



Object-Oriented Programming Report

Assignment 1-2

Professor	Donggyu Sim
Department	Computer engineering
Student ID	2020202031
Name	Jaehyun Kim

Program 1

□ 문제 설명

두개의 수를 입력 받습니다. 첫번째 수의 맨 앞자리 수가 입력된 두 수에 몇 개 포함돼 있는지 세고, 그 수와 개수를 차례로 출력합니다. 그 다음 첫번째 수의 두번째 자리 수가 몇 개 포함돼 있는지 세고 마찬가지로 그 수와 개수를 출력합니다. 같은 방식으로 첫번째 입력된 수의 끝까지 마무리했다면 두번째 입력된 수도 똑같이 실행해줍니다. (단, 이미 중복되는 수는 넘어가도록 합니다.)

입력 받은 두 수를 하나의 배열에 공백 없이 저장해줍니다. for 문으로 배열 전체를 돌면서 중복된 수가 아닐 경우 record 배열에 그 수를 저장하고 그 수와 개수를 출력합니다. 중복일 경우 continue 를 사용해 경우를 pass 해줍니다.

□ 결과 화면

```
두 수 입력(공백으로 구분): 79 897687543217
749282615141312111
```

앞에서부터 입력된 수를 중복되지 않게 나열하면 7 9 8 6 5 4 3 2 1 이고 이 수들의 개수를 세면 4 2 2 1 1 1 1 1 1 개 이므로 749282615141312111 이 출력된다.

```
두 수 입력(공백으로 구분): 897687543217 89
839273615141312111
```

마찬가지로 8 9 7 6 5 4 3 2 1 이 각각 3 2 3 1 1 1 1 1 1 개 이므로 839273615141312111 이 출력된다.

두 수 입력(공백으로 구분): 897687543217 897687543217
849276625242322212

마찬가지로 8 9 7 6 5 4 3 2 1 이 각각 4 2 6 2 2 2 2 2 개 이므로
849276625242322212 가 출력된다.

□ 고찰

두 문자를 입력 받을 때, 공백을 기준으로 두 수를 구분하게 되는데, 이때 공백까지 하나의 배열에 입력을 받으려고 했으나, cin 함수가 공백을 기준으로 전후의 문자를 따로 입력 받는 메커니즘을 가지고 있다는 점을 간과하여 처음 입력 받은 문자만으로 프로그램이 돌아가는 오류를 범했습니다. 그래서 두 문자를 각각의 배열에 저장한 후 하나의 배열에 이어 붙이는 과정을 거쳤습니다.

Program 2

□ 문제 설명

2 진수 데이터를 전송할 때 Sender 에서 암호화한 후 Transmission_Chnnel 를 거쳐 Receiver 로 전달될 때 에러가 있는지 없는지 검사하는 일련의 과정을 재현해보는 문제입니다. 2 진수 데이터는 char 형 배열로 표현합니다.

XOR 비트연산자 ^는 비트단위로 XOR 연산을 진행합니다. 예를 들어,

5=00000101

9=00001001

→00001100 = 12

OR 비트연산자 |, AND 비트연산자 & 등이 같은 방식으로 비트연산을 진행합니다.

SHIFT 비트연산자 <<는 모든 비트를 왼쪽으로 이동하고 오른쪽에는 0 을 채우는 비트연산을 진행합니다. 예를 들어

$$5 << 2$$

$$5 = 00000101 \rightarrow 00010100 = 20$$

결국 <<연산을 한번 진행할 때마다 2 배가 되는 것을 확인할 수 있습니다.

저는 Sender, Transmission_channel, Receiver, XOR 함수를 구현했습니다. XOR 함수는 배열 포인터를 인자로 받아, 맨 앞 5 자리를 int 형으로 저장하여 0b11011 과 XOR 연산을 진행하고 그 결과값을 다시 배열 앞 5 자리에 저장해줍니다. Sender 는 사용자로부터 입력받은 data 를 인자로 받아 copy 배열에 복사해줍니다. 그리고 포인터 ptr 로 copy 를 가리키고, XOR 함수에 ptr 을 전달합니다. 그 결과 copy 의 맨 앞쪽에는 0 이 여러 개 생길 것이므로 ptr 이 0 을 지나치고 가장 앞에있는 1 을 가리키도록 한 후 XOR 함수를 실행시킵니다. 이 과정을 반복하다보면 ptr 이 FCS 를 가리키게 되고 이를 data 에 이어붙인 후 Transmission_channel 에 전달합니다. Transmission_channel 에서는 rand()를 활용하여 확률에 따라 0 을 1 로 1 을 0 으로 바꿔줍니다. Receiver 에서는 Sender 와 같은 방식으로 XOR 연산을 하되 나머지 값이 0000 이 아닐 경우 에러가 있음을 출력해줍니다.

□ 결과 화면

```
Data: 110011001111
Coded frame: 1100110011111111
Received frame: 1100110011111111
Reconstructed data: 110011001111
No detected error
```

Original data 가 Sender 에서 FCS "1111"이 붙은 채로 Transmission 으로 전달되고, 전달 과정에서 아무런 에러도 발생하지 않았다. 그대로 Receiver 로 전달되어 XOR 연산을 진행해보니 나머지가 0000 이었기에 error 가 없음을 표시하는 모습이다.

```
Data: 110011001111
Coded frame: 1100110011111111
Received frame: 1110111011111111
Reconstructed data: 111011101111
Receiver has detected error
```


Original data 가 Sender 에서 FCS "1111"이 붙은 채로 Transmission 으로 전달되고, 3, 7 번째 비트에 에러가 발생했다. 그대로 Receiver 로 전달되어 XOR 연산을 진행해보니 나머지가 0000 이 아닌 것이 나와서 error 를 표시하는 모습이다.

□ 고찰

- char 형 배열로 정수 0 과 1 을 저장하여 비트와 유사하게 구현을 했습니다. 이 배열을 출력하는 과정에서 `cout<<data[i]+'0';` 하면 문자'0' 혹은 '1'이 출력될 것으로 예상했습니다. 하지만 문자 '0'과 '1'의 아스키코드값인 48, 49 가 출력됐습니다. 여기에 대해 고찰을 해보니, char 형 `data[i]`와 char 형 '0' 사이의 연산은 int 형으로 자동형변환이 되어 연산이 진행되기 때문에 int 형으로 출력이 된 것으로 판단하여, `cout<<(char)(data[i] + '0');` `data[i]+'0'`을 char 형으로 강제 형 변환하여 출력했습니다.
- `rand()` 함수의 범위는 0~32767 입니다. 이를 %100 연산하면 0~99 까지 100 개의 수만 사용할 수 있기에 확률을 적용할 수 있습니다. 그러나 $32767\%100=67$ 이므로 0~67 까지는 68~99 보다 나올 확률이 높습니다. 그러나 그 차이가 미미하여 `if(rand()%100 < 95)`로 확률에 따른 error 발생을 구현했습니다.

Program 3

□ 문제 설명



사용자로부터 파일명, 조건을 받아올 개수, 조건들을 입력 받은 후, 파일명에 맞는 파일을 열어서 그 안에 있는 문자열들을 조건들과 비교하여 조건에 부합하는 문자열들만 출력하는 문제입니다.

저는 '*'과 '?' 중 '*'이 핵심이 되는 문제라고 생각했습니다.('*'은 대신할 수 있는 문자의 개수에 제한이 없기 때문) 그래서 먼저 '*'을 기준으로 조건 문자열을 split 한 후, split 된 조건을 문자열과 비교하는 방식을 택했습니다. Split 된 조건 안에 '?'가 포함되었을 가능성을 생각하여 strncmp 를 사용하는 것이 아닌, '?'의 경우 아무 문자와도 매칭이 되도록 Qcompare 라는 함수를 만들어서 사용했습니다.

□ 결과 화면

```
Input file name : filename_list.txt
5
hello*
oop_assignment?.zip
*
*****ello***
o???assi??ment*

hello_world.cpp
hello.txt
oop_assignment1.zip
oop_assignment2.zip
oop_assignment3.zip
hello_world.cpp
oop_assignment1.zip
oop_assignment2.zip
hello.txt
oop_assignment3.zip
hello_world.cpp
hello.txt
oop_assignment1.zip
oop_assignment2.zip
oop_assignment3.zip
```

hello* 조건에 따라 "hello_world.cpp", "hello.txt"가 출력

oop_assignment?.zip 조건에 따라 "oop_assignment1.zip", "oop_assignment2.zip"
 , "oop_assignment3.zip"이 출력

*조건에 따라 모든 파일 "hello_world.cpp", "oop_assignment1.zip",
 "oop_assignment2.zip", "hello.txt", "oop_assignment3.zip"이 출력

*****ello*** 조건에 따라 "hello_world.cpp", "hello.txt"가 출력

o???assi??ment* 조건에 따라 "oop_assignment1.zip", "oop_assignment2.zip"
 , "oop_assignment3.zip"이 출력

□ 고찰

미리 대략적인 설계를 한 후 프로그램을 짜야한다는 것을 느꼈습니다. 저 같은 경우는 비교적 복잡한 '*'조건을 먼저 구현하면 '?'는 어떻게는 구현할 수 있을거라 생각하고 '*'를 먼저 구현했는데, '?'를 고려하지 않고 작성한 코드에 '?'를 적용하려고 하니 어디부터 손을 대야할지, 감조차 잡히지 않았습니다. 다행히도 아이디어가 떠올라 잘 수습했지만, 앞으로 코딩할 때, top-down 방식으로 하려고 노력해야겠다는 생각을 했습니다.

Program 4

□ 문제 설명

파일형식이 .raw 인 그림의 각 픽셀을 2 차원 배열에 입력 받고, 사용자로부터 입력 받은 좌표에 맞게 자르고, 상하좌우로 뒤집은 배열을 .raw 파일에 출력하는 문제입니다.

먼저 horizontal 과 vertical 함수는 512x512 배열을 동적 할당하고, fopen 을 통해 읽어온 파일의 픽셀 값을 배열에 저장합니다. 그리고 horizontal 은 좌우 대칭되는 인덱스를 swap 해줌으로서 좌우를 반전시키고, verticaldms 상하 대칭되는 인덱스를 swap 해줌으로서 상하를 반전시킵니다.

Crop 은 좌상단, 우하단 좌표로 높이와 너비를 계산하여 그에 맞는 크기의 배열을 할당 받고, 입력 받은 좌표에 해당하는 original 배열을 복사합니다.





Crop, horizontal, vertical 함수에서 파일을 출력할 때, 파일 이름을 조건에 맞게 바꿔주기 위해서 문자열을 복사하는 strcpy, 문자열을 이어붙이는 strcat 을 사용했습니다.

Crop 은 사용자로부터 좌표의 입력을 문자열로 받기 때문에 문자열 숫자를 정수로 바꾸주는 atoi 와 정수를 문자열 숫자로 바꾸주는 itoa 함수를 직접 구현해서 사용했습니다.

□ 결과 화면

```
Couple_512x512_yuv400_8bit.raw
1st coordinate : 100, 100
2nd coordinate : 355, 455
```

위의 입력을 통해 cropped 파일의 크기는 256x356 임을 계산할 수 있다.

 Couple_512x512_yuv400_8bit	2023-03-23 오후 3:19	RAW 파일	256KB
 Couple_512x512_yuv400_8bit_cropped_256x356	2023-03-24 오전 12:14	RAW 파일	89KB
 Couple_512x512_yuv400_8bit_horizontalflip	2023-03-24 오전 12:14	RAW 파일	256KB
 Couple_512x512_yuv400_8bit_verticalflip	2023-03-24 오전 12:14	RAW 파일	256KB

Cropped 파일이름을 보면 크기가 256x356 으로 잘 crop 됐음을 확인할 수 있다.

또한, 원본 파일, cropped 파일 horizontal 파일, vertical 파일이 잘 생성 됐음을 확인할 수 있다.

원본파일



cropped 파일



horizontal 파일



vertical 파일



□ 고찰

2 차원배열을 동적 할당하는 `memory_alloc2D` 의 구조가 처음보는 형식이었습니다. 보통 이중포인터를 할당하고, 이중포인터 각각에 동적할당하는 구조를 사용했는데, `memory_alloc2D`에서는 이중포인터를 할당하고, 첫번째 이중포인터에 2 차원배열의 크기만큼을 다 할당받고 각 이중포인터가 동적할당 받은 메모리의 중간중간을 참조하는 방식이었습니다. 이러한 동적할당 방식은 동적할당 메모리를 해제할 때, `for` 문으로 일일이 해제하지 않아도 된다는 장점이 있음을 알게되었습니다.