SUBSTITUTION CIPHER

STUDENTS: BASTIDA PRADO JAIME ARMANDO

SOLORIO PAREDES DANIEL

PROFESSOR: DÍAZ SANTIAGO SANDRA

SUBJECT: CRYPTOGRAPHY

GROUP: 3CM6

February 2nd 2019

1. First Programming Exercise

This programm is an implementation of the affine cipher considering the set of printable ASCII characters.

There are three important parts. The first one is the algorithm that validates the "a" value provided by the user. This part is basically computating the gcd(a, 96) because 96 is the number of printable ASCII characters.

If the gcd between "a" and 96 is not 1, then the programm asks for a valid value.

```
//Validating "a" using the Euclid algorithm
n = 96;
remainder = 0;
aux = a;
while(1)
{
    remainder = aux % n;
    aux = n;
    n = remainder;
    if(remainder == 0)
        break;
    previousRemainder = remainder;
}
if(previousRemainder != 1)
{
    printf("|+|+|+|ERROR: Value for \"a\" not in the set Zn* where n = 96. Try again.\n\n\n");
    goto tryagaina;
}
```

Figura 1: gcd algorithm

The second part, where the encipher occurs.

First the program checks if the character read is in the printable ASCII set. Then we calculate the ciphertext by using the following formula:

$$c_i = (am_i + b) mod\ n$$

where n = 96, a and b provided by the user.

```
//Ciphering and writing down to the new file
while((ch = getc(read_fp)) != EOF)
{
    if(ch > 31 && ch < 128)
        putc((((a * (ch - 32)) + b) % 96) + 32, write_fp);
    else
        putc(ch, write_fp);
}
```

Figura 2: Encipher algorithm

The third part consist of calculate the inverse of "a", this is done using the algorithm depicted in the next picture. It uses two arrays, one stores the ecuations generated when applying the Extended Euclid algorithm, the second stores the values of the numbers that multiply each factor in each ecuation.

```
//Calculating inverse of a using the Extended Euclid Algorithm
n = 96;
index = 0;
while(1)
{
    remainder = n % a;
    if(remainder == 0)
        break;

    array[index][0] = remainder;
    array[index][1] = n;
    array[index][2] = a;
    array[index][3] = - (n - remainder) / a;

    n = a;
    a = remainder;
    index++;
}
array1[0][0] = 1;
array1[0][1] = array[0][3];
factor1 = array[0][1];
factor2 = array[0][2];
for(i = 1; i < index; i++)
{
    product1 = array[i][3] * array1[i - 1][0];
    product2 = array[i][3] * array1[i - 1][1];

    if(array[i][1] == factor1)
        product1++;
    else if(array[i][1] == factor2)
        product2++;
```

Figura 3: Extended Euclid Algorithm

```
for(i = 1; i < index; i++)
{
    product1 = array[i][3] * array1[i - 1][0];
    product2 = array[i][3] * array1[i - 1][1];

    if(array[i][1] == factor1)
        product1++;
    else if(array[i][1] == factor2)
        product2++;
    else
    {
        for(j = 0; j < index; j++)
            if(array[j][0] == array[i][1])
                break;
        product1 += array1[j][0];
        product2 += array1[j][1];
    }
    array1[i][0] = product1;
    array1[i][1] = product2;
}
aInverse = array1[index - 1][1];
if(aInverse < 0)
    aInverse = 96 - (-aInverse % 96);
```

Figura 4: Extended Euclid Algorithm

The last one is the decipher part. Here, we calculate using the following formula:

$$d_i = (a^{-1}(c_i - b))mod\ n$$

where n = 96, inverse of a is calculated by the program and b is given by the user.

```
//Ciphering and writing down to the new file
while((ch = getc(read_fp)) != EOF)
{
    if(ch > 31 && ch < 128)
    {
        if((ch - 32) < b)
            putc((96 - (-(aInverse * (ch - 32 - b)) % 96)) + 32, write_fp);
        else
            putc(((aInverse * (ch - 32 - b)) % 96) + 32, write_fp);
    }
    else
        putc(ch, write_fp);
}
fclose(write_fp);
```

Figura 5: Decipher algorithm

2. Second Programming exercise

This program implements another kind of substitution algorithm that implies the use of a keyword.

The most important part of this program is the enchiper and dechipher which are very similar. To enchipher we use a variable named "index" which travels along the array adding the ith value with the character read, as we learn in the lab. If the character is not in the set of printable characters then we just put it as it is. To decipher we do the same but instead of adding we subtract the ith value of the array.

```
//CIPHERING and writing down to the new file
index = 0;
while((ch = getc(read_fp)) != EOF)
{
    if(ch > 31 && ch < 128)
    {
        if(index >= k_len)
            index = 0;
        putc((((ch - 32) + key[index++]) % 96) + 32, write_fp);
    }
    else
        putc(ch, write_fp);
}
```

Figura 6: Encipher algorithm