РАЗВИТИЕ ИНТЕЛЛЕКТА ШКОЛЬНИКОВ

АЛГОРИТМЫ КОМПЬЮТЕРНОЙ АРИФМЕТИКИ

Издательство БИНОМ

РАЗВИТИЕ ИНТЕЛЛЕКТА ШКОЛЬНИКОВ

АЛГОРИТМЫ КОМПЬЮТЕРНОЙ АРИФМЕТИКИ

Москва
БИНОМ. Лаборатория знаний

УДК 519.85(023)
ББК 22.18
О-52

Серия основана в 2008 г.

Окулов С. М.

О-52 Алгоритмы компьютерной арифметики / С. М. Окулов, А. В. Лялин, О. А. Пестов, Е. В. Разова. — М. : БИНОМ. Лаборатория знаний, 2014. — 285 с. : ил. — (Развитие интеллекта школьников).

ISBN 978-5-9963-1549-9

В книге речь идет о традиционных алгоритмах, которые кажутся очевидными, — об алгоритмах выполнения арифметических операций: о том, сколько тайного смысла и усилий интеллекта многих специалистов по информатике заложено в эти алгоритмы. Материал книги формирует содержательную основу деятельностного изучения алгоритмов компьютерной арифметики, чему способствует стиль изложения, синтезирующий в себе и математический материал, и формализованную запись логики работы компьютера.

Для школьников, преподавателей информатики и студентов информационно-технологических специальностей.

УДК 519.85(023)
ББК 22.18

16+

Учебное издание

Серия: «Развитие интеллекта школьников»

Окулов Станислав Михайлович
Лялин Андрей Васильевич
Пестов Олег Александрович
Разова Елена Владимировна

АЛГОРИТМЫ КОМПЬЮТЕРНОЙ АРИФМЕТИКИ

Ведущий редактор *Д. Ю. Усенков*
Ведущий методист *И. Л. Сретенская*. Художник *Н. А. Новак*
Технический редактор *Е. В. Денюкова*
Корректор *Е. Н. Клитина*
Компьютерная верстка: *Е. А. Голубова*

Подписано в печать 17.12.13. Формат 60×90/16.
Усл. печ. л. 18,00. Тираж 1000 экз. Заказ

Издательство «БИНОМ. Лаборатория знаний»
125167, Москва, проезд Аэропорта, д. 3
Телефон: (499) 157-5272, e-mail: binom@Lbz.ru
http://www.Lbz.ru, http://e-umk.Lbz.ru, http://metodist.Lbz.ru

ISBN 978-5-9963-1549-9

Содержание

Введение

> Обучение программированию, конечно, включает обучение фактам — о системах, машинах, языках программирования и т. д. Все это очень легко сделать явным. Но неприятность состоит в том, что эти факты составляют всего лишь *десять* процентов. То, что должно преподаваться в оставшиеся *девяносто* процентов, — это как решать задачи, как избежать непреодолимых трудностей. Короче, это обучение умению мыслить, не больше и не меньше.
>
> *Эдсгер Дейкстра,*
> *«Ремесленник или ученый?»*

Дано число:

$$m = 114381625757888867669325779976146612010218296721242362562561842935706935245733897830597123563958705058989075147599290026879543541.$$

Оно состоит из 129 десятичных цифр. С помощью этого числа Р. Ривест, А. Шамир и Л. Адлеман в 1978 г. зашифровали определенный текст с помощью широко известного алгоритма *RSA*. Алгоритм расшифровки тоже известен: чтобы получить исходный текст, число m надо разложить на произведение множителей. Но лишь через 17 лет (в 1994 г.) эта задача была решена — за 220 дней на 1600 компьютерах[1)].

Ваш ноутбук (например, с тактовой частотой 1,67 ГГц) выполняет одно умножение примерно за $t = 3{,}5 \cdot 10^{-9}$ с. Количество чисел, которые можно представить 129 десятичными цифрами, равно 10^{129}. Чтобы решить вышеупомянутую задачу самым простым способом, надо для половины этих 10^{129} чи-

---

1) *Ященко В. В.* Введение в криптографию. — М.: МЦНМО: «ЧеРо», 2000. — С. 90.

сел проверить, являются ли они делителями числа m. Пусть время выполнения операции деления совпадает с временем выполнения операции умножения. Тогда требуется выполнить порядка $10^{128}/2$ умножений (будем считать, что первая цифра известна). Это потребует $3{,}5 \cdot 10^{-9} \cdot 10^{128}/2 = 1{,}75 \cdot 10^{119}$ с. Один год — это приблизительно $3{,}155 \cdot 10^{7}$ с. Тогда для решения этой задачки на вашем ноутбуке потребуется ни много ни мало $0{,}55 \cdot 10^{112}$ лет (небольшая погрешность в вычислении количества лет уже не принципиальна).

Описанный выше пример — один из самых ярких, но далеко не единственный. Количество проблем, нерешаемых с помощью классического варианта компьютера (настольного или ноутбука), по-прежнему не уменьшается несмотря на рост быстродействия современных процессоров. Требуется всё большая производительность компьютеров.

В информатике можно выделить по крайней мере два направления в достижении этой цели. Первый, в самой общей формулировке, заключается в повышении производительности компьютера на аппаратном уровне. Второй — это поиск на алгоритмическом уровне способов ускорения выполнения как традиционных арифметических операций, так и методов решения задач.

В этой книге (а она в первую очередь предназначена для школьников и учителей информатики) арифметические операции рассматриваются на алгоритмическом уровне. Эти знания представляют собой необходимый компонент образованности человека, выбравшего информатику в качестве области своей профессиональной деятельности.

Второе направление в достижении вышеупомянутой цели, вероятно, самое эффективное, заключается в создании новых алгоритмов. Но универсальных алгоритмов не существует! Каждая конкретная область обработки данных с помощью компьютера, например обработка строк, требует своих открытий и изобретений. И эти открытия способен сделать только человек, имеющий алгоритмическую культуру, на формирование которой направлен в том числе и материал этой книги, включающий кроме методов реализации арифметических операций рассмотрение таких фундаментальных алгоритмов, как бинарный алгоритм возведения в степень, алгоритм Евклида, китайский алгоритм об остатках и алгоритм быстрого преобразования Фурье. Без знания последних невозможно понять

многие достижения современной информатики, например в криптографии, обработке видеоинформации и т. д.

Заметим, что задачи, связанные с эффективным выполнением арифметических операций и обработкой больших чисел, нередко появляются в олимпиадной информатике и являются одним из обязательных разделов подготовки школьника к олимпиадам по программированию. Однако систематическое изучение этого раздела в рамках элективных курсов для физико-математического, информационно-технологического и естественнонаучного профилей обычно отсутствует.

Основные алгоритмы входят в примерную программу по олимпиадной информатике под названием «числовых алгоритмов»[2]. К обязательным для изучения при этом относятся: алгоритм Евклида, методы решения линейных сравнений и т. д. Эти алгоритмы считают дидактическими единицами, «изучение которых формирует у школьников ключевые умения в области олимпиадной подготовки, открывает перед участником олимпиадного состязания возможность проявить свой творческий потенциал на достойном уровне ... — дипломов победителей и призеров заключительных этапов Всероссийской олимпиады школьников»[3].

В этой книге синтезируются два подхода в изложении материала, а именно математический и компьютерный. Подобный стиль изложения подставляет авторов под огонь критики как с той, так и с другой стороны. Но авторы не претендуют на некую абсолютизацию своего подхода в изучении данного раздела алгоритмистики, а предлагают рассматривать его как один из возможных вариантов.

При написании книг по компьютерным алгоритмам и при преподавании этого раздела информатики нередко приходится сталкиваться с удивительным фактом. Что бы автор ни писал, что бы ни делал, достаточно заглянуть в монументальный труд Дональда Кнута[4], и практически всегда оказывается, что он так или иначе уже описал и рассмотрел этот вопрос. Конечно, книги Д. Кнута написаны непростым языком и работать с ними не просто. Однако, так или иначе, вдохновителем мно-

---

[2] *Кирюхин В. М.* Информатика: всероссийские олимпиады. — М.: Просвещение, 2008. — С. 71.

[3] Там же — С. 67.

[4] *Кнут Д.* Искусство программирования для ЭВМ. В 3 тт. — М.: Мир, 1976, 1977, 1978. (Всего автором задумано семь томов, но остальные четыре тома еще не изданы. — *Прим. авт.*)

гих страниц данной книги (а она отражает реальное преподавание!) является именно этот труд Д. Кнута (даже при отсутствии ссылок на него).

Настольными и даже, если можно так выразиться, «священными» книгами для программиста являются также труды Никлауса Вирта[5)] и Витольда Липского[6)]. Их влияние на материал данной книги не так заметно, но оно есть — если не прямое, то косвенное. Оно заметно и в стиле изложения, и в синтезе математических фактов с реально работающими алгоритмами (при их проверке на компьютере). Последнее позволяет строить изучение материала не на уровне фактографического «вливания в сосуд[7)]», а путем экспериментального исследования этих фактов. Этим достигается самостоятельное осознание материала учащимся (школьником или студентом младших курсов), что требует от учащегося «пребывания в состоянии мысли». А это очень не просто — пребывать в состоянии мысли, думать, — это требует значительных усилий от человека (попробуйте, например, в течение часа осмысливать некую проблему, не связанную с бытовой прагматикой!), и в этом случае компьютер снова становится нашим помощником.

Эта книга — коллективный труд. Каждый автор внес посильную лепту в ее создание. Но мы обязаны сказать слова благодарности и всем школьникам и студентам за их критическое отношение к изложению материала, за многочисленные замечания и вопросы. Также и наши коллеги по факультету информатики, математики и физики Вятского государственного гуманитарного университета оказали нам незаменимую поддержку даже в том, что слегка разгружали нас от рутинной работы во время написания книги, не говоря уже о сделанных ими ценных замечаниях и предложениях.

---

5) *Вирт Н.* Алгоритмы + Структуры данных = Программы. — М.: Мир, 1985.

6) *Липский В.* Комбинаторика для программистов. — М.: Мир, 1988.

7) Аналогия с мыслью К. Поппера, когда он рассуждает о традиционной педагогике и сравнивает ее с механизмом «вливания в голову» учеников некой суммы фактов.

Часть 1

Компьютерная арифметика

Программист должен обладать способностью первоклассного математика к абстракции и логическому мышлению в сочетании с эдисоновским талантом сооружать все, что угодно, из нуля и единиц.

А. П. Ершов, «О человеческом и эстетическом факторах в программировании»

1.1. Алгоритмы целочисленной арифметики

Любая гениальная идея, в сущности, очень проста.

К. Чапек

Примечание. В этом параграфе рассмотрена программная реализация особенностей выполнения арифметических операций с целыми числами в компьютере. Вопросы аппаратной (схемной) реализации данных операций (а они достаточно интересны) выходят за рамки данной книги.

Известно, что компьютер обрабатывает данные, представленные в двоичной системе счисления[8]. В данном параграфе рассматриваются алгоритмы выполнения операций сложения, вычитания, умножения и деления целых неотрицательных чисел.

Однако прежде чем начать работу, обратим внимание на два удивительных факта.

При реализации операций сложения, вычитания, умножения и деления неотрицательных целых чисел мы будем ис-

[8] Считаем, что с темой «системы счисления» читатель знаком. Если нет, то она прекрасно изложена в книге: *Андреева Е. В., Босова Л. Л., Фалина И. Н.* Математические основы информатики. 2-е изд. — М.: БИНОМ. Лаборатория знаний, 2007.

пользовать только элементарные действия. Предположим, что компьютер может выполнять только сложение двух одноразрядных (однобитовых) чисел, операции сдвигов влево и вправо и логические операции, — другими словами, некое заданное множество примитивных действий[9]. Естественно, что без стандартных алгоритмических конструкций (следования, ветвления и повторения) сделать ничего нельзя. Но оказывается, что этих действий достаточно для решения задачи!

Итак, арифметику неотрицательных целых чисел мы конструируем из элементарных действий. Но если есть эта арифметика, то можно продолжить процесс и создавать более сложные конструкции, которые опять же будут построены из этих простых «кирпичиков». Удивительно, но вся сложная система под названием «компьютер» со всеми ее возможностями может быть создана только из этих элементарных действий.

Компьютерная арифметика — это арифметика конечных (ограниченных) чисел. В одном разряде можно хранить только два различных значения, в двух — четыре, в n разрядах — 2^n. Например, при хранении целых неотрицательных чисел в n разрядах возможно представить только числа от 0 до 2^n-1. То есть число 2^n — это нуль с точки зрения компьютера. Число 2^n в двоичном представлении — это единственный единичный $(n+1)$-й бит, а все остальные биты нулевые. В результате мы получаем n нулевых разрядов. Другими словами, число 2^n — это нуль в компьютерном представлении, или 2^n *сравнимо* с нулем ($2^n \equiv 0$).

А теперь мысленно представим себе то множество проблем, которое решается с помощью компьютера. И все эти проблемы спроецированы, если можно так выразиться, в конечную область значений. Другими словами, «мир компьютера» — это конечный мир.

Вспомогательные инструменты

Для наглядности изложения необходимо определить количество разрядов в представлении неотрицательных целых чисел. Будем считать, что оно равно 16 (двум байтам). В среде программирования Паскаль это тип данных *Word*. Определим в разделе констант выбранное значение:

```
Const Nmax=16;
```

[9] Это не совсем точное предположение. В реальном компьютере множество примитивов еще меньше, но в качестве допущения оно может быть принято.

В процессе работы необходимо реализовать вывод числа в двоичном виде. Эту задачу решает процедура Print:

```
Procedure Print(s:Word);
  Var t:String;
      j:Word;
  Begin
   t:=''; {Строка результата}
   j:=Nmax;
   While j>0 Do Begin
   {Выделяем младший разряд числа
    и анализируем его значение}
    If (s And 1)=1 Then t:='1'+t
     Else t:='0'+t;
    s:=s ShR 1; {Сдвигаем число}
    j:=j-1;
   End;
   WriteLn(t);
  End;
```

Примечание. Можно было бы использовать конструкцию:

```
While s<>0 Do …,
```

но в этом случае длины строк будут различны и нет требуемой наглядности в представлении результата.

При наличии процедуры Print основная программа для анализа операции сложения двух целых неотрицательных чисел (u, v) записывается так:

```
Begin
 ReadLn(u,v);
 Print(u);
 Print(v);
 Add(u,v,w); {Операция сложения}
 Print(w);
End.
```

Неотрицательные целые числа u и v в двоичном представлении имеют вид $u_{15}u_{14}...u_1u_0$ и $v_{15}v_{14}...v_1v_0$. Разряды пронумерованы справа налево, как это принято для двоичных чисел.

В ряде случаев требуется знать точное количество задействованных разрядов в представлении числа (например, чтобы отбросить старшие нулевые разряды). Эти значения фиксиру-

ются с помощью функции Nbit в глобальных переменных n и m:

```
Function Nbit(a:Word):Word;
  Var cnt:Word;
  Begin
   cnt:=0;
   While a>0 Do Begin
     cnt:=cnt+1;
     a:=a ShR 1;
    End;
   Nbit:=cnt;
  End;
```

Тогда основная программа, например для анализа операции деления, имеет вид:

```
Begin
 ReadLn(u,v);
 n:=Nbit(u);
 m:=Nbit(v);
 Divs(u,v,w,r);
 {Операция деления, где w — целая часть
  частного, а r — остаток}
 Print(w);
 Print(r);
End.
```

Сложение неотрицательных целых чисел

Даны два неотрицательных целых числа $u = u_{15}u_{14}...u_1u_0$ и $v = v_{15}v_{14}...v_1v_0$. Требуется найти их сумму — число $r = r_{15}r_{14}...r_1r_0$.

На рис. 1.1 приведен пример сложения 8-разрядных двоичных чисел $u = 182$ и $v = 43$.

```
  7 6 5 4 3 2 1 0   Номер разряда
  1 0 1 1 0 1 1 0   u
+     1 0 1 0 1 1   v
-----------------
  1 1 1 0 0 0 0 1
```

Рис. 1.1. Сложение чисел

Фактически требуется реализовать логику сложения одноразрядных чисел с учетом переноса единицы из младшего в старший разряд (переменная k). Эта логика представлена в табл. 1.1, где k' — новое (пересчитанное) значение k.

Таблица 1.1

u_i	v_i	k	r_i	k'
0	0	0	0	0
0	1	0	1	0
1	0	0	1	0
1	1	0	0	1
0	0	1	1	0
1	0	1	0	1
0	1	1	0	1
1	1	1	1	1

Один из возможных способов формализованной записи логики сложения двух неотрицательных целых чисел может иметь вид:

```
Procedure Add(u,v:Word;Var r:Word);
  Var j,k,a,b:Word;
  Begin
   j:=0;
   k:=0; {Разряд переноса}
   r:=0;
   While j<Nmax Do Begin
    a:=u And 1;
    {Выделяем первые разряды чисел}
    b:=v And 1;
    r:=r Or ((a+b+k) And 1) ShL j;
    {Прибавляем одноразрядное число
     к результату}
    k:=((a+b+k)ShR 1) And 1;
    {Выделяем разряд переноса}
    u:=u ShR 1; {Сдвигаем числа}
    v:=v ShR 1;
    j:=j+1;
   End;
  End;
```

Методическое отступление. Фактически у нас уже есть работоспособная программа. Осталось провести эксперименты и, в частности, убедиться, что 2^{16} (число 65 536) — это нуль в компьютерном представлении ($2^{16} \equiv 0$). Для этого следует запустить решение на простых тестах типа $u = 65\,535$, $v = 1$ ($r = 0$,

а не 65 536) или $u = 65\,535$, $v = 1024$ ($r = 1023$, а не 66 559). Фактически целые неотрицательные числа, для представления которых выделено 16 разрядов, можно рассматривать как числа на круге (рис. 1.2): после максимального числа идет вновь минимальное.

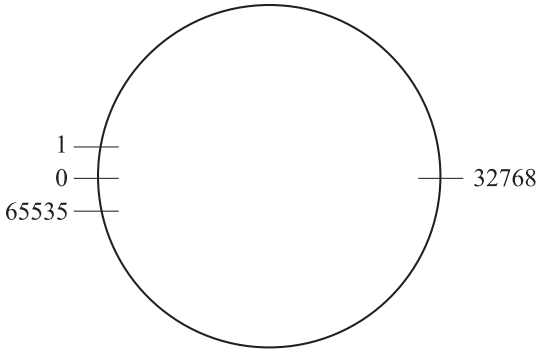


Рис. 1.2. Представление 16-разрядных целых неотрицательных чисел

В криптографии и ряде других областей возникает необходимость работы с числами, занимающими в памяти 1024 разряда и более (назовем такие числа «большими»). Обобщается ли полученный результат для данного случая?

Пусть требуется сложить два 64-разрядных числа. Это четыре слова по 16 бит.

Примечание. Диапазон целых чисел (знаковых) при 64 разрядах не очень велик — от –9 223 372 036 854 775 808 до 9 223 372 036 854 775 807. Указанное значение взято для простоты изложения логики.

Естественным выглядит следующее описание данных.

```
Const Tmax=4;
Type Nb=Array[1..Tmax] Of Word;
Var u,v,r:Nb;
```

При сложении 1024-битовых чисел изменяется только значение константы *Tmax* и, соответственно, размер массива для хранения величин.

Пример. На рис. 1.3 показан пример сложения двух 64-битовых чисел (числа представлены «столбиком», но корректнее было бы записать их в виде одной строки).

```
1111111111111111   0000000000000001   0000000000000000
1111111111111111 + 0000000000000000 = 0000000000000000
1111111111111111   0000000000000000   0000000000000000
0111111111111111   0000000000000000   1000000000000000
```

Рис. 1.3. Пример сложения двух 64-битовых чисел

Нетрудно видеть, что в этом случае следует предусмотреть передачу бита переноса между словами. Процедура Add при этом претерпит несущественные изменения: переменную k следует исключить из локальных переменных и внести в передаваемые параметры:

```
Procedure Add(u,v:Word;Var r,k:Word);
```

Основная же процедура сложения больших чисел будет выглядеть так:

```
Procedure AddBig(Const u,v:Nb;Var r:Nb);
  Var i,t:Word;
  Begin
   i:=1;
   t:=0;
   While i<=Tmax Do Begin
    Add(u[i],v[i],r[i],t);
    i:=i+1;
   End;
  End;
```

Примечание. Для работы с большими числами необходимо разработать процедуру их ввода и модифицировать процедуру Print.

Вычитание неотрицательных целых чисел

Даны два неотрицательных целых числа $u = u_{15}u_{14}...u_1u_0$ и $v = v_{15}v_{14}...v_2v_0$. Считаем, что $u \geq v$. Требуется найти их разность — число $r = r_{15}r_{14}...r_1r_0$. На рис. 1.4 приведен пример вычитания для $u = 103$, $v = 45$[10].

```
  6 5 4 3 2 1 0   Номер разряда
- 1 1 0 0 1 1 1   u
    1 0 1 1 0 1   v
  1 1 1 0 1 0
```

Рис. 1.4. Вычитание чисел

[10] Мы пока используем операцию вычитания для одноразрядных чисел. Как «избавиться» от нее и использовать только три простые операции (сложение одноразрядных чисел, сдвиг и инверсию), будет сказано позже.

Начнем рассмотрение этой операции с ошибочного варианта реализации логики. Напишем по аналогии с процедурой Add процедуру Sub_Bad, заменив сложение одноразрядных чисел их разностью:

```
Procedure Sub_Bad(u,v:Word;Var r:Word);
  Var j,k,a,b:Word;
  Begin
   j:=0;
   k:=0;
  {Бит заимствования единицы
   из старшего разряда}
   r:=0;
   While j<Nmax Do Begin
    a:=u And 1;
    {Выделяем первые разряды чисел}
    b:=v And 1;
    r:=r Or ((a-b+k) And 1) ShL j;
    {Вычитаем  одноразрядные числа,
     прибавляем разряд заимствования
     и добавляем к результату}
    k:=((a-b+k)ShR 1) And 1;
    {Формируем бит заимствования}
    u:=u ShR 1; {Сдвигаем числа}
    v:=v ShR 1;
    j:=j+1;
   End;
  End;
```

Эксперименты с решением показывают, что заимствование единицы из старших разрядов выполняется не так, как требуется. Например, операция 103 – 45 дает ответ 90. Причина в том, что в ряде случаев величина $a - b + k$ равна 65 535 (из нуля вычитается единица).

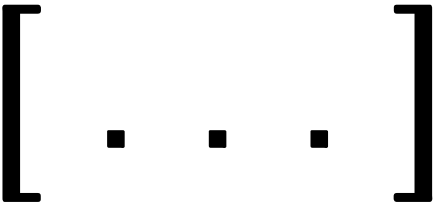
Проведем анализ операции вычитания (см. табл. 1.2, где k' — измененное значение разряда заимствования). Видим, что в пяти случаях из восьми приведенная выше по тексту реализация операции вычитания работает неверно.

Таблица 1.2

a	*b*	*k*	*r*	*k'*
0	0	0	0	0
0	1	0	1	1
1	0	0	1	0
1	1	0	0	0
0	0	1	1	1
0	1	1	0	1
1	0	1	0	0
1	1	1	1	1

Формализованная запись логики вычитания чисел имеет вид:

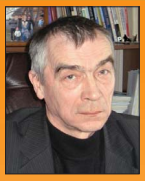
```
 Procedure Sub(u,v:Word;Var r:Word);
  Var j,k,a,b,t:Word;
  Begin
   j:=0;
   k:=0;
   r:=0;
   While j<Nmax Do Begin
    a:=u And 1;
    b:=v And 1;
    If (k=0) And (a=0) And (b=1)
     Then Begin {Вторая строка табл. 1.2}
           k:=1;
           r:=r+1 ShL j;
          End
     Else Begin
           r:=r Or ((a-b+k) And 1) ShL (j-1);
           If (k=1) And (a=1) And (b=0)
            Then k:=0;
{Единственный случай, когда значение k
 изменяется с единицы на нуль}
          End;
    u:=u ShR 1;
    v:=v ShR 1;
    j:=j+1;
   End;
  End;
```



РАЗВИТИЕ ИНТЕЛЛЕКТА ШКОЛЬНИКОВ

В книге речь идет о традиционных алгоритмах, которые кажутся очевидными, – об алгоритмах выполнения арифметических операций: о том, сколько тайного смысла и усилий интеллекта многих специалистов по информатике заложено в эти алгоритмы. Материал книги формирует содержательную основу деятельностного изучения алгоритмов компьютерной арифметики, чему способствует стиль изложения, синтезирующий в себе и математический материал, и формализованную запись логики работы компьютера.

Для школьников, преподавателей информатики и студентов информационно-технологических специальностей.



Станислав Михайлович ОКУЛОВ

Кандидат технических наук, доктор педагогических наук, профессор факультета информатики Вятского государственного гуманитарного университета, с 2001 г. почетный работник высшего профессионального образования РФ. Автор 9 изобретений по элементам ассоциативных вычислительных структур и автор (соавтор) более 15 книг по информатике для школьников и студентов.

Андрей Васильевич ЛЯЛИН

Ассистент кафедры прикладной математики и информатики Вятского государственного гуманитарного университета.

Олег Александрович ПЕСТОВ

Ассистент кафедры прикладной математики и информатики Вятского государственного гуманитарного университета. Преподаватель Кировского физико-математического лицея, летней компьютерной школы программистов.

Елена Владимировна РАЗОВА

Кандидат педагогических наук, доцент, заведующая кафедрой прикладной математики и информатики Вятского государственного гуманитарного университета.