Term project

도형과 문자열을 이용한

로고 만들기

(Making Your Logo)

컴퓨터프로그래밍 II (CSE2035-02)

유연이가 유연해 팀

20231604 정유연, 20231624 한송이

목차

1. 프로젝트 목표 : 도형과 문자열 동시에 출력

2. 기능

1) 도형 : 사각형, 삼각형, 원

2) 문자열 : 알파벳 소문자&대문자, 숫자, 특수문자, 줄바꿈, 알파벳 뒤집기

3) 도형과 알파벳 색상 설정

3. 입출력 예시

1) 컴파일 명령어 : g++ main.cpp color.cpp ppng.cpp draw.cpp drawShape.cpp floodfill.cpp merge.cpp input.cpp printcolorful.cpp -lpng -o lines

```
*****write your string (enter a backslash for line change)*****
your string: Sogang University\\2023\\Computer Science
**********************choose your color**********************
1. red
2. orange
3. yellow
4. green
5. blue
6. purple
7. brown
8. pink
your choice: 3
***************which way do you want to flip it?***************
1. I don't want to flip.
2. from side to side.
3. from up and down.
your choice: 1
**********************choose your shape**********************
1. Rectangle
2. Trianglen
3. Circle
If you input other number, the shape will be default value:NONE
your choice: 3
**********************choose your color**********************
1. red
2. orange
3. yellow
4. green
5. blue
6. purple
7. brown
8. pink
your choice: 4
Your LOGO.png is made!
```

2) 문자열 입력

3) 문자열 색상 선택

4) 문자열 뒤집기 여부

5) 도형 선택

6) 도형 색상 선택

그림 1 입력 예시

7) 출력 파일

그림 2 출력 예시

4. input.h

4.1. Inp 클래스

```
class Inp{
    public:
        void getInput(string* wrd, int *e, int* flip, int* co);
        void findTarget(string wrd, int* cnt, int* max);
        ~Inp(){}

};
```

그림 3 Inp 클래스

1) wrd: 원하는 문자열을 입력받는다. 이때 ₩(백슬래쉬)를 통해 줄바꿈을 할 수 있다.

2) len: 입력받은 문자열의 길이를 저장한다.

3) flip: 문자열을 어떻게 뒤집을지 선택한다.

4) co: 문자의 색깔을 선택한다.

5) findTarget

: 입력받은 wrd의 문자열에서 가로 최대길이인 max와 세로 최대길이인 cnt를 구한다. 구한 값을 바탕으로 메인함수에서 pixel포인터를 선언한다.

```
    inp.getInput(&wrd, &e, &r, &co); //input.cpp

    inp.findTarget(wrd, &cnt, &maxx); //max와 cnt찾음, target.cpp
struct Pixel * str[ HEIGHT*(cnt+1)+cnt ];
```

그림 4 findTarget을 이용해 main함수에서 Pixel 포인터 선언

```
*********choose your color*********
1. red
2. orange
3. yellow
4. green
5. blue
6. purple
7. brown
8. pink
your choice: e
Invalid input. Please enter correct number.
your choice: 0
Invalid input. Please enter correct number.
your choice: 4
*******which way do you want to flip it?*******
1. I don't want to flip.
2. from side to side.
3. from up and down.
your choice: 4
Invalid input. Please enter correct number.
your choice: 2
```

그림 5 잘못된 입력 시 무한루프 예시

6) 뒤집기와 색깔을 선택할 때 주어진 숫자를 입력하지 않으면 무한루프가 돌도록 했다.

4.2. SelectShape 클래스

```
class SelectShape{
    public:
        enum Shape{
            NONE = 'N',
            REC = 'R',
            TRI = 'T',
            CIRCLE = 'C'
        };
        int png_HEIGHT;
        int png_WIDTH;
        int word_height;
        int word_width;
        void select(int* co_s);
        Rectangle rec; //초기 도형 설정
        Triangle tri;
        Circle circle;
        Shape shape;
        SelectShape(int _word_height, int _word_width)
        : word_height(_word_height),word_width(_word_width),rec(0, 0), tri(0), circle(0), ②
        shape(Shape::NONE),png_HEIGHT(0),png_WIDTH(0){} ①
        ~SelectShape(){}
};
#endif
```

그림 5 SelectShape 클래스

1) 멤버변수

열거형 (Enumerated type) Shape은 도형의 모양을 결정한다. 기본값은 도형을 그리지 않는 'N'이고 이를 shape의 생성자로 하였다. -①

글자의 높이 word_height와 너비 word_width, png파일의 높이 png_HEIGHT와 너비 png_WIDTH이다. 글자를 입력 받고, 이것의 높이 및 너비에 따라 도형의 크기와 png 파일의 높이 및 너비를 설정한다.

도형의 객체인 Rectangle, Triangle, Circle은 생성자가 형성될 때 초깃값으로 0을 갖는다. (도형의 크기가 0이다.) -②

2) void select(int* co_s);

도형의 색상을 저장할 co_s를 매개변수로 받으며, 이 함수 내부에서 도형과 도형의 색상을 선택한다. 또한, 도형의 선택에 따라 색상을 결정할 필요가 정해지므로 부울형 shapecolor를 선언하였으며, 초깃값으로 true를 설정하여 도형을 선택하지 않을 때만 false로 설정하며 이를 한 번만 코드를 작성하였다.

ⅰ) switch 문으로 선택한 번호에 따라 shape를 결정한다.

ⅱ) 글자 길이 정보를 바탕으로 도형 크기를 계산한다.

iii) 정해진 도형의 크기를 객체인 각 도형에 전달하여 도형의 크기를 설정한다.

iv) 도형의 정보가 설정된 후, 도형의 색상을 입력하게 하였다. 이때 색상을 올바르게 입력하지 않으면 무한 루프를 벗어날 수 없도록 하였다. (4.2.5)

```
void SelectShape::select(int* co_s){
    PrintColor p;
    p.Print("Please choose the shape for your LOGO ",p.Pink,true);
    cout<<endl;
    p.Print("1. Rectangle\n2. Trianglen\n3. Circle\n",p.Default,true);
    p.Print("If you input other number, the shape will be default value:NONE.\n",p.Default,true);
    p.Print("your choice: ",p.Default,true);
    char choice; cin>>choice;
    bool shapecolor = true;

    int width=0, height=0;

    switch (choice){
        case '1':
            shape=Shape::REC;
            width = std::min(word_width + 4, 1000);
            height = std::min(word_height + (word_height < 9 ? 4 : 5), 1000);

            rec=Rectangle(height, width);
            png_HEIGHT=rec.height;
            png_WIDTH=rec.width;
            break;
```

그림 6 SelectShape의 멤버함수 void select(int* co_s);

5. pixel.hpp

```cpp
#pragma once
#ifndef PIXEL_HPP
#define PIXEL_HPP
#include <stdlib.h>
#include <iostream>
using namespace std;
#define PNG_SETJMP_NOT_SUPPORTED
#include <png.h>

struct Pixel {
    png_byte r, g, b, a;
};
#endif
```

그림 7 pixel.hpp

Pixel 구조체의 r, g, b, a의 4개의 필드에서 PNG 이미지의 각 픽셀을 표현한다.
PNG_SETJMP_NOT_SUPPORTED 매크로를 통해 libpng 라이브러리에서 setjmp/longjmp 기반의 오류 처리를 사용하지 않도록 한다.

6. ppng.h

```cpp
class SetPng{
    public:
        void setpng(int height, int width, Pixel ** str);
        ~SetPng(){}

};
class Png:public SetPng{
    private:
        Pixel ** str;
        int cnt;
        int maxx;
        const char* filename;
    public:
        Png(Pixel ** _str,int _cnt,int _maxx):str(_str),cnt(_cnt),maxx(_maxx),filename("out.png"){};
        ~Png(){}
        void allo();
        void image();
        const char* GetFile() const { return filename; }
};
```

그림 8 SetPng 클래스와 이를 상속받는 Png 클래스

6.1. SetPng 클래스와 void setpng(int height, int width, Pixel ** str);

이차원 배열str의 모든 칸을 투명도(opacity) 0으로 설정하며, png 파일에 버그 생성을 방지한다.

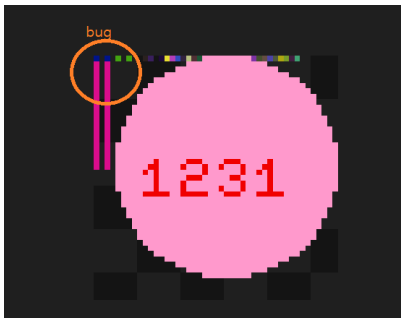

그림 9 버그 예시

```
10  void SetPng::setpng(int height, int width,Pixel** str){
11      for (int col =0 ; col <width; col++) {
12          for (int row= 0; row <height; row++) {
13              str[row][col].r = 0; // red
14              str[row][col].g = 0; // green
15              str[row][col].b = 0; // blue
16              str[row][col].a = 0; // alpha (opacity)
17
18          }
19      }
20
21  }
```

그림 10 SetPng 클래스의 void setpng(int height, int width, Pixel ** str);

6.2. Png 클래스

1) 멤버 변수: str은 그림을 그릴 pixel, cnt는 문자열에서 세로의 최대길이, maxx는 문자열에서 가로의 최대길이이다.

2) 생성자 : pixel에 할당해줄 가로, 세로의 최대길이를 설정한다.

3) void allo() : 문자열의 가로 maxx, 세로 cnt+1 길이의 직사각형만큼의 pixel을 할당한다.

4) void image() : png파일을 여는 코드가 있다.

6.3. Png_shape 클래스

```
31  class Png_shape:public SetPng {
32  private:
33      Pixel ** str;
34      SelectShape& select;
35      int height; //png_HEIGHT, png_WIDTH
36      int width;
37      const char* filename;
38
39  public:
40      Png_shape(Pixel ** _str,SelectShape& _select) :
41      str(_str),select(_select), height(select.png_HEIGHT), width(select.png_WIDTH),filename("shape.png"){}
42      ~Png_shape(){}
43      void allo();
44      void image();
45      const char* GetFile() const { return filename; }
46  };
```

그림 11 Png_shape 클래스

Png 클래스와 거의 같지만 도형을 위해 새로운 클래스를 만들었다. 추가된 멤버변수는 도형의 정보를 담고 있는 SelectShape& select과 도형이 생성될 파일의 이름인 const char* filename가 있다. 또한 함수 const char* GetFile() const { return filename; }로 filename의 정보를 반환한다.

7. color.hpp

```
#ifndef COLOR_HPP
#define COLOR_HPP
class Color{
   protected:
      int r, g, b,a;
   public:
      Color(): r(0), g(0), b(0),a(255){}
      ~Color(){}

      int getR() const { return r; }
      int getG() const { return g; }
      int getB() const { return b; }

      void Red(){ r=240; g=0; b=0; a=255;}
      void Pink(){ r=255; g=153; b=204; a=255;}
      void Orange(){ r=255; g=128; b=0; a=255;}
      void Yellow(){ r=255; g=255; b=70; a=255;}
      void Green(){ r=0; g=153; b=0; a=255;}
      void Blue(){ r=0; g=100; b=255; a=255;}
      void Purple(){ r=153; g=51; b=255; a=255;}
      void Brown(){ r=139; g=69; b=19; a=255;}
      Color* setColor(int);
};
#endif
```

그림 12 color.hpp

1) 생성자 : 기본 색깔을 검정으로 한다.

2) 색깔 함수

: 총 8가지 색깔의 함수를 만들어 color class의 멤버변수인 r, g, b, a를 설정한다.

3) Color* setColor(int)

: 함수 내에서 새로운 color 객체를 만들어 매개변수에 따라 각 색깔의 Color*를 반환하도록 했다.

8. draw.h

```
class Draw{
    public:
        Draw (Pixel ** _str, int _flip,int _cnt, Color * _color){
            str = _str;
            flip = _flip;
            color = _color;
            cnt = _cnt;
        }
        void fill(int i, int a, int b);
        void draw(int i, char c);
        void alph(Pixel** str, string wrd, int len);
    private:
        Pixel ** str;
        int z=0; // 몇번째 줄인지 나타내는 변수
        int flip; // 어떻게 뒤집는지
        int cnt; //cnt 줄 바꿈 개수
        Color * color; // 글자 색 저장
};
```

그림 13 color.hpp

1) flip: 어떻게 뒤집는지 선택에 따라 fill함수에서 알파벳을 채우는 방식이 달라진다.

2) cnt: 위아래 뒤집기에서 세로의 최대값이 필요하다.

3) alph(Pixel** str, string wrd, int len) : 문자열의 길이인 len만큼 반복문을 돌리며 draw 함수를 호출한다

4) draw(int i, char c)

: switch문을 이용해 문자 c에 따라 각 문자를 그리도록 했다. 이때 fill함수를 이용했다. i는 같은 줄 안에서 몇 번째 글자인지를 뜻한다.

```
switch(c){
case ' ':
break;

case 'a':
fill(i, 2, 2);fill(i,2,3);fill(i,2,4);
 fill(i, 3, 5);
fill(i, 4, 2); fill(i, 4, 3); fill(i,4,4);fill(i,4,5);
fill(i,5,1);fill(i,5,5);
fill(i, 6, 2); fill(i, 6, 3); fill(i,6,4);fill(i,6,5);
break;

case 'b':
```

그림 13 draw 함수의 switch문

알파벳 대문자와 소문자, 숫자 0부터 9, 각종 특수문자(!@#$%^&*-=,./등)를 포함한다.

5) fill(int i, int a, int b)

: 이차원 배열의 str(Pixel**)의 지정된 좌표를 채우도록 했다. I는 같은 줄에서 몇 번째 글자인지, (a, b)는 7x7크기의 한 박스에서 채울 좌표를 뜻한다.

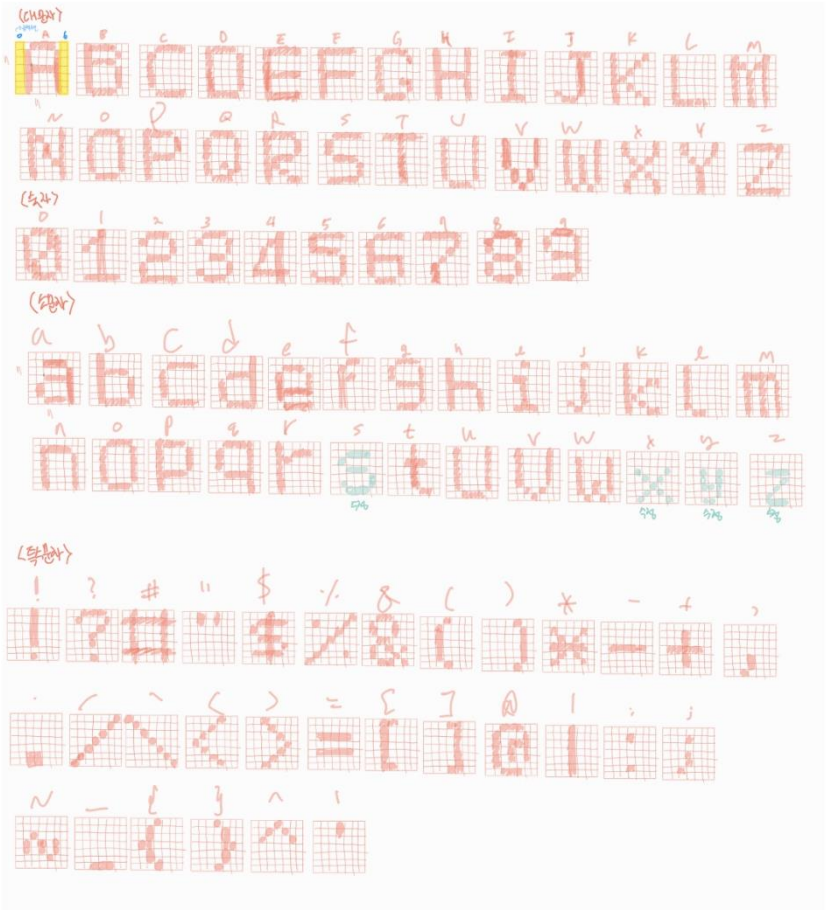

그림 15 fill 함수를 위해 스케치한 폰트

9. drawShape.hpp

```
#pragma once
#ifndef DRAWSHAPE_HPP
#define DRAWSHAPE_HPP
#include <iostream>
#include "shape.hpp"
#include "pixel.hpp"
#include "color.hpp"
#include "input.h"
#include "ppng.h"
using namespace std;
class draw_data:public SetPng{
    public:
        Pixel ** str;
        Color * color;
        SelectShape& select;
        draw_data(Pixel ** _str, Color * _color,SelectShape& _select)
        :str(_str), color(_color),select(_select){}
        ~draw_data(){}
        void draw(int x1, int x2,  int y2,int height,Pixel** str, Color * color);
        void draw_line(int x1, int x2, int y1, int y2,Pixel** str,Color * color);
        void setpng(int height, int width,Pixel** str);
        void draw_rectangle(Rectangle& rectangle,Pixel** str, Color * color);
        void draw_triangle(Triangle& triangle,Pixel** str, Color * color);
        void draw_circle(Circle& circle,Pixel** str, Color * color);
        void DrawShape(SelectShape& select);
};
#endif
```

1) 클래스 SetPng의 멤버함수 이용을 위해 이를 상속한다. (6. ppng.h)

2) 멤버 변수

str은 그림을 그릴 2차원 배열의 Pixel 구조체, color은 도형의 색상, select는 도형의 정보를 담은 객체이다

그림 16 drawShape.hpp

9.1. void draw(int x1, int x2, int y2,int height, Pixel** str, Color * color);

가로 x1<=col<=x2, 세로 y2-height<=row<y2 인 str의 유효 범위 내에 color의 색상으로 칸을 채운다. (수직) 사각형 및 원 그리기에 이용.

9.2. void draw_line(int x1, int x2, int y1, int y2, Pixel** str, Color * color);

Str의 유효범위 내에 (x1, y1)에서 (x2, y2)의 칸을 color의 색상으로 채운다. (대각선)

삼각형과 원 그리기에 이용하며 브레젠험 직선 알고리즘(Bresenham's line algorithm)[1]을 채용했다.

9.3. void setpng(int height, int width, Pixel** str);

클래스 SetPng 속한 멤버 함수로(6. ppng.h), 클래스 내부에 호출을 하였다.

str의 모든 칸을 투명도(opacity)를 0으로 설정하여 png 파일 버그 생성을 방지한다.

drawShape함수 호출 시 png파일 초기 설정을 위해 내부에서 호출한다.

9.4. void DrawShape(SelectShape& select);

최종적으로 메인 함수에서 호출하는 함수이다. SelectShape에 저장된 도형의 정보를 바탕으로 그림을 그릴 도형을 결정한다. SelectShape의 열거형 (Enumerated type)인 Shape형 shape의 값에 따라 호출할 함수를 결정하며, setpng함수를 호출하여 png를 초기화하고 draw_도형이름 함수를 호출한다.

---

[1] "Bresenham's line algorithm", 위키백과, [웹사이트], 2023.11.23. 수정, 2023.12.01. 접속, https://en.wikipedia.org/wiki/Bresenham%27s_line_algorithm.

9.5. draw_도형이름(도형이름&도형이름, Pixel** str, Color * color);

5-1) void draw_rectangle(Rectangle& rectangle, Pixel** str, Color * color);

내부에 draw함수 호출 후 SelectShape에 저장된 정해진 좌표를 바탕으로 사각형 칸을 수직으로 채워간다.

5-2) void draw_triangle(Triangle& triangle, Pixel** str, Color * color);

SelectShape에 저장된 좌표를 바탕으로 draw_line 함수를 호출하여 삼각형의 세 변을 그린다. 이후 길이의 조건(짝수 혹은 홀수)에 구애받지 않기 위해 상단 꼭짓점을 draw함수로 채워 삼각형 라인을 완성한다. 완성된 삼각형 라인의 중심 좌표인 centerY, centerX를 floodfill 함수의 시작점으로 실행하여 삼각형의 내부를 채운다.

5-3) void draw_circle(Circle& circle, Pixel** str, Color * color);

SelectShape에 저장된 원의 반지름을 바탕으로 정수 i를 360도까지 변화시키며 새로운 x 및 y좌표를 찾아낸다. 또한, 반지름이 커짐에 따라 원의 외곽선에 구멍이 생기는 문제를 다른 x 및 y로 보완한다.

새로운 x1 및 y1 : 기존의 반지름-1
새로운 x2 및 y2 : 기존의 반지름-1 및 상단으로 한 칸 이동(y에 -1)
새로운 x3 및 y3 : 기존의 반지름+1 및 하단으로 한 칸 이동(y에 +1)

이러한 x 및 y좌표가 str의 범위 내에 존재할 때, draw 함수를 호출하여 원의 외곽선을 한 칸씩 채워 원의 외곽선을 완성한다. 이후 원의 중심을 floodfill 함수의 시작점으로 하여 원을 채운다.

10. floodfill.hpp

floodfill.hpp > floodfill(Pixel **, Color *, SelectShape &, int, int)

```
1  #include "pixel.hpp"
2  #include "color.hpp"
3  #include "shape.hpp"
4  #include "input.h"
5  #include <cmath>
6  Pixel**floodfill(Pixel**str, Color* color,SelectShape & sel,int row, int col);
```

그림17 floodfill.hpp

1)매개변수

: str은 그림을 그릴 2차원 배열의 Pixel 구조체, color은 채울 색상, select는 png의 범위 등 도형의 정보를 담은 객체, row와 col은 시작 좌표값이다.

```cpp
#include "floodfill.hpp"
using namespace std;
Pixel**floodfill(Pixel**str, Color* color,SelectShape & sel,int row, int col){
    int HEIGHT=sel.png_HEIGHT;
    int WIDTH=sel.png_WIDTH;

    // 범위 유효성 체크
    if (row >= HEIGHT || col >= WIDTH || row < 0 || col < 0) {
        return str;
    }

    bool select_color=false;
    if(str[row][col].r == color->getR()&&str[row][col].g == color->getG()&&str[row][col].b == color->getB()){
        select_color=true; //사용자가 선택한 색이 맞다!(외곽선을 만났다!)
    }
    if (row >= HEIGHT || col >= WIDTH || row < 0 || col < 0 ||select_color) {
        return str; //범위 넘거나 사용자가 선택한 색(외곽선을 만났다!)이면 재귀 함수 종료.
    }

    str[row][col].r = color->getR();
    str[row][col].g = color->getG();
    str[row][col].b = color->getB();
    str[row][col].a = 255;
    floodfill(str, color, sel,row + 1, col); //우
    floodfill(str, color,sel,row - 1, col); //좌
    floodfill(str, color,sel,row, col + 1); //하
    floodfill(str, color,sel,row, col - 1); //상

    return str;
}
```

재귀

그림 18 floodfill.cpp 재귀함수

2) 정수 HEIGHT와 WIDTH로 png파일의 범위(외곽선)를 설정한다.
3) 부울형 select_color는 외곽선을 판별하기 위함이다. 조건문을 통해 str 배열에서 매개변수로 받은 color가 가리키는 색상과 같을 경우 true가 된다. 즉, true는 외곽선을 만났음을 의미한다.

4) 범위를 벗어나거나 외곽선을 만났을 경우 str을 반환하여 재귀하지 않고 함수를 종료한다.

5) 모든 조건을 만족하면 'str' 배열의 해당 위치를 채우고, floodfill 함수를 재귀적으로 호출 후 str을 반환한다. 재귀적으로 호출하는 시작 좌표는 현재 좌표에서 상하좌우로 한 칸씩 이동한 위치다. 범위를 벗어나거나 2)의 외곽선을 만날 때까지 3) 과정은 재귀적으로 계속 반복된다.

11. merge.hpp

```cpp
class Png_merge:public SetPng{
    private:
        Pixel ** str;
        int height;
        int width;


    public:
        Png_merge(Pixel ** _str,int height,int width):str(_str),height(height), width(width){};
        ~Png_merge(){}
        void overlayImages(const char* largeImageFile, const char* smallImageFile, const char* filename);
        void allo();
        void image(const char * filename);
};
```

그림 19 merge.hpp

11.1. 멤버 변수 및 void alloc(); 과 void image(const char * filename);의 역할은 ppng.h의 void allo(); 및void image();의 역할과 동일하되, const char * filename로 생성할 png파일의 이름을 지정할 수 있게 했다.

11.2. void overlayImages(const char* largeImageFile, const char* smallImageFile, const char* filename);

첫번째와 두번째 매개변수는 합병할 파일의 이름이다. (예: shape.png, word.png) 세번째 매개변수는 합병된 파일의 이름이다. 이 함수는 항상 첫번째 파일의 크기가 더 크다고 가정하며, 이는 메인함수에서 판별한 후 함수를 호출한다.

1) 큰 이미지, 작은 이미지, 출력 파일을 연다. 출력 파일의 이름은 세번째 매개변수로 받는다.

2) png 파일 헤더(header) 세팅

파일 첫 8바이트 읽고 헤더 유효성 체크한다. 즉, 큰 이미지 파일 및 작은 이미지 파일이 유효한 png파일인지 확인한다. 유효한 파일이 아닐 경우 오류 메시지를 출력하고 파일을 닫으며 함수를 종료한다.

3) libpng 구조체 및 정보 구조체를 생성 및 초기화 하며 실패 시, 오류 메시지를 출력하고 파일을 닫으며 함수를 종료한다. (큰 이미지, 작은 이미지, 출력 파일)

4) libpng에 파일 연결하고, 파일의 첫 8바이트는 PNG 시그니처로 이미 읽은 후이므로 파일 포인터를 올바른 위치로 설정한다. 이후 png 파일 정보를 읽는다.

5) 작은 이미지를 큰 이미지의 중앙에 배치하기 위한 좌표 계산(좌상단 꼭짓점)을 위해 libpng 라이브러리에서 이미지의 속성을 가져오는 함수들을 호출하며, 너비와 높이 좌표를 계산한다.

6) 큰 이미지 파일에서 행 별로 데이터 읽기 및 쓰기를 위해 libpng 라이브러리를 사용하여 PNG 이미지를 처리하기 위한 동적으로 메모리를 할당한다.

```
150     // 큰 이미지 파일에서 행 별로 데이터 읽기 및 쓰기
151     png_bytep largeRow = (png_bytep)malloc(sizeof(png_byte) * largeWidth * 4);
152     png_bytep smallRow = (png_bytep)malloc(sizeof(png_byte) * smallWidth * 4);
```

그림 20 메모리의 동적 할당

7) 반복문으로 큰 이미지 파일에서 행 데이터 읽으며 5)의 좌표값의 범위에 해당할 경우, 작은 이미지의 행 별 데이터를 memcpy를 이용하여 덮어쓴다. 단, 작은 이미지의 투명도(opacity)가 255일 경우에만 진행한다. 이는 투명할 경우 덮어쓰지 않기 위함이다.

12. printcolorful.hpp

```cpp
#include <iostream>
using namespace std;
class PrintColor{
    public:
        string Red="\033[0;31m";
        string Orange="\033[38;5;208m";
        string Yellow="\033[0;33m";
        string Green="\033[0;32m";
        string Blue="\033[0;34m";
        string Purple="\033[38;5;93m";
        string Brown="\033[38;5;94m";
        string Pink= "\033[38;5;13m";
        string Default= "\033[0m";

        bool bold;
        bool tilt;
        void Print(const char * text,string color= "\033[0m", bool bold=false, bool tilt=false);
        PrintColor(){}
        ~PrintColor(){}
};
```

그림 21 printcolorful.hpp

1) ANSI 이스케이프 코드[2]를 이용하여 프로그램이 텍스트를 출력(cout)할 때 색상 및 굵기(bold), 기울임(italic)을 조정할 수 있게 했다. 직접 지원하지 않는 색상은 256 색상 모드[3]를 참조했다.

2) 멤버 변수에서 string color는 각각의 색상 정보, bool bold와 tilt로 굵기 및 기울임 활성화 여부를 설정하며 기본값은 false로 하였다.

3) void Print(const char * text, string color= "₩033[0m", bool bold=false, bool tilt=false);

text는 출력할 텍스트의 정보이며, const로 하였다. Color는 선택할 색상이며, 기본값은 검정이다. 네번째와 다섯번째 매개변수는 굵기 및 기울임 활성화 여부로 기본값은 false로 하여 원래의 글꼴이 출력되게 하였다. 함수의 마지막에 모든 속성을 초기화하여 일반 출력(cout)시 영향을 주지 않도록 하였다.

```cpp
void Inp::getInput(string* wrd, int* e, int* r, int* co){
  //문자열 입력
  PrintColor p;   객체 생성
  p.Print("write the word (enter a backslash for line change) : ",p.Default,true);
  getline(cin, *wrd);
  *e=wrd->length();
  int bu;
  //색깔 설정

  p.Print("*********choose your color*********\n",p.Default,true);   텍스트, 색상 기본값, 굵기(bold) 활성화
  p.Print("1. red\n", p.Red);
  p.Print("2. orange\n", p.Orange);   텍스트, 색상 설정
  p.Print("3. yellow\n", p.Yellow);
  p.Print("4. green\n", p.Green);
  p.Print("5. blue\n", p.Blue);
  p.Print("6. purple\n", p.Purple);
  p.Print("7. brown\n", p.Brown);
  p.Print("8. pink\n", p.Pink);
```

그림 22 void Print(const char * text,string color= "₩033[0m", bool bold=false, bool tilt=false); 예시

[2] "이스케이프 문자", 위키백과, [웹사이트], 2022.03.11. 수정, 2023.12.01. 접속, https://w.wiki/8SFR.

[3] "ANSI Escape Sequences", GitHub Gist, [웹사이트], 2021.11.03 수정, 2023.12.01. 접속, https://gist.github.com/fnky/458719343aabd01cfb17a3a4f7296797.

```cpp
#include "printcolorful.hpp"
using namespace std;

void PrintColor::Print(const char * text,string color, bool bold, bool tilt){
    cout << color.c_str();  // 색 설정

    if(bold) {
        cout << "\033[1m";  // 굵게
    }

    if(tilt) {
        cout << "\033[3m";  // 기울임
    }

    cout << text;          // 텍스트 출력

    cout << "\033[0m" ; // 모든 속성 초기화
}
```

그림 23 함수의 마지막에 모든 속성을 초기화하여 일반 출력(cout)을 할 때 영향을 주지 않음

13. 동적할당 해제

13.1. main.cpp

```cpp
Color *color;
color=color->setColor(co);

int maxx=1, cnt=0;
findTarget(wrd, &cnt, &maxx, e); //max와 cnt찾음, target.cpp

struct Pixel * str[ HEIGHT*(cnt+1)+cnt ];        동적할당
Png png(str,cnt,maxx);
png.allo();//png.cpp
int z=0;
Draw draw(str,r,cnt,maxx,color);

int word_height=HEIGHT*(cnt+1)+cnt;
int word_width=WIDTH*maxx;

SelectShape sel(word_height, word_width); //png 사이즈 정보 sel의 png_HEIGHT, png_WIDTH
sel.select(&co_s);
Color *color_shape;
color_shape=color_shape->setColor(co_s);
```

```cpp
struct Pixel * str_shape[sel.png_HEIGHT];
Png_shape png_shape(str_shape, sel);
png_shape.allo();//png.cpp
//png 가로 세로 정보 : //sel : png_HEIGHT, png_WIDTH = png_shape : height, width
draw_data d(str_shape,color_shape,sel);
d.DrawShape(sel);
png_shape.image();

int merge_height, merge_width;
merge_height=(word_height<sel.png_HEIGHT)?sel.png_HEIGHT:word_height;
merge_width=(word_width<sel.png_WIDTH)?sel.png_WIDTH:word_width;

struct Pixel * str_logo[merge_height];
Png_merge png_merge(str_logo, merge_height, merge_width);
png_merge.allo();
png_merge.image("mergesource.png");

png_merge.overlayImages("mergesource.png",png_shape.GetFile(),"new_shape.png");
```

그림 24 main.cpp 동적 할당

```cpp
    delete color;
    delete color_shape;        동적할당 해제

    return 0;
}
```

그림 25 main.cpp 동적 할당 해제

13.2. ppng.h의 png 클래스

```cpp
void Png::allo(){
    /* allocate image data */
    //str=new Pixel*[ HEIGHT*(cnt+1)+cnt ];
    for(int row=0;row<HEIGHT*(cnt+1)+cnt;row++){
        str[row]=new Pixel[WIDTH*maxx];
    }
}

void Png::image(){
    /* begin writing PNG File */
    FILE*f = fopen("out.png", "wb");
     if (!f) {
    fprintf(stderr, "could not open out.png\n");
    exit(1);
    }
    png_structp png_ptr;
    png_infop info_ptr;
    png_ptr = png_create_write_struct(PNG_LIBPNG_VER_STRI
```

그림 26 ppng.h의 Png클래스 내부 동적할당

```cpp
    /* write image data to disk */
    png_write_image(png_ptr, (png_byte **)str);
    /* clean up PNG-related data structures */
    png_write_end(png_ptr, NULL);

    png_destroy_write_struct(&png_ptr, &info_ptr);
    fclose(f);

    for(int i = 0; i <HEIGHT*(cnt+1)+cnt; i++) {
        delete[] str[i];
    }
}

void Png_shape::allo(){
    /* allocate image data */
    for(int row=0;row<height;row++){
        str[row]=new Pixel[width]; //png_HEIGHT, png_WIDTH
    }
}
```

그림 27 ppng.h의 Png클래스 내부 동적할당 해제

13.3. ppng.h의 png_shape클래스

```
70  void Png_shape::allo(){
71      /* allocate image data */
72      for(int row=0;row<height;row++){
73          str[row]=new Pixel[width]; //png_HEIGHT, png_WIDTH
74      }
75  }
76
77  void Png_shape::image(){
78      /* begin writing PNG File */
79      FILE*fs = fopen("shape.png", "wb");
80      if (!fs) {
81      fprintf(stderr, "could not open shape.png\n");
82      exit(1);
83      }
84
85      png_structp png_ptr;
86      png_infop info_ptr;
```

그림 28 ppng.h의 Png_shape클래스 내부 동적할당

```
109     png_write_end(png_ptr, NULL);
110     png_destroy_write_struct(&png_ptr, &info_ptr);
111     fclose(fs);
112     for(int i = 0; i < height; i++) {
113         delete[] str[i];
114     }
115 }
116
```

그림 29 ppng.h의 Png_shape클래스 내부 동적할당 해제

13.4. 메모리누수 점검

```
root@DESKTOP-0NKCM8O:~/tasks# g++ -g -o tasks main.cpp color.cpp ppng.cpp draw.cpp drawShape.cpp floodfill.cpp merge.cpp in
put.cpp printcolorful.cpp -lpng -o lines
root@DESKTOP-0NKCM8O:~/tasks#  valgrind --leak-check=full ./lines
==32935== Memcheck, a memory error detector
==32935== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==32935== Using Valgrind-3.18.1 and LibVEX; rerun with -h for copyright info
==32935== Command: ./lines
```

그림 30 Valgrind 프로그램을 통한 메모리 누수 및 동적 할당 해제 점검

```
=32935== HEAP SUMMARY:
=32935==     in use at exit: 0 bytes in 0 blocks
=32935==   total heap usage: 188 allocs, 188 frees, 1,065,449 bytes allocated
=32935==
=32935== All heap blocks were freed -- no leaks are possible
```

그림 31 Valgrind 프로그램의 힙 영역 요약 결과

Valgrind 프로그램을 통해 메모리를 점검한 결과 메모리 누수는 없으며 동적 할당이 모두 해제되었음을 알 수 있다.

이상입니다.