



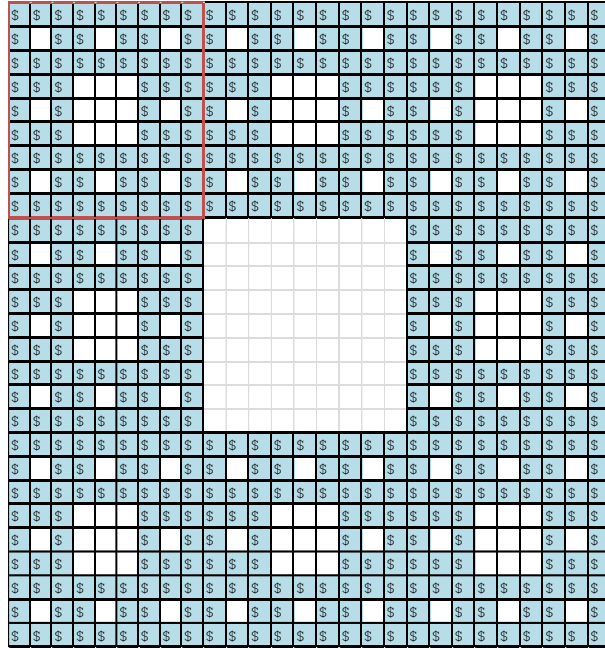
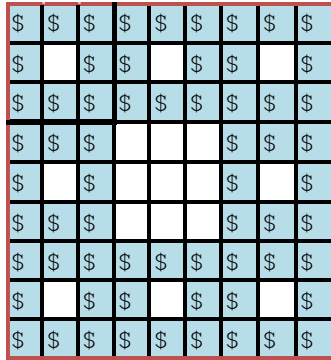
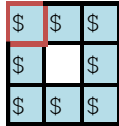
1차 과제

과목명	객체지향프로그래밍
실습분반	B (02)
담당교수	신영주 교수님
학과	컴퓨터정보공학부
학년	2학년
학번	2019202009
이름	서여지
제출일	20200416

1. 시어핀스키 카펫 출력하기

❖ 문제 및 설명

사용자로부터 정수 N 값을 입력 받아 크기가 $N \times N$ 인 시어핀스키 카펫을 출력하는 프로그램을 작성하는 문제이다. 시어핀스키 카펫은 9등분 된 정사각형의 가운데 칸을 제외한 나머지 8개의 칸에 특정한 패턴이 반복되는 도형으로, 부분의 모습이 전체의 모습과 유사한 특징을 갖는 프랙탈 도형이다.



시어핀스키 카펫을 관찰하여 발견한 특징 중 프로그램 작성에 사용된 것은 다음과 같다.

- 1) 모든 칸을 기호로 채운 뒤, 빈칸을 뚫는 것이 프로그램 작성에 유리할 수 있다.
- 2) $N \times N$ 크기의 카펫에서 크기로 구분한 빈칸의 종류는 N 을 3의 거듭제곱 꼴로 나타냈을 때 지수의 크기와 같다.
- 3) $N \times N$ 크기의 카펫에서 가장 큰 빈칸의 크기는 $(N/3 * N/3)$ 이며, 빈칸의 개수는 1개이다.
- 4) 두 번째로 큰 빈칸의 크기는 $(N/3^2 * N/3^2)$ 이며, 빈칸의 개수는 9개이다. (더 큰 빈칸에 포함되는 부분까지 계산하기로 한다)
- 5) a 번째로 큰 빈칸의 크기는 $(N/3^a * N/3^a)$ 이며, 빈칸의 개수는 9^a 개 이다.
- 6) a 번째로 큰 빈칸 사이의 간격은 가로와 세로의 간격 모두 (빈칸의 크기*3)이다.

프로그램은 다음과 같이 작동한다.

N*N크기의 시어핀스키 카펫을 출력하기 위해 사용자로부터 입력 받은 수를 int형 변수 N에 저장한다. 입력 받은 수를 3으로 반복하여 나누어 3의 지수를 찾아 int형 변수 power에 저장한다. 만약 N이 3의 곱으로 이루어진 수가 아니라면, 오류메시지를 출력하고 프로그램을 종료한다.

N*N크기의 2차원 char 배열 arr을 동적할당하고, 배열의 모든 칸에 '\$'기호를 채운다.

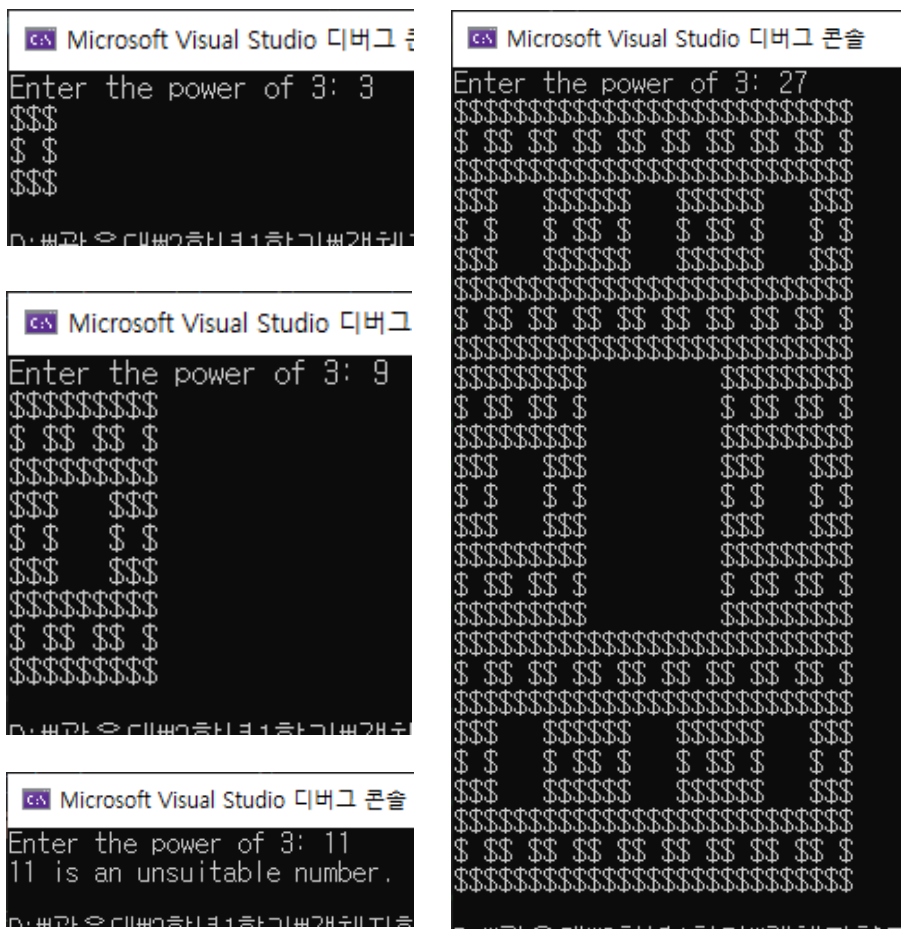
power의 수만큼 hole함수를 반복하여 호출한다.

hole함수는 N의 크기와 hole함수를 호출한 횟수 i, 배열 arr의 포인터 값을 인자로 전달받는다. N을 3으로 i번 나누어 구멍의 크기를 구하고 int형 변수 hole_size에 저장한다. int형 변수 repeat을 선언하여 1로 초기화 하고, 9를 (i-1)번 곱하여 빈칸 삽입을 반복할 횟수를 구한다.

int형 변수 x와 y에 hole_size를 대입하고, 반복문을 이용하여 (hole_size*hole_size) 크기의 공간에 공백 문자를 삽입한 뒤, (hole_size*3) 만큼 이동하여 다음 공백문자를 삽입하는 것을 반복한다.

빈칸 삽입이 모두 종료된 후, 배열을 모두 출력하고 배열을 해제한다.

❖ 결과화면



❖ 고찰

가장 처음 문제를 접했을 때, 정해진 칸에 기호를 채우는 방법을 찾으려 했으나 고려해야 하는 칸이 많아 규칙을 찾기 어려웠고, if 조건문이 복잡하게 연결되었다. 시간을 들여 주어진 도형을 천천히 분석하여 빈칸의 개수가 훨씬 작으며 규칙을 발견하기 쉽다는 결론을 얻었고, 모든 칸에 기호를 채워 넣은 다음 공백으로 덮어씌우는 방법을 채택하게 되었다. 가장 큰 빈칸을 제외한 빈칸의 반복 회수를 8회로 정하고 빈칸을 삽입할 위치를 찾는 것 보다, 더 큰 빈칸에 포함된 공백을 포함하여 반복 회수를 9회로 정하는 것이 코드를 간략하게 나타내는 것에 유리했다.

2. 에라토스테네스의 체를 이용한 소수구하기

❖ 문제 및 설명

사용자로부터 정수 n 을 입력 받아 n 이하의 소수를 모두 출력하고, 마지막으로 출력된 소수의 개수를 출력하는 프로그램을 작성하는 문제이다.

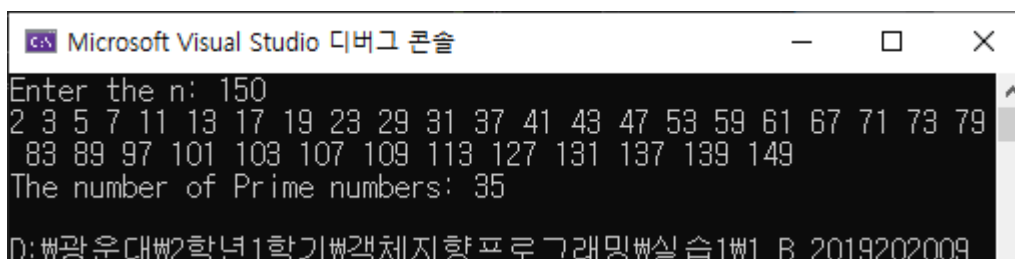
에라토스테네스의 체를 이용한 알고리즘은 2부터 n 까지의 수 중에서 가장 작은 소수에서 시작하여 \sqrt{n} 이하의 모든 소수에 대하여 각 소수의 배수들을 모두 지우는 과정을 반복해서 소수를 구한다.

사용자로부터 정수를 입력 받아 int형 변수 n 에 저장하고, $(n-1)$ 크기의 int형 배열 $list$ 를 동적할당한다.

int형 변수 p 에 가장 작은 소수를 저장하고 배열의 처음부터 끝까지 조회하여 p 로 나누어 떨어지는 수 대신 0을 저장한다. 배열의 끝까지 p 의 배수 삭제를 마친 후, p 에 다음 소수를 저장하여 합성수를 삭제하는 과정을 반복한다. 이 과정은 while 반복문을 통해 $(p * p \leq n)$ 인 동안 반복된다.

while문이 종료된 후 $list$ 에 남은 0이 아닌 수들을 모두 출력하며 소수의 개수를 세고, 출력이 모두 종료된 후 마지막으로 소수의 개수를 출력한 후 프로그램이 종료된다.

❖ 결과화면



```
Microsoft Visual Studio 디버그 콘솔
Enter the n: 150
2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79
83 89 97 101 103 107 109 113 127 131 137 139 149
The number of Prime numbers: 35
D:\광운대\2학년1학기\객체지향프로그래밍\실습\1\1_B_2019202009
```

❖ 고찰

이 문제는 주어진 알고리즘을 c++을 이용해 코드로 구현하는 문제였다. 소수를 판별하는 내용이 담긴 반복문을 \sqrt{n} 이하 소수에 대해 반복해야 하는데, 제곱근 연산을 하는 것이 아닌 p*p로 나타낼 수 있는 점이 특이했다. 구한 소수의 개수를 구하기 위해 for반복문을 따로 사용했었는데, 찾아낸 소수를 출력하는 반복문에 해당하는 내용을 통합하여 더 간단하게 나타낼 수 있었다.

3. Bitstream 계산하기

❖ 문제 및 설명

다음과 같은 bit연산을 반복하는 프로그램을 작성하는 문제이다.

- 1) g(): bitstream을 역순으로 정렬한다.
- 2) f(): bitstream의 각 요소에 대해 not연산을 하여 bit를 반전시킨다.
- 3) 반복한 횟수가 짝수일 경우 마지막에 1, 홀수인 경우 마지막에 0을 붙인다.

사용자로부터 계산을 반복할 회수를 입력 받아 int형 변수 N에 저장한다. 이후 Bitstream을 입력받아 char형 배열 bitstream에 저장한다. bitstream의 길이를 length에 저장하고, 길이가 (length+2)인 char형 배열 p를 동적할당한다. 반복문을 이용하여 p에 bitstream의 내용을 역순으로 저장한다. 이후 각 자리 비트에 not연산을 하고, p의 끝에 '1' 또는 '0'을 붙인 뒤, p의 마지막 칸에 0(NULL)을 넣어 문자열의 마지막을 표시한다.

완성된 bitstream을 10진수로 변환하는 과정은 bitstream의 길이를 확인하여 overflow인 경우의 처리를 하는 것으로 시작한다. 이후 p의 마지막 자리부터 조회하여 10진수로 표현한 값을 int형 변수 deci에 저장하여 출력한다.

❖ 결과화면

```
Microsoft Visual Studio 디버그 콘솔
Input N: 12
Input Bitstream: 0100
1: 11011
27
2: 001000
8
3: 1110111
119
4: 00010000
16
5: 111101111
495
6: 0000100000
32
7: 11111011111
2015
8: 000001000000
64
9: 111111011111
8127
10: 00000010000000
128
11: 11111110111111
32639
12: 0000000100000000
256
D:\광운대\2학년1학기\객체지향프로그래밍
```

❖ 고찰

배열 `p`를 생성하여 `g()` 연산을 했을 때, 출력이 정상적으로 이루어지지 않는 오류를 만들었다. 배열 `p`의 길이를 1 늘리고, 마지막에 `NULL` 문자를 저장하여 문제를 해결했다. 그 밖에도 비트스트림을 10진수로 변환하는 과정에서 오버플로우가 발생할 경우의 예외처리를 하지 않거나 10진수 변환이 비정상적으로 이루어지는 문제가 발생하기도 했었다. 위의 문제들은 이진수로 표현된 수를 10진수로 변환하는 과정을 프로시저 단위로 실행하며 오류를 수정하였다.

4. 문자열 뒤집기

❖ 문제 및 설명

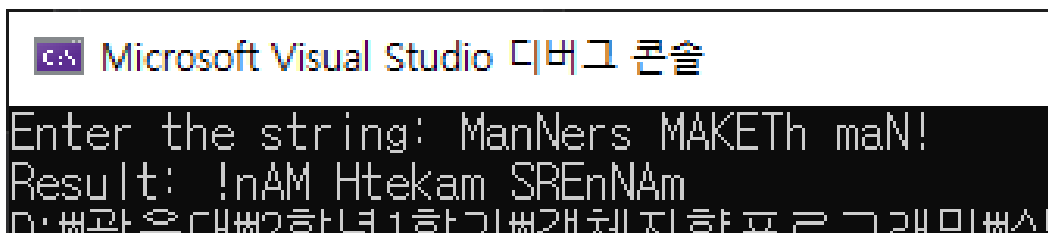
이 문제는 문자열을 입력 받아 순서를 거꾸로 정렬한 뒤, 소문자와 대문자를 서로 바꾸는 프로그램을 작성하는 문제이다. 이 문제를 해결하기 위해 알아야 하는 점은, 대문자와 소문자의 아스키코드 차이가 32로, 대문자A가 소문자a보다 32만큼 작다는 것이다.

프로그램은 다음과 같이 작동한다.

사용자로부터 문자열을 입력받아 `char`형 배열 `str`에 저장하고, 반복문을 이용하여 구한 길이를 `length`에 저장한다. (`length+1`)크기의 `char`형 배열 `p`를 동적 할당 한 후, `str`의 내용을 역순으로 저장하고, 배열 `p`의 마지막에 0(`NULL`)을 저장한다.

배열 `p`를 처음부터 조회하여 문자가 소문자 알파벳인 경우 아스키코드값에 32를 뺀 값을 저장하여 대문자 알파벳으로 바꾼다. 반대로 배열 `p`에서 조회한 문자가 대문자 알파벳인 경우 아스키코드값에 32를 더하여 소문자 알파벳으로 변환하고, 결과를 출력한다.

❖ 결과화면



```
Microsoft Visual Studio 디버그 콘솔
Enter the string: ManNers MAKETH maN!
Result: !nAM Htekam SREnNAm
n·##과온대##2하년1하기##개체지하표르그래민##시
```

❖ 고찰

`std::cin` 을 이용하여 문자열을 입력 받았을 때 문자열에 띄어쓰기가 있는 경우 띄어쓰기의 앞부분까지만 저장되었다. 따라서 다른 방법을 고민해야 했고, 이 문제에서는 `string`형을 이용하지 않고

프로그램을 구성해야 하므로, cin.getline을 이용하여 개행문자(\n) 이전의 문자열 전체를 입력 받도록 수정하였다. 문자열을 역순으로 저장하는 것은 문제 1-3에서 비트스트림의 순서를 뒤바꾸는 것과 같은 방법으로 처리하였다.

5. Random matrix, transposed matrix

❖ 문제 및 설명

사용자로부터 행과 열의 수를 입력 받아서 난수 값으로 채워 origin matrix를 출력하고, origin matrix의 전치행렬 transposed matrix를 출력하는 프로그램을 작성하는 문제이다. 이때 행렬에 채워지는 값은 0이상 row*column미만의 값이고, 중복되지 않아야 한다.

프로그램은 다음과 같이 동작한다.

사용자로부터 row와 column을 입력 받아 각각 int형 변수 row, column에 저장한다. 반복문을 사용하여 row*column 크기의 int형 2차원 배열 origin_matrix를 동적할당 한다. 난수를 발생시킨 후, repetition_check 함수를 이용하여 origin_matrix에 중복되는 요소가 있는지 확인한다. 중복되는 요소가 있는 경우, 난수를 다시 발생시키고, 중복검사를 통과한 이후 origin_matrix에 저장한다.

반복문을 이용하여 origin_matrix를 순서대로 출력한다. Transpose Matrix는 origin_matrix의 출력순서를 조정하여 출력한다.

❖ 결과화면

```
Microsoft Visual Studio 디버깅 콘솔
Enter the row: 7
Enter the column: 9

Origin Matrix >>>
61 53 8 22 47 56 51 62 28
39 36 59 54 58 11 3 12 27
23 26 13 40 38 33 52 21 42
2 25 49 31 24 43 15 34 4
46 60 44 10 57 35 0 50 45
9 37 41 1 18 7 17 55 20
16 32 5 29 48 14 6 30 19

Transpose Matrix >>>
61 39 23 2 46 9 16
53 36 26 25 60 37 32
8 59 13 49 44 41 5
22 54 40 31 10 1 29
47 58 38 24 57 18 48
56 11 33 43 35 7 14
51 3 52 15 0 17 6
62 12 21 34 50 55 30
28 27 42 4 45 20 19
```

D:\관우대\2학년1학기\객체지향프로그래밍\실험1\B_20

❖ 고찰

행렬을 생성하기 위해 2차원 배열을 동적할당 해야 했는데 동적할당 한 배열의 row와 column을 거꾸로 처리해서 행과 열이 뒤바뀌는 오류를 만들었었다. 전치행렬을 출력할 때 새로운 배열을 할당 하여 전치행렬을 저장하려 했으나, 한 번의 출력만 하면 되는 문제이기 때문에 row와 column값을 조정하여 출력하도록 구성하였다. 모든 출력이 끝나고 동적할당 했던 2차원 배열의 메모리 공간을 해제하는 것을 잊었다가 후에 추가하였다.

6. Tic-Tae-Toe

◆ 문제 및 설명

3*3크기의 칸에서 가로 혹은 세로, 대각선에 같은 기호로 채우면 승리하는 게임인 Tic-Tae-Toe를 프로그램으로 구현하는 문제이다.

프로그램이 실행되면 3*3 크기의 char형 2차원 배열 arr을 할당하여 ' '을 저장한다. print_tictactoe함수를 이용하여 게임판을 출력한다. 이후 while반복문에서 사용자로부터 x와 y값을 입력 받아 int형 변수 x,y에 각각 저장하고, 입력 값이 적합한지 판단하여 적합하지 않은 경우 -1을 반환하고 프로그램을 종료한다. x,y값이 적합할 때, 사용자에게 따라 'X' 또는 'O'를 게임판의 해당하는 칸에 저장한다. 그 다음, win함수를 이용하여 승리조건을 확인하고, 승리조건이 만족되었거나, 반복 횟수가 10회가 되었을 때 반복문을 탈출하여 게임이 종료된다. 반복 횟수가 10회가 되어 반복문이 종료된 경우 "Draw" 를 출력하고, 승리조건이 만족되어 반복문이 종료된 경우는 승리한 플레이어를 출력한다. 이후 동적할당한 배열을 해제하고 프로그램이 종료된다.

◆ 결과화면

```
Microsoft Visual Studio 디버깅 콘솔

x 0 1 2
Y  _ _ _ _
0 | X |  | 0 |
  _ _ _ _
1 | 0 | X |  |
  _ _ _ _
2 | X | 0 | 0 |
  _ _ _ _

[Play 2] Enter your location [x,y]: 2 1
x 0 1 2
Y  _ _ _ _
0 | X |  | 0 |
  _ _ _ _
1 | 0 | X | X |
  _ _ _ _
2 | X | 0 | 0 |
  _ _ _ _

Draw

D:\개발은대박\한글1학기\배경체지향프로그래밍\
```



```

Microsoft Visual Studio 디버그 콘솔
Y  _ _ _ _ _
0 | X | 0 |   |
  _ _ _ _ _
1 |   | 0 |   |
  _ _ _ _ _
2 |   |   |   |
  _ _ _ _ _
[Play 2] Enter your location [x,y]: 2 2
  x 0  1  2
Y  _ _ _ _ _
0 | X | 0 |   |
  _ _ _ _ _
1 |   | 0 |   |
  _ _ _ _ _
2 |   |   | X |
  _ _ _ _ _
Winner is [Player 2]

```

❖ 고찰

틱택토 게임의 판이 되는 부분의 출력문을 함수로 정의하여 코드를 간략하게 정리할 수 있었다. 게임의 승리조건을 판별하는 win함수에서 ($arr[x][y] \neq ' '$) 조건을 고려하지 않고 프로그램을 작성했다가 게임이 정상적으로 진행되지 않는 오류를 만들었다. win함수가 int값을 반환하여 게임의 종료 여부와 승리자가 누구인지 한 번에 전달할 수 있었다. 문제 1-5에서와 유사하게 x와 y값을 혼동하여 배열에 (y,x)위치에 표시하는 오류가 생기기도 했다.

7. clockwise, dounter-clockwise

❖ 문제 및 설명

세 개의 점 p_1, p_2, p_3 을 순서대로 이었을 때 삼각형을 그리게 되는 방향을 출력해주는 프로그램을 작성하는 문제이다. 시계방향으로 그릴 때 "Clockwise", 반시계 방향으로 그릴 때 "Counter-Clockwise", 세 점이 직선상에 있을 때 "Straight"를 출력한다.

프로그램을 작성하는 데 이용한 방법은 벡터의 외적을 이용하는 것이다.

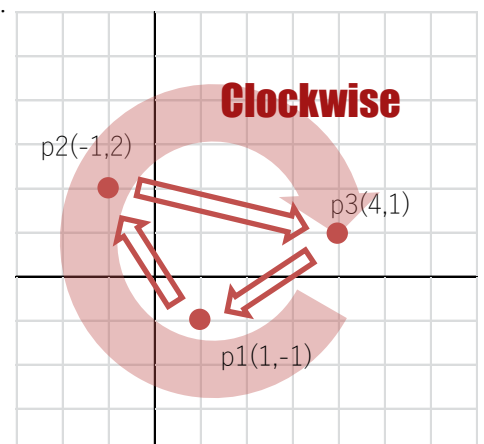
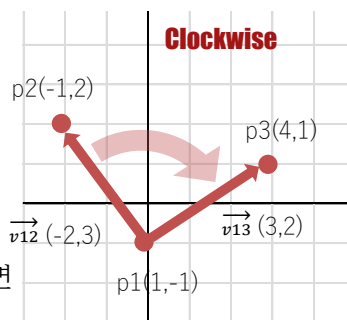
주어진 점 p_1, p_2, p_3 가 각각

$p_1(1,-1), p_2(-1,2), p_3(4,1)$ 일 때,

p_1 에서 p_2 를 가리키는 벡터를 v_{12} ,

p_1 에서 p_3 를 가리키는 벡터를 v_{13} 라 하면

$\vec{v_{12}}=(-2,3), \vec{v_{13}}=(3,2)$ 이다.



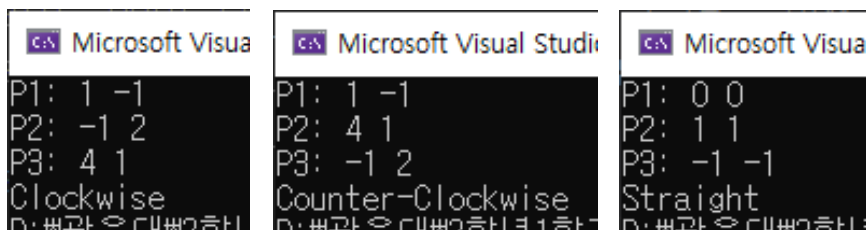
벡터 v_{12} 와 v_{13} 의 외적을 구하면 $\vec{v_{12}} \times \vec{v_{13}} = (-2 * 3) - (3 * 2) = -12$ 이고, 좌표평면으로 들어가는 방향을 가르키며 오른손 법칙에 의해 회전 방향이 시계방향임을 알 수 있다.

두 벡터의 외적 값이 0이면 세 점이 한 직선에 있는 것을 의미하고, 외적이 양수이면, 좌표평면에서 나오는 방향을 가르키며 오른손 법칙에 의해 회전방향이 반시계방향임을 알 수 있다.

이 프로그램은 다음과 같이 동작한다.

사용자 정의 구조체로 struct dot을 선언하여 사용한다. dot 구조체 내부에는 int형 변수 x와 y가 들어있다. 프로그램이 실행되면 사용자에게 3개의 점에 대한 x,y좌표 값을 입력 받아 각각 struct dot형 변수 p1, p2, p3에 저장한다. 점 p2와 p1의 x,y 좌표값을 이용해서 벡터를 뜻하는 v_{12} 를 얻는다. 마찬가지로 점 p3와 p1를 이용해서 v_{13} 을 얻는다. v_{12} 와 v_{13} 에 대한 외적 계산 결과를 int형 변수 z에 저장한다. z가 0인 경우 세 점이 직선상에 있으므로 "Straight"를 출력하고, z가 양수인 경우 "Counter-Clockwise", 음수인 경우 "Clockwise"를 출력한다.

❖ 결과화면



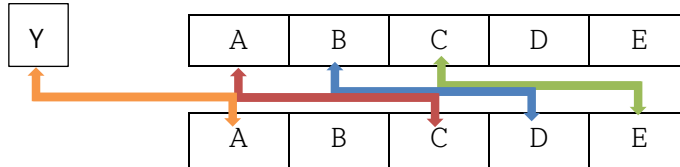
❖ 고찰

x와 y값을 각각 int형 변수로 선언하여 프로그램을 만들었다가, 더 직관적이고 간략하게 수정하기 위해 x값과 y값을 항상 함께 사용하는 특징을 이용하여 struct dot 이라는 사용자 정의 구조체를 활용하여 프로그램을 작성하였다. 가장 처음에는 p1과 p2를 이어 연장한 직선에 대한 p3의 위치를 이용하여 회전방향을 찾으려 했으나 여러가지로 나뉘는 경우의 수와 세 점이 직선상에 위치하는 경우 예외처리의 복잡함을 고려하여 다른 방법을 이용하기로 했다. 벡터의 외적을 이용했을 때의 유리한 점은 방향을 찾는 것과 세 점이 직선상에 있는 경우까지 한 번에 처리가 가능하다는 점 이었다. 수학적인 방법을 통해 프로그램을 단순하게 작성한 것이 인상적인 문제였다. 삼항연산자(?:)를 이용하여 출력문을 한 줄로 정리하였다.

8. ciphertext.txt 해독

❖ 문제 및 설명

Caesar cipher로 암호화 된 txt파일을 해독하는 프로그램을 작성하는 문제이다. Caesar cipher는 문서의 문자를 특정한 key 만큼 shift한 문자에 대입하여 암호화 하는 방식이다.



필요한 key의 값이 주어지지 않았기 때문에, 0이상 127이하의 모든 경우의 수를 고려하여 해독한 결과를 출력한다.

cpp파일이 있는 디렉토리에서 "ciphertext.txt"를 찾아 연다. 텍스트 파일의 한 줄을 읽고 내용을 string 타입 변수 str에 저장한다. string의 함수 .length를 이용하여 int형 배열 ptr를 선언한다. stringstream을 이용하여 공백문자로 구분 된 str의 내용을 string형 변수 a에 하나씩 저장한다. while문 내부에서 string의 내용을 int형으로 전환해주는 stoi 함수를 이용하여 a에 저장된 내용을 ptr배열에 순서대로 저장한다. ptr에 저장된 내용을 한 번 출력하고, 0이상 128미만의 key값을 이용하는 for반복문에서 print_str함수를 이용해 ciphertext.txt를 해독하여 출력한다.

print_str함수에서는 ptr의 내용에 key를 더한 값을 int형 변수 a에 저장하고, a가 128이상이라면 128미만이 될 때까지 128을 빼는 연산을 진행한다. 128은 char형에 저장할 수 있는 경우의 수를 의미하며, 위의 그림에서 Y가 한 바퀴 순환하여 다시 A로 암호화 되듯 전체 128가지 아스키코드가 순환하도록 만드는 것이다. 연산이 종료된 후 a의 값을 unsigned char로 형변환 하여 출력한다.

❖ 결과화면

```
Microsoft Visual Studio 디버그 콘솔
key(77): <krimh kZiar&fma rLZR%fbLma flmkhg. lmfegbd bghma fWazBg. lmkhg. tikZbL^tbg. ^)%&[nm]bms
key(78): =lsjnia[ibbs+cm+nb. s+ms+cm+nb. mliha_mn+fcbe+ch+nb. jlb[ch(JMnliha+jl[cm+ch^_&+Won+cnf
key(79): >mkolbmWket c t Wt. n c omjib no difi c cWdi. lNom]b mWdn. j. p o ln gni jh rcWe dnhdng
key(80): ?nulpkcn]ldu(LpdauLo]u(LeoLpdaLopnkjcaopLl]nplkbLuknLouopai(LsduLejraopLpeiaLeiInkrejClep
key(81): @ovmqldo mevf+pfkwc. qgebw+pqolkdbpgwifkhw+fk+qebw. e f k+Paolkd+mo fpb+fkabba)+_raqfq+
key(82): Apwnrmep nfwa+rfcw+q. w+agq+rfc+arpmlecar. jgl+gl+rfc+af. gl. Orpmle+np. ggc+gl+bccb+^sr+arDc
key(83): Bqxo snfa ogx+vsedxfr x+thr+sedfrs qnmfdrskhmjhm+sedbg. hm. Rsqnmf+oq+hrd+hmcddc+VatsVhsd
key(84): Cryptography, they say, is the strongest link in the chain.
Strong praise indeed, but it's also somewhat dismissive.
If cryptography is in fact the strongest part of your system, why invest time improving it
when there are so many other areas of the system that will benefit more from your attention?
key(85): Dsqquphsbqiz-luifzltbz-lit!uifltuspohtulmjo!joluif!dibjo/ztuspohtqsbjtf!joefte-lcvu!ju
key(86): Et(rvaitcrj. l. vja. t. u. c. l. ku. vja. uvtapiquv. nkpm. kp. vja. ejckp0?Uvtapi. rtckug. kpfggf. "dvw"kv
key(87): Hefbxosnfa ogx+vsedxfr x+thr+sedfrs qnmfdrskhmjhm+sedbg. hm. Rsqnmf+oq+hrd+hmcddc+VatsVhsd
```

key가 84일 때 문구가 출력되는 것을 찾을 수 있다.

Cryptograpy, they say, is the strongest link in the chain.

Strong praise indeed, but it's also somewhat dismissive.

If cryptography is in fact the strongest part of your system, why invest time improving it when there are so many other areas of the system that will benefit more from your attention?

❖ 고찰

c++을 이용하는 파일입출력이 처음이라 여러 번의 시행착오를 경험했다. 그 외에도 stringstream의 사용과 그 결과물로 얻은 a를 16진수 정수로 바꾸는 것에서 어려움이 있었다. c언어에서 사용했던 atoi함수와 유사한 stoi를 이용하여 문제를 해결할 수 있었다. 또한 a에 저장된 수가 128 이상일 경우 아스키코드의 순환을 고려해야 하는데, 이 점을 간과하여 오류를 만들어냈다. 결과물을 출력했을 때 제어문자로 인해 key(n): 문자열이 지워지는 것 때문에 결과 해석에 문제를 겪기도 했다. 이렇듯 프로그램을 작성하는 데 많은 어려움을 겪었지만, 코드를 수정하며 오류를 고치고, 완전히 해독된 암호를 찾는 과정에서 전체적으로 흥미를 느낄 수 있었다.

9. 휴대폰 요금 계산

❖ 문제 및 설명

휴대폰 요금제의 종류와 사용한 문자와 통화의 양을 입력하면, 추가 문자요금과 추가 전화요금, 총 금액을 출력해주는 프로그램을 작성하는 문제이다. 각 요금제마다 기본 요금, 기본제공 문자와 통화, 건당 추가 요금(문자), 분당 추가 요금(전화)가 설정되어있다.

요금제	A	B	C	D
기본 요금 (₩)	10,000	15,000	20,000	25,000
기본제공 문자 (건)	100	150	200	250
기본제공 통화 (분)	60	90	120	150
건당 추가 요금 (₩)	20	15	10	5
분당 추가 요금 (₩/분)	180	150	60	30

사용자로부터 type을 입력받아 char형 변수 type에 저장한다. 문자 사용량과 통화 사용량을 입력받아 각각 int형 변수 text와 call에 저장한다. type을 이용한 스위치 문에서 int형 변수 monthly_fee(기본요금), free_text(기본제공 문자), free_call(기본제공 통화), extra_text_charges(건당 추가 요금), extra_call_charges(분당 추가 요금)에 요금제 별 지정된 값을 저장한다. 이후, text -= free_text 연산을 통해 초과한 문자 이용량을 구하고, 마찬가지로 call -= free_call 연산을 통해

초과한 통화 이용량을 구한다. 이때 text와 call이 음수라면, 초과한 양이 없는 것이므로 0을 저장한다.

int형 변수 result_text를 선언하고, 초과한 문자사용량과 건당 요금을 곱하여 저장한다. 마찬가지로 int형 변수 result_call을 선언하여 초과한 통화사용량과 분당 요금을 곱하여 저장한다. 마지막으로 int형 변수 result_total을 선언하여 기본요금과 result_text, result_call을 모두 더하여 값을 구하고 출력한다.

❖ 결과화면

```

Microsoft Visual Studio
Which type: C
Text: 230
CALL: 100
=====
Result
Extra text: 300
Extra call: 0
Total: 20300
  
```

❖ 고찰

요금제마다 정해져 있는 수치를 대입하여 계산하는 간단한 문제였다. 어떤 요금제가 입력되더라도 계산하는 방법은 모두 같기 때문에 switch-case문을 이용하여 요금제별 수치를 변수에 저장하였다. 초과한 문자와 통화의 양을 구해야 하는데 이때 초과량이 음수인 경우를 고려하지 않아 나타나는 오류를 만들어냈었다. 이 문제에서는 변수의 이름을 명확하게 만드는 것에 집중하였고, 이것은 코드의 전체적인 이해를 쉽게 하는 것에 큰 도움을 주었다.

10. Hit & Blow

❖ 문제 및 설명

무작위로 주어진 4자리 수를 정확하게 맞추는 게임인 Hit & Blow를 수행하는 프로그램을 작성하는 문제이다. Hit&Blow는 4자리 수를 추측하여 제시하면, 제시한 수가 무작위 수와 일치하는 정도를 Hit과 Blow로 제시하여 정해진 횟수 내에 4자리를 모두 맞추는 것을 목표로 하는 게임이다.

6	1	4	0	주어진 수		
8	4	3	7	Hit: 0	Blow: 1	주어진 수에 4가 포함되지만 위치가 다름
0	1	3	4	Hit: 0	Blow: 2	주어진 수에 0, 1, 4가 포함되지만 0,4 위치가 다름, 1은 일치함
6	1	5	9	Hit: 2	Blow: 0	주어진 수에 6,1이 포함되고 위치도 일치함

rand함수를 이용하여 난수를 발생시키고, 사용자로부터 입력받은 수를 처리하여 Hit과 Blow를 출력한다. 반복문을 이용하여 시도한 횟수가 10번이거나, 4개의 Hit을 모았을 때 종료되도록 설계한다.

프로그램은 다음과 같이 동작한다.

시간을 이용하여 난수를 발생시킨 뒤, 10으로 나눈 나머지를 int형 배열 random_numbers에 저장한다. 이때 중복되는 수가 있는 경우 다시 난수를 발생시켜 중복이 일어나지 않도록 한다. 사용자로부터 수를 입력받아 int형 변수 num에 저장한다. num에서 각 자리 수를 분리하여 int형 배열 your_numbers에 저장한다. your_numbers도 중복검사를 시행하여 중복되는 수가 있는 경우 "unsuitable input"을 출력하고, -1을 리턴하여 프로그램을 종료시킨다. your_numbers가 정상적으로 저장되었다면, 이중 for문을 이용하여 your_numbers와 random_numbers의 요소를 비교한다. 일치하는 요소가 있는 경우 각각 요소의 순서까지 일치한다면 hit의 값을 1 증가시키고, 순서는 일치하지 않는다면 blow의 값을 증가시킨다. your_numbers의 모든 요소에 대한 비교가 끝나면 hit과 blow를 출력하여 결과를 알려준다. 이 과정을 10번 반복하며, 중간에 hit의 수가 4가 되는 경우 반복문을 빠져나와 "Win"을 출력하고 프로그램이 종료된다. 10번 시도한 후 반복문을 빠져나온 경우에는 "Lose"를 출력하고 프로그램이 종료된다.

❖ 결과화면



```
Microsoft Visual Studio
Random numbers :
Your number: 0123
>>HIT: 0, BLOW: 1
Your number: 3456
>>HIT: 0, BLOW: 3
Your number: 6789
>>HIT: 2, BLOW: 0
Your number: 6349
>>HIT: 2, BLOW: 1
Your number: 6539
>>HIT: 1, BLOW: 1
Your number: 6384
>>HIT: 4, BLOW: 0
Win

Microsoft Visual Studio
Random numbers :
Your number: 0123
>>HIT: 1, BLOW: 0
Your number: 4567
>>HIT: 2, BLOW: 0
Your number: 8901
>>HIT: 0, BLOW: 2
Your number: 2345
>>HIT: 0, BLOW: 1
Your number: 6789
>>HIT: 1, BLOW: 1
Your number: 1234
>>HIT: 0, BLOW: 2
Your number: 5678
>>HIT: 0, BLOW: 2
Your number: 9012
>>HIT: 0, BLOW: 1
Your number: 3456
>>HIT: 0, BLOW: 1
Your number: 7890
>>HIT: 0, BLOW: 2
Lose

Microsoft Visual Studio
Random numbers :
Your number: 0102
unsuitable input

Microsoft Visual Studio
Random numbers :
Your number: 0102
unsuitable input
```

❖ 고찰

Hit & Blow, 숫자야구 예제는 다른 프로그래밍 언어로 이미 접해본 익숙한 문제였기 때문에 프로그램을 작성하는 데 큰 어려움은 없었다. 그러나 사용자가 입력한 수가 4자가 아닌 경우의 예외처리를 하지 못한 점이 아쉽다. c언어와 비교하였을 때, 변수의 선언을 함수의 가장 앞에 하지 않아도 되는 점이 코드를 작성하고 이해하는 데 큰 도움을 주었다.