№ 2 (195) январь 1999

еженедельное
ПРИЛОЖЕНИЕ К ГАЗЕТЕ
«ПЕРВОЕ СЕНТЯБРЯ»

ИНФОРМАТИК

Электронная версия газеты Информатик http://www.glasnet.ru/~infosef

Интернет для начинающих

А.А. ДУВАНОВ

Продолжение. См. № 36, 40, 44, 46/98

1.5. Мой адрес не дом и не улица
(работа с адресной книгой)

Вася. Я уже проверил почту: пришло два одинаковых письма от Паука. Для тренировки я сначала получил письма в Windows, а потом при помощи программы UUCP.

Папа. Что же написал Паук?

Вася. Вот его ответ:

```
                    Уважаемый корреспондент!
  Ваше письмо получено роботландским почтовым роботом.
  Письмо получено от: Vasil Kuk vasil@kuk.botik.ru
  Вы отправили письмо: 24.06.98 12:10
  Ваша почтовая программа: dMail [Demos Mail for DOS v2.07a]
  Тема Вашего письма: Hallo

  Мне нужна дополнительная информация! Когда я получу ее, то пошлю Вам сочиненный мною
рассказ.
   Пожалуйста, пошлите мне письмо, указав в поле "Тема" слово "Hallo". Текст письма
должен состоять из следующих строчек:

  ЭТО Привет
  Маша         // Ваше имя
  ж            // Ваш пол ("м" или "ж")
  Буратино          // Кто Вам нравится
  м            // Род ("м", "ж", "с")
  веселый           // Какой он
  Баба Яга          // Кто Вам не нравится
  ужасная           // Какой он
  КОНЕЦ

  Ваше письмо — это программа для меня, поэтому оно должно точно соответствовать приве-
денному выше описанию.
  Текст письма должен открываться строчкой  "ЭТО Привет", а закрываться строчкой "КОНЕЦ". В
остальных строчках указывается информация, описанная после знаков комментария "//". Сами
комментарии писать не обязательно, но можно их и оставить. Конечно, вместо слова "Маша"
Вы должны написать свое имя и другие строчки тоже заполнить по-своему. Я люблю сочинять
забавные истории и с радостью сделаю это для Вас!

  Жду Вашу программу!
  \#/
  =O=  Паучок (почтовый робот Роботландии)
  / \
```

Продолжение на с. 2

КУДА ПОЙТИ СО ШКОЛЬНОГО ДВОРА

Эту новую рубрику мы открыли в № 48/98 статьей А.И. Сенокосова (г. Екатеринбург) "Профессия, или Проблема высшего образования для ваших лучших учеников".

Чем руководствуются мои выпускники при выборе вуза?

С.М. ОКУЛОВ

Автор — заведующий кафедрой информатики, доцент Вятского педагогического университета, учитель информатики физико-математического лицея г. Вятки.
Хорошо известен нашим читателям как один из ведущих авторов рубрик "Олимпиады" и "Языки программирования". Антон Лапунов, Виктор Матюхин, Виталий Беров — победители Международных олимпиад по информатике 1993, 1994, 1995, 1996 годов, в настоящее время студенты МГУ — его воспитанники.

Есть ряд соображений, влияющих на их решение.

1. Оценка вуза в мире. Традиционно у вятских школьников котируется МГУ. Они считают, что диплом этого университета оценивается в мире достаточно высоко.
2. Оснащенность. Оценка оснащенности вуза средствами вычислительной техники; система доступа; возможность работы в Интернет. Если ребятам известно, что с первого курса у них есть свободный доступ (например, в СПбГИТМО(ТУ)), то это является весомым аргументом.
3. Содержание обучения. Количество и виды (типы) дисциплин по информатике. Если нет современных технологий или они ограничиваются "стандартным букетом", то сие, безусловно, является минусом.
4. Преподаватели. Идут на личности в информатике. Цель — через сотрудничество с ними стать профессионалами в информатике.
5. Второе образование. Возможность бесплатно получить в вузе второе высшее образование.

Как ребята получают информацию по пп. 2—5? В основном от своих сверстников. Во всяком случае, я рекомендую им получить эту информацию и только после этого принимать самостоятельно решение.

НАШИ ДЕТИ БУДУТ ЖИТЬ В XXI ВЕКЕ

СПЕЦВЫПУСК

ЗАДАЧИ ПО ТЕМЕ "КОМПЬЮТЕРНАЯ ГРАФИКА"
Преобразования на плоскости

С.Л. ОСТРОВСКИЙ, Ю.А. СОКОЛИНСКИЙ

Можно ли изобразить пространственный объект на плоскости, следуя только строгим, однозначным правилам (выполняя некоторый алгоритм) без всякой апелляции к художественной интуиции? Этот вопрос интересовал еще итальянских художников эпохи Возрождения. Тогда же сложились основы "учения о перспективе", определяющего методы решения данной задачи.

В выпуске кратко описывается общий подход к проблеме и разбираются соответствующие задания. Используются три языка программирования: Паскаль, Си и Бейсик.

Выпуски 1, 2, 3 были опубликованы в № 30, 31, 47/98.

Интернет для начинающих

Продолжение. Начало на с. 1

Папа. Я вижу, что с Пауком завязывается активная переписка! Надо занести его адрес в адресные книги наших почтовых программ.

Вася. А я записал адрес Паука ручкой на листочке бумаги.

Папа. В этом нет необходимости, когда под рукой компьютер, а на дворе бурный прогресс компьютерной информатики.

Вася. Согласен! Я это сделал по старой привычке. Мне, конечно, немного стыдно, но когда я вижу запись на бумаге, то больше уверен в ее сохранности.

Папа. Листочек можно легко потерять, а важные файлы можно дублировать в резервном архиве.

Вася. Звучит убедительно! Итак, почтовые программы имеют специальные хранилища — адресные книги.

Папа. Удобство их еще в том, что при посылке письма не надо вводить адрес корреспондента "руками": достаточно одного щелчка в нужной строке книги (или нажатия клавиши) — и адрес запишется на конверт автоматически.

Вася. Двойная выгода: во-первых, экономится время, во-вторых, исключаются ошибки. Мне по душе такой аккуратный помощник!

Папа. Адресная книга позволяет автоматизировать не только извлечение данных, но и их добавление. Можно, конечно, записать новый адрес в книгу вручную, но лучше, если он попадет в хранилище автоматически, прямо из пришедшего письма. При этом опять же исключаются ошибки ручного ввода.

Вася. Эта услуга почтовой программы прямо создана для меня — ты ведь знаешь, как я не люблю скучной работы.

Папа. Да уж, какой ты лентяй, мне известно! Вероятно, и адресную книгу придумал такой же, как ты, специалист по лени.

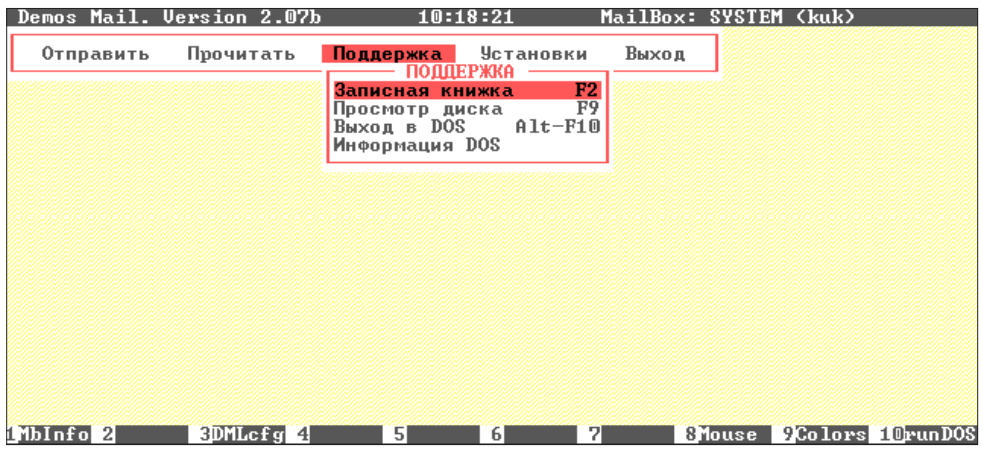
Вася. Мне кажется, что вообще все изобретения так и случаются. Лень — это мощный двигатель прогресса!

Папа. А ты еще и философ! Думаю, ты не прав. Изобретения происходят от любознательности, а сказка про лень — это просто смешная выдумка.

Хорошо, давай посмотрим, как работать с адресными книгами в разных почтовых программах. Начнем с программы DMAIL.

Вася. Вот я запустил эту программу и вижу на экране главное меню:

Папа. Вызвать адресную книгу можно двумя способами. Первый — через строку **Записная книжка** позиции **Поддержка** в главном меню:



Второй — более короткий — клавишей F2. Напоминание об этой клавише расположено в нижней строке экрана.

Вася. Там написано **2AdrLst**.

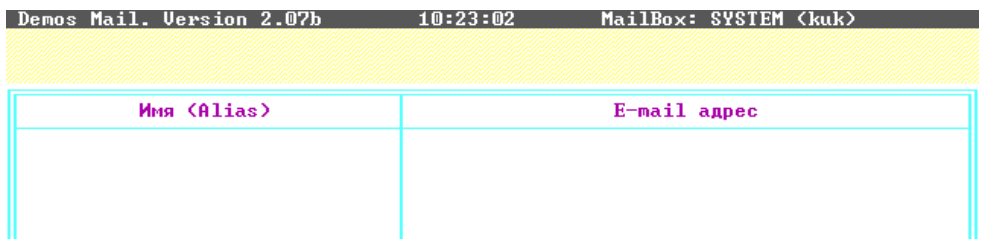
Папа. Это означает, что команда **AdrLst** выполняется нажатием клавиши F2. `AdrLst` — это сокращение от английской фразы *address list* — адресный список.

Вася. Нажимаю F2.

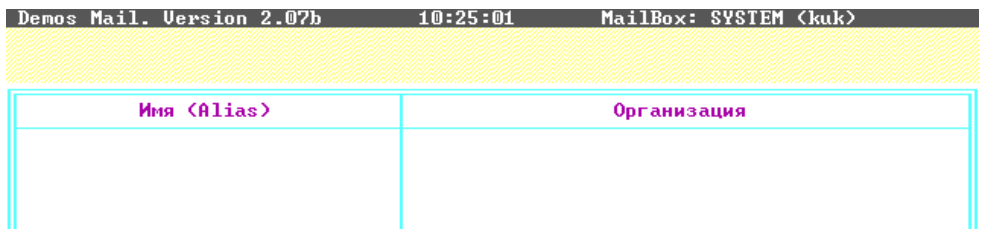
Папа. Список адресов в книге представлен табличкой из нескольких столбцов.

Вася. Я вижу только два столбца: один — для имен, другой — для электронных адресов.

Папа. Другие столбики можно увидеть, используя клавиши ← → как переключатели. Заголовок книги может стать таким:



Или таким:



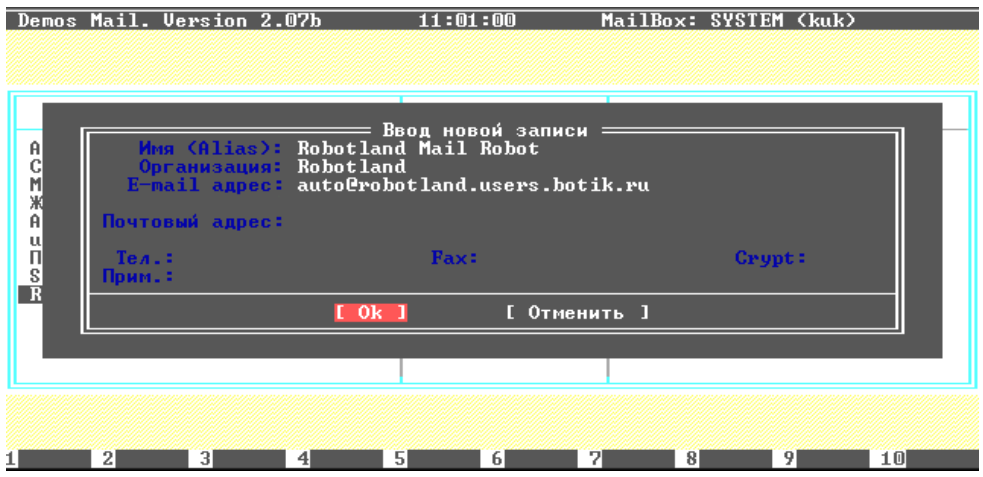
Вася. Сейчас книга пуста. Как добавить в нее адрес Паука?

Папа. Добавление происходит по команде **7Append**, то есть по клавише F7 (*append* — добавить). На экран выводится пустой "листочек" для записи:

Информацию в это окно можно занести вручную или взять из письма, расположенного в почтовом ящике.

Вася. Второй способ мне нравится больше.

Папа. Сейчас можно нажать клавишу F2 (команда **MesLst** — *message list* — список сообщений), и на экране отобразится содержимое почтового ящика. В списке писем нужно выбрать письмо Паука и нажать клавишу Enter. Информация из письма занесется в адресную книгу:



Вася. Заполнились не все поля на экране.

Папа. Почтовая программа сделала все, что смогла. Ты можешь заполнить другие поля вручную, но для отправки писем они не нужны.

Адресную книгу можно вызвать по аккорду Alt F2 и в момент просмотра списка писем почтового ящика. Правила добавления нового адреса остаются прежними.

Вася. Мне бы хотелось узнать назначение других команд адресной книги — они высвечиваются в нижней строчке экрана.

Папа. На самом деле нужно хорошо знать только одну команду **Help** — она в строке всегда записывается первой.

Вася. Намек понял. Я знаю, что по клавише F1 вызывается подсказка, инструкция, а английское слово *help* переводится на русский как "помощь".

Папа. Рекомендую тебе чаще использовать эту клавишу.

Вася. Знаю, знаю: когда уж совсем ничего не получается, нужно наконец прочитать инструкцию!

Окончание на с. 15

ТЕЛЕ-КОММУНИКАЦИИ

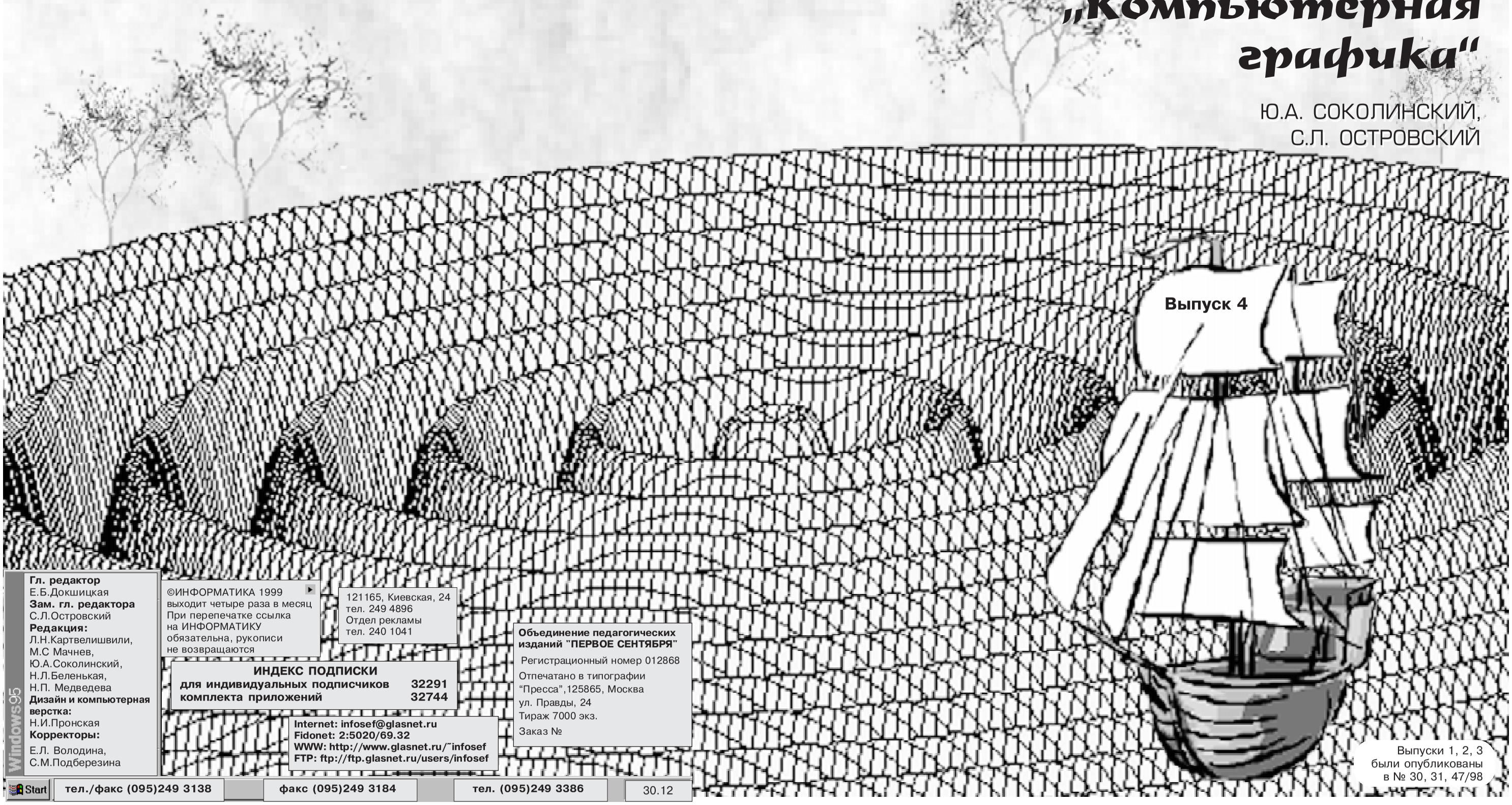
№ 2 (195) январь 1999

еженедельное
ПРИЛОЖЕНИЕ К ГАЗЕТЕ
«ПЕРВОЕ СЕНТЯБРЯ»

ИНФОРМАТИК

Задачи по теме „Компьютерная графика“

Ю.А. СОКОЛИНСКИЙ,
С.Л. ОСТРОВСКИЙ



Выпуски 1, 2, 3
были опубликованы
в № 30, 31, 47/98

Windows 95

Гл. редактор
Е.Б.Докшицкая
Зам. гл. редактора
С.Л.Островский
Редакция:
Л.Н.Картвелишвили,
М.С Мачнев,
Ю.А.Соколинский,
Н.Л.Беленькая,
Н.П. Медведева
Дизайн и компьютерная верстка:
Н.И.Пронская
Корректоры:
Е.Л. Володина,
С.М.Подберезина

121165, Киевская, 24
тел. 249 4896
Отдел рекламы
тел. 240 1041

ИНДЕКС ПОДПИСКИ
для индивидуальных подписчиков 32291
комплекта приложений 32744

Internet: infosef@glasnet.ru
Fidonet: 2:5020/69.32
WWW: http://www.glasnet.ru/~infosef
FTP: ftp://ftp.glasnet.ru/users/infosef

Объединение педагогических изданий "ПЕРВОЕ СЕНТЯБРЯ"
Регистрационный номер 012868
Отпечатано в типографии
“Пресса”,125865, Москва
ул. Правды, 24
Тираж 7000 экз.
Заказ №

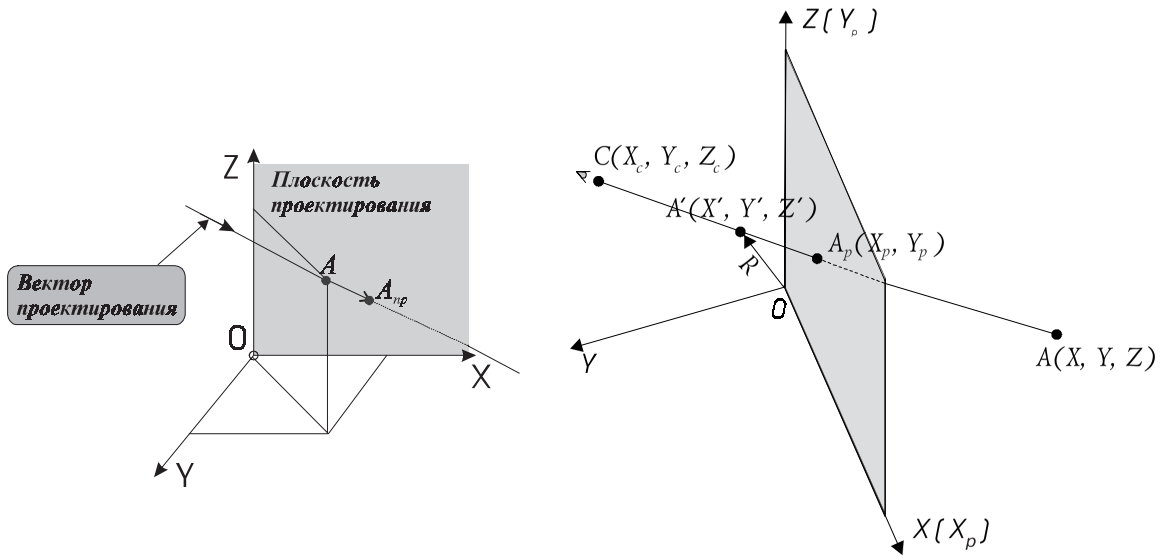
Start | тел./факс (095)249 3138 | факс (095)249 3184 | тел. (095)249 3386 | 30.12

Изображение пространственных тел с помощью параллельного и перспективного проектирования

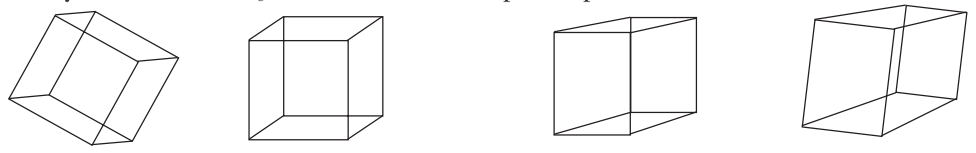
Можно ли изобразить пространственный объект на плоскости, следуя только строгим, однозначным правилам (т.е. выполняя некоторый алгоритм), без всякой апелляции к художественной интуиции? Этот вопрос интересовал еще итальянских художников эпохи Возрождения, и в том числе великого Леонардо да Винчи. В те времена сложились основы "учения о перспективе", в котором рассматривались методы решения этой задачи. Общий подход заключается в следующем. Выбирается плоскость, в которой будет находиться изображение объекта. Обычно ее называют "плоскостью проектирования", или "картинной плоскостью". Через каждую точку изображаемого объекта проводится прямая — линия проектирования. Точка пересечения этой линии и картинной плоскости и есть образ соответствующей точки пространства. Мы рассмотрим два варианта проектирования: *параллельное* и *перспективное*, отличающиеся выбором линии проектирования.

В случае параллельного проектирования задается общее для всех точек объекта направление линий проектирования (вектор проектирования). Вектор проектирования не должен быть параллелен картинной плоскости.

При перспективном проектировании фиксируется некоторая точка в пространстве — точка наблюдения, или центр проектирования. Линия проектирования проводится через центр проектирования и изображаемую точку объекта. Центр проектирования не должен лежать в картинной плоскости и совпадать с какой-либо точкой изображаемого объекта.



Перспективное проектирование дает более реалистичное изображение, чем параллельное, поскольку оно учитывает удаленность объектов от наблюдателя и плоскости проектирования, зато параллельное проектирование несколько проще. При этом параллельное проектирование можно рассматривать как частный случай перспективного, когда точка наблюдения бесконечно удалена. На рисунках представлены изображения кубика, полученные с помощью обоих методов проектирования.



Отметим ряд общих свойств данных методов проектирования:

— по точке в картинной плоскости нельзя однозначно восстановить ее прообраз в пространстве;

— точки пространства, лежащие на какой-либо линии проектирования, переходят в картинной плоскости в одну и ту же точку;

— прямая линия в пространстве переходит также в прямую линию в картинной плоскости.

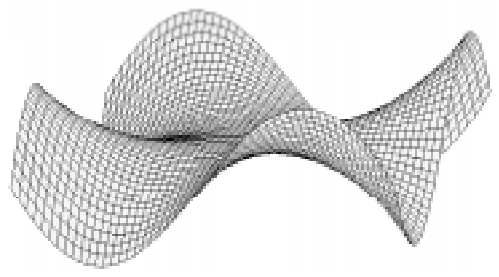
В качестве картинной плоскости мы выберем плоскость *ZOX*. Если обозначить через *Xp*, *Yp* координаты точек в картинной плоскости, то при *Y*=0 (это координата по *Y* точек плоскости *ZOX*) имеем:

$$Xp = X;\ Yp = Z$$


```
 RectC(k).y = (Rect2(k).y - y20) * M
 RectScr(k).x = Xscr!(RectC(k).x)
 RectScr(k).y = Yscr!(RectC(k).y)
NEXT k          'цикл элементарного четырехугольника на экране
'Совмещаем 5-ю точку массива RectScr с первой
RectScr(5) = RectScr(1)
'Изображаем элементарный четырехугольник красным цветом
FOR k = 1 TO 4
 Xb = RectScr(k).x: Yb = RectScr(k).y
 Xe = RectScr(k + 1).x: Ye = RectScr(k + 1).y
 LINE (Xb, Yb)-(Xe, Ye), Red%
NEXT k
'Находим границы элементарного четырехугольника
Corner(1) = RectScr(1): Corner(2) = RectScr(1)
FOR k = 1 TO 4
 CALL Verge(RectScr(k), Corner())
NEXT k
'Xm - абсцисса вертикали, на которой ищется опорная точка
Xm = (Corner(1).x + Corner(2).x) / 2
'v - индекс точек вдоль опорной вертикали
'Status имеет значения:
'для точек выше первой красной зоны - 0
'для точек первой красной зоны - 1
'для точек между первой и второй красной зоной - 2
'для точек второй красной зоны - 3
v = INT(Corner(1).y) - 10     'Начальное значение v
vmax = INT(Corner(2).y) + 10 'Конечное значение v
Status = 0                    'Начальное значение Status
DO 'Цикл поиска первой точки второй красной зоны
 IF POINT(Xm, v) = Red% THEN
  SELECT CASE Status
   CASE 0 TO 1: Status = 1
   CASE 2: Status = 3
  END SELECT
 ELSE
  SELECT CASE Status
   CASE 0: Status = 0
   CASE 1 TO 2: Status = 2
  END SELECT
 END IF
 IF v < vmax AND Status < 3 THEN v = v + 1
LOOP UNTIL v = vmax OR Status = 3
'Если вторая красная зона существует, то получаем опорную точку
'и закрашиваем элементарный четырехугольник синим цветом
IF Status = 3 THEN
 v = v - 1: PAINT (Xm, v), Blue%, Red%
END IF
'Изображаем элементарный четырехугольник белым цветом
FOR k = 1 TO 4
 Xb = RectScr(k).x: Yb = RectScr(k).y
 Xe = RectScr(k + 1).x: Ye = RectScr(k + 1).y
 LINE (Xb, Yb)-(Xe, Ye), White%
NEXT k
 NEXT j          'цикл X-плоскостей сечения
NEXT i           'цикл Y-плоскостей сечения
END SUB
```

```
'Вектор проектирования
DIM SHARED Vpp AS tPoint3
Vpp.x = -.25: Vpp.y = -1: Vpp.z = -.25
DIM SHARED St AS SINGLE: St = 5 'Шаг плоскостей сечения
DIM SHARED PrScr AS SINGLE 'Процент использования экрана
PrScr = 80
DIM SHARED W AS SINGLE, H AS SINGLE
W = 640: H = 480 'ширина и высота экрана
DIM NF AS INTEGER
FOR NF = 1 TO 3 'цикл поверхностей
SCREEN 12 'Переход в графический режим для монитора VGA
CALL Planes(NF)
DO: LOOP UNTIL INKEY$ = CHR$(13)
SCREEN 0 'Переход в текстовый режим
NEXT NF          'цикл поверхностей
END

SUB Planes (NF AS INTEGER)
'Основная процедура построения поверхности
DIM M AS SINGLE, x20 AS SINGLE, y20 AS SINGLE, Xm AS SINGLE
DIM Xb AS SINGLE, Yb AS SINGLE, Xe AS SINGLE, Ye AS SINGLE
DIM i AS INTEGER, j AS INTEGER, k AS INTEGER, N AS INTEGER
DIM P3 AS tPoint3, P2 AS tPoint, Corner(1 TO 2) AS tPoint
'Rect3,Rect2 — массивы вершин элементарного четырехугольника
'на  поверхности и в картинной плоскости
'RectC,RectScr — массивы вершин элементарного четырехугольника
'на экране в центральных и экранных координатах
DIM Rect3(1 TO 4) AS tPoint3, Rect2(1 TO 4) AS tPoint
DIM RectC(1 TO 4) AS tPoint, RectScr(1 TO 5) AS tPoint
Blue% = 1: Red% = 4: White% = 15
DIM v AS INTEGER, vmax AS INTEGER, Status AS INTEGER
'X0,Y0 — координаты левой, удаленной вершины рабочей области
DIM X0 AS SINGLE, Y0 AS SINGLE: X0 = -B: Y0 = -B
N = INT(2 * B / St) 'число шагов
'Находим начальные значения угловых точек
Z0 = func!(X0, Y0, NF)
P3.x = X0: P3.y = Y0: P3.z = Z0
CALL pp(P3, Vpp, P2)
Corner(1) = P2: Corner(2) = P2
'Находим фактические значения угловых точек
FOR i = 0 TO N 'цикл по Y
 y1 = Y0 + i * St
 FOR j = 0 TO N 'цикл по X
  x1 = X0 + j * St: z1 = func!(x1, y1, NF)
  P3.x = x1: P3.y = y1: P3.z = z1
  CALL pp(P3, Vpp, P2)
  CALL Verge(P2, Corner())
 NEXT j          'цикл по X
NEXT i          'цикл по Y
'Определение масштаба и центра картинной плоскости
CALL ScaleC2(PrScr, Corner(), M, x20, y20)
FOR i = 0 TO N — 1'цикл Y-плоскостей сечения
 y1 = Y0 + i * St
 FOR j = 0 TO N — 1'цикл X-плоскостей сечения
  x1 = X0 + j * St:               z1 = func!(x1, y1, NF)
  x2 = x1:          y2 = y1 + St: z2 = func!(x2, y2, NF)
  x3 = x1 + St:     y3 = y2:      z3 = func!(x3, y3, NF)
  x4 = x3:          y4 = y1:      z4 = func!(x4, y4, NF)
  Rect3(1).x = x1: Rect3(1).y = y1: Rect3(1).z = z1
  Rect3(2).x = x2: Rect3(2).y = y2: Rect3(2).z = z2
  Rect3(3).x = x3: Rect3(3).y = y3: Rect3(3).z = z3
  Rect3(4).x = x4: Rect3(4).y = y4: Rect3(4).z = z4
  FOR k = 1 TO 4 'цикл элементарного четырехугольника на экране
   CALL pp(Rect3(k), Vpp, Rect2(k))
   RectC(k).x = (Rect2(k).x — x20) * M
```



Пусть $\boldsymbol{a}=(a_x, a_y, a_z)$ — вектор, определяющий направление линии проектирования. В случае параллельного проектирования этот вектор задается, а при перспективном проектировании он зависит от изображаемой точки. При проектировании точки $A(X, Y, Z)$ его проекции имеют такой вид:

$$a_x = X - Xc; \qquad a_y = Y - Yc; \qquad a_z = Z - Zc$$

где Xc, Yc, Zc — координаты центра проектирования.

Получим формулы, выражающие координаты точки $Ap(Xp, Yp)$ в картинной плоскости через координаты ее прообраза — точки $A(X, Y, Z)$ и вектора проектирования $\boldsymbol{a}$. Уравнение прямой, проходящей через точку A, можно записать в виде:

$$\boldsymbol{R} = \boldsymbol{RA} + t\boldsymbol{a}$$

Здесь $\boldsymbol{R}$ — радиус-вектор точки A' на прямой, $\boldsymbol{RA}$ — радиус-вектор точки A, t — параметр. В проекциях это выглядит как:

$$X' = X + ta_x$$
$$Y' = Y + ta_y$$
$$Z' = Z + ta_z$$

В картинной плоскости проекция по оси Y равна нулю, поэтому параметр t составляет: $t = -Y/a_y$ и координаты точки в картинной плоскости выражаются таким образом:

$$Xp = X' = X - Y\frac{a_x}{a_y} \qquad Yp = Z' = Z - Y\frac{a_z}{a_y}$$

Подробнее эти вопросы рассмотрены в № 20, 22 “Информатики” за 1998 г. в статье: «Дети просят задачку “на лето”».

При решении задач изображения пространственных тел и поверхностей на плоскости нам понадобятся типы данных: *точка в пространстве* `tPoint3` и *точка в картинной плоскости* `tPoint2`. Вот как они выглядят в различных языках программирования.

Язык Паскаль

```
tPoint3=record x,y,z:real; end;
tPoint2=record x,y:real; end;
```

Язык Си

```
typedef struct {float x; float y; float z;} tPOINT3;
typedef struct {float x; float y;} tPOINT2;
```

Язык Бейсик

```
TYPE tPoint3: x AS SINGLE: y AS SINGLE: z AS SINGLE: END
```

Указанные типы данных мы внесем в подключаемые файлы GRTOOL.PAS и GRTOOL.H. Приведем две процедуры (функции) проектирования точки в пространстве `P3` в точку картинной плоскости `P2`:

`pp` — параллельное проектирование по направлению вектора `Vpp` и

`cp` — перспективное проектирование с центром `Pc`.

Язык Паскаль

```
procedure pp(P3,Vpp:tPoint3; var P2:tPoint2);
var ax,ay,az:real;
begin
ax:=Vpp.x; ay:=Vpp.y; az:=Vpp.z;
P2.x:=P3.x-P3.y*ax/ay; P2.y:=P3.z-P3.y*az/ay
end;

procedure cp(P3,Pc:tPoint3; var P2:tPoint2);
var ax,ay,az:real;
begin
ax:=P3.x-Pc.x; ay:=P3.y-Pc.y; az:=P3.z-Pc.z;
P2.x:=P3.x-P3.y*ax/ay; P2.y:=P3.z-P3.y*az/ay
end;{cp}
```

Язык Си

```
tPOINT2 pp(tPOINT3 P3, tPOINT3 Vpp)
{float ax,ay,az; tPOINT2 P2;
 ax=Vpp.x; ay=Vpp.y; az=Vpp.z;
 P2.x=P3.x-P3.y*ax/ay; P2.y=P3.z-P3.y*az/ay;
 return P2;
}

tPOINT2 cp(tPOINT3 P3, tPOINT3 Pc)
{float ax,ay,az; tPOINT2 P2;
 ax=P3.x-Pc.x; ay=P3.y-Pc.y; az=P3.z-Pc.z;
 P2.x=P3.x-P3.y*ax/ay; P2.y=P3.z-P3.y*az/ay;
 return P2;
}
```

Язык Бейсик

```
SUB pp (P3 AS tPoint3, Vpp AS tPoint3, P2 AS tPoint)
DIM ax AS SINGLE, ay AS SINGLE, az AS SINGLE
ax = Vpp.x: ay = Vpp.y: az = Vpp.z:
P2.x = P3.x - P3.y * ax / ay
P2.y = P3.z - P3.y * az / ay
END SUB
SUB cp (P3 AS tPoint3, Pc AS tPoint3, P2 AS tPoint)
DIM ax AS SINGLE, ay AS SINGLE, az AS SINGLE
ax = P3.x - Pc.x: ay = P3.y - Pc.y: az = P3.z - Pc.z
P2.x = P3.x - P3.y * ax / ay
P2.y = P3.z - P3.y * az / ay
END SUB
```

Пусть мы получили изображение в картинной плоскости. Вполне возможно, что изображение или его часть окажется за пределами экрана. Возможен и другой случай, когда изображение окажется внутри экрана, но будет слишком сжатым. Видимо, надо выполнить преобразование растяжения, выбрав центр и коэффициент растяжения таким образом, чтобы изображение занимало заданную часть экрана. Займемся этой задачей. Обозначим через *Xmin, Xmax* — абсциссы самой левой и самой правой точки в картинной плоскости и аналогично *Ymin, Ymax* — ординаты самой нижней и самой верхней точки. Изображение оказалось вписано в прямоугольник со сторонами, параллельными координатным осям. Левый нижний угол этого прямоугольника — это точка (*Xmin, Ymin*), а правый верхний — точка (*Xmax, Ymax*). В качестве центра растяжения выберем центр указанного прямоугольника, т.е. точку с координатами

$$Xp_0 = (Xmax + Xmin)/2; \qquad Yp_0 = (Ymax + Ymin)/2$$

Перейдем к определению коэффициента растяжения или, другими словами, масштаба. Масштаб по оси *X* составляет

$$Mx = mW/(Xmax - Xmin)$$

где *W* — ширина экрана, а *m* — коэффициент, определяющий, какая часть экрана используется для вывода изображения. Аналогично масштаб по оси *X* составляет

$$My = mH/(Ymax - Ymin)$$

где *H* — высота экрана. Чтобы избежать искажения формы изображения, надо выбрать единый масштаб по обеим осям как меньший из *Mx* и *My*: $M = min(Mx, My)$. Теперь мы можем получить выражения для центральных координат (*X, Y*) точки на экране, которая соответствует точке (*Xp,Yp*) в картинной плоскости:

$$X = (Xp - Xp_0)M; \qquad Y = (Yp - Yp_0)M$$

Координаты углов прямоугольника, в который вписано изображение, будем хранить в массиве из двух элементов `Corner`. Первый из них — левый нижний угол, а второй — правый верхний. Описания этого массива имеют вид:

зоваться лишь процедурой `PAINT`, которая закрашивает замкнутую область заданным цветом при условии, что заданы опорная точка внутри области и цвет контура (похожую процедуру мы составляли, решая задачу 9 — №31/98). На первый взгляд кажется, что и эта процедура не поможет. Действительно, мы закрашиваем очередной элементарный четырехугольник, чтобы скрыть границы предыдущих четырехугольников, попавших внутрь него. Но процедура `PAINT` не знает, какая граница новая, а какая старая. Тогда поступим следующим образом. Первый раз проведем очередной элементарный четырехугольник не выбранным цветом контура (таким является белый цвет), а каким-нибудь другим, а именно красным. Теперь, выполнив процедуру `PAINT`, мы закрасим старые границы, а после этого вторично обведем четырехугольник уже белым цветом.

Следующий вопрос — как выбрать опорную точку? Казалось бы, это легкая задача. Например, можно взять любую точку одной из двух диагоналей (в частности, среднюю). Однако это правильно только для выпуклого четырехугольника. А некоторые элементарные четырехугольники — вогнутые (стреловидной формы), и выбранная диагональ и тем самым опорная точка могут оказаться вне четырехугольника, после чего нормальная работа программы прекратится. Поэтому для поиска опорной точки используется не такой простой, но зато надежный алгоритм, к описанию которого мы и приступим. Найдем углы четырехугольника, в который вписан наш элементарный четырехугольник, т.е. массив `Corner`, используя процедуру `Verge`. Опорную точку будем искать на вертикали с абсциссой

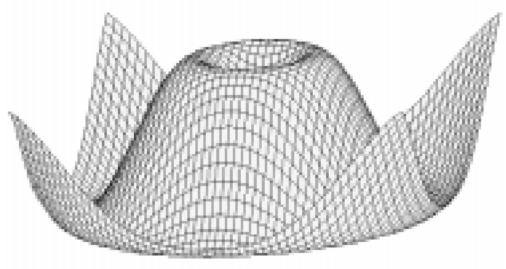
```
Xm = (Corner(1).x+ Corner(2).x)/2.
```

Естественно считать, что эта опорная вертикаль пересекает наш четырехугольник в двух или четырех точках, имеющих красный цвет. (В последнем случае четырехугольник наверняка вогнутый.) При этом точки, расположенные между первой и второй красной точкой (а также между третьей и четвертой, если они существуют), находятся внутри четырехугольника. Однако это утверждение не точное, что связано с дискретностью растра. Отрезки с углом наклона, близким к 90^0, изображаются на экране в виде вертикальных групп пикселей, отстоящих друг от друга на один пиксель. (Разумеется, аналогичное утверждение имеет место и для отрезков с малым углом наклона.) Поэтому точке пересечения опорной вертикали с четырехугольником соответствует красная зона, состоящая из нескольких красных пикселей. В качестве опорной мы выберем последнюю точку (пиксель), предшествующую второй красной зоне. Поиск будем вести, начиная с пикселя, находящегося несколько выше верхней границы `Corner(1).y` (скажем, на 10 пикселей), и кончая пикселем, находящимся на столько же ниже нижней границы `Corner(2).y`. (Напомним, что мы сейчас используем экранные координаты и ось *Y* направлена сверху вниз.) Дадим точкам этого интервала *статус* следующим образом: точки, лежащие выше первой красной зоны, имеют статус 0; точки первой красной зоны имеют статус 1; точки между первой и второй красной зоной имеют статус 2; точки второй красной зоны имеют статус 3. Первая точка имеет статус 0. Просматривая точки указанного интервала сверху вниз, начиная со второй, определяем их статус в зависимости от цвета очередной точки и статуса предшествующей. Если цвет точки — красный и статус предыдущей — 0 или 1, то она получает статус 1. При том же цвете и статусе предыдущей точки 2 новый статус равен 3. Если же цвет точки отличается от красного, а статус предыдущей — 1 или 2, то она получает статус 2. Если точка также не красная, а статус предыдущей — 0, то он не меняется. Этот процесс прекращаем, когда очередная точка получила статус 3 или она последняя. Если по завершении цикла статус отличается от 3, то это означает, что четырехугольник выродился в отрезок и закрашивать его не надо. В нормальном случае предпоследняя точка — опорная.

Приведем саму программу.

```
'Изображения трех пространственных поверхностей методом
'кривых с помощью параллельного проектирования,
'не содержащие невидимых линий
'Тип точки на экране или картинной плоскости
TYPE tPoint: x AS SINGLE: y AS SINGLE: END TYPE
'Тип точки в пространстве
TYPE tPoint3: x AS SINGLE: y AS SINGLE: z AS SINGLE: END TYPE
DECLARE SUB Verge (Ppl AS tPoint, Corner() AS tPoint)
DECLARE SUB Planes (NF AS INTEGER)
DECLARE SUB pp (P3 AS tPoint3, Vpp AS tPoint3, P2 AS tPoint)
DECLARE SUB ScaleC2 (PrScr!, Corner() AS tPoint, M!, x20!, y20!)
DECLARE FUNCTION func! (x AS SINGLE, y AS SINGLE, i AS INTEGER)
DECLARE FUNCTION Xscr! (x!)
DECLARE FUNCTION Yscr! (y!)
DIM SHARED B AS SINGLE
B = 250'Половина габарита рабочей области
```

```
 int RectScr[8];
 /*X0,Y0 — координаты левой, удаленной вершины рабочей области*/
 X0=-B; Y0=-B;
 N=round(2*B/Step);/* Число шагов */
 /* Находим начальные значения угловых точек */
 Z0=func(X0,Y0,NF);
 P3.x=X0; P3.y=Y0; P3.z=Z0;
 P2=pp(P3,Vpp);
 for (k=0; k<2; k++)
 {Corner[k].x=P2.x; Corner[k].y=P2.y;}
 /* Находим фактические значения угловых точек */
 for (i=0; i<=N; i++) /* Цикл по Y */
  {y1=Y0+i*Step;
   for (j=0; j<=N; j++) /* Цикл по X */
    {x1=X0+j*Step; z1=func(x1,y1,NF);
     P3.x=x1; P3.y=y1; P3.z=z1;
     P2=pp(P3,Vpp);
     Verge(P2,Corner);
    }                   /* Цикл по X */
  }                 /* Цикл по Y */
/*Определение масштаба и центра картинной плоскости*/
ScaleC2(PrScr,Corner,&M,&x20,&y20);
setcolor(WHITE); /*цвет линий — белый*/
setfillstyle(SOLID_FILL,BLUE);/* Цвет закраски — синий */
for(i=0;i<N;i++) /* Цикл Y-плоскостей сечения */
 {y1=Y0+i*Step;
  for(j=0;j<N;j++) /* Цикл X-плоскостей сечения */
   {x1=X0+j*Step;               z1=func(x1,y1,NF);
    x2=x1;        y2=y1+Step;    z2=func(x2,y2,NF);
    x3=x1+Step;   y3=y2;         z3=func(x3,y3,NF);
    x4=x3;        y4=y1;         z4=func(x4,y4,NF);
    Rect3[0].x=x1; Rect3[0].y=y1; Rect3[0].z=z1;
    Rect3[1].x=x2; Rect3[1].y=y2; Rect3[1].z=z2;
    Rect3[2].x=x3; Rect3[2].y=y3; Rect3[2].z=z3;
    Rect3[3].x=x4; Rect3[3].y=y4; Rect3[3].z=z4;
    for (k=0; k<4; k++)/*Цикл элементарного четырехугольника на экране*/
     {Rect2[k]=pp(Rect3[k],Vpp);
      RectC[k].x=round((Rect2[k].x-x20)*M);
      RectC[k].y=round((Rect2[k].y-y20)*M);
      RectScr[2*k]=Xscr(RectC[k].x);
      RectScr[2*k+1]=Yscr(RectC[k].y);
     }                  /*Цикл элементарного четырехугольника на экране*/
    /* Изображение элементарного четырехугольника на экране */
    fillpoly(4,RectScr);
   }                /* Цикл X-плоскостей сечения */
 }              /* Цикл Y-плоскостей сечения */
}
void main(void)
{int NF;
 Vpp.x=Ax; Vpp.y=Ay; Vpp.z=Az;/*Вектор проектирования*/
 for (NF=1; NF<=3; NF++) /* Цикл поверхностей */
  {initgraph(&gd,&gm,PATH); /* Переход в графический режим */
   WH(&W,&H); /* Определение ширины и высоты экрана */
   planes(NF);/* Изображаем очередную поверхность */
   getch(); /* Выход — нажатием любой клавиши */
   closegraph(); /* Переход в текстовый режим */
  }                      /* Цикл поверхностей */
}
```



Язык Бейсик

Программа на Бейсике отличается от аналогичных программ на Паскале и Си не только синтаксисом, но и существенными деталями алгоритма. Это связано с тем, что в Паскале и Си имеется процедура закраски многоугольника fillpoly, а в системе QBasic такого средства нет. Мы можем восполь-

Язык Паскаль

```
type tCorner=array[1..2] of tPoint2;
var Corner:tCorner;
```

Язык Си

```
tPOINT2 Corner[2];
```

Язык Бейсик

```
DIM Corner(1 TO 2) AS tPoint
```

Эти описания вносим в подключаемые файлы GRTOOL.PAS и GRTOOL.H.

Составим процедуру Verge, которая решает следующую задачу. Пусть в ходе анализа нескольких первых точек изображения найдены границы изображения, включающего эти точки, хранящиеся в массиве Corner. Добавилась новая точка — Ppl. Узнать, как изменились границы.

Язык Паскаль

```
procedure Verge(Ppl:tPoint2; var Corner:tCorner);
begin
  if Ppl.x<Corner[1].x then Corner[1].x:=Ppl.x;
  if Ppl.x>Corner[2].x then Corner[2].x:=Ppl.x;
  if Ppl.y<Corner[1].y then Corner[1].y:=Ppl.y;
  if Ppl.y>Corner[2].y then Corner[2].y:=Ppl.y;
end;{Verge}
```

Язык Си

```
void Verge(tPOINT2 Ppl, tPOINT2 *Corner)
{if (Ppl.x<Corner[0].x) Corner[0].x=Ppl.x;
 if (Ppl.x>Corner[1].x) Corner[1].x=Ppl.x;
 if (Ppl.y<Corner[0].y) Corner[0].y=Ppl.y;
 if (Ppl.y>Corner[1].y) Corner[1].y=Ppl.y;
}
```

Язык Бейсик

```
SUB Verge (Ppl AS tPoint, Corner() AS tPoint)
IF Ppl.x < Corner(1).x THEN Corner(1).x = Ppl.x
IF Ppl.x > Corner(2).x THEN Corner(2).x = Ppl.x
IF Ppl.y < Corner(1).y THEN Corner(1).y = Ppl.y
IF Ppl.y > Corner(2).y THEN Corner(2).y = Ppl.y
END SUB
```

Как в программах использовать эту процедуру? Значение первой точки изображения присваиваем обоим элементам массива Corner. Далее в цикле применяем процедуру Verge для второй, третьей точки и т.д. вплоть до последней точки. Получаем окончательное значение массива Corner. Возможно, у читателя возникнут возражения. Не лучше ли объединить все точки изображения в единый массив — параметр процедуры Verge, изменить ее соответствующим образом и получить значение массива Corner однократным обращением к этой процедуре? Дело в том, что точек может быть очень много (порядка десятков тысяч), и в этом случае такой подход становится неприемлемым.

Теперь приведем процедуру ScaleC2, которая, имея на входе массив Corner и заданный процент использования экрана PrScr, находит масштаб M и координаты центра растяжения x20, y20.

Язык Паскаль

```
procedure ScaleC2(PrScr:real; var Corner:tCorner; var M,x20,y20:real);
var
  Mx,My,Xmin,Xmax,Ymin,Ymax:real;
begin
  Xmin:=Corner[1].x; Ymin:=Corner[1].y;
  Xmax:=Corner[2].x; Ymax:=Corner[2].y;
  {координаты центра картинной плоскости}
  x20:=(Xmin+Xmax)/2;
  y20:=(Ymin+Ymax)/2;
  {Масштаб по X и Y}
```

```
  if Xmax>Xmin then
     Mx:=PrScr/100*W/(Xmax-Xmin)
  else
  begin M:=PrScr/100*H/(Ymax-Ymin); exit end;
  if (Ymax>Ymin) then
     My:=PrScr/100*H/(Ymax-Ymin)
  else begin M:=Mx; exit end;
  {Определяем масштаб как меньший из Mx и My}
  if Mx<My then M:=Mx else M:=My;
end;{ScaleC2}
```

Язык Си

```
void ScaleC2(float PrScr,tPOINT2 *Corner,
          float *M,float *x20,float *y20)
{float Mx,My,Xmin,Xmax,Ymin,Ymax;
 Xmin=Corner[0].x; Xmax=Corner[1].x;
 Ymin=Corner[0].y; Ymax=Corner[1].y;
 *x20=(Xmin+Xmax)/2;  *y20=(Ymin+Ymax)/2;
 /*Масштаб по X и Y*/
 if (Xmax>Xmin) Mx=PrScr/100.0*W/(Xmax-Xmin);
 else {*M=PrScr/100.0*H/(Ymax-Ymin); return;}
 if (Ymax>Ymin)
 My=PrScr/100.0*H/(Ymax-Ymin);
 else {*M=Mx; return;}
 /*Определяем масштаб как меньший из Mx и My*/
 if (Mx<My) *M=Mx; else *M=My;
}
```

Язык Бейсик

```
SUB ScaleC2 (PrScr!, Corner() AS tPoint, M!, x20!, y20!)
DIM Mx!, My!, Xmin!, Xmax!, Ymin!, Ymax!
'Corner(1) — левая, нижняя точка
'Corner(2) — правая, верхняя точка
Xmin! = Corner(1).x: Ymin! = Corner(1).y
Xmax! = Corner(2).x: Ymax! = Corner(2).y
'координаты центра картинной плоскости
x20! = (Xmin! + Xmax!) / 2
y20! = (Ymin! + Ymax!) / 2
'Масштаб по X и Y
IF Xmax! > Xmin! THEN
   Mx! = PrScr! * .01 * W / (Xmax! — Xmin!)
ELSE
   M! = PrScr! * .01 * H / (Ymax! — Ymin!): EXIT SUB
END IF
IF Ymax! > Ymin! THEN
   My! = PrScr! * .01 * H / (Ymax! — Ymin!)
ELSE
   M! = Mx!: EXIT SUB
END IF
'Определяем масштаб как меньший из Mx и My
IF Mx! < My! THEN M! = Mx! ELSE M! = My!
END SUB
```

Разумеется, все приведенные процедуры: `pp`, `cp`, `Verge`, `ScaleC2` — мы внесем в подключаемые файлы GRTOOL.PAS и GRTOOL.H.

Перейдем к очередной задаче.

Задача 14. Изобразить кубик, используя параллельное и перспективное проектирование. Получить различные изображения кубика, варьируя вектор проектирования для случая параллельного проектирования, а при перспективном проектировании, меняя центр проектирования.

Решение. Прежде всего обсудим способы представления кубика в пространстве, в картинной плоскости и на экране. Как и в предыдущих случаях, используем соответствующие массивы и матрицу связи


```
      Verge(P2, Corner);
                  end;  {цикл по X}
               end; {цикл по Y}
{Определение масштаба и центра картинной плоскости}
ScaleC2(PrScr,Corner,M,x20,y20);
SetColor(White);{цвет линий — белый}
SetFillStyle(SolidFill,Blue); {цвет закраски — синий}
for i:=0 to N-1 do begin {цикл Y-плоскостей сечения}
 y1:=Y0+i*Step;
 for j:=0 to N-1 do begin {цикл X-плоскостей сечения}
 x1:=X0+j*Step;                z1:=func(x1,y1,NF);
 x2:=x1;        y2:=y1+Step;   z2:=func(x2,y2,NF);
 x3:=x1+Step;   y3:=y2;        z3:=func(x3,y3,NF);
 x4:=x3;        y4:=y1;        z4:=func(x4,y4,NF);
 Rect3[1].x:=x1; Rect3[1].y:=y1; Rect3[1].z:=z1;
 Rect3[2].x:=x2; Rect3[2].y:=y2; Rect3[2].z:=z2;
 Rect3[3].x:=x3; Rect3[3].y:=y3; Rect3[3].z:=z3;
 Rect3[4].x:=x4; Rect3[4].y:=y4; Rect3[4].z:=z4;
 for k:=1 to 4 do begin {цикл элементарного четырехугольника на экране}
 pp(Rect3[k],Vpp,Rect2[k]);
 RectC[k].x:=round((Rect2[k].x-x20)*M);
 RectC[k].y:=round((Rect2[k].y-y20)*M);
 RectScr[k].x:=Xscr(RectC[k].x);
 RectScr[k].y:=Yscr(RectC[k].y);
                end; {цикл элементарного четырехугольника на экране}
 {Изображение элементарного четырехугольника на экране}
 FillPoly(4,RectScr);
                 end; {цикл X-плоскостей сечения}
                end; {цикл Y-плоскостей сечения}
end;{Planes}
BEGIN
Vpp.x:=ax; Vpp.y:=ay; Vpp.z:=az;{вектор проектирования}
for NF:=1 to 3 do begin {цикл поверхностей}
gd:=DETECT;
initgraph(gd,gm,Path);
WH(W,H);{находим ширину и высоту экрана}
planes(NF); {изображаем очередную поверхность}
readln;
closegraph;
                end; {цикл поверхностей}
END.
```

Язык Си

```
/* Изображения трех пространственных поверхностей методом кривых  с  помощью
параллельного  проектирования, не содержащие невидимых линий */
#include "grtool.h"
/* Вектор проектирования */
#define Ax -0.25
#define Ay -1
#define Az -0.25
#define Step 5 /* Шаг плоскостей сечений */
#define B 250 /* Половина габарита рабочей области */
#define PrScr 80 /* Процент использования экрана */
void planes(int NF)
/* Основная функция построения поверхности */
{int i,j,k,N;
 float X0,Y0,Z0,x1,x2,x3,x4,y1,y2,y3,y4,z1,z2,z3,z4;
 float x20,y20,M,Mx,My,Xmin,Xmax,Ymin,Ymax;
 tPOINT3 P3; tPOINT2 P2;
 /*Rect3,Rect2 — массивы вершин элементарного четырехугольника на  поверхности и в
картинной плоскости RectC,RectScr — массивы вершин элементарного четырехугольника
на экране в центральных и экранных координатах */
 tPOINT3 Rect3[4]; tPOINT2 Rect2[4]; tPOINT RectC[4];
```

изображать элементарные четырехугольники, заключенные между ними, а затем двигаться “к нам”, увеличивая координату по *Y*. Находясь в очередной полосе, образованной указанными плоскостями, проводим соседние *X*-плоскости сечения, двигаясь слева направо, т.е. увеличивая координату по *X*. На каждом таком шаге получаем элементарный четырехугольник, который проектируем в картинную плоскость, а оттуда переносим его на экран, где и изображаем этот четырехугольник, рисуя контур и закрашивая его.

Из сказанного следует, что в отличие от предыдущей задачи здесь не предполагается совпадение координат картинной плоскости и центральных координат экрана. (На самом деле, как мы убедились в предыдущей задаче, оно имеет место. Однако, если изменить числовые параметры уравнений поверхностей и тем более их структуру, его, конечно, не будет.) Переход от первой из этих координатных систем ко второй выполняется, как указано выше.

На первом этапе решения задачи варьируем независимые переменные *x, y* с достаточно малым шагом в заданной области. Для каждой точки *(x, y)* находим значение *z* на поверхности. Точку поверхности *(x, y, z)* проектируем в картинную плоскость, получая точку P_2. Если она первая, то каждому элементу массива `Corner` присваиваем значение этой точки. Для всех остальных выполняем процедуру `Verge(P2,Corner)` и уточняем массив `Corner`. Заметим, что именно в этой задаче число просматриваемых точек настолько велико, что их нельзя организовать в массив. Получив окончательное значение массива `Corner`, с помощью процедуры `ScaleC2` находим центр растяжения — точку `(x20, y20)` и масштаб `M`, что и даст возможность перейти от элементарного четырехугольника в картинной плоскости к этому же четырехугольнику на экране.

Приводим программы изображения поверхностей способом сетки.

Язык Паскаль

```
{Изображения трех пространственных поверхностей методом кривых с помощью
параллельного проектирования, не содержащие невидимых линий }
uses graph,crt;
{$I grtool.pas}
const
{Проекции вектора проектирования}
ax:real=-0.25; ay:real=-1; az:real=-0.25;
Step:real=5; {шаг плоскостей сечений}
B:real=250; {Половина габарита рабочей области}
PrScr:real=90;{Процент использования экрана}
var NF:byte;
Procedure Planes(NF:byte);
{Основная процедура построения поверхности}
Var
X0,Y0,Z0,x1,x2,x3,x4,y1,y2,y3,y4,z1,z2,z3,z4:real;
P3:tPoint3; P2:tPoint2;
i,j,k,N:integer;
{Rect3,Rect2 — массивы вершин элементарного четырехугольника на поверхности и в
картинной плоскости RectC,RectScr — массивы вершин элементарного четырехугольника
на экране в центральных и экранных координатах}
Rect3:array[1..4] of tPoint3;
Rect2:array[1..4] of tPoint2;
RectC,RectScr:array[1..4] of tPoint;
begin
  {X0,Y0 — координаты левой, удаленной вершины рабочей области}
  X0:=-B; Y0:=-B;
  N:=round(2*B/Step);{число шагов}
  {Находим начальные значения угловых точек}
  Z0:=func(X0,Y0,NF);
  P3.x:=X0; P3.y:=Y0; P3.z:=Z0;
  pp(P3,Vpp,P2);
  Corner[1]:=P2; Corner[2]:=P2;
  {Находим фактические значения угловых точек}
  for i:=0 to N do begin {цикл по Y}
    y1:=Y0+i*Step;
    for j:=0 to N do begin {цикл по X}
       x1:=X0+j*Step; z1:=func(x1,y1,NF);
       P3.x:=x1; P3.y:=y1; P3.z:=z1;
       pp(P3,Vpp,P2);
```


`CubeMatrix`, которая была представлена выше. Массив вершин кубика в пространстве обозначим как `aP3`, в картинной плоскости — `aP2`, для экрана используем старое обозначение `aP`. Эти массивы объявляются так:

Язык Паскаль

```
type taP3=array[1..Nmax] of tPoint3;
type taP2=array[1..Nmax] of tPoint2;
var aP3:taP3; aP2:taP2;
```

Язык Си

```
tPOINT3 aP3[Nmax]; tPOINT2 aP2[Nmax];
```

Язык Бейсик

```
DIM aP3(1 TO Nmax) AS tPoint3, aP2(1 TO Nmax) AS tPoint
```

Напомним, что `Nmax` — константа, введенная выше, означает максимальное число точек изображения. Сторона кубика обозначается `a` и рассматривается в программах как константа. Кубик располагается так, чтобы его ребра были параллельны координатным осям. В начальном положении его первая точка совмещена с началом координат. Для вершин кубика в начальном положении используется массив-константа `aPcube3`.

В Си он выглядит так:

```
tPOINT3 aPcube3[Nmax]={
{0, 0, 0},{a, 0, 0},{a, a, 0},{0, a, 0},
{0, 0, a},{a, 0, a},{a, a, a},{0, a, a}};
```

в других языках — аналогично. Пусть “рабочее” положение кубика фиксируется координатами его первой вершины x_1, y_1, z_1 (для получения рабочего положения кубика его надо сдвинуть из начального положения на соответствующий вектор). Все указанные описания вносим в подключаемые файлы GRTOOL.PAS и GRTOOL.H, а в программу на Бейсике включаем процедуру `SUB PictCube3`, которая формирует массивы `aPcube3` и `CubeMatrix`.

Для получения различных изображений кубика будем использовать “плоскость проектирования”, проходящую через ось *Z*. Угол φ этой плоскости с плоскостью *ZOX* и будем варьировать. Он будет меняться от начального значения φ_0 с таким же шагом φ_0, пока будет оставаться меньше 180 градусов.

Для случая параллельного проектирования плоскость проектирования содержит единичный “вектор наблюдения”, наклоненный к горизонту под углом. (Мы смотрим на кубик сверху вниз.) Тогда проекции вектора проектирования выражаются так:

$$a_x = \cos(\pi+\alpha) \cdot \cos\varphi$$
$$a_y = \cos(\pi+\alpha) \cdot \sin\varphi$$
$$a_z = \sin(\pi+\alpha)$$

При перспективном проектировании в плоскости проектирования находится центр проектирования, координаты которого в этой плоскости обозначим как *b, Zc*. Координаты *Xc, Yc* центра проектирования находятся по формулам:

$$Xc = b\cos\varphi; \qquad Yc = b\sin\varphi$$

Демонстрируем программы изображения кубика.

Язык Паскаль

```
{изображение кубика с помощью параллельного и перспективного проектирования}
Uses Graph,Crt;
{$I grtool.pas}
const a=5;{сторона кубика}
{$I picture.pas}
const
Method: array[1..2] of string=(
'параллельное', 'перспективное');
```

```
PrScr:real=70;{% использования экрана}
AnglePro=30; {Угол проектирования для случая, когда проектирование параллельное}
z0=16; b= 25;{вертикальная координата и расстояние от оси Z центра проектирования
              для случая, когда проектирование перспективное}
Fi0=30; {Начальный угол плоскости проектирования}
{координаты первой точки кубика}
x1=1; y1=1; z1=0;
Var
i,j,Fi:Integer;
FiRad,AngleRad:real;
BEGIN
  Gd:=DETECT;
  for j:=1 to 2 do begin{цикл методов}
     writeln('демонстрируем ',Method[j],' проектирование');
    readln;
    initGraph(Gd,Gm,Path);{Переход в графический режим}
    WH(W,H);{находим ширину и высоту экрана}
    {Находим координаты вершин кубика}
    aP3[1].x:=x1; aP3[1].y:=y1; aP3[1].z:=z1;
    for i:=1 to Ncube do begin
        aP3[i].x:=aP3[1].x+aPcube3[i].x;
        aP3[i].y:=aP3[1].y+aPcube3[i].y;
        aP3[i].z:=aP3[1].z+aPcube3[i].z;
                        end;
    Fi:=Fi0;{Начальный угол плоскости проектирования}
    repeat          {Цикл демонстрации}
      FiRad:=Fi*Pi/180;
 {для случая, когда проектирование параллельное,находим вектор проектирования Vpp}
    if j=1 then begin
       AngleRad:=(AnglePro+180)*Pi/180;
       Vpp.x:=cos(AngleRad)*cos(FiRad);
       Vpp.y:=cos(AngleRad)*sin(FiRad);
       Vpp.z:=sin(AngleRad);
              end
     {для случая, когда проектирование перспективное,
     находим координаты Pc центра проектирования}
     else begin
        Pc.x:=b*cos(FiRad); Pc.y:=b*sin(FiRad); Pc.z:=z0;
          end;
    {Находим координаты вершин кубика в картинной плоскости}
    for i:=1 to Ncube do
       if j=1 then  pp(aP3[i],Vpp,aP2[i]) {проектирование параллельное}
       else         cp(aP3[i],Pc,aP2[i]); {проектирование перспективное}
      {Находим границы кубика в картинной плоскости}
    Corner[1]:=aP2[1]; Corner[2]:=aP2[1];
    for i:=2 to Ncube do Verge(aP2[i],Corner);
    {Находим масштаб и координаты центра картинной плоскости}
    ScaleC2(PrScr,Corner,M,x20,y20);
    {Находим центральные координаты вершин кубика на экране}
    for i:=1 to Ncube do begin
        aP[i].x:=round((aP2[i].x-x20)*M);
        aP[i].y:=round((aP2[i].y-y20)*M);
                        end;
    DrawPicture(Ncube,aP,CubeMatrix,white);{Рисуем кубик}
   Fi:=Fi+Fi0;{Наращиваем угол плоскости проектирования}
   readln;{Продолжение — нажатием клавиши Enter}
   DrawPicture(Ncube,aP,CubeMatrix,black);{Стираем кубик}
 until Fi>=180; {Цикл демонстрации}
 closegraph; {Переход в текстовый режим}
 end; {цикл методов}
END.
```



```
 'координаты по x начальной и конечной точки в
 'картинной плоскости для очередной плоскости сечения
 xstart = xb + i * dx: xstop = xe + i * dx
 FOR x = xstart TO xstop 'цикл вдоль плоскости сечения
  'проектируя "назад", находим точку в основании
  'плоскости сечения — на горизонтальной плоскости
  x0 = x — y * (ax / az): y0 = -y * (ay / az)
  'находим значение функции для x0,y0
  z0 = func!(x0, y0, NF)
  'находим аппликату образа точки (x0,y0,z0) на картинной плоскости
  z = z0 — (az / ay) * y0
  'Выясняем, надо ли выводить найденную точку
  f = 0:  xx = INT(x)
  IF z < Min(xx%) THEN
   Min(xx%) = z: f = 1
  END IF
  IF z > Max(xx%) THEN
   Max(xx%) = z: f = 1
  END IF
  IF f = 1 THEN PSET (Xscr!(x), Yscr!(z)), White
 NEXT x         'цикл вдоль плоскости сечения
NEXT i          'цикл плоскостей сечения
END SUB
```

Изображение пространственной поверхности, не содержащее невидимых точек, способом сетки

В предыдущей задаче поверхность изображалась как совокупность точек. Имеется другой, широко используемый способ изображения поверхности в виде сетки. Проводятся два семейства взаимно перпендикулярных плоскостей. Их пересечение с поверхностью дает два семейства кривых, т.е. сетку, состоящую из четырехугольников, которые мы будем называть элементарными. В качестве секущих плоскостей будут использоваться плоскости, перпендикулярные осям *X* и *Y*. Назовем их "*X*-плоскости сечения" и аналогично "*Y*-плоскости сечения". Пересечение с поверхностью двух соседних, *X* и *Y*, плоскостей сечения и образует элементарный четырехугольник. Эти понятия нам пригодятся в следующей задаче.

Задача **16. Изобразить способом сетки поверхности, заданные теми же уравнениями, что и в предыдущей задаче.**
Переменные *x, y* находятся в квадрате

$$-H \le x \le H$$

$$-H \le y \le H$$

Использовать параллельное проектирование. Поверхность должна содержать только видимые линии сетки. Все проекции вектора проектирования имеют отрицательные значения.

Решение. Для изображения только видимых линий используем "алгоритм художника", изложенный в статье: «Дети просят задачку "на лето"», "Информатика", № 20/98. Он заключается в следующем. Изображается каждый элементарный четырехугольник сетки. При этом не только рисуется его контур, но и сам он закрашивается каким-либо цветом. Главное здесь — соблюдать определенный порядок: вначале изображаются наиболее удаленные элементарные четырехугольники, при условии, что мы смотрим вдоль вектора проектирования. Тогда видимые элементарные четырехугольники просто закроют невидимые. Если координатные оси ориентированы так, что ось *X* направлена слева направо, а ось *Y* — "на нас", то отрицательность проекций вектора проектирования означает, что направление проектирования — сверху вниз и справа налево. Поэтому мы должны начинать с двух соседних *Y*-плоскостей сечения, наиболее удаленных от нас (имеющих минимальные координаты по *Y*), т.е.

```
  {initgraph(&gd,&gm,PATH); /* Переход в графический режим */
   WH(&W,&H); /* определение ширины и высоты экрана */
   Planes(NF); /* Изображение очередной поверхности */
   getch(); /* Продолжение — нажатием любой клавиши */
   closegraph(); /* Переход в текстовый режим */
  }                      /* цикл поверхностей */
}
```

Язык Бейсик

```
'Изображения трех пространственных поверхностей
'с помощью параллельного проектирования,
'не содержащие невидимых точек
DECLARE SUB Planes (NF AS INTEGER)
DECLARE FUNCTION func! (x AS SINGLE, y AS SINGLE, i AS INTEGER)
DECLARE FUNCTION Xscr! (x!)
DECLARE FUNCTION Yscr! (y!)
'Половина габарита рабочей области
DIM SHARED B AS SINGLE: B = 250
'Вектор проектирования
DIM SHARED ax AS SINGLE, ay AS SINGLE, az AS SINGLE
ax = -.25: ay = -1: az = -.25:
DIM SHARED d1 AS SINGLE: d1 = 5 'Шаг плоскостей сечения
DIM SHARED W AS SINGLE, H AS SINGLE
W = 640: H = 480 'ширина и высота экрана
DIM NF AS INTEGER
FOR NF = 1 TO 3 'цикл поверхностей
SCREEN 12 'Переход в графический режим для монитора VGA
CALL Planes(NF) 'Изображение очередной поверхности
'Продолжение — нажатием клавиши Enter
DO: LOOP UNTIL INKEY$ = CHR$(13)
SCREEN 0 'Переход в текстовый режим
NEXT NF          'цикл поверхностей
END

SUB Planes (NF AS INTEGER)
'Основная процедура построения поверхности
DIM x AS SINGLE, y AS SINGLE, z AS SINGLE
DIM dx AS SINGLE, dy AS SINGLE
DIM x0 AS SINGLE, y0 AS SINGLE, z0 AS SINGLE
DIM xb AS SINGLE, yb AS SINGLE, xe AS SINGLE
DIM xstart AS SINGLE, xstop AS SINGLE
DIM N AS INTEGER, i AS INTEGER, f AS INTEGER
DIM Nmima AS INTEGER, xx AS INTEGER
DIM White AS INTEGER: White = 15
Nmima = INT(10 * B)
DIM Min(-Nmima TO Nmima) AS SINGLE
DIM Max(-Nmima TO Nmima) AS SINGLE
N = INT(2 * B / d1) 'число шагов цикла плоскостей сечения
'инициализируем массивы Min,Max минимальных
'и максимальных аппликат картинной плоскости
FOR i = -Nmima TO Nmima
 Min(i) = 1000: Max(i) = -1000
NEXT i
'Находим начальную точку (xb,yb) на картинной плоскости,
'проектируя точку (-B,B,0)
xb = -B — (ax / ay) * B: yb = -(az / ay) * B
'Находим координату xe конечной точки,
'проектируя точку (B,B,0)
xe = B — (ax / ay) * B
'Находим шаги по x и y в картинной плоскости
dx = d1 * (ax / ay): dy = d1 * (az / ay)
FOR i = 0 TO N 'Цикл плоскостей сечения
 'Координата по y начальной точки на картинной плоскости
 'для очередной плоскости сечения
 y = yb + i * dy
```


Язык Си

```
/* изображение кубика с помощью параллельного и перспективного проектирования */
#include "grtool.h"
#define a 5 /* сторона кубика */
#include "picture.h"
#define PrScr 70 /* использования окна экрана */
#define AnglePro 30 /* Угол проектирования для случая, когда проектирование параллельное */
#define Z0 16 /* вертикальная координата и расстояние       */
#define B  25 /* от оси Z центра проектирования для         */
              /* случая, когда проектирование перспективное */
#define Fi0 30 /* Начальный угол плоскости проектирования */
/* координаты первой точки кубика */
#define x1 1
#define y1 1
#define z1 0
void main(void)
{int i,j,k,Fi;
 float FiRad,AngleRad;
 char *Method[2]={"параллельное","перспективное"};
 for (j=0; j<2; j++) /* цикл методов */
  {printf("\nдемонстрируем %.13s проектирование",Method[j]);
   getch();
   initgraph(&gd,&gm,PATH); /*Переход в графический режим*/
   WH(&W,&H); /* определение ширины и высоты экрана */
   /* Находим координаты вершин кубика */
   aP3[0].x=x1; aP3[0].y=y1; aP3[0].z=z1;
   for (i=0; i<Ncube; i++)
    {aP3[i].x=aP3[0].x+aPcube3[i].x;
     aP3[i].y=aP3[0].y+aPcube3[i].y;
     aP3[i].z=aP3[0].z+aPcube3[i].z;
    }
  Fi=Fi0; /* Начальный угол плоскости проектирования */
  do              /* Цикл демонстрации  */
   {FiRad=Fi*PI/180;
    /* для случая, когда проектирование параллельное,
    находим вектор проектирования Vpp  */
    if (j==0)
     {AngleRad=(AnglePro+180)*PI/180;
      Vpp.x=cos(AngleRad)*cos(FiRad);
      Vpp.y=cos(AngleRad)*sin(FiRad);
      Vpp.z=sin(AngleRad);
     }
    else /* для случая, когда проектирование перспективное,
         находим координаты Pc центра проектирования */
     {Pc.x=B*cos(FiRad); Pc.y=B*sin(FiRad); Pc.z=Z0;}
/*Находим координаты вершин кубика в картинной плоскости */
    for (i=0; i<Ncube; i++)
     {if (j==0)
    aP2[i]=pp(aP3[i],Vpp);/*проектирование параллельное*/
      else
    aP2[i]=cp(aP3[i],Pc);/*проектирование перспективное*/
     }
    /* Находим границы кубика в картинной плоскости */
    for (k=0; k<2; k++)
    {Corner[k].x=aP2[0].x; Corner[k].y=aP2[0].y;}
    for (i=1; i<Ncube; i++) Verge(aP2[i],Corner);
    /*Находим масштаб и координаты центра картинной плоскости*/
    ScaleC2(PrScr,Corner,&M,&x20,&y20);
/*Находим центральные координаты вершин кубика на экране*/
    for (i=0; i<Ncube; i++)
     {aP[i].x=round((aP2[i].x-x20)*M);
      aP[i].y=round((aP2[i].y-y20)*M);
     }
    /* Рисуем кубик */
```

```
     drawpicture(Ncube,aP,CubeMatrix,WHITE);
     Fi=Fi+Fi0;/*Наращиваем угол плоскости проектирования */
     getch(); /* Продолжение — нажатием любой клавиши */
     /* Стираем кубик */
     drawpicture(Ncube,aP,CubeMatrix,BLACK);
    }
  while (Fi<180);  /* Цикл демонстрации  */
  closegraph(); /* Переход в текстовый режим */
  }                  /* цикл методов */
}
```

Язык Бейсик

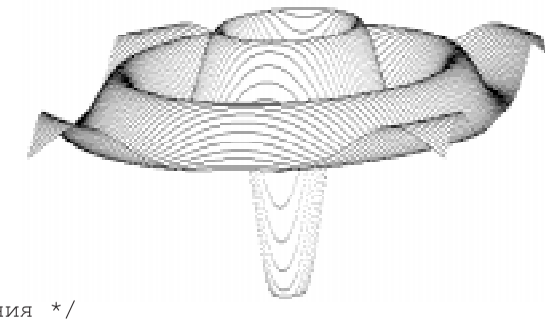
```
'изображение кубика с помощью параллельного и перспективного проектирования
'Тип точки на экране или картинной плоскости
TYPE tPoint: x AS SINGLE: y AS SINGLE: END TYPE
'Тип точки в пространстве
TYPE tPoint3: x AS SINGLE: y AS SINGLE: z AS SINGLE: END TYPE
DECLARE SUB Verge (Ppl AS tPoint, Corner() AS tPoint)
DECLARE SUB PictCube3 ()
DECLARE SUB pp (P3 AS tPoint3, Vpp AS tPoint3, P2 AS tPoint)
DECLARE SUB cp (P3 AS tPoint3, Pc AS tPoint3, P2 AS tPoint)
DECLARE SUB DrawPicture (N%, aP() AS tPoint, Matrix%(), ColorP%)
DECLARE SUB ScaleC2 (PrScr!, Corner() AS tPoint, M!, x20!, y20!)
DECLARE FUNCTION Xscr! (x!)
DECLARE FUNCTION Yscr! (y!)
Pi! = 3.14159
DIM Method(1 TO 2) AS STRING
Method(1) = "параллельное": Method(2) = "перспективное"
'Максимальное число точек фигуры
DIM SHARED Nmax AS INTEGER: Nmax = 10
DIM Ncube AS INTEGER: Ncube = 8 'число вершин кубика
DIM PrScr!: PrScr! = 70  '% использования экрана
DIM SHARED a AS SINGLE: a = 5 'сторона кубика
AnglePro% = 30 'Угол проектирования для случая, когда проектирование параллельное
'вертикальная координата и расстояние от оси Z
'центра проектирования для случая, когда проектирование перспективное
Z0! = 16: B! = 25
Fi0% = 30 'Начальный угол плоскости проектирования
'координаты первой точки кубика
x1! = 1: y1! = 1: z1! = 0
'массив вершин кубика в пространстве
DIM aP3(1 TO Nmax) AS tPoint3
'относительные координаты точек пространственного кубика
DIM SHARED aPcube3(1 TO Nmax) AS tPoint3
'массив вершин кубика в картинной плоскости
DIM SHARED aP2(1 TO Nmax) AS tPoint
'массив вершин кубика на экране в центральных координатах
DIM SHARED aP(1 TO Nmax)  AS tPoint
'матрица связи вершин кубика
DIM SHARED CubeMatrix(1 TO Nmax, 1 TO Nmax) AS INTEGER
DIM SHARED W AS SINGLE, H AS SINGLE
W = 640: H = 480 'ширина и высота экрана_
DIM Fi AS INTEGER, M AS SINGLE, x20 AS SINGLE, y20 AS SINGLE
'Corner() — граничные точки: Corner(1) — левая, нижняя
'Corner(2) — правая, верхняя
DIM Corner(1 TO 2) AS tPoint
DIM Vpp AS tPoint3, Pc AS tPoint3
White% = 15: Black% = 0
CALL PictCube3
FOR j = 1 TO 2 'цикл методов
  PRINT "демонстрируем "; Method(j); " проектирование"
```



```
gd:=DETECT;
initgraph(gd,gm,Path);
WH(W,H);{находим ширину и высоту экрана}
planes(NF); {Изображение очередной поверхности}
readln;
closegraph;
                    end; {цикл поверхностей}
END.
```

Язык Си

```
/* Изображения трех пространственных поверхностей  с помощью  параллельного
 проектирования, не содержащие невидимых точек */
#include "grtool.h"
#define B 250 /* Половина габарита рабочей области */
#define Nmima 5001 /* 20*B+1 Размерность массивов Min,Max */
/* Вектор проектирования */
#define ax -0.25
#define ay -1
#define az -0.25
#define d1 5 /* Шаг плоскостей сечения */
int Min[Nmima],Max[Nmima];
/* Основная функция построения поверхности */
void Planes(int NF)
{float y,yb,xb,xe,dx,dy,z0,x0,y0;
 int N,xstart,xstop,i,iC,x,z,f;
 N=2*B/d1; /* число шагов цикла плоскостей сечения */
 /* инициализируем массивы Min,Max минимальных
  и максимальных аппликат картинной плоскости */
 for (i=0; i<Nmima; i++) {Min[i]=1000; Max[i]=-1000;}
 /* Находим начальную точку (xb,yb) на картинной плоскости,
    проектируя точку (-B,B,0)                              */
 xb=-B-(ax/ay)*B; yb=-(az/ay)*B;
 /* Находим координату xe конечной точки,
    проектируя точку (B,B,0)            */
 xe=B-(ax/ay)*B;
 /* Находим шаги по x и y в картинной плоскости */
 dx=d1*(ax/ay); dy=d1*(az/ay);
 for (i=0; i<=N; i++) /* Цикл плоскостей сечения */
  {/* Координата по y начальной точки на картинной плоскости
      для очередной плоскости сечения */
   y=yb+i*dy;
   /* координаты по x начальной и конечной точки в
      картинной плоскости для очередной плоскости сечения */
   xstart=round(xb+i*dx); xstop=round(xe+i*dx);
   for (x=xstart; x<=xstop; x++)/* цикл вдоль плоскости сечения*/
    {/* проектируя "назад", находим точку в основании
     плоскости сечения — на горизонтальной плоскости */
     x0=x-y*(ax/az); y0=-y*(ay/az);
     /* находим значение функции для x0,y0 */
     z0=func(x0,y0,NF);
     /* находим аппликату образа точки (x0,y0,z0)на картинной плоскости */
     z=round(z0-(az/ay)*y0);
     /* Выясняем, надо ли выводить найденную точку */
     f=0;
     iC=x+10*B;/* пересчет индекса */
     if (z<Min[iC]) {Min[iC]=z; f=1;}
     if (z>Max[iC]) {Max[iC]=z; f=1;}
     if (f) putpixel(Xscr(x),Yscr(z),WHITE);
    }                          /* цикл вдоль плоскости сечения*/
  }                   /* Цикл плоскостей сечения */
}
void main(void)
{int NF;
 for (NF=1; NF<=3; NF++) /* цикл поверхностей */
```



ных массивов. Их нахождение относится к этапу подготовительной, “ручной”, работы. Анализ показал, что если индекс меняется от $-10H$ до $10H$, то этого вполне достаточно. Второй и последний вопрос: каковы начальные значения наших массивов? Как обычно, максимуму надо дать заведомо малое значение, а минимуму — большое. В нашем случае подходящими значениям будут числа -1000 и 1000.

Описанный алгоритм носит название “метод минимакса”, он подробно описан в указанной выше статье, содержащейся в № 20, 22 “Информатики” за 1998 г.

Приводим программы построения поверхностей.

Язык Паскаль

```
{Изображения трех пространственных поверхностей с помощью параллельного
 проектирования,не содержащие невидимых точек }
uses graph,crt;
{$I grtool.pas}
Const
B=250;{Половина габарита рабочей области}
{Вектор проектирования}
ax=-0.25; ay=-1; az=-0.25;
d1=5;{Шаг плоскостей сечения}
var NF:byte;
Min,Max:array[-10*B..B*10] of integer;
Procedure Planes(NF:byte);{Основная функция построения поверхности}
Var
  y,yb,xb,xe,dx,dy,z0,x0,y0:Real;
  N,xstart,xstop,i,x,z:Integer;
  f:boolean;
Begin
 N:=2*B div d1;{число шагов цикла плоскостей сечения}
 {инициализируем массивы Min,Max минимальных
  и максимальных ординат картинной плоскости}
 for i:=-10*B to 10*B do
 begin Min[i]:=1000; Max[i]:=-1000 end;
 {Находим начальную точку (xb,yb) на картинной плоскости, проектируя точку (-B,B,0) }
 xb:=-B-(ax/ay)*B; yb:=-(az/ay)*B;
 {Находим координату xe конечной точки, проектируя точку (B,B,0) }
 xe:=B-(ax/ay)*B;
 {Находим шаги по x и y в картинной плоскости}
 dx:=d1*(ax/ay); dy:=d1*(az/ay);
 for i:=0 to N do begin {цикл плоскостей сечения}
  {Координата по y начальной точки на картинной плоскости
   для очередной плоскости сечения                        }
  y:=yb+i*dy;
  {координаты по x начальной и конечной точки в
   картинной плоскости для очередной плоскости сечения}
  xstart:=round(xb+i*dx); xstop:=round(xe+i*dx);
  for x:=xstart to xstop do begin{цикл вдоль плоскости сечения}
   {проектируя "назад", находим точку в основании
    плоскости сечения — на горизонтальной плоскости}
   x0:=x-y*(ax/az); y0:=-y*(ay/az);
   {находим значение функции для x0,y0}
   z0:=func(x0,y0,NF);
   {находим аппликату образа точки (x0,y0,z0)
    на картинной плоскости }
   z:=round(z0-(az/ay)*y0);
   {Выясняем, надо ли выводить найденную точку}
   f:=false;
   if z<Min[x] then begin Min[x]:=z; f:=true end;
   if z>Max[x] then begin Max[x]:=z; f:=true end;
   if f then putpixel(Xscr(x),Yscr(z),white);
                              end;{цикл вдоль плоскости сечения}
                    end; {Цикл плоскостей сечения}
end;{Planes}
BEGIN
for NF:=1 to 3 do begin {цикл поверхностей}
```



```
'Продолжение — нажатием клавиши Enter
DO: LOOP UNTIL INKEY$ = CHR$(13)
SCREEN 12 'Переход в графический режим для монитора VGA
'Находим координаты вершин кубика
aP3(1).x = x1!: aP3(1).y = y1!: aP3(1).z = z1!
FOR i = 1 TO Ncube
  aP3(i).x = aP3(1).x + aPcube3(i).x
  aP3(i).y = aP3(1).y + aPcube3(i).y
  aP3(i).z = aP3(1).z + aPcube3(i).z
NEXT i
Fi = Fi0% 'Начальный угол плоскости проектирования
DO        'Цикл демонстрации
  FiRad! = Fi * Pi! / 180
  'для случая, когда проектирование параллельное,
  'находим вектор проектирования Vpp
 IF j = 1 THEN
   AngleRad! = (AnglePro% + 180) * Pi! / 180
   Vpp.x = COS(AngleRad) * COS(FiRad)
   Vpp.y = COS(AngleRad) * SIN(FiRad)
   Vpp.z = SIN(AngleRad)
   'для случая, когда проектирование перспективное,
   'находим координаты X0, Y0 центра проектирования
 ELSE
   Pc.x = B! * COS(FiRad): Pc.y = B! * SIN(FiRad): Pc.z = Z0
END IF
'Находим координаты вершин кубика в картинной плоскости
FOR i = 1 TO Ncube
   IF j = 1 THEN
      CALL pp(aP3(i),Vpp,aP2(i)) 'проектирование параллельное
   ELSE
      CALL cp(aP3(i),Pc,aP2(i)) 'проектирование перспективное
   END IF
  NEXT i
  'Находим границы кубика в картинной плоскости
  Corner(1) = aP2(1): Corner(2) = aP2(1)
  FOR i = 2 TO Ncube
    CALL Verge(aP2(i), Corner())
  NEXT i
  'Находим масштаб и координаты центра картинной плоскости
  CALL ScaleC2(PrScr!, Corner(), M, x20, y20)
  'Находим центральные координаты вершин кубика на экране
  FOR i = 1 TO Ncube
    aP(i).x = (aP2(i).x — x20) * M
    aP(i).y = (aP2(i).y — y20) * M
  NEXT i
  'Рисуем кубик
  CALL DrawPicture(Ncube, aP(), CubeMatrix(), White%)
  Fi = Fi + Fi0% 'Наращиваем угол плоскости проектирования
  'Продолжение — нажатием клавиши Enter
  DO: LOOP UNTIL INKEY$ = CHR$(13)
  'Стираем кубик
  CALL DrawPicture(Ncube, aP(), CubeMatrix(), Black%)
  LOOP WHILE Fi < 180
  SCREEN 0 'Переход в текстовый режим
NEXT j     'цикл методов
END
```

Выполнив приведенные выше программы, можно увидеть, что изображение кубика содержит все его вершины и ребра, в том числе и невидимые. Для кубика это не является серьезным недостатком. Однако если изображение сложного пространственного объекта содержит невидимые линии, то в нем трудно разобраться и польза от такого изображения невелика.

Изображение пространственной поверхности, не содержащее невидимых точек

Задача 15. Изобразить поверхности, заданные уравнениями:

а) $z = 100 \times \sin(\pi / 50 \times \sqrt{x^2 + y^2})$;

б) $z = x_1 \times y_1 \times (x_1^2 - y_1^2) \times (\sqrt{x_1^2 + y_1^2} + 1)$,

где $x_1 = x/30$, $y_1 = y/30$;

в) $z = 100 \times \cos(x_1 \times y_1 \times \pi / 150)$,

где $x_1 = x/13$, $y_1 = y/13$.

Значения x, y удовлетворяют неравенствам

$$-H \le x \le H$$
$$-H \le y \le H$$

причем H=250. Использовать параллельное проектирование. Проекции вектора проектирования имеют значения:

$$a_x = -0{,}25; \qquad a_y = -1; \qquad a_z = -0{,}25$$

Изображение поверхности должно содержать только видимые точки.

Решение. Параметры задачи подобраны так, чтобы координаты картинной плоскости совпали с центральными координатами экрана. В общем случае этого, конечно, не будет. Используя подход, предложенный в предыдущей задаче, можно усовершенствовать приводимое ниже решение, чтобы оно годилось для любых параметров. Мы предлагаем осуществить это заинтересованному читателю.

Приведем сначала запись функций на используемых языках программирования.

Язык Паскаль

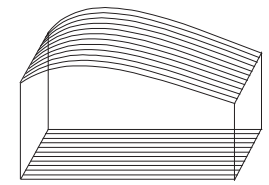
```
function func(x,y:real;i:byte):real;
begin
Case i of
 1: begin func:=100*sin(pi/50*sqrt(x*x+y*y)); end;
 2: begin
     x:=x/30; y:=y/30;
     func:=x*y*(x*x-y*y)/(sqrt(x*x+y*y)+1);
    end;
 3: begin
      x:=x/13; y:=y/13; func:=100*cos(x*y*Pi/150);
    end;
end;{Case}
end;
```

Язык Си

```
float func(float x,float y, int i)
{ switch(i)
   {case 1: return 100*sin(PI/50*sqrt(x*x+y*y));
    case 2:
    {x/=30; y/=30; return x*y*(x*x-y*y)/(sqrt(x*x+y*y)+1);}
    case 3: {x/=13; y/=13; return 100*cos(x*y*PI/150);}
   }
}
```

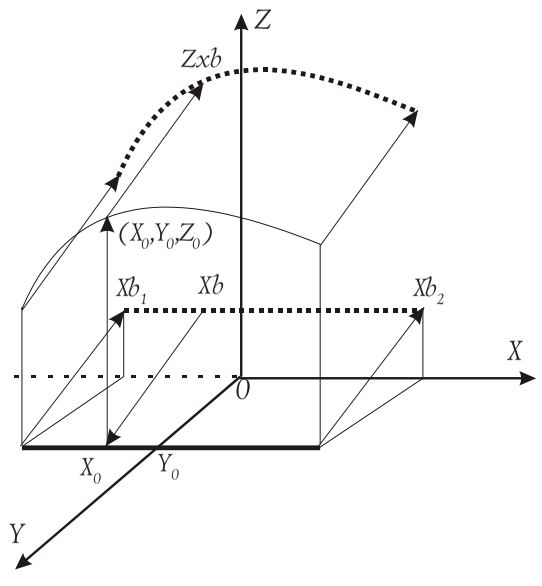

Язык Бейсик

```
FUNCTION func! (x AS SINGLE, y AS SINGLE, i AS INTEGER)
DIM x1 AS SINGLE, y1 AS SINGLE, R AS SINGLE, S AS SINGLE
Pi! = 3.14159
SELECT CASE i
CASE 1
  func! = 100 * SIN(Pi! / 50 * SQR(x * x + y * y))
CASE 2
  x1 = x / 30: y1 = y / 30:
  R = x1 * y1 * (x1 * x1 - y1 * y1)
  S = (SQR(x1 * x1 + y1 * y1) + 1)
  func! = R / S
CASE 3
   x1 = x / 13: y1 = y / 13:
   func! = 100 * COS(x1 * y1 * Pi! / 150)
  END SELECT
END FUNCTION
```



Решение данной задачи выполним следующим образом. Будем проводить плоскости, параллельные плоскости ZOX (т.е. картинной плоскости). Назовем их *плоскостями разреза*, а отрезок, лежащий на пересечении указанной плоскости с плоскостью XOY в заданном квадрате, — *основанием*. Расстояние от плоскостей разреза до картинной плоскости (т.е координату Y) меняем от H до $-H$ с достаточно малым шагом. Возьмем очередную плоскость разреза с координатой Y_0. Она пересекает нашу поверхность по линии, точки которой проектируются на картинную плоскость. При этом мы следим за тем, видима ли проектируемая точка, и если ее "заслоняют" предыдущие точки, то не выводим ее на экран. Делается это так. Проектируем основание очередной плоскости разреза на картинную плоскость. Получаем горизонтальный отрезок, координаты концов которого в картинной плоскости обозначим (Xb_1, Zb), (Xb_2, Zb). Заметим, что координаты в картинной плоскости мы считаем целыми числами, выполняя в необходимых случаях соответствующие округления. Теперь двигаемся вдоль полученного отрезка с единичным шагом по оси X. Пусть Xb — абсцисса текущей точки этого отрезка. Проектируя "назад", находим абсциссу X_0 прообраза текущей точки на основании плоскости разреза. Для этого через текущую точку проводим прямую по направлению вектора $\boldsymbol{a}$ до пересечения с плоскостью XOY. Выполнив несложные выкладки, получим:

$$X_0 = Xb - Zb a_x / a_z$$



Вычисляем значение функции для текущей точки на основании $Z_0 = F(X_0, Y_0)$. Проектируем точку (X_0, Y_0, Z_0) на картинную плоскость. Понятно, что абсцисса полученной точки есть Xb, а аппликату обозначим Zxb. Сейчас мы подошли к центральному моменту излагаемого алгоритма. Будем считать, что для каждого значения абсциссы X картинной плоскости известны максимальное и минимальное значения аппликат уже нарисованных точек с этой абсциссой. Обозначим их через $\mathrm{Max}[X]$, $\mathrm{Min}[X]$. Как видно из обозначений, указанные величины организованы как массивы. Теперь сравниваем Zxb с величинами $\mathrm{Max}[X]$, $\mathrm{Min}[X]$. Если $Zxb > \mathrm{Max}[X]$, то полагаем $\mathrm{Max}[X] = Zxb$ и выводим точку (Xb, Zxb) на экран. Если $Zxb < \mathrm{Min}[X]$, то действуем аналогичным образом: полагаем $\mathrm{Min}[X] = Zxb$ и также выводим указанную точку на экран. Наконец, в случае, когда Zxb лежит в интервале $(\mathrm{Max}[X], \mathrm{Min}[X])$, наша точка не видна и выводить ее на экран не следует. Чтобы закончить описание алгоритма, осталось выяснить два момента. Каковы размеры массивов $\mathrm{Max}[X]$, $\mathrm{Min}[X]$? Спроектируем вершины квадрата, ограничивающего значения переменных X, Y в плоскости XOY. Найдем максимальные и минимальные значения абсцисс этих точек и округлим их до ближайшего целого. Это и есть максимальное и минимальное значения индекса указан-

Интернет для начинающих

Окончание. См. с. 1, 2

Папа. Тем более что все инструкции в наших почтовых программах написаны на русском языке.

Вася. Этот русский больше походит на "тарабарский"!

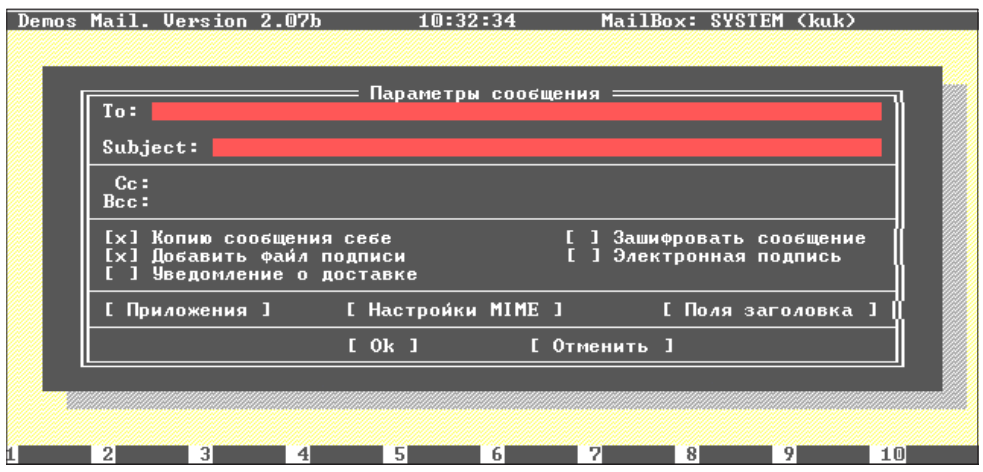
Папа. Если что-то в инструкции не очень понятно, можно поэкспериментировать: выполнить указанные действия и посмотреть, что получится.

Вася. Я хочу отправить Пауку такое письмо с приветом:

```
ЭТО Привет
Вася
м
Змей Горыныч
м
стремительный
Соловей-Разбойник
дурной
КОНЕЦ
```

Как мне воспользоваться адресной книгой для автоматической записи адреса на конверт?

Папа. Когда курсор расположен в поле **To:** на конверте



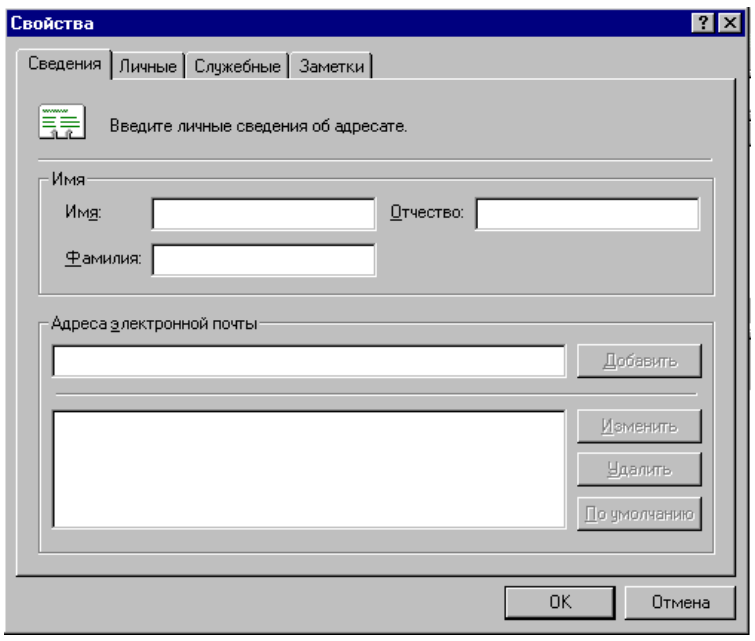
нужно нажать клавишу F2. На экране появится адресная книга.

Остается выбрать в ней нужную строку и нажать Enter.

Вася. Получилось! Длинный адрес Паука сам появился на конверте. В поле **Subject:** записываю слово "Hallo" и отсылаю письмо в Роботландию.

Папа. Давай теперь поработаем с почтовой программой Windows, Вася. Я вижу на инструментальной панели кнопку **Адресная книга**, щелкаю по ней мышкой. Появилось окно адресной книги, и в нем видна кнопка **Создать адрес**.

Вася. Нажимаю ее... Ого! Запись в адресной книге почтовой программы Windows очень обширна. В новом окне **Свойства** столько закладок и столько разных полей:



Папа. В этой адресной книге можно хранить очень подробную информацию о каждом твоем корреспонденте, но можно обойтись и минимальными сведениями, необходимыми для отправки электронной почты: они вводятся на закладке **Личные**.

Вася. Назначение некоторых полей мне не совсем ясно. Имя, отчество, фамилия — это понятно. А что означает запись **Вид**?

Папа. Сюда вписывается текст, который будет виден в кратком списке корреспондентов — своеобразном оглавлении адресной книги.

Вася. Я вижу, что для электронного адреса предусмотрено не одно поле, а целое окошко.

ТЕЛЕ-КОММУНИКАЦИИ

Папа. В него можно занести при необходимости несколько адресов, например, служебный и домашний. Кнопкой **По умолчанию** можно отметить тот, по которому обычно будут отправляться письма. Пользователь записывает адрес в поле **Добавить новый**, а затем нажимает кнопку **Добавить** и запись попадает в окошко с адресами.

Вася. Понятно. А как запомнить адрес Паучка, ничего не набирая в адресной книге?

Папа. Очень просто: при просмотре письма нужно щелкнуть правой кнопкой мыши в поле **От:** на конверте и в открывшемся списке выбрать команду **Добавить в адресную книгу**.

Можно настроить почтовую программу так, чтобы при ответе на сообщение адресат автоматически попадал в адресную книгу.

Вася. Это очень удобно! Теперь остается только понять, как использовать сохраненные адреса при отправке сообщений.

Папа. Посмотри внимательно на окошко, в котором создается сообщение:

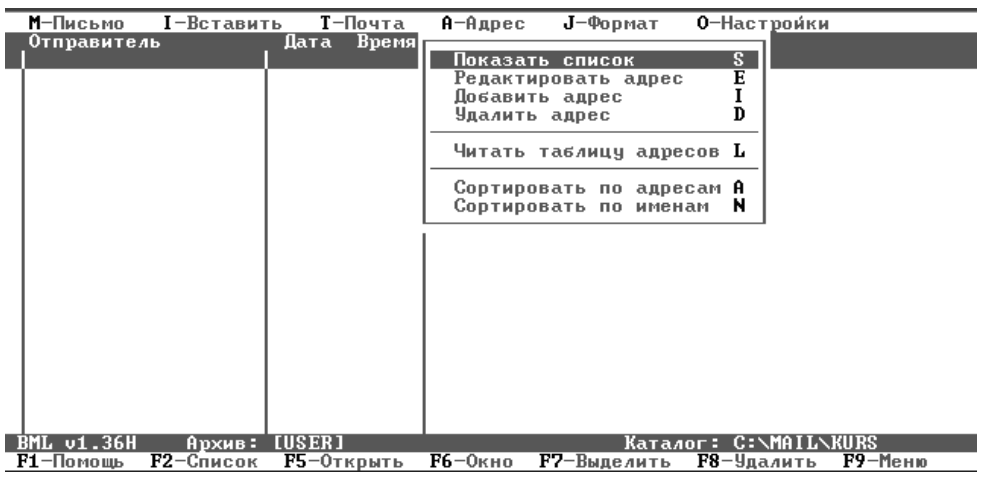
Перед каждым адресным полем **Кому:**, **Копия:**, **Слепая:** расположен значок с "оторванным" листочком. Щелчок мыши по этой пиктограмме и есть команда — показать оглавление адресной книги. В оглавлении отмечаются нужные строчки, и адреса автоматически вписываются в адресное поле конверта.

ДЛЯ УЧИТЕЛЯ (КОММЕНТАРИИ ДЛЯ СЕРЬЕЗНЫХ ЧИТАТЕЛЕЙ)

Адресная книга, обсуждаемая в разделе, является удобным средством, дополняющим все современные почтовые программы. В ней можно хранить не только электронные адреса корреспондентов для автоматизации заполнения соответствующих полей отправляемого сообщения, но и любую детальную информацию о человеке, с которым вы переписываетесь. Таким образом, адресная книга является примером базы данных, встроенной в почтовую программу. Как и в любой другой базе данных, в адресной книге можно выполнять такие операции, как поиск, сортировка, обновление, удаление и добавление записей. На базе адресной книги можно построить несколько полезных занятий по простейшим приемам работы с базами данных, но этот материал выходит за рамки нашего изложения.

Работа с адресной книгой в программе BML

Работа выполняется через позицию **A-Адрес** главного меню программы или при помощи "горячих" клавиш, указанных в меню и в нижней строке подсказки:



Для вызова записей на экран оглавления в адресной книге используется команда **Показать список**.

Команда **Добавить адрес** позволяет запомнить адрес корреспондента. На экране появляется окно диалога с полями **Адрес** и **Комментарий**, в которые следует ввести требуемые значения. Автоматически запомнить адрес в книге, считав его из пришедшего письма, программа BML (версии 1.36H) не в состоянии.

Чтобы вызвать оглавление книги при вводе адреса (в командах **Послать письмо**, **Переслать** и др.), следует, находясь в поле адреса, нажать клавишу F2 (или Enter).

Вопросы и упражнения

1. Для чего нужна адресная книга в почтовой программе?

Ответ. Адресная книга позволяет хранить информацию о корреспондентах электронной переписки. Адресную книгу можно использовать так же, как и обычный блокнот, но основное ее назначение — автоматизация набора электронного адреса на конверте отправляемого по сети письма.

2. Какая информация хранится в адресной книге?

Ответ. Каждая запись в адресной книге содержит имя корреспондента и его электронный адрес. Структура дополнительной информации, которую может содержать каждая запись, зависит от конкретной почтовой программы. Это может быть: обычный почтовый адрес, телефон, название организации, должность, другие сведения, произвольные пометки.

3. Опишите алгоритмы работы с адресной книгой вашей почтовой программы.

4. Исследуйте операции с адресной книгой, которые не рассматривались в разделе: удаление, редактирование, сортировка, поиск. В качестве дополнительной информации используйте встроенные справочники и подсказки, а также пользовательские инструкции, сопровождающие почтовые программы.

5. Занесите в адресную книгу своей почтовой программы информацию о роботландском Паучке и других постоянных корреспондентах.

6. Подготовьте и отправьте письмо-программу для робота Паучка с командой **Hallo** в поле **Тема** и содержанием, аналогичным содержанию письма Васи.

Продолжение следует

Информатика для гуманитариев

Информатика — один из фундаментальных предметов общеобразовательного цикла. Следовательно, существует проблема — как учить информатике гуманитариев? С просьбой указать методическую и учебную литературу к нам обращаются учителя. Учебное пособие для гуманитарных факультетов педагогических вузов (Екатеринбург, 1995 г.) С.А. Бешенкова, А.Г. Гейна, С.Г. Григорьева "Информатика и информационные технологии" оказалось единственным изданием, адресованным этому кругу читателей. По просьбе редакции книгу рецензирует *В.Р. Лещинер, кандидат пед. наук, доцент ИПК и ПРНО Московской обл.*

Прошло то время, когда абитуриент исторического или филологического факультета забывал про математику, физику и другие естественные науки сразу после окончания выпускных экзаменов в школе. Когда автор этих строк попытался в 1979 г. устроиться работать программистом в вычислительный центр педагогического института, где учился в тот момент на историческом факультете, проректор института отказывался верить в возможность такого сочетания профессий и отказывался подписать разрешение на совместительство. Сейчас даже первоклассник знает, что без компьютера не может обойтись ни один специалист, в том числе и самый далекий от техники гуманитарий. Поэтому появление учебника по информатике для студентов гуманитарных специальностей можно только приветствовать.

К сожалению, рецензенту неизвестны программа соответствующего курса и требования к уровню подготовки специалистов по данному предмету. Можно предполагать, что учебное пособие в целом соответствует программе. Тем не менее в подобного рода рецензии стоит отметить не только то, как то или иное положение изложено в книге, но и то, чего в учебном пособии нет, а должно быть. Начать, видимо, стоит с разбора существующего текста.

Пособие состоит из двух неравных частей. 27 параграфов первой части посвящены теоретическим вопросам информатики, 9 параграфов второй — "информационным технологиям". Такая диспропорция может быть объяснена отсутствием в гуманитарных университетах компьютерной техники, пригодной для организации практических занятий по использованию информационных технологий. Видимо, единственное программное средство, имеющееся в распоряжении, — интерпретатор ПРОЛОГА-Д, причем какой-то конкретной модификации. Как иначе объяснить то, что из 32 страниц главы, посвященной инструментальным средствам, ПРОЛОГУ отводится 27 страниц. Вряд ли это инструментальное средство используется гуманитариями чаще, чем редактор текстов, описанию возможностей которого в пособии отводится ровно один абзац, хотя и имеется ссылка на возможную лабораторную работу. Параграф 30 состоит из трех коротеньких абзацев и сводится к сомнительному определению "Интеллектуальное обеспечение [компьютера] можно определить как совокупность интеллектуальных методов, приемов, технологий, обеспечивающих решение задач из данной предметной области при помощи компьютера". Если учесть, что соседний 33-й параграф занимает 8,5 страницы, связь между структурой курса и структурой учебника становится не совсем очевидной. Короче, трудно поверить, что найдется хотя бы один студент, способный освоить информационные технологии при помощи данного пособия.

Первая часть пособия также не отличается внятностью и стройностью изложения. Помимо откровенно ошибочных суждений, вроде существовавшего в Иудее "минората" или не совсем адекватного изложения космологии Данте (круги ада не расположены непосредственно под раем, кроме того, существует и чистилище, которое, если прочесть текст буквально, расположено ниже ада), многое в учебном пособии кажется не совсем уместным или неточно изложенным. Не следует забывать, что учебник ориентирован на студентов гуманитарных специальностей, которые если не читали, то хотя бы слышали на лекциях изложение учений Платона, Аристотеля, Канта и Гегеля и понимают, как различаются родовые и видовые понятия, как связано между собой общее и частное, в чем состоит процесс познания и так далее. Невнятица начинается уже в первом параграфе, где сделана попытка объяснить различие между обозначением группы предметов и отдельным предметом на основе различения неопределенного и определенного артиклей. В связи с этим можно заметить, что отсутствие в русском языке артиклей не означает невозможности определить по контексту, идет ли речь о конкретном предмете или о множестве предметов. В то же время даже пятикласснику понятно, что в фразе "*I have a book*" речь идет об одной книге, а не о множестве. Вообще это ли тема для первого (!) параграфа учебника информатики? Куда как лучше было бы посвятить его, как обычно, объекту, предмету и методам базовой науки.

Во втором параграфе содержатся противоречивые определения грамматики и языка, при попытке применения которых к естественному языку внимательный читатель сразу проникнется недоумением. Это недоумение не покинет его и при дальнейшем чтении книги. Так, на с. 6 имеется упражнение 3:

Роботу дана команда на русском языке: "взять большой красный шар". Опишите возможные действия робота.

Я начал придумывать обоснования вложения циклов — сначала распознать шар, потом отобрать красный, потом проранжировать их по диаметру и найти наибольший, — а выяснилось, что робот "может действовать двояким образом: взять красный шар или начать перебирать все возможные "красные шары" (с. 63).

Нетрудно представить, каково должно быть недоумение преподавателя, которому поручено провести занятие по параграфу 16 "Еще раз об истине". Привычка формулировать триединую цель урока (совокупность образовательных, воспитательных и развивающих задач) может сыграть с учителем злую шутку: как можно планировать цели образования и развития при изучении текста, к которому даже авторы не смогли придумать упражнений?

Хватит вести речь о существующем тексте, важнее сказать о том, чего нет в этой книге. Нет того, что действительно необходимо знать гуманитарию из области информатики и информационных технологий. Это прежде всего понятие кода, кодовой таблицы, алгоритмов кодирования и преобразования текстов, написанных на естественном языке, и графических образов, а отсюда ключевая для изучения прикладных пакетов тема форматов файлов и совместимости данных. Близко примыкает к данной теме группа вопросов, связанных с хранением информации, алгоритмами архивации, потерями и искажениями данных при передаче информации и протоколами обмена информацией. (Кстати, в параграфе 18, посвященном телекоммуникациям, есть термины "телематика", "телетекст" и "видеотекст", но нет понятий "протокол" и "электронный адрес".) Важным с онтологической точки зрения, хотя и сложным для понимания, является разъяснение в учебнике архитектуры сети Интернет и общих принципов ее функционирования. Полезным является изложение общих принципов построения иерархических, реляционных и сетевых информационно-справочных систем с разбором реальных примеров. Все перечисленное выше составляет предмет науки информатики и может быть изложено чисто теоретически. Не уверен, что такой курс вызовет в силу своей сложности и сухости большой интерес у студентов-гуманитариев, но по крайней мере они отнесутся к нему с уважением.

При написании практической части курса, посвященной собственно информационным технологиям, у авторов возникнет естественная трудность, связанная с быстрым старением конкретных прикладных пакетов. Выходом из положения может быть структурирование курса на основе конкретных задач, решаемых с помощью компьютеров. В их числе форматирование текстов, вставка и замена с использованием буфера обмена, контекстный поиск и замена, обработка изображений, распознавание текста, связь и внедрение объектов, автоматизация расчетов, проектирование и моделирование сложных систем и так далее. Учебник в этой ситуации может содержать только теоретический материал, не привязанный к конкретному программному пакету, а реальное овладение изучаемой технологией будет осуществляться студентами в ходе лабораторных и практических занятий.

Хотелось бы видеть в учебнике информатики для гуманитариев также и третий раздел, посвященный использованию информационных технологий и количественных методов собственно в работе специалистов — историков, лингвистов и литературоведов. Современный специалист-гуманитарий должен знать основы математической статистики и теории вероятностей, иметь представление о способах вычисления средних величин, коэффициентов корреляции, оценке распределения и многом другом. В практику исследователей давно вошли построение математических моделей статистических процессов, контент-анализ литературных текстов и исторических источников, создание и использование реляционных баз данных и многое другое. Для всех этих методов требуется профессиональное владение компьютером и хорошее знание математики.

В заключение хочется сделать вывод, что при всей актуальности и практической значимости разработки курса информатики для студентов гуманитарных специальностей рецензируемое учебное пособие в силу низкого качества может сыграть лишь негативную роль, дискредитируя самое идею такого курса.

Гл. редактор
Е.Б.Докшицкая
Зам. гл. редактора
С.Л.Островский
Редакция:
Л.Н.Картвелишвили,
Ю.А.Соколинский,
Н.Л.Беленькая,
Н.П.Медведева
Дизайн и компьютерная верстка:
Н.И.Пронская
Корректоры:
Е.Л.Володина,
С.М.Подберезина

Отпечатано с готовых диапозитивов редакции в типографии "ПРЕССА", 125865, Москва, ул. Правды, 24

Тираж 7000 экз.
Заказ №