Решение варианта 7991 для 9 классов

Задание 1

Древние Майя использовали 20-ичную систему счисления за одним исключением: во втором разряде было не 20, а 18 ступеней, то есть за числом (17)(19) сразу следовало число (1)(0)(0). Это было сделано для облегчения расчётов календарного цикла, поскольку (1)(0)(0) = 360 – примерно равно числу дней в солнечном году. Разработайте алгоритм перевода натуральных чисел из десятичной с.с. в систему Майя.

Решение. Для хранения чисел будем использовать массив, каждый элемент которого будет хранить одну цифру числа в системе счисления Майя. Для перевода будем использовать обычный алгоритм – делим число в десятичной системе счисления на 20 и записываем полученные цифры. После получения всех цифр надо проверить цифру второго разряда – если она больше или равна 18, то вычитаем 18 из этой цифры и осуществляем перенос 1 в следующий разряд. Но поскольку в третьем разряде может оказаться цифра (19), то перенос может перейти в четвёртый разряд и т.д.

```
число = запись
  цел digits[100]                              // Массив цифр
  цел count                                    // Количество цифр в числе
кон

алг ПереводЧислаВСистемуМайя()
нач
  цел n, tr, i
  число m

  ввод n

  count = 0
  выполнить
    m.digits[count] = n mod 20
    count = count + 1
    n = n div 20
  до n = 0 или count > 100

  если count > 100 то
    вывод 'Слишком большое число'
  иначе
    если m.digits[2] >= 18 то
      m.digits[2] = m.digits[2] – 18
      tr = 1
      i = 3
      пока tr = 1 и i <= 100
      нц
        m.digits[i] = m.digits[i] + tr
        если m.digits[i] >= 20 то
          m.digits[i] = m.digits[i] - 20
        иначе
          tr = 0
        всё
        i = i + 1
      кц
    всё
    если i > 100 то
      вывод 'Слишком большое число'
    иначе
      если i > count то
        count = i
      всё
      для i от count + 1 до 100
      нц
        m.digits[i] = 0
      кц
      для i от count до 1 шаг -1
      нц
        вывод '(', m.digits[i], ')'
      кц
    всё
```

```
  всё
кон
```

Задание 2

На листе бумаги в строке записано N натуральных чисел. Разработайте алгоритм, который переупорядочивает их (в новой строке) так, чтобы начало строки составляли числа, имеющие чётные значения, расположенные в порядке возрастания. Оставшуюся часть строки должны составлять нечётные значения, расположенные в порядке убывания.

Решение. Переставим элементы массива так, чтобы сначала шли чётные элементы, а потом нечётные. Для этого найдём элемент с номером i – первый нечётные элемент и элемент с номером j – последний чётный элемент, и поменяем их местами. Затем найдём номер i следующего нечётного элемента и номер j предыдущего чётного элемента, и снова поменяем их местами. Продолжать эти действия надо до тех пор, пока i не станет больше j. На тот случай, если в массиве нет чётных или нечётных элементов, проверяем, чтобы i не стало больше общего количества элементов массива, а j не стало меньше 1. После переупорядочения i будет хранить номер первого нечётного элемента, а j – номер последнего чётного элемента, и теперь нужно отсортировать обе части массива.

Одним из самых быстрых методов сортировки является *сортировка Хоара*, которую также называют *быстрой сортировкой*. Основная идея алгоритма состоит в том, что случайным образом выбирается некоторый элемент массива y, после чего массив просматривается слева, пока не встретится элемент $x[i]$ такой, что $x[i] \geq y$, а затем массив просматривается справа, пока не встретится элемент $x[j]$ такой, что $x[j] \leq y$. Эти два элемента меняются местами, и процесс просмотра, сравнения и обмена продолжается, пока i не окажется больше j. В результате массив окажется разбитым на две части – левую, в которой значения элементов будут меньше или равны y, и правую, в которой значения элементов будут больше или равны y. Далее процесс рекурсивно продолжается для левой и правой частей массива до тех пор, пока каждая часть не будет содержать один элемент – массив из одного элемента по определению считается упорядоченным. Можно также отдельно рассмотреть случай для массива из двух элементов.

```
алг Переупорядочение()
нач
  цел n, a[n], i, j, c

  ввод n
  для i от 1 до n
  нц
    ввод a[i]
  кц

  i = 1
  j = n
  пока i < j
  нц
    пока i <= n и a[i] mod 2 = 0
    нц
      i = i + 1
    кц
    пока j >= 1 и a[j] mod 2 = 1
    нц
      j = j - 1
    кц
    если i < j
      c    = a[i]
      a[i] = a[j]
      a[j] = c
    всё
  кц

  QuickSort(a, 1, j)
  QuickSort(a, i, n)
кон
```

```
алг QuickSort(арг цел x[1000], арг цел n1, n2)
нач
  цел i, j
  цел y, k

  если n2 - n1 = 1 то
    если x[n1] > x[n2] то
      y = x[n1]
      x[n1] = x[n2]
      x[n2] = y
    всё
  иначе
    если n2 - n1 > 1 то
      k = x[(n1 + n2) div 2]
      i = n1
      j = n2
      повторять
        пока (x[i] < k)
        нц
          i = i + 1
        кц
        пока (x[j] > k)
        нц
          j = j - 1
        кц
        если i <= j то
          y = x[i]
          x[i] = x[j]
          x[j] = y
          i = i + 1
          j = j - 1
        всё
      до i > j
      QuickSort(x, n1, j)
      QuickSort(x, i, n2)
    всё
  всё
кон
```

Задание 3

В качестве ключа для шифрования секретных сведений использовалось число S, являющееся суммой некоторых целых положительных чисел A, B и C ($A < B < C$). Причём $B - A = C - B$. Для дешифровки используется число B. Найти число B, если известно число S.

Единственная строка входных данных содержит целое положительное число не длиннее 100 знаков – число S.

Выходные данные содержат искомое число B, или слово "Ошибка", если не существует чисел A, B, C, удовлетворяющих условию задачи.

Примеры

Исходные данные	**Результат**
111111111	37037037
1000000000	Ошибка
603360336033	201120112011

Решение. $S = A + B + C$. Поскольку $B - A = C - B$, то можно записать следующее уравнение $S = A + (A + x) + (A + 2 \cdot x)$. Отсюда $S = 3 \cdot A + 3 \cdot x$. Следовательно, $B = A + x = S / 3$. Поэтому если S делится на 3, то $B = S / 3$, а если не делится – решения нет.

```
алг Ключ()
нач
  цел s

  ввод s

  если s <= 3 то
    вывод 'Неверные исходные данные'
  иначе
    если s mod 3 <> 0 то
      вывод 'Ошибка'
    иначе
      вывод 'B = ', S div 3
    всё
  всё
кон
```

Задание 4

У прилавка в магазине выстроилась очередь из n покупателей. Время обслуживания продавцом i-го покупателя равно t_i ($i = 1, 2, ..., n$). Пусть даны натуральное n и действительные $t_1, ..., t_n$. Получить $c_1, ..., c_n$, где c_i – время пребывания i-го покупателя в очереди. Указать номер покупателя, для обслуживания которого продавцу потребовалось самое малое время.

Решение. $c_1 = t_1$, $min = t_1$. Далее в цикле от 2 до n вычисляем $c_i = c_{i-1} + t_i$. В этом же цикле вычисляем минимум из элементов t_i по обычному алгоритму.

```
алг Очередь()
нач
  цел n, i
  вещ t[n], c[n], min

  ввод n
  для i от 1 до n
  нц
    ввод t[i]
  кц

  c[1] = t[1]
  min  = t[1]
  для i от 2 до n
  нц
    c[i] = c[i – 1] + t[i]
    если t[i] < min то
      min = t[i]
    всё
  кц
кон
```

Задание 5

Разработайте алгоритм, который определяет (в порядке возрастания) номера разрядов, содержащих цифру 6 в десятичной записи числа 64^{513}.

Схема решения. Число 64^{513} слишком большое для того, чтобы его можно было точно представить в современном компьютере. Поэтому для решения задачи необходимо разработать способ хранения чисел нужной длины и алгоритм умножения чисел в новом представлении. Для представления чисел можно использовать массив, каждый элемент которого хранит одну цифру числа. После нахождения значения 64^{513} можно будет просмотреть этот массив и найти номера разрядов, содержащих цифру 2. Для умножения чисел реализуем школьный алгоритм умножения «в столбик». Первое число надо умножить на каждую цифру второго числа, начиная с последней, и сложить результаты, учитывая, что результат умножения на очередную цифру надо сдвигать влево на соответствующее количество позиций. Для этого берётся первый сомножитель, помещается в промежуточную переменную, и каждая цифра первого сомножителя умножается на текущую цифру второго сомножителя. После умножения

очередной цифры надо прибавить перенос из предыдущего разряда и вычислить собственно новую цифру как остаток от деления на 10 и число для переноса в следующий разряд как результат от деления на 10. Для ускорения работы алгоритма желательно отдельно рассмотреть случаи умножения числа на цифру 0 и цифру 1. После умножения выполняем сдвиг на нужное количество позиций и прибавляем результат в общую сумму. Для сложения чисел будем последовательно складывать цифры, начиная с последней. Если сумма двух цифр и переноса из предыдущего разряда больше или равна 10, то оставляем значение (<цифра$_1$> + <цифра$_2$> + <перенос> – 10) и запоминаем наличие переноса в следующий разряд.

Решение варианта 7992 для 9 классов

Задание 1

Как известно, современная система измерения времени ведёт начало от древнего Вавилона, где использовались 60-ричная с.с. Разработайте алгоритм перевода натуральных чисел из десятичной с.с. в 60-ричную с.с. Каждая цифра 60-ричной с.с. записывается в десятичной системе в круглых скобках, например, (21).

Решение. Для хранения чисел будем использовать массив, каждый элемент которого будет хранить одну цифру числа в 60-ричной системе счисления. Для перевода будем использовать обычный алгоритм – делим число в десятичной системе счисления на 60 и записываем полученные цифры.

```
число = запись
  цел digits[100]                                  // Массив цифр
  цел count                                        // Количество цифр в числе
кон

алг ПереводЧислаВ60Систему()
нач
  цел n, tr, i
  число m

  ввод n

  count = 0
  выполнить
    m.digits[count] = n mod 60
    count = count + 1
    n = n div 60
  до n = 0 или count > 100

  если count > 100 то
    вывод 'Слишком большое число'
  иначе
    для i от count + 1 до 100
    нц
      m.digits[i] = 0
    кц
    для i от count до 1 шаг -1
    нц
      вывод '(', m.digits[i], ')'
    кц
  всё
кон
```

Задание 2

Числа Сабита – натуральные числа, задающиеся формулой $3 \cdot 2^n - 1$ для неотрицательных n. Разработайте алгоритм нахождения суммы чисел Сабита в диапазоне от P до Q.

Решение. Перебираем значения n, начиная с 0. Ищем первое число n, для которого выражение $3 \cdot 2^n - 1$ будет больше или равно P. Используем полученное число как начальное значение суммы. Для следующих n, пока значение выражения $3 \cdot 2^n - 1$ будет меньше или равно Q, суммируем полученные числа Сабита.

```
алг ЧислаСабита()
нач
  цел p, q, n, s, sum

  ввод p, q

  n = 0
  s = 3 * pow(2, n) - 1
  пока s < p
  нц
    n = n + 1
```

```
    s = 3 * pow(2, n) – 1
  кц

  sum = s
  n = n + 1
  s = 3 * pow(2, n) – 1
  пока s <= q
  нц
    sum = sum + s
    n = n + 1
    s = 3 * pow(2, n) – 1
  кц
кон
```

Задание 3

В основе алгоритма RSA лежит использование пары простых чисел P и Q и производного числа (модуля) $N = P * Q$. Простое число – это натуральное число, которое имеет ровно два различных натуральных делителя: единицу и самого себя.

Принципиальным отличием нового алгоритма RSA++ от алгоритма RSA состоит в выборе ключей. Если в алгоритме RSA требуется пара простых чисел P и Q, то в алгоритме RSA++ числа P и Q должны быть взаимно простыми, т.е. они имеют только один общий делитель, равный 1.

Для анализа надёжности нового алгоритма необходимо узнать количество различных пар чисел P и Q, таких, что $1 < P < Q$ и соответствующий им модуль удовлетворяет условию $N \le K$.

Первая строка входных данных содержит одно целое число K ($1 \le K \le 109$).

Результат должен содержать одно целое число – количество различных пар чисел P и Q.

Примеры

Входные данные	**Результат**
12	3
18	6

Решение. Число P меняется в диапазоне от 2 до $\sqrt{K}$. Число Q меняется в диапазоне от $P + 1$ до K / P. Перебираем все возможные значения P и соответствующие им значения Q. Если числа являются взаимно простыми, прибавляем единицу к итоговому результату.

Для проверки, что числа являются взаимно простыми, надо найти НОД по алгоритму Евклида. Если НОД равен 1, то числа являются взаимно простыми.

```
алг АнализНадёжности()
нач
  цел k, p, q, n

  ввод k

  если k < 1 или k > 109 то
    вывод 'Неверное значение K'
  иначе
    n = 0
    для p от 2 до целая_часть(sqrt(k))
    нц
      для q от p + 1 до k div p
      нц
        если НОД(p, q) = 1 то
          n = n + 1
        всё
      кц
    кц
    вывод 'Количество пар взаимно простых чисел равно ', n
```

```
  всё
кон

алг НОД(арг цел x, y)
нач
  пока x <> y
  нц
    если x > y то
      x = x - y
    иначе
      y = y - x
    всё
  кц
  вернуть x
кон
```

Задание 4

Службой Внешней разведки был перехвачен фрагмент двоичного сообщения

xxxx0101010100110101010001010101010101000101001001010000010101000101010101010100
01010000010xxxx,

где xxxx – неизвестное количество (т.е. не обязательно 4) неизвестных бит – утерянная часть сообщения.

Достоверно известно, что сообщение кодирует текст, записанный русскими буквами без пробелов и знаков препинания с использованием следующей кодировки в шестнадцатеричной системе исчисления:

А – 50_{16}

Б – 51_{16}

В – 52_{16}

Г – 53_{16}

Д – 54_{16}

Е – 55_{16}

При этом на каждый символ отводится по 8 бит.

Расшифруйте текст доступного дешифровке фрагмента сообщения, учитывая возможность наличия в начале и конце сообщения произвольного (меньшего, чем по 8) количества бит, оставшихся от утерянной части сообщения.

Решение. Поскольку количество потерянных бит неизвестно, возможно, что биты, относящиеся к какой-либо букве, могут начинаться не с начала известной части сообщения. Поэтому следует рассмотреть 8 вариантов. Берём первые 8 бит известной части сообщения, выводим соответствующую букву, затем берём следующие 8 бит и т.д. Если последняя часть будет содержать меньше 8 бит, отбрасываем эту часть. Затем аналогично рассматриваем сообщение, в котором условно отброшен один бит, два бита и т.д. Таким образом, мы получим 8 вариантов расшифровки, и человек сможет определить, какой вариант является осмысленным.

```
алг Расшифровка()
нач
  строка s = '0101010100110101010001010101010101000101001001010000010101000101010101010100010100000010'
  строка res
  цел i, j, k, n

  для i от 1 до 8
  нц
    res = ''
    j = i
    пока j + 8 <= Length(s)
    нц
```

```
      n = 0
      для k от 0 до 7
      нц
        n = n * 2
        если str[k + j] = '1' то
          n = n + 1
        всё
      кц
      res = res + chr(n – 0x50 + ord('A'))      // 0x50 – число, записанное в 16-ричной системе счисления
      j = j + 8
    кц
    вывод res
  кц
кон
```

Задание 5

Разработайте алгоритм, который определяет (в порядке убывания) номера разрядов, содержащих цифру 8 в десятичной записи числа 64^{216}.

Схема решения. Число 64^{216} слишком большое для того, чтобы его можно было точно представить в современном компьютере. Поэтому для решения задачи необходимо разработать способ хранения чисел нужной длины и алгоритм умножения чисел в новом представлении. Для представления чисел можно использовать массив, каждый элемент которого хранит одну цифру числа. После нахождения значения 64^{216} можно будет просмотреть этот массив и найти номера разрядов, содержащих цифру 8. Для умножения чисел реализуем школьный алгоритм умножения «в столбик». Первое число надо умножить на каждую цифру второго числа, начиная с последней, и сложить результаты, учитывая, что результат умножения на очередную цифру надо сдвигать влево на соответствующее количество позиций. Для этого берётся первый сомножитель, помещается в промежуточную переменную, и каждая цифра первого сомножителя умножается на текущую цифру второго сомножителя. После умножения очередной цифры надо прибавить перенос из предыдущего разряда и вычислить собственно новую цифру как остаток от деления на 10 и число для переноса в следующий разряд как результат от деления на 10. Для ускорения работы алгоритма желательно отдельно рассмотреть случаи умножения числа на цифру 0 и цифру 1. После умножения выполняем сдвиг на нужное количество позиций и прибавляем результат в общую сумму. Для сложения чисел будем последовательно складывать цифры, начиная с последней. Если сумма двух цифр и переноса из предыдущего разряда больше или равна 10, то оставляем значение (<цифра$_1$> + <цифра$_2$> + <перенос> – 10) и запоминаем наличие переноса в следующий разряд.