Объектно-ориентированное программирование. Лабораторная работа №7. Простейшие примеры полноценного ООП

Макаров П. А.

25 ноября 2022 г.

Содержание

1 Краткая теория **1**

2 Задания для самостоятельной работы **1**

3 Список источников **7**

1 Краткая теория

Цель данной лабораторной работы — рассмотреть совместно на самых простых примерах все основные идеи объектно-ориентированной парадигмы программирования: абстракцию, инкапсуляцию, наследование и полиморфизм.

2 Задания для самостоятельной работы

1. Изучите все материалы и реализуйте на практике все примеры, рассмотренные нами в Лекциях №6 и №7.

2. Проработайте пример работы с графической сценой при использовании разных библиотек.

   (a) В отсутствие любых библиотек в принципе. В этом случае сценой будет считатся обычный текстовый файл, в который будут записываться координаты графических объектов. Строить изображение при этом можно даже в консоли при помощи, например, Gnuplot или ncurses.

(b) С использованием простейшей библитеки gfx, основанной на использовании X11 Window System и разработанной профессором Дугласом Тейном для поддержки своего учебного курса по основам программирования CSE 20211.

```
#include <cstdio>
#include "main.h"
#include "geometry.h"
extern "C" {
    #include "time.h"
    #include "stdlib.h"
}
#if VIDEOLIB == 1
extern "C" {
   #include "gfx.h"
}
#endif

FILE * fileScreen;

int main(int argc, char* argv[]) {
    srand(time(NULL));
    int N;
    if (argc == 1)
        N = rand()%51;
    else
        N = atoi(argv[1]);
    double r = (Width <= Height) ? Width/50. : Height
       /50.;
    #if VIDEOLIB == 0
        fileScreen = fopen("screen.txt", "a");
    #elif VIDEOLIB == 1
        gfx_open(Width, Height, "screen");
    #endif
    for(int i = 0; i < N; i++) {
        double x = r + (Width - 2*r)*rand()/RAND_MAX;
        double y = r + (Height - 2*r)*rand()/RAND_MAX;
        Circle P = Circle(x, y, r, rand()%0x1000000);
        P.Show();
    }
    #if VIDEOLIB == 0
        fclose(fileScreen);
    #elif VIDEOLIB == 1
        gfx_wait();
    #endif

    return 0;
}
```

Листинг 1: Текст файла `main.cpp`

```
extern FILE * fileScreen;
```

Листинг 2: Текст файла `main.h`

```
1  extern const int Width;
2  extern const int Height;
3
4  class GraphObject {
5  protected:
6      double x, y;
7      int color;
8  public:
9      GraphObject(double ax, double ay, int acolor)
10         : x(ax), y(ay), color(acolor) {}
11     virtual ~GraphObject() {}
12     virtual void Show() = 0;
13     virtual void Hide() = 0;
14     void Move(double nx, double ny);
15     double X() const;
16     double Y() const;
17 };
18
19 class Pixel : public GraphObject {
20 public:
21     Pixel(double x, double y, int color)
22         : GraphObject(x, y, color) {}
23     virtual ~Pixel() {}
24     void Show();
25     void Hide();
26 };
27
28 class Circle : public GraphObject {
29     double radius;
30 public:
31     Circle(double x, double y, double r, int color)
32         : GraphObject(x, y, color), radius(r) {}
33     virtual ~Circle() {}
34     void Show();
35     void Hide();
36 };
```

Листинг 3: Текст файла geometry.h

```
1  #include <cstdio>
2  #include "geometry.h"
3  #include "main.h"
4  extern "C" {
5     #include "math.h"
6  }
7  #if VIDEOLIB == 1
8  extern "C" {
9     #include "gfx.h"
10 }
11 #endif
12
13 const int Width  = 600;
14 const int Height = 600;
15
```

```
16 void GraphObject::Move(double nx, double ny) {
17     Hide();
18     x = nx;
19     y = ny;
20     Show();
21 }
22
23 double GraphObject::X() const {
24     return x;
25 }
26
27 double GraphObject::Y() const {
28     return y;
29 }
30
31 void Pixel::Show() {
32 #if VIDEOLIB == 0
33     fprintf(fileScreen, "%g\t%g\n", x, y);
34 #elif VIDEOLIB == 1
35     gfx_color((color&0xFF0000)>>16, (color&0x00FF00)
          >>8, color&0x0000FF);
36     gfx_point(x, y);
37     gfx_flush();
38 #endif
39 }
40
41 void Pixel::Hide() {
42 #if VIDEOLIB == 0
43     fprintf(fileScreen, "%g\t%g\n", x, y);
44 #elif VIDEOLIB == 1
45     gfx_color(0, 0, 0);
46     gfx_point(x, y);
47     gfx_flush();
48 #endif
49 }
50
51 void Circle::Show() {
52     int i, N = 75;
53     double phi = 0;
54 #if VIDEOLIB == 0
55     for(i = 0; i < N; i++) {
56         double xi = x + radius*cos(phi);
57         double yi = y + radius*sin(phi);
58         phi += 2*M_PI/(N-1);
59         fprintf(fileScreen, "%g\t%g\n", xi, yi);
60     }
61 #elif VIDEOLIB == 1
62     gfx_color((color&0xFF0000)>>16, (color&0x00FF00)
          >>8, color&0x0000FF);
63     for(i = 0; i < N; i++) {
64         double xi = x + radius*cos(phi);
65         double yi = y + radius*sin(phi);
66         phi += 2*M_PI/(N-1);
67         gfx_point(xi, yi);
```

```
68      }
69      gfx_flush();
70  #endif
71  }
72
73  void Circle::Hide() {
74      int i, N = 75;
75      double phi = 0;
76  #if VIDEOLIB == 0
77      for(i = 0; i < N; i++) {
78          double xi = x + radius*cos(phi);
79          double yi = y + radius*sin(phi);
80          phi += 2*M_PI/(N-1);
81          fprintf(fileScreen, "%g\t%g\n", xi, yi);
82      }
83  #elif VIDEOLIB == 1
84      gfx_color(0, 0, 0);
85      for(i = 0; i < N; i++) {
86          double xi = x + radius*cos(phi);
87          double yi = y + radius*sin(phi);
88          phi += 2*M_PI/(N-1);
89          gfx_point(xi, yi);
90      }
91      gfx_flush();
92  #endif
93  }
```

Листинг 4: Текст файла geometry.cpp

```
1  -include videolib.mk
2  ifneq ($(OLDVIDEOLIB),$(VIDEOLIB))
3  $(file > videolib.mk,OLDVIDEOLIB:=$(VIDEOLIB))
4  endif
5
6  ifneq ($(VIDEOLIB),)
7  CXXFLAGS=-DVIDEOLIB=$(VIDEOLIB)
8  endif
9
10 CC = gcc
11 CXX = g++
12 CFLAGS = -g -Wall
13 LDFLAGS = -lX11 -lm
14
15 objects = main.o geometry.o gfx.o
16 program = graph
17
18 default: $(program)
19
20 $(program): $(objects)
21     $(CXX) $(CFLAGS) $(objects) -o $(program) $(
          LDFLAGS)
22
23 main.o: main.cpp main.h
24     $(CXX) -c $(CXXFLAGS) $(CFLAGS) main.cpp -o main.o
```

```
25
26 geometry.o: geometry.cpp geometry.h
27     $(CXX) -c $(CXXFLAGS) $(CFLAGS) geometry.cpp -o
          geometry.o
28
29 gfx.o: gfx.c gfx.h
30     $(CC) -c $(CFLAGS) gfx.c -o gfx.o
31
32 clean:
33     rm $(objects) videolib.mk screen.txt
```

Листинг 5: Текст файла `Makefile`

Скачайте исходные тексты библиотеки gfx в папку с вашим проектом[1]. Для компиляции приложения в зависимости от использования той или иной графической библиотеки используйте команды:

```
make
make VIDEOLIB=0
make VIDEOLIB=1
```

Для отрисовки сцены с помощью Gnuplot используйте терминал `dumb` примерно следующим образом[2]

```
$ rm screen.txt
$ ./graph
$ gnuplot
gnuplot> set term dumb 105,35 aspect 1
gnuplot> plot "screen.txt" w d
```

3. До этого, в Лабораторной работе №3, мы работали с матрицами без привлечения идей ООП, в рамках структурной парадигмы (несмотря на использование объектов `cin` и `cout`). Теперь подойдём к этой задаче с точки зрения объектно-ориентированного программирования.

   Разработайте классы `Matrix` (Прямоугольная матрица) и `QMatrix` (Квадратная матрица), которые должны осуществлять стандартные операции матричного исчисления: сложение, вычитание, матричное умножение, умножение на число, транспонирование. Класс `QMatrix` также должен содержать методы вычисления определителя и получения обратной матрицы.

   Объект класса `Matrix` определяется размерностью матрицы и двумерным массивом её элементов. Поведенческие свойства

[1] В консоли для этого можно использовать команду `wget`

[2] В этом примере числа, напечатанные после имени терминала — это разрешение текстового экрана указанное в символах. Вы можете менять его так, как вам удобно.

класса определяются операциями матричного исчисления. Так как квадратная матрица — это частный случай прямоугольной матрицы, то структурные и поведенческие свойства класса `QMatrix` идентичны свойствам класса `Matrix`. В связи с этим, в рамках данной задачи реализуйте класс `QMatrix` публичным наследованием класса `Matrix`, добавив в него методы, специфичные для квадратной матрицы. Для обеспечения доступа к свойствам базового класса из производного, поместите их объявление в секцию `protected`.

Создание объекта класса `Matrix` требует выделения памяти для хранения его элементов. Поэтому класс `Matrix` обязательно должен содержать конструктор копирования, оператор присваивания и деструктор. Кроме того, дополнительно можно определить конструктор по умолчанию и конструктор с параметрами, определяющими размеры матрицы.

4. Спроектируйте иерархию классов «Вагоны пассажирского поезда» с разделением на общие, сидячие, плацкартные, купейные и СВ. Каждый класс вагона должен содержать информацию о количестве мест разных типов (нижнее, верхнее, нижнее боковое, верхнее боковое), о наличии дополнительных услуг. Реализуйте виртуальные методы, позволяющие рассчитать полный доход от эксплуатации вагона. Создайте класс «Пассажирский поезд», который должен хранить список вагонов. С помощью этой системы классов, напишите программу, определяющую доход от одного рейса поезда.

5. Решите задачи №3–4 на языке программирования Python.

3 Список источников

1. http://www.cplusplus.com.
2. Столяров А.В. Введение в язык Си++.
3. Андрианова А.А., Исмагилов Л.Н., Мухтарова Т.М. Объектно-ориентированное програмирование на C++.
4. Gnuplot.
5. ncurses.
6. gfx.
7. X11 Window System.
8. CSE 20211.

9. make.

10. https://www.python.org/doc/.