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

Макаров П. А.

Содержание

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 следующий простейший пример работы с графической сценой с применением библитеки `gfx`, основанной на использовании X11 Window System и разработанной профессором Дугласом Тейном для поддержки своего учебного курса по основам программирования CSE 20211. Данный пример в несколько урезанном виде работоспособен даже в отсутствии графической подсистемы. При этом под сценой можно подразумевать обычный текстовый файл, в котором в текстовом виде хранятся координаты созданных и перемещённых графических объектов. При

этом изображение в консоли можно построить при помощи, например, Gnuplot или ncurses.

```
#include <cstdio>
#include "main.h"
#include "geometry.h"
extern "C" {
    #include "time.h"
    #include "stdlib.h"
    #include "gfx.h"
}

FILE * fileScreen;

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

Листинг 1: Текст файла `main.cpp`

```
extern FILE * fileScreen;
```

Листинг 2: Текст файла `main.h`

```
extern const int Width;
extern const int Height;

class GraphObject {
protected:
    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 Save() = 0;
13    virtual void Show() = 0;
14    virtual void Hide() = 0;
15    void Move(double nx, double ny);
16    double X() const;
17    double Y() const;
18 };
19
20 class Pixel : public GraphObject {
21 public:
22    Pixel(double x = 0, double y = 0, int color = 0)
23        : GraphObject(x, y, color) {}
24    virtual ~Pixel() {}
25    void Set(double nx = 0, double ny = 0, int ncolor
        = 0);
26    void Save();
27    void Show();
28    void Hide();
29 };
30
31 class Circle : public GraphObject {
32    double radius;
33 public:
34    Circle(double x = 0, double y = 0, double r = 0,
        int color = 0)
35        : GraphObject(x, y, color), radius(r) {}
36    virtual ~Circle() {}
37    void Set(double nx = 0, double ny = 0, double nr =
        0, int ncolor = 0);
38    void Save();
39    void Show();
40    void Hide();
41 };
```

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     #include "gfx.h"
7 }
8
9 const int Width  = 600;
10 const int Height = 600;
11
12 void GraphObject::Move(double nx, double ny) {
13     Hide();
```

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

```
66
67 void Circle::Show() {
68     int i, N = 75;
69     double phi = 0;
70     gfx_color((color&0xFF0000)>>16, (color&0x00FF00)
          >>8, color&0x0000FF);
71     for(i = 0; i < N; i++) {
72         double xi = x + radius*cos(phi);
73         double yi = y + radius*sin(phi);
74         phi += 2*M_PI/(N-1);
75          gfx_point(xi, yi);
76     }
77     gfx_flush();
78 }
79
80 void Circle::Hide() {
81     int i, N = 75;
82     double phi = 0;
83     gfx_color(0, 0, 0);
84     for(i = 0; i < N; i++) {
85         double xi = x + radius*cos(phi);
86         double yi = y + radius*sin(phi);
87         phi += 2*M_PI/(N-1);
88         gfx_point(xi, yi);
89     }
90     gfx_flush();
91 }
```

Листинг 4: Текст файла geometry.cpp

```
1 CC = gcc
2 CXX = g++
3 CFLAGS = -g -Wall
4 LDFLAGS = -lX11 -lm
5
6 objects = main.o geometry.o gfx.o
7 program = graph
8
9 default: $(program)
10
11 $(program): $(objects)
12     $(CXX) $(CFLAGS) $(objects) -o $(program) $(
          LDFLAGS)
13
14 main.o: main.cpp main.h
15     $(CXX) -c $(CFLAGS) main.cpp -o main.o
16
17 geometry.o: geometry.cpp geometry.h
18     $(CXX) -c $(CFLAGS) geometry.cpp -o geometry.o
19
20 gfx.o: gfx.c gfx.h
21     $(CC) -c $(CFLAGS) gfx.c -o gfx.o
22
23 clean:
```

```
24      rm $(objects)
```

Листинг 5: Текст файла `Makefile`

Замечание 1. *При наборе файла, текст которого приведён в Листинге 5, не забудьте*[1] *про символ табуляции!*

Скачайте исходные тексты библиотеки `gfx` в папку с вашим проектом[2]. Для компиляции приложения используйте команду:

```
$ make
```

Для отрисовки сцены с помощью Gnuplot в консоли используйте терминал `dumb` примерно следующим образом[3]

```
$ ./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 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`.

[1] В текстовом редакторе `vim` может оказаться полезной настройка `set noet`.

[2] В консоли для этого можно использовать команду `wget`.

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

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/.