**Числови алгоритми****. Прости числа**

 Простите числа се разглеждат в математиката още от дълбока древност. Те са свързани с много интересни задачи и ползата от тях се простира далеч извън пределите на математиката. В информатиката те намират приложение в криптирането, архивирането, и много др.

Едно цяло число *p* (*p*>1) се нарича просто, ако няма други делители освен 1 и себе си. Ако *p* не е просто, то се нарича съставно.Редицата от простите числа започва така:

 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, …

Доказано е (*Евклид –* 300 г.пр.н.е.), че простите числа са безброй много.

Когато разглеждаме редицата на простите числа, възникват следните въпроси:

* Колко прости числа има в даден интервал [*a*, *b*]?
* Каква част от безкрайността представляват простите числа?
* Съществува ли формула за намиране на *n***-**тото поред просто число?

Преди да преминем към алгоритмите за проверка дали едно число е просто, ще споменем още няколко интересни свойства и теореми за простите числа:

***Хипотези на Голдбах***

1. Всяко цяло четно число *n*>2 може да се представи като сума на 2 прости числа.
2. Всяко цяло число *n*>17 може да се представи като сума на 3 различни прости числа.
3. Всяко цяло число може да се представи като сума на най-много 6 прости числа.
4. Всяко цяло нечетно число *n*>5 може да се представи като сума на 3прости числа.
5. Всяко четно число може да се представи като разлика на две прости числа.

***Теорема***. Съществуват безброй много прости числа от вида *n*2*+m*2 и *n*2*+m*2*+*1.

***Хипотеза*.** Съществуват безброй много прости числа от вида *n*2*+*1.

***Теорема*.** (*Оперман*) Винаги съществува просто число между *n*2и(*n+*1)2.

**Проверка дали дадено число е просто:**

 Един очевиден алгоритъм, пряко следствие от дефиницията, е следният: Проверяваме всяко число от интервала 2 до –1 дали дели *p* и ако намерим някое, което го дели, следва, че *p* е съставно.

Съществуват някои “формули” за проверка дали число е просто, но на практика те се оказват неприложими, тъй като реализацията им изисква много повече компютърни изчисления, отколкото току-що описаният алгоритъм. Един пример е следната

***Теорема на Уилсън.*** Числото *p*е просто тогава и само тогава, когато(*p*–1)! *=* –1(*mod p*)

При нея трябва да се изчисли (*p*–1)!, което е доста по-трудно като реализация и предполага многократно повече изчисления от извършването на –1 деления в горния алгоритъм. Ще се опитаме да го подобрим още.

Лесно може да се съобрази, че е излишно да проверяваме всички числа до –1, а е достатъчно да проверим за делимост само до . Това е така, тъй като винаги, когато *p* има делител *x*, *x*>, то следва, че *p* се представя във вида *p*=*x*.*y*, *y*<, т.е. има и делител по-малък от . Ето една реализация на този алгоритъм:

***Алгоритъм:***

Ако n = 2, то връща 1,

 Иначе i = 2

 Докато I <= корен квадратен от n прави

 Ако n mod I = 0, то връща 0

 I = I+1

 връща 1

#include <iostream>

#include <math.h>

using namespace std;

/\* Функцията връща 1 ако n е просто и 0 ако n е съставно \*/

int isPrime(int n) {

 if (n==2) return 1;

 int i = 2;

 while (i <= sqrt(n)) {

 if (n % i == 0) return 0;

 i++;

 }

 return 1;

}

int main() {

 int n;

 cout<<"N=";

 cin >>n;

 if (isPrime(n)==1) cout <<"Number is Prime"<<endl;

 else cout <<"Number is not Prime"<<endl;

}

Можем да разширим още малко последния резултат: за да установим, че *p* е просто, е достатъчно да сме сигурни, че не се дели на нито едно *друго просто число* от интервала [2, ] (Оставяме като упражнение за читателя да покаже, че последното заключение е коректно). Така, ако разполагаме с първите *k* прости числа, ще можем да проверяваме дали произволно число от интервала [2, *k*2] е просто. Ето програма извършваща последното:

#include <iostream>

using namespace std;

const int K=25; /\* брой прости числа, които сме изчислили предварително\*/

int prime[K] = {2,3,5,7,11,13,17,19,23,29,31,37,41,

 43,47,53,59,61,67,71,73,79,83,89,97 };

/\* проверяваме дали число е просто, като проверяваме дали има делители сред числата от масива prime[] \*/

int checkprime(int n) {

 int i = 0;

 while (i < K && prime[i]\*prime[i] <= n) {

 if (n % prime[i] == 0) return 0;

 i++;

 }

 return 1;

}

int main() {

 int n;

 cout << "N=";

 cin >>n;

 if (checkprime(n)) cout<<"Chisloto e prosto."<<endl;

 else cout<<"Chisloto e sustavno. "<<endl;

}