



# Parallel Programming

## Laboratory 9

~ 2022 ~

Balázs Benedek

Group 30444/1



## Problem:

Some screenshots with the execution (pairs for encrypt / decrypt)

```
C lab09.c U X C stdlib.h
Lab09 > C lab09.c > encryptFile()
27

PROBLEMS 1 OUTPUT TERMINAL GITLENS DEBUG CONSOLE
PS E:\TNT\University\__4YEAR__\AN4_SEM2\PP\lab09> ./lab09
> Keyword: BENEDEKBALAZS
> Seed = 939
> 0 = 10
> 1 = 255
> 2 = 11
> 3 = 63
> 4 = 4
> 5 = 100
> 6 = 116
> 7 = 51
> 8 = 145
> 9 = 27
> 10 = 32
> 11 = 251
> 12 = 191
> 13 = 93
> 14 = 18
> 15 = 44
> 16 = 137
> 17 = 143
> 18 = 120
> 19 = 164
> 20 = 159
> 21 = 126
> 22 = 194
> 23 = 215
> 24 = 5
> 25 = 198
> 26 = 102
> 27 = 217
> 28 = 48
> 29 = 14
> 30 = 129
> 31 = 124
> 32 = 46
> 33 = 40
> 34 = 249
> 35 = 219
> 36 = 59
> 37 = 206
> 38 = 104
> 39 = 128
> 40 = 218
> 41 = 229
> 42 = 90
> 43 = 61
> 44 = 118
> 45 = 186
> 46 = 110
> 47 = 0

C lab09.c U X C stdlib.h
Lab09 > C lab09.c > encryptFile()
27

PROBLEMS 1 OUTPUT TERMINAL GITLENS
> 239 = 75
> 240 = 210
> 241 = 241
> 242 = 99
> 243 = 122
> 244 = 158
> 245 = 28
> 246 = 196
> 247 = 184
> 248 = 130
> 249 = 41
> 250 = 228
> 251 = 82
> 252 = 171
> 253 = 235
> 254 = 152
> 255 = 242
> 0 = 47
> 1 = 235
> 2 = 109
> 3 = 90
> 4 = 4
> 5 = 24
> 6 = 64
> 7 = 137
> 8 = 111
> 9 = 144
> 10 = 0
> 11 = 2
> 12 = 134
> 13 = 135
> 14 = 29
> 15 = 183
> 16 = 103
> 17 = 58
> 18 = 14
> 19 = 157
> 20 = 168
> 21 = 97
> 22 = 176
> 23 = 129
> 24 = 181
> 25 = 209
> 26 = 210
> 27 = 9
> 28 = 245
> 29 = 221
> 30 = 164
> 31 = 82
> 32 = 10
> 33 = 73
```



## Results:

The result is becoming the same, after running the program.

C lab09.c U Lab09\_Enc\_Dec.pdf X

### Lab 09 - Cryptography (1)

**1. Programming Setup**  
Use whatever OS and programming language you prefer.

**2. PROBLEM – Monoalphabetic substitution using a pseudorandom permutation**

All monoalphabetic substitution encryption algorithms do the same thing: compute a permutation of the alphabet somehow; they then apply that permutation to the message to compute the cryptogram.

Example: Caesar's cipher computes the permutation of the alphabet by circularly shifting the alphabet to the left with K positions (K is the key here, an integer, typical value is K=3); thus A will be replaced (substituted) by D, B by E etc.

Mathematically, this actually computes a permutation of the alphabet (considering the English alphabet A = 0, B = 1, ..., Z = 25) by addition modulo 26 (the size of the alphabet):

$$y = (x+K) \bmod 26$$

## Code:

```
#include <stdio.h>
#include <stdlib.h>

/* defines */
#define SIZE_OF_KEYWORD 14

/* global variables */
char keyword[SIZE_OF_KEYWORD] = "BENEDEKBALAZS";
int seed;
int alphabet[256];
int random[1024] = {0};

/* prototypes */
int computeSeed(char keyword[]);
void swap(int posX, int posY);
void encryptFile();
void decryptFile();

/* the main function of the file */
```



Parallel Programming - Laboratory

```
int main()
{
    /* the encryption part of the assignment */
    encryptFile();

    /* the decryption part of the assignment */
    decryptFile();

    return 0;
}

void encryptFile()
{
    FILE* readFile = fopen("Lab_09.pdf", "rb");
    FILE* writtenFile = fopen("Lab_09.enc", "wb");

    fseek(readFile, 0, SEEK_END);
    long size = ftell(readFile);
    fseek(readFile, 0, SEEK_SET);
    unsigned char character;
    unsigned char newCharacter;

    seed = computeSeed(keyword);
    srand(seed);

    /* initializing the alphabet and frequency arrays */
    for (int i = 0; i < 256; alphabet[i] = i, ++i);

    /* creating the vector of random values in range [0 - 255] */
    for (int i = 0; i < 1024; random[i] = (int)rand() % 256, ++i);

    /*
     * performing a swap between every 2 position, where these 2 values
     * are taken from the i-th, and (i + 1)st position of the random vector
     */
    for(int i = 0; i < 1023; ++i)
    {
        swap(i, i + 1);
    }

    /* print the value and its encrypted mate */
    for(int i = 0; i < 256; ++i)
```



Parallel Programming - Laboratory

```
{
    printf("> %d = %d\n", i, alphabet[i]);
}

/* read the characters from the pdf, and write the encrypted pair of it */
for (long i = 0; i < size; ++i)
{
    fread(&character, 1, 1, readFile);
    newCharacter = alphabet[character];
    fwrite(&newCharacter, 1, 1, writtenFile);
}

fclose(readFile);
fclose(writtenFile);
}

void decryptFile()
{
    FILE* readFile = fopen("Lab_09.enc", "rb");
    FILE* writtenFile = fopen("Lab09_Enc_Dec.pdf", "wb");

    fseek(readFile, 0, SEEK_END);
    long size = ftell(readFile);
    fseek(readFile, 0, SEEK_SET);
    unsigned char character;
    unsigned char newCharacter;

    srand(seed);

    /* initializing the alphabet and frequency arrays */
    for (int i = 0; i < 256; alphabet[i] = i, ++i);

    /* creating the vector of random values in range [0 - 255] */
    for (int i = 0; i < 1024; random[i] = (int)rand() % 256, ++i);

    /*
     * performing a swap between every 2 position, where these 2 values
     * are taken from the i-th, and (i - 1)st position of the random vector
     */
    for(int i = 1023; i > 0; --i)
    {
        swap(i, i - 1);
    }
}
```



Parallel Programming - Laboratory

```
}

/* print the value and its decrypted mate */
for(int i = 0; i < 256; ++i)
{
    printf("> %d = %d\n", i, alphabet[i]);
}

/* read the characters from the encrypted file, and write the decrypted pair
of it */
for (long i = 0; i < size; ++i)
{
    fread(&character, 1, 1, readFile);
    newCharacter = alphabet[character];
    fwrite(&newCharacter, 1, 1, writtenFile);
}

fclose(readFile);
fclose(writtenFile);
}

/* computing the seed by keyword */
int computeSeed(char keyword[])
{
    int seed = 0;

    printf("> Keyword: ");

    for (int i = 0; i < SIZE_OF_KEYWORD; ++i)
    {
        seed += keyword[i];
        printf("%c", keyword[i]);
    }

    printf("\n> Seed = %d\n", seed);

    return seed;
}

/* the swap function */
void swap(int posX, int posY)
{
```



---

Parallel Programming - Laboratory

```
int temp = alphabet[random[posX]];
alphabet[random[posX]] = alphabet[random[posY]];
alphabet[random[posY]] = temp;
}
```