



2차 과제

과목명	객체지향프로그래밍
실습분반	B(02)
담당교수	신영주 교수님
학과	컴퓨터정보공학부
학년	2학년
학번	2019202009
이름	서여지
제출일	2020.05.15

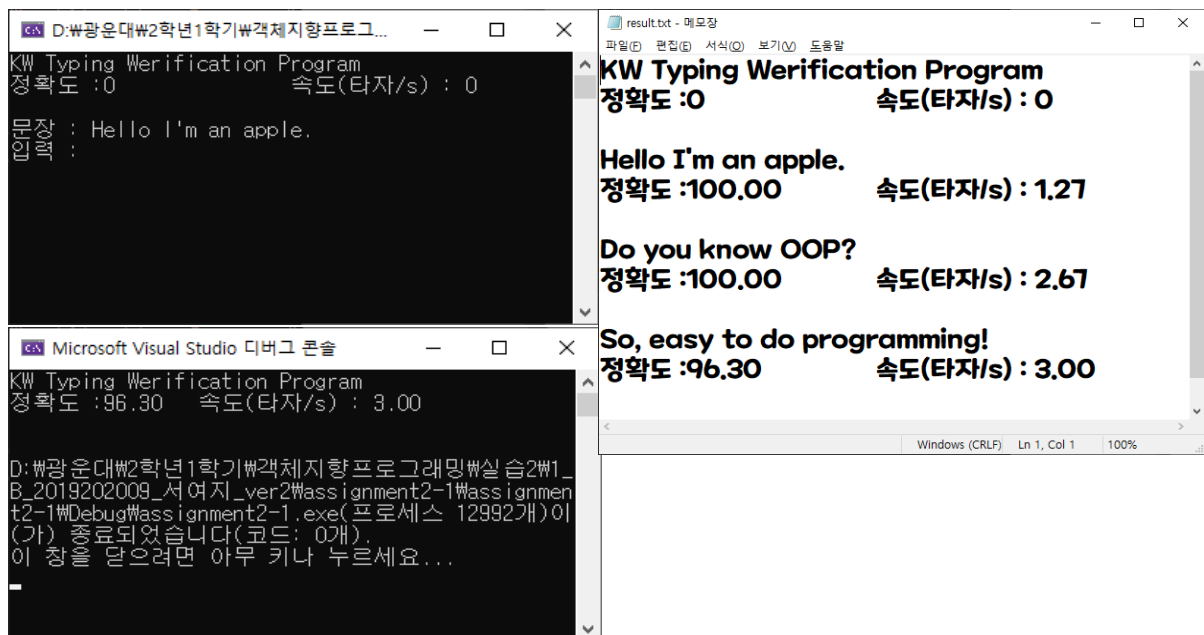
1. KW Typing Verification Program

1) 문제 및 설명

타자의 정확도와 속도를 측정하는 프로그램이다. 작성해야 하는 문장은 source.txt에 적혀있고, 사용자가 문장을 입력하면 정확도와 속도를 계산하여 출력한다. 측정의 결과는 result.txt에 저장한다. txt파일을 읽고 쓰는 것과 시간을 측정하는 것, 문장을 비교하는 것을 고려해야 한다. 문제 해결을 위해 파일입출력과 time함수, string을 이용하였다.

프로그램이 실행되면 source.txt를 찾는다. 이때 파일이 없으면 오류 메시지를 출력하고 종료한다. 프로그램의 제목을 콘솔과 result.txt에 각각 출력하고 source.txt의 파일을 모두 읽을 때까지 타자 검정을 반복한다. 타자 검정은 source.txt의 문장을 한 줄 읽어 출력한 다음 시작 시간을 기록하고, 사용자가 문장을 입력하면 종료 시간을 기록하여 둘의 차이로 걸린 시간을 계산한다. 타자속도는 사용자가 입력한 전체 글자 수를 걸린 시간으로 나눈 값이고, 정확도는 사용자가 알맞게 입력한 글자의 개수를 사용자가 입력한 전체 글자의 수로 나누어 100을 곱한다. 정확도와 속도의 계산이 끝나면 화면을 지우고, 콘솔 창과 result.txt파일에 결과를 출력하여 1회의 타자 검정을 종료한다.

2) 결과화면



3) 고찰

코드를 작성하고 테스트하는 과정에서 source.txt파일을 cpp파일과 같은 경로에 두지 않아 제대로 작동하지 않았었다. 파일입출력을 사용하는 경우 파일의 경로를 확실히 하고, 편의를 위해 주석에도 기록하는 것이 좋을 것이라는 생각이 들었다.

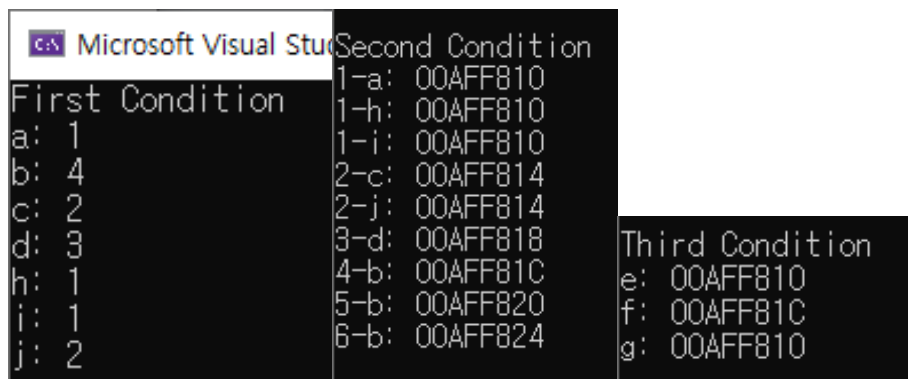
2. Pointer variable

1) 문제 및 설명

여러 가지 포인터 변수를 이차원 배열 `two_d_array`의 각 요소나 행, 혹은 배열 자체로 초기화한 후, 그 값을 이용해 제시된 조건에 맞는 값을 출력하는 문제이다. 이 문제를 해결하기 위해서는 포인터 변수에 대한 전반적인 이해가 필요하다.

프로그램은 가장 먼저 문제에서 제시된 대로 포인터 변수를 선언하고 초기화 한다. 이후 `a, b, c, d, h, i, j` 포인터가 가리키는 정수를 출력한다(조건1). 다음은 조건에 따라 포인터를 이용하여 1~6의 주소 값을 출력한다. 어떤 포인터를 이용하던 같은 결과를 얻을 수 있지만, 제시된 힌트를 참고하여 결과를 출력하였다(조건2). 마지막 5, 6의 경우 같은 행을 가리키는 포인터 `b`를 이용하여 쉽게 나타내었다. 마지막으로, `e, f, g`에 저장된 주소 값을 출력해주었다(조건3).

2) 결과화면



```
Microsoft Visual Studio
First Condition
a: 1
b: 4
c: 2
d: 3
h: 1
i: 1
j: 2

Second Condition
1-a: 00AFF810
1-h: 00AFF810
1-i: 00AFF810
2-c: 00AFF814
2-j: 00AFF814
3-d: 00AFF818
4-b: 00AFF81C
5-b: 00AFF820
6-b: 00AFF824

Third Condition
e: 00AFF810
f: 00AFF81C
g: 00AFF810
```

3) 고찰

조건1에서 `h`와 `i`가 포인터 변수 `a`를, `j`가 포인터 변수 `c`를 가리키고 있으므로, 이 세 개의 변수는 더블 포인터 형으로 선언해야 하고, `g`는 2*3 크기의 배열 전체를 가리키고 있으므로, 배열 포인터로 선언해준다. 과제를 작성하면서 `h, i, j` 변수를 더블포인터가 아닌 단순한 포인터로 작성하여도 결과가 원하는 대로 출력되었는데, 이는 세 변수가 포인터형이 아닌, 배열의 요소를 직접 가리키고 있기 때문이었다. 이후 더블포인터로 수정하였다. 조건2에서는 각각 요소들의 주소 값 차이가 4인 것을 알 수 있다. 이는 `two_d_array`가 `int`배열이기 때문에 `int`형의 크기인 4바이트 차이가 나는 것이다. 마지막으로, 조건3에서는 행을 가리키는 포인터는 각행의 첫 번째 요소의 주소 값과 같다는 것과, 배열 전체를 가리키는 포인터는 첫 번째 행의 첫 번째 열의 요소의 주소 값과 같다는 것을 알 수 있었다.

3. Clockwise array

1) 문제 및 설명

이 프로그램은 사용자로부터 두 개의 정수를 입력 받아 해당하는 크기의 배열을 생성하고, 배열의 외부에서부터 시계방향으로 들어가며 숫자가 1씩 증가하도록 출력하는 프로그램이다. 이 문제를 해결하기 위해서 2차원 배열의 동적 할당을 이용했다.

프로그램이 실행되면 사용자로부터 두 개의 정수를 입력 받는다. 각각의 수는 변수에 저장되어 해당하는 크기의 2차원 배열을 동적 할당하게 된다. 이때 두 수가 문제에서 제시된 범위에 해당하지 않으면 오류 메시지를 출력하고 프로그램을 종료한다. 배열의 할당 이후에는 배열의 밖에서부터 시계방향으로 수를 채워 넣는다. x와 y변수를 이용하여 배열의 칸을 이동하고, while문 내부에서 두 개의 for문을 이용해 각각 가로로 이동, 세로로 이동한다. 오른쪽으로 이동, 아래쪽으로 이동을 +방향으로 놓고, direction을 이용하여 반대방향으로 이동도 제어할 수 있다. 이때 for반복문의 반복회수가 1씩 감소하는 점을 유의해야 한다. while문에서 숫자를 채우는 작업이 종료되면 배열을 모두 출력하고, 동적 할당한 배열을 해제한다.

2) 결과화면

Microsoft Visual Studio 디버그 콘솔	Microsoft Visual Studio 디버그 콘솔
Please input 2D array size: 7 3	Please input 2D array size: 8 8
1 2 3	1 2 3 4 5 6 7 8
16 17 4	28 29 30 31 32 33 34 9
15 18 5	27 48 49 50 51 52 35 10
14 19 6	26 47 60 61 62 53 36 11
13 20 7	25 46 59 64 63 54 37 12
12 21 8	24 45 58 57 56 55 38 13
11 10 9	23 44 43 42 41 40 39 14
	22 21 20 19 18 17 16 15

3) 고찰

처음 프로그램을 작성할 때 while반복문 내의 for문을 4개 이용하여 각 방향으로의 이동을 제어했는데, 두 개의 for문이 다른 두 개의 동작과 방향만 제외하고 동일하다는 점을 이용하여 반복문의 개수를 줄일 수 있었다. 이 프로그램에서는 큰 문제가 되지는 않았지만, 2차원 배열을 해제하는 것을 빼놓아 메모리 누수가 일어나기도 했었다. 동적 할당한 배열을 해제하지 않으면 메모리를 낭비하게 되기 때문에 더 이상 사용하지 않는다면 메모리를 해제해야 한다.

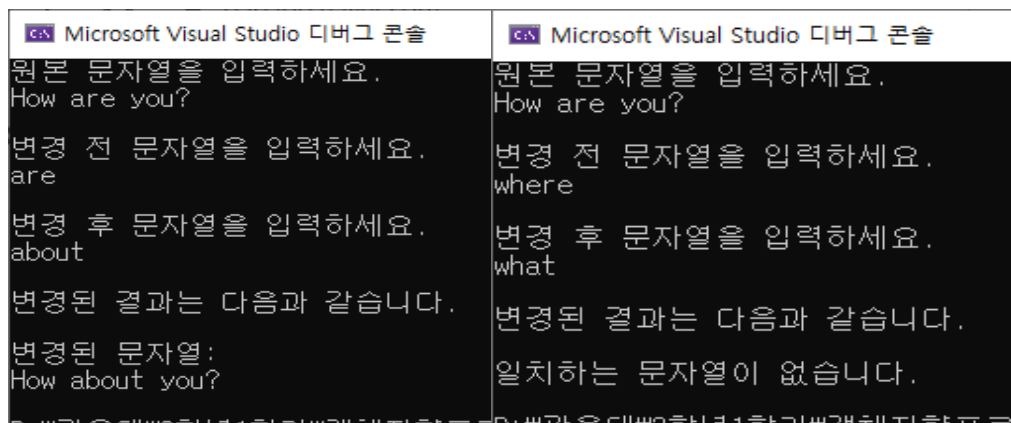
4. My strstr

1) 문제 및 설명

사용자로부터 세 개의 문자열을 입력 받아, 두 번째 문자열을 첫 번째 문자열에서 찾아 세 번째 문자열로 치환하는 동작을 하는 함수를 string function의 사용 없이 작성하는 문제이다. 변경할 문자를 저장할 문자열을 동적 할당하는 것과 일치하는 문자열을 찾고, 해당 부분을 제외하고 문자열에 남겨져야 하는 부분을 옮겨 저장하는 동작, 세 번째 문자열의 내용을 복사하는 동작이 시행되어야 한다. 이 문제를 해결하기 위해 세 개의 문자열의 크기를 세어서 이용하였다.

프로그램을 실행하면 사용자에게 세 개의 문자열을 입력 받는다. 이 문자열들을 모두 인자로 하는 함수 my_strstr에서 각각 문자열의 크기를 세어서 변수에 저장한다. 이후 두 번째 문자열의 요소를 첫 번째 문자열의 요소와 비교하여 완전히 일치하는 부분이 있는지 확인한다. 일치하는 부분이 없는 경우 NULL값을 반환한다. 일치하는 부분이 있으면 내용을 변경한 문자열을 저장할 공간을 할당하고, 첫 번째 문자열의 내용을 모두 복사한 뒤, 마지막에 NULL문자를 삽입한다. 두 번째 문자열과 세 번째 문자열의 크기를 고려해서 문자열이 변경 된 이후 위치가 바뀌는 글자를 이동시킨다. 마지막으로 세 번째 문자열의 내용을 새로 생성한 문자열에 복사하고, 완성된 문자열을 반환한다. 다시 main함수에서, my_strstr의 결과에 따라 오류메시지 혹은 변경된 문자열을 출력하고 프로그램을 종료한다.

2) 결과화면



3) 고찰

string 함수를 사용하지 않고 문자열을 처리 하는 과정이 번거로웠다. 반복문을 통해 문자열의 요소를 하나씩 고려해서 문제를 해결할 수 있었다. 두 번째 문자열의 크기와 세 번째 문자열의 크기에 따라 두 가지 경우로 나누어서 프로그램을 설계했었는데, 삼항 연산자(?:)를 이용해서 반복되는 내용을 줄이고, 코드를 단순화할 수 있었다.

5. The class of hospital patient information

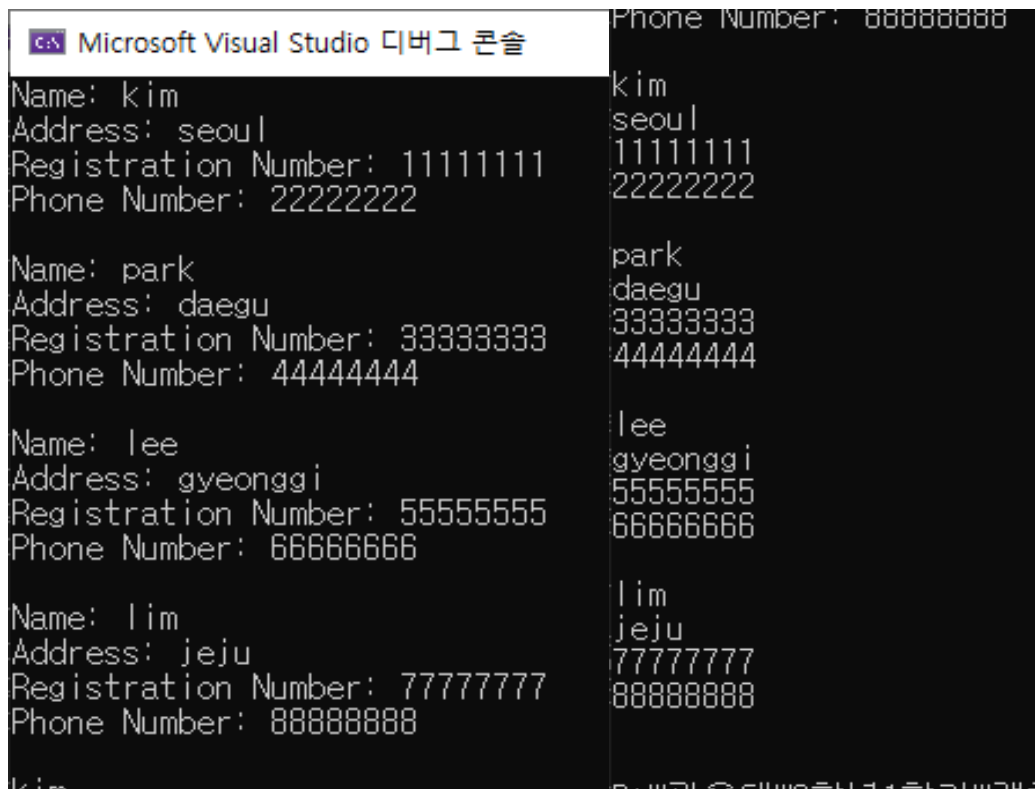
1) 문제 및 설명

PatientInfo클래스를 작성하고, 입력 받은 정보를 통해 4개의 객체를 생성해서 출력하는 문제이다. 클래스에 대한 기본적인 이해와 동적 할당에 대한 이해가 필요하다. 추가적으로, 동일한 PatientInfo클래스를 반복해서 생성하기 때문에 클래스 배열을 이용하여 문제를 해결하였다.

PatientInfo 클래스는 private영역에 이름과 주소를 저장할 char형 포인터변수와 주민번호와 전화번호를 저장 할 int형 변수를 각각 2개씩 가지고 있다. public영역에는 생성자, 소멸자와 함께 private영역의 정보를 반환해주는 함수를 총 4개 가진다. PatientInfo의 생성자는 이름과 주소정보가 저장된 char형 포인터와 주민번호와 전화번호가 저장된 int형 변수를 각각 2개씩 인자로 받는다. int형으로 전달된 정보는 바로 값을 복사하고, char*형으로 전달된 정보는 동적메모리 할당을 이용해 메모리 공간을 할당한 뒤, 내용을 복사하여 PatientInfo 객체를 생성한다. 객체가 소멸할 때는 소멸자에서 이름과 주소를 저장할 때 할당했던 메모리 공간들을 해제해준다.

프로그램이 실행되면 test case의 수만큼 클래스 포인터 배열을 생성하고, 사용자로부터 이름과 주소를 입력 받을 문자열을 동적 할당한다. for반복문을 이용해 사용자에게 정보를 입력 받아 test case의 수만큼 객체생성을 반복한다. 객체의 생성이 모두 끝나면 정보입력을 위해 할당했던 메모리를 해제하고, 객체에 저장된 정보를 PatientInfo의 메소드를 이용해 출력한다. 이후 메모리를 해제하고 프로그램을 종료한다.

2) 결과화면



```
Microsoft Visual Studio 디버그 콘솔
Name: kim
Address: seoul
Registration Number: 11111111
Phone Number: 22222222
kim
seoul
11111111
22222222

Name: park
Address: daegu
Registration Number: 33333333
Phone Number: 44444444
park
daegu
33333333
44444444

Name: lee
Address: gyeonggi
Registration Number: 55555555
Phone Number: 66666666
lee
gyeonggi
55555555
66666666

Name: lim
Address: jeju
Registration Number: 77777777
Phone Number: 88888888
lim
jeju
77777777
88888888
```

3) 고찰

프로그램의 설계 단계에서 네 개의 클래스를 각각 생성하고, 메소드의 사용도 따로 진행해서 코드가 과도하게 길어졌었다. 같은 작업을 반복하는 특성을 이용하여 클래스의 배열을 고려하였지만, 클래스 배열은 선언과 동시에 생성자가 동작하기 때문에 사용자가 입력한 값으로 초기화하려면 다른 함수를 선언하여 따로 초기화 해야 했다. 그렇다면 코드의 길이와 복잡성에서 큰 이득이 없으므로, 다른 방법을 생각해야 했다. 결국 찾아낸 방법은 클래스의 포인터 배열을 이용하는 것 이었다. 포인터 배열을 이용하기 위해 PatientInfo를 동적 할당하였다. 또한 사용자로부터 이름과 주소 정보를 입력 받는 경우, char배열을 통해 입력 받으면 문자열의 내용을 복사하는 과정에서 오류가 생겨 마찬가지로 동적 할당을 이용하였다. 정보를 출력하는 과정에서도 포인터를 이용하였기 때문에 연산자의 변화가 있었다. 메모리를 해제하는 과정에서도 여러 오류가 만들었다. 클래스 포인터 배열은 동적 할당하지 않았지만, delete[]를 이용했다가 오류가 발생했고, 소멸자를 잘못 작성하여 프로그램이 정상적으로 종료되지 않는 문제가 있었다. 앞의 문제에서도 동적으로 할당한 메모리의 해제에서 문제를 겪었기 때문에 관련 부분을 다시 살펴보는 계기가 되었다.

6. 369 Game

1) 문제 및 설명

1부터 사용자가 입력한 숫자까지, 숫자에 3,6,9가 포함되어있을 때 포함된 개수만큼 '!'를 출력하는 프로그램이다. ThreeSixNine 클래스를 이용하고, 메소드를 정의해야 했다. 그렇기 때문에 클래스에 대한 기초적인 이해가 필요한 문제였다. 369게임 자체는 단순하게 구현 할 수 있었다.

ThreeSixNine 클래스는 private으로 선언한 내용은 없고, public에 3개의 함수로 구성하였다. print369 메소드는 for문을 이용하여 출력할 수를 지정해준다. for문 내부에서 호출된 find369 메소드는 i에 3,6,9가 포함되어있는지 확인하고, 출력해야 하는 느낌표의 개수를 반환한다. i와 find369에서 반환한 느낌표의 개수를 인자로 받는 print_i는 느낌표의 개수가 하나 이상이라면 느낌표를 개수만큼 출력하고, 출력해야 하는 느낌표가 없다면 i를 출력한다. 다시 print369함수로 돌아와 '\t'을 출력하고, 한 줄에 10개의 수를 출력했다면 다음 줄로 넘어간다.

프로그램이 실행되면 사용자에게 수를 입력받아 1~300사이 지정된 구간에 속하는지 확인 한 후, 범위 밖인 경우 오류메시지를 출력하고 종료한다. 범위내의 수가 입력된 경우 game 객체의 print369 메소드를 num을 인자로 넘겨주어 호출한 뒤 출력이 완료되면 프로그램이 종료된다.

2) 결과화면

```
Microsoft Visual Studio 디버그 콘솔
input your number
222
1      2      !      4      5      !      7      8      !      10
11     12     !     14     15     !     17     18     !     20
21     22     !     24     25     !     27     28     !     30
!      !      !!     !      !      !!     !      !      !!     !
41     42     !     44     45     !     47     48     !     50
51     52     !     54     55     !     57     58     !     60
!      !      !!     !      !      !!     !      !      !!     !
71     72     !     74     75     !     77     78     !     80
81     82     !     84     85     !     87     88     !     90
!      !      !!     !      !      !!     !      !      !!     !
101    102    !    104    105    !    107    108    !    110
111    112    !    114    115    !    117    118    !    120
121    122    !    124    125    !    127    128    !    130
!      !      !!     !      !      !!     !      !      !!     !
141    142    !    144    145    !    147    148    !    150
151    152    !    154    155    !    157    158    !    160
!      !      !!     !      !      !!     !      !      !!     !
171    172    !    174    175    !    177    178    !    180
181    182    !    184    185    !    187    188    !    190
!      !      !!     !      !      !!     !      !      !!     !
201    202    !    204    205    !    207    208    !    210
211    212    !    214    215    !    217    218    !    220
221    222
```

3) 고찰

ThreeSixNine 클래스에 print369 메소드 하나만을 이용하여 같은 동작을 하는 프로그램을 작성했지만, 하나의 메소드가 처리하는 일이 너무 많아서 총 3개의 메소드로 분리하였다. 클래스 내부에서 공통으로 사용하는 i 같은 경우는 클래스의 private에 정의하여 이용하는 방법도 고려할 수 있을 것이다.

7. Situation: sells a mask in pharmacy

1) 문제 및 설명

이 문제는 약국에서 마스크를 구매하는 상황을 Pharmacist 클래스와 Buyer 클래스를 이용하여 나타내는 것이다. 이 문제를 해결하기 위해서는 클래스에 대한 이해가 필요하다. 프로그램은 5개의 기능을 포함하고 있다. 구매자가 약사에게 마스크의 재고, 가격을 묻거나, 구매자가 자신의 마스크 보유량, 지갑의 잔액을 확인하고, 약사에게서 마스크를 구매하는 (소비자에게 마스크를 판매하는) 기능이다. 구매자가 자신의 마스크 보유량과 지갑의 잔액을 확인하는 동작은 pharmacy 클래스와 상호작용하지 않고, 나머지 동작은 pharmacy 클래스와 상호작용 한다.

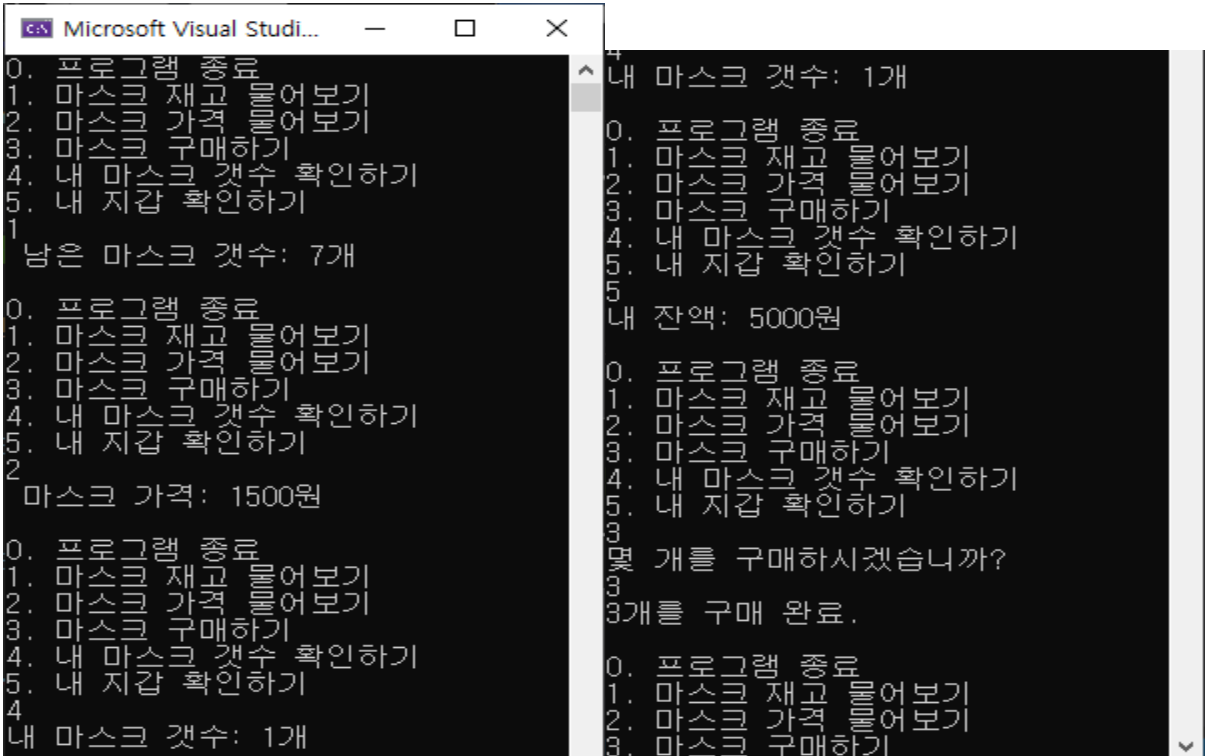
Buyer 클래스에는 현재 마스크 보유량과 지갑의 잔액을 뜻하는 두 개의 변수가 private에 저장되고, 각각 기능에 맞는 메소드를 가지고 있다. func1은 약사에게 재고를 물어보는 기능, func2는 가격을 묻고, say_hold는 현재 자신의 마스크 보유량을, say_wallet은 지갑의 잔액을 콘솔창에 출력한

다. how_many는 마스크를 구입할 때 약사에게 구입을 희망하는 수량을 전달하고, give_money는 자신이 보유한 잔액을 약사에게 모두 전달한다. 마지막으로 receive는 구매완료한 마스크와 잔돈을 약사에게서 돌려받는다.

Pharmacist 클래스에는 마스크 재고와 가격을 뜻하는 두 개의 변수가 private에 저장되어있고, 재고를 출력하는 say_stock, 가격을 출력하는 say_price 메소드와 구매자에게 마스크를 판매하는 상황을 총괄하는 sell 메소드를 포함한다. sell 메소드는 구매자에게 몇 개의 마스크를 구매할 것인지 물어보고, 구입을 희망하는 수량을 전달받고, 구매자가 보유한 잔액을 모두 건네받아 구매자가 원하는 수량만큼 마스크를 구매할 수 있는지 여부를 판단하고, 결과에 따라 마스크와 잔돈을 건네주거나, 잔액, 재고 혹은 양쪽 다 부족하다는 메시지를 콘솔창에 출력한다.

main함수에서는 Pharmacist 클래스와 Buyer 클래스에 초기값을 주어 객체를 생성하고, while문에서 사용자에게 메뉴를 입력 받아 해당하는 메소드를 호출하여 동작한다. 이때 사용자가 메뉴를 잘못 입력한 경우 오류 메시지를 출력하고, 다시 입력받는다.

2) 결과화면



```
Microsoft Visual Studi...
0. 프로그램 종료
1. 마스크 재고 물어보기
2. 마스크 가격 물어보기
3. 마스크 구매하기
4. 내 마스크 갯수 확인하기
5. 내 지갑 확인하기
1
남은 마스크 갯수: 7개

0. 프로그램 종료
1. 마스크 재고 물어보기
2. 마스크 가격 물어보기
3. 마스크 구매하기
4. 내 마스크 갯수 확인하기
5. 내 지갑 확인하기
2
마스크 가격: 1500원

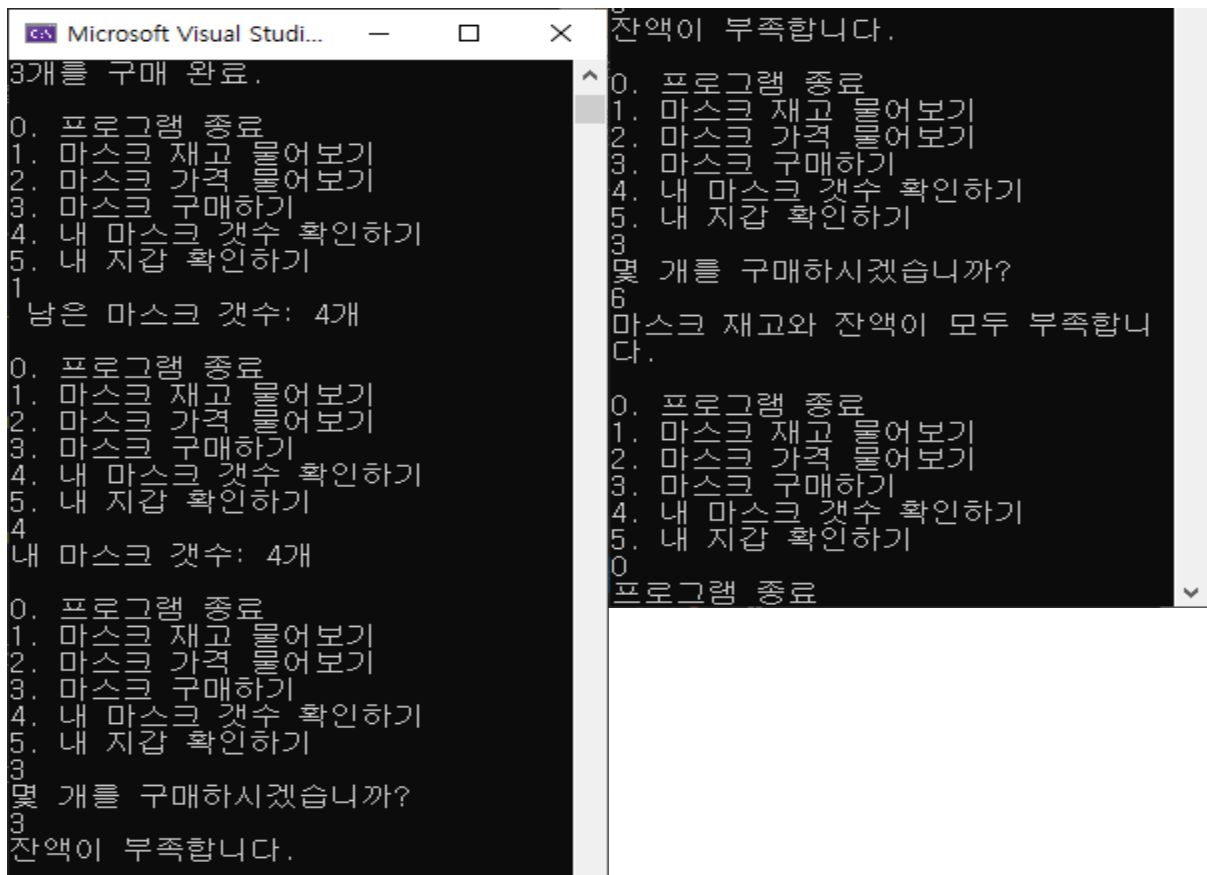
0. 프로그램 종료
1. 마스크 재고 물어보기
2. 마스크 가격 물어보기
3. 마스크 구매하기
4. 내 마스크 갯수 확인하기
5. 내 지갑 확인하기
4
내 마스크 갯수: 1개

4
내 마스크 갯수: 1개

0. 프로그램 종료
1. 마스크 재고 물어보기
2. 마스크 가격 물어보기
3. 마스크 구매하기
4. 내 마스크 갯수 확인하기
5. 내 지갑 확인하기
5
내 잔액: 5000원

0. 프로그램 종료
1. 마스크 재고 물어보기
2. 마스크 가격 물어보기
3. 마스크 구매하기
4. 내 마스크 갯수 확인하기
5. 내 지갑 확인하기
3
몇 개를 구매하시겠습니까?
3
3개를 구매 완료.

0. 프로그램 종료
1. 마스크 재고 물어보기
2. 마스크 가격 물어보기
3. 마스크 구매하기
```



3) 고찰

절차지향 프로그래밍을 해오다가 클래스 객체를 이용하여 객체지향 프로그래밍을 한 것은 이번 실습이 처음이었는데, 클래스를 이용해서 상황을 구성하는 것이 흥미로웠다. 이 프로그램의 경우 클래스의 private에 있는 정수를 콘솔 창에 출력하거나 수정하는 등, Pharmacist 클래스와 Buyer 클래스의 기능이 겹치는 부분이 있기 때문에, 부모 클래스를 만들고 상속을 받아 프로그램을 작성하는 것도 고려해볼 수 있을 것 같다.

8. Exchange

1) 문제 및 설명

이 문제는 사용자의 입력에 따라 돈을 환전하는 프로그램이다. 그러나 주의할 점은 사용자의 입력이 문자열로 이루어진다는 것이다. 이 문제를 해결하기 위해 문자를 처리하는 방법과 클래스의 상속에 대한 이해가 필요하다.

부모클래스가 되는 Exchange 클래스는 private에 환전 할 돈과 원하는 화폐의 종류, 환전할 때

사용되는 수치가 저장된 변수가 있고, 환전할 돈과 원하는 화폐의 종류를 반환하는 2개의 메소드와 환전할 때 사용하는 수치를 변경하는 메소드 2개, 환전 결과를 반환하는 메소드 2개가 public에 선언되어있다.

자식클래스는 부모클래스인 Exchange 클래스의 생성자를 사용한다. 자식클래스들은 Exchange 클래스의 메소드를 이용하여 Exchange 클래스의 private에 존재하는 변수를 수정하는 메소드 하나와, 환전의 결과를 double형으로 반환하는 메소드 하나를 가지고 있다. 앞의 메소드는 Exchange 클래스의 메소드를 오버라이딩 한 것이다.(SetMulti / SetDenomi)

자식클래스 중 ToKRW 클래스는 다른 클래스와 환전에 사용하는 방법이 조금 다르다. 처음 입력된 금액을 모두 KRW로 변환한 다음 각기 다른 화폐로 최종 변환하는 방법을 이용했기 때문이다. 따라서 ToKRW클래스는 모든 화폐에 대한 처리를 하지만, 다른 자식클래스들은 KRW에 대한 처리만 한다. ToKRW클래스는 입력된 화폐 종류에 따라 해당하는 비율의 값을 곱해서 환전하지만, 다른 클래스들은 KRW의 값에 비율을 나누어서 환전한다. 공통적으로 strcmp 함수를 이용하여 화폐의 종류를 찾고, 화폐 종류와 일치하지 않는 입력오류에 대해서는 음수 값을 반환하여 오류처리를 가능하게 한다.

2) 결과화면

```
★Currency Converter★
e.g. [Price] KRW to USD

7000 KRW to EUR
5 EUR

Would you like to continue? (Y/N)
-

★Currency Converter★
e.g. [Price] KRW to USD

400 CNY to CNY
400 CNY

Would you like to continue? (Y/N)
N
Bye!

D:\광운대\2학년1학기\객체지향프로그래밍\실습2\1_B_2019202009_서여지_ver2\assignment2-8\assignment2-8\Debug\assignment2-8.exe(프

★Currency Converter★
e.g. [Price] KRW to USD

5000 JPY to USD
46 USD

Would you like to continue? (Y/N)
-

★Currency Converter★
e.g. [Price] KRW to USD

50 KRW TO 000
Input error

D:\광운대\2학년1학기\객체지향프로그래밍\실습2\1_B_2019202009_서여지_ver2\assignment2-8\assignment2-8\Debug\assignment2-8.exe(프
```

3) 고찰

화폐의 종류가 여러가지 이고, 환전할 수 있는 경우의 수는 더욱 다양해서 많은 수의 메소드를 정의해야 한다고 생각했었는데, 실습강의 내용에서 모든 화폐를 1차적으로 KRW로 변환하면 문제를 간단하게 해결할 수 있다는 것을 알았다. 언뜻 보기엔 중간에 KRW로 변환하는 과정이 추가되어 문

제를 더 복잡하게 푸는 것 같지만, 전체 코드를 작성하는 것에서 큰 이득을 볼 수 있었다. 문제의 해법을 찾을 때, 대부분 문제 풀이의 단계를 줄이는 방향으로 생각했었는데, 문제 풀이의 단계는 약간 늘어나도 한 번 정의한 변수나 함수를 여러 번 사용하는 방법을 찾는다면 프로그램의 작성과 유지에 유리할 수 있다는 것을 알게 되었다.

9. Pokémon: A Great War

1) 문제 및 설명

이 문제는 체스나 장기와 유사한 게임을 제작하는 문제이다. 전체 9*9 크기의 게임판에서 두 명의 플레이어는 각각 4개의 말(포켓몬)을 갖는다. 이 말들은 각각 3의 체력과 2의 공격력을 가지고 있으며, 서로 상성관계가 있어서 더 강하거나, 약한 공격을 할 수 있다. 플레이어는 각자의 말을 이용해서 상대방의 말의 체력을 모두 0이하로 만들어 쓰러트리는 것을 목표로 한다. 이 문제를 푸는 것에는 클래스와 상속에 대한 이해가 필요했다. 특히 순수 가상함수와 관련한 조건이 있기 때문에 관련 지식이 필요하다. 이 문제는 대략적인 클래스의 형태와 함께 일부분을 제외하고 이미 완성된 main함수를 제시해주었기 때문에, 제시된 코드를 분석하여 특정 클래스가 가져야 할 정보와 메소드가 수행해야 하는 동작을 파악 하는 것이 문제풀이의 시작이 되었다.

main함수는 가장 먼저 사용자에게 이름을 입력 받고, 입력 받은 정보를 바탕으로 player클래스를 동적 할당하여 객체를 생성 한다. 플레이어 동적 할당 부분이 작성되지 않았기 때문에 player클래스의 생성자를 고려하며 동적 할당 부분을 작성하였다. 이후 field를 출력하여 현재 플레이어의 말들이 위치한 모습을 보이고, 순서를 돌아가며 사용자에게서 명령을 입력 받아 동작한다. 명령은 크게 두 가지로, 말을 이동하는 것과 지정된 범위에 공격을 하는 것이다. 말의 이동은 방향을 입력 받아 pokemon객체의 move 메소드에서 이루어지고, 공격도 방향을 입력 받아 player객체의 battle 메소드에서 이루어진다. 한 플레이어의 말들이 모두 0이하의 체력을 갖게 된다면 게임의 진행이 종료되고, 종료 메시지를 출력하고 두 개의 player 객체를 해제한 뒤 종료된다.

main함수를 정상적으로 작동시키기 위해 player 클래스를 작성하였다. player클래스를 private에 pokemon클래스의 상속을 받은 네 개의 자식클래스 Charmander, Squirtle, Bulbasaur, Pikachu의 포인터 값을 갖는다. 각각의 player가 가지고 있는 말을 나타낸 것이다. 또한 플레이어가 입력한 이름 정보와, 플레이어의 순서를 결정하는데 사용된 숫자자료가 private에 선언되어있다. player클래스는 public영역에 여러 개의 메소드를 갖는다. 그 중 생성자는 두 가지 형태로 사용되었다. 사용자가 이름으로 default를 입력했을 경우, 인자로 정수 하나만을 전달하여 기본 이름인 player1, plater2가 이름 정보에 저장된다. 사용자가 다른 이름을 입력한 경우, 하나의 정수와 입력 받은 이름을 이용하여 클래스를 생성한다. 두 생성자 모두 네 개의 말에 해당하는 클래스를 동적할당 한다. 이 객체 생성에 필요한 정보들은 모두 고정되어있어 매 실행에 같은 값으로 선언된다. 소

멸자는 객체가 소멸하는 시점에 플레이어가 가지고 있는 말의 정보를 통해 플레이어가 승리했는지 혹은 패배했는지 결과를 출력한 뒤 남은 말의 메모리를 해제한다. main함수에 플레이어의 이름을 받아 출력하는 부분이 있기 때문에 이름 정보를 반환하는 getName메소드와 player가 가지고 있는 말의 포인터를 반환하는 메소드가 각각 필요하고, 말이 체력이 0이되어 쓰러진 경우 말의 메모리를 해제할 때 오류를 막기 위해 player가 가진 말들의 포인터를 NULL로 지정해주는 메소드들을 가지고 있다. 이 외에도 player 클래스는 세 개의 메소드를 더 가지고 있다. 포켓몬이 특정 위치에 존재하는 경우 포켓몬의 이름을 출력하는 print메소드와 자신이 보유한 모든 포켓몬의 체력을 출력해주는 HPprint메소드, 게임에서 공격을 실행하는 battle 메소드 이다. 이 중 battle 메소드는 공격하는 포켓몬과 공격받는 포켓몬의 상황에 따라 pokemon 클래스의 Attack 메소드를 호출하여 공격의 성공여부를 확인하고, 마찬가지로 pokemon 클래스에 있는 damage 메소드를 이용하여 공격받은 말의 체력을 깎는다. 만약 공격받은 말의 체력이 0 이하라면 field에서 해당하는 말을 지우고, 메모리를 해제한다.

pokemon 클래스는 각각 말의 종류에 따라 선언된 클래스들을 자식클래스로 가지고 있는 부모클래스이다. 즉, Charmander, Squirtle, Bulbasaur, Pikachu 클래스는 모두 pokemon 클래스를 상속받고 있다. pokemon 클래스는 private에 체력, 말의 소유자, x위치, y위치를 뜻하는 정수와 상성을 의미하는 문자 하나를 가지고 있다. public에는 생성자와 소멸자, private의 정보를 반환하는 메소드와 말이 받은 피해량에 따라 체력을 깎는 메소드, 말이 이동했을 때 x위치와 y위치를 새로 지정하는 클래스를 가지고 있고, 이 외에 세 개의 메소드를 더 가지고 있다. 말의 이동을 관리하는 메소드와, 상성에 따라 다른 피해량을 입히는 것을 표현한 메소드, 마지막으로 포켓몬마다 다른 패턴으로 공격하는 것을 표현하는 순수 가상 함수이다. 가상함수의 정의는 자식클래스에 각각 정의되어 있다.

네 개의 자식클래스 Charmander, Squirtle, Bulbasaur, Pikachu는 부모클래스의 생성자와 자신의 생성자를 모두 이용하는 방법으로 초기화된다. 소멸자는 따로 작성하지 않았고, 부모클래스의 소멸자를 사용한다. 클래스 내부에서 별도로 처리해야 하는 것이 없기 때문이다. 메소드의 이름은 같지만, 자식클래스마다 내용이 모두 다른 Attack메소드가 존재한다. 이 메소드는 게임에서 다른 말을 공격하는 상황을 표현하고 있으며, 자신이 공격 가능한 범위에 다른 포켓몬이 있는지를 확인한다. 공격을 받을 수 있는 말이 있는 경우 참을, 없는 경우 거짓을 반환한다. 이 문제에서 각각의 말들은 정해진 형태의 공격을 오른쪽과 왼쪽 두 가지 방향으로 사용할 수 있다. 공격의 방향을 고려하여 공격할 수 있는 상대의 말을 확인해야 하기 때문에 인자로 방향을 뜻하는 값과, 특정 포켓몬의 위치값을 받았다. 공격을 하는 자신의 기준에서 공격이 가능한 각각의 칸에 player의 메소드 battle에서 분류된 특정 포켓몬이 위치하는지를 확인하는 방법을 사용하였다.

2) 결과화면

```

player1 name : default
player2 name : .

C1 C2
S1 S2
P1 P2
B1 B2
player1 Turn
C: 3 S: 3 B: 3 P: 3
1. Move 2.Battle : 1
Pokemon Select(1.C 2.S 3.B 4.P) : 4
Direction(U, D, R, L) : r

C1 C2
S1 S2
P1 P2
B1 B2
player2 Turn
C: 3 S: 3 B: 3 P: 3
1. Move 2.Battle : 1
Pokemon Select(1.C 2.S 3.B 4.P) : 1
Direction(U, D, R, L) : r
wrong direction!
Direction(U, D, R, L) : _

C1 C2
S1 S2
P1 P2
B1 B2
player1 Turn
C: 3 S: 3 B: 3 P: 3
1. Move 2.Battle : 2
Pokemon Select(1.C 2.S 3.B 4.P) : 4
Attack Direction(R, L) : r
player1의 피카츄의 스파크!

C1 C2
S1 S2
P1 P2
B1 B2
player2 Turn
C: 3 S: 3 B: 1 P: X
1. Move 2.Battle :

C1 C2
S1 S2
P1 P2
B1 B2
player1 Turn
C: 3 S: 3 B: 3 P: X
1. Move 2.Battle : 2
Pokemon Select(1.C 2.S 3.B 4.P) : 2
Attack Direction(R, L) : r
player1의 꼬부기의 물대포!
파이리는 쓰러졌다!

*** GAME END ***
player1 win!
player2 lose.
  
```

3) 고찰

클래스가 익숙하지 않은 상황에서 복잡해 보이는 문제를 풀어야 한다는 것이 막막하게 느껴졌지만, main함수의 내용을 따라 각각의 메소드가 가져야 하는 역할을 파악하고, 하나씩 구현해 나가는 과정이 재미있었다. 처음부터 코드를 구성하는 것이 아니라 전체적인 흐름이 제시되어 있다는 점이 특이했다. 사용되는 객체와 메소드가 다양하고, 문제가 복잡해서 문제를 몇 개의 부분과 단계로 나누고, pokemon클래스를 상속받는 자식클래스도 처음에는 하나만 이용해서 전체적인 문제를 파악하여 해결하였다. 하나의 자식클래스를 완성 한 이후에 다른 자식클래스를 구성하는 것은 어렵지 않았고, 기존 객체와 상호작용에서 발생한 몇 가지 오류를 해결한 다음에는 대부분의 기능이 정상적으로 작동하였다.