ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ

Государственное образовательное учреждение
высшего профессионального образования

“МАТИ” — РОССИЙСКИЙ ГОСУДАРСТВЕННЫЙ
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. К. Э. ЦИОЛКОВСКОГО

Кафедра “Моделирование систем и информационные технологии”

ИСПОЛЬЗОВАНИЕ СРЕДСТВ НАСЛЕДОВАНИЯ В ПАСКАЛЕ ПРИ СОЗДАНИИ ГРАФИЧЕСКИХ ИНТЕРФЕЙСОВ, ИЗОБРАЖЕНИЕ ГРАФИКОВ ФУНКЦИЙ

Методические указания к лабораторной работе по курсу
“Алгоритмические языки и программирование”

Составитель В. В. Лидовский

Москва 2006

ВВЕДЕНИЕ

Настоящие методические указания предназначены для обеспечения учебного процесса студентов второго курса дневной формы обучения специальности 220200 “АСОИиУ” при выполнении лабораторной работы по предмету “Алгоритмические языки и программирование”.

Цель лабораторной работы: изучить методы программирования на языке Паскаль с использованием средств *наследования* концепции ООП на примере реализации простого графического интерфейса с построением графиков заданных функций.

1. СИСТЕМНЫЕ ТРЕБОВАНИЯ

Для выполнения лабораторной работы необходимы следующие системные компоненты:

а) компьютер, работающий под управлением операционной системы Linux;
б) компилятор Free Pascal версии не ранее 2000 года;
в) пакет поддержки средств Tcl/Tk;
г) графический сервер `sgserv` версии 1.11 или новее.

2. ГРАФИЧЕСКИЙ ИНТЕРФЕЙС И ООП

Как показала практика разработки и использования графических интерфейсов, ООП является идеальным фундаментом для подобных разработок.

Схема графического интерфейса при использовании ООП — это только иерархия графических объектов. На вершине такой иерархии, как правило, помещается абстрактный объект, являющийся базовым (родовым) по отношению ко всем остальным объектам иерархии. Сам этот объект в силу своей общности не соответствует никакому графическому изображению. Помимо удобства в представлении иерархии, этот абстрактный объект позволяет иметь *необходимые* переменные-указатели, которые можно использовать для указания на *любой* объект иерархии. Производным (видовым) объектам иерархии соответствуют изображения: точка, круг, прямая, текст, ...

Все видимые на экране графические объекты являются компонентами однонаправленного списка. При необходимости показать реакцию объектов на то или иное событие производится обход списка с начала. При таком обходе *каждому* объекту передается сообщение о событии, а каждый объект проверяет, относится ли это сообщение к нему и, если относится, то реагирует на него известным ему способом. Сообщения могут генерироваться извне средствами интерфейса (мышкой, клавиатурой, ...) или самими объектами.

3. ГРАФИЧЕСКИЙ СЕРВЕР

Программа графического сервера `sgserv` написана на языке Tcl с использованием обращений к графической библиотеке Tk. Этот язык — интерпретирующийся, поэтому `sgserv` — это не закрытый для анализа бинарный файл, а текст самодокументированной программы.

Назначение этого простейшего сервера — в трансляции текстовых команд в действия стандартного для среды Linux графического сервера X Window или X Free. Он позволяет простейшим образом создать несложные графические изображения, в том числе анимированные, средствами *любых* языков программирования.

Программа `sgserv` распознает всего 14 команд, некоторые из которых при выполнении данной лабораторной работы не необходимы (полный список команд см. в тексте `sgserv`).

Рассмотрим некоторые команды (большие и маленькие буквы *различаются*).

1) **area** XSIZE YSIZE COLOR — эта команда создает графическое полотно (окно) на экране дисплея с шириной XSIZE точек, с высотой YSIZE точек и цвета COLOR. Цвет задается английским названием, например, `white`, `yellow`, `cyan`, `gold`, ...
2) **circle** NAME X Y R COLOR — эта команда рисует круг на созданном ранее графическом полотне с центром в точке X, Y и радиусом R. Цвет круга устанавливает параметр COLOR так же как и для предыдущей команды. Кроме того, для возможности дальнейших ссылок на нарисованный объект ему сопоставляется *идентификатор* NAME.
3) **move** NAME DX DY — эта команда сдвигает графический объект, в частности, круг, с именем NAME на DX точек по горизонтали и DY точек по вертикали.
4) **delete** NAME — эта команда уничтожает изображение графического объекта с именем NAME.
5) Команда **пустая строка** означает, что работа с графическим сервером должна быть закончена. Ее выполнение заключается в уничтожении холста для рисования.
6) **pause** — эта команда приостанавливает работу графического сервера до тех пор, пока не будет нажата левая кнопка мыши.
7) **setAxes** DX XBEGIN XEND DY YBEGIN YEND COLOR — эта команда *подготавливает* изображение декартовой системы координат на плоскости. Первые три параметра устанавливают ось абсцисс, а три последующих — ось ординат. Первый параметр в тройке — это шаг между делениями на шкале выбранной оси, второй параметр — это начальная отметка, а третий — конечная. Последний параметр — это цвет обоих осей. Знак XBEGIN и YBEGIN должен быть отрицательным, а знак XEND и YEND — положительным

8) **showAxes** — показ *подготовленных* предыдущей командой осей координат. Используется *после* изображения всех функций.
9) **showFunc** XBEGIN XEND FX LEGEND COLOR — эта команда изображает график функции FX между абсциссами XBEGIN и XEND цветом COLOR. Следует иметь в виду, что сервер требует, чтобы аргумент x *всегда* был в скобках, т. е. вместо $x+7$ нужно писать $(x)+7$. Параметр LEGEND — это легенда к графику функции.

С графическим сервером можно работать автономно, с командной строки. Например, наберите `sgserv` в терминале, вызванном в графической среде. Нажмите клавишу Enter. Теперь можно вводить команды серверу. Введите следующие строки (ENTER — это нажатие клавиши Enter на клавиатуре).

```
area 320 200 red      ENTER
circle c1 100 100 blue 5      ENTER
move c1 20 20      ENTER
ENTER
```

Сначала должно появиться изображение красного холста, на котором затем возникнет синий круг, который затем сместится к правому нижнему углу холста.

Программа на Паскале или любом другом языке программирования для работы с сервером `sgserv` должна лишь печатать обычным образом текстовые команды серверу. Например, если программа `grtest` печатает сначала строки

```
area 600 400 darkgreen
circle c1 200 300 50 red ,
```

а затем в цикле строки

```
move c1 40 -20
move c1 -40 20 ,
```

то будет получено изображение постоянно перемещающегося между двумя точками круга. Совместный запуск `grtest` и `sgserv` производится командой

```
grtest | sgserv .
```

Нужно иметь в виду, что *все* сообщения, напечатанные программой в стандартный поток вывода (`output`), будут перехватываться графическим сервером. Поэтому для печати подсказок пользователю и т. п. нужно использовать поток сообщений об ошибках ошибок (`stderr`). Например, оператор Паскаля `writeln(stderr, 'Hello, user!')` напечатает сообщение на экран дисплея, которое не может быть перехвачено.

При задании функции нужно использовать принятые в англоязычных странах сокращения, а также в некоторых случаях выражать одни функции через другие (см. следующую таблицу).

квадратный корень	$\sqrt{x}$	sqrt(x)
экспонента	e^x	exp(x)
натуральный логарифм	$\ln x$	log(x)
синус	$\sin x$	sin(x)
секанс	$\sec x$	1/cos(x)
косеканс	$\mathrm{cosec}\, x$	1/sin(x)
тангенс	$\mathrm{tg}\, x$	tan(x)
арктангенс	$\mathrm{arctg}\, x$	atan(x)
арксинус	$\arcsin x$	asin(x)
арккосинус	$\arccos x$	acos(x)
арккотангенс	$\mathrm{arcctg}\, x$	$\pi/2$-atan(x)
гиперболический синус	$\mathrm{sh}\, x$	sinh(x)
гиперболический косинус	$\mathrm{ch}\, x$	cosh(x)
гиперболический тангенс	$\mathrm{th}\, x$	tanh(x)
модуль	$\lvert x \rvert$	abs(x)

Вместо величины $\pi/2$ нужно брать ее приближенное числовое значение.

Каждый графический объект должен иметь свое уникальное имя. Для получения такого имени можно воспользоваться тем, что каждая переменная имеет свой уникальный адрес в памяти компьютера, который доступен внутри каждого метода объектной переменной через неявный параметр `self`. Оси координат в силу их единственности в имени не нуждаются.

4. ПОРЯДОК ВЫПОЛНЕНИЯ ЛАБОРАТОРНОЙ РАБОТЫ

Программа должна содержать иерархию из трех графических объектных типов: базовый абстрактный тип и производные от него тип-круг и тип оси координат.

В программе должен быть также объектный тип холст, который должен содержать указатель на базовый абстрактный тип, который можно будет использовать как указатель на начало однонаправленного списка графических объектов, изображаемых на холсте. Для организации такого однонаправленного списка необходимо, чтобы базовый абстрактный тип включал в себя указатель на следующий объект базового абстрактного типа.

Для представления функции, график которой нужно отобразить, в программе должен быть соответствующий объектный тип. Этот тип должен включать в себя указатель на следующий объект-функцию, что позволяет организовать из объектов такого типа однонаправленный список. Объектный тип оси координат должен содержать указатель на объект-функцию — его можно использовать как указатель на первый элемент

такого списка.

Для организации ввода данных в программе должны быть реализованы три подпрограммы ввода текста: целого числа, вещественного числа, строки.

Каждый объектный тип должен содержать конструктор или деструктор и хотя бы одну виртуальную функцию. Все изображаемые объекты должны иметь метод для их показа.

Входные данные должны задаваться тремя строками текста. Первая строка должна содержать параметры холста (размеры, цвет), вторая — параметры осей координат, третья — список функций (в нем могут быть 1, 2, 3, ... функций). Например,

```
700 400 green
1 -10 5 2 -7 12 yellow
-9.42 4 (x)*sin(x) x*sin_x brown -9 4 0.5*abs(x) 0.5|x| red .
```

Если поместить эти строки в файл `grdata` и имя скомпилированной программы — `grshow`, то запуск удобнее всего производить командной строкой

```
grshow < grdata | sgserv .
```

Программа должна отобразить графики заданных функций и фигуру круга, после чего закончить работу по нажатию на левую кнопку мышки.

5. КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Почему указатель на следующий графический объект в объекте-круге должен быть типа указатель на базовый абстрактный объект? Почему нельзя использовать указатель типа указатель на объект-круг?
2. Какой метод программы наиболее естественно сделать виртуальным?
3. Производным от какого объектного типа следовало бы сделать тип-кольцо? Как добавить в иерархию графических объектов такой тип?
4. Как добавить в список графических объектов еще один круг?
5. Почему отказ от использования наследования значительно усложнил бы программирование графических интерфейсов?
6. Необходим ли программе механизм полиморфизма концепции ООП?
7. Какие недостатки в реализации программы простого графического сервера `sgserv` вы обнаружили?
8. Почему базовый абстрактный тип не нуждается в конструкторе? Чем полезно введение виртуального деструктора в абстрактный тип?
9. Сколько раз будет вызван деструктор круга при работе программы?
10. Нужно ли при реализации объектной иерархии программы использовать механизм инкапсуляции ООП?
11. Обнаружили ли вы какие-нибудь недостатки в реализации концепций ООП на Паскале?

6. ВАРИАНТЫ РАБОТ

Лабораторная работа по теме “Использование средств наследования в Паскале при создании графических интерфейсов, изображение графиков функций” имеет 27 вариантов заданий, определяемых следующей таблицей.

№	$f_1(x)$	$f_2(x)$	x	y
1	$\sin x$	$\operatorname{sh} x$	$-\pi \dots \pi$	$-9 \dots 9$
2	$\sin x$	$\operatorname{ch} x$	$-\pi \dots \pi$	$-9 \dots 9$
3	$\cos x$	$\operatorname{sh} x$	$-\pi \dots \pi$	$-9 \dots 9$
4	$\cos x$	$\operatorname{ch} x$	$-\pi \dots \pi$	$-9 \dots 9$
5	$\sin x$	$\cos x$	$-2\pi \dots 2\pi$	$-1 \dots 1$
6	$\sin x + \cos x$	$\sin x$	$-2\pi \dots 2\pi$	$-\sqrt{2} \dots \sqrt{2}$
7	$\sin x + \cos x$	$\cos x$	$-2\pi \dots 2\pi$	$-\sqrt{2} \dots \sqrt{2}$
8	$\sin x + \cos x$	$\operatorname{th} x$	$-2\pi \dots 2\pi$	$-\sqrt{2} \dots \sqrt{2}$
9	$\sin x$	$\operatorname{th} x$	$-2\pi \dots 2\pi$	$-1 \dots 1$
10	$\cos x$	$\operatorname{th} x$	$-2\pi \dots 2\pi$	$-1 \dots 1$
11	$\operatorname{sh} x$	$\operatorname{ch} x$	$-\pi \dots \pi$	$-9 \dots 9$
12	$\operatorname{sh} x$	$\operatorname{th} x$	$-\pi \dots \pi$	$-9 \dots 9$
13	$\operatorname{ch} x$	$\operatorname{th} x$	$-\pi \dots \pi$	$-9 \dots 9$
14	$\operatorname{tg} x$	$\operatorname{th} x$	$-\pi/2 \dots \pi/2$	$-7 \dots 7$
15	$\operatorname{tg} x$	$\sec x$	$-\pi/2 \dots \pi/2$	$-7 \dots 7$
16	$\operatorname{th} x$	$\sec x$	$-\pi/2 \dots \pi/2$	$-5 \dots 5$
17	$\operatorname{arctg} x$	$\operatorname{tg} x$	$-\pi/2 \dots \pi/2$	$-\pi/2 \dots \pi/2$
18	$1 + x^2$	$\operatorname{ch} x$	$-\pi \dots \pi$	$-7 \dots 7$
19	$1 + x^2$	$\sec x$	$-\pi/2 \dots \pi/2$	$-4 \dots 4$
20	e^x	$\operatorname{sh} x$	$-\pi \dots \pi$	$-9 \dots 9$
21	e^x	$\operatorname{ch} x$	$-\pi \dots \pi$	$-9 \dots 9$
22	e^x	$\operatorname{th} x$	$-\pi \dots \pi$	$-7 \dots 7$
23	e^x	$\sin x$	$-\pi \dots \pi$	$-7 \dots 7$
24	$\operatorname{cosec} x$	$\lvert x \rvert$	$-2\pi \dots 2\pi$	$-7 \dots 7$
25	$-\sqrt{9 - x^2}$	$\sqrt{9 - x^2}$	$-3 \dots 3$	$-3 \dots 3$
26	$\operatorname{arcctg} x$	$\operatorname{tg} x$	$-\pi/2 \dots \pi/2$	$-\pi \dots \pi$
27	e^x	$\cos x$	$-\pi \dots \pi$	$-7 \dots 7$

В таблице для каждого варианта задаются две функция $f_1(x)$ и $f_2(x)$, их область определения и диапазон показываемых значений.

ЛИТЕРАТУРА

1. Бронштейн И.Н., Семендяев К.А. *Справочник по математике* — М.: Наука, 1986. — 544 с.
2. Зуев Е.А. *Язык программирования Turbo Pascal 6.0* — М.: Унитех, 1992. — 298 с.
3. Петерсен Р. *Linux: полное руководство* — Киев: “Ирина” BHV, 2000. — 642 с.

ПРИЛОЖЕНИЕ 1

ПРИМЕР ПРОГРАММЫ, РЕШАЮЩЕЙ ПОСТАВЛЕННУЮ ЗАДАЧУ

```
function GetString(var s: string): string;
  var p: word;
  begin
    p := pos(' ', s);
    if p > 0 then begin
      GetString := copy(s, 1, p - 1);
      delete(s, 1, p);
      while s[1] = ' ' do
        delete(s, 1, 1);
      exit
    end;
    GetString := s;
    s := ''
  end;
function GetInteger(var s: string): integer;
  var
    t: string;
    i, n: integer;
  begin
    t := GetString(s);
    val(t, i, n);
    if n <> 0 then
      halt(1);
    GetInteger := i
  end;
function GetReal(var s: string): real;
  var
    t: string;
    r: real;
    n: integer;
  begin
    t := GetString(s);
    val(t, r, n);
    if n <> 0 then
      halt(1);
    GetReal := r
  end;
Type
  PTGobj = ^TGObj;
  TGObj = object
    next: PTGobj;
    destructor Done; virtual;
    procedure Show; virtual;
  end;
procedure TGObj.Show;
```

```
    begin end;
destructor TGObj.Done;
    begin end;
Type
  TCircle = object (TGObj)
    x, y, r: integer;
    color: string;
    constructor Init(x0, y0, r0: integer; color0: string);
    procedure Show; virtual;
    destructor Done; virtual;
  end;
constructor TCircle.Init(x0, y0, r0: integer; color0: string);
    begin
      x := x0;
      y := y0;
      r := r0;
      color := color0
    end;
procedure TCircle.Show;
    begin
      writeln('circle c', abs(longint(@self)), x:5, y:5, r:5, color:12)
    end;
destructor TCircle.Done;
    begin
      writeln('delete c', abs(longint(@self)))
    end;
Type
  PTGFunc = ^TGFunc;
  TGFunc = object
    xfstart, xfend: real;
    spec, legend, color: string;
    next: PTGFunc;
    constructor Init(xfstart0, xfend0: real; spec0, legend0, color0: string);
    procedure Show;
  end;
constructor TGFunc.Init(xfstart0, xfend0: real; spec0, legend0, color0: string);
    begin
      xfstart := xfstart0;
      xfend := xfend0;
      spec := spec0;
      legend := legend0;
      color := color0
    end;
procedure TGFunc.Show;
    begin
      writeln('showFunc ', xfstart, ' ', xfend, ' ', spec, ' ', legend,
        color:12)
    end;
```

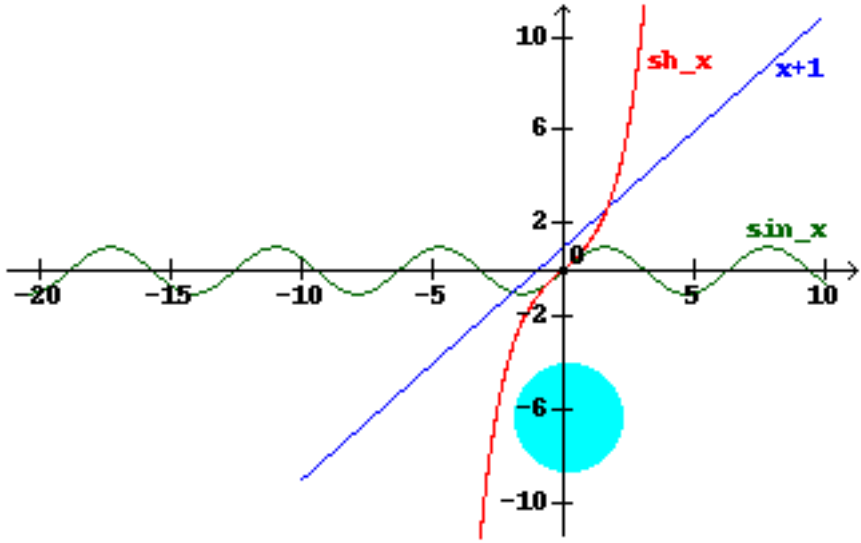
```
Type
  TAxes = object (TGObj)
    dx, xstart, xend, dy, ystart, yend: integer;
    color: string;
    pGFunc1: PTGFunc;
    constructor Init(dx0, xstart0, xend0, dy0, ystart0, yend0: integer;
      color0, funcSpec: string);
    procedure Show; virtual;
    destructor Done; virtual;
  end;
constructor TAxes.Init(dx0, xstart0, xend0, dy0, ystart0, yend0: integer;
      color0, funcSpec: string);    (* параметры осей - целые ! *)
    var
      t1, t2: PTGFunc;
      r1, r2: real;
      s1, s2: string[31];
    begin
      dx := dx0;
      xstart := xstart0;
      xend := xend0;
      dy := dy0;
      ystart := ystart0;
      yend := yend0;
      color := color0;
      pGFunc1 := nil;
      while funcSpec <> '' do begin
        r1 := GetReal(funcSpec);
        r2 := GetReal(funcSpec);
        s1 := GetString(funcSpec);
        s2 := GetString(funcSpec);
        new(t1, Init(r1, r2, s1, s2, GetString(funcSpec)));
        if pGFunc1 = nil then begin
          pGFunc1 := t1;
          pGFunc1^.next := nil;
          t2 := t1
        end else begin
          t2^.next := t1;
          t2 := t1;
          t2^.next := nil
        end
      end
    end;
procedure TAxes.Show;
    var t: PTGFunc;
    begin
      if (dx <= 0) or (dy <= 0) then
        halt(1);
      writeln('setAxes ', dx, xstart:5, xend:5, dy:5, ystart:5, yend:5,
        color:12);
      t := pGFunc1;
```

```
    while t <> nil do begin
      t^.Show;
      t := t^.next
    end;
    writeln('showAxes')
  end;
destructor TAxes.Done;
  var
    t1, t2: PTGFunc;
  begin
    t1 := pGFunc1;
    while t1 <> nil do begin
      t2 := t1^.next;
      dispose(t1);
      t1 := t2
    end
  end;
Type
  TGArea = object
    xsize, ysize: word;
    color: string;
    pGObj1: PTGobj;
    constructor Init(xsize0, ysize0: word; color0, axesSpec, funcSpec: string);
    procedure Show;
    destructor Done;
  end;
constructor TGArea.Init(xsize0, ysize0: word; color0, axesSpec,
    funcSpec: string);    (* размеры полотна - целые ! *)
  var
    pCircle: ^TCircle;
    pAxes: ^TAxes;
    i: integer;
    p: array[1..6]of integer;
  begin
    if (xsize0 > 1024) or (ysize0 > 768) then
      halt(1);
    xsize := xsize0;
    ysize := ysize0;
    color := color0;
    new(pCircle, Init(70, 70, 20, 'cyan'));
    pGObj1 := pCircle;
    for i := 1 to 6 do
      p[i] := GetInteger(axesSpec);
    new(pAxes, Init(p[1], p[2], p[3], p[4], p[5], p[6],
      GetString(axesSpec), funcSpec));
    pGObj1^.next := pAxes;
    pGObj1^.next^.next := nil
  end;
procedure TGArea.Show;
  var
```

```
    t: PTGobj;
  begin
    writeln('area ', xsize, ysize:5, color:12);
    t := pGObj1;
    while t <> nil do begin
      t^.Show;
      t := t^.next
    end
  end;
destructor TGArea.Done;
  var
    t1, t2: PTGobj;
  begin
    t1 := pGObj1;
    while t1 <> nil do begin
      t2 := t1^.next;
      dispose(t1, Done);
      t1 := t2;
    end;
    writeln    (*выход из графики*)
  end;
Var
  pGArea: ^TGArea;
  i1, i2: integer;
  s1, s2, s3: string;
Begin
  writeln(stderr, 'Введите параметры графической области: ширину, ',
    'высоту, цвет: ');
  readln(s1);
  writeln(stderr);
  writeln(stderr, 'Введите параметры системы координат (через пробел): ',
    'шаг шкалы по оси x,'#10,
    '  начальную и конечную абсциссы, шаг шкалы по оси y,'#10,
    '  начальную и конечную ординаты, цвет: ');
  readln(s2);
  writeln(stderr);
  writeln(stderr, 'Введите список из элементов вида'#10'  ',
    '<начальная> <конечная абсцисса графика> <функция> <легенда> <цвет>,'#10,
    '  например, -10 20 sin(x) sin_x red -8 20 4-pow((x),3) 4-x^3 blue '#10,
    '  (x должен заключаться в скобки!): ');
  readln(s3);
  i1 := GetInteger(s1);
  i2 := GetInteger(s1);
  new(pGArea, Init(i1, i2, GetString(s1), s2, s3));
  pGArea^.Show;
  writeln('pause');
  dispose(pGArea, Done)
End.
```

ПРИЛОЖЕНИЕ 2

ПРИМЕР РЕЗУЛЬТАТА РАБОТЫ ПРОГРАММЫ



ОГЛАВЛЕНИЕ

Владимир Викторович Лидовский

ИСПОЛЬЗОВАНИЕ СРЕДСТВ НАСЛЕДОВАНИЯ В ПАСКАЛЕ ПРИ СОЗДАНИИ ГРАФИЧЕСКИХ ИНТЕРФЕЙСОВ, ИЗОБРАЖЕНИЕ ГРАФИКОВ ФУНКЦИЙ

Методические указания к лабораторной работе по курсу
“Алгоритмические языки и программирование”

Редактор А. Н. Прохорова

Оригинал-макет подготовлен в пакете Plain-TEX

Под. в печ. 6.10.2006 Объем 1 п.л. Тираж 50 экз. Зак. 164

Издательский центр МАТИ, 109240, Москва, Берниковская наб., 14.