PRAKTIKUM 12
IMPLEMENTASI BEBERAPA ALGORITMA

12.1 TUJUAN PRAKTIKUM

Tujuan Umum

Mahasiswa dapat memahami implementasi beberapa algoritma sorting, searching, dan kriptografi klasik menggunakan bahasa pemrograman C

Tujuan Khusus

Mahasiswa dapat :

1. Memahami Implementasi algoritma sorting pada bahasa C
2. Memahami Implementasi algoritma searching pada bahasa C
3. Memahami Implementasi algoritma kriptografi klasik pada bahasa C

12.2 PELAKSANAAN PRAKTIKUM

12.2.1 Algoritma Bubble Sort

Terdapat banyak metode pengurutan data yang terdapat dalam teori algoritma dan pemrograman, diantaranya metode gelembung (bubble sort), sisipan (insertion sort), seleksi (selection sort), dan banyak lagi yang lainnnya. Namun, dari banyak metode yang ada tersebut, di sini kita hanya akan mengimplementasikan bubble sort ke dalam sebuah program.
Bubble sort yang disebut juga dengan sinking sort atau exchange sort merupakan cara pengurutan elemen yang paling sederhana, menggunakan metode pembandingan dan pertukaran pada tiap putaran, elemen yang bersebelahan akan dibandingkan dan isinya akan langsung ditukar jika nilainya tidak berurut.

Praktikum 12.1 Algoritma Bubble Sort

```c
#include <stdio.h>

main() {
 int a[50], n, i, j, temp = 0;

 printf("Input jumlah bilangan\t: ");
 scanf("%d", &n);

 for (i = 0; i < n; i++) {
    printf("Input bilangan ke-%d: ", i+1);
    scanf("%d", &a[i]);
 }

 printf("\nBilangan sebelum terurut\t: ");
 for (i = 0; i < n; i++) {
  printf("%d  ", a[i]);
 }

```

```
for (i = 0; i < n; i++) {
 for (j = i + 1; j < n; j++) {
  if (a[i] > a[j]) {
   temp = a[i];
   a[i] = a[j];
   a[j] = temp;
  }
 }
}

printf("\nBilangan terurut\t\t: ");
for (i = 0; i < n; i++) {
 printf("%d  ", a[i]);
}
}
```

12.2.2 Algoritma Sequential Search

Searching adalah sebuah proses untuk mendapatkan (retrieve) informasi berdasarkan kunci (key) tertentu dari sejumlah informasi yang telah disimpan. Pencarian Sekuensial (sequential searching) atau pencarian berurutan sering disebut pencarian linear merupakan metode pencarian yang paling sederhana. Pencarian sekuensial adalah proses membandingkan setiap elemen larik satu per satu secara beruntun, mulai dari elemen pertama sampai elemen yang dicari ditemukan atau seluruh elemen sudah diperiksa.

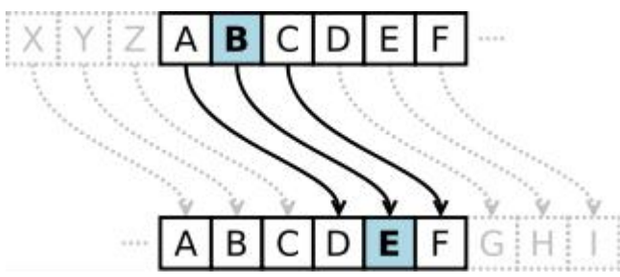
Praktikum 12.2 Algoritma Sequential Search

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

main()
{
    int arr[50],i,j,k,l, pos[50];
    printf("Input jumlah bilangan random (maks. 50) : ");
    scanf("%d",&l);
    printf("Generate %d bilangan random\n",l);

    for( i = 0 ; i < l ; i++ ) {
        arr[i] = rand() % 99;
        printf("%d ", arr[i]);
    }

    printf("\nInput binlangan yang ingin dicari : ");
    scanf("%d",&k);
    j = 0;
    for(i=0;i<l;i++)
    {
        if(arr[i]==k)
        {
            pos[j] = i + 1;
            j++;
        }
    }
```

```c
    if(pos!=0)
    {
        printf("%d is found in the list, at position ",k);
        for (i = 0; i < j; i++) {
            printf("%d  ", pos[i]);
        }

    } else {
        printf("%d is not in the list\n",k);
    }
}
```

12.2.3 Algoritma Kriptografi Caesar Cipher

Dalam kriptografi, sandi Caesar, atau sandi geser, kode Caesar atau Geseran Caesar adalah salah satu teknik enkripsi paling sederhana dan paling terkenal. Sandi ini termasuk sandi substitusi dimana setiap huruf pada teks terang (plaintext) digantikan oleh huruf lain yang memiliki selisih posisi tertentu dalam alfabet. Misalnya, jika menggunakan geseran 3, W akan menjadi Z, I menjadi L, dan K menjadi N sehingga teks terang "wiki" akan menjadi "ZLNL" pada teks tersandi. Nama Caesar diambil dari Julius Caesar, jenderal, konsul, dan diktator Romawi yang menggunakan sandi ini untuk berkomunikasi dengan para panglimanya.



Contoh

Untuk menyandikan sebuah pesan, cukup mencari setiap huruf yang hendak disandikan di alfabet biasa, lalu tuliskan huruf yang sesuai pada alfabet sandi. Untuk memecahkan sandi tersebut gunakan cara sebaliknya. Contoh penyandian sebuah pesan adalah sebagai berikut.

teks terang : kirim pasukan ke sayap kiri
teks tersandi : NLULP SDVXNDQ NH VDBDS NLUL

Praktikum 12.3 Algoritma Caesar Cipher - Enkripsi/Dekripsi Teks

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

char* decrypt(char arr[], int key)
{
    int i;
    for(i = 0; i < strlen(arr); i++)
    {
        arr[i] = arr[i] - key;
    }

    return arr;
}

```

```c
char* encrypt(char arr[], int key)
{
    int i;
    for(i = 0; i < strlen(arr); i++)
    {
        arr[i] = arr[i] + key;
    }
    return arr;
}

main()
{
    char teks[100], ch;
    int pil, kunci;
        do {
        system("cls");
            printf("Algoritma Caesar Cipher\n");
            printf("==========================\n");
            printf("Silahkan pilih menu berikut : \n");
            printf("1. Enripsi\n");
            printf("2. Dekripsi\n");
            printf("Pilihan anda (1/2) \t: ");
            scanf("%d", &pil);
            printf("\n");
            fflush(stdin);
            if(pil==1) {
              printf("Input plainteks\t: ");
              gets(teks);
              printf("Input kunci\t: ");
              scanf("%d", &kunci);
              printf("\nCipherteks\t: %s\n",encrypt(teks, kunci));
            } else if(pil==2) {
              printf("Input cipherteks\t: ");
              gets(teks);
              printf("Input kunci\t: ");
              scanf("%d", &kunci);
              printf("\nPlainteks\t: %s\n",decrypt(teks, kunci));
            } else {
              printf("\nPilihan anda tidak ada ");
            }

            printf("\nUlangi (y/n) ? ");
            scanf("%s", &ch);

       } while(ch=='y'||ch=='Y');
}
```

Praktikum 12.4 Algoritma Caesar Cipher - Enkripsi/Dekripsi File

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

char* decrypt(char arr[], int key)
{
    int i;
    for(i = 0; i < strlen(arr); i++)
    {
        arr[i] = arr[i] - key;
    }

    return arr;
}

char* encrypt(char arr[], int key)
{
    int i;
    for(i = 0; i < strlen(arr); i++)
    {
        arr[i] = arr[i] + key;
    }
    return arr;
}

main()
{
    char teks[100], ch, c, *cipherteks, path[100], cpath[100];
    FILE *fptr;
    int pil, kunci;
    do {
        system("cls");
        printf("Algoritma Caesar Cipher\n");
        printf("==========================\n");
        printf("Silahkan pilih menu berikut : \n");
        printf("1. Enripsi\n");
        printf("2. Dekripsi\n");
        printf("Pilihan anda (1/2) \t: ");
        scanf("%d", &pil);
        printf("\n");
        fflush(stdin);
        if(pil==1) {
          printf("Alamat file\t: ");
          gets(path);
          printf("Input kunci\t: ");
          scanf("%d", &kunci);
          fptr = fopen(path,"r");
            if (fptr == NULL){
               printf("Error! file tidak ditemukan.");
               exit(1);
            }

```

```
53              while (fgets(teks, sizeof teks, fptr) != NULL)
54                  printf("Plainteks\t: %s", teks);
55
56              cipherteks=encrypt(teks, kunci);
57              printf("\nCipherteks\t: %s\n",cipherteks);
58
59              fptr = fopen(strcat(path, ".enc"),"w");
60              if(fptr == NULL)
61              {
62                printf("Error!");
63                exit(1);
64              }
65
66              fprintf(fptr,"%s",cipherteks);
67              fclose(fptr);
68              printf("File cipherteks tersimpan di %s", path);
69          } else if(pil==2) {
70              printf("Alamat file\t: ");
71              gets(path);
72              printf("Input kunci\t: ");
73              scanf("%d", &kunci);
74              if ((fptr = fopen(path,"r")) == NULL){
75                 printf("Error! opening file");
76                 exit(1);
77              }
78
79              while (fgets(teks, sizeof teks, fptr) != NULL)
80                  printf("Cipherteks\t: %s", teks);
81              fclose(fptr);
82              printf("\nPlainteks\t: %s\n",decrypt(teks, kunci));
83          } else {
84              printf("\nPilihan anda tidak ada ");
85          }
86
87          printf("\nUlangi (y/n) ? ");
88          scanf("%s", &ch);
89
90      } while(ch=='y'||ch=='Y');
91  }
```

12.2.4 Algoritma Kriptografi Vigenère Cipher

Algoritma Vigenère adalah metode menyandikan teks alfabet dengan menggunakan deretan sandi Caesar berdasarkan huruf-huruf pada kata kunci. Sandi Vigenère merupakan bentuk sederhana dari sandi substitusi polialfabetik. Kelebihan sandi ini dibanding sandi Caesar dan sandi monoalfabetik lainnya adalah sandi ini tidak begitu rentan terhadap metode pemecahan sandi yang disebut analisis frekuensi.

Vigenère Cipher sebenarnya merupakan pengembangan dari Caesar Cipher. Pada Caesar Cipher, setiap huruf teks terang digantikan dengan huruf lain yang memiliki perbedaan tertentu pada urutan alfabet. Misalnya pada Caesar Cipher dengan geseran 3, A menjadi D, B menjadi E and dan seterusnya. Vigenère Cipher terdiri dari beberapa Caesar Cipher dengan nilai geseran yang berbeda.

Untuk menyandikan suatu pesan, digunakan sebuah tabel alfabet yang disebut tabel Vigenère atau menggunakan operasi matematis.

Enkripsi (penyandian) dengan Vigenère Cipher dapat dituliskan secara matematis, dengan menggunakan penjumlahan dan operasi modulus, yaitu:

$$C_i \equiv (P_i + K_i) \mod 26$$

atau
Ci = (Pi + Ki) – 26 kalau hasil penjumlahan Pi dan Ki lebih dari 26
Dan dekripsi,

$$P_i \equiv (C_i - K_i) \mod 26$$

atau
Pi = (Ci – Ki) + 26 kalau hasil pengurangan Ci dengan Ki minus

Dimana:
Ci = nilai desimal karakter ciphertext ke-i
Pi = nilai desimal karakter plaintext ke-i
Ki = nilai desimal karakter kunci ke-i
Nilai desimal karakter : A=0 B=1 C=2 ... Z=25

Sebagai contoh, jika plaintext adalah SANDINEGARA dan kunci adalah SIBER maka proses enkripsi yang terjadi adalah sebagai berikut :
Plainteks : SANDINEGARA
Kunci : SIBERSIBERS
Cipherteks : KIOHZFMHEIS

Ci = (Pi + Ki) mod 26
= (18 + 18) mod 26
= 36 mod 26
= 10

Ci=10 maka huruf cipherteks dengan nilai 2 adalah K . Begitu seterusnya dilakukan pergeseran sesuai dengan kunci pada setiap huruf hingga semua plainteks telah terenkripsi menjadi cipherteks . Setelah semua huruf terenkripsi maka proses dekripsinya dapat dihitung sebagai berikut:

Pi = (Ci – Ki) + 26
= (10 – 18) + 26
= –8 + 26
= 18

Pi=18 maka huruf plainteks dengan nilai 18 adalah S. Begitu seterusnya dilakukan pergeseran sesuai dengan kunci pada setiap huruf hingga semua cipherteks telah terdekripsi menjadi plainteks.

Praktikum 12.5 Algoritma Vigenere Cipher - Enkripsi teks

```c
#include <stdio.h>
#include <string.h>

char* encrypt(char arr[], char key[])
{
    int i;
    for(i = 0; i < strlen(arr); i++)
    {
        arr[i] = arr[i] + key[i];
    }
    return arr;
}

main()
{
    char teks[100], kunci[100], kuncibaru[100];
    int i;

    printf("Algoritma Vigenere Cipher\n");
    printf("=========================\n");
    printf("Input plainteks\t: ");
    gets(teks);
    printf("Input kunci\t: ");
    gets(kunci);

    // membuat kunci baru dari kunci yang diinput
    for(i=0;i<strlen(teks);i++) {
        kuncibaru[i] = kunci[i % strlen(kunci)] ;
    }

    printf("\nKunci baru\t: %s", kuncibaru);
    printf("\nCipherteks\t: %s\n",encrypt(teks, kuncibaru));
}
```

Latihan

1. Modifikasi kode pada praktikum 12.5 sehingga program dapat mendekripsi cipherteks menggunakan vigenere cipher.
2. Buatlah program implementasi vigenere cipher untuk men-enkripsi dan men-dekripsi file.