



Object-Oriented Programming Report

Assignment 1-2

Professor	Donggyu Sim
Department	Computer engineering
Student ID	2022202065
Name	박나림
Class (Design / Laboratory)	2 / C
Submission Date	2023. 3. 23

Program 1

□ 문제 설명

사용자로부터 두 개의 수를 입력 받으면 첫번째 자리의 숫자부터 먼저 출력하고 그 숫자가 총 나온 횟수를 출력한다. 그 다음 두번째 자리의 숫자를 출력하고 마찬가지로 그 숫자가 나온 횟수를 출력한다. 두 개의 수를 입력 받지만 개수를 셀 때에는 전체적으로 하나로 세서 결과적으로도 켜켜이 이어져 출력하게 만든다. 단, 이때 앞서 나온 숫자에 대해서는 중복으로 처리하여 패스하도록 한다.

-구현 방법

입력 받을 숫자를 조금 더 확실히 비교하기 위해 정수가 아닌 문자열로 받을 것이므로, char 형 배열 2개 c1, c2를 선언하고 개수를 세기 위한 int 형 배열 count를 선언해준다. 그 다음 사용자로부터 차례대로 두 숫자를 문자로써 입력 받고, 첫번째 숫자에 대한 개수를 계산해준다. for 반복문을 사용하여 c1 배열의 인덱스 값을 NULL 만나기 전까지 반복하고, 이중 for 문에서 또 다른 변수를 '0'~'9'까지 아스키코드 값으로 반복시킨다. 그리고 안쪽에서 if 문으로 c1 배열의 문자 값이 아스키 코드의 문자 값과 같아지면 count 배열의 값을 1씩 증가시켜서 개수를 세어준다. 여기서 count의 인덱스 값이 중요한데, 아스키코드의 값을 이용하여 변수를 정해줬으므로 count[변수 - '0'] 형태로 해주어야 한다. 예를 들어 변수 값이 변수 값이 '3'이라고 쳤을 때 아스키코드로 '3'은 51이므로 '0'(48)을 뺄셈해야 count[3]으로 나올 수 있다. 두번째 숫자에 대한 개수도 마찬가지로 같은 방법으로 c2 배열을 이용하여 구한다. 이렇게 하면 각 수마다 문자로 취급되어 그 문자당 개수를 하나하나 구할 수 있다.

마지막으로 출력단계의 반복문을 구현한다. 먼저 c1 배열 차례에서, 개수 구할 때와 같이 반복문과 조건문을 쓰고 c1의 배열 값, 해당 수의 개수를 출력해주면 된다. 같은 조건문으로 해야 해당 문자의 개수를 출력할 수 있다. 그 다음 c2 배열도 새로 반복문과 조건문을 써서 출력을 하는데, 이때 중복을 피하기 위한 패스 단계가 필요하므로 안쪽에 해당 조건문을 더 써준다.

□ 결과 화면

```
Microsoft Visual Studio 디버그 콘솔
79 897687543217
74928261825141312111
C:\Users\박나림\OneDrive\바탕 화면\
있습니다(코드: 0개).
```

C1: 79, c2: 897687543217 을 입력했을 때

앞의 배열에선 총 7 이 4 번, 9 가 2 번이므로 >> 7492

다음 배열에선 총 8 이 2 번, 9 와 7 은 앞의 배열에서 나온 중복 수이니 패스, 6 은 1 번.

(다음 8 은 중복 검사 코드를 구현하지 못해 다시 나옴)

□ 고찰

처음에 입력 받을 타입을 char 형으로 하다 보니, count 배열의 인덱스 값을 표현할 때 시행착오를 겪었다. 그냥 변수로만 반복시켰다가 쓰레기 값이 나와서 다시 고치다가 아스키코드 값이니 본래의 '0' 값을 빼주어야 한다는 걸 깨달았다. 그러지 않으면 변수가 0 일경우 count[80] 값이 되어버리니 존재하지 않아서 그런 쓰레기 값이 나오기 때문이었다. 출력하는 단계에서 특히 제일 문제점이 많았는데, 아직 완전히 보안을 못하였다. 하나의 입력이 아닌 두 개의 입력이기 때문에 중복을 검사하는 게 까다로웠다. 첫번째 배열부터 개수를 검사하여 출력하는 건 성공했는데 두번째 배열부터는 어떻게 구현해야 할지 잘 모르겠다. 그래서 일단 문제에서 나온 입력을 성공시켜 보기 위해 구상해봤지만 이 역시 다른 수를 입력했을 때는 모든 중복을 다 검사하지 못한다. 이 부분은 조금 더 고민해 볼 점인 것 같다.

Program 2

□ 문제 설명

사용자가 데이터를 전송할 때, 그 데이터에 오류가 있는지 검사하는 프로그램을 만드는 것이 목표이다. 이러한 오류 검사를 CRC(Cyclic Redundancy Check), 순환 중복 검사라고 한다. 전송된 데이터의 오류 판단을 위한 오류 검출 부호이며, 먼저 데이터를 전송하기 전에 그 값에 따라 CRC 값을 붙여서 전송하고, 전송이 끝난 후에는 전송 받은 데이터의 값으로 다시 CRC 값을 계산해야 한다. 그때 CRC 값이 0 이면 정상적으로 작동, 0 이 아닐 경우에는 데이터 전송 과정 중 오류가 발생했던 것으로 판단한다. 즉, 처음에 발신자는 사용자로부터 12 자리의 바이너리 데이터를 받아서, 그 12bit 이후의 FCS 4bit 를 계산하여 총 16bit 가 되게 붙인다. 그래서 오류 검출을 위한 FCS 가 전송되도록 한다. 이때 총 오류가 발생할 확률은 5%로 가정한다. 수신자는 데이터를 확인하는 과정을 거치며, 나머지에 대한 결과 여부를 출력하기 위해 XOR 비트 연산자를 활용하도록 한다.

□ 결과 화면

(미완성)

□ 고찰

일단 이번 문제는 문제부터 이해하는 데 조금 힘들었던 것 같다. 사용자로부터 전달받은 데이터를 다시 전송할 때 오류검사를 진행하는 코드를 구현해야 되는 것 같다. 그래서 전송 데이터를 계산하기 위해 XOR 연산을 진행해야 되는데 아직 이 부분에 대해 미숙하다는 걸 다시한번 깨달았다. 단순히 연산만 진행하는 것이라면 가능할 것 같은 문제로 보이지만, 그걸 코드를 통해 구현하려니 더 어렵게 느껴졌다. 이 분야와 관련된 개념들을 먼저 충분히 공부하고 진행해야 될 것 같다. 그만큼 성공했을 시 얻어갈 점이 많은 문제 같아서 더 고민해보고 싶은 문제이다.

Program 3

□ 문제 설명

파일 이름을 간략하게 입력해도 원본 파일 이름을 찾을 수 있는 프로그램이다. '와일드 카드'라는 것을 사용하여 찾을 수 있는데, 그 종류는 *와 ?기호로 구성된다. *은 한 단어를 나타내어서, 찾고자 하는 파일의 앞 단어를 입력하고 *을 입력하면 앞 단어의 이름을 가지고 있는 파일들의 이름들을 출력 값으로 보여준다. ?은 한 단어를 나타내는 것으로, 찾고자 하는 파일 이름을 입력하고 ?을 결합하여 붙이면 그 이름을 가지고 있고 해당 위치만 다른 파일 이름들을 출력 값으로 보여준다.

C++에서 텍스트 파일들을 이용하여 코드를 구현할 때에는 ifstream(파일 입력), ofstream(파일 출력), fstream(읽기 쓰기 둘 다 수행)을 이용하도록 한다. 또한 파일을 열고나서는 필수적으로 파일을 닫는 작업도 같이 수행해주어야 한다. 이러한 과정 속에서 오류가 날 수 있으므로 검사해주는 함수들도 필요하다. 주로 EOF(End Of File)의 파일 끝 도달 시 -1을 반환하는 특성을 이용한다. 파일의 내용을 한 줄씩 읽어 저장하는 경우에는 getline 함수를 사용한다.

-구현 방법

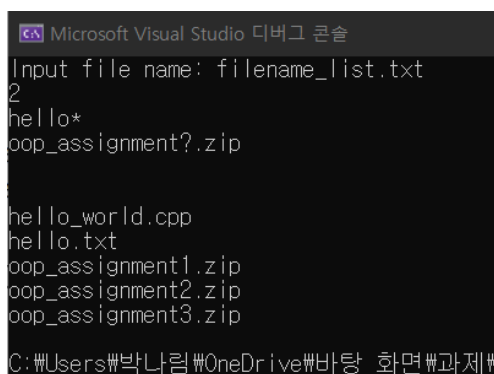
텍스트 파일을 이용해야하므로 ifstream으로 파일 입력 객체를 생성해준다. char형 타입으로 텍스트 파일 이름을 저장할 fname 배열, 찾고자 하는 파일 이름을 저장할 배열, 탐색용 배열, 파일 내용들을 저장할 fstr 배열을 선언한다. 그리고 사용자로부터 텍스트 파일의 이름과 그 안에서 찾고자 하는 파일들의 개수, 이름과 와일드 카드를 같이 입력 받도록 한다. 텍스트 파일의 이름은 fname에 저장하여 open 함수를 이용해 파일을 열어준다. 이때 파일을 열 수 없는 경우 안내문과 함께 프로그램을 종료한다.

정상적으로 파일을 연 경우, 파일의 이름을 탐색하기 위해 와일드카드를 분리시켜야 하므로 반복문과 조건문을 이용하여 와일드카드를 만나기 전까지의 인덱스 값을 계산한다. 만약 와일드카드가 없어서 배열이 NULL 값까지 도달한 경우는 마찬가지로 안내문을 띄우고 종료한다. 인덱스 값은 변수 n에 저장하여 탐색용 배열인 빈공간에다 n만큼 복사하고 마지막 n자리에는 NULL 값으로 문자열의 끝을 넣는다. 다른 입력 받은 문자열도 마찬가지로 과정을 반복한다.

그 후 문자열 검색 단계로, 텍스트 파일에서 한 줄 씩 내용들을 불러와 저장하기 위해 파일의 끝에 도달할 때까지 eof()를 while 문으로 이용하여 getline()으로 불러와 fstr 배열에 저장한다. 그리고 strstr()을 이용하여 fstr과 아까 새로 저장했던 탐색용 배열을 비교하여 char형

포인터로 가르키게 한다. 만약 같은 문자열을 찾는 데 성공하면 NULL 값을 반환하지 않으므로 조건문을 이용하여 그럴 때에만 fstr 문자열을 출력하도록 한다. 이 검색 while 문이 끝나면 파일을 닫고 다시 새로운 파일 객체를 생성하여 다음 문자열을 처음부터 검색하도록 한다. 마찬가지로 같은 문자열만 출력하고 파일을 닫고 프로그램을 종료시킨다.

□ 결과 화면



```
Microsoft Visual Studio 디버깅 콘솔
Input file name: filename_list.txt
2
hello*
oop_assignment?.zip

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

C:\Users\박나림\OneDrive\바탕 화면\과제\
```

찾고자 하는 첫번째 문자열 hello 에 와일드카드 *을, 두번째 문자열 oop_assignment 에는 한 단어를 가리키는 ?을 함께 입력한다. 그러면 그 아래에 찾은 문자열들이 순서대로 나오게 된다.

□ 고찰

C++에서는 파일 스트림을 객체를 통해 생성하고 활용해야 된다는 걸 이번에 알게 돼서 시행착오가 여러 번 있었다. 특히 하나의 문자열을 탐색한 다음 두번째로 넘어갈 때, 이미 열린 파일 스트림에서 그대로 진행하였더니 제대로 탐색이 실행되지 않았다. 첫번째 단계에서 이미 파일의 끝까지 실행하였으니 다시 처음으로 돌아가지 않는 것 같다. 이 부분은 보완이 필요해 보이지만 일단 새로운 파일 객체를 더 생성하여 진행하였다. 또한 제일 보완이 필요한 부분은 문자열을 입력할 개수에 관해서인데, 현재 코드에서는 그 부분이 제대로 구현이 안되었다. 사용자로부터 입력 받은 개수에 따라 그만큼 문자열에 관한 계산을

반복하여 진행해야 되는데 그때마다 새로운 문자열 배열 지정을 어떻게 해줘야 될 지 몰라서 이 점이 부족한 것 같다. 와일드 카드에 대해서도 아직 *과 ?의 구분이 제대로 되질 않아서 이 부분 또한 더 고쳐 나가야 될 점이다.

Program 4

□ 문제 설명

512*512 픽셀(1byte 로 구성) 사이즈의 흑백 2D 이미지 파일을 불러온다. 그리고 원본의 이미지의 두 모서리를 크롭하고, 파일 이름은 (원본 이미지 이름)_cropped_(가로)x(높이).raw 로 만든다. 다음으로 원본 이미지를 수평과 수직으로 각각 반전시킨다. 파일 이름은 (원본 이미지 이름)_horizontalflip.raw, (원본 이미지 이름)_verticalflip.raw 로 만들도록 한다. 이 과정 속에서 발생하는 모든 예외들은 처리할 수 있게 한다.

이러한 영상의 데이터는 가로 M 개, 세로 N 개인 픽셀로 구성되며 2 차원 배열로 이루어진다. 따라서 배열의 인덱스 값은 해당 위치의 픽셀 값을 가리킨다. Raw 데이터 포맷을 다루기 위해서는 영상의 크기 정보가 필요하다. 그래서 이번 문제에서는 크기를 고정하므로, 처음에 define 으로 가로 길이와 높이를 각각 512 로 정의하도록 한다.

Unsigned char 형을 다루기 위해 uint8_t 를, unsigned int 형을 다루기 위해 uint32_t 를 사용한다.

-구현 방법

FILE 함수를 이용하여 입력할 이미지 값과 출력할 이미지 값을 각각 0 으로 초기화 해준다. 그리고 Uint8_t 타입의 이중 포인터로 입력 이미지의 버퍼 값을 0 으로 초기화 해준다. 그리고 고정되는 값을 위해 const char 형 포인터를 이용하여 원본 이미지의 파일 이름을 저장한다. 그 다음 fopen()을 사용하여 바이너리 읽기 모드(rb)로 이미지 파일을 불러온다. 버퍼에는 메모리를 할당하는 함수를 저장한다. 메모리 할당 함수는 main 함수 위에 구현하여, uint8_t 이중 포인터 형으로 만들어 준다. 인수로 가로길이와 세로길이를 전달받는데 이때 이 길이들은 int 형이므로 uint32_t 형으로 불러와준다. 그 아래에는 필수적으로 할당된 메모리를 해제하는 함수도 만들어

준다. 다시 이어서 main 함수에서, 이미지의 높이만큼 반복하여 fread()를 사용해 이미지 파일을 읽어들인다. 그 다음 파일을 각각 크롭, 수평 반전, 수직 반전하는 형태를 구현하고 출력 이미지 변수에 해당 파일을 저장한다. 마지막으로 다시 이미지 높이만큼 반복하여 결과물들을 fwrite()를 사용해 출력하고, 메모리 해제 함수를 불러온 다음에 fclose()로 입력 출력 파일을 모두 닫고 종료한다.

□ 결과 화면

(미완성)

□ 고찰

우선 실습 자료를 참고하여 코드를 구현해보긴 했으나 제일 중요한 부분인 이미지의 변환에 대한 코드를 어떻게 해야 될 지 잘 모르겠다. 이미지가 픽셀 단위로, 1 바이트로 이루어지니 그 단위만큼 사진을 크롭하면 된다는 원리는 이해했다. 하지만 막상 코드로 직접 구현하려니 쉽지 않았던 것 같다. 또한 raw 파일을 처음 다뤄보았는데, 프로그램을 실행시킬 때 마다 런타임 에러가 걸려서 그 원인을 찾느라 여러 시행착오를 더 겪었다. 결국 프로그램을 다 구현하지는 못하였지만 어떻게 해야 성공적으로 바이너리 데이터를 변환할 수 있는지 더 고민해보고 싶다.