Зад.1 Приятелски числа *Да се състави програма, която по зададен интервал [**K,L**] (**K** и **L** цели положителни не по-големи от 10000) отпечатва най-голямата двойка приятелски числа в този интервал. **K** и **L** се въвеждат от клавиатурата. Две числа са приятелски ако са прости и разликата им по модул е равна на 2. Ако няма приятелски числа в интервала програмата отпечатва 0.*

Примерен вход **Примерен изход**

10 50 43 41

Решение

```
#include<iostream>
using namespace std;
int simple(int x)
{
int d;
    if(x==0||x==1)return 0;
     for(d=2; d<=x/2;d++)
     if(x%d==0)return 0;
    return 1;
}
int main()
{
int m,n;
bool b=false;
   cin>>m>>n;
   for(int i=n;i>=m;i--)
    {
        int j=i-2;
        if(simple(i)&&simple(j))
        { b=true; cout<<i<<","<<j<<endl; break; }
    }
    if(b==false) cout<<"Niama takiva chisla!"<<endl;
   return 0;
  }
```

Трябва да съобразим, че най-бързия начин да намерим най-голямата двойка приятелски числа в интервала [**K,L**] е да започнем обхождането на интервала отзад напред, като на всяка стъпка проверяваме дали числото, което е с 2 по-малко от текущото и текущото са прости. Ако това е така, означава, че сме намерили двойка приятелски числа и тя е най-голямата, което означава, че трябва да преустановим търсенето. Това ще се осъществи с оператор **break**. Ако търсенето е приключило и не е намерена такава двойка числа програмата трябва да отпечата 0. За тази цел ще декларираме променлива **b** която ще има стойност **true** ако сме намерили приятелски числа и **false** в противен случай.

Зад.2 Сума от цифри *Да се състави програма, която въвежда от клавиатурата две цели числа **L** и **K** и извежда всички числа в интервала [**L,K**](0<**L**<**K**<=1000000), които имат сума от цифрите, кратна на 11.*

Примерен вход **Примерен изход**

545 597 551
560
589

Решение

```
#include  <iostream>
using  namespace  std;
int Suma(int a)
{
int sum=0;
  while(a)
   {
    sum=sum+a%10 ;
    a=a/10 ;
   }
return sum;
}
int main()
{
int K,L,i;
  cout<<"Vavedi nachalo na intervala=";
   cin>>K;
  cout<<"Vavedi kray na intervala=";
   cin>>L;
  for (i=K; i<=L; i++)
    if((Suma(i)%11)==0)
    cout<<"Chislata sa:"<<i<<endl;
return 0;
}
```

Зад.3 Най-голям общ делител *Да се състави програма, която въвежда от клавиатурата три цели числа* ***m, n*** *и* ***к*** *и извежда всички двойки числа в интервала* ***[m,n]*** *(0<****m****<****n****<=1000), които имат най-голям общ делител по-голям или равен на* ***к****.*

Примерен вход

11 27 8

Примерен изход

11 22
12 24
13 26
16 24
18 27

Решение

```
#include  <iostream>
using  namespace  std;
int NOD (int a, int b)
{
   while(a!=b)
    if (a<b) b-=a;
    else  a-=b;
return a;
}
int main()
{
int m,n,k;
cout<<"Vavedi nachalo na intervala=";
  cin>>m;
cout<<"Vavedi kray na intervala=";
  cin>>n;
cout<<"Vavedi delitel=";
  cin>>k;
    for (int i=m ; i<n; i++)
    for (int j=i+1; j<=n; j++)
        if(NOD(i,j)>=k) cout<<i<<" "<<j<<endl;
return 0;
}
```

За решението на задачата ще използваме алгоритъма на Евклид за намиране на най-голям делител на две числа a**и** b**:**
1. Докато числата не са равни повтаряме следното:
-ако a<b **изваждаме** a **от** b **и резултата присвояваме на** b**;**
-в противен случай изваждаме b **от** a **и резултата присвояваме на** a**.**

Когато двете числа се изравнят, е намерен най-големият им общ делител и той е равен на получената една и съща стойност за двете числа.

На базата на така описания алгоритъм можем да съставим функция, която да приема като параметри числата, чийто НОД търсим и връща като резултат намерения НОД. Както параметрите, така и типа на функцията са цели числа.

За да намерим всички двойки числа в интервала [m, n**], които имат най-голям общ делител по-голям или равен на** к**, е необходимо да разгледаме всички възможни двойки числа в този интервал, за всяка да проверим дали отговаря на условието и ако е така – да я отпечатаме. Обхождането на всички двойки ще реализираме чрез два вложени цикъла** for**, в единия от които управляващата променлива** i **приема последователно като стойности всички числа в интервала, а другия обхожда всички числа от** i+1 **до края на интервала** n**. За всяка получена двойка се проверява дали резултатът от функцията NOD е по-голям или равен на** к**. Ако е така, съответната двойка се отпечатва на екрана.**

Източник: Школа по програмиране Telerik Kids Academy