Organización de Computadoras 66.20

Trabajo Práctico 0

Autor	Padron	Correo electrónico
Flórez Del Carpio, Christian	91011	chris.florez.d.c@gmail.com
Montenegro, Josefina	94289	mariajosefina.mont@gmail.com
Quino López, Julián	94224	julianquino2@gmail.com

Facultad de Ingeniería
Universidad de Buenos Aires
Av. Paseo Colón 850 - C1063ACV
Ciudad Autónoma de Buenos Aires - Rep. Argentina
Tel: +54 (11) 4343-0893 / 4343-0092
http://www.fi.uba.ar

Historial de Revisiones

Fecha	Revisor	Detalle
05/09/2017		Primera versión del TP

Resumen

El siguiente trabajo práctico tiene como finalidad determinar, para un determinado conjunto de palabras, cuáles de ellas son palíndromos, entendiendo como palabras a aquellas compuestas por letras [A-Z], números [0-9], guiones bajos y medios, es decir, cualquier combinación posible de los anteriormente mencionados.

1. Introducción

Pueden haber tres escenarios posibles, el caso en el cual el usuario ingresa archivo de entrada y salida, el caso en el que se ingresa un archivo de entrada solamente y por último el caso donde se recibe el archivo de salida. En caso de no proporcionar un archivo de texto como entrada, se requerirá ingresar el stream por entrada standard, con un máximo de 300 caracteres. Si no se especifica un archivo de salida, se mostrarán los resultados por salida standard.

2. Desarrollo

El algoritmo propuesto por el grupo consiste en parsear las palabras ingresadas para luego procesar una por una y decidir si son palíndromos o no, esto se realiza ya sea desde el archivo o utilizando el stream leído por entrada standard. Si se debe leer de la entrada standard, se crea un archivo auxiliar donde se escribe lo previamente ingresado por el usuario, a fines de reutilizar el código desarrollado para el caso donde se ingresa un archivo de texto.

2.1. Comandos para compilar y ejecutar el programa

Se puede compilar el programa con el siguiente comando:

```
$ gcc isPalindrome.c -o tp0
```

Y luego ejecutarlo con el comando:

```
$ ./tp0 -i input.txt -o output.txt
```

En caso de sólo querer especificar el archivo de entrada, debe ejecutarse, por ejemplo, de la siguiente manera:

```
$ ./tp0 -i input.txt
```

Análogamente si se quiere ingresar un archivo de salida:

```
$ ./tp0 -o output.txt
```

2.2. Otros comandos

Pueden utilizarse comandos tales como help y version, de la siguiente forma:

```
$ ./tp0 -h

$ ./tp0 -V
```

2.3. Código fuente

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>
#include <getopt.h>
#include <stdbool.h>
```

```c
#include <stdlib.h>

#define MAXLINEA 260
#define MAXCHARS 300

// Verifica que el archivo no est  vac o
bool empty(FILE *file) {
    long savedOffset = ftell(file);
    fseek(file, 0, SEEK_END);

    if (ftell(file) == 0) {
        return true;
    }

    fseek(file, savedOffset, SEEK_SET);
    return false;
}

// Primero se valida que el archivo exista, despu s que no est  vac o en caso
bool validFile(FILE *file, char modo, char *argopt) {
    if (file == NULL) {
        printf("El archivo %s no existe, por favor ingrese un archivo existente \
        return false;
    }

    if (empty(file) && modo != 'w') {
        printf("El archivo %s est  vac o, por favor ingrese un archivo no vac
        return false;
    }

    printf("Se recibi  el archivo %s \n", argopt);
    return true;
}

bool isPalindrome(char *palabra) {
    int posInicial, posFinal;
    posFinal = strlen(palabra) - 1;
    for (posInicial = 0; posInicial < strlen(palabra) / 2; posInicial++, posFinal
        if ((toupper(*(palabra + posInicial))) != (toupper(*(palabra + posFinal))
            return false;
        }
    }
    return true;
}

void seekPalindromes(char palabras[][MAXLINEA], FILE *archivo) {
    int contadorPalabra = 0;
    while (palabras[contadorPalabra][0] != '$') {
        if (isPalindrome(palabras[contadorPalabra])) {
            fputs(palabras[contadorPalabra], archivo);
            fputs("\n", archivo);
        }
        contadorPalabra++;
    }
```

```
}

void printPalindromes(FILE *archivo) {
    char bufferLinea[MAXLINEA];
    memset(&bufferLinea, 0, MAXLINEA);
    rewind(archivo);
    fgets(bufferLinea, MAXLINEA, archivo);
    printf("Las palabras pal ndromas detectadas son: \n");
    while (!feof(archivo)) {
        printf("%s", bufferLinea);
        memset(&bufferLinea, 0, MAXLINEA);
        fgets(bufferLinea, MAXLINEA, archivo);
    }
}

bool validCharacter(char character) {
    int asciiNumber = (int) character;
    if ((asciiNumber <= 57) && (asciiNumber >= 48)) {
        return true;
    }
    if ((asciiNumber <= 90) && (asciiNumber >= 65)) {
        return true;
    }
    if ((asciiNumber <= 122) && (asciiNumber >= 97)) {
        return true;
    }
    if (asciiNumber == 45) {
        return true;
    }
    if (asciiNumber == 95) {
        return true;
    }
    return false;
}

void parseLine(char *linea, char palabras[][MAXLINEA]) {
    bool salir = false;
    int contador = 0;
    int contDePalabrasGuardadas = 0;
    int contDeCaracteresGuardados = 0;
    while (salir == false) {
        if (validCharacter(linea[contador])) {
            palabras[contDePalabrasGuardadas][contDeCaracteresGuardados] = linea[
            contDeCaracteresGuardados++;
        } else if (contDeCaracteresGuardados != 0) {
            palabras[contDePalabrasGuardadas][contDeCaracteresGuardados] = '\0';
            contDeCaracteresGuardados = 0;
            contDePalabrasGuardadas++;
        }
        if ((linea[contador] == '\n') || (linea[contador] == '\0')) {
            salir = true;
        }
        contador++;
    }
```

```
    palabras[contDePalabrasGuardadas][0] = '$';
}

void processInput(FILE *inputFile, FILE *outputFile, bool showResultsInStdOut) {
    char bufferLinea[MAXLINEA];
    char palabras[MAXLINEA][MAXLINEA];
    // para reposicionar el puntero del archivo a la primera linea
    // lectura anticipada del archivo para q no de mas lecturas
    rewind(inputFile);
    do {
        fgets(bufferLinea, MAXLINEA, inputFile);
        parseLine(bufferLinea, palabras);  // carga en la matriz las palabras
        seekPalindromes(palabras, outputFile);
    } while (!feof(inputFile));

    fclose(inputFile);

    printf("Se proces  el archivo de entrada \n");

    if (showResultsInStdOut) {
        printPalindromes(outputFile);//usamos rewind(outputFile) para llevar el i
    }
    fclose(outputFile);
}


int main(int argc, char *argv[]) {
    int option = 0;
    const char *short_opt = "i:o:hV";
    struct option long_opt[] = {
            {"version", no_argument,       NULL, 'V'},
            {"help",    no_argument,       NULL, 'h'},
            {"input",   required_argument, NULL, 'i'},
            {"output",  required_argument, NULL, 'o'},
            {NULL, 0,                      NULL, 0}
    };
    FILE *inputFile = NULL;
    FILE *outputFile = NULL;
    bool takeStreamFromStdIn = false;
    bool showResultsInStdOut = false;
    char inputByStd[MAXCHARS];
    char *inputFileAux = "inputFileAux.txt";
    char *outputFileAux = "outputFileAux.txt";

    if (argc == 1) {
        printf("Debe ingresar alg n argumento, para mas informaci n ingrese -h
        return 0;
    }

    while ((option = getopt_long(argc, argv, short_opt, long_opt, NULL)) != -1) {
        switch (option) {
            case 'V':
                printf("TP #0 de la materia Organizaci n de Computadoras \n");
                printf("Alumnos: \n");
```

```
                printf("        Fl rez Del Carpio Christian\n  Montenegro Josefi
                return 0;
            case 'h':
                printf("Usage: \n");
                printf("        %s -h \n", argv[0]);
                printf("        %s -V \n", argv[0]);
                printf("        %s [options] \n", argv[0]);
                printf("Options: \n");
                printf("        -V, --version  Print version and quit. \n");
                printf("        -h, --help     Print this information. \n");
                printf("        -o, --output   Location of the output file. \n");
                printf("        -i, --input    Location of the input file. \n");
                return 0;
            case 'i':
                inputFile = fopen(optarg, "r");
                if (!validFile(inputFile, 'r', optarg)) {
                    return 0;
                }
                break;
            case 'o':
                outputFile = fopen(optarg, "w");
                if (!validFile(outputFile, 'w', optarg)) {
                    return 0;
                }
                break;
            default:
                printf("Opci n inv lida. Para ver m s informaci n ingrese -h.
        }
    }

    if (inputFile == NULL) {
        printf("Ingrese el stream a procesar (m ximo 300 caracteres): \n");
        scanf("%*c%[^\n]",inputByStd);
        inputFile = fopen(inputFileAux, "w+");
        fputs(inputByStd, inputFile);
        takeStreamFromStdIn = true;
    }

    if (outputFile == NULL) {
        printf("Se mostrar  el resultado en pantalla. \n");
        outputFile = fopen(outputFileAux, "w+");
        showResultsInStdOut = true;
    }

    processInput(inputFile, outputFile, showResultsInStdOut);

    // Borramos los archivos auxiliares utilizados
    if (takeStreamFromStdIn) remove(inputFileAux);
    if (showResultsInStdOut) remove(outputFileAux);

    return 0;
}
```

3. Este es el Título de Otra Sección

Texto de la otra sección. En la figura 1 se muestra un ejemplo de cómo presentar las ilustraciones del informe.

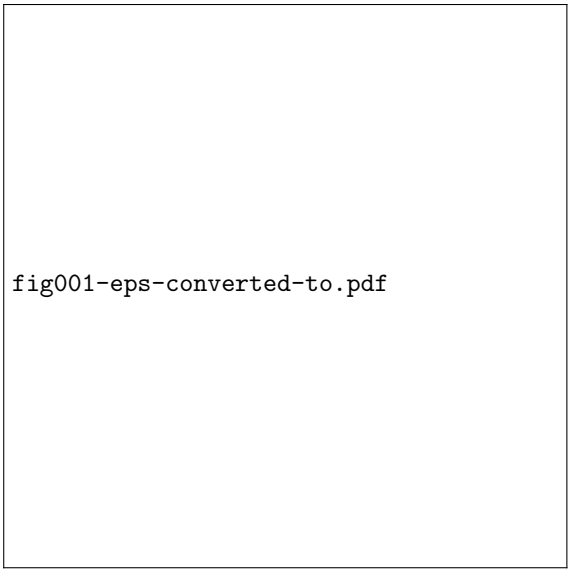

Figura 1: Facultad de Ingeniería – Universidad de Buenos Aires.

3.1. Este es el Título de Otra Subsección

Texto de la otra subsección...

4. Conclusiones

Se presentó un modelo para que los alumnos puedan tomar como referencia en la redacción de sus informes de trabajos prácticos.

Referencias

[1] Intel Technology & Research, "Hyper-Threading Technology," 2006, http://www.intel.com/technology/hyperthread/.

[2] J. L. Hennessy and D. A. Patterson, "Computer Architecture. A Quantitative Approach," 3ra Edición, Morgan Kaufmann Publishers, 2000.

[3] J. Larus and T. Ball, "Rewriting Executable Files to Mesure Program Behavior," Tech. Report 1083, Univ. of Wisconsin, 1992.