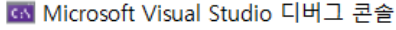
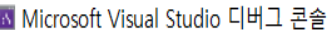
PareCheck() 함수를 마친 후 start와 end 사이의 식 부분을 검사하는 bool FormulCheck(const char* str, const int start, const int end) 함수를 수행하고 반환되는 값이 false인 경우 오류를 출력한다.

FormulCheck() 함수는 우선 end와 start가 붙어있는지, 즉 "()"와 같이 입력되어 있는지 확인하여 해당 방식으로 입력된 경우 false를 반환한다. 그 후 문자열의 start의 위치부터 각 자리마다 아래 내용을 반복한다. 우선 확인하는 문자가 연산자, 숫자, 'x' 중 하나인지 확인한다. 아닐 경우 false를 반환한다. 그리고 연산자를 확인하는 경우 구간에서 양쪽 끝에 있거나 좌우에 수나 문자가 없을 경우 false를 반환한다. 만약 확인하는 문자가 'x'일 경우 좌우에 수가 붙어있을 경우 false를 호출한다. 이 반복은 문자열의 end위치까지 반복한다. 이를 정상적으로 완료된 경우 truth를 반환한다.

식 검사과정을 마친 후 start가 초기값, 즉 괄호가 없는 경우 integrity를 출력 후 함수를 종료한다. 아닐 경우 검사한 부분을 'x'로 치환하는 과정을 가진다. Char형 변수 strn[100]에 미리 "x"를 입력해두고 문자열 copy의 end 위치 뒤쪽을 strn 뒤에 입력하고 copy의 시작위치를 'w0'로 바꾼다. 그리고 strcat_s()를 통하여 copy에 strn을 이어붙인다. 그리고 치환한 식을 출력한다.

- 결과 사진

초기화면	0차식 괄호 없음
	
1차식 괄호 없음	0차식 괄호 1쌍

 <pre>fomula:1+2+x integrity C:\Users\82109\source\repos\c++ test\ 이 창을 닫으려면 아무 키나 누르세요...</pre>	 <pre>fomula:1+(2*3/1)-3 1+x-3 integrity C:\Users\82109\source\repos\c++ test\Debug 이 창을 닫으려면 아무 키나 누르세요...</pre>
1차식 괄호 1쌍	1차식 괄호 2쌍
 <pre>fomula:x+(1*x+9-1)/2 x+x/2 integrity C:\Users\82109\source\repos\c++ test\De 이 창을 닫으려면 아무 키나 누르세요...</pre>	 <pre>fomula:1+x/(2*(x*x)-1)-7 1+x/(2*x-1)-7 1+x/x-7 integrity C:\Users\82109\source\repos\c++ test\Debu 이 창을 닫으려면 아무 키나 누르세요...</pre>
괄호 오류	식 오류
 <pre>fomula:1+((1*x) 1+(x faulty C:\Users\82109\source\repos\c++ test\ 이 창을 닫으려면 아무 키나 누르세요...</pre>	 <pre>fomula:+1/x faulty C:\Users\82109\source\repos\c++ test\Debug 이 창을 닫으려면 아무 키나 누르세요...</pre>

- 고찰

해당 문제를 접근한 방법은 가장 안쪽 괄호부터 확인하는 방법이다. 가장 안쪽 괄호가 정상적으로 되어있는지 확인 후 안쪽 식을 확인한 후 이 식을 인식할 수 있는 문자인 'x'로 바꾸어 다시 확인을 진행하는 방식으로 알고리즘을 구현하였다. 만약 해당 문제를 0차식의 식을 계산하는 것으로 바뀔 경우 안쪽부터 결과를 확인한 후 이를 결과값으로 치환하는 방법으로 구현할 수 있겠다는 생각이 들었다.

8. Assignment 8

- 문제 설명

5*5 크기의 2차원 문자열 배열을 만들고 첫번째 문자는 'a', 나머지는 '0'으로 초기화한다. 그 후 'w', 'a', 's', 'd' 중 하나로 방향을 입력받고 해당 방향으로 갈 수 있을 경우 그 방향의 값을 'a'로 바꾼다. 그리고 화면을 지우고 다시 출력하고 입력을 받는다. 만약 모든 방향으로 갈 수 없을 경우 END를 출력한 후 프로그램을 종료한다.

- 해결 과정

처음으로 5*5 크기의 2차원 char형 배열 arr[5][5]를 생성하고 첫번째 문자는 'a', 나머지는 '0'으로 초기화한다. 그 후 입력받은 방향을 저장할 char형 변수 cmd를 생성하고, int형 변수 x, y를 초기값을 0으로 생성한다. 그 후 아래 내용을 계속 반복한다.

우선 배열을 void Printarr(char (*parr)[MAX_SIZE])를 통해 배열을 출력한다. 그 후 bool Checkarr(char(*parr)[MAX_SIZE], int x, int y)를 통해 각 방향이 막혀있거나 'a'로 채워져있는지 확인하고 전부 막혀있을 경우 false를 반환한다. 반환된 값이 false인 경우에는 END를 출력 후 프로그램을 종료한다. Checkarr() 함수가 참인 경우 방향을 입력받는다.

만약 'w'를 입력받은 경우 해당 방향이 첫번째 행이 아니고 위쪽에 'a'가 없을 경우 void Up(char(*parr)[MAX_SIZE], int &x, int &y) 함수를 통해 좌표와 값을 바꾼다.

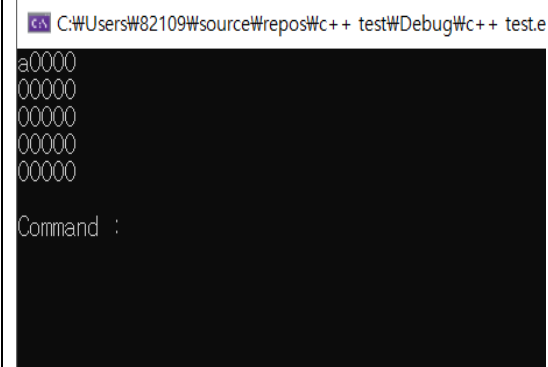

만약 'a'를 입력받은 경우 해당 방향이 첫번째 열이 아니고 왼쪽에 'a'가 없을 경우 void Left(char(*parr)[MAX_SIZE], int &x, int &y) 함수를 통해 좌표와 값을 바꾼다.



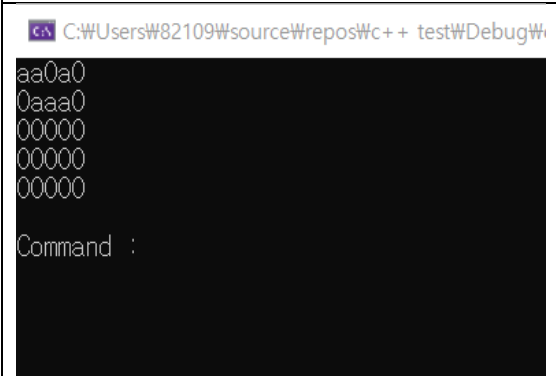
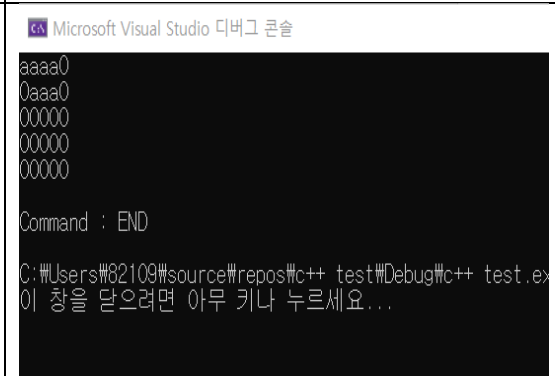
만약 's'를 입력받은 경우 해당 방향이 마지막 행이 아니고 아래쪽에 'a'가 없을 경우 void Down(char(*parr)[MAX_SIZE], int &x, int &y) 함수를 통해 좌표와 값을 바꾼다.

만약 'd'를 입력받은 경우 해당 방향이 마지막 열이 아니고 오른쪽에 'a'가 없을 경우 void Right(char(*parr)[MAX_SIZE], int &x, int &y) 함수를 통해 좌표와 값을 바꾼다.

위의 작업을 완료한 후 화면을 system("cls") 함수를 통해 지운다.

- 결과 사진

초기 화면	잘못된 입력
	

D	S
	
W	A, End
	

- 고찰

해당 과제를 구현하는 과정을 통해 좌표를 통해 배열을 조작할 수 있는 방법을 알 수 있었다. 그리고 window.h 헤더파일의 Command 명령어를 활용할 수 있는 함수에 대해 알 수 있었다.