«Мир информатики»

Журнал для тех, у кого информатика – любимый школьный предмет

Выпуск № 102, август 2025 г.

Спасибо вам, уважаемые коллеги!

За активную работу с учащимися по материалам интернет-журнала “Мир информатики” редакция журнала “Информатика в школе” выражает благодарность:

— Бойко Ирине Николаевне, Московская обл., г. Мытищи, школа № 19;

— Брунову Александру Сергеевичу, филиал МОУ "СОШ "Созвездие" с. Тёпловка" в селе Ириновка, Новобурасский р-н Саратовской обл.;

— Букиной Елене Юрьевне, г. Зеленокумск Ставропольского края, школа № 1;

— Викторовой Наталье Валентиновне, Владимирская обл., г. Александров, школа № 3;

— Вознесенской Марине Васильевне, Иркутская обл., г. Слюдянка, школа № 2;

— Волковой Татьяне Петровне, Владимирская обл., г. Струнино, школа № 11;

— Волкову Юрию Павловичу, Владимирская обл., г. Струнино, школа № 11;

— Волобуевой Ирине Константиновне, Алтайский край, г. Новоалтайск, школа № 15;

— Дикову Андрею Валентиновичу, г. Пенза, школа № 73;

— Долговой Галине Александровне, средняя школа села Восточное Нижегородской обл.;

— Евдокимовой Алевтине Ивановне, средняя школа поселка Осиновка, Алтайский край;

— Зудину Василию Павловичу, Ардатовский областной многопрофильный техникум, поселок Ардатов Нижегородской обл.;

— Каменецкой Татьяне Сергеевне, Москва, школа № 1530 «Школа Ломоносова»;

— Кашаповой Рамиле Хабибовне, Республика Татарстан, г. Лениногорск, школа № 8;

— Комбаровой Светлане Ивановне, г. Воронеж, лицей № 2;

— Лукьяновой Наталии Владимировне, Барнаульский государственный педагогический колледж имени В.К. Штильке;

— Муравьевой Ольге Викторовне, средняя школа деревни Муравьево, Вологодская обл.;

— Плюсниной Наталье Петровне, Хабаровский край, г. Комсомольск-на-Амуре, школа № 38;

— Порываевой Нафисе Сабитовне, Республика Татарстан, Мамадышский р-н, совхоз “Мамадышский”, политехнический колледж;

— Сысолятиной Виктории Александровне, Барнаульский государственный педагогический колледж имени В.К. Штильке;

— Тощевой Татьяне Валентиновне, Владимирская обл., г. Карабаново, школа № 7 им. А.П. Чулкова;

— Фирсовой Марии Николаевне, Волгоградская обл., г. Фролово, школа № 1 имени А.М. Горького;

— Черновой Ларисе Ивановне, средняя школа села Сердар, Республика Марий Эл;

— Чудутовой Елене Берчиевне, Москва, Кадетская школа-интернат «Навигацкая школа»;

— Шитовой Любови Алексеевне, средняя школа села Горелово Тамбовской обл.

Основы программирования на языке Python

Редакция начинает публикацию фрагментов книги с одноимённым названием, изданной издательством «ДМК Пресс» https://dmkpress.com/catalog/computer/programming/python/978-5-97060-641-4/

1. Понятия «алгоритм» и «программа»

Алгоритм решения задачи – точное описание порядка действий, которые надо выполнить для решения задачи.

Приведем ряд примеров.

Вот старинная задача: «Однажды крестьянину понадобилось перевезти через реку волка, козу и капусту. У крестьянина есть лодка, в которой может поместиться, кроме самого крестьянина, только одно существо или предмет – или волк, или коза, или капуста. Если крестьянин оставит без присмотра волка с козой, то волк съест козу; если крестьянин оставит без присмотра козу с капустой – коза съест капусту. Как крестьянину перевезти на другой берег все свое имущество в целости и сохранности?»

Чтобы решить задачу, крестьянин должен действовать так:

1) погрузить на лодку козу;
2) перевезти ее на другой берег;
3) выгрузить ее;
4) вернуться;
5) погрузить на лодку волка;
6) перевезти его на другой берег;
7) выгрузить его;
8) погрузить на лодку козу;
9) вернуться с ней на первый берег;
10) выгрузить ее;
11) погрузить на лодку капусту;
12) перевезти ее на другой берег;
13) оставить капусту на этом берегу;
14) вернуться на первый берег;
15) погрузить на лодку козу;
16) перевезти ее на другой берег;
17) выгрузить козу;
18) остаться на втором берегу.

Возможен и второй вариант решения задачи (найдите его). Самостоятельно составьте также алгоритм решения такой задачи: «К реке подошла группа из 20 солдат, которым нужно переправиться на другой берег. Река глубокая и бурная, и ее без лодки переплыть невозможно. Солдаты увидели двух мальчиков с лодкой. Лодка такова, что в ней размещается только один солдат, только один мальчик или только два мальчика. Как солдатам переправиться?»

Можно говорить об алгоритме приготовления торта в домашних условиях (только в этом случае мы алгоритм называем «рецептом»).

Пример из математики. Представьте, что товарищ принес вам чертеж прямоугольника и просит сообщить ему площадь этой фигуры. Какие действия вы должны выполнить, чтобы решить поставленную перед вами задачу? Вот ответ:

1) узнать (измерить) длину одной из сторон прямоугольника и запомнить (или записать) ее;
2) узнать (измерить) длину второй стороны фигуры и запомнить (или записать) ее;
3) рассчитать площадь, перемножив два найденных значения;
4) сообщить результат товарищу.

Каждый алгоритм рассчитан на какого-то *исполнителя* – человека (группу людей) или устройство (робот и др.), который выполняет действия, описанные в алгоритме.

Задание для самостоятельной работы

Представьте, что на стержень *A* надеты три диска:

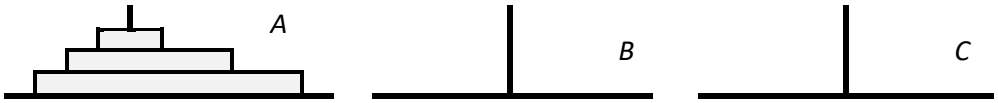


Рис. 1.1

Задача состоит в том, чтобы перенести диски со стержня *A* на стержень *C*, используя стержень *B* как промежуточный. При этом должны соблюдаться три условия:

1) за один «ход» можно переносить лишь один диск;

2) нельзя класть больший диск на меньший;

3) снятый диск нельзя отложить в сторону – он должен быть надет на один из стержней.

Опишите алгоритм решения задачи.

Понятие *«программа решения задачи»* возникает, когда речь идет о решении задачи на компьютере. Программа решения задачи – это алгоритм решения данной задачи, записанный в виде (на языке), «понятном» данному компьютеру. А какой язык «понимает» компьютер? В нем, как в техническом устройстве, информация может быть представлена в двоичном виде – в виде последовательности условных нулей и единиц. Значит, и описание действий, которые нужно выполнить для решения (в программировании их называют «командами»), должно поступать в компьютер в двоичном виде. Поэтому в первых электронных компьютерах программы представляли из себя последовательность двоичных чисел[1]:

1010 1101 0101 1000 …

Представляете, как трудно было программисту найти нужное место в программе, чтобы что-то изменить или добавить, найти и исправить ошибку? Чтобы устранить этот недостаток, были разработаны так называемые «языки программирования высокого уровня» (ЯПВУ)[2] – Фортран, Алгол, Бейсик и др. Программы на ЯПВУ оформляются в привычном человеку[3] виде: числа записываются как десятичные, а команды и другие служебные слова – на естественном (пусть и иностранном, но достаточно понятном) языке: PRINT, IF, BEGIN и т. п. Однако при этом получается противоречие – человеку-программисту удобно разрабатывать и читать программу, но компьютер такую программу не «поймет»!

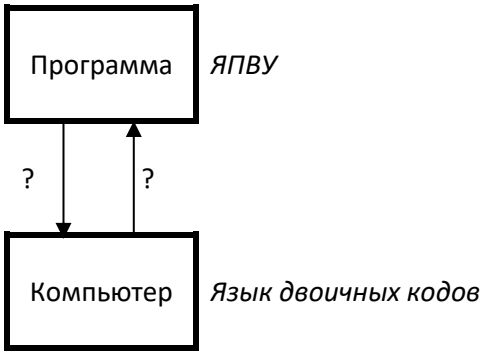


Рис. 1.2

[1] Так как числа в двоичной системе «длинные», то для записи программ использовалась также восьмеричная или шестнадцатеричная система счисления.

[2] Мы не рассматриваем использование для разработки программ языков ассемблера, имевшее место до применения ЯПВУ.

[3] ЯПВУ поэтому так и называются, что они, так сказать, «ориентированы» на человека, в отличие от машиноориентированных языков (языков двоичных кодов и ассемблеров).

Как же поступить, чтобы компьютер мог выполнять программу на ЯПВУ? А так, как можно общаться двум людям, один из которых знает только китайский язык, а второй – только русский: нужен переводчик! Поэтому во всех современных системах программирования предусмотрен транслятор – системная программа, осуществляющая перевод прикладной программы, написанной программистом на ЯПВУ, в язык машинных кодов и ее выполнение (см. рис. 1.3)[4].

Существуют два вида трансляторов:

1) интерпретаторы;

2) компиляторы.

Интерпретатор читает очередную команду программы и сразу ее выполняет, не переводя всю программу в машинный код.

Компилятор читает всю программу целиком, делает ее перевод и создает законченный вариант программы на «машинном языке», который затем и выполняется.

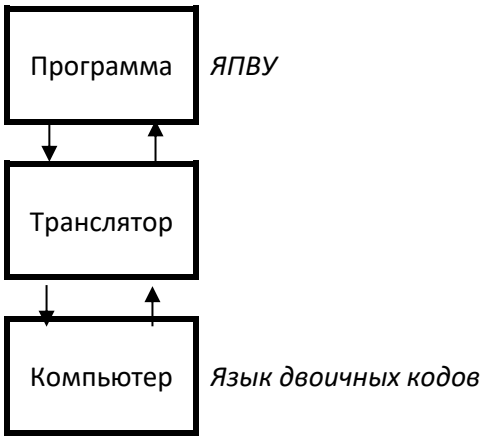


Рис. 1.3

Вопрос о преимуществах и недостатках каждого варианта транслятора здесь обсуждать не будем[5], а скажем, что язык программирования Python[6], как правило, работает в режиме интерпретации.

В качестве примера программы на Python приведем программу решения задачи расчета площади любого прямоугольника:

```
#Ввод размеров прямоугольника
a = int(input('Введите длину прямоугольника в см '))
b = int(input('Введите высоту прямоугольника в см '))
#Расчет площади
pl = a * b
#Вывод ответа на экран
print('Площадь прямоугольника равна', pl, 'кв. см')
```

Итак, Python – это язык программирования, на котором разрабатываются программы. Чтобы сделать этот процесс удобным для вас и других программистов, разработаны *системы программирования*, включающие транслятор, текстовый редактор (позволяющий копировать, удалять и перемещать фрагменты программ), справочную систему, средства отладки (поиска ошибок в программе) и др. Имеется несколько вариантов языка Python и систем программирования для него.

---

[4] Транслятор обычно выполняет также диагностику ошибок и др.

[5] Подумайте над этим вопросом.

[6] Язык Python иногда называют языком программирования не просто высокого, а «очень высокого уровня».

Контрольные вопросы

1. Что такое «алгоритм решения задачи»?
2. Какие вы знаете способы записи алгоритма?
3. В чем особенности алгоритма, который называют «программой»?
4. Почему языки программирования высокого уровня так называются?
5. Что такое «транслятор»? Какие функции он выполняет?
6. Какие виды трансляторов вы знаете? В чем особенность каждого вида?
7. Что включает в себя система программирования?

Что такое стек?[7]

Стек (от английского. *stack* — «стопка») — это структура данных в компьютере, работающая по принципу **LIFO** (**L**ast **I**n, **F**irst **O**ut, «последним пришёл, первым ушёл»).

Объясним принцип работы стека на простом примере.

Представьте стопку тарелок:

Как можно пользоваться ею?

1. Новую тарелку можно поставить только наверх стопки.
2. Взять тарелку можно только верхнюю (остальные извлечь, не трогая остальные, нельзя).
3. Сверху стопки видно только верхнюю тарелку (остальные скрыты).

Ещё пример – с мужской одеждой. Первой надевается майка, затем рубашка, за ней джемпер, пиджак и куртка. Надетая последней куртка будет сниматься первой и т.д. Вспомните также о солёных огурцах в банке 😊 (последний вложенный будет извлечён первым) и о герое фильма «Операция “Ы” и другие приключения Шурика», нарушившем правила работы со стеком.

По аналогичным правила происходят операции со стеком в компьютере. Возможно три операции:

1) добавление в него элемента (говорят – проталкивание); новый элемент, как и новая тарелка, оказывается на вершине стека;

2) удаление (извлечение) элемента, находящегося на вершине стека (остальные элементы смещаются к вершине стека);

3) чтение элемента на вершине (без его удаления).

---

[7] Эта статья, в отличие от предыдущей, предназначена для читателей, уже имеющих опыт программирования.

Стек как структура данных в компьютере используется при решении различных задач. Вот наглядный пример. Если вы работали с программой Microsoft Word, затем свернули её окно и стали работать с графическим редактором, а потом пользовались браузером, то после сворачивания последнего вы окажетесь в окне графического редактора, а после его закрытия – в окне программой Microsoft Word (для отслеживания активных программ и их документов используется стек вызовов). В текстовом редакторе Microsoft Word и других программах по правилам стека «работают» инструменты  (Отменить и Повторить), управляющие изменениями в документе Стек применяется для обхода графа, при использовании рекурсивных функций и процедур и т.д.

Возможность реализации стека предусмотрена во многих языках программирования. Первые две указанные операции с ним проводятся с помощью методов, соответственно, `push` и `pop`, третья с помощью функции `top`. Для проверки того факта, что стек пуст, используется функция логического типа `empty`.

Приведём ряд примеров использования стека при решении прикладных задач.

Как принято в нашем журнале, используем школьный алгоритмический язык (система программирования КуМир). Русский синтаксис этого языка и большое число комментариев делает программы максимально понятными и легко переводимыми на любой другой язык программирования.

Задача 1 «Разворот последовательности»

Условие

Вводится последовательность целых чисел, оканчивающаяся нулем. Число 0 в последовательность не входит.

Выведите элементы последовательности в обратном порядке. Для хранения данных используйте стек.

Входные данные

Вводится последовательность целых чисел, по модулю не превосходящих 10000. Ввод заканчивается, когда будет введено число 0. Всего чисел не более 100 (не считая нуля).

Выходные данные

Выведите элементы этой последовательности в обратном порядке, через пробел.

Решение

Без использования стека для решения следует использовать массив из 100 элементов, заполнить его заданными числами, подсчитывая при заполнении количество чисел кол, после чего вывести.

Соответствующая программа:

```
алг
нач цел таб м[1:100], цел n, кол, i
  | Ввод чисел и запись их в массив
  | с подсчётом их количества
  кол := 0
  нц
    вывод нс, "Введите очередное число "
    ввод n
    кол := кол + 1
    м[кол] := n
  кц_при n = 0
  | Вывод чисел в обратном порядке
  нц для i от кол - 1 до 1 шаг -1 | Число 0 не учитываем
    вывод м[i], " "
  кц
```

```
кон
```

Общий вид[8] соответствующей программы с использованием стека:

```
алг
нач цел таб м[1:100], цел n
  Создание стека
  | Ввод чисел и запись их в стек
  нц
    вывод нс, "Введите очередное число "
    ввод n
    добавить_в_стек(n)
  кц_при n = 0
  извлечь | число 0 без вывода на экран
  | Вывод чисел в обратном порядке
  нц пока стек не пуст
    вывод значения на вершине стека, " "
    удалить из стека
  кц
кон
```

Видно, что в последнем варианте не требуется использовать массив и подсчитывать количество чисел последовательности.

Задача 2

Условие

Дана строка, содержащая скобки “(“ и “)”. Определить, является ли расстановка скобок правильной. В правильной последовательности:

- каждой открывающей скобке соответствует закрывающая;.
- скобки закрываются в правильном порядке (последняя открытая — первая закрывается).

Более формально: пустая последовательность является правильной. Если А – правильная, то последовательность (А) тоже правильная Если А и В – правильные последовательности, то последовательность АВ – правильная.

Например, в строках “()”, “(())”, “()()”, “()((()))(())” расстановка правильная, а в строках “())”, “(()”, “)(()”, “)(())” – неправильная[9].

Решение

Как определить – правильная расстановка или нет? Равенство общих количеств скобок двух видов в строке не означает правильность расстановки: “))((”.

Можно рассуждать так. Если очередная скобка открывающая, то её количество будем увеличивать на 1. В противном случае это количество увеличим на 1 и проверим, не стало ли закрывающих скобок больше. Если больше (например, “)())”), то расстановка неправильная и дальнейшие проверки можно не проводить.

Ещё один признак неправильной расстановки такой – когда после просмотра всей строки общее количество открывающих скобок больше общего количества закрывающих.

Разработаем программу, учитывающую эти рассуждения.

Используем в ней следующие основные переменные величины

`ст` –заданная строка;

`кол1` – текущее количество открывающих скобок;

`кол2` – то же, закрывающих;

---

[8] В школьном алгоритмическом языке процедуры для работы со стеком не предусмотрены.

[9] Такая задача возникает в системах программирования при проверке правильности расстановки скобок в арифметическом выражении.

прав – величина логического типа, фиксирующая правильность или неправильность расстановки скобок.

Соответствующая программа:

```
алг
нач лит ст, цел кол1, кол2,i, лог прав
  | Ввод заданной строки
  ввод ст
  | Начальные значения переменных
  кол1 := 0
  кол2 := 0
  прав := да | Сначала принимаем, что расстановка правильная
  | Рассматриваем все символы строк
  нц для i от 1 до длин(ст)
    если ст[i] = "("
      то
        кол1 := кол1 + 1
      иначе | Скобка закрывающая
        кол2 := кол2 + 1
        | Сравниваем количества
        если кол2 > кол1
          то
            прав := нет
            выход
        все
    все
  кц
  | Проверка количества после рассмотрения всех символов
  если кол1 > кол2
    то
      прав := нет
  все
  | Вывод ответа
  если прав
    то
      вывод "Расстановка скобок правильная"
    иначе
      вывод "Расстановка скобок неправильная"
кон
```

Более сложная задача 3

Дана строка, содержащая скобки “(“, “)”, “[“, “]”, “{“, и “}”. Нужно определить, является ли расстановка скобок правильной. Например, в строках “([{}])” и “[()]{}{[()()]()}” расстановка правильная**,** а в строках “({[}])” и “([)]” – нет.

Для её решения приведённые рассуждения (использование шести переменных-счётчиков) не подходят[10]. А при использовании стека будет очень простое и логичное решение. Продемонстрируем это сначала на примере первой задачи.

Когда начальные скобки – открывающие, будем последовательно заполнять ими стек:

Элементы строки		
(	(	(
Операция		
доб	доб	доб
Состояние стека		
(	(	(
	(	(
		(

Если после этого встретится закрывающая скобка, то удалим скобку на вершине стека:

Элементы строки					
(	(	(	)	)	)
Операция					
доб	доб	доб	уд	уд	уд
Состояние стека					
(	(	(	(	(	
	(	(	(		
		(			

Если строка закончилась, то в случае, если стек пуст, расстановка скобок правильная.

Если же строка продолжается, то:

- если очередной символ – открывающая скобка, то дальнейшие действия аналогичны;

Элементы строки							
(	(	(	)	)	)	(	(
Операция							
доб	доб	доб	уд	уд	уд	доб	доб
Состояние стека							
(	(	(	(	(		(	(
	(	(	(				(
		(					

- если очередной символ – закрывающая скобка (напомним, что стек пуст), то расстановка скобок неправильная:

Элементы строки						
(	(	(	)	)	)	)
Операция						
доб	доб	доб	уд	уд	уд	!
Состояние стека						
(	(	(	(	(		
	(	(	(			
		(				

[10] Например, для строки “({[}])” значения в трёх парах счётчиков равны, а расстановка неправильная.

После обработки всех символов строки при правильной расстановке скобок стек должен быть пустым (все скобки закрыты). Обратим также внимание на то, что скобки всех подстрок с правильными расстановками в ходе обработки строки из стека удалялись.

Общий вид соответствующей программы:

```
алг
нач лит ст, цел i, лог прав
  ввод ст
  прав := да
  создание стека
  нц для i от 1 до длин(ст)
    если ст[i] = "("
      то | Добавляем её в стек
        добавить_в_стек(ст[i])
      иначе  | Скобка закрывающая
        если стек не пуст
          то
            извлечь
          иначе
            прав := нет
            выход
        все
    все
  кц
  | Проверка стека на "пустоту"
  если стек не пуст
    то
      прав := нет
  все
  | Вывод ответа
  …
```

Для задачи с разными типами скобок общая схема решения такая:

1. Проходим по строке:
   1.1. Если скобка открывающая – добавляем её в стек
   1.2. Если скобка закрывающая:
        проверяем стек.
        если он пуст
          то
            расстановка неправильная
          иначе (стек не пуст);
            сравниваем эту скобку с последней открытой (она на вершине стека):
              если совпадают – удаляем открывающую скобку из стека;
              если типы скобок не совпадают – расстановка неправильная.
2. В конце стек должен быть пустым (все скобки закрыты).

Общий вид соответствующей программы:

```
алг
нач лит ст, цел i, лог прав
  ввод ст
  прав := да
  нц для i от 1 до длин(ст)
    | Если скобка открывающая
    если ст[i] = "(" или ст[i] = "[" или ст[i] = "{"
      то
        добавить_в_стек(ст[i])
      иначе | Скобка закрывающая
```

```
          если стек пуст
            то
              прав := нет
              выход
            иначе
              | Сравниваем её со скобкой на вершине стека
              если ст[i] = ")" и верш = "(" или
                  ст[i] = "]" и верш = "[" или ст[i] = "}" и верш = "{"
                то
                  извлечь
                иначе
                  прав := нет
                  выход
              все
          все
        все
      кц
      | Проверка стека на "пустоту"
      если стек не пуст
        то
          прав := нет
      все
      | Вывод ответа
      …
кон
```

где *верш* – функция, возвращающая значение на вершине стека.

Эффектно, не правда ли?

В целом можно сказать, что стек – это мощный инструмент, с помощью которого можно решать сложные системные и прикладные задачи.

Задание для самостоятельной работы

На языке программирования, который вы изучаете, разработайте программу решения задачи «Удалить лишние скобки» с использованием стека. С особенностями создания и использования стека в этом языке ознакомьтесь самостоятельно.

Условие

Дана строка, составленная из круглых скобок. Определите, какое наименьшее количество символов необходимо удалить из этой строки, чтобы оставшиеся символы образовывали правильную скобочную последовательность.

Входные данные

Дана строка из круглых скобок. Длина строки не превосходит 100000 символов.

Выходные данные

Выведите единственное целое число — ответ на поставленную задачу.

Примеры входных данных выходные данные

()(()
))(((

Выходные данные, соответственно

2
5

Необычная пирамида

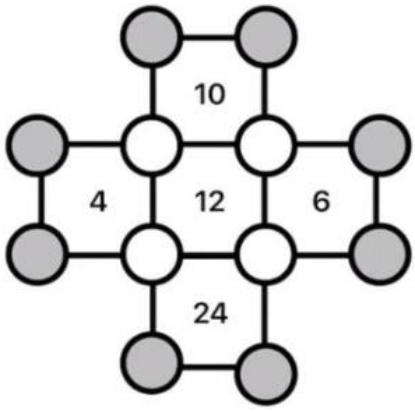
У Мити было шесть кубиков разного размера, из которых он решил сложить пирамиду. Для того чтобы пирамида была устойчивой, он решил всегда ставить кубик меньшего размера на кубик большего размера. Как ему правильно расположить кубики, если известна длина ребра каждого кубика, указанная в различных системах счисления:

Номер кубика	Длина ребра
1	101_9
2	131_8
3	202_7
4	222_6
5	303_5
6	313_4

В ответе укажите последовательность номеров кубиков, согласно которой должна формироваться пирамида.

Пять квадратов с числами

Числа, записанные на рисунке внутри квадратов, указывают сумму чисел в вершинах соответствующего квадрата. Например, 12 – это сумма четырёх чисел, расположенных в белых кружочках. Определите сумму чисел в серых кружках.



Три футбольных матча

Футбольная команда 8Б класса сыграла три матча. В одном она победила, второй сыграла вничью, а третий – проиграла. Известно, что всего за эти матчи команда забила 3 гола и пропустила 1. Можно ли сказать, каков был счёт в каждом матче?

Об использовании двоичной системы счисления в докомпьютерную эпоху

В статье приводятся примеры (в том числе малоизвестные) применения двоичной системы счисления до момента появления электронных вычислительных машин. Дальнейшее содержательное знакомство с рассмотренными примерами может стать тематикой интересных исследовательских проектов по информатике и межпредметных проектов по математике и информатике.

«…эта система [двоичная] лучше всякой другой годилась бы для устройства арифметической машины…»
Эдуард Люка, французский математик,
цитата из книги «Занимательная математика» [11], 1882 г.[11]

Во многих источниках, в том числе и учебных пособиях, утверждается, что впервые привычный современный вид двоичной системе придал великий немецкий ученый Готфрид Вильгельм Лейбниц (1646–1716) в 1703 г. в статье «Explication de l'Arithmétique Binaire» («Объяснение двоичной арифметики») [16]. Однако это, как можно прочитать в статьях [5, 6], не совсем так. Двоичная система была известна и до Лейбница: в качестве примера приводились и описывались работы Томаса Харриота и Хуана Карамюэля. При этом отмечалось, что в глазах учёных двоичная система не представляла практического значения (например, для производства вычислений), и виделась им лишь неким математическим курьёзом.

Первый пример практического применения двоичной системы счисления привел Лейбниц в упоминавшейся работе [16]. В ней он пишет, что двоичная запись числа может быть использована для определения минимального набора гирь для взвешивания грузов или для минимального набора монет (или купюр) для формирования некоторых сумм денег[12]. Таким набором являются гири массой 1, 2, 4, … до степени двойки, не превышающей заданное число, а конкретный набор гирь для уравновешивания на чашечных весах конкретного груза определяется по разрядам двоичной записи массы груза, в которых записана единица. Например, для уравновешивания груза массой 13 кг следует использовать гири массой 8, 4 и 1 кг ($13_{10} = 1101_2$).

***

В 1623 г. английский философ, историк, политик Фрэнсис Бэкон (1561–1626) в трактате «De Dignitate et AugentisScientiarum» («О достоинстве и приумножении наук») описал следующий метод шифрования текста [10]. Каждую букву алфавита в сообщении он предложил шифровать некоторой последовательностью букв двухлитерного алфавита, например, A и B. Тогда шифрующие последовательности для букв латинского алфавита будут иметь следующий вид:

Буква	Шифрующая последовательность
a	AAAAA
b	AAAAB
…	…
t	BAABA
…	…
z	BABBB

[11] На русском языке цитата публикуется впервые.
[12] Имелось в виду, что монеты (купюры) имеют достоинство 1, 2, 4, … .

По сути, пользуясь современной терминологией, это был 5-битный двоичный код (если иметь в виду, что буква А соответствует нулю, а В – единице).

***

Если код Бэкона был предназначен для шифрования текста, и неизвестно, пользовался ли им кто-либо, то так называемый «шрифт Брайля», разработанный в 1824 году французом Луи Брайлем, был первой системой записи с двоичным кодированием символов, которой пользовались (и пользуются) миллионы людей [1]. Он предназначен для письма и чтения незрячими и плохо видящим людям. В нём буквы на бумаге изображаются в виде проколов. Всего в предложенном Брайлем варианте могло быть до 6 проколов (точек), расположенных в два столбца. Каждой букве соответствовало определённое сочетание точек:

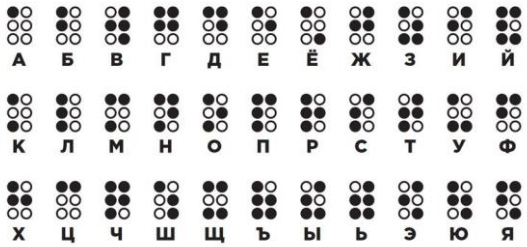


Рис. 2

Тёмные точки – это выпуклости на бумаге, светлые – плоские элементы кода символа.

Всего шрифтом Брайля можно было записать $2^6 = 64$ различных символа. При письме точки прокалываются, и поскольку читать можно только по выпуклым точкам, «писать» текст приходится с обратной стороны листа. Текст «пишется» справа налево, затем страница переворачивается, и текст читается слева направо.

Если пронумеровать места для точек так, как показано на рис. 3, и выпуклым точкам сопоставить цифру 1, а плоским элементам кода – 0, то можно сказать, что, например, для буквы Т двоичный код будет таким: 011110.

Рис. 3

***

По сути двоичным был так называемый «код Морзе», который предложил в 1838 г. американец Сэмюэл Морзе, хотя некоторые исследователи полагают, что её автором был Альфред Вейл – партнер Морзе по бизнесу, известный тем, что ввёл «коммерческий код» из групп по 5 символов (правда, и здесь автор/авторы системы не использовали цифры 0 и 1).

В современном варианте международного кода Морзе каждый символ кодируется на бумаге некоторой последовательностью из тире и точек, а при передаче с помощью технических средств (телеграфом, по радио и т. п.) – последовательностью длинных и коротких сигналов [8]. Если точку интерпретировать как 0, а тире – как единицу, то получим двоичный код. Отличие кода Морзе от двоичного кода с фиксированной разрядностью будет в том, что буквам с меньшей частотой использования в тексте того или иного языка, поставлена в соответствие более короткая последовательность точек и тире. Так, код Морзе для русской буквы Т состоит лишь из одного тире (в двоичном виде – 1), а для русской буквы Щ имеет вид – – · – (или 1101).

В связи с тем, что в коде Морзе символам соответствуют различные длины последовательностей точек и тире, учащимся может быть предложен следующий вопрос на смекалку: «Рассказывают, что, создавая свой код, Морзе отправился в ближайшую типографию, в которой в те времена, да и в течение длительного времени потом, текст в книгах, газетах и т. п. набирался вручную из отдельных литер. Литера – металлический,

пластмассовый или деревянный брусок прямоугольного сечения с рельефным изображением буквы или знака на одной из его сторон. Зачем Морзе ходил в типографию?» [2].

(Целью визита в типографию было узнать, какие буквы наборщики текста используют чаще, а какие – реже. Такой подход уменьшает общее количество передаваемых сигналов, что ускоряет передачу информации по каналу связи).

***

Помимо кодирования информации, идеи использования двоичного представления чисел можно встретить в различных математических фокусах и головоломках. Так, известен фокус по отгадыванию числа (например, возраста человека или т. п.) по таблице (рис. 3 [3]).

I.	II.	III.	IV.	V.	VI.
1	2	4	8	16	32
3	3	5	9	17	33
5	6	6	10	18	34
7	7	7	11	19	35
9	10	12	12	20	36
11	11	13	13	21	37
13	14	14	14	22	38
15	15	15	15	23	39
17	18	20	24	24	40
19	19	21	25	25	41
21	22	22	26	26	42
23	23	23	27	27	43
25	26	28	28	28	44
27	27	29	29	29	45
29	30	30	30	30	46
31	31	31	31	31	47
33	34	36	40	48	48
35	35	37	41	49	49
37	38	38	42	50	50
39	39	39	43	51	51
41	42	44	44	52	52
43	43	45	45	53	53
45	46	46	46	54	54
47	47	47	47	55	55
49	50	52	56	56	56
51	51	53	57	57	57
53	54	54	58	58	58
55	55	55	59	59	59
57	58	60	60	60	60
59	59	61	61	61	61
61	62	62	62	62	62
63	63	63	63	63	63

Рис. 4

С помощью этой таблицы можно отгадать любое задуманное число от 1 до 63, если задумавший число скажет, в каких столбцах оно встречается (номера столбцов приведены в первой строке таблицы). Например, если задумано число 22, то по названным номерам столбцов (5, 3 и 2) его можно вычислить следующим образом: $2^{5-1} + 2^{3-1} + 2^{2-1} = 16 + 4 + 2$ (это можно сделать, даже не смотря на таблицу). Можно также запомнить, что с первым столбцом связано число 1, со вторым – 2, с третьим – 4, с четвёртым – 8, с пятым – 16, с шестым – 32.

Секрет фокуса заключается в том, что в таблице в первом столбце справа выписаны те десятичные числа, которым в двоичной системе счисления соответствуют числа, оканчивающиеся на 1, во втором – числа, у которых в двоичном представлении вторая от конца цифра равна 1, в третьем – десятичные числа, которым в двоичной системе соответствуют числа, имеющие 1 в третьем справа разряде, и т. д. Так, уже приведённое число 22 в двоичной системе счисления имеет вид: 10110, и в таблице это число записано в пятом, третьем и втором столбцах. Если вспомнить правило перевода чисел из двоичной системы в десятичную, то по записи 10110 соответствующее десятичное число можно получить так: $2^4 + 2^2 + 2^1$ (это так называемая «развернутая запись числа»), или, «привязываясь» к нашей нумерации столбцов таблицы – $2^{5-1} + 2^{3-1} + 2^{2-1}$.

Описанный фокус упоминается в книге «История двоичной и других недесятичных систем счисления» [15], которую в 1971 году опубликовал Антон Глазер, профессор математики Пенсильванского университета, США. Этот фундаментальный труд, выдержавший два издания, содержит подробный обзор всех известных работ по истории недесятичных систем счисления с конца XVI века до момента появления компьютеров.

А.Глазер сначала пишет, что фокус на отгадывание числа в 1875 году описал немецкий математик Мориц Кантор (1829–1920), а потом уточняет: «Кантор был бы очень расстроен, если бы ему сообщили, что в издании 1814 года книги Жака Озана́ма[13] “Математические и физические развлечения” этот фокус уже был описан» [15, с . 93]. Интересная информация, связанная с этим уточнением, приведена в статье [3].

***

В 1872 году неизвестный автор опубликовал в Лионе, Франция, 16-страничную брошюру «Théorie du baguenaudier…» [17]. Она посвящена старинной головоломке, происходящей, предположительно, из Китая, состоящей из замкнутой проволочной «вилки», имеющей вид длинной шпильки для волос и воткнутой обоими свободными концами в рукоятку, и нескольких колец, связанных между собой довольно сложным образом (см. рис. 5). Задача состоит в том, чтобы снять всю систему колец с вилки или надеть её обратно. Французское название головоломки – *baguenaudier*. Русский вариант головоломки называется *меледа́* [7].

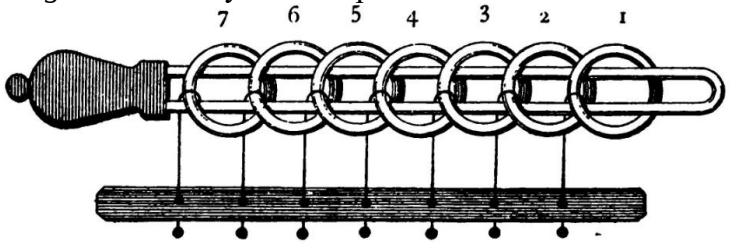


Рис. 5

Эдуард Люка в изданной в Париже в 1882 г. книге «Récréations mathématiques» («Математические развлечения») написал об анонимном авторе брошюры: «Нам недавно удалось узнать, что был Луи Гро, советник лионского апелляционного суда» [12, с. 169].

Луи Гро в своей работе, среди прочего, приводит таблицу, начальный фрагмент которой показан на рис. 6.

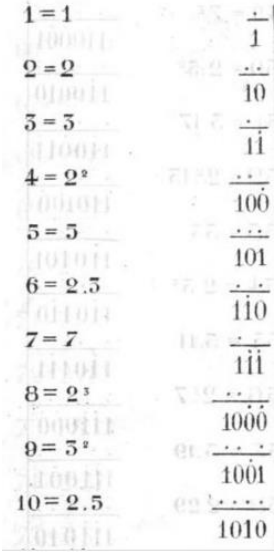


Рис. 6

[13] Французский математик, профессор Сорбонны (1640–1718). В 1694 году он опубликовал книгу «Récréations mathématiques et physiques» («Математические и физические развлечения»).

Анализ показывает, что в таблице:

1) схемы с линией и точками выше и ниже её иллюстрируют текущую ситуацию на головоломке, когда стоит задача надеть на «вилку» некоторое количество колец. Точки соответствуют кольцам; расположенные над линией – надеты, находящиеся ниже линии – сняты;

2) десятичные числа слева – порядковые номера манипуляций с кольцами («ходов»);

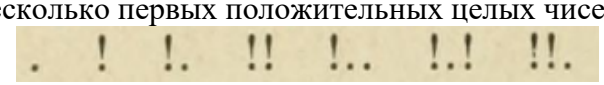
3) числа под схемами – двоичный эквивалент номера хода.

Такая таблица позволяет определить общее число ходов, требующихся для решения головоломки при том или ином количестве колец.

***

В 1899 году итальянский математик Джузеппе Пеано (1858–1932) опубликовал статью, предлагающую новую систему записи чисел, в основе которой лежала двоичная система счисления. Подробно он описал ее в сборнике «Основания математики» [13] и в брошюре «Применение двоичной нумерации в стенографии» [14].

Учёный предложил использовать символы "." и «!»[14], обозначающие, соответственно, для 0 и 1 и привел ноль и несколько первых положительных целых чисел в виде:



Для представления больших чисел Пеано ввел изображение, которые можно описать как «ромашку», в которой может быть до восьми лепестков. Каждый имеющийся лепесток соответствует единице, каждый отсутствующий – нулю. Таким образом, «ромашка» определяла восьмиразрядное двоичное число. Нумерация разрядов в числе показана на рис. 7, а несколько вариантов «цветка» и соответствующих десятичных значений – на рис. 8.

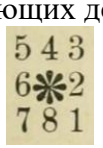


Рис. 7

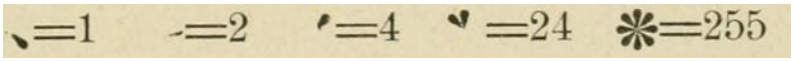


Рис. 8

Примечание. Приведённые на рис. 8 изображения соответствуют двоичным числам 1, 10, 100, 11000 и 11111111.

При использовании двух изображений-«ромашек» можно было представлять и бóльшие числа (см. рис. 9).

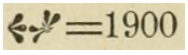
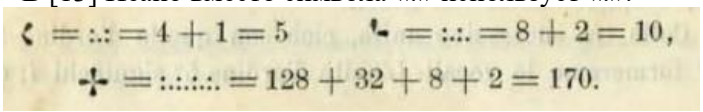


Рис. 9

Свою систему Пеано предложил использовать в стенографической машине – разновидности пишущей машинки для стенографирования[15]. В брошюре [14] он упоминает использовавшуюся в то время в Сенате Италии 10-клавишную стенографической машину профессора Антонио Микела Зукко и утверждает, что машина, созданная на основе его системы, проще и удобнее. При использовании двух 8-клавишных клавиатур можно одновременно записать две буквы или два слога, а общее число возможных вариантов букв, цифр и слогов составляет 65 536.

---

[14] В [13] Пеано вместо символа «!» использует «:»:



(о графических изображениях — см. далее).

[15] Пеано писал, что его система кодирования текста может быть применена и в телеграфной связи, «используя весь ее [связи] потенциал…».

Для удобства запоминания «системы кодировки» (соответствия между буквами, цифрами и слогами и их числовым кодами-«ромашками») Пеано разработал систему, учитывающую особенности итальянского языка, а также применил ряд приемов, облегчающих запоминание. Так, например, «глухой» согласной *f* соответствует код  (напоминающий букву *f*), при удалении из которого одного «лепестка» можно получить код «звонкой» согласной *v* (). Предложенные учёным коды 25 латинских букв (без буквы *w*) и 10 цифр приведены на рис. 10 и 11, соответственно, а код слога *pas* – на рис. 12 [13].

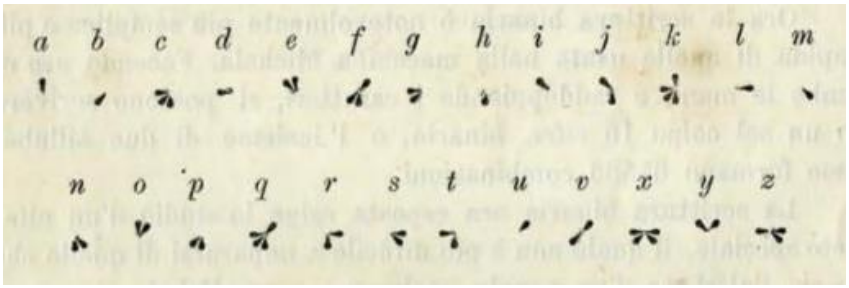


Рис. 10

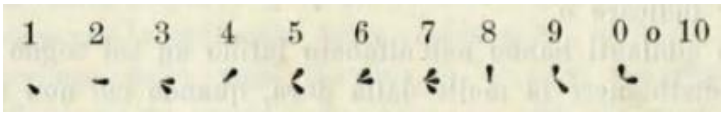


Рис. 11

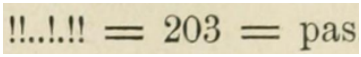


Рис. 12

***

В 1901 году Чарльз Бутон, профессор математики Гарвардского университета (США), опубликовал статью [11], связанную с игрой, которую он назвал «ним». Интересная информация о происхождении игры и её названии, в том числе убедительно опровергающая распространённые гипотезы и мифы, связанные с этим, приведена в книге [9].

Суть игры заключается в следующем. Имеется некоторое количество мелких предметов (камешков, монет, спичек и т. п.), разложенных в несколько кучек. Два игрока по очереди забирают по одному или нескольку предметов из одной любой кучки (или все предметы из нее). Выигрывает тот, кто возьмёт последний предмет (или несколько последних предметов)[16].

В статье Бутон провел полный анализ игры с тремя кучками предметов и представил доказательство существования оптимальной выигрышной стратегии.

Целью статьи, как пишет сам ее автор, было доказательство того, что, если один из игроков *A* может каждым своим ходом оставлять некоторый набор значений предметов в кучках, другой игрок *B* не может выиграть. Такой набор значений Бутон назвал «безопасное сочетание» (*safe combination*), или «безопасная позиция». Он пишет: «Запишите количество предметов в каждой кучке в двоичной системе счисления[17] и разместите эти числа одно под другим так, чтобы единицы были в одном столбце. Если при этом сумма в каждом столбце равна 2 или 0, то такой набор предметов образует безопасную позицию».

Например, Бутон приводит пример безопасной позиции для трех кучек из 9, 5 и 12 предметов:

1 0 0 1,
1 0 1,
1 1 0 0,

[16] Можно играть и наоборот — считать того, кому достанется последний предмет, проигравшим.

[17] И приводит при этом такой пример двоичной записи числа 9:
$1 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 1001$.

Двоичная запись позволяет понять, что для двух любых различных чисел всегда однозначно определяется третье, которое образует безопасную комбинацию с двумя заданными числами, и наоборот – если числа *a, b, c* образуют безопасную комбинацию, любые два из них определяют оставшееся третье. Кроме того, видно, что игрок *B*, делая очередной ход в безопасной позиции, которую ему «оставил» соперник, никогда сам не сможет получить такую же – любой его ход делает позицию опасной (для себя). Выигрышная стратегия состоит в том, чтобы оставлять после каждого своего хода безопасную позицию.

В последующих доказательствах Бутон также везде использует двоичную запись чисел.

В заключение он приводит признак безопасной позиции для любого количества кучек (аналогичный признаку для трёх кучек – общее количество единиц в каждом столбце при записи двоичных значений чисел предметов в кучках одно под другим должно быть чётным), а также анализирует другой вариант игры – при котором проигравшим считается взявший последний предмет.

***

Через 52 года после слов Э.Люка (см. эпиграф к статье) немецкий студент Конрад Цузе (1910–1985) начал работу по созданию вычислительной машины, использующей двоичную систему счисления. В 1937 году такая механическая машина, названная Z1 (что означало «Цузе 1»), была готова и заработала!

А в конце 40-х годов XX столетия началась эра электронных вычислительных машин, в основе устройства и работы которых лежала двоичная система счисления…

Литература

1. Брайль Луи. https://ru.wikipedia.org/wiki/Брайль_Луи

2. *Златопольский Д.М.* 400 вопросов по информатике на логику и смекалку. М. ДМК Пресс, 2021. 226 с.

2. *Златопольский Д.М.* Два сюжета об истории математики, или Почему надо работать с первоисточниками // Потенциал. Математика, физика, информатика. 2023. № 1. С. 13–16.

4. *Златопольский Д.М.* Занимательная информатика. М. Бином. Лаборатория знаний. 2010.

5. *Златопольский Д.М., Шилов В.В*. Из истории двоичной системы счисления. Томас Харриот // Информатика в школе. 2020. № 6. С. 19–23.

6. *Златопольский Д.М., Шилов В.В*. Из истории двоичной системы счисления. Хуан Карамюэль // Информатика в школе. 2020.№ 8. С. 61–63.

7. Меледа. https://ru.wikipedia.org/wiki/Меледа

8. Морзе Сэмюэл. https://ru.wikipedia.org/wiki/Морзе_Сэмюэл

9. *Шилов В.В*. Удивительная история информатики и автоматики. М.: ЭНАС, 2011.

10. Шифр_Бэкона. https://ru.wikipedia.org/wiki/Шифр_Бэкона

11. *Bouton C.L*. Nim, a game with a complete mathematical theory // The Annals of Mathematics, 2nd Ser., Vol. 3, No. 1/4. (1901–1902). P. 35–39.

12. *Édouard Lucas*. Récréations mathématiques. Paris. 1882. https://archive.org/stream/rcrationsmathma00lucagoog#page/n26/mode/1up

13. *Giuseppe Peano*. Formulaire de mathematiques.3 vols. Paris. 1901. P. 77–78. https://archive.org/details/formulairedemath00pean/page/211/mode/1up

14. *Giuseppe Peano*. La numerazione binaria applicata alla stenografia. Torino. 1898. https://archive.org/details/PeanoNumerazioneBinaria/page/n11/mode/2up

15. *Glaser A.* History of binary and other nondecimal numeration. 1981.

16. *Leibnitz, Godefroy-Guillaume*. Explication de l'Arithmétique Binaire… // Mémoires de mathématique et de physique de l'Académie royale des sciences, Académie royale des sciences. 1703. P. 85–89.

17. Théorie du baguenaudier par un clerc de notaire lyonnais. https://books.google.fr/books?id=EcoBJRekd-sC&pg=PP1#v=onepage&q&f=false

Конкурс № 73 «Числа из четырёх четвёрок»

В качестве задания этого конкурса предлагаем решить такую задачу «Используя четыре цифры 4 (отдельно или несколько подряд), получить *как можно больше* чисел от 1 до 50. Между числами с цифрами 4 можно записывать:

- знаки арифметических операций (в том числе операции извлечения квадратного корня);
- знак факториала «!» (напомним, что 3! = 1 × 2 × 3 = 6, 5! = 1 × 2 × 3 × 4 × 5 =120 и т.п.);
- запись $.4$, которая будет означать число 0,4;
- запись $\frac{4}{.4}$, которая будет означать число 9;
- запись $\sqrt[.4]{4}$, которая будет означать число 32»

При формировании чисел 9 и 32 указанные условные записи не использовать».

Ответы присылайте по адресу: mirinfo@infojournal.ru. Укажите в письме свои фамилию и имя, населенный пункт, номер школы и фамилию, имя и отчество учителя информатики. Срок представления ответов — 15 октября 2025 г.

Уважаемые читатели!

Сообщаем, что очередные номера нашего интернет-журнала публикуются в период с 15 по 20 число каждого месяца, кроме июля.