Fenerbahçe Üniversitesi

BLM 103 – Algoritmalar ve Programlama I

RLE Veri Sıkıştırma

Ahmet Hazar Haspolat

190301012

Ömer Sait Yorulmaz

190301025

- **Projenin Amacı**

- RLE (Run-Lenght-Encoding) kayıpsız bir veri sıkıştırma yöntemidir. Birçok büyük sıkıştırma algoritmalarının alt algoritmalarından biridir. Verileri sıkıştırma algoritmaları bir çok alanda çok farklı işler için kullanılabilmektedir. RLE de temel olarak tekrar eden verileri tespit edip bu tekrarları ortadan kaldırır.

```c
int main() {

    char processType;
    printf("Select the process type you want to get \n e for encoding \n d for decoding ...\n");
startAgain:
    scanf_s("%c", &processType);

    if (processType == 'e') {
        fileOperations(processType);
        //encode();
    }
    else if (processType == 'd') {
        fileOperations(processType);
    }
    else {
        printf("Invalid selection. Choose e (encoding) or d (decoding) ... \n");
        goto startAgain;
    }
    return 0;


}
```

```c
char* encode(char* src)
{
    int rLen;
    char count[MAXCHAR];
    int len = strlen(src);

    char* dest = (char*)malloc(sizeof(char) * (len * 5 + 1));

    int i, j = 0, k;

    for (i = 0; i < len; i++) {

        dest[j++] = src[i];

        rLen = 1;

        while (i + 1 < len && src[i] == src[i + 1]) {
            rLen++;
            i++;
        }

        if (src[i] != '\n') {
            sprintf_s(count, MAXCHAR, "%d", rLen);

            for (k = 0; *(count + k); k++, j++) {
                dest[j] = count[k];
            }
        }

    }

    dest[j] = '\0';
    return dest;
}
```

```
char* decode(char* src)
{
    int rLen;
    char count[3];
    char chr;
    int len = strlen(src);

    char* dest = (char*)malloc(sizeof(char) * (len * 100 + 1));


    int i, j = 0, k = 0;

    for (i = 0; i < len; i++) {

        chr = src[i];

        while (i + 1 < len && ((int)(src[i + 1])) < 64) {
            count[k] = src[i + 1];
            i++;
            k++;
        }

        sscanf_s(count, "%d", &rLen);

        for (int l = j; l < j + rLen; ++l) {
            dest[l] = chr;
        }

        j += rLen;

        k = 0;

    }
    dest[j] = '\n';
    dest[j + 1] = '\0';
    return dest;
}
```

```c
void fileOperations(char type) {



    FILE* filePointer;

    char* girisFilePtr = "giris.txt";
    char* sikistirilmisFilePtr = "sikistirilmis.txt";
    char* cikisFilePtr = "cikis.txt";

    char* encodeArr[MAXCHAR];
    char* decodeArr[MAXCHAR];
    char str[MAXCHAR];

    int i = 0;
    int line = 0;

    if (type == 'e') {
        fopen_s(&filePointer, girisFilePtr, "r");
        if (filePointer == NULL) {
            printf("Could not open file %s", girisFilePtr);
            printf("\nTry again");
        }
        else {
            while (fgets(str, MAXCHAR, filePointer) != NULL) {
                *(encodeArr + i) = encode(str);
                i++;
                line++;
            }
            fclose(filePointer);
            fopen_s(&filePointer, sikistirilmisFilePtr, "w");
            for (int j = 0; j < line; j++) {
                fputs(*(encodeArr + j), filePointer);
            }
        }
    }
```

```
else
{
    fopen_s(&filePointer, sikistirilmisFilePtr, "r");
    if (filePointer == NULL) {
        printf("Could not open file %s", sikistirilmisFilePtr);
        printf("\nTry again");
    }
    else
    {
        while (fgets(str, MAXCHAR, filePointer) != NULL) {
            *(decodeArr + i) = decode(str);
            i++;
            line++;
        }
        fclose(filePointer);
        fopen_s(&filePointer, cikisFilePtr, "w");
        for (int j = 0; j < line; j++) {
            fputs(*(decodeArr + j), filePointer);
        }
        fclose(filePointer);
    }
}
```

- **Kullanılan Araçlar**

Proje kapsamında 2 araç kullanılmaktadır.

CLion :

JetBrains ürünü olan CLion, güçlü bir IDE olarak C ve C++ desteği vermektedir. Asıl RLE mimarisinin tasarım, yapım ve test aşamaları CLion üzerinden gerçekleştirilmiştir.

Visual Studio 2019 :

Teslim aşamasında istenilen Microsoft'un derleyicisi olan VS'e kod aktarımı yapılıp, kodun zorunluluktan değiştirilmesi gereken kısımları değiştirilip projenin tamamlanması sağlanmıştır.

- **Sonuçlar**

Piyasada popüler olan sıkıştırma programlarının en temeldeki yaklaşımlarını anlayıp, vizyon olarak gelişim sağlanmış, derslerde görülen özellikle pointer ve pointer aritmetiklerinden yararlanılıp, konuların pekişmeleri sağlanmıştır.

Dosyaların github adresi: https://github.com/espeniola/Run-Length-Encoder-Decoder