**2023 Spring OOP Assignment Report**

과제 번호 : 3

학번 : 20220124

이름 : 김문겸

Povis ID : kkomy

**명예서약 (Honor Code)**

나는 이 프로그래밍 과제를 다른 사람의 부적절한 도움 없이 완수하였습니다.

I completed this programming task without the improper help of others.

프로그램을 하다 보면 결정해야 할 세부 사항이 많은데, 이러한 세부 사항을 처리한 방법과 이유를 보고서에 쓰십시오.

독창적인 아이디어와 추가 기능은 보너스 점수를 받을 수 있으므로, 보고서에 명확히 기재하십시오.

문제가 여러 개인 경우, 각 문제별로 정리해서 작성합니다.

아래 문항별 설명은 편의를 위한 것으로, 삭제하고 제출한다.

1. **프로그램 개요**
   * 해당 프로그램은 높이, 넓이, 그리고 각각의 픽셀값이 주어진 텍스트 파일을 변환하여 ASCII 문자로 구성된 이미지를 만들어내는 프로그램이다. 이렇게 ASCII 문자만으로 그림과 도형을 만드는 예술 형식을 ASCII art라고 한다. 본 프로그램에서는 5가지 스타일의 ASCII art(classic, iclassic, sobelx, sobely, gradient)를 구현한다. 또한 ASCII art로 변환된 그림을 원하는 비율로 확대 및 축소하는 기능도 제공한다.
   * 본 프로그램의 main함수는 2개의 argument를 필요로 한다. 프로그램 실행 시 픽셀 데이터가 저장되어 있는 input파일, 변환할 ASCII art의 스타일에 대한 정보를 저장하고 있는 config파일, 그리고 최종적으로 변환된 ASCII art를 저장할 output파일에 대한 경로를 제공함으로써 프로그램이 시작된다. 이때 기입할 파일의 개수에 대한 argument와 파일의 경로를 저장할 argument가 main함수에 사용된다.
   * 본 프로그램은 4개의 cpp파일과 3개의 h파일로 구성되어 있다.

parser.h : input 파일과 config 파일을 불러오고, output파일을 저장하는 기능을 수행하는 parser 클래스가 선언되어 있다.

artist.h : ASCII art의 스타일들을 구현하는 기능을 수행하는 artist 클래스와 그의 상속 클래스들이 선언되어 있다.

drawer.h : artist 클래스를 통해 구현된 ASCII art를 확대, 축소 하는 기능을 수행하는 drawer 클래스와 그의 상속 클래스들이 선언되어 있다.

artist.cpp : artist와 그의 상속 클래스들의 constructor, method 등이 정의되어 있다.

drawer.cpp : : drawer와 그의 상속 클래스들의 constructor, method 등이 정의되어 있다.

parser.cpp : parser에 선언된 method들이 정의되어 있다.

main.cpp : 전체적으로 설계된 ASCII art의프로그램을 구현하는 파일이다.

1. **프로그램의 구조 및 알고리즘**

* parser class

vector<int> load\_image(const char\*) : input 파일의 경로를 argument로 받아서 해당 txt파일에 저장 되어 있는 ‘그림의 높이’, ‘그림의 너비’, ‘그림을 구성하는 픽셀 각각의 데이터값들을 vector로 저장한다.

vector<string> load\_config(const char\*) : config 파일의 경로를 argument로 받아서 해당 txt파일에 저장되어 있는 변환할 ASCII art의 스타일, 적용할 확대/축소 기능의 종류, 그리고 확대/축소할 높이, 너비의 비율을 vector로 저장한다.

void write\_result(const char\*, const std::string&) : 최종적으로 만들어진 ASCII art가 저장되어 있는 string을 argument로 받고, ASCII art를 저장할 output 파일의 경로를 argument로 받아 해당 파일에 저장한다.

* artist class
* artist는 총 5개의 class를 상속해준다. classic, iclassic, sobelx, sobely, gradient class이다.

<protected>

int width : input 파일의 있는 그림의 너비를 저장한다.

int height : 그림의 높이를 저장한다.

vector<int> pixel : 그림을 구성하는 픽셀 값들을 저장한다.

<public> :

artist<int, int, const vector<int>&) : width에 저장할 그림의 너비와 height에 저장할 높이, 그리고 픽셀들을 argument로 받고, 해당 값으로 width, height, pixel을 초기화해준다.

* derived class인 classic, iclassic sobelx sobely gradient class들도 똑 같은 argument를 가지는 constructor가 존재하고, 같은 기능을 수행한다.

virtual char mapper(int, int) : 받아온 pixel 값들을 스타일에 따라 문자로 변환하는 함수이다. 이 함수는 virtual 함수로서 artist를 상속받는 여러 클래스에 대해서 새롭게 정의된다.

* classic::mapper(int, int) : pixel 값을 범위에 따라 ‘@’ ‘&’ ‘%’ ‘W’ ‘X’ ‘A’ ‘H’ ‘O’ ‘T’ ‘\*’ ‘^’ ‘+’ ‘-’ ‘.’ ‘ ‘ 15개의 문자로 변환한다. 픽셀 값이 [0,16]이면 @, [17.33]이면 &로, 이런 식으로 16개씩 끊어 순서대로 변환한다. 이때 마지막 문자인 ‘ ‘은 예외적으로 [238,255] 17개의 숫자 범위를 커버한다.
* iclassic::mapper(int, int) : classic::mapper을 정반대로 수행한다. [0,16]이면 ‘ ‘, … [238,255]은 @로 변환한다.
* sobelx::mapper(int, int) : x축 양의 방향으로 인접한 pixel과의 차이가 50이상인 경우 ‘|’를, 아닌 경우 공백문자 ‘ ‘를 리턴한다.
* sobely::mapper(int, int) : y축 양의 방향으로 인접한 pixel과의 차이가 50이상인 경우 ‘-‘를, 아닌 경우 공백문자 ‘ ‘를 리턴한다.
* gradient::mapper(int, int) : sobelx와 sobely를 합친다. 두 가지 기능 모두 수행하지만, ‘|’ㅇ와 ‘-‘를 출력해야 하는 조건을 모두 만족한 경우, ‘+’를 리턴한다.

int call\_width() : protected에 속한 width 값을 외부에서 사용하기 위해 return하는 함수

int call\_height() : protected에 속한 height 값을 외부에서 사용하기 위해 return하는 함수

* drawer class
* drawer class는 downsample, upsample, scale 총 3개의 class를 상속해주고 있다.

<protected>

int wid : drawer의 constructor은 artist 타입의 데이터를 argument로 받는다. 이때 해당 artist 변수의 width 값을 저장한다.

int hei : drawer의 constructor은 artist 타입의 데이터를 argument로 받는다. 이때 해당 artist 변수의 height 값을 저장한다.

vector<char> map : drawer의 constructor은 artist 타입의 데이터를 argument로 받는다. 이때 해당 artist가 가지고 있는 픽셀 값들을 문자로 변환한 것들을 이 벡터에 저장해 놓는다.

artist\* art : artist 타입의 자료형을 저장한다. drawer의 constructor에서 픽셀 값들을 artist의 method들을 사용해서 최종적으로 그려지는 string을 만들어낸다. 이 변수는 인자로 받아들이는 artist 변수의 메모리 할당을 destructor에서 해제하기 위해 후에 사용된다.

<public>

drawer(artist\*) : 이 constructor은 우리가 최종적으로 변환할 그림의 원본 픽셀값들을 저장하고 있는 artist 변수를 argument로 받는다. 이후 artist의 call\_width(), call\_height() 함수를 통해 wid, hei를 초기화시키고, 받아들인 artist 변수는 art에 저장한다. 그리고 이 constructor은 artist 변수의 픽셀 값이 저장되어 있는 vector을 해당 constructor의 mapper() 함수로 문자 변환하여, map vector에 저장하는 기능을 수행한다.

* drawer을 상속받는 scale class의 경우 확대, 축소를 얼마나 할 것인지에 대한 값을 필요로 하기 때문에, 추가적으로 확대/축소할 x축 비율, y축 비율 총 2개의 인자를 추가로 받아서( drawer(artist\*, int, int) ) scale class의 멤버변수인 x, y에 저장한다.

virtual std::string draw() : drawer(artist\*)를 통해 변환된 문자가 저장되어 있는 map 벡터의 원소들을 하나씩 불러 string에 저장하고, 줄바꿈을 적용하여 최종적인 하나의 그림으로 이미지화 한다. 그리고 그 string을 리턴한다.

* downsample::draw() : draw()로 표현되는 이미지를 가로, 세로 모두 1/2배로 축소시킨다.
* upsample::draw() : draw()로 표현되는 이미지를 가로, 세로 모두 2배로 확대한다.
* scale::draw() : drawer(artist\*, int, int)를 통해 초기화한 x, y 값을 이용하여 이미지를 변환한다. 축소의 경우 예를 들어 가로가 10이지만 1/4배를 해야 하는 상황처럼 나누어 떨어지지 않는 경우가 있는데, 그럴 때는 1번째, 5번째, 9번째만 선택하고 나머지는 버린다.
* main 함수 변수 설명

int argc : 입력받는 파일의 경로의 개수를 저장한다. 입력하는 파일의 개수 + 본 프로그램의 실행 경로의 값을 저장한다. 본 프로그램의 경우 input, config, output 3개의 본 프로그램 실행 경로 +1 해서 4를 저장한다.

char \*argv[] : 입력하는 파일들의 경로를 저장한다. argv[0]에는 본 프로그램의 실행경로로 고정되고, 뒤부터 입력한 파일들의 경로가 순차적으로 저장된다.

vector<int> tokens : input 파일에 저장되어 있는 숫자들(너비, 높이, 픽셀 데이터들)을 순차적으로 저장한다. input 파일의 첫번째 숫자는 너비, 두번째 숫자는 높이, 그리고 이후 세 번째 숫자부터는 이미지를 구성하는 픽셀의 데이터값이 나열되어 있다.

vector<string> configs : config에 저장되어 있는 ASCII art 스타일, 확대 축소 기능 종류, 추가적으로 scale을 사용하는 경우 필요한 확대/축소하는 x,y의 비율값들을 받아서 벡터에 저장한다.

string style\_target : configs[0], 즉 ASCII art 스타일 값을 저장한다.

string drawer\_target : configs[1], 즉 어떤 확대/축소 기능을 사용할 건지 그 값을 저장한다.

char\* path\_putput : argv[3], 즉 최종적인 이미지를 저장할 output 파일의 경로를 저장한다.

width : tokens[0], 즉 변환할 이미지의 너비를 저장한다.

height : tokens[1], 즉 변환할 이미지의 높이를 저장한다.

vector<int> vals : token[3]부터 끝까지, 즉 변환할 이미지의 픽셀들의 대한 데이터값들을 저장한다.

artist\* style : 문자로 변환할 스타일들에 대한 artist 타입 변수를 저장한다. 이 변수들은 config 파일에 저장된 style\_target에 따라서 각각의 artist의 derived class object로 저장된다.

drawer\* d : 확대/축소 기능의 종류들을 적용하기 위해 drawer형 변수를 저장한다. drawer\_target에 따라 각각의 drawer의 derived class object로 저장된다.

string output : artist와 drawer을 거치며 변환된 최종적인 ASCII art를 저장한다. 이후 parser class의 멤버 함수인 write\_result를 통해서 output 파일에 저장된다.

* 전체적인 알고리즘은 다음과 같다 :

가장 처음 main함수가 input, config, output 파일에 대한 경로를 입력받으면서 프로그램이 실행된다. 만약 세 파일에 대한 경로가 모두 입력되지 않는다면, 에러를 표시한다.

이후 input, config 파일에 기입되어 있는 ASCII art를 만드는데 필요한 데이터들을 변수를 선언하여 따로 저장해 둔다.

이후 style\_target의 값에 따라서, style 변수에 classic, iclassic, sobelx, sobely, gradient 객체를 동적 할당한다.

이후 drawer\_target의 값을 조사해서, 그 값에 따라서 “d” 변수에 drawer, upsample, downsample, scale 객체를 동적할당한다. 이때 앞서 할당했던 style 변수를 인자로 갖는다.

이후 drawer의 멤버 함수인 draw()를 통해 drawer, artist를 거치면서 만들어진 최종적인 ASCII art를 그려낸다.

이후 ASCII art를 출력함과 동시에 p.write\_result 함수를 통해서 output 파일에 내용을 저장한다.

완료 후 프로그램을 종료한다.

1. **토론 및 개선**
   * + 문제4> drawer 생성자가 artist가 아닌 artist\*를 인자로 받는 이유

drawer 클래스에서는 artist가 저장하고 있는 픽셀 데이터 vector을 문자로 변환한다. 그리고 해당 기능은 artist의 멤버함수 “mapper”가 수행하며, 이 mapper은 virtual 함수이다. virtual 함수는 overriding하게 되면, 같은 유형의 포인터가 다른 유형의 객체를 가리킬 수 있게 된다. 따라서 artist의 derived class들인 classic, iclassic… 들이 자신만의 mapper 함수를 가질 수 있는 것이다. 그런데 만약 artist\*가 아닌 artist를 받는 경우, artist가 아닌 이를 상속받는 class에 대한 객체의 mapper 함수들, 즉 virtual 함수로서의 기능이 작동되지 않는다. 따라서 artist\*을 받아옴으로써 derived class들의 함수들을 사용알 수 있게끔 한다.

* + - 문제5> artist 클래스를 상속받아서 자신만의 style을 정의하고 그림 출력하기

gradient의 성능을 높여서, 조금 더 테두리를 강조한 style을 정의해 보았다.

본래는 ‘|’와 ‘-‘, ‘+’만을 이용했지만, 이 문자에서 더 세부적인 픽셀의 차이를 표현하기 위해 ‘|’의 경우 ‘U’를(세로선이 두 개가 있는 문자), ‘-‘의 경우 ‘=’, ‘+’의 경우 ‘#’을 이용했다. 원래는 양의 방향에 대해서, x, y 모두 픽셀값이 50이상 차이날 때 ‘|’와 ‘-‘를 사용했다. 이번에는 좀 더 세부화 하여, 25부터 50이하만큼 차이 날 때 ‘|’와 ‘-‘를, 50 이상 차이날 때에는 ‘U’와 ‘=’를 사용했다. 그리고 U와 -가 겹칠 경우 ‘H’를, ‘|’와 ‘=’가 겹칠 경우 ‘t’를, ‘U’와 ‘=’가 겹칠 경우 ‘#’를 사용하여 표현한다. 따라서 기존 gradient보다 더 섬세한 edge 표현이 가능하다.

나만의 style을 구현하기 위해 작성한 코드는 아래와 같다.

|  |
| --- |
| ‘char mystyle::mapper(int x, int y) {  int value;  int comp;  char xtemp = 0;  char ytemp = 0;  value = pixel.at(y \* width + x);  if (x < width - 1) {  comp = pixel.at(y \* width + x + 1);  int dif = value - comp;  if (dif >= 50 || dif <= -50) {  xtemp = '|';  }  else xtemp = ' ';  }  else if (x == width - 1) {  xtemp = ' ';  }  if (y < height - 1) {  comp = pixel .at((y + 1) \* width + x);  int dif = value - comp;  if (dif >= 50 || dif <= -50) {  ytemp = '-';  }  else ytemp = ' ';  }  else if (y == height - 1) {  ytemp = ' ';  }  if (xtemp == '|' && ytemp == '-') {  return '+';  }  else if (xtemp == '|' && ytemp != '-') {  return '|';  }  else if (xtemp != '|' && ytemp == '-') {  return '-';  }  else return ' ';  } |

“mystyle|scale|1|-2”로 config을 설정하여, 제공받은 input2.txt 파일을 이용해

변환을 하게 되면 아래와 같은 이미지가 나온다.

텍스트, 전자제품이(가) 표시된 사진

자동 생성된 설명

기존의 gradient는 아래와 같다.

도표이(가) 표시된 사진

자동 생성된 설명

비교하면, 좀 더 세세한 픽셀들의 표현을 확인할 수 있다.

* + - 문제6> 제시된 클래스들을 객체 지향 프로그래밍 관점에서 개선할 수 있다면 어떤 점을 개선할 수 있을까?

현재 drawer 클래스는 생성자에서 artist\*를 인자로 받고 있다. 그리고 artist의 멤버 함수를 사용하여 숫자로 구성된 pixel 데이터 값들을 문자로 변환시킨다. 이 기능은 artist에 대해서 외부에서 artist의 기능을 사용하는 것으로, 해당 기능은 충분히 artist 내부에서 수행할 수 있고, 또 artist 내부에서 사용하는 것이 객체 지향 프로그래밍의 관점에서 encapsulation과 information hiding을 충족하는 더 좋은 방법이다. 따라서 drawer의 constructor가 artist 자체를 인자로 받는 것이 아니라, 미리 artist에서 픽셀 값들을 문자로 변환시켜서, 해당 값을 call 함수를 통해 인자로 받는 것이 하나의 개선방법이라 볼 수 있다.

1. **참고 문헌**

* 인터넷 등의 다른 소스에서 참고한 내용이 있으면 출처와 내용을 명확히 기술.
  + - 숙제를 위하여 다른 소스를 참고하고 사용하는 것은 좋으나, 그 경우에 반드시 출처를 명시해야 한다. 출처가 명시되어 있지 않은 경우는 부정행위로 간주될 수 있다.