**2023 Spring OOP Assignment Report**

과제 번호 : #3

학번 : 20220923

이름 : 이윤혁

Povis ID : leeyoonhyuk0

**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**

이 프로그램은 이미지의 명암값을 가진 파일을 읽어, 이미지를 ASCII 아트로 만들어 파일로 출력한다. 파일로 출력하는 ASCII아트의 스타일과 크기는 여러가지이며, 스타일과 크기는 또 다른 입력파일로 설정해줄 수 있다. 프로그램 디렉토리는 [표 1]과 같이 구성되어있다.

[표 ] 프로그램 디렉토리 설명

|  |  |
| --- | --- |
| 파일명 | 설명 |
| artist.cpp | artist class의 소스파일, artist class의 method가 정의되어 있다. |
| artist.hpp | aritist class의 헤더파일, artist class가 선언되어있다. |
| classic.cpp | classic class의 소스파일, classic class의 method가 정의되어 있다. |
| classic.hpp | classic class의 헤더파일, classic class가 선언되어있다. |
| downsample.cpp | Downsample class의 소스파일, downsample class의 method가 정의되어 있다. |
| downsample.hpp | downsample class의 헤더파일, downsample class가 선언되어있다. |
| drawer.cpp | drawer class의 소스파일, drawer class의 method가 정의되어 있다. |
| drawer.hpp | drawer class의 헤더파일, drawer class가 선언되어있다. |
| gradient.cpp | gradient class의 소스파일, gradient class의 method가 정의되어 있다. |
| gradient.hpp | gradient class의 헤더파일, gradient class가 선언되어있다. |
| iclassic.cpp | iclassic class의 소스파일, iclassic class의 method가 정의되어 있다. |
| iclassic.hpp | iclassic class의 헤더파일, iclassic class가 선언되어있다. |
| main.cpp | main()이 정의된 소스파일. 이 파일에서 프로그램이 시작되고 종료된다. |
| parser.cpp | parser class의 소스파일, parser class의 method가 정의되어 있다. |
| parser.hpp | parser class의 헤더파일, parser class가 선언되어있다. |
| scale.cpp | scale class의 소스파일, scale class의 method가 정의되어 있다. |
| scale.hpp | scale class의 헤더파일, scale class가 선언되어있다. |
| sobelx.cpp | sobelx class의 소스파일, sobelx class의 method가 정의되어 있다. |
| sobelx.hpp | sobelx class의 헤더파일, sobelx class가 선언되어있다. |
| sobely.cpp | sobely class의 소스파일, sobely class의 method가 정의되어 있다. |
| sobely.hpp | sobely class의 헤더파일, sobely class가 선언되어있다. |
| upsample.cpp | upsample class의 소스파일, upsample class의 method가 정의되어 있다. |
| upsample.hpp | upsample class의 헤더파일, upsample class가 선언되어있다. |

1. **프로그램의 구조 및 알고리즘**
   1. parser

parser는 이미지 값을 담은 파일과 config 파일을 파싱하거나, 변환된 ASCII 이미지를 출력하는 파일입출력 클래스이다. [표 2]는 Parser의 method를 설명한다.

[표 ] parser의 method

|  |  |  |
| --- | --- | --- |
| 접근 제한자 | {반환형}{함수명}({파라미터명}) | 설명 |
| public | std::vector<int> load\_image(const char\*) | 이미지를 읽어 직렬화된 이미지 데이터를 벡터로 반환하는 함수 |
| public | std::vector<std::string> load\_config(const char\*) | Config 파일을 읽어 변환하려는 이미지의 데이터를 벡터로 반환하는 함수이다. |
| public | void write\_result(const char\*, const std::string) | 입력된 ASCII 아트를 파일로 출력하는 함수이다. |

* 1. artist

artist는 이미지를 받아 ASCII 아트로 변환할 수 있도록 ASCII 문자와 이미지 값을 매핑하는 클래스이다. [표 3]은 artist의 method를 설명한다. [표 4]는 artist의 member variable을 설명한다.

[표 ] artist의 method

|  |  |  |
| --- | --- | --- |
| 접근 제한자 | {반환형}{함수명}({파라미터명}) | 설명 |
| public | artist() | Default constructor |
| public | artist(int, int, const std::vector<int>&) | constructor, 이미지의 높이, 너비, 이미지값 벡터를 차례로 입력받는다. |
| public | virtual ~artist() | Destructor |
| public | int get\_width() | 이미지의 높이 접근자 |
| public | Int get\_height() | 이미지의 너비 접근자 |
| public | virtual char mapper(int, int) | 주어진 x,y 좌표의 이미지 픽셀값을 ascii문자로 매핑하는 함수 |
| protected | int get\_pixel(int, int) | 주어진 x,y 좌표의 이미지 픽셀값을 반환하는 함수 |

[표 ] artist의 member variable

|  |  |  |
| --- | --- | --- |
| 접근 제한자 | {자료형}변수명 | 설명 |
| private | int width | 이미지의 너비 |
| private | int height | 이미지의 높이 |
| private | std::vector<int> image\_data | 이미지의 픽셀값 벡터 |

* + 1. classic

classic은 artist의 derived class로, classic 매핑(0~255 값을 15등분)으로 ASCII 아트로 주어진 이미지 픽셀값으로 매핑하는 클래스이다. [표 5]는 classic의 method를 설명한다.

[표 ] classic의 method

|  |  |  |
| --- | --- | --- |
| 접근 제한자 | {반환형}{함수명}({파라미터명}) | 설명 |
| public | classic() | Default constructor |
| public | classic(int, int, const std::vector<int>&) | constructor, 이미지의 높이, 너비, 이미지값 벡터를 차례로 입력받는다. |
| public | virtual ~classic() | Destructor |
| public | char mapper(int, int) | 주어진 x,y 좌표의 이미지 픽셀값을 ascii문자로 매핑하는 함수 |

* + 1. iclassic

iclassic은 artist의 derived class로, classic 매핑의 문자를 inverse하여 ASCII 아트로 주어진 이미지 픽셀값으로 매핑하는 클래스이다. [표 6]은 iclassic의 method를 설명한다.

[표 ] iclassic의 method

|  |  |  |
| --- | --- | --- |
| 접근 제한자 | {반환형}{함수명}({파라미터명}) | 설명 |
| public | iclassic() | Default constructor |
| public | iclassic(int, int, const std::vector<int>&) | constructor, 이미지의 높이, 너비, 이미지값 벡터를 차례로 입력받는다. |
| public | virtual ~iclassic() | Destructor |
| public | char mapper(int, int) | 주어진 x,y 좌표의 이미지 픽셀값을 ascii문자로 매핑하는 함수 |

* + 1. sobelx

sobelx는 artist의 derived class로, 같은 y좌표의 인접한 픽셀값의 차가 50이 넘을 때 선을 매핑하는(즉 세로 가장자리를 검출하는) 클래스이다. [표 7]은 sobelx의 method를 설명한다.

[표 ] sobelx의 method

|  |  |  |
| --- | --- | --- |
| 접근 제한자 | {반환형}{함수명}({파라미터명}) | 설명 |
| public | sobelx() | Default constructor |
| public | sobelx(int, int, const std::vector<int>&) | constructor, 이미지의 높이, 너비, 이미지값 벡터를 차례로 입력받는다. |
| public | virtual ~sobelx() | Destructor |
| public | char mapper(int, int) | 주어진 x,y 좌표의 이미지 픽셀값을 ascii문자로 매핑하는 함수 |

* + 1. sobely

sobely는 artist의 derived class로, 같은 x좌표의 인접한 픽셀값의 차가 50이 넘을 때 선을 매핑하는(즉 가로 가장자리를 검출하는) 클래스이다. [표 8]은 sobely의 method를 설명한다.

[표 ] sobely의 method

|  |  |  |
| --- | --- | --- |
| 접근 제한자 | {반환형}{함수명}({파라미터명}) | 설명 |
| public | sobely() | Default constructor |
| public | sobely(int, int, const std::vector<int>&) | constructor, 이미지의 높이, 너비, 이미지값 벡터를 차례로 입력받는다. |
| public | virtual ~sobely() | Destructor |
| public | char mapper(int, int) | 주어진 x,y 좌표의 이미지 픽셀값을 ascii문자로 매핑하는 함수 |

* + 1. gradient

gradient는 artist의 derived class로, sobelx와 sobely의 매핑을 동시에 작동시키는 (즉 가로 가장자리를 검출하는) 클래스이다. [표 9]는 gradient의 method를 설명한다.

[표 ] gradient의 method

|  |  |  |
| --- | --- | --- |
| 접근 제한자 | {반환형}{함수명}({파라미터명}) | 설명 |
| public | gradient() | Default constructor |
| public | gradient(int, int, const std::vector<int>&) | constructor, 이미지의 높이, 너비, 이미지값 벡터를 차례로 입력받는다. |
| public | virtual ~gradient() | Destructor |
| public | char mapper(int, int) | 주어진 x,y 좌표의 이미지 픽셀값을 ascii문자로 매핑하는 함수 |

* 1. drawer

drawer는 artist를 받아 artist의 mapper를 통해 ASCII 아트를 실제 문자열로 출력하는 클래스이다. [표 10]은 drawer의 method를 설명한다. [표 11]은 drawer의 member variable를 설명한다.

[표 ] drawer의 method

|  |  |  |
| --- | --- | --- |
| 접근 제한자 | {반환형}{함수명}({파라미터명}) | 설명 |
| public | drawer() | Default constructor |
| public | drawer(artist\*) | constructor, artist class instance를 입력받는다. |
| public | ~drawer() | Destructor |
| public | virtual std::string draw() | ASCII 아트를 그리는 함수 |
| protected | artist\* get\_artist() const | artist\_instance의 접근자 |
| protected | void get\_original\_mapping() const | original\_mapping의 접근자 |
| protected | set\_original\_mapping(char\*\*) | original\_mapping의 설정자 |

[표 ] drawer의 member variable

|  |  |  |
| --- | --- | --- |
| 접근 제한자 | {자료형}변수명 | 설명 |
| private | artist\* artist\_instance | 입력된 artist 클래스 |
| private | char\*\* original\_mapping | ASCII 아트를 저장하는 원본 2차원 배열 |

* + 1. downsample

downsample은 drawer의 derived class로, drawer의 draw()를 할 때 높이와 너비를 각각 1/2배, 1/2배해서 문자열을 출력하는 클래스이다. [표 12]는 downsample의 method를 설명한다.

[표 ] downsample의 method

|  |  |  |
| --- | --- | --- |
| 접근 제한자 | {반환형}{함수명}({파라미터명}) | 설명 |
| public | downsample() | Default constructor |
| public | downsample(artist\*) | constructor, artist class instance를 입력받는다. |
| public | virtual std::string draw() | ASCII 아트를 그리는 함수 |

* + 1. upsample

upsample은 drawer의 derived class로, drawer의 draw()를 할 때 높이와 너비를 각각 2배, 2배해서 문자열을 출력하는 클래스이다. [표 13]는 upsample의 method를 설명한다.

[표 ] upsample의 method

|  |  |  |
| --- | --- | --- |
| 접근 제한자 | {반환형}{함수명}({파라미터명}) | 설명 |
| public | upsample() | Default constructor |
| public | upsample(artist\*) | constructor, artist class instance를 입력받는다. |
| public | virtual std::string draw() | ASCII 아트를 그리는 함수 |

* + 1. scale

upsample은 drawer의 derived class로, drawer의 draw()를 할 때 높이와 너비를 각각 입력한 배수만큼 늘리거나 줄여서 문자열을 출력하는 클래스이다. [표 14]는 scale의 method를 설명한다. [표 15]는 scale의 member variable를 설명한다.

[표 ] scale의 method

|  |  |  |
| --- | --- | --- |
| 접근 제한자 | {반환형}{함수명}({파라미터명}) | 설명 |
| public | scale() | Default constructor |
| public | scale(artist\*, int, int) | constructor, artist class instance와 x\_scale, y\_scale을 차례로 입력받는다. |
| public | virtual std::string draw() | ASCII 아트를 그리는 함수 |

[표 ] scale의 member variable

|  |  |  |
| --- | --- | --- |
| 접근 제한자 | {자료형}변수명 | 설명 |
| private | int x\_scale | x 방향 scale 배수 |
| private | int y\_scale | y 방향 scale 배수 |

* 1. main

main()의 pseudo-code는 [표 16]에 작성했다.

[표 ] main()의 pseudo-code

|  |
| --- |
| 1. 명령인자 입력(input\_path, config\_path, output\_path) 2. 명령인자 검사, 올바르지 않으면 프로그램 종료 3. parser로 이미지 로드 4. parser로 config 로드 5. config의 정보로 artist 클래스 설정 6. config의 정보로 drawer 클래스 설정, artist 넘김 7. drawer로 ASCII 아트 문자열 출력 8. 콘솔과 파일로 출력 9. 동적할당 해제 10. 종료 |

1. **토론 및 개선**

이번 과제를 해결하면서 separate compilation, class inheritance의 개념을 제대로 이해하고 직접 구현해볼 수 있었다. 특히 상속과 접근제어자로 오류와 부딫히며 알 수 있었다. 또한 링커 오류도 많이 겪었는데, 이를 통해 separate compilation을 제대로 익힐 수 있었다.

문제4의 답: artist의 포인터로 입력받아야 drawer의 생성자가 실행되는 도중 artist의 불필요한 copy가 일어나지 않기 때문이다. 또한 drawer가 파괴되면서 destructor를 통해 동적할당 해제 할 수 있어, main()의 입장에서 동적할당 해제를 따로 할 필요가 없다.

문제 5의 답: sobel operation은 색이 있는 선을 통해서 모서리를 출력하는데, 이를 양각이라고 본다면, sobel operation을 색이 없는 공백을 통해서 모서리를 출력해 음각으로도 출력할 수 있을 것이다. 이 operation을 isobel(inverse sobel)이라 지칭하겠다.

그럼 inverse sobel을 수행하기 위해 [표 17] , [표 18], [표 19]와 같이 코드를 추가하거나 수정할 수 있다,

[표 ] isobel.hpp

|  |
| --- |
| #pragma once  #include <vector>  #include "artist.hpp"  class isobel : public artist {  public:  isobel();  isobel(int, int, const std::vector<int>&);  virtual ~isobel();  char mapper(int, int);  }; |

[표 ] isobel.cpp

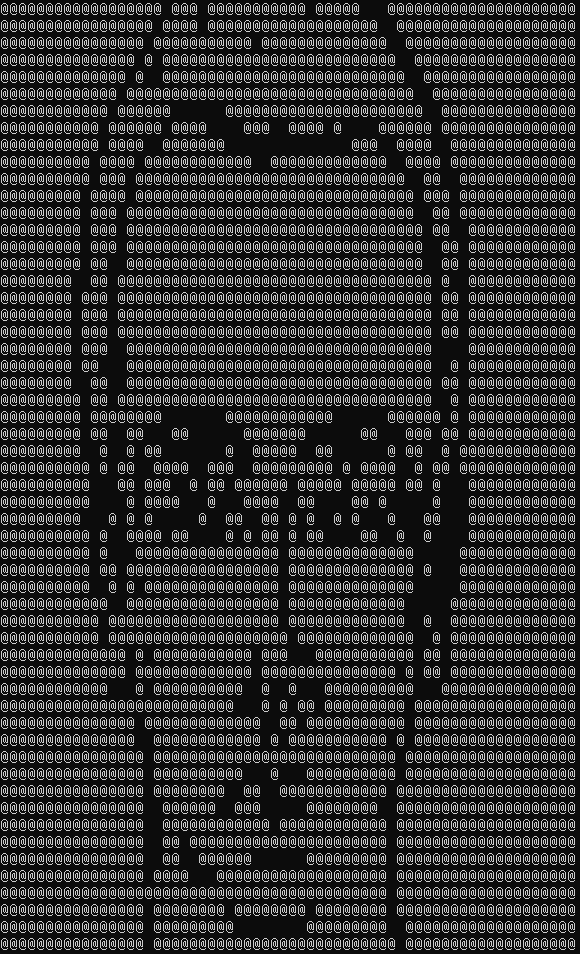
|  |
| --- |
| #include "isobel.hpp"  #include <vector>  isobel::isobel(int \_width, int \_height, const std::vector<int>& \_image\_data) : artist(\_width, \_height, \_image\_data) { }  isobel::~isobel() { }  char isobel::mapper(int x, int y) {  /\*  \* The function that performs "isobel" operation using pixel value of given corrdinate  \*  \* param: int x: x-coordinate  \* param: int y: y-coordinate  \*  \* return: char value: "gradient" result  \*/  int value = get\_pixel(x, y);  int right\_value, below\_value;  bool is\_x\_differ\_50 = false, is\_y\_differ\_50 = false;  if (x != get\_width() - 1) { // Not rightmost pixel  right\_value = get\_pixel(x + 1, y);  if (abs(value - right\_value) >= 50) is\_x\_differ\_50 = true;  }  if (y != get\_height() - 1) { // Not bottommost pixel  below\_value = get\_pixel(x, y + 1);  if (abs(value - below\_value) >= 50) is\_y\_differ\_50 = true;  }  // Map pixel to ASCII code  if (is\_x\_differ\_50 && is\_y\_differ\_50) return ' ';  else if (is\_x\_differ\_50) return ' ';  else if (is\_y\_differ\_50) return ' ';  else return '@';  } |

[표 ] main.cpp

|  |
| --- |
| #include "artist.hpp"  #include "drawer.hpp"  #include "parser.hpp"  #include "classic.hpp"  #include "iclassic.hpp"  #include "sobelx.hpp"  #include "sobely.hpp"  #include "gradient.hpp"  #include "downsample.hpp"  #include "upsample.hpp"  #include "scale.hpp"  #include "isobel.hpp"  #include <iostream>  #include <vector>  using namespace std;  int main(int argc, char\* argv[]) {  if (argc != 4) {  cout << "argc is not 4, but " << argc << endl;  throw;  }  // CREATE PARSER  parser p;  // LOAD IMAGE AND CONFIG  vector<int> tokens = p.load\_image(argv[1]);  vector<string> configs = p.load\_config(argv[2]);  string style\_target = configs[0];  string drawer\_target = configs[1];  char\* path\_output = argv[3];  int width = tokens[0];  int height = tokens[1];  vector<int> vals = { tokens.begin() + 2, tokens.end() };  // CREATE ARTIST  artist\* style;  if (style\_target == "classic") {  style = new classic(width, height, vals);  }  else if (style\_target == "iclassic") {  style = new iclassic(width, height, vals);  }  else if (style\_target == "sobelx") {  style = new sobelx(width, height, vals);  }  else if (style\_target == "sobely") {  style = new sobely(width, height, vals);  }  else if (style\_target == "gradient") {  style = new gradient(width, height, vals);  }  else if (style\_target == "isobel") {  style = new isobel(width, height, vals);  }  else {  throw;  }  // CREATE DRAWER  drawer\* d;  if (drawer\_target == "drawer") {  d = new drawer(style);  }  else if (drawer\_target == "upsample") {  d = new upsample(style);  }  else if (drawer\_target == "downsample") {  d = new downsample(style);  }  else if (drawer\_target == "scale") {  int scale\_x = stoi(configs[2]);  int scale\_y = stoi(configs[3]);  d = new scale(style, scale\_x, scale\_y);  }  else {  throw;  }  // PERFORM DRAWING  string output = d->draw();  cout << output;  // WRITE OUTPUT  p.write\_result(path\_output, output);  delete d;  return 0;  } |

input2.txt를 isobel|drawer로 출력하면 [그림 1]과 같이 출력된다.

[그림 ] isobel example



문제 6의 답: drawer에서는 virtual로 draw()클래스를 선언한다. derived class에서 무조건 function override를 하라고 표기하기 위함이다. 그런데 main에서는 drawer()의 draw()를 사용한다. 이는 base 클래스의 virtual로 선언된 method가 완벽히 추상적이지는 않다는 것을 의미하며, virtual을 사용하는 의미가 퇴색된다. 따라서 drawer()의 draw()를 하는 새로운 derive class를 만들고, drawer()의 draw()는 정의하지 않음으로써 완벽히 추상적으로 만들어야한다.

1. **참고 문헌**

C++ reference : https://en.cppreference.com/w/

2023S object oriented programming slides

Microsoft learn c++ : https://learn.microsoft.com/en-us/cpp/?view=msvc-170