```
#include <stdio.h>
#include <stdlib.h>
#include <locale.h>
#include <stdbool.h>
#include <string.h>
#include <unistd.h>
#include <math.h>

/*
================================================I
  TDE de Criptografia RSA em C, componentes:    I
  João Pedro Thomaz Kairalla dos Santos (6898366)I
  Lucas Ferreira de Brito               (8697002)I
  Miguel Mussalam Silva                 (1123299)I
  Thiago Danilow de Araujo              (9979454)I
================================================I
*/

// Menu
void menu();

// Função de verificação de números primos
bool verificaPrimo(long long num);

// Função que descobre o produto de p e q
long long produtopq(long long p,long long q);

//Função que descobre o phi por meio do totiente de euler
long long totienteEuler(long long p,long long q);
```

```
// Função que faz a divisão recursiva de phi e 'e', guardando o quociente e o resto para a proxima funcao
void divisao(long long a,long long b,long long *quociente,long long *resto);

// Função que procura o número d que será usado na chave privada, sendo usada a função divisão dentro dela
long long achar_d(long long e,long long produto);

// Criptografia caractere por caractere usando a formula C = M ^ e mod n
long long potencia(long long a, long long e, long long produto);

// Função que criptografa a mensagem caractere por caractere
long long *criptografar(char* mensagem,long long e,long long produto);

// Descriptografa caractere por caractere usando a formula D = M ^ d mod n
long long *descriptografar(long long* mensagemCript,long long d,long long produto);

int main()
{
   setlocale(LC_ALL,"Portuguese");

//  Numeros que serão a base do cálculo de chaves públicas e privadas
   long long p,q,produto,phi,e = 65537,d,chavePublica[1],chavePrivada[1],*mensagemCript;
   int qtdBit;

//  Mensagem que sera criptografada
   char *mensagem;

//  variavel que guardara a mensagem descriptografada
   char *mensagemDecript;

//  loop infinito do codigo
```

```
   while(1){
   mensagem = (char *)malloc(100 * sizeof(char));


   bool primo = false,iguais = false;


//  Do while da Aceitação apenas das entradas permitidas (Somente números,primos e diferentes)
   do
   {
      menu();


      primo = false,iguais = false;


      printf("p <- Digite o primeiro número primo: ");
      while(fflush(stdin) || !scanf("%lld",&p)){
         printf("\nAVISO: São aceitos apenas números\n\n");
         system("pause");
         system("cls");
         menu();
         printf("p <- Digite o primeiro número primo: ");
      }
      printf("q <- Digite o segundo número primo: ");
      while(fflush(stdin) || !scanf("%lld",&q)){
         printf("\nAVISO: São aceitos apenas números\n\n");
         system("pause");
         system("cls");
         menu();
         printf("p <- Digite o primeiro número primo: %lld\n",p);
         printf("q <- Digite o segundo número primo: ");
      }
      if(p == q) iguais = true;
```

```
        primo = verificaPrimo(p);
        primo = verificaPrimo(q);
        if(p <= 10 || q <= 10){
           printf("São aceitos apenas números maiores que 10, tente novamente\n");
           system("Pause");
           system("cls");
        }
        else if(iguais){
          printf("Os números são iguais, tente novamente\n");
          system("pause");
          system("cls");
        }
        else if(!primo){
           printf("\nSão aceitos apenas números primos, tente novamente\n\n");
           system("pause");
           system("cls");
        }
    }
    while(iguais == true || primo == false || p <= 10 || q <= 10);

//  Leitura da entrada de mensagem
    printf("Digite a mensagem que sera criptografada(até 100 caracteres):");
    fflush(stdin);
    fgets(mensagem,100,stdin);
    system("cls");
    printf("p(%lld) q(%lld)\n\n",p,q);
    printf("mensagem: %s\n",mensagem);

//  produto de p e q
    produto = produtopq(p,q);
```

```
// calculo do totiente de Euler
   phi = totienteEuler(p,q);

// Chave pública
   chavePublica[0] = e;
   chavePublica[1] = produto;
   printf("Chave publica(%lld,%lld)\n\n",chavePublica[0],chavePublica[1]);

// Calculo de d
   d = achar_d(phi,e);

// Chave Privada
   chavePrivada[0] = d;
   chavePrivada[1] = produto;
   printf("Chave privada(%lld,%lld)\n\n",chavePrivada[0],chavePrivada[1]);

// Conta para calcular a quantidade de bits da RSA produzida
   qtdBit = log2(produto);

// Guardando na variavel de mensagem criptografada, a função de criptografia
   mensagemCript = criptografar(mensagem,e,produto);

// Printando a criptografia
   printf("Mensagem Criptografada: ");
   int i;
   for(i = 0; i < strlen(mensagem); i++)
   {
      printf("%lld",mensagemCript[i]);
   }
   printf("\n\n");
```

```
//  Guardando na variavel de mensagem descriptografada, a função de descriptografia
   mensagemDecript = descriptografar(mensagemCript,d,produto);

//  Printando a descriptografia
   printf("Mensagem descriptografada: ");
   for(i = 0; i < strlen(mensagem); i++)
   {
      printf("%c",mensagemDecript[i]);
   }
   printf("\n");
   printf("Quantidade de bits da criptografia simulada é de %d bits, ou seja, [RSA
%d]\n\n",qtdBit,qtdBit);
   system("pause");
   system("cls");

//  limpando as memórias para o próximo loop
   free(mensagem);
   free(mensagemDecript);
   free(mensagemCript);
   }
   return 0;
}

void menu(){

   printf("\t           Criptrafia RSA em C             \n");
   printf("\t=============================================\n");
   printf("\t|       Componentes do grupo        |   RA  |\n");
   printf("\t|João Pedro Thomaz Kairalla dos Santos|6898366|\n");
   printf("\t|Lucas Ferreira de Brito            |8697002|\n");
   printf("\t|Miguel Mussalam Silva              |1123299|\n");
```

```
    printf("\t|Thiago Danilow de Araujo            |9979454|\n");
    printf("\t==============================================\n\n");
    printf("\t A entrada necessita ser primo e maior que 10\n\n");
}

bool verificaPrimo(long long num)
{
    // 1 não é primo
    if (num == 1) return false;

    // 2 é primo
    if (num == 2) return true;

    // Números pares não são primos (exceto o 2)
    if (num % 2 == 0) return false;

    long long Raiznum = sqrt(num),i = 0;
    for (i = 3; i <= Raiznum; i += 2)
    {
        if (num % i == 0) return false;
    }
    return true;
}

long long produtopq(long long p,long long q)
{
    //Calculo de n
    long long n = p * q;
    return n;
}
```

```
long long totienteEuler(long long p,long long q)
{
   //Calculo de phi
   long long phi = (p - 1) * (q - 1);
   return phi;
}

void divisao(long long a,long long b,long long *quociente,long long *resto)
{
   //Divisão recursiva
   if (b > a)
   {
      *quociente = 0;
      *resto = a;
   }
   else
   {
      divisao(a - b, b, quociente, resto);
      (*quociente)++;
   }
}

// Procurando o d por meio do cálculo de MDC estendido, usando ambos phi e 'e' como
parâmetros do MDC
long long achar_d(long long a,long long b)
{
   long long resto, quociente, xB = 1, yB = 0, x = 0, y = 1, alpha, beta,phi;
   phi = a;

   resto = a;
   while (resto != 0)
```

```
    {
        divisao(a, b, &quociente, &resto);
        a = b;
        b = resto;
        if (resto > 0)
        {
            alpha = xB - quociente * x;
            beta = yB - quociente * y;

            xB = x;
            yB = y;
            x = alpha;
            y = beta;
        }
    }
    if (beta < 0)
    {
        beta = phi + beta;
    }

    return beta;
}

// Função de potência que muda binariamente o caractere da mensagem que será criptografada
long long potencia(long long a, long long e, long long produto)
{

    long long A = a, P = 1, E = e;

    while(1)
```

```
    {

        if(E == 0)
            return P;

        else if(E%2 != 0)
        {
            P = (A * P) % produto;
            E = (E-1)/2;
        }

        else
        {
            E = E/2;
        }
        A = (A*A) % produto;
    }
}
// Criptografia caractere por caractere usando a formula C = M ^ e mod n
long long *criptografar(char* mensagem, long long e, long long produto)
{

    long long *mensagemCript;
    int i;

    mensagemCript = malloc(100 * sizeof(long long));

    for (i = 0; mensagem[i] != '\0'; i++)
    {
        mensagemCript[i] = potencia((long long)mensagem[i], e, produto);
    }
```

```
    return mensagemCript;
}

// Descriptografa caractere por caractere usando a formula D = M ^ d mod n
long long *descriptografar(long long* mensagemCript,long long d,long long produto)
{

    char *mensagemDecript;
    int i;
    mensagemDecript = (char *)malloc(100 * sizeof(char));

    for(i = 0; i < 100; i++)
    {
        mensagemDecript[i] = potencia(mensagemCript[i],d,produto);
    }
    return mensagemDecript;
}
```