"Ss. Cyril and Methodius" University in Skopje
FACULTY OF COMPUTER SCIENCE AND ENGINEERING

Structured programming

Exercises 11

Version 1.0, 15 December, 2016

Table of Contents

1. Files

1.1. Remainders from lectures

- Processing files includes writing, reading and changing contents of files to some standard media as disk.
- Processing files in C is done using the struct `FILE` defined in `stdio.h`.
- To start processing the file first it must be open using the function `fopen` that returns pointer to the struct `FILE*`.

1.1.1. Opening file for reading/writing

Function for opening file:

```
FILE* fopen(const char* file_path, const char* mode);
```

`file_path` - full path of the file we want to open

`mode` - opening mode

1.1.2. Modes of opening files

Mode	Meaning
`r`	Opens existing file only for reading
`w`	Opens existing file for writing (the file must exist)
`a`	Opens existing file for appending (the file must exist)
`r+`	Opens file for reading and writing at the beginning of the file
`w+`	Opens file for reading and writing (deletes the contents of the file)
`a+`	Opens file for reading and writing (appends at the end of the file if it exists)

Example of opening file

```
FILE* fp = fopen("test.txt", "r");
```

- Opens the file "test.txt" in reading mode.
- To open in binary mode `b` should be appended at the mode of opening ex `rb`.

1.1.3. Closing file

Function for closing file:

```
int fclose(FILE* fp);
```

`fp` - pointer to FILE we want to close

Example of closing file

```
fclose(fp);
```

- After the end of processing, the file should be closed using the function `fclose()`
- Calling this functions closes the file associated with the file pointer `fp` passed as argument

1.1.4. Reading and writing from/to file

Functions for reading from file:

```
int fscanf(FILE* fp, "format_specifier", arguments_list);
```

```
int fgetc(FILE* fp);
```

```
char* fgets(char* str, int num, FILE* fp);
```

Functions for writing to files:

```
int fprintf(FILE* fp, "format_specifier", arguments_list);
```

```
int fputc(char c, FILE* fp);
```

```
int fputs(const char* str, FILE* fp);---
```

1.2. Problem 1

Write a program that for given textual file `text.txt` will find the ratio of vowel/consonants.

Example

If the file `text.txt` has the following contents:

```
Hello, how are you?
I'm OK. How about you?
I'm fine too.
```

then the output should be:

```
Ratio vowels/consonats: 20/18 = 1.11
```

Solution `p11_1_en.c`

```
#include <stdio.h>

int is_letter(char c) {
    return (c >= 'a' && c <= 'z') || (c >= 'A' && c <= 'Z');
}

int is_vowel(char c) {
    c = tolower(c);
    switch (c) {
    case 'a':
    case 'e':
    case 'i':
    case 'o':
    case 'u':
        return 1;
    default:
        return 0;
    }
}

int main() {
    char c;
    int consonants = 0, vowels = 0;
    FILE *dat;
    // Opening file for reading
    if ((dat = fopen("text.txt", "r")) == NULL) {
        printf("The file `text.txt` can not be opened.\n");
        return -1;
    }
    // Reading char by char until EndOfFile (EOF)
    while ((c = fgetc(dat)) != EOF) {
        if (is_letter(c)) {
            if (is_vowel(c))
                vowels++;
            else
                consonants++;
        }
    }
    fclose(dat);
    printf("Ratio vowels/consonants: %d/%d = %5.2f\n", vowels, consonants,
           (float) vowels / consonants);
    return 0;
}
```

1.3. Problem 2

Write a program that each row from given input file `input.txt` will copy in other file `output.txt`, so that for each row from `input.txt` will add one more with the length of that row. Each row can have at most 80 characters.

Example

If the file `input.txt` has the following contents:

```
I'm learning Structured Programming.
When is the second midterm?
I don't know, still it is not published on the web.
```

тогаш по извршувањето на програмата содржината на датотеката `izlezna.txt` треба да биде следнава:

```
36
I'm learning Structured Programming.
27
When is the second midterm?
51
I don't know, still it is not published on the web.
```

Solution `p11_2_en.c`

```c
#include <stdio.h>
#define MAX 81

int main() {
    char row[MAX], *c;
    FILE *input, *output;
    if ((input = fopen("input.txt", "r")) == NULL) {
        printf("The file `%s` can not be opened.\n", "input.txt");
        return -1;
    }
    if ((output = fopen("output.txt", "w")) == NULL) {
        printf("The file `%s` can not be opened.\n", "output.txt");
        return -1;
    }

    while ((fgets(row, MAX, input)) != NULL) {
        int br = strlen(row);
        fprintf(output, "%d\n%s", br, row);
    }
    fclose(input);
    fclose(output);
    return 0;
}
```

1.4. Problem 3

Write a program that will read elements of a matrix written in text file with name `mat1.txt`. In the first line of the file are written the number of rows and columns of the matrix. Each element of the matrix is floating point number written in separate line. The transposed matrix write in a new output file `mat2.txt` using the same format.

Example

If the file `mat1.txt` has the following content:

```
3 4
2.1
3.2
4.3
5.4
1.1
2.2
3.3
4.4
6.0
5.5
3.9
1.8
```

then at the end of the execution of the program, the file `mat2.txt` should have the following content:

```
4 3
2.1
1.1
6.0
3.2
2.2
5.5
4.3
3.3
3.9
5.4
4.4
1.8
```

Solution `p11_3_en.c`

```
#include <stdio.h>
#include <stdlib.h>
#define MAX 100
int main() {
    int i, j, m, n;
    float a[MAX][MAX], b[MAX][MAX];
    FILE *input, *output;
    if ((input = fopen("mat1.txt", "r")) == NULL) {
        printf("The file `mat1.txt` can not be opened!\n");
        exit(1);
    }
    if (!feof(input))
        fscanf(input, "%d %d", &m, &n);

    if ((m > MAX) || (n > MAX)) {
        printf("Very large matrix!");
        return (-1);
    }
    for (i = 0; i < m && !feof(input); i++)
        for (j = 0; j < n && !feof(input); j++)
            fscanf(input, "%f", &a[i][j]);
    fclose(input);
    if (i != m || j != n) {
        printf("Not enough data in the file!");
        return (-1);
    }
    for (i = 0; i < m; i++)
        for (j = 0; j < n; j++)
            b[j][i] = a[i][j];
    if ((output = fopen("mat2.txt", "w")) == NULL) {
        printf("The file `mat2.txt` can not be opened!\n");
        exit(1);
    }
    fprintf(output, "%d %d\n", n, m); /* reverse */

    for (i = 0; i < n; i++)
        for (j = 0; j < m; j++)
            fprintf(output, "%7.2f\n", b[i][j]);
    fclose(output);
    return (0);
}
```

1.5. Problem 4

Given is the file `SP_example.txt`. Write a program that will read the file and will print the number of rows with more than 10 vowels, and the total vowels in the file.

Example

If the file `SP_example.txt` has the following content:

```
Zdravo, kako si?
Eve, dobro sum. A ti?
I jas dobro. Kako se tvoite? Ima li neshto novo?
Dobri se i tie. Si kupiv avtomobil.
```

then the program should print:

```
Total 2 rows have more than 10 vowels.
The file has total 38 vowels.
```

Solution `p11_4_en.c`

```
#include <stdio.h>
#include <stdlib.h>
int is_vowel(char c) {
    return c == 'a' || c == 'e' || c == 'i' || c == 'o' || c == 'u';
}
int main() {
    int row = 0, total = 0;
    FILE *dat; char c;
    if ((dat = fopen("SP_example.txt", "r")) == NULL) {
        printf("The file `SP_example.txt` can not be opened");
        exit(-1);
    }
    int vowels = 0;
    while ((c = fgetc(dat)) != EOF) {
        if(is_vowel(tolower(c))) {
            ++vowels;
            ++total;
        }
        if (c == '\n') {
            if (vowels > 10) {
                row++;
            }
            vowels = 0;
        }
    }
    printf("Total of %d rows has more than 10 vowels\n", row);
    printf("The file has total %d vowels.\n", total);
    return 0;
}
```

1.6. Problem 5

Write a program that for given file `words.txt` will print all the words that have three or more equal letters (some letter occurs three or more times). The comparison of letters is not case sensitive. At the end it should print the count of words that satisfy this condition.

The file contains one word per row. Each word is composed only from letters. The maximal length of a word is 20 chars.

Example

If the file `words.txt` has the follwoing content:

```
banana
jabolko
Obratnoto
binarnata
dekadniot
Kopakabana
```

then the program should print:

```
banana
Obratnoto
binarnata
Kopakabana
Total 4 words.
```

Solution `p11_5_en.c`

```
#include <stdio.h>
#include <ctype.h>
#define SIZE 21

int has_more_than_2eq(char *w) {
    char *c;
    int equal;
    while (*w) {
        c = w + 1;
        equal = 1;
        while (*c) {
            if (tolower(*w) == tolower(*c))
                equal++;
            c++;
        }
        if (equal > 2)
            return 1;
        w++;
    }
    return 0;
}
int main() {
    char word[SIZE];
    FILE *f;
    int words_count = 0;
    if ((f = fopen("words.txt", "r")) == NULL) {
        printf("The file `words.txt` can not be opened.\n");
        return -1;
    }
    while (fgets(word, SIZE, f) != NULL) {
        if (has_more_than_2eq(word)) {
            puts(word);
            words_count++;
        }
    }
    printf("\nTotal %d words.\n", words_count);
    fclose(f);
    return 0;
}
```

1.7. Problem 6

Write a program that will print the count of occurrences of a word composed only from digits (read from SI) in a file named `dat.txt`.

Example

If we read the word

```
123
```

and if the file `dat.txt` has the following content:

```
Zdravo 123, kako si?
Eve 321, dobro sum. A ti?
I jas dobro. Kako se tvoite 123? Ima li neshto novo? 123
Dobri se i tie. Si kupiv avtomobil.
```

then it should print:

```
The word 123 occurs 3 times in the file.
```

Solution `p11_6_en.c`

```c
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
int main() {
    char c;
    int occurrences = 0;
    FILE *file;
    if ((file = fopen("dat.txt", "r")) == NULL) {
        printf("The file `dat.txt` can not be opened!\n");
        exit(-1);
    }
    char word[50];
    printf("Enter word for searching:");
    gets(word);
    int i = 0, counter = 0;
    while ((c = fgetc(file)) != EOF) {
        if (isdigit(c)) {
            if (c != word[i++]) {
                if (counter == strlen(word)) {
                    occurrences++;
                }
                counter = 0;
                i = 0;
            } else {
                counter++;
            }
        } else {
            if (counter == strlen(word)) {
                occurrences++;
            }
            counter = 0;
            i = 0;
        }
    }
    printf("The word `%s` occurs %d times in the file\n", word, occurrences);
    return 0;
}
```

2. Source code of the examples and problems

https://github.com/finki-mk/SP/ Source code ZIP