2023 Spring OOP Assignment Report

과제 번호 : 3
학번 : 20220100
이름 : 박기현
Povis ID : kihyun

명예서약 (Honor Code)

나는 이 프로그래밍 과제를 다른 사람의 부적절한 도움 없이 완수하였습니다.
I completed this programming task without the improper help of others.

1. 프로그램 개요

- 본 프로그램은 ASCII 문자를 활용하여 그림 또는 도형을 만드는 'ASCII Art'를 생성하는 프로그램이다.
- 본 과제에서는 클래스 상속(class inheritance)과 다형성(polymorphism)을 이해하고, 이를 활용하여 프로그램을 구현하는 것을 목적으로 한다.
- 프로그램 구현을 위해 사용되는 클래스와 상속 관계는 다음과 같다.
- parser 클래스
  - image 파일과 config 파일을 입력 받아 파일 내용을 벡터에 저장한다.
  - 출력하고자 하는 내용을 output 파일에 저장한다.
- artist 클래스 (부모 클래스)
  - parser 클래스를 통해 입력 받은 벡터를 자식 클래스의 스타일에 따라 문자로 변환한다. 이때 virtual을 활용하여 다형성을 구현한다.
  - classic 클래스 (자식 클래스)
    - pixel 값이 작을수록 어둡게 나타낸다. 즉 pixel 값이 낮을수록 시각적으로 밀도가 높은 ASCII 문자를 대입시켜 그림을 표현한다.
  - iclassic 클래스 (자식 클래스)
    - inverted classic의 약자로, classic 클래스와 반대로 pixel 값이 높을수록 밝게 나타낸다.
  - sobelx 클래스 (자식 클래스)

- sobel 연산은 영상에서 가장자리 성분을 검출할 때 사용되는 연산으로, sobelx 클래스는 x축 방향으로 인접한 pixel 값과 50 이상의 차이가 있을 때 '|'를 반환한다. 이외의 경우에는 공백을 반환한다.

◆ sobely 클래스 (자식 클래스)

- sobel 연산은 영상에서 가장자리 성분을 검출할 때 사용되는 연산으로, sobely 클래스는 y축 방향으로 인접한 pixel 값과 50 이상의 차이가 있을 때 '-'를 반환한다. 이외의 경우에는 공백을 반환한다.

◆ gradient 클래스 (자식 클래스)

- sobelx 클래스와 sobely 클래스의 논리를 그대로 반영하고, 추가로 x, y축 바향 모두 인접한 pixel 값의 차이가 50 이상인 경우에는 '+'를 반환한다.

□ drawer 클래스 (부모 클래스)

◆ artist 클래스를 통해 ASCII 문자로 변환된 벡터를 string으로 변환한다.

◆ 이때, drawer 생성자는 인자로 artist pointer instance를 받는다.

◆ **문제 4) drawer 생성자가 artist가 아닌 artist*를 인자로 받는 이유를 설명하라.**

- artist를 인자로 받는 경우, 함수 호출시 암묵적으로 정의된 복사 생성자 (copy constructor)를 호출하여 artist의 모든 원소를 복사한다. 또한, 함수를 수행하고 값을 반환할 때에도 복사 생성자가 호출된다. 따라서 한번의 함수 호출에 두 번의 복사 생성자가 호출된다. 반면에, artist*를 인자로 받는 경우에는 객체의 포인터를 이용하여 주소만 전달함으로써 복사 생성자가 호출되지 않는다. 이를 통해 불필요한 메모리 복사를 방지하고, 속도를 향상시킬 수 있다.

◆ downsample 클래스 (자식 클래스)

- Nearest-neighbor 기반으로 그림의 크기를 절반으로 줄인다. 구체적으로 설명하면, 좌표 (0,0)을 기준으로, 홀수 값을 갖는 좌표의 문자들은 무시된다.

◆ upsample 클래스 (자식 클래스)

- Nearest-neighbor 기반으로 그림의 크기를 두 배로 확대한다. 구체적으로 설명하면, 한 pixel의 문자를 x축으로 2배, y축으로 2배 반복한다.

◆ scale 클래스 (자식 클래스)

- artist* 외에 추가적인 두 개의 int형 인자를 전달받는다. 이때, 두 개의 int형 인자는 각각 x축, y축으로 확장 또는 축소할 배수의 값을 의미한다. 이를 위

해 config 파일에 총 4개의 정보가 포함된다.

- 배율이 자연수인 경우 해당 값의 배율로 그림을 확장하고, 음의 정수인 경우 해당 값의 음의 역수를 배율로 사용한다.

2. 프로그램의 구조 및 알고리즘

□ **main.cpp**

◆ <iostream>, <string>, <vector> 라이브러리와 "parser.hpp", "artist.hpp", "drawer.hpp" 헤더 파일을 포함한다.

◆ int main(int argc, char *argv[])

■ main 함수의 첫 번째 인자는 int형으로, 프로그램 실행시 입력받은 argument의 수를 의미하고, 두 번째 인자는 char*[]형으로, 실제 argument 값들을 의미한다. 예를 들어서 임의의 컴파일된 프로그램 pkh를 ./pkh 2 oop 0411라는 명령어로 실행시킬 경우 argc는 4, argv[1]은 2, argv[2]는 oop, argv[3]은 0411이 된다. argv[0]은 실행파일경로가 자동으로 할당된다.

■ 본 과제에서 최종적으로 컴파일된 프로그램은 첫 번째로 input 파일의 경로, 두 번째로 configuration 파일의 경로, 세 번째로 output 파일의 경로를 입력 받는다.

□ **parser.hpp**

◆ parser 클래스

◆ <iostream>, <fstream>, <string>, <vector> 라이브러리를 포함한다.

◆ public 멤버로, image 파일을 불러오는 함수(vector<int> load_image), config 파일을 불러오는 함수(vector<string> load_config), 결과를 출력하는 함수(void write_result)를 선언한다.

□ **parser.cpp**

◆ "parser.hpp" 헤더 파일을 포함한다.

◆ vector<int> parser::load_image(const char* filename)

■ image 파일을 열고, image 파일의 delimiter로 구분된 내용을 순서대로 int형 벡터에 저장한다. 이때 처음 두 값은 그림의 너비와 높이이므로, 세 번째 내용부터 저장한다.

- 함수 종료시 저장한 벡터를 반환한다.

◆ vector<string> parser::load_config(const char* filename)

- config 파일을 열고, config 파일의 delimiter로 구분된 내용을 순서대로 string형 벡터에 저장한다. 이때 첫 번째 값은 사용될 artist 클래스의 이름, 두 번째 값은 사용될 drawer 클래스의 이름을 의미한다.
- 함수 종료시 저장한 벡터를 반환한다.

◆ void parser::write_result(const char* filename, const string& contents)

- output 파일을 열고, 인자로 전달받은 string을 파일에 저장한다.

□ **artist.hpp**

◆ artist 클래스, classic 클래스, iclassic 클래스, sobelx 클래스, sobely 클래스, gradient 클래스, mystyle 클래스

◆ <iostream>, <vector> 라이브러리를 포함한다.

◆ artist 클래스

- 그림의 스타일을 정의하는 클래스이다.
- private 멤버로, 이미지의 너비, 높이, 픽셀별 문자 벡터를 선언한다.
- public 멤버로, artist 생성자(artist), artist 소멸자(~artist), 명도값 문자 변환 함수(virtual char mapper), 너비 반환 함수(int getWidth), 높이 반환 함수(int getHeight), 픽셀별 문자 벡터 저장 함수(void savePixel), 픽셀별 문자 벡터 반환 함수(vector<char> getPixel)를 선언한다.

◆ classic 클래스

- artist 클래스를 상속받는다.
- private 멤버로, 픽셀별 명도값 벡터와 픽셀별 문자 벡터를 선언한다.
- public 멤버로, classic 생성자(classic), classic 소멸자(~classic), 명도값 문자 변환 함수(char mapper)를 선언한다.

◆ iclassic 클래스

- artist 클래스를 상속받는다.
- private 멤버로, 픽셀별 명도값 벡터와 픽셀별 문자 벡터를 선언한다.
- public 멤버로, iclassic 생성자(iclassic), iclassic 소멸자(~iclassic), 명도값 문자

변환 함수(char mapper)를 선언한다.

- ◆ sobelx 클래스
  - ■ artist 클래스를 상속받는다.
  - ■ private 멤버로, 픽셀별 명도값 벡터와 픽셀별 문자 벡터를 선언한다.
  - ■ public 멤버로, sobelx 생성자(sobelx), sobelx 소멸자(~sobelx), 명도값 문자 변환 함수(char mapper)를 선언한다.
- ◆ sobely 클래스
  - ■ artist 클래스를 상속받는다.
  - ■ private 멤버로, 픽셀별 명도값 벡터와 픽셀별 문자 벡터를 선언한다.
  - ■ public 멤버로, sobely 생성자(sobely), sobely 소멸자(~sobely), 명도값 문자 변환 함수(char mapper)를 선언한다.
- ◆ gradient 클래스
  - ■ artist 클래스를 상속받는다.
  - ■ private 멤버로, 픽셀별 명도값 벡터와 픽셀별 문자 벡터를 선언한다.
  - ■ public 멤버로, gradient 생성자(gradient), gradient 소멸자(~gradient), 명도값 문자 변환 함수(char mapper)를 선언한다.
- ◆ mystyle 클래스
  - ■ artist 클래스를 상속받는다.
  - ■ private 멤버로, 픽셀별 명도값 벡터와 픽셀별 문자 벡터를 선언한다.
  - ■ public 멤버로, mystyle 생성자(mystyle), mystyle 소멸자(~mystyle), 명도값 문자 변환 함수(char mapper)를 선언한다.

□ **artist.cpp**

- ◆ "artist.hpp" 헤더 파일을 포함한다.
- ◆ artist::artist(int _width, int _height, const std::vector<int>& _pixel)
  - ■ 너비와 높이를 저장한다.
- ◆ classic::classic(int _width, int _height, const std::vector<int>& _pixel) : artist(_width, _height)

- 픽셀 명도값 벡터를 저장하고, mapper 함수를 호출하여 문자로 변환하여 픽셀멸 문자 벡터를 저장한다.

◆ char classic::mapper(int x_coordinate, int y_coordinate)

- 픽셀 값이 낮을수록 시각적으로 밀도가 높은 ASCII 문자를 대입시켜 그림을 표현한다. 구체적으로 '@', '&', '%', 'W', 'X', 'A', 'H', 'O', 'T', '*', '^', '+', '-', '.', ' ', 총 15개의 ASCII 문자를 사용한다.
- 예를 들면, 픽셀 명도값이 0~16인 경우 '@', 17~33인 경우 '&'를 반환한다. 예외적으로 마지막 문자는 18개, 238~255의 범위를 가진다.

◆ iclassic::iclassic(int _width, int _height, const std::vector<int>& _pixel) : artist(_width, _height)

- 픽셀 명도값 벡터를 저장하고, mapper 함수를 호출하여 문자로 변환하여 픽셀멸 문자 벡터를 저장한다.

◆ char iclassic::mapper(int x_coordinate, int y_coordinate)

- classic::mapper의 사용되는 ASCII 문자의 순서를 반대로 한다.

◆ sobelx::sobelx(int _width, int _height, const std::vector<int>& _pixel) : artist(_width, _height)

- 픽셀 명도값 벡터를 저장하고, mapper 함수를 호출하여 문자로 변환하여 픽셀멸 문자 벡터를 저장한다.

◆ char sobelx::mapper(int x_coordinate, int y_coordinate)

- x축으로 인접한 픽셀과의 차이가 50 이상인 경우, '|'를 반환한다. 이외의 경우에는 공백을 반환한다.

◆ sobely::sobely(int _width, int _height, const std::vector<int>& _pixel) : artist(_width, _height)

- 픽셀 명도값 벡터를 저장하고, mapper 함수를 호출하여 문자로 변환하여 픽셀멸 문자 벡터를 저장한다.

◆ char sobely::mapper(int x_coordinate, int y_coordinate)

- y축으로 인접한 픽셀과의 차이가 50 이상인 경우, '-'를 반환한다. 이외의 경우에는 공백을 반환한다.

◆ gradient::gradient(int _width, int _height, const std::vector<int>& _pixel) :

artist(_width, _height)

- sobelx::mapper와 sobely::mapper의 논리를 그대로 반영하고, x축과 y축 모두 인접한 픽셀과의 차이가 50 이상인 경우에는 '+'를 반환한다.

◆ mystyle::mystyle(int _width, int _height, const std::vector<int>& _pixel) : artist(_width, _height)

- 픽셀 명도값 벡터를 저장하고, mapper 함수를 호출하여 문자로 변환하여 픽셀멸 문자 벡터를 저장한다.

◆ char mystyle::mapper(int x_coordinate, int y_coordinate)

- **문제 5) artist 클래스를 상속받아서 자신만의 style을 정의하고 그림을 출력하라.**
  - 명도값을 ASCII 문자로 변환하는 과정에서 (0,0)의 명도값을 (width-1, height-1) 좌표에 대입함으로써 classic style과 x축으로 대칭인 그림을 생성한다.
  - 

    본 이미지는 input2.txt와 mystyle|scale|4|2로 생성한 그림이다.

□ **drawer.hpp**

- drawer 클래스, downsample 클래스, upsample 클래스, scale 클래스
- <iostream>, <string>, <vector> 라이브러리와 "artist.hpp" 헤더 파일을 포함한다.
- drawer 클래스
  - private 멤버로, 이미지 문자열을 선언한다.
  - public 멤버로, drawer 생성자(drawer), drawer 소멸자(~drawer), 이미지 문자열 반환 함수(virtual string draw), 이미지 문자열 저장 함수(void saveString)를 선언한다.
- downsample 클래스
  - drawer 클래스를 상속받는다.
  - private 멤버로, 이미지 문자열을 선언한다.
  - public 멤버로, downsample 생성자(downsample), downsample 소멸자(~downsample), 이미지 문자열 반환 함수(string draw)를 선언한다.
- upsample 클래스
  - drawer 클래스를 상속받는다.
  - private 멤버로, 이미지 문자열을 선언한다.
  - public 멤버로, upsample 생성자(upsample), upsample 소멸자(~upsample), 이미지 문자열 반환 함수(string draw)를 선언한다.
- scale 클래스
  - drawer 클래스를 상속받는다.
  - private 멤버로, 이미지 문자열을 선언한다.
  - public 멤버로, scale 생성자(scale), scale 소멸자(~scale), 이미지 문자열 반환 함수(string draw)를 선언한다.

□ **drawer.cpp**

- "drawer.hpp" 헤더 파일을 포함한다.
- drawer::drawer(artist* _artist)
  - artist로부터 전달 받은 픽셀별 문자 벡터를 저장하고, 원소를 순서대로 이미지 문자열에 저장한다.

- string drawer::draw()
  - 이미지 문자열을 반환한다.
- downsample::downsample(artist* _artist) : drawer(_artist)
  - Nearest-neighbor 기반으로 그림의 크기를 절반으로 줄인다. 좌표 (0,0)을 기준으로, 홀수 값을 갖는 좌표의 문자들은 무시된다.
- string downsample::draw()
  - 이미지 문자열을 반환한다.
- upsample::upsample(artist* _artist) : drawer(_artist)
  - Nearest-neighbor 기반으로 그림의 크기를 두 배로 확대한다. 한 pixel의 문자를 x축으로 2배, y축으로 2배 반복한다.
- string upsample::draw()
  - 이미지 문자열을 반환한다.
- scale::scale(artist* _artist, int _x_scale, int _y_scale) : drawer(_artist)
  - 배율이 자연수인 경우 해당 값의 배율로 그림을 확장하고, 음의 정수인 경우 해당 값의 음의 역수를 배율로 사용한다.
- string scale::draw()
  - 이미지 문자열을 반환한다.

□ 프로그램 실행 예제는 다음과 같다.

```
@@@@%%%WWXXXHHHOOTTT***++---....
@@@@%%%WWXXXHHHOOTTT***++---....
@@@@%%%WWXXXHHHOOTTT***++---....
@@@@%%%WWXXXHHHOOTTT***++---....
@@@@%%%WWXXXHHHOOTTT***++---....
@@@@%%%WWXXXHHHOOTTT***++---....
@@@@%%%WWXXXHHHOOTTT***++---....
@@@@%%%WWXXXHHHOOTTT***++---....
....---++***TTTOOHHHXXXWW%%%@@@@
....---++***TTTOOHHHXXXWW%%%@@@@
....---++***TTTOOHHHXXXWW%%%@@@@
....---++***TTTOOHHHXXXWW%%%@@@@
....---++***TTTOOHHHXXXWW%%%@@@@
....---++***TTTOOHHHXXXWW%%%@@@@
....---++***TTTOOHHHXXXWW%%%@@@@
....---++***TTTOOHHHXXXWW%%%@@@@
```

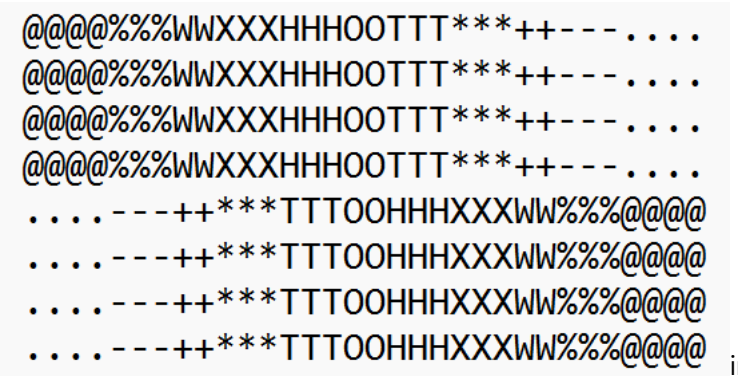
input1.txt / classic|drawer

```
@@%%WXHHOT**+-..
@@%%WXHHOT**+-..
@@%%WXHHOT**+-..
@@%%WXHHOT**+-..
..--+*TTOHXXW%@@
..--+*TTOHXXW%@@
..--+*TTOHXXW%@@
..--+*TTOHXXW%@@
```

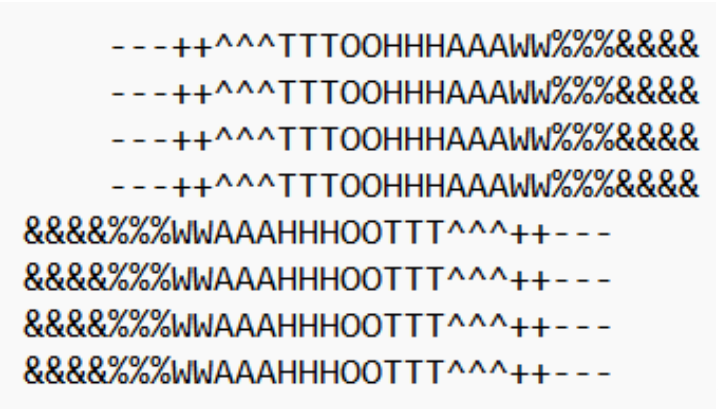
input1.txt / classic|downsample

```
@@@@@@@@%%%%%%WWWWXXXXXXHHHHHHOOOOTTTTTT******++++------........
@@@@@@@@%%%%%%WWWWXXXXXXHHHHHHOOOOTTTTTT******++++------........
@@@@@@@@%%%%%%WWWWXXXXXXHHHHHHOOOOTTTTTT******++++------........
@@@@@@@@%%%%%%WWWWXXXXXXHHHHHHOOOOTTTTTT******++++------........
@@@@@@@@%%%%%%WWWWXXXXXXHHHHHHOOOOTTTTTT******++++------........
@@@@@@@@%%%%%%WWWWXXXXXXHHHHHHOOOOTTTTTT******++++------........
@@@@@@@@%%%%%%WWWWXXXXXXHHHHHHOOOOTTTTTT******++++------........
@@@@@@@@%%%%%%WWWWXXXXXXHHHHHHOOOOTTTTTT******++++------........
@@@@@@@@%%%%%%WWWWXXXXXXHHHHHHOOOOTTTTTT******++++------........
@@@@@@@@%%%%%%WWWWXXXXXXHHHHHHOOOOTTTTTT******++++------........
@@@@@@@@%%%%%%WWWWXXXXXXHHHHHHOOOOTTTTTT******++++------........
@@@@@@@@%%%%%%WWWWXXXXXXHHHHHHOOOOTTTTTT******++++------........
@@@@@@@@%%%%%%WWWWXXXXXXHHHHHHOOOOTTTTTT******++++------........
@@@@@@@@%%%%%%WWWWXXXXXXHHHHHHOOOOTTTTTT******++++------........
@@@@@@@@%%%%%%WWWWXXXXXXHHHHHHOOOOTTTTTT******++++------........
@@@@@@@@%%%%%%WWWWXXXXXXHHHHHHOOOOTTTTTT******++++------........
........------++++******TTTTTTOOOOHHHHHHXXXXXXWWWW%%%%%%@@@@@@@@
........------++++******TTTTTTOOOOHHHHHHXXXXXXWWWW%%%%%%@@@@@@@@
........------++++******TTTTTTOOOOHHHHHHXXXXXXWWWW%%%%%%@@@@@@@@
........------++++******TTTTTTOOOOHHHHHHXXXXXXWWWW%%%%%%@@@@@@@@
........------++++******TTTTTTOOOOHHHHHHXXXXXXWWWW%%%%%%@@@@@@@@
........------++++******TTTTTTOOOOHHHHHHXXXXXXWWWW%%%%%%@@@@@@@@
........------++++******TTTTTTOOOOHHHHHHXXXXXXWWWW%%%%%%@@@@@@@@
........------++++******TTTTTTOOOOHHHHHHXXXXXXWWWW%%%%%%@@@@@@@@
........------++++******TTTTTTOOOOHHHHHHXXXXXXWWWW%%%%%%@@@@@@@@
........------++++******TTTTTTOOOOHHHHHHXXXXXXWWWW%%%%%%@@@@@@@@
........------++++******TTTTTTOOOOHHHHHHXXXXXXWWWW%%%%%%@@@@@@@@
........------++++******TTTTTTOOOOHHHHHHXXXXXXWWWW%%%%%%@@@@@@@@
........------++++******TTTTTTOOOOHHHHHHXXXXXXWWWW%%%%%%@@@@@@@@
........------++++******TTTTTTOOOOHHHHHHXXXXXXWWWW%%%%%%@@@@@@@@
........------++++******TTTTTTOOOOHHHHHHXXXXXXWWWW%%%%%%@@@@@@@@
........------++++******TTTTTTOOOOHHHHHHXXXXXXWWWW%%%%%%@@@@@@@@
```

input1.txt / classic|upsample

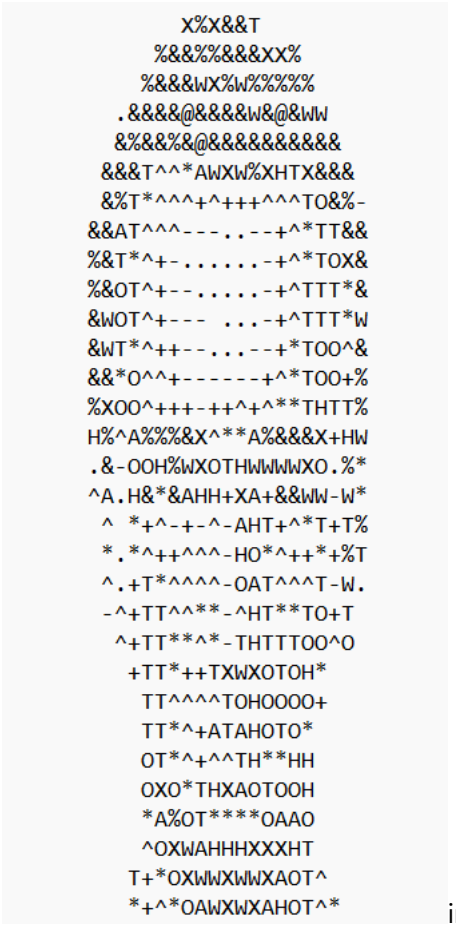


input1.txt / classic|scale|1|-2

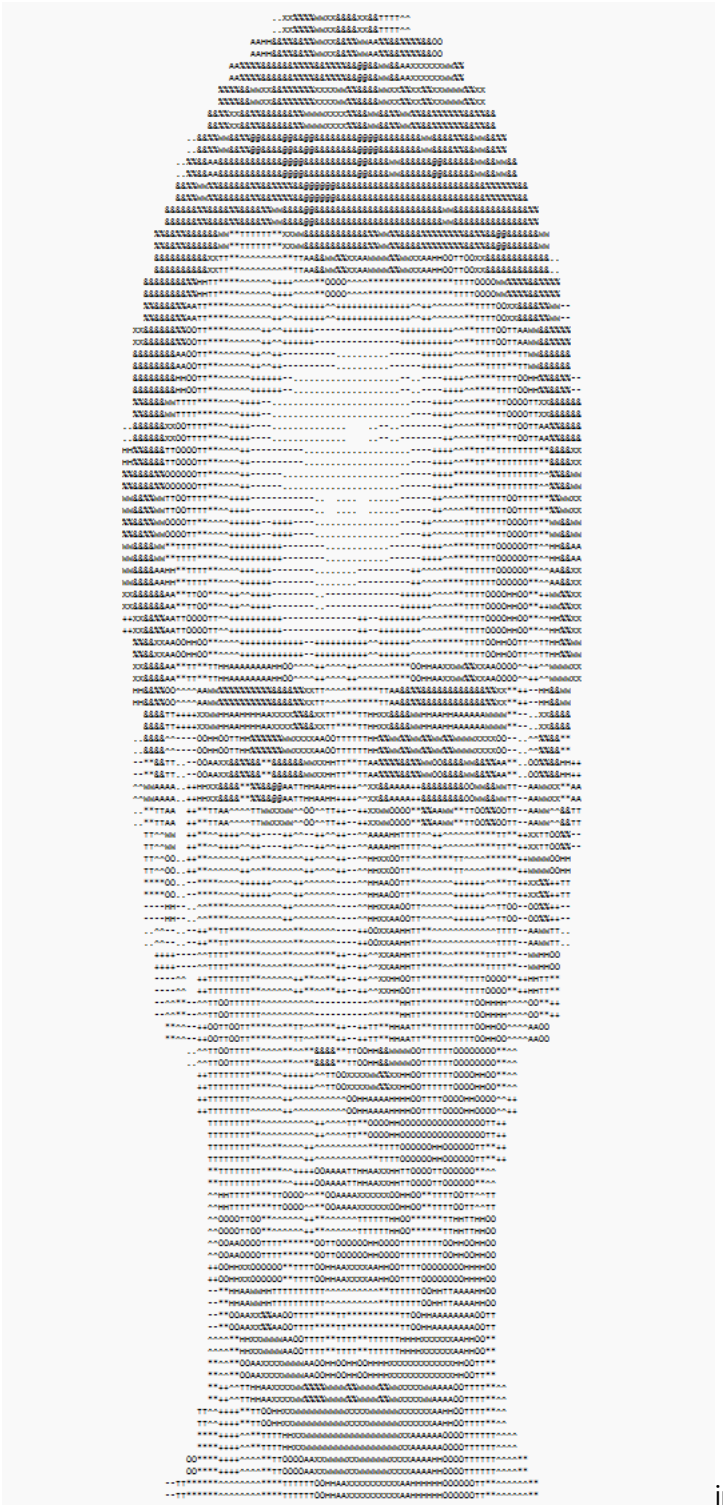


input1.txt / iclassic|scale|1|-2

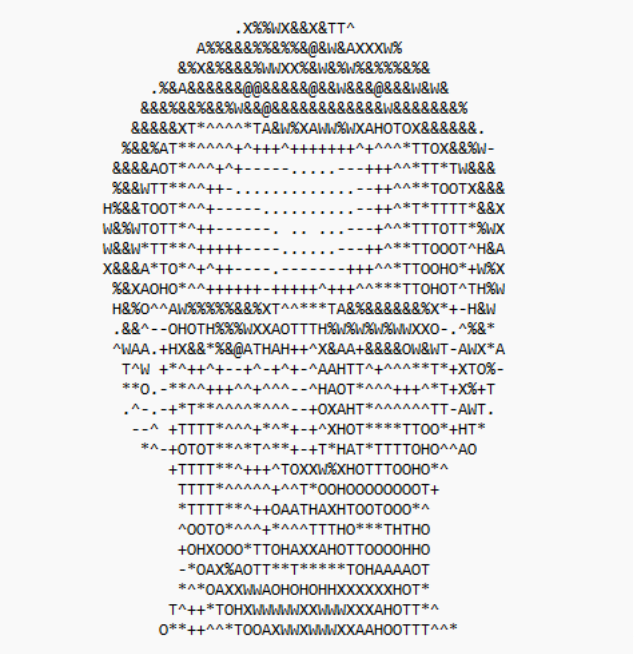
input2.txt / classic|drawer



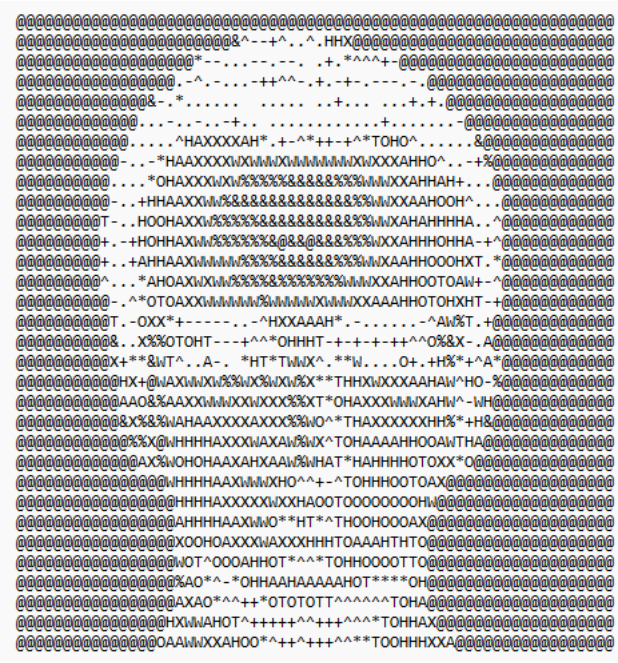
input2.txt / classic|downsample



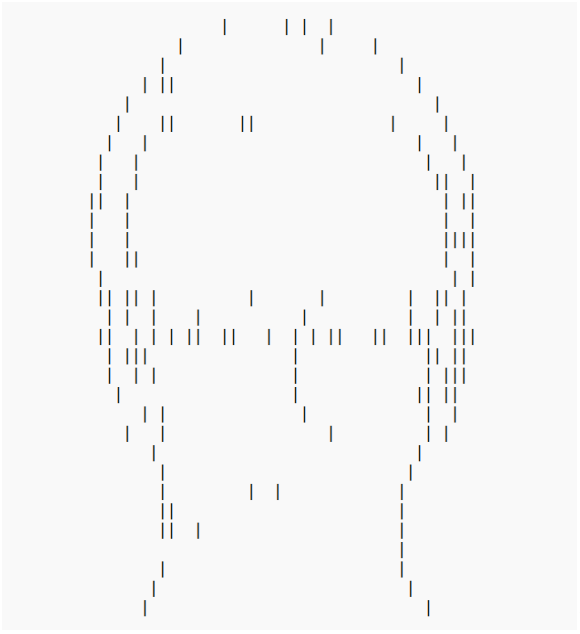
input2.txt / classic|upsample



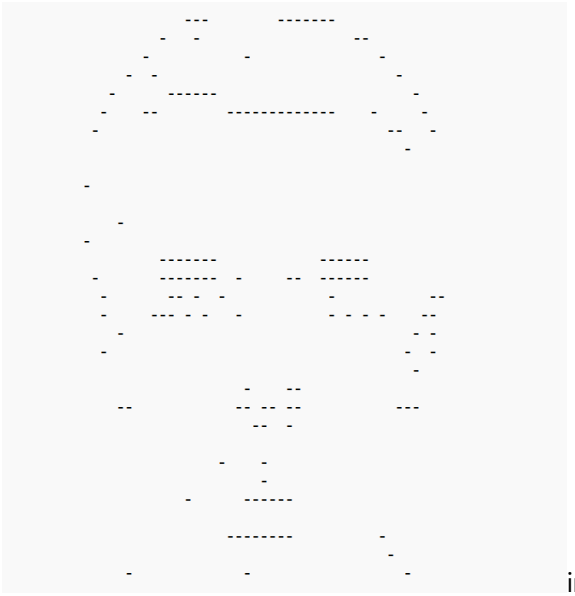
input2.txt / classic|scale|1|-2



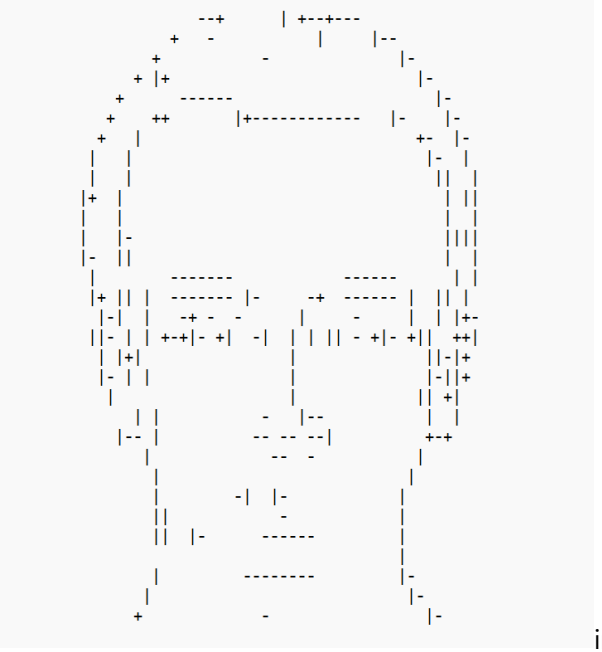
input2.txt / iclassic|scale|1|-2



input2.txt / sobelx|scale|1|-2



input2.txt / sobely|scale|1|-2



input2.txt / gradient|scale|1|-2

3. 토론 및 개선

- □ **문제 6) 제시된 클래스들을 객체 지향 프로그래밍 관점에서 개선할 수 있다면 어떤 점을 개선할 수 있을지 토의하라.**

- □ 객체 지향의 특징으로는 추상화(Abstraction), 캡슐화/정보은닉(Encapsulation/data hiding), 다형성(polymorphism), 상속성(inheritance), 재사용성(reusability)이 있다. parser 클래스에서 load_image 함수와 load_config 함수는 전반적인 알고리즘은 동일하지만, 파일로부터 읽어들이는 데이터의 타입과 반환하는 데이터의 타입이 다르다. 이를 template을 활용한다면 중복성을 피할 수 있고, 객체 지향 관점에서 코드의 재사용성을 향상시킬 수 있을 것이다.

- □ 또한, 아직 상속에 대한 개념이 완벽하지 않아 더욱 효율적으로 구현하지 못한 것 같다. drawer 클래스에서 private 멤버변수로 vector를 선언하여 artist*로부터 값을 읽고, public 멤버함수로 값을 저장한 vector를 image_string으로 변환하는 함수를 구현함으로써 자식 클래스의 생성자에서는 부모 클래스의 벡터를 초기화하는 방식으로 코드의 중복성을 더욱 줄일 수 있을 것이란 생각이 들었다.

- □ 이번 과제를 통해 오버라이딩, virtual 함수에 대한 개념을 이해할 수 있었고, 이를 활용한다면 클래스 상속을 활용할 때 더욱 직관적으로 프로그램의 복잡성을 줄일 수 있다는 것을 알게 되었다. 지난 과제에서는 캡슐화가 잘 이루어지지 않았지만, 이번 과제에서는 캡슐화에 대해 이해도가 향상되어 더욱 쉽게 구현할 수 있었던 것 같다. 이번 프로그램은 오버로딩보다는 오버라이딩에 초점을 맞추어 함수를 구현하는 경우

가 많았던 것 같은데, 오버로딩도 활용한다면, 즉 다형성을 적극적으로 활용한다면 객체 지향 관점에서 더욱 개선된 프로그램을 구현할 수 있을 것 같다.

4. 참고 문헌

□ 조성현 교수님의 <객체지향프로그래밍> 수업 자료