МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н.Г. ЧЕРНЫШЕВСКОГО»

Кафедра теоретических основ компьютерной безопасности и криптографии

Проверка чисел на простоту с помощью полиномиального теста распознавания простоты

ЛАБОРАТОРНАЯ РАБОТА

студента 4 курса 431 группы
специальности 10.05.01 Компьютерная безопасность
факультета компьютерных наук и информационных технологий
Серебрякова Алексея Владимировича

Научный руководитель
доцент, к. п. н. _____ А. С. Гераськин
подпись, дата

Саратов 2022

Описание алгоритма

Алгоритм.
На входе задано нечетное число $n \in \mathbb{N}$, $n > 1$.

1 шаг. Если n имеет вид a^b, где $a \in \mathbb{N}$, $b \in \mathbb{N}$, $b \geqslant 2$, то выдать сообщение о том, что n составное, и закончить работу. (В работе [64] показано, что этот шаг может быть выполнен за $O(\log n^{1+o(1)})$ арифметических операций.)

2 шаг. $r := 2$.

3 шаг. Для текущего значения r выполнить шаги 4—8.

4 шаг. Если $r < n$ и НОД$(r, n) > 1$, то n — составное; в этом случае закончить работу с выдачей сообщения о том, что n — составное.

5 шаг. Если r — простое число, то выполнить шаги 6—7, иначе перейти на шаг 8.

6 шаг. Найти q — наибольший простой делитель числа $r - 1$.

7 шаг. Если $q \geqslant 4\sqrt{r}\log_2 n$ и $n^{\frac{r-1}{q}} \not\equiv 1 \pmod{r}$, то перейти к 9 шагу с данным значением r.

8 шаг. $r := r + 1$. Если $r \geqslant n$, то выдать сообщение, что n — простое, и закончить работу. Иначе вернуться на 3 шаг.

9 шаг. 1 случай. Если $n - 1 \leqslant [2\sqrt{r}\log_2 n]$, то для всех a из промежутка $r < a \leqslant n - 1$ проверить выполнение условия $(a, n) = 1$.

2 случай. Если $n - 1 > [2\sqrt{r}\log_2 n]$, то для всех a из промежутка $1 \leqslant a \leqslant [2\sqrt{r}\log_2 n]$ проверить выполнение соотношения

$$(x - a)^n \equiv x^n - a \pmod{x^r - 1}$$

в кольце $\mathbb{Z}/n\mathbb{Z}[x]$. Если для некоторого a в 1-м случае выполнено неравенство $(a, n) > 1$, либо во 2-м случае соотношение по модулю $x^r - 1$ не выполняется, то n — составное, и алгоритм заканчивает работу.

10 шаг. Если мы дошли до этого шага, то число n — простое.
Конец алгоритма.

Код программы

```
#include <bits/stdc++.h>
#include <iostream>

using namespace std;

int validated_input()
{
   int s = 0;
   while (!(cin >> s))
   {
      cin.clear();
      cin.ignore(numeric_limits<streamsize>::max(), '\n');
      printf("! Неверный ввод. Повторите ввод, начиная с первого неверного элемента.\n");
   }
```

```
    return s;
}

int gcd(int a, int b)
{
    return b ? gcd(b, a % b) : a;
}

float log_2(int n)
{
    return (log(n) / log(2));
}

int modexp(int x, int y, int N)
{
    int result(1);

    for (int i = 0; i < y; i++)
    {
        result *= x;
        result %= N;
    }

    return result;
}

bool wilson_crit(int n)
{
    int f(1), i(1);

    while (i < n)
    {
        f *= i;
        f %= n;
        i++;
    }

    return ((f + 1) % n == 0);
}

bool ch(int a)
{
    for (int i = 0; i < 15; i++)
    {
        float n = 0;
        n = sqrt(a);

        if (n == int(n))
            return true;

    }

    return false;
}

bool AKS(int n)
{
    int r(2);
```

```
if (n == 2 || n == 3 || n == 5 || n == 7){
   //printf("\n[1] Число %d простое. Выход\n", n);
   return true;
}

if (n % 2 == 0){
   //printf("\n[1] Число %d четное - оно не простое. Выход\n", n);
   return false;
}

for (int i = 2; i < 10; i++)
{

   int s(0), t(n);

   while (t != 0 && t % i == 0)
   {
      s++;
      t /= i;
   }

   if (t == 1){
      //printf("\n[1] Число %d вида a^b - оно составное. Выход\n", n);
      return false;
   }
}

//printf("\n[1] Число %d может быть простым. Продолжаем работу\n", n);
//printf("\n+ [2] Положим значение r = %d", r);

while (r < n)
{
   //printf("\n\n+ + [3] Выполним шаги 4-8, текущее значение r = %d\n", r);

   if (r < n && gcd(r, n) > 1){
      //printf("\n+ + + [4] НОД(%d, %d) >  1. Число %d составное. Выход", r, n, n);
      return false;
   } else //printf("\n+ + + [4] НОД(%d, %d) < 1. Продолжаем работу", r, n);

   if (wilson_crit(r))
   {
      //printf("\n+ + + + [5] r = %d - простое. Выполним шаги 6-7", r);

      int q = 1;
      for (int i = r - 1; i >= 1; i--)
      {
         if (r - 1 % i == 0 && wilson_crit(i))
         {
            q = i;
            break;
         }
      }

      //printf("\n+ + + + + [6] Наибольший простой делитель q = %d", q);

      if (q >= 4 * sqrt(r) * log_2(n) && modexp(n, r, q) != 0)
      {
         //printf("\n+ + + + + + [7] Условие выполнено. Переходим к шагу 9");
```

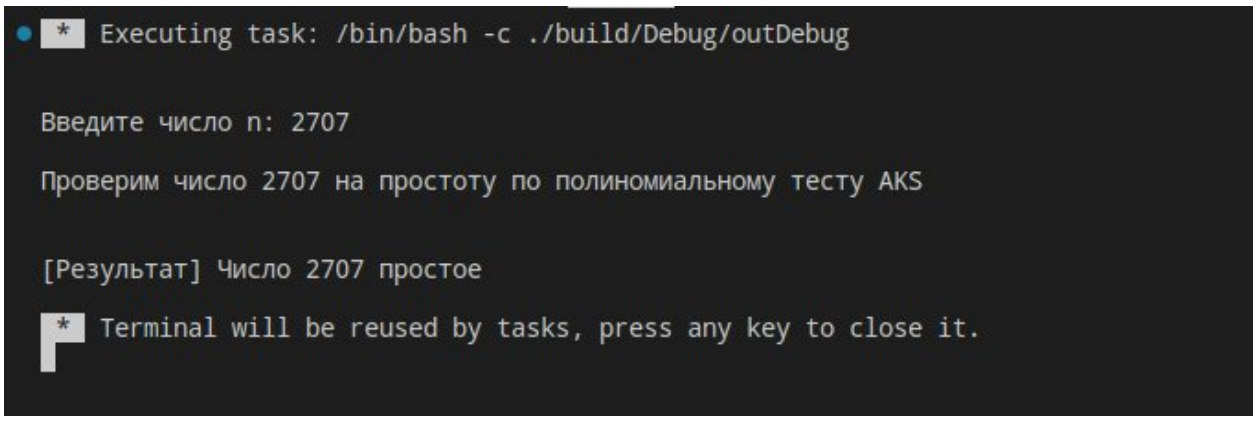
```
            if (n - 1 <= 2 * sqrt(r) * log_2(n))
            {
               //printf("\n+ + + + + + + + [9] Рассматривается 1й случай. Продолжаем работу");
               for (int a = r + 1; a < n; a++)
               {
                  if (gcd(a, n) > 1)
                  {
                     //printf("\n+ + + + + + + + + [10] Условие не выполнено. Число %d составное. Выход", n);
                     return false;
                  }
               }
            }
            //printf("\n+ + + + + + + + [9] Рассматривается 2й случай. Продолжаем работу");
            //printf("\n+ + + + + + + + + [10] Условие не выполнено. Число %d простое. Выход", n);
            return true;
         }
         //printf("\n+ + + + + + [7] Условие не выполнено. Переходим к шагу 8");
         r++;
         if (r >= n){
            //printf("\n+ + + + + + + [8] Условие выполнено. Число %d простое. Выход", n);
            return true;
         }
         else {
            printf("");
            //printf("\n+ + + + + + + [8] Условие не выполнено. Переходим к шагу 3")
            };
      }
      else
      {
         //printf("\n+ + + + [5] r = %d - не простое. Переходим к шагу 8", r);
         r++;
         if (r >= n){
            //printf("\n+ + + + + + + [8] Условие выполнено. Число %d простое. Выход", n);
            return true;
         }
         else {
            printf("");
            //printf("\n+ + + + + + + [8] Условие не выполнено. Переходим к шагу 3")
            };
      }
   }
   return true;
}

int main()
{
   srand(time(0));
   setlocale(0, "");
   int n;

   printf("\nВведите число n: ");
   n = validated_input();

   printf("\nПроверим число %d на простоту по полиномиальному тесту AKS\n  ", n);

   AKS(n) ?
      printf("\n\n[Результат] Число %d простое\n\n", n):
      printf("\n\n[Результат] Число %d составное\n\n", n);
```

```
    return 0;
}
```

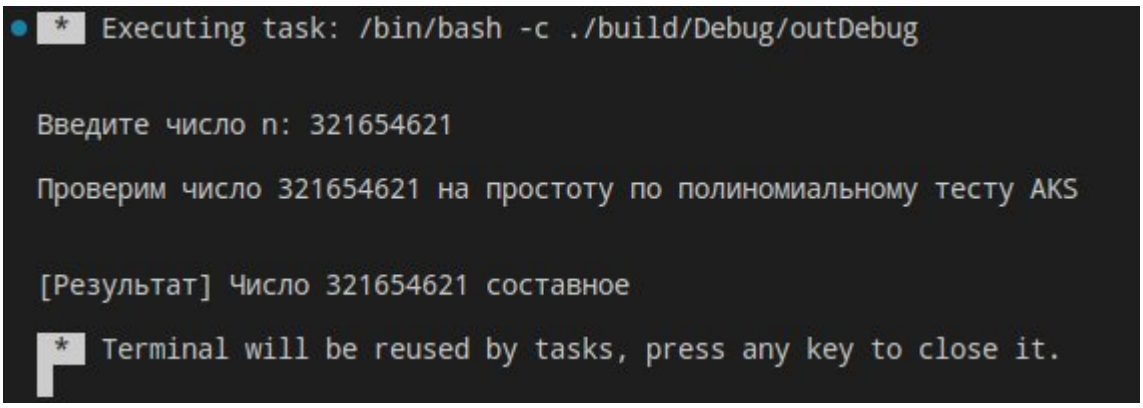
Пример запуска программы

```
 *  Executing task: /bin/bash -c ./build/Debug/outDebug

Введите число n: 1700

Проверим число 1700 на простоту по полиномиаотному тесту AKS

[1] Число 1700 четное - оно не простое. Выход


[Результат] Число 1700 составное

 *  Terminal will be reused by tasks, press any key to close it.
```

```
 *  Executing task: /bin/bash -c ./build/Debug/outDebug

Введите число n: 1105

Проверим число 1105 на простоту по полиномиаотному тесту AKS

[1] Число 1105 может быть простым. Продолжаем работу

+ [2] Положим значение r = 2

+ + [3] Выполним шаги 4-8, текущее значение r = 2

+ + + [4] НОД(2, 1105) < 1. Продолжаем работу
+ + + + [5] r = 2 - простое. Выполним шаги 6-7
+ + + + + [6] Наибольший простой делитель q = 1
+ + + + + + [7] Условие не выполнено. Переходим к шагу 8
+ + + + + + + [8] Условие не выполнено. Переходим к шагу 3

+ + [3] Выполним шаги 4-8, текущее значение r = 3

+ + + [4] НОД(3, 1105) < 1. Продолжаем работу
+ + + + [5] r = 3 - простое. Выполним шаги 6-7
+ + + + + [6] Наибольший простой делитель q = 1
+ + + + + + [7] Условие не выполнено. Переходим к шагу 8
+ + + + + + + [8] Условие не выполнено. Переходим к шагу 3

+ + [3] Выполним шаги 4-8, текущее значение r = 4

+ + + [4] НОД(4, 1105) < 1. Продолжаем работу
+ + + + [5] r = 4 - не простое. Переходим к шагу 8
+ + + + + + + [8] Условие не выполнено. Переходим к шагу 3

+ + [3] Выполним шаги 4-8, текущее значение r = 5

+ + + [4] НОД(5, 1105) >  1. Число 1105 составное. Выход

[Результат] Число 1105 составное

 *  Terminal will be reused by tasks, press any key to close it.
```


* Executing task: /bin/bash -c ./build/Debug/outDebug
Введите число n: 2707
Проверим число 2707 на простоту по полиномиальному тесту AKS
[Результат] Число 2707 простое
* Terminal will be reused by tasks, press any key to close it.

```
 *  Executing task: /bin/bash -c ./build/Debug/outDebug


Введите число n: 11

Проверим число 11 на простоту по полиномиаотному тесту AKS

[1] Число 11 может быть простым. Продолжаем работу

+ [2] Положим значение r = 2

+ + [3] Выполним шаги 4-8, текущее значение r = 2

+ + + [4] НОД(2, 11) < 1. Продолжаем работу
+ + + + [5] r = 2 - простое. Выполним шаги 6-7
+ + + + + [6] Наибольший простой делитель q = 1
+ + + + + + [7] Условие не выполнено. Переходим к шагу 8
+ + + + + + + [8] Условие не выполнено. Переходим к шагу 3

+ + [3] Выполним шаги 4-8, текущее значение r = 3

+ + + [4] НОД(3, 11) < 1. Продолжаем работу
+ + + + [5] r = 3 - простое. Выполним шаги 6-7
+ + + + + [6] Наибольший простой делитель q = 1
+ + + + + + [7] Условие не выполнено. Переходим к шагу 8
+ + + + + + + [8] Условие не выполнено. Переходим к шагу 3

+ + [3] Выполним шаги 4-8, текущее значение r = 4

+ + + [4] НОД(4, 11) < 1. Продолжаем работу
+ + + + [5] r = 4 - не простое. Переходим к шагу 8
+ + + + + + + [8] Условие не выполнено. Переходим к шагу 3

+ + [3] Выполним шаги 4-8, текущее значение r = 5

+ + + [4] НОД(5, 11) < 1. Продолжаем работу
+ + + + [5] r = 5 - простое. Выполним шаги 6-7
+ + + + + [6] Наибольший простой делитель q = 1
+ + + + + + [7] Условие не выполнено. Переходим к шагу 8
+ + + + + + + [8] Условие не выполнено. Переходим к шагу 3

+ + [3] Выполним шаги 4-8, текущее значение r = 6

+ + + [4] НОД(6, 11) < 1. Продолжаем работу
+ + + + [5] r = 6 - не простое. Переходим к шагу 8
+ + + + + + + [8] Условие не выполнено. Переходим к шагу 3

+ + [3] Выполним шаги 4-8, текущее значение r = 7

+ + + [4] НОД(7, 11) < 1. Продолжаем работу
+ + + + [5] r = 7 - простое. Выполним шаги 6-7
+ + + + + [6] Наибольший простой делитель q = 1
+ + + + + + [7] Условие не выполнено. Переходим к шагу 8
+ + + + + + + [8] Условие не выполнено. Переходим к шагу 3

+ + [3] Выполним шаги 4-8, текущее значение r = 8

+ + + [4] НОД(8, 11) < 1. Продолжаем работу
+ + + + [5] r = 8 - не простое. Переходим к шагу 8
+ + + + + + + [8] Условие не выполнено. Переходим к шагу 3

+ + [3] Выполним шаги 4-8, текущее значение r = 9

+ + + [4] НОД(9, 11) < 1. Продолжаем работу
+ + + + [5] r = 9 - не простое. Переходим к шагу 8
+ + + + + + + [8] Условие не выполнено. Переходим к шагу 3

+ + [3] Выполним шаги 4-8, текущее значение r = 10

+ + + [4] НОД(10, 11) < 1. Продолжаем работу
+ + + + [5] r = 10 - не простое. Переходим к шагу 8
+ + + + + + + [8] Условие выполнено. Число 11 простое. Выход

[Результат] Число 11 простое

 *  Terminal will be reused by tasks, press any key to close it.
```


* Executing task: /bin/bash -c ./build/Debug/outDebug
Введите число n: 321654621
Проверим число 321654621 на простоту по полиномиальному тесту AKS
[Результат] Число 321654621 составное
* Terminal will be reused by tasks, press any key to close it.