Учреждение образования
«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Н. В. Пацей
В. А. Пласковицкий

Программирование повышенной сложности

Сборник задач по комбинаторике для студентов и магистрантов всех специальностей

Минск 2014

УДК 004.415.25(076.1)
ББК 32.973-01я73
П 21

Рассмотрен и рекомендован редакционно-издательским советом
Белорусского государственного технологического университета

Рецензенты:
кандидат технических наук, доцент кафедры систем автоматизированного проектирования Белорусского национального технического университета *В. Т. Придухо*;
кандидат технических наук, доцент кафедры автоматизации производственных процессов и электротехники Белорусского государственного технологического университета *Д. А. Гринюк*

Пацей, Н. В.

П 21 Программирование повышенной сложности : сб. задач по комбинаторике для студентов и магистрантов всех специальностей / Н. В. Пацей, В. А. Пласковицкий. – Минск : БГТУ, 2014. – 44 с.

Сборник задач предназначен для студентов и магистрантов, изучающих программирование и стремящихся достичь профессионального уровня. Составлен на основе олимпиадных задач по программированию из раздела «Комбинаторика».

В сборнике представлены краткие теоретические сведения по комбинаторике, рассмотрены отдельные решения задач с их анализом и реализацией на языке C++. Содержит раздел с задачами для самостоятельного решения с условиями в формате олимпиады ACM/ICPC.

УДК 004.415.25(076.1)
ББК 32.973-01я73

ПРЕДИСЛОВИЕ

Международная студенческая олимпиада по программированию (принято сокращение ACM/ICPC или просто ICPC – International Collegiate Programming Contest) – крупнейшая студенческая командная олимпиада по программированию.

Олимпиада проводится на двух уровнях. Первый – региональный чемпионат. На этом уровне участвуют команды высших учебных заведений региона. Победитель регионального чемпионата может участвовать в финале мирового первенства ACM/ICPC.

Международный командный чемпионат по программированию уходит своими корнями в соревнование, проводившееся в Техасском университете в 1970 г. С 1977 по 1989 г. в олимпиаде участвовали преимущественно команды вузов из США и Канады. К настоящему моменту олимпиада превратилась во всемирное соревнование: в 2009 г. в ней приняли участие 7109 команд из 88 стран. Количество команд продолжает расти ежегодно. С 2006 г. БГТУ также участвует в командном студенческом первенстве мира по программированию ACM/ICPC.

Однако задачи по программированию, которые студенты решают на подобных соревнованиях, выходят за рамки читаемых в вузе дисциплин. Сборник написан с целью восполнить пробелы в данном направлении и ориентирован на участников:

- ✓ командных чемпионатов мира по программированию для студентов, проводимых под эгидой ACM;
- ✓ региональных олимпиад по информатике для студентов и школьников;
- ✓ вузовских олимпиад по программированию.

Материалом для сборника послужили задачи международных и национальных олимпиад по информатике и программированию для школьников и студентов [1–20].

ВВЕДЕНИЕ

Олимпиады бывают личные и командные. В командных олимпиадах обычно участвуют три человека и им на все время олимпиады дается один компьютер для решения задач.

Участникам предлагается некоторый набор задач (10–12) различного уровня сложности. Продолжительность тура – 5 часов. Решением задачи является программа, написанная на одном из допустимых языков программирования. Эта программа должна корректно считывать любые входные данные указанного формата из определенного входного потока, корректно обрабатывать их согласно условию задачи и выводить в определенный выходной поток в указанном виде.

Для ввода-вывода могут использоваться как стандартные консольные потоки, так и файловые (часто входной файл имеет имя «input.txt», а выходной – «output.txt»).

Все решения проверяются автоматизированной тестирующей системой. Она запускает каждое решение на некотором наборе тестов.

После завершения работы программы она оценивает правильность полученных выходных данных – сравнивает с эталонными или производит более сложные действия.

Побеждает команда, решившая правильно наибольшее число задач. Если несколько команд решают одинаковое количество задач, то их положение в рейтинге определяется штрафным временем. Изначально штрафное время каждой команды равно нулю. За каждую правильно сданную задачу к штрафному времени команды прибавляется время, прошедшее с момента начала соревнования до момента сдачи задачи. Если зачетной попытке предшествовало несколько неудачных попыток сдать эту задачу, то за каждую из них к штрафному времени прибавляют двадцать минут.

Особенность олимпиадных задач – художественность их условия. В условиях речь идет не о структурах данных и алгоритмах, приводящих к решению. Чаще условие задачи представляет собой короткий рассказ с сюжетом, героями и конфликтом.

Таким образом, чтобы решить олимпиадную задачу, нужно составить математическую модель событий и уже по ней построить подхо-

дящий алгоритм. Алгоритм может быть как одним из известных, так и абсолютно новым.

В данном сборнике рассматриваются задачи и примеры их решения с помощью комбинаторики.

При решении практических задач часто приходиться делать выбор из некоторого конечного множества объектов, обладающих теми или иными свойствами, размещать элементы множеств в определенном порядке и т.д. Эти задачи принято называть *комбинаторными задачами* [1, 2].

Некоторые комбинаторные объекты так часто встречаются и настолько важны, что имеют собственные названия и приведены в Приложении1.

1. РАЗБОР ТИПОВЫХ ЗАДАЧ

Задача 1. Набор команд

Условие. В олимпиаде по программированию может участвовать команда из трех студентов группы. Сколько возможностей составить команду, если в группе 20 студентов?

Решение. $C_{20}^{3} = \frac{A_{20}^{3}}{3!} = \frac{20 \cdot 19 \cdot 18}{3 \cdot 2 \cdot 1} = 1140$. Т. е. существует 1140 возможных команд на олимпиаду по программированию из трех студентов этой группы.

Задача 2. Дележ билетов

Условие. Есть по одному билету в театр, в цирк и на концерт. Сколькими способами их можно распределить между четырьмя студентами (если каждый студент может получить сколько угодно билетов)?

Решение. $\overline{A}_{4}^{3} = 4^{3} = 64$. Т. е. существует 64 способа распределения билетов между студентами.

Задача 3. Выбор открыток

Пример. На почте пять видов открыток к Новому году. Сколькими способами из них можно выбрать семь открыток?

Решение. $\overline{C}_{5}^{7} = C_{5+7-1}^{7} = C_{11}^{7} = \frac{11 \cdot 10 \cdot 9 \cdot 8}{4!} = 330$. Т. е. существует 330 возможностей составить набор из семи поздравительных открыток к Новому году.

Задача 4. Счастливые номера

В одном городе с P-ричной системой счисления номера троллейбусных билетов состоят из $2K$ цифр. Билет считается счастливым, если сумма первых K разрядов равна сумме последних K разрядов. Определить, сколько существует счастливых билетов.

Реализация

```
1:  long int countFortuneNumbers(int p, int k){
2:
3:    long int **N = new long int*[2];
4:    for(int i=0;i<2;i++) N[i]=new long int[p*k];
5:
6:    for(int s_= 0; s_ < p; s_++) N[0][s_] = 1;
```

```
 7:
 8:     for(int k_=2; k_ <= k; k_++)
 9:     {
10:             for(int s=0; s<=k_*(p-1);s++)
11:             {
12:                     N[1][s] = 0;
13:                     for(int s_= 0; s_ < p; s_++)
14:                     {
15:                             if((s-s_  >=0)  &&  (s-s_)  <=  (k_-1)*(p-1))  N[1][s]  =
   N[1][s]+N[0][s-s_];
16:                     }
17:             }
18:
19:             for(int i=0;i<p*k;i++) N[0][i] = N[1][i];
20:     }
21:     long int N_tot = 0;
22:
23:     for(int s=0;s<=k*(p-1);s++) N_tot = N_tot + N[0][s]*N[0][s];
24:
25:     return N_tot;
26:   }
27:
28:   int _tmain(int argc, _TCHAR* argv[])
29:   {
30:     int p=0,k=0;
31:     scanf_s("%d",&p);
32:     scanf_s("%d",&k);
33:     printf("%ld",countFortuneNumbers(p,k));
34:
35:     return 0;
36:   }
```

Задача 5. Торговые точки. Числа Фибоначчи

Условие. Вдоль дороги, разбитой на N участков, решили разместить торговые точки. При этом, чтобы избежать конфликтов между торговцами, никакие две торговые точки не должны размещаться на соседних (смежных) участках. Найти количество возможных вариантов их расположения, если вдоль дороги должна находиться хотя бы одна торговая точка.

Решение. Представим участок в виде нулей (0) и единиц (1), где 1 – участок с торговой точкой, 0 – пустой участок. Участок длины N, не содержащий двух единиц подряд, может начинаться либо с 1, либо с 0. Если начинается с 0, то дальше подходит любая последовательность длины N–1, не содержащая двух единиц подряд. Если начинается с 1, то следующее значение должно быть обязательно 0, после которого может опять идти любая последовательность длины N–2. Пусть $F(N)$ – число возможных комбинаций для участка из N отрезков, тогда $F(N) = F(N–1) + F(N–2)$. Таким образом, получается последовательность Фибоначчи, где $F(1) = 1$, $F(2) = 2$ – рассчитаны вручную.

Пример. $N = 3$, ответ: 3, $N = 8$, ответ 34.

Другие вариации задач на числа Фибоначчи:

- количество способов подняться по лестнице, состоящей из N ступенек, если можно подниматься на следующую ступеньку либо через одну, равно F_{N+1};
- количество способов построить стену высотой 2 и длиной N с помощью кирпича со сторонами 2 и 1 равно F_{N+1};
- количество листьев AVL дерева (сбалансированное бинарное дерево) высотой N равно F_N;
- количество вариантов прохождения луча, направленного на две соединенные пластины стекла, если направление луча меняется N раз, равно F_{N+2}.

Задача 6. Расстановка скобок. Числа Каталана

Условие. Сколько существует способов расстановки N пар скобок таким образом, чтобы справа от каждой открывающейся скобки была своя закрывающаяся?

Решение. При правильной расстановке самой левой (первой) открывающейся скобке будет соответствовать закрывающаяся скобка, стоящая на позиции R, которая разбивает последовательность на две правильные части. При этом если левая часть будет содержать K пар скобок, то правая $N–K–1$ пар соответственно. Перебор всех возможных позиций R приводит к формуле для чисел Каталана: $C_N = \sum_{R=0}^{N-1} C_i C_{N-R-1}$.

Другие вариации задач на числа Каталана:

- количество бинарных деревьев с количеством листьев, равным N равняется C_N;
- количество маршрутов из одной вершины квадрата $N \times N$ в другую при неубывающем движении и не пересекая соответствующую диагональ равно C_{N+1};
- количество способов разбить выпуклый многоугольник из N вершин на треугольники с помощью соединения между собой отдельных вершин (проведения диагоналей) равно C_N;
- количество способов соединить по парам $2N$ точек на окружности равно C_{2N};

Всего существует более 60 различных конструкций, которые содержат числа Каталана.

Задача 7. Последовательность Соломона Голомба

Условие. Неубывающая последовательность, в которой элемент под номером *k* встречается ровно *f(k)* раз. При этом запись осуществляется следующим образом:

Номер элемента	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	…
f(k)	1	2	2	3	3	4	4	4	5	5	5	6	6	6	6	7	7	…

Т. е. если под числом 4 записано число 3, то в последовательности число 4 идет три раза подряд. Какое число будет стоять в данной последовательности на позиции *N*?

Реализация:

```
#include <iostream>
#include <vector>
#include <algorithm>
#include <queue>
void self_describing_by_iterate2(long int VALUE){
    int FN=0;
    queue < int > sequence;
    int n = 3, total = 2, index = 0;
    if (VALUE <= 2) {  FN = VALUE;  goto out;  }
    sequence.push(2);
    // пока не достигнем последнего значения
    while (total < VALUE){
        // получить текущее соответствие
        int fn = sequence.front();
        for (int c = 1; c <= fn; c++)
            sequence.push(n);

        // увеличить предел определяемых значений
        total += fn * n;

        if (VALUE == n)  {  FN = fn;  goto out;  }
        sequence.pop();
        n++;
    }
    total = n + sequence.size();
    while (sequence.size())  {
        int fn = sequence.front();
        if (VALUE == n)  {  FN = fn;  goto out;  }
        if (total <= VALUE && VALUE <= (total + fn - 1))
        {
            FN = n; goto out;
        }
        total += fn;
        sequence.pop();
        n++;
    }
out:  cout << FN << endl;
}
int _tmain(int argc, _TCHAR* argv[])
{
```

```
41:     long int VALUE=10;
42:     cin>>VALUE;
43:     self_describing_by_iterate2(VALUE);
44:     return 0;
45: }
```

Задача 8. Вывод перестановок заданного набора

Условие. Вывести на экран все перестановки последовательности длиной N, состоящей из нулей и единиц, если единиц должно быть ровно R. Ограничения $0 < R < N < 20$.

Реализация:

```
 1: bool genNextComb(int *c, int n)
 2: {
 3:   int k=0;
 4:
 5:   while(c[k] != 1) k++;
 6:
 7:   int p = 0;
 8:
 9:   for(int m=k+1; m < n && c[m]==1; m++, p++);
10:
11:   if((p+k) == n-1) return 0;
12:
13:   c[k+p+1]=1;
14:
15:   for(int m = k + p; m >= k; m--) c[m] = 0;
16:   for(int m = 0; m<p; m++) c[m] = 1;
17:
18:   return 1;
19: }
20:
21: int _tmain(int argc, _TCHAR* argv[])
22: {
23:
24:
25:   int n=10, r=2;
26:   int *c=new int[n];
27:   for(int i=0;i<r;i++) c[i]=1;
28:   for(int i=r;i<n;i++) c[i]=0;
29:
30:   do{
31:          for(int i = 0; i < n; i++) cout<<c[i];
32:          cout<<endl;
33:   } while(nexcom2(c, n));
34:
35:   return 0;
36: }
```

Задача 9. Разложение числа с условием

Условие. Разложить натуральное число A на сумму натуральных чисел, дающих максимальное произведение.

Решение. Математическое правило гласит: максимальное произведение двух слагаемых получается при делении числа пополам. Поэтому

алгоритм можно свести к рекурсивному делению числа на две равные части или максимально равные, в случае деления нечетного числа.

Пример. Дано число 7. Вариант разложения: 2 + 2 + 3 = 7. Произведение: 2 · 2 · 3 = 12.

Реализация:

```
1:  //Функция получает число и возвращает максимальное произведение его слагаемых
2:  int getMax(int val){
3:
4:          int max, left, right;
5:
6:    //число меньше 4 раскладывать на составляющие не надо
7:    if(val < 4)
8:    {
9:           max = val;
10:          //также здесь можно сохранить значение
11:          // очередного полученного слагаемого либо сразу вывести
12:            //в поток (консоль, файл и т. д.)
13:   }
14:   else
15:   {
16:          //если число четное, то деление пополам можно
17:            // осуществлять делением на 2
18:          if(val % 2 == 0)
19:          {
20:                 left = right = val / 2;
21:          }
22:          else
23:          {
24:          //если число нечетное, то нужно сделать его четным,
25:            //а потом внести поправки
26:                 left = right = (val - 1) / 2;
27:                 right++;
28:          }
29:          //запустить разложение каждой из частей
30:            //и получить результат их произведения
31:          max = getMax(left) * getMax(right);
32:   }
33:
34:   return max;
35: }
```

Хранить полученные слагаемые можно в динамическом массиве (размер ограничен значением $A/2$) либо векторе (см. приложение 1.).

Задача 10. Разложение числа на слагаемые

Условие. Составить все возможные варианты комбинаций слагаемых, на которые можно разложить натуральное число A. Порядок их расположения не учитывается.

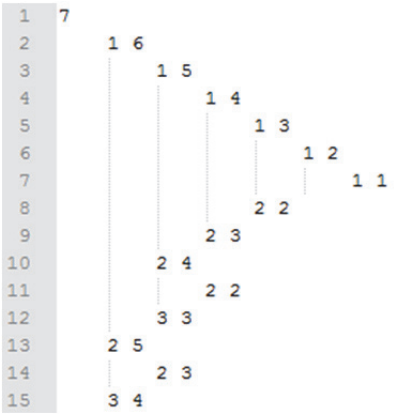
Пример:

Дано число: 4.

Варианты разложения:

4
2+2
1+3
1+1+2
1+1+1+1

Решение. Для генерации уникальных последовательностей можно воспользоваться следующим алгоритмом:



Чтобы получить из данной структуры все последовательности, сначала выводится последовательность, получаемая с помощью текущей пары, например: 1 + 6.

Потом последовательности, получаемые подстановкой вместо второго слагаемого вложенных пар:

1 + 1 + 5
1 + 2 + 4
1 + 3 + 3

Поскольку каждая из этих пар тоже имеет вложенные пары, то они тоже выводятся:

1 + 1 + 5
1 + 1 + 1 + 4
1 + 1 + 1 + 1 + 3
1 + 1 + 1 + 1 + 1 + 2
1 + 1 + 1 + 1 + 1 + 1 + 1
1 + 1 + 2 + 3
1 + 2 + 4

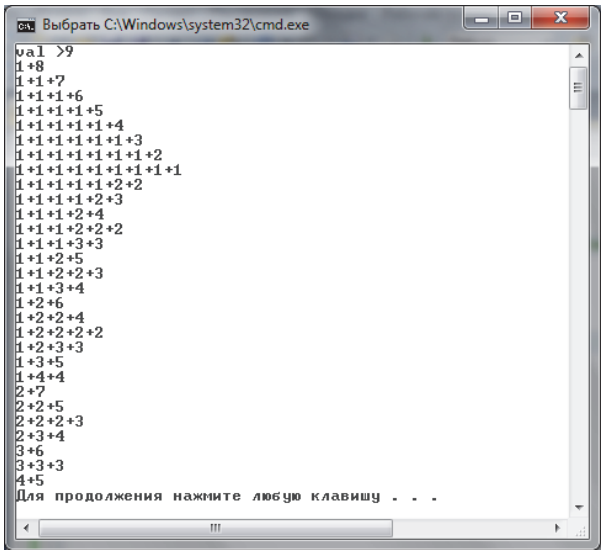
Правило разбиения описывается следующим алгоритмом: число N раскладывается на пары чисел (i; $N - i$). Затем по такому же принципу раскладывается второе слагаемое каждой пары. Изначально $i = 1$, за-

тем *i* равно первому слагаемому пары. Разбиение осуществляется до тех пор, пока $i <= N - i$.

Реализация:

```
#include "stdafx.h"
#include <iostream>
#include <string>

using namespace std;

void decomp(string str, int number, int start)
{
  //разделитель чисел
  string pl="+";

  //если не нужно выводить начальное число
  if(str!="")
        cout<< str+pl+to_string((long long)number)<<endl;
     //вывод текущей пары
  else pl="";

  //перебор вложенных пар
  for(int i=start;i<=number/2;i++)
        decomp(str+pl+to_string((long long)i), number-i, i);
}

int main()
{
  int val;
  cout << "val >";
  cin >> val;
  decomp("", val, 1);

  return 0;
}
```

Результат работы:



Проблема: один из недостатков предложенного алгоритма – способ хранения генерируемых последовательностей. Для этого используется строковая переменная, копии которой создаются при каждом вызове рекурсивной функции.

Решение проблемы: исправить это можно, сделав строковую переменную глобальной, а в функцию добавить еще один параметр, следящий за текущей позицией в строке, чтобы не захватывать лишний хвост, образовавшийся во время генерации более глубокого уровня вложенности.

Проблема: в C++ тип `string` не является внутренним и реализован в виде отдельного класса. И хоть его использование часто значительно облегчает реализацию алгоритма, это накладывает дополнительные расходы на используемую память, а также время на преобразование к строке других форматов.

Решение проблемы: использование вместо строковой переменной массива чисел.

Проблема: в алгоритме постоянно проверяется, происходит ли текущий вызов функции для начального числа. Такая проверка нужна, если необходимо убрать из вывода начальное число, а также корректно расставить разделитель между числами.

Решение проблемы: вынести разложение первого числа из рекурсивной функции.

Вариант реализации программы с учетом перечисленных моментов:

```
#include "stdafx.h"
#include <iostream>
using namespace std;

int *m;

void decomp2(int number, int start, int pos)
{
  for(int i=0;i<pos;i++) cout<<m[i]<<"+";
  cout<<number<<endl;

  for(int i=start;i<=number/2;i++)
  {
         m[pos]=i;
         decomp2(number-i, i, pos+1);
  }
}

int main()
{

  int val;
  cout << "val >";
  cin >> val;
```

```
25:
26:     m = new int[val];
27:
28:     for(int i=1;i<=val-i;i++)
29:     {
30:             m[0]=i;
31:             decomp2(val-i, i, 1);
32:     }
33:
34:     delete []m;
35:
36:     return 0;
37:   }
```

Проблема: если раньше во время работы функции уже сразу генерировалась пригодная для вывода строка, то теперь ее каждый раз приходится собирать с нуля и отправлять в поток вывода по частям. Это существенно снижает скорость работы алгоритма, в котором необходим вывод данных, т. е. практически в любом.

Решение проблемы: использовать для хранения символьный массив.

Проблема: работа с динамической памятью всегда происходит медленнее, чем со статической.

Решение проблемы: если в задаче стоит ограничение на поступающие данные, зачастую имеет смысл использовать статический массив приемлемого размера.

Проблема: в цикле `for(int i = start; i <= number / 2; i++)` происходит постоянное выполнение операции деление на 2.

Решение проблемы: производить разовое вычисление ограничивающего значения. Использовать для деления на 2 операцию битового сдвига.

Вариант реализации программы с учетом перечисленных моментов:

```
 1:  #include "stdafx.h"
 2:  #include <iostream>
 3:  using namespace std;
 4:
 5:  char m[1000];
 6:
 7:  void decomp3(int number, int start, int pos)
 8:  {
 9:    _itoa_s(number, m+pos, sizeof(int),10);
10:    int pos2 = strlen(m);
11:
12:    m[pos2] = '\n';
13:    m[pos2+1] = 0;
14:    printf(m);
15:
16:    for(int i=start, lim = number/2; i<=lim; i++)
17:    {
18:            _itoa_s(i,m+pos,sizeof(int),10);
19:            pos2 = strlen(m);
```

```
            m[pos2]='+';
            decomp3(number-i, i, pos2+1);
    }

}

int main()
{
int val;
    cout << "val >";
    cin >> val;

    for(int i=1,lim = val>>1; i<=lim; i++)
    {
            _itoa_s(i, m, sizeof(int), 10);
            int len = strlen(m);
            m[len] = '+';
            decomp3(val-i, i, len+1);
    }

    return 0;
}
```

Попробуйте изменить функцию `decomp3` так, чтобы код `main()` можно было сократить до вида:

```
int main()
{
    int val;
    cout << "val >";
    cin >> val;
    decomp3(val,1,0);

return 0;
}
```

Другой подход к разложению числа на уникальные составляющие основан на идеи «размазывания» числа. Например:

```
7
      6+1
      5+2
      5+1+1
      4+3
      4+2+1
      4+1+1+1
      3+3+1
      3+2+2
      3+2+1+1
      3+1+1+1+1
      2+2+2+1
      2+2+1+1+1
      2+1+1+1+1+1
      1+1+1+1+1+1+1
```

Реализация:

```
int main()
{
  int val;
  cout << "val >";
  cin >> val;

  int *m = new int[val];

  int len; //длина текущей последовательности
  int pos; //позиция текущего элемента в последовательности
  int sourse; //ресурсы для распределения

  sourse = pos = len = 0;

  m[0] =  val;
  do
  {
        m[pos]--;
        sourse = 1 + len - pos;
        len = pos;
        while (sourse > 0)
        {
              pos++;
              if (sourse < m[pos - 1])
              {
                    m[pos] = sourse;
                    sourse = 0;
              }
              else
              {
                    m[pos] = m[pos - 1];
                    sourse -= m[pos];
              }
              len++;
        }
        for (int i = 0; i < len+1; i++)
              printf("%d%c",m[i],(i<len)?'+':'\n');
        for(pos=len; pos >-1 && m[pos] < 2; pos--);
  } while (pos>-1);

  delete [] m;
}
```

Чтобы подсчитать возможные варианты разложения натурального числа на слагаемые, можно использовать следующий алгоритм:

```
int val;
  cout << "val >";
  cin >> val;

  int *m = new int[++val];

  for (int i=0; i<val; i++) m[i] = 0;

  m[0] = 1;
  for (int i=1;i<val;i++)
  {

```

```
13:             for (int j=1;j<val;j++)
14:             {
15:                     if (j-i >= 0)  m[j] += m[j-i];
16:             }
17:     }
18:
19:     printf("%d",m[val-1]);
20:     delete [] m;
```

Для более общего случая, когда нужно найти количество разложений числа на заданный набор чисел, алгоритм примет вид:

```
 1:     int val;
 2:     cout << "val >";
 3:     cin >> val;
 4:
 5:     int *m=new int[++val];
 6:
 7:     //список ограничений
 8:     int list[]={1,3,4,5,8};
 9:     int size_list=5;
10:
11:     for (int i=0;i<val;i++) m[i]=0;
12:
13:     for (int i=0;i<size_list;i++)
14:     {
15:             m[0] = 1;
16:             for (int j=1;j<val;j++)
17:             {
18:                     if (j-list[i] >=0)  m[j] += m[j-list[i]];
19:             }
20:     }
21:
22:     printf("%d",m[val-1]);
23:     delete [] m;
```

Можно предложить еще вариант для подсчета уникальных разложений числа:

```
 1:  int n;
 2:    cin >> n;
 3:
 4:  int **d;
 5:   d = new int*[n+1];
 6:   for(int i=0;i<n+1;i++)
 7:             d[i] = new int [n+1];
 8:
 9:    d[0][0] = 1;
10:    for (int i = 1; i <= n; ++i)
11:           d[0][i] = 1, d[i][0] = 0;
12:    for (int i = 1; i <= n; ++i)
13:    {
14:           for (int j = 1; j <= i; ++j)
15:                   d[i][j] = d[i-j][j] + d[i][j-1];
16:           for (int j = i+1; j <= n; ++j)
17:                   d[i][j] = d[i][j-1];
18:    }
19:    cout << d[n][n];
20:    cin >> n;
```

2. ЗАДАЧИ ДЛЯ САМОСТОЯТЕЛЬНОГО РЕШЕНИЯ

Задача № 1

Шахматы

Требуется найти число способов расставить на шахматной доске $N \times N$ K ладей так, чтобы они не били друг друга. Все ладьи считаются одинаковыми.

Входные данные: файл input.txt

Первая строка ввода содержит натуральные числа N и K (N, $K \leq 8$).

Выходные данные: файл ouput.txt

Одна строка с целым числом.

Пример:

input.txt	ouput.txt
8 8	40320

Задача № 2

Дележ

В результате очередной хитроумной комбинации у главаря банды и его K компаньонов оказалось X рублей пятирублевыми банкнотами. И вот дело, как водится, дошло до дележа... Один предложил «по справедливости», т. е. всем поровну. Другой порешил себе взять половину, а остальным «по заслугам». Однако у главаря имелось свое мнение на этот счет... Ваша задача – найти количество способов разделить имеющиеся деньги между всеми участниками ограбления: K бандитами и главарем.

Входные данные: файл input.txt

Во входном файле на одной строке записаны целые числа X ($0 \leq X \leq 500$) и K ($0 \leq K \leq 100$). Естественно, что число X делится на 5. При дележе рвать пятирублевые банкноты не разрешается.

Выходные данные: файл ouput.txt

Одна строка с целым числом – количество способов дележа.

Пример:

input.txt	ouput.txt
15 2	10

Задача № 3

Фишки

Имеются разноцветные фишки K ($K \leq 25$) цветов, на каждой из которых записано любое целое неотрицательное число, выбранное из N ($N \leq 101$) заданных чисел. Набор из K фишек разных цветов назовем комплектом. Два комплекта считаются одинаковыми, если на любой паре фишек одинакового цвета числа совпадают. В противном случае комплекты считаются разными. Ценой комплекта назовем сумму чисел на фишках, входящих в его состав.

Вы должны определить количество различных комплектов, цена которых равна S ($S \leq 300$).

Входные данные: файл input.txt

Первая строка файла содержит значение K, вторая – S, третья – N. Следующие N строк содержат значения чисел (по одному в строке), которые написаны на фишках.

Выходные данные: файл ouput.txt

Состоит из одной строки. В этой строке содержится одно число – искомое количество комплектов.

Задача № 4

Разложение числа на одинаковые слагаемые

Дано целое положительное число, большее единицы и не превышающее N ($1 \leq N \leq 2\,000\,000\,000$). Необходимо найти H – любое натуральное число меньше N, которое может быть представлено суммой одинаковых чисел меньших H, при условии, что общее число сумм C будет максимальным. Суммы считаются различными, если одна из них составлена из одного числа, а другая – из другого. Например, для числа 6:

6 = 3+3

6 = 2+2+2

6 = 1+1+1+1+1+1

Входные данные: файл input.txt

Содержит единственное число – N.

Выходные данные: файл ouput.txt

Содержит числа H и C по одному в строке. Если задача имеет несколько решений, выведите любое из них.

Пример:

input.txt	ouput.txt
10	6 3
100	60 12

Задача № 5

Деление на 3

Дана последовательность чисел 1, 12, 123, …, 12345678910, …. Определите, сколько чисел среди первых N ($1 \le N \le 2^{31}-1$) элементов этой последовательности делятся на 3.

Входные данные: файл input.txt

Одно число – N.

Выходные данные: файл ouput.txt

Количество чисел – одно целое число.

Пример:

input.txt	ouput.txt
4	2

Задача № 6

Деление на 15

Дана строка, содержащая произвольные десятичные цифры. Длина строки – от 1 до 1000 символов. Используя цифры строки, получите максимально возможное число, которое делится на 15 без остатка. Каждый символ строки не может быть использован более одного раза.

Входные данные: файл input.txt

Единственная строка файла содержит исходную строку.

Выходные данные: файл ouput.txt

Содержит одну строку с максимальным числом (число не должно начинаться с нуля). Если число невозможно получить, запишите строку «нельзя».

Пример:

input.txt	ouput.txt
7	нельзя
02041	4200

Задача № 7

Максимальное произведение

Задано натуральное число A от 1 до 1000. Представить его в виде суммы натуральных чисел, перемножив которые получается макси-

мальное число. В ответ вывести получившееся произведение, а также используемые для этого числа (в формате, приведенном ниже).

Входные данные: файл input.txt

Одна строка содержащая число – *A*.

Выходные данные: файл ouput.txt

Получившееся произведение, знак равенства, слагаемые со знаком умножения «*» между ними, в порядке убывания.

Пример:

input.txt	ouput.txt
7	12=3*2*2

Задача № 8

Карточки

На день рождения Пете подарили набор карточек с буквами. Теперь Петя с большим интересом составляет из них разные слова. И вот однажды, составив очередное слово, Петя заинтересовался: «А сколько различных слов можно составить из тех же карточек, что и данное?». Помогите ему ответить на этот вопрос.

Входные данные: файл input.txt

Содержит заданное слово, составленное Петей, – строка из маленьких латинских букв не длиннее 15 символов.

Выходные данные: файл ouput.txt

Одно целое число – ответ на поставленную задачу.

Пример:

input.txt	ouput.txt
Solo	12

Задача № 9

Коллекционирование этикеток

Вася коллекционирует спичечные этикетки. Для этого у него есть N ($2 \le N \le 1000$) альбомов вместимостью K_1, K_2, …, K_N этикеток. Вася хочет, чтобы в случае утери одного любого альбома каждая этикетка осталась у него хотя бы в одном экземпляре. Для этого он покупает каждую этикетку в двух экземплярах и наклеивает их в два разных альбома. Какое максимальное количество различных этикеток при этом может оказаться в его коллекции?

Входные данные: файл input.txt

Первая строка файла содержит сначала число N – количество альбомов. Вторая строка – N чисел K_1, K_2, …, K_N, задающих вместимость

альбомов. N – натуральное число из диапазона от 2 до 1000. Вместимость каждого альбома задается натуральным числом, суммарная вместимость всех альбомов не превышает 100 000 этикеток.

Выходные данные: файл ouput.txt

Должен содержать число E – максимальное количество различных этикеток, которое может собрать Вася с соблюдением выдвинутого условия.

Пример:

input.txt	ouput.txt
4 1 2 1 1	2

Задача № 10

Зоопарк

В городском зоопарке содержатся животные N ($1 \le N \le 1000$) разных видов. Для участия в международной выставке «Три призера» зоопарк должен представить трех животных различных видов.

Требуется написать программу, которая вычислит число способов выбрать трех животных для участия в выставке.

Например, если в зоопарке два медведя, тигр, лев и пингвин, то есть семь способов выбрать трех животных:

1. первый медведь, тигр и лев;
2. первый медведь, тигр и пингвин;
3. первый медведь, лев и пингвин;
4. второй медведь, тигр и лев;
5. второй медведь, тигр и пингвин;
6. второй медведь, лев и пингвин;
7. тигр, лев и пингвин.

Входные данные: файл input.txt

Содержит в первой строке натуральное число N – количество видов животных в городском зоопарке. Во второй строке через пробел записаны N натуральных чисел – количество животных соответствующего вида. Число животных каждого вида не превышает 1000.

Выходные данные: файл ouput.txt

Должен содержать одно число – количество способов выбрать трех животных для международной выставки.

Пример:

input.txt	ouput.txt
4 2 1 1 1	7
3 100 100 100	1000000

Задача № 11

Салаты

Как-то раз, придя домой из школы, Света обнаружила записку от мамы, в которой она просила сделать салат. Света знала, что салат – это смесь двух или более ингредиентов, поэтому ей не составило труда выполнить мамину просьбу.

Но Света хочет стать математиком, поэтому для тренировки решила посчитать, сколько различных салатов она сможет сделать из имеющихся продуктов (майонез, огурцы, помидоры). После небольших расчетов она получила ответ: 4.

Напишите программу, которая определяет количество различных салатов для произвольного числа ингредиентов.

Входные данные: файл input.txt

Содержит натуральное число N – количество имеющихся ингредиентов ($N < 32$).

Выходные данные: файл ouput.txt

Одно число – количество различных салатов.

Пример:

input.txt	ouput.txt
3	4
4	11

Задача № 12

Несчастливые номера

Обычно автобусный билет с номером, состоящим из 6 цифр, считается счастливым, если сумма первых трех цифр его номера равна сумме трех последних. Вася очень любил получать счастливые билеты, однако это случалось не так часто. Поэтому для себя он изменил определение счастливого билета.

Счастливым он считал тот номер, сумма некоторых цифр которого равнялась сумме оставшихся цифр. В его представлении билет с номером 561743 счастливый, так как 5 + 1 + 4 + 3 = 6 + 7.

Вася вырос, но по привычке в номерах различных документов пытается найти признаки счастливого номера. Для этого он расширил свое определение счастливого номера на *N*-значные номера лицевых счетов и других документов, состоящих из цифр от 0 до *K*. Номер документа он называет счастливым, если сумма некоторых цифр этого номера равняется сумме оставшихся. Остальные номера для него несчастливые. К сожалению, несмотря на расширенное понимание «счастья», несчастливых номеров остается еще много...

Вам предлагается определить количество несчастливых *N*-значных номеров ($1 \le N \le 100$), которые можно составить, используя цифры от 0 до K ($1 \le K \le 9$, но $N \cdot K \le 300$). В номерах допускается любое количество ведущих нулей.

Входные данные: файл input.txt

Строка содержит описание вида номеров в виде двух чисел *N* и *K*, разделенных пробелом.

Выходные данные: файл ouput.txt

Выведите количество несчастливых номеров для заданных *N* и *K*.

Пример:

input.txt	ouput.txt
1 7	7
11 9	50184219171

Задача № 13

Волейбол

Партия в волейболе выигрывается командой, которая первой набирает 25 очков с преимуществом минимум в два очка. В случае равного счета 24:24 игра продолжается до достижения преимущества в два очка (26:24; 27:25).

Две сыгранные партии, закончившиеся с одинаковым счетом, будем считать разными, если строки, в которых выписан порядок набора очков командами, не равны.

Комитет по проведению соревнований по волейболу заинтересовался количеством различных партий, заканчивающихся счетом 25:23. Их оказалось 16123801841550.

Определить, сколько существует различных партий, заканчивающихся заданным счетом.

Входные данные: файл input.txt

Указан конечный счет в партии (то есть такой, при котором победа в партии отдается одной из команд). Также известно, что ни одна из команд не набрала более 40 очков.

Выходные данные: файл ouput.txt

Выведите количество всевозможных партий, которые заканчиваются данным счетом.

Пример:

input.txt	ouput.txt
25:12	1251677700
25:23	16123801841550

Задача № 14

Нолики

Для заданных натуральных чисел N и K требуется вычислить количество чисел от 1 до N, имеющих в двоичной записи ровно K нулей.

Например, если $N = 8$ и $K = 1$, то мы можем записать все числа от 1 до 8 в двоичной системе счисления:

1, 10, 11, 100, 101, 110, 111 и 1000.

Откуда видно, что только числа 10, 101 и 110 имеют ровно один ноль в записи, т. е. правильный ответ – 3.

Входные данные: файл input.txt

Единственная строка входного файла содержит два натуральных числа через пробел N и K, не превышающих 10^9.

Выходные данные: файл ouput.txt

Нужно вывести одно целое число – количество чисел от 1 до N с K нулями в двоичном представлении.

Пример:

input.txt	ouput.txt
13 2	4
1000 5	210

Задача № 15

Две кучки камней

У Вас есть N ($1 \le N \le 18$) камней с массами $W_1, W_2, \ldots W_N$ ($1 \le W_i \le 10^5$). Требуется разложить камни на две кучки так, чтобы разница масс этих кучек была минимальной.

Входные данные: файл input.txt

В первой строке входного файла записано число N – количество камней. Во второй строке через пробел перечислены массы камней $W_1, W_2, \ldots W_N$.

Выходные данные: файл ouput.txt

В единственную строку выходного файла нужно вывести одно неотрицательное целое число – минимально возможную разницу между массами двух кучек.

Пример:

input.txt	ouput.txt
5 5 8 13 27 14	3

Задача № 16

Новое домино

В казино был придуман специальный набор домино для VIP-клиентов. Обычные кости домино представляют собой набор из различных комбинаций двух плиток, на каждой из которых отображается от 0 до 6 точек. А этот набор представляет собой подобные сочетания плиток, но количество точек на каждой может быть от 0 до заданного значения, которое зависит от интеллектуального уровня игроков. В таком наборе костей присутствуют всевозможные сочетания плиток, но при этом ни одна из костей не повторяется (даже такие комбинации, как 2-5 и 5-2, считаются одинаковыми).

Перед изготовителем данного набора костей встала проблема вычисления суммарного количества точек на всех костях домино. Это связано с тем, что домино украшается бриллиантами, которые представляют собой точки на плитках, и при изготовлении необходимо оценить стоимость.

Помогите написать программу, которая решит эту задачу.

Входные данные: файл input.txt

Содержит одно натуральное число N – максимальное количество точек на одной плитке домино ($N \leq 10000$).

Выходные данные: файл ouput.txt

Выведите количество бриллиантовых камней, которые необходимо изготовить для заданного набора костей.

Пример:

input.txt	ouput.txt
2	12

Задача № 17

Гладкие числа

Назовем число *гладким*, если его цифры, начиная со старшего разряда, образуют неубывающую последовательность. Упорядочим все такие числа в возрастающем порядке и присвоим каждому номер.

Вам требуется по номеру N вывести N-ое гладкое число.

Входные данные: файл input.txt

Во входном файле содержится номер N ($1 \le N \le 2147483647$).

Выходные данные: файл ouput.txt

Должен содержать искомое N-е гладкое число.

Пример:

input.txt	ouput.txt
11	12
239	1135

Задача № 18

День рождения

Иван Иванович пригласил на свой день рождения много гостей. Он написал на карточках фамилии всех гостей и разложил эти карточки на столе, полагая, что каждый гость сядет там, где обнаружит карточку со своей фамилией (фамилии у всех гостей различны). Однако гости не обратили внимания на карточки и сели за стол в произвольном порядке. При этом Иван Иванович с удивлением обнаружил, что ни один гость не сел на предназначенное ему место.

Требуется написать программу, которая найдет, сколькими способами можно рассадить гостей так, чтобы ни один из них не сидел там, где лежала карточка с его фамилией.

Входные данные: файл input.txt

Записано целое число N – количество гостей ($1 \le N \le 100$).

Выходные данные: файл ouput.txt

Должен содержать одно целое число – количество способов рассадить гостей.

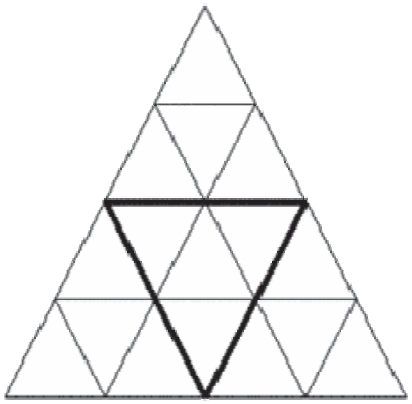
Пример:

input.txt	ouput.txt
1	0
20	895014631192902121

Задача № 19

Треугольники

Рассмотрим фигуру, аналогичную показанной на рисунке (большой равносторонний треугольник, составленный из маленьких равносторонних треугольников). На рисунке приведена фигура, состоящая из четырех уровней треугольников.



Требуется написать программу, которая будет определять, сколько всего в ней треугольников (необходимо учитывать не только «маленькие» треугольники, а вообще все треугольники – в частности, треугольник, выделенный жирным, а также вся фигура, являются интересующими нас треугольниками).

Входные данные: файл input.txt

Содержит одно число N – количество уровней в фигуре ($1 \le N \le 10^5$).

Выходные данные: файл ouput.txt

Должен содержать одно число – количество треугольников в такой фигуре.

Пример:

input.txt	ouput.txt
1	1
4	27

Задача № 20

Обмен валюты

Петя работает в обменном пункте в Димонии. Недавно он получил от начальства набор цифр для отображения обменного курса. К сожалению, набор содержит всего по две копии каждой цифры. Теперь Петя хочет узнать, сколько различных обменных курсов он сможет отобразить.

Петя обменивает димонские доллары на машландские тугрики. Петя уверен, что курс обмена будет целым числом, которое находится в диапазоне от L до R включительно.

Входные данные: файл input.txt

Содержит два целых числа L и R ($1 \le L \le R \le 10^{18}$) введенных через пробел.

Выходные данные: файл ouput.txt

Выведите одно целое число – количество обменных курсов, которые Петя может отобразить с использованием полученного набора.

Пример:

input.txt	ouput.txt
1 1000	990
1 100	100

Задача № 21

Кодовый замок

Компания «Замки и замки» недавно разработала новый тип кодового замка для размещения на воротах замков. Панель замка представляет собой прямоугольник шириной W ячеек и высотой H ячеек. В некоторых из них расположены кнопки.

Код на этом замке вводится одновременным нажатием k кнопок. Для того чтобы код было легче запомнить, используемые в нем кнопки должны образовывать связную область. Она называется связной, если из ее любой клетки можно добраться до любой другой, перемещаясь только между клетками области с общей стороной. Важным критерием надежности замка является число различных кодов, которые на нем можно набрать.

Для оценки надежности замков требуется написать программу для вычисления указанной величины.

Входные данные: файл input.txt

В первой строке входного файла находятся три целых числа H, W и k ($1 \le H, W \le 30$; $1 \le k \le 10$). Каждая из последующих H строк содержит W символов. Символ «#» обозначает кнопку, а «.» – ее отсутствие.

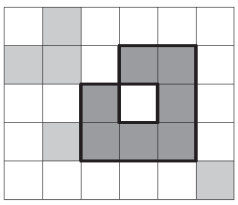
Выходные данные: файл ouput.txt

В выходной файл выведите единственное число – количество кодов, удовлетворяющих указанным требованиям.

Пример:

input.txt	ouput.txt
2 2 2 #. ##	2
5 6 7 .#.... ##.##. ..#.#. .####. .....#	3

На рисунке изображен один из возможных кодов для второго примера.



ПРИЛОЖЕНИЕ 1

БАЗОВЫЕ ФОРМУЛЫ КОМБИНАТОРИКИ

Название	Формула	Описание
Размещение	$A_N^K = \frac{n!}{(n-k)!}$	Количество различных размещений K элементов, взятых из множества N
Размещение с повторениями	$\overline{A}_N^K = N^K$	Размещение с возможностью повторного использования одних и тех же элементов
Перестановка	$P_N = A_N^N = N!$	Количество способов переставить N различных объектов
Сочетание (выборка)	$C_N^K = \frac{A_N^K}{K!} = \frac{N!}{K! \cdot (N-K)!}$ $C_N^K = C_{N-1}^{K-1} + C_{N-1}^K = \frac{N}{K} C_{N-1}^{K-1}$	Количество способов выбора K из N различных предметов (последовательность расположения элементов значения не имеет)
Сочетание с повторениями	$\overline{C}_N^K = C_{N+K-1}^K = \frac{(N+K-1)!}{K! \cdot (N-1)!}$	Сочетание с возможностью повторного использования одних и тех же элементов
Включение-исключение	$\left\lvert \bigcup_{i=1}^n X_i \right\rvert = \sum_i \lvert A_i \rvert + \sum_{i<j} \lvert A_i \cap A_j \rvert + \sum_{i<j<k} \lvert A_i \cap A_j \cap A_k \rvert + (-1)^n \cdot \lvert A_1 \cap A_2 \cap \ldots \cap A_n \rvert$	Количество неповторяющихся элементов в объединении, состоящем из ряда множеств $\{A_1, .., A_n\}$
Число Стирлинга первого рода (без знака)	$S(N,N) = 1,\ S(N,0) = 0$ $S(N,K) = S(N-1,K-1) + (N-1) \cdot S(N-1,K)$	Количество способов разбиения множества из N элементов на K непустых упорядоченных подмножеств (циклов)
Число Стирлинга второго рода	$S(0,0) = 1,\ S(N,0) = 0$ $S(N,K) = S(N-1,K-1) + K \cdot S(N-1,K)$ $S(N,K) = \sum_{i=k-1}^{N-1} C_{N-1}^i S(i,K-1);$ $S(N,K) = \frac{1}{K!} \sum_{i=1}^K C_k^i i^N (-1)^{K-i}$	Количество способов разбиения множества из N элементов на K непустых неупорядоченных подмножеств

Название	Формула	Описание
Число Белла	$B_0 = 1$ $B_N = \sum_{k=0}^{N} S(N,k);$ $B_{N+1} = \sum_{i=1}^{N} C_N^i B_i$	Количество способов разбиения множества из N элементов на все неупорядоченные подмножества
Число Фибоначчи	$F_0 = 0,\ F_1 = 1, F_n = F_{n-1} + F_{n-2}$ $F_n = \frac{\left(\frac{1+\sqrt{5}}{2}\right)^n - \left(\frac{1-\sqrt{5}}{2}\right)^n}{\sqrt{5}}$ $F_1 + F_2 + F_3 + \ldots + F_n = F_{n+2} - 1$ $F_1 + F_3 + F_5 + \ldots + F_{2n-1} = F_{2n}$ $F_2 + F_4 + F_6 + \ldots + F_{2n} = F_{2n+1} - 1$ $F_n = F_k F_{n-k+1} + F_{k-1} F_{n-l}$	Количество способов на текущем шаге является суммой способов двух предыдущих шагов
Число Каталана	$C_0 = 1,\quad C_n = \sum_{i=0}^{n-1} C_i C_{n-1-i} = \frac{(2n)!}{(n-1)(n!)^2}$ $C_n = \prod_{k=2}^{n} \frac{n+k}{k}$	Количество взаимных расстановок N пар при возможности расположения одних пар внутри других, а также количественные подсчеты других аналогичных показателей

ПРИЛОЖЕНИЕ 2

ЭФФЕКТИВНЫЕ РЕАЛИЗАЦИИ ОТДЕЛЬНЫХ КОМБИНАТОРНЫХ ВЕЛИЧИН

Подсчет сочетаний

$$C_N^M = \frac{N!}{M!(N-K)!}$$

Отдельное нахождение значений факториалов может привести к переполнению. Для того чтобы избежать переполнения, можно вычислять количество сочетаний по формуле $C_N^M = C_{N-1}^{M-1} + C_{N-1}^M$, используя рекурсию:

```
1:  int C(int m, int n)
2:  {
3:      if(n==0 || n==m) return 1;
4:      else return C(m-1, n-1) + C(m-1, n);
5:  }
```

либо динамическое программирование:

```
1:  int C(int m, int n)
2:  {
3:     int **B;
4:     B = new int*[m+1];
5:
6:     for(int i=0;i<=m;i++)
7:     {
8:       B[i] = new int[m+1];
9:       B[i][0]=1;
10:      B[i][i]=1;
11:      for(int j=1; j<i; j++)
12:      {
13:            B[i][j]=B[i-1][j-1]+B[i-1][j];
14:      }
15:    }
16:
17:    return B[m][n];
18: }
19:
```

либо пошаговое вычисление:

```
1:  int C(int m,int n)
2:  {
3:    if(n>m-n) n = m-n;
4:
```

```
  m = m + 1;

  int a = 1;

  for(int i=1; i<=n; i++) a = a * (m-i) / i;

  return a;
}
```

Модификация предыдущего алгоритма, позволяющая немного ускорить вычисление за счет памяти, а также предотвратить возможные переполнения при больших значениях:

```
int C(int m, int n)
{
    int p = (n<=m-n)?n:(m-n);
    n = m-p;

    int r = (p==0)?1:n+1;

    for(int i=2; i<=p; i++) r = (n+i)/i * r;

    return r;
}
```

Генератор последовательностей сочетаний

При решении ряда задач (например, по формуле включения-исключения) необходимо сгенерировать наборы R чисел из всех возможных сочетаний N элементов. Алгоритм, генерирующий все комбинации при заданных параметрах (если требуется найти сочетания из конкретного множества, то генерируемые числа можно использовать как индексы при обращении к массиву, хранящему нужные данные):

```

bool comb1(int *c, int n, int r, bool prim)
{
    if(prim){
           prim = false;
           for(int j=0;j<r;j++) c[j]=j;
           return prim;
    }

    if(c[r-1]<n-1){
           c[r-1]=c[r-1]+1; return prim;
    }

    for(int j = r-2; j > -1; j--){
           if(c[j]<n-r+j){
                  c[j]=c[j]+1;
                  for(int i=j+1;i<r;i++)c[i] = c[j]+i-j;
                  return prim;
```

```
            }
    }
    prim = true;
    return prim;
}

int _tmain(int argc, _TCHAR* argv[])
{

  int c[4];

  int r=3;
  int n=4;

  bool prim = true;

  do{
         prim = comb1(c,n,r,prim);
         if(!prim){
                for(int j=0;j<r;j++)cout<<c[j]+1;
                cout<<endl;
         }
  } while(!prim);

  return 0;
}
```

Подсчет числа Фибоначчи по номеру *n*

```
// простое сложение двух последних значений
int fib_n(int n)
{
        if (n <= 2) return 1;
        //F(n-2)
        int x = 1;
        //F(n-1)
        int y = 1;
        //F(n)
        int ans = 0;
        for (int i = 3; i <= n; i++)
        {
                ans = x + y;
                x = y;
                y = ans;
        }
        return ans;
}
```

Более быстрый способ:

```
//возведение матрицы в степень
int fib(int n)
{
  int a = 1, ta,
      b = 1, tb,
      c = 1, rc = 0,  tc,
      d = 0, rd = 1;
```

```
 8:
 9:    while (n)
10:    {
11:      if (n & 1)    // Если степень нечетная
12:      {
13:        // Умножение вектора R на матрицу A
14:        tc = rc;
15:        rc = rc*a + rd*c;
16:        rd = tc*b + rd*d;
17:      }
18:
19:      // Умножение матрицы A на саму себя
20:      ta = a; tb = b; tc = c;
21:      a = a*a  + b*c;
22:      b = ta*b + b*d;
23:      c = c*ta + d*c;
24:      d = tc*tb+ d*d;
25:
26:      n >>= 1;  // Уменьшение степени вдвое
27:
28:    }
29:    return rc;
30:  }
```

Вычисление числа Каталана

```
 1:  int getCatalan(int N)
 2:    {
 3:      int * C = new int[N+1];
 4:      C[0] = 1;
 5:      for(int i=1; i<=N; i++)
 6:      {
 7:         C[i] = 0;
 8:         for(int j=0; j<i; j++)
 9:           {
10:            C[i] += C[j]*C[i-j-1];
11:         }
12:      }
13:
14:      return C[N-1];
15:    }
```

ПРИЛОЖЕНИЕ 3

РЕАЛИЗАЦИЯ ЧАСТО ИСПОЛЬЗУЕМЫХ ОПЕРАЦИЙ НА ЯЗЫКЕ С++

Типичные библиотеки и настройки:

```
1:  #include <iostream> // для работы с потоком ввода/вывода (например, cout, cin)
2:  #include <string> // для работы со строками
3:  #include <vector> // для работы с векторами
4:  #include <time.h> // для замера и других операций со временем
5:  #include <algorithm> // для использования дополнительных возможностей при работе
    с  объектами STL (например, сортировки)
6:  using namespace std; // для обращения к объектам без указания имен пространств
7:  setlocale (LC_ALL,".1251"); // для вывода на консоль русскоязычных символов
```

Способы хранения данных

Динамический массив:

```
1:  int size = 100; //определение размера массива
2:  int *mas = new int[size]; // динамическое выделение памяти
3:
4:    for(int i = 0; i < size; i++)
5:       {
6:           //mas[i] - доступ к элементам
7:       }
8:  //очистка памяти после завершения работы с массивом
9:  delete [] mas;
```

Вектор:

```
1:  #include <vector>
2:  #include <algorithm> //для использования встроенной сортировки и др.
3:  void main()
4:  {
5:     vector<int> vec;
6:     int val;
7:     cin<<val;
8:     vec.push_back(val);//добавить элемент
9:    sort(vec.begin(), vec.end());//сортировка элементов по возврастанию
10:   for(int i=0;i<vec.size();i++)
11:   {
12:     //vec[i] - доступ к элементам
13:   }
14: }
```

Строка:

```
1:  #include <string>
2:
```

```
3:    string s="Hello, World!";
4:           cout<<"Длина > " << s.length()<<endl;
5:                  //13 , также можно использовать метод size()
6:
7:    cout<<"Первый символ > "<<s[0]<<endl;//H
8:    cout<<"Пять символов начиная с 8-го >"<<s.substr(7,5)<<endl;
9:                                          //World
10:   s.replace(7,5,"ISiT"); // "Hello, ISiT!"
11:   s.insert(6," my"); // "Hello, my ISiT!"
12:   s += " How are you?"; // "Hello, my ISiT! How are you?"
13:   s.erase(14,13);       // "Hello, my ISiT?"
14:   cout<<"Слово 'ISiT' начинается с символа > "<<s.find("ISiT")<<endl;//   10
15:   string s1="abc";
16:   string s2="xyz";
17:
18:   //Сравнение строк: 1 - больше, 0 - равны, -1 - меньше
19:   cout<<<<s1.compare(s2)<<endl;  // -1
20:
21:       //Обмен данными между строками
22:   s1.swap(s2); // s1="xyz", s2="abc"
23:
24:      //преобразование string в char*
25:      //(без возможности дальнейшего изменения)
26:   const char *cstr = s.c_str();
27:
28:   //использование strcpy_s (безопасная функция копирования)
29:   int len = s.length() + 1;
30:   char *cstr2 = new char[len];
31:   strcpy_s(cstr2, len*sizeof(char), s.c_str());
32:
33:   //использование strcpy (обычная функция копирования)
34:   int len = s.length() + 1;
35:   char *cstr3 = new char[len];
36:   strcpy(cstr3, s.c_str());
37:
38:   //преобразование *char в string
39:   char *chs="La-la-la";
40:   string s4(chs); // во время создания переменной
41:   s =  (const char*) chs; // запись в существующую переменную
```

Чтение и запись данных

Перенаправление потока. Команды `printf`, `cout` вместо вывода на консоль будут производить запись в файл, команды `cin`, `scanf` – чтение из файла.

```
1:  int main(){
2:  int a;
3:
4:    freopen("input.txt","r",stdin);
5:    freopen("output.txt","w",stdout);
6:
7:    scanf(&a);
8:    printf(a);
9:
10: //в случае использования безопасных функций
11:
12:   //#include <fstream>
```

```
13:    std::FILE *file,*file2;
14:
15:    freopen_s(&file, "input.txt","r",stdin);
16:    freopen_s(&file2,"output.txt","w",stdout);
17:
18:    scanf_s(&a);
19:    printf(a);
20:  }
```

Использование потокового класса `fstream`:

```
1:  #include <fstream>
2:
3:  ////////////////////////////////////////////////////запись
4:    ofstream out("file.txt");
5:    out<<"Hello World!";
6:    out.close();
7:
8:
9:    out.open("file.txt",ios::app);
10:            //открыть файл в режиме добавление записи
11:   out<<1<<endl<<2;
12:   out.close();
13:
14:   /** содержимое file.txt
15:   * Hello World
16:   * 1
17:   * 2
18:   **/
19:
20: ////////////////////////////////////////////////////чтение
21:   ifstream in("file.txt");
22:   char ch;
23:
24:   //посимвольное чтение
25:   while((ch = in.get()) != EOF )
26:   {
27:          //cout<<ch;
28:   }
29:   in.close();
30:
31:   in.open("file.txt");
32:   string s;//  #include <string>
33:
34:   //построчное чтение
35:   while (in.good())
36:   {
37:     getline(in,s);
38:     //cout<<s<<endl;
39:   }
40:   in.close();
```

Вывод в файл данных из другого потока (например, из выводимого на консоль через `cout`):

```
1:  #include <fstream>
2:
```

```
   //использование безопасной функции перевода потока
   FILE *file;
   freopen_s(&file,"output.txt", "w", stdout);
   cout<<"Hello, World!";

   //использование небезопасной функции перевода потока
   freopen ("output.txt","w",stdout);
   cout<<"Hello, World!";
```

Замер времени

```
#include <time.h>

int main()
{
clock_t begin_s=0, end_s=0;
   begin_s=clock();
  // code
  end_s=clock();

std::cout << "Время работы: \t\t\t\t\t"
          << (double)(end_s-begin_s)/(double)CLOCKS_PER_SEC<<std::endl;
}
```

ЛИТЕРАТУРА

1. Гаврилов, Г. П. Задачи и упражнения по дискретной математике / Г. П. Гаврилов, А. А. Сапоженко. – М.: Физматлит, 2004. – 232 с.
2. Селезнева, С. Н. Основы дискретной математики / С. Н. Селезнева. – М.: МАКС Пресс, 2010. – 120 с.
3. Липский, В. Комбинаторика для программистов: учеб. пособие / В. Липский. – М.: Мир, 1998. – 167 с.
4. Задачи по программированию / С. М. Окулов [и др.]; под ред. С. М. Окулова. – М.: БИНОМ. Лаборатория знаний, 2006. – 820 с.
5. Златопольский, Д. М. Программирование: типовые задачи, алгоритмы, методы / Д. М. Златопольский. – М.: БИНОМ. Лаборатория знаний, 2007. – 223 с.
6. Кирюхин, В. М. Методика решения задач по информатике. Международные олимпиады / В. М. Кирюхин, С. М. Окулов. – М.: БИНОМ. Лаборатория знаний, 2007. – 600 с.
7. Скиена, С. С. Олимпиадные задачи по программированию. Руководство по подготовке к соревнованиям / пер. с англ. С. С. Скиена, М. А. Ревилла. – М.: КУДИЦ-ОБРАЗ, 2005. – 416 с.
8. Меньшиков, Ф. В. Олимпиадные задачи по программированию / Ф. В. Меньшиков. – СПб.: Питер, 2006. – 315 с.
9. Интернет-портал [Электронный ресурс] / Российская коллекция задач международных и всероссийских олимпиад по информатике с методическими рекомендациями по их решению. – Режим доступа: http://info.rusolymp.ru. – Дата доступа: 20.05.2013.
10. Интернет-портал [Электронный ресурс] / Олимпиадные задачи по программированию. – Режим доступа: http://algolist.manual.ru/olimp. – Дата доступа: 22.05.2013.
11. Интернет-портал [Электронный ресурс] / Московские олимпиады по информатике. – Режим доступа: http://www.olympiads.ru/moscow. – Дата доступа: 24.05.2013.
12. Интернет-портал [Электронный ресурс] / Задачи Всероссийских командных олимпиад школьников по программированию. – Режим доступа: http://neerc.ifmo.ru/school/russia-team/archive.html. – Дата доступа: 24.05.2013.
13. Интернет-портал [Электронный ресурс] / Открытая Всесибирская олимпиада по программированию им. И. В. Поттосина. – Режим доступа: http://www.olympiads.nnov.ru. – Дата доступа: 24.05.2013.
14. Интернет-портал [Электронный ресурс] / Олимпиадная информатика. – Режим доступа: http://www.olympiads.ru. – Дата доступа: 24.05.2013.

15. Интернет-портал [Электронный ресурс] / Портал обучения информатике и программированию. – Режим доступа: http://school.sgu.ru. – Дата доступа: 24.05.2013.
16. Интернет-портал [Электронный ресурс] / Timus Online Judge. – Режим доступа: http://acm.timus.ru. – Дата доступа: 24.05.2013.
17. Интернет-портал [Электронный ресурс] / Школа программиста. – Режим доступа: http://acmp.ru. – Дата доступа: 24.05.2013.
18. Интернет-портал [Электронный ресурс] / Дистанционная подготовка по информатике. – Режим доступа: http://informatics.mccme.ru/moodle. – Дата доступа: 24.05.2013.
19. Интернет-портал [Электронный ресурс] / ACM-ICPC Live Archive. – Режим доступа: https://icpcarchive.ecs.baylor.edu. – Дата доступа: 24.05.2013.
20. Интернет-портал [Электронный ресурс] / Задачи. – Режим доступа: http://www.problems.ru. – Дата доступа: 24.05.2013.

СОДЕРЖАНИЕ

Учебное издание

Пацей Наталья Владимировна
Пласковицкий Владимир Александрович

ПРОГРАММИРОВАНИЕ ПОВЫШЕННОЙ СЛОЖНОСТИ

Сборник задач

Редактор *А. Д. Микитюк*
Компьютерная верстка *А. Д. Микитюк*
Корректор *А. Д. Микитюк*

Издатель:
УО «Белорусский государственный технологический университет».
Свидетельство о государственной регистрации издателя,
изготовителя, распространителя печатных изданий
№ 1/227 от 20.03.2014.
Ул. Свердлова, 13а, 220006. г. Минск