Идеи за решаване на задачите

Задача 1. Парола

Тази задача е предложена от Евгений Василев,
катедра Информатика, НПМГ „Акад. Л. Чакалов" – София.

Прочитаме паролата в масив с достатъчен брой елементи за символите ѝ, като не забравяме да декларираме масива с един елемент повече от максималната дължина на паролата заради маркера за край на текст ('\0'). Разглеждаме последователно символите на паролата:

```
for (i = 0; password[i] > '\0'; i++) {
  if (i-я символ на паролата е буквен) {
    // обработваме буквен символ
    }
  else if (i-я символ на паролата е цифров) {
    // обработваме цифров символ
    }
  else { // или i-я символ на паролата е недопустим
    // отпечатваме код на грешка 1 и приключваме програмата
  }
}
```

Обработката на буквен символ се състои в следното:

- Ако предишният символ също е бил буквен, добавяме още един към поредицата буквени (увеличаваме броя им, записан в променливата count_prev_equ). Ако предишният не е буквен, текущият символ е начало на нова поредица от буквени, т.е. поредицата в момента вече има един буквен символ (count_prev_equ=1). Допълнително трябва да запомним, че текущият символ е буквен (променливата prev_is_char), така при следващата итерация на цикъла (следващия символ от паролата) ще се знае какъв е бил вида на предходния символ (текущият символ ще е предходен за следващия го)
- В края на обработката на буквен символ проверяваме дали поредицата буквени не е станала прекалено дълга (count_prev_equ > 3). Ако е така – отпечатваме кода на грешката и приключваме програмата още преди да сме анализирали всички символи от паролата.

Обработката на цифров символ се извършва по подобен начин, може да видите в приложения код по-долу. След приключване на цикъла остава да проверим само дали паролата не е по-къса от допустимото (i < 8), ако е така – печата се код на грешката, в противен случай се печата 0 (нула). И...това е!

Кодът на програмата, написан на езика C++

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

char password[51];

int main() {
  int i, prev_is_char = 0, count_prev_equ=0;
  cin >> password;
```

```
  for (i = 0; password[i] > '\0'; i++) {

    // Обработка на буквен символ
    if ((password[i] >= 'A' and password[i] <= 'Z') or
        (password[i] >= 'a' and password[i] <= 'z')) {
      if (prev_is_char >0) count_prev_equ++;
      else {
        prev_is_char = 1;
        count_prev_equ = 1;
      }
      if (count_prev_equ > 3) {
        cout << 2 << endl;
        return 0;
      }
      }

    // Обработка на цифров символ
    else if (password[i] >= '0' and password[i] <= '9') {
      if (prev_is_char == 0) count_prev_equ++;
      else {
        prev_is_char = 0;
        count_prev_equ = 1;
      }
      if (count_prev_equ > 3) {
        cout << 2 << endl;
        return 0;
      }
      }

    // Обработка на недопустим символ
    else {
      cout << 1 << endl;
      return 0;
    }

  }

  if (i < 8) cout << 3 << endl;
  else cout << 0 << endl;

  return 0;
}
```

Задача 2. Час

Тази задача е предложена от Явор Никифоров,
Школа по информатика, СМГ „Паисий Хилендарски" – София.

Я да видим сега справяте ли се със задачи в които няма нещо за измисляне, обаче трябва да сте внимателни. Примерно да се усетите, че първият символ (а и изобщо винаги, когато очаквате число, ама може и да е някъФ символ) трябва да го въвеждате като char и в зависимост от този първи въведен char да разклоните програмата си на два случая (от единия към другия формат или обратното).

Ами ако char-а се окаже цифра, да не сте си забравили как cifravchar-'0' ни дава истинската ѝ цифрова стойност (а не ASCII номер).

А внимателни ли бяхте да не пропускате нулите, които трябваше да се пишат в единия формат? Аз примерно съм го правил с много if-ове. Да знаете, че със cin има и начин да

си слагате излишни нули в печатането автоматично, ама за който не го знае (като мен ☺) ще буха if-ове.

Та така ... по състезания има и хамалски задачи и, срам – не срам, то и моето решение стана 46 реда. Много е важно в такива случаи да си подреждате кода и да сте си кръстили променливите по подходящ начин (ех, че се изтърка тоя мой съвет)... пък да не забравите и новия ред в края на изхода...

Кодът на програмата, написан на езика C++

```
#include<iostream>
using namespace std;
int main(){
char ch;
long long h,m,chasovezadobavqne;
chasovezadobavqne = 0;
cin>>ch;
if(ch=='a' || ch=='p'){
  if(ch=='p'){
    chasovezadobavqne=12;
  }
  cin>>ch>>ch>>ch;//preskacham .m.
  cin>>h;
  h=h+chasovezadobavqne;
  cin>>ch;
  cin>>m;
  if(h<10){
    cout<<0;
  }
  cout<<h;
  cout<<"-";
  if(m<10){
    cout<<0;
  }
  cout<<m;
  cout<<"\n";
  }
else{
  //znachi ch e cifra! moje i nula da e, ama kvo ot tva
  h=10*(ch-'0');
  cin>>ch;
  h=h+ch-'0';
  cin>>ch;//preskachame znaka -
  cin>>ch;
  m=10*(ch-'0');
  cin>>ch;
  m=m+ch-'0';
  //probvaite mnogo drugi po- kulturni, prosti i qsni nachini da si vuvedete
  //dve cifri ot koito ednata moje i da e 0 i da gi oburnete na chislo.
  if(h<12){
    cout<<"a.m. ";
    }
  else{
    cout<<"p.m. ";
  }
  cout<<h%12<<":"<<m<<"\n";
}
return 0;
}
```

Задача 3. Манджа

Тази задача е предложена от Явор Никифоров,
Школа по информатика, СМГ „Паисий Хилендарски" – София.

Явна е нуждата ни да си броим коя буква колко се среща. Има само 26 букви, 'начи ни трябват само 26 променливи... Или може би има мъъничко по-културен вариант, а?

Културен вариант би било да си имаме масив който ползваме да broisreshtanianabukva, в който бихме искали индексите да са самите букви (а стойностите – броя срещания на съответната буква). Ъъъм... масив дето индексът му е буква?! Ем', понеже на масивите индексът им е непременно число, 'що да не си направим просто масив, в който индексът е кой номер е буквата в азбуката! Накрая трябва само да го обходим, че да издирим търсения номер на буква.

Кодът на програмата, написан на езика C++

```
#include<iostream>
using namespace std;
int main(){
  char mandja[100],maxchar;
  cin>>mandja;
  long long broi[26],i,maxbroi;
  for(i=0;i<26;i=i+1){
    broi[i]=0;
  }

  for(i=0;mandja[i]!='\0';i=i+1){
    broi[mandja[i]-'A']=broi[mandja[i]-'A']+1;
  }
  maxbroi=0;
  for(i=0;i<26;i=i+1){
    if(maxbroi<broi[i]){
      maxbroi=broi[i];
      maxchar='A'+i;
    }
  }
  cout<<maxbroi<<" "<<maxchar<<"\n";
  return 0;
}
```

Задача 4. Най-къса дума

Тази задача е предложена от Сюзан Феимова,
МГ „Баба Тонка" – Русе.

Използваме, че cin стандартно чете само отделни думи от дълъг текст – четем дума по дума текста, докато не достигнем дума с последен символ **.** (точка). За всяка прочетена дума проверяваме дали първият ѝ и последният ѝ символи са еднакви. Ако е така и ако е по-къса от най-късата такава дума, открита до момента, записваме поредната дума като рекордно къса. Не трябва да забравяме, че е възможно текущата дума да е последна от текста – съдържа точка като последен символ. В такъв случай премахваме точката – „върху нея" записваме маркера за край на текст '\0' и помним, че думата е станала с един символ по-къса. В този случай запомняме, че трябва да излезем от цикъла за четене на думи от текста. Това се прави като в дадена управляваща

променлива се постави избрана стойност, съответстваща на ситуацията „прекъсни цикъла" (в кода по-долу това е променливата must_break, която прекъсва цикъла, чрез избраното от автора значение true). Накрая се отпечатва запомнената рекордно къса дума с еднакви първи и последен символи.

Отбележете любопитната, но и много важна за практиката (!), техника за задаване на условие на цикъла while в кода по-долу: като условие на цикъла е поставена операцията за четене на данни! Успешното прочитане на данни е еквивалентно на стойност true, неуспешното – на стойност false. Като знаете това, вече ще можете да си пишете програми, които четат неизвестно количество данни – просто четете в един безкраен цикъл докато четенето е успешно, като свършат данните (за тази ситуация се използва термина „при край на входа") цикълът приключва. Нещо такова:

```
while (cin>>danna) {
  //правим нещо с поредно прочетената данна
}
// тук продължаваме с останалата част от програмата,
// когато няма вече данни за четене от входа
```

И последно: как да кажем на така написана програма по време на работа, че данните вече са свършили? Нали cin чака някой да въведе данни от клавиатурата, ако още не са написани в момента, т.е. предполага, че все някога ще се появят данни и значи те все още не са свършили? За да кажете на cin, че данните вече са свършили, трябва при работата на програмата Ви да въведете от клавиатурата клавишната комбинация Ctrl+Z след което да натиснете Enter Това е валидно за програма, която се изпълнява под управлението на операционната система Windows©™ Как е при други операционни системи – питайте по-опитните си приятели.

Кодът на програмата, написан на езика C++

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

int main() {
  char word[22], shortword[21];          // поредната и най-късата думи
  int lenght, lenght_short_word = 1000; // дължините им
  bool must_break = false;               // да свършим ли с четенето на думи

  while (cin >> word) {
    lenght = strlen(word);
    if (word[lenght-1]=='.') {
      must_break = true;
      word[lenght-1] = '\0';
      lenght--;
    }
    if (lenght < lenght_short_word and word[0]==word[lenght-1]) {
      lenght_short_word = lenght;
      strcpy(shortword, word);
    }
    if (must_break) break;
  }

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