**Homework 5**

|  |
| --- |
| 그림입니다. 원본 그림의 이름: YU_UI_RGB-10.png 원본 그림의 크기: 가로 2256pixel, 세로 3047pixel 프로그램 이름 : Adobe ImageReady |

|  |  |
| --- | --- |
| 과목명 | 객체지향프로그래밍과자료구조 |
| 교수님 | 김영탁 교수님 |
| 이 름 | 김주환 |
| 학 번 | 21812158 |
| 일 자 | 2021.10.06.수 |

|  |
| --- |
| /\* main.cpp \*/  /\* Description  \* 타원기둥에 대한 클래스  \* Programmed by J. H. Kim  \* Last updated : 2021-10-06 \*/  #include <iostream>  #include <fstream>  #include <string>  #include "Color.h"  #include "Shape.h"  #include "Elps.h"  #include "ElpsCylinder.h"  using namespace std;  int main()  {  fstream fout;    // 클래스 생성 및 초기화  Shape shape(1, 1, 0, RGB\_BLACK, "Shape");  Elps red\_elps(8, 8, 3.0, 4.0, 0, RGB\_RED, "Red\_Elps");  ElpsCylinder blue\_elpcyl(9, 9, 5.0, 6.0, 7.0, 0.0, RGB\_BLUE, "Blue\_Elp\_Cyl");  fout.open("output.txt", 'w'); // 파일 열기  if (fout.fail())  {  cout << "Failed in opening output.txt file !!" << endl;  exit;  }    // 출력  fout << "List of shapes using operator<<() friend function " << endl;  fout << shape << endl;  fout << red\_elps << endl;  fout << blue\_elpcyl << endl;  fout.close(); // 파일 닫기  return 0;  } // end of main() |
| /\* Color.h \*/  #ifndef C  #define C  #include <Windows.h>  #include <iostream>  #include <string>  #include <iomanip>  using namespace std;  // COLORREF is defined in <Windows.h>  // The COLORREF value is used to specify an RGB color,  // in hexadecimal form of 0x00bbggrr  const COLORREF RGB\_BLACK = 0x00000000;  const COLORREF RGB\_RED = 0x000000FF;  const COLORREF RGB\_GREEN = 0x0000FF00;  const COLORREF RGB\_BLUE = 0x00FF0000;  const COLORREF RGB\_ORANGE = 0x0000A5FF;  const COLORREF RGB\_YELLOW = 0x0000FFFF;  const COLORREF RGB\_MAGENTA = 0x00FF00FF;  const COLORREF RGB\_WHITE = 0x00FFFFFF;  ostream& printRGB(ostream& ostr, const COLORREF color);  // RGB color code chart: https://www.rapidtables.com/web/color/RGB\_Color.html  /\* Note: RGB(red, green, blue) macro also provides COLORREF data  . RGB(FF, 00, 00) => 0x000000FF (RGB\_RED)  . RGB(00, FF, 00) => 0x0000FF00 (RGB\_GREEN)  . RGB(00, 00, FF) => 0x00FF0000 (RGB\_BLUE)  \*/  #endif // !C |
| /\* Shape.h \*/  #ifndef S  #define S  #include <string>  #include "Color.h"  using namespace std;  class Shape  {  friend ostream& operator<<(ostream&, Shape&);  public:  Shape(); // default constructor  Shape(string name); // 생성자  Shape(int px, int py, double angle, COLORREF color, string name); // constructor  ~Shape(); // 소멸자  void draw(); // 실제 그릴 때 사용되는 정보 출력  void rotate(double rt\_ang) { angle += rt\_ang; } // 회전  void move(int dx, int dy) { pos\_x += dx; pos\_y += dy; } // 좌표 이동  void print(ostream&);  int get\_pos\_x() const { return pos\_x; }  int get\_pos\_y() const { return pos\_y; }  void set\_pos\_x(int x) { pos\_x = x; }  void set\_pos\_y(int y) { pos\_y = y; }  void setName(string n) { name = n; }  string getName() { return name; }  Shape& operator=(const Shape& s); // 대입연산자  protected:  int pos\_x; // position x  int pos\_y; // position y  double angle; // in radian  string name;  COLORREF color; // COLORREF is defined in <Windows.h>  };  #endif // !S |
| /\* Shape.cpp \*/  #include <iomanip>  #include "Shape.h"  #include "Color.h"  ostream& operator<<(ostream& fout, Shape& s) {  fout.setf(ios::fixed);  fout << " " << s.name;  fout << " : pos (" << setw(3) << s.pos\_x << ", " << setw(3) << s.pos\_y;  fout << "), angle (" << showpoint << setprecision(2) << s.angle;  fout << "), color (" << setw(6) << setfill('0') << uppercase << hex << s.color << ")";  fout << setfill(' ');    return fout;  }  Shape::Shape()  : pos\_x(), pos\_y(), angle(), name(), color()  { // default constructor  pos\_x = 0;  pos\_y = 0;  angle = 0.0;  name = "";  color = RGB\_BLACK;  }  Shape::Shape(string name)  : name(name)  {  pos\_x = 0;  pos\_y = 0;  angle = 0.0;  color = RGB\_BLACK;  }  Shape::Shape(int px, int py, double angle, COLORREF color, string name)  :pos\_x(px), pos\_y(py), angle(angle), color(color), name(name)  { // constructor  pos\_x = px;  pos\_y = py;  this->angle = angle;  this->color = color;  this->name = name;  }  Shape::~Shape() {  //cout << "Shape::Destructor (" << name << ")." << endl;  }  void Shape::draw() {  /\* virtual function that will be late-binded to sub-class's draw() \*/  }  void Shape::print(ostream& fout) {  fout.setf(ios::fixed);  fout << " " << name;  fout << " : pos (" << setw(3) << pos\_x << ", " << setw(3) << pos\_y;  fout << "), angle (" << showpoint << setprecision(2) << angle;  fout << "), color (" << setw(6) << setfill('0') << uppercase << hex << color << ")";  fout << setfill(' ');  }  Shape& Shape::operator=(const Shape& s) {  pos\_x = s.pos\_x;  pos\_y = s.pos\_y;  angle = s.angle;  name = s.name;  color = s.color;  return \*this;  } |
| /\* Elps.h \*/  #ifndef E  #define E  #include <string>  #include "Shape.h"  #include "Color.h"  class Elps : public Shape // Ellipse 타원  {  friend ostream& operator<<(ostream&, const Elps&);  public:  Elps(); // 생성자  Elps(string name); // 생성자  Elps(int px, int py, double r1, double r2, double ang, COLORREF clr, string name); // 생성자  ~Elps(); // 생성자  double getArea() const; // 타원 면적 계산  void draw(); // 타원 작도에 필요한 파라메터 출력  void print(ostream&); // 타원에 대한 출력기능  double getRadius\_1() const { return radius\_1; }  double getRadius\_2() const { return radius\_2; }  void setRadius(double r1, double r2) { radius\_1 = r1; radius\_2 = r2; }  Elps& operator=(const Elps& elp); // 대입연산자  protected:  double radius\_1;  double radius\_2;  };  #endif // !E |
| /\* Elps.cpp \*/  #include "Elps.h"  using namespace std;  ostream& operator<<(ostream& fout, const Elps& e) {  fout << " " << e.name;  fout << " : pos (" << setw(3) << e.pos\_x << ", " << setw(3) << e.pos\_y;  fout << "), angle (" << e.angle;  fout << "), color (" << setw(6) << setfill('0') << uppercase << hex << e.color;  fout << "), radius (" << setprecision(2) << e.radius\_1 << ", " << setprecision(2) << e.radius\_2;  fout << "), ellipse area (" << e.getArea() << ")";  fout << setfill(' ');  return fout;  }  Elps::Elps()  : Shape()  {  radius\_1 = 0;  radius\_2 = 0;  }  Elps::Elps(string name)  : Shape(name)  {  radius\_1 = 0;  radius\_2 = 0;  }  Elps::Elps(int px, int py, double r1, double r2, double ang, COLORREF clr, string name)  : Shape(px, py, ang, clr, name)  {  radius\_1 = r1;  radius\_2 = r2;  }  Elps::~Elps() {  //cout << "Elps::Destructor (" << name << ")." << endl;  }  double Elps::getArea() const {  return radius\_1 \* radius\_2 \* 3.141592;  }  void Elps::draw() {  /\* virtual function that will be late-binded to sub-class's draw() \*/  }  void Elps::print(ostream&) {  cout << " " << name;  cout << "pos (" << setw(3) << pos\_x << ", " << setw(3) << pos\_y;  cout << "), angle (" << angle;  cout << "), color (" << setw(6) << setfill('0') << uppercase << hex << color;  cout << "), radius (" << setprecision(2) << radius\_1 << ", " << setprecision(2) << radius\_2;  cout << "), ellipse area (" << getArea() << ")";  cout << setfill(' ');  }  Elps& Elps::operator=(const Elps& elp) {  Shape::operator=(elp);  radius\_1 = elp.radius\_1;  radius\_2 = elp.radius\_2;  return \*this;  } |
| /\* ElpsCylinder.h \*/  #ifndef EC  #define ED  #include <string>  #include "Elps.h"  #include "Color.h"  using namespace std;  class ElpsCylinder : public Elps // 타원 기둥  {  friend ostream& operator<<(ostream&, const ElpsCylinder&);  public:  ElpsCylinder(); // default constructor  ElpsCylinder(string n); // 생성자  ElpsCylinder(int px, int py, double r1, double r2, double h, double ang, COLORREF clr, string n); // 생성자  virtual ~ElpsCylinder(); // 소멸자  double getArea() const; // 타원기둥 표면적  double getVolume() const; // 타원기둥 체적  void draw();  void print(ostream&);  ElpsCylinder& operator=(const ElpsCylinder& right); // 대입연산자  protected:  double height; // Cylinder height  };  #endif // !EC |
| /\* ElpsCylinder.cpp \*/  #include "ElpsCylinder.h"  using namespace std;  ostream& operator<<(ostream& ostr, const ElpsCylinder& elpcyl) {  ostr << elpcyl.name;  ostr << " : pos (" << setw(3) << elpcyl.pos\_x << ", " << setw(3) << elpcyl.pos\_y;  ostr << "), angle (" << elpcyl.angle;  ostr << "), color (" << setw(6) << setfill('0') << uppercase << hex << elpcyl.color;  ostr << "), radius (" << elpcyl.radius\_1 << ", " << elpcyl.radius\_2;  ostr << "), height (" << elpcyl.height<< ")," << endl;  ostr << " ellipse area (" << elpcyl.Elps::getArea();  ostr << "), elp\_cyl area (" << elpcyl.getArea();  ostr << "), elp\_cyl volume (" << elpcyl.getVolume() << ")";  ostr << setfill(' ');  return ostr;  }  ElpsCylinder::ElpsCylinder()  : Elps()  {  height = 0;  }  ElpsCylinder::ElpsCylinder(string n)  : Elps(n)  {  height = 0;  }  ElpsCylinder::ElpsCylinder(int px, int py, double r1, double r2, double h, double ang, COLORREF clr, string n)  : Elps(px, py, r1, r2, ang, clr, n)  {  height = h;  }  ElpsCylinder::~ElpsCylinder() {  //cout << "ElpsCylinder::Destructor (" << name << ")." << endl;  }  double ElpsCylinder::getArea() const {  double c = 2.0 \* 3.141592 \* (radius\_1 + radius\_2) / 2.0;  return c \* height + 2.0 \* Elps::getArea();  }  double ElpsCylinder::getVolume() const {  return height \* radius\_1 \* radius\_2 \* 3.141592;  }  void ElpsCylinder::draw() {  /\* virtual function that will be late-binded to sub-class's draw() \*/  }  void ElpsCylinder::print(ostream&) {  cout << name;  cout << "pos (" << setw(3) << pos\_x << ", " << setw(3) << pos\_y;  cout << "), angle (" << angle;  cout << "), color (" << setw(6) << setfill('0') << uppercase << hex << color;  cout << "), radius (" << radius\_1 << ", " << radius\_2;  cout << "), height (" << height << ")," << endl;  cout << " ellipse area (" << Elps::getArea();  cout << "), elp\_cyl area (" << getArea();  cout << "), elp\_cyl volume (" << getVolume() << ")";  cout << setfill(' ');  }  ElpsCylinder& ElpsCylinder::operator=(const ElpsCylinder& right) {  Elps::operator=(right);  height = right.height;  return \*this;  } |
|  |