Идеи за решаване на задачите

Задача 1. Домашно

Тази задача е предложена от Добрин Пасков,
студент във ФМИ на СУ „Св. Климент Охридски"

Всеки ред от триъгълника съдържа една начална звездичка, която трябва да се отпечата в поле с определена широчина и следващи по-десни звездички, които трябва да се отпечатат предхождани от интервал. За първия ред на триъгълника широчината на полето за отпечатване на началната звездичка е *n* символа и намалява всеки следващ ред с 1 символ. Броят по-десни звездички за отпечатване е 0 за първия ред и се увеличава с 1 всеки следващ ред. Реализираме един цикъл, в който печатим за всеки ред началната звездичка и след това по-десните звездички, които печатим чрез вложен цикъл.

Това е! Ако Ви е било прекалено елементарно – бързо подхващайте следващата задача.

Кодът на програмата, написан на езика C++

```
#include <iostream>
#include <iomanip>
using namespace std;

int main() {
  int n,    // брой редове със звездички
      i,    // брой следващи звездички в реда (след първата)
      j;    // отпечатани следващи звездички

  cin >> n;
  for (i=0; n; n--, i++) {
    cout << setw(n) << '*';           // отпечатва първата * в реда
    for (j=0; j<i; j++) cout << " *"; // отпечатва следващите * в реда
    cout << endl;
  }
  return 0;
}
```

Задача 2. Разширяващо се число

Тази задача е предложена от Евгений Василев,
катедра Информатика, НПМГ „Акад. Л. Чакалов" – София.
(варианти на задачата са давани и на други състезания)

Ако $N \le 17$ е лесно – взимаме необходимата цифра и я отпечатваме. В тази връзка изглежда по-удобно да прочетем *M* като низ за да имаме директен достъп до отделните му цифри. Ако прочетем *M* като число ще трябва да отделяме цифрите му с допълнителни изчисления (вижте задача 4 от Първото състезание на Урановата дивизия). При $N > 17$ ще трябва да изчисляваме дописваните цифри. Дали наистина е така?

Да означим сумата от цифрите на *M* със S_{17} Тогава средно аритметичното от цифрите на *M* е:

$$\frac{S_{17}}{17} = d + \frac{\delta}{17} \qquad (2.1)$$

Където *d* и *δ* са съответно частното и остатъка при целочисленото делене на 17. Разглеждаме 2 варианта:

Вариант 1: $\delta < \frac{17}{2}$

По близкото цяло до $\frac{S_{17}}{17}$ е *d* и това е дописваната цифра. След добавянето на 18-тата цифра в *M*, да определим каква е 19-та цифра за добавяне:

$$\frac{S_{18}}{18} = \frac{S_{17} + d}{18} = \frac{17d + \delta + d}{18} = d + \frac{\delta}{18}$$

Постъпвайки аналогично за всяка следваща цифра установяваме:

$$\frac{S_i}{i} = \frac{S_{i-1} + d}{i} = d + \frac{\delta}{i} \quad , i > 17 \qquad (2.2)$$

Понеже $\frac{17}{2} > \frac{\delta}{17} > \frac{\delta}{18} > \cdots > \frac{\delta}{i} > \cdots$ следва, че в (2.2) *d* е по-близкото цяло и винаги (!!!) трябва да се дописва към *M*.

Вариант 2: $\delta > \frac{17}{2}$

По близкото цяло до $\frac{S_{17}}{17}$ е *d*+1 и това е дописваната цифра. Преобразуваме (2.1) в по удобна форма:

$$\frac{S_{17}}{17} = d + \frac{\delta}{17} = d + \frac{\delta}{17} + 1 - 1 = (d+1) - \frac{17 - \delta}{17} = (d+1) - \frac{\delta'}{17} \qquad (2.1')$$

Аналогично както във Вариант 1 получаваме:

$$\frac{S_i}{i} = \frac{S_{i-1} + (d+1)}{i} = (d+1) - \frac{\delta'}{i} \quad , i > 17 \qquad (2.2')$$

Важно е да се отбележи, че в (2.1') $\delta' < \frac{17}{2}$ от там и $\frac{17}{2} > \frac{\delta'}{17} > \frac{\delta'}{18} > \cdots > \frac{\delta'}{i} > \cdots$

Следователно и тук цифрата *d*+1 се дописва винаги към *M*.

Да обобщим: Всички цифри в *M* след 17-тата са еднакви!

Кодът на програмата, написан на езика C++

```
#include <iostream>
#define DIGITS 17
using namespace std;

int main() {
  char A[DIGITS+1];
  int n, sum=0;
  cin >> A >> n;
```

```
  if (n>DIGITS) {
    for (int i=0; A[i]; sum+=A[i++]);
    cout << (char)(sum/DIGITS + (sum%DIGITS > DIGITS/2)) << endl;
    }
  else cout << A[n-1] << endl;
  return 0;
}
```

Задача 3. Ламята Фаска

Тази задача е предложена от Даниел Балчев,
студент във ФМИ на СУ „Св. Климент Охридски"

Решаваме задачата чрез моделиране на процеса на порастване на глави на Фаска. Да разгледаме ситуацията в *i*-я ден от годината: днешния брой глави *today* утре ще нарасне с броя негладуващи глави от вчера (*yesterday* –(*i*-1)) и ще стане *tomorrow*. Утре повтаряме изчислението, в другиден – също и така, докато не достигнем до определяне на главите на Фаска за търсения ден. Всичко това правим в цикъл, който започва от 2-я ден на годината. Можем лесно да определим стартовите условия на цикъла: главите в първия ден са 3, а във втория – 5. Може да записваме резултатите от моделирането в масиви (по един за броя глави и броя гладуващи глави всеки ден), но може и да изхитруваме: това, което за днешния ден е *tomorrow* и *today*, утре ще е съответно *today* и *yesterday*. Така може да решим задачата с помощта само на 3 основни променливи.

В допълнение трябва да съобразим, че броят глави на Фаска нараства много бързо и трябва да подберем възможно най-широкия целочислен тип за съответните им променливи – 64 битово безнаково цяло.

Кодът на програмата, написан на езика C++

```
#include <iostream>
using namespace std;

int main (){

  unsigned long long
    yesterday = 3,          // глави вчера
    today = 5,              // глави днес
    tomorrow,               // глави утре
    y_day_no = 1;           // номер на вчерашния ден
  int day;                  // ден, за който търсим

  cin >> day;
  if (day == 1) today = 3;
  else
    for (; y_day_no + 1 < day; ++y_day_no){
      tomorrow = today + yesterday - y_day_no;
      yesterday = today;
      today = tomorrow;
    }

  cout << today << endl;
  return 0;
}
```

Задача 4. Лампи

Тази задача е предложена от Зорница Дженкова,
фирма „Шест плюс" ЕООД, Габрово.
(задачата е давана и на други състезания)

Очевидно е, че след края на представлението една лампа няма да свети ако нейният ключ е натискан от Петьо четен брой пъти и ще свети, ако е натискан нечетен брой пъти. Или с други думи трябва да се определи броят делители на числото – номер на лампата. Ако едно число А е кратно на В, може да запишем

$A : B = C$, където C е частното (4.1)

Но тогава A е кратно и на C ! На пръв поглед се създава впечатлението, че числата имат четен брой делители. След известно взиране в (4.1) ни хрумва, че е възможно $B = C$, т.е. $A=B^2$ (A е точен квадрат) и в този случай делителите на числото са нечетен брой. От тук задачата се свежда до намирането на най-голямото число , което е точен квадрат и такова, че да не е по-голямо от n. Авторът на задачата е предложил един вариант как да намерим това число: започвайки от 1 повдигаме поредното число i на квадрат и ако $i^2<=n$ увеличаваме i с 1 и повтаряме проверката. В момента, когато $i^2>n$ връщаме се към i-1 и това е решението.

Съществува и по-директен начин за намиране на отговора, базиран на обратната на степенуването математическа операция – коренуване. K-ти корен от A (бележи се $\sqrt[K]{A}$) е такова число B, за което $B^K = A$. Езиците за програмиране предлагат средства (най-често вградени функции) за определяне на 2-ри корен, какъвто е нашия случай. (За езика C++ това е функцията sqrt(A)). Понеже резултатът от коренуването в повечето случаи е число с дробна част, взимаме цялата му част, повдигаме я на квадрат и това е търсеният отговор!

Кодът на програмата, написан на езика C++

```
// Първи вариант
#include<iostream>
using namespace std;
int main(){
  long long int n;
  cin >> n;
  long long int p=1;
  while(p*p<=n) p++;
  p--;
  cout << p*p << endl;
  return 0;
}
/*// Втори вариант
#include<iostream>
#include <cmath>
using namespace std;
int main(){
  long long int n, p;
  cin >> n;
  p = sqrt(n);
  cout << p*p << endl;
  return 0;
}
*/
```