Tutorial

Programming & Data Structure: CS 11001

Section - 4/D

Department of Computer Science and Engineering

I.I.T. Kharagpur

Spring Semester: 2013 - 2014 (03.04.2014)

Download

Download the file **tut030414.pdf** from Programming & Data Structures ... of

http://cse.iitkgp.ac.in/~goutam

View the file using the command acroread & or xpdf &

File Access

A few C Library Functions

```
FILE *fopen(const char *path, const
char *mode);
int fclose(FILE *stream);
int fscanf(FILE *stream, const char
*format, ...);
int fprintf(FILE *stream, const char
*format, ...);
```

A few C Library Functions

```
void rewind(FILE *stream);
int fseek(FILE *stream, long offset,
int whence);
int fgetc(FILE *stream);
int fputc(int c, FILE *stream);
```

All prototypes are in `stdio.h`.

Open, Access and Close a File

File Access

```
#include <stdio.h>
int main()   // tutXII.ex1.c
{
    FILE *fpI, *fpO ;
    int n;

    fpI = fopen("data", "r");
    fpO = fopen("out", "w");
    fscanf(fpI, "%d", &n);
    fprintf(fpO, "%d\n", n);
    fclose(fpI); fclose(fpO);
    return 0;
}
```

Data File

Consider the data file `data` with the following content.

```
886 2777 6915 7793 8335 5386 492 6649
1421 2362 27 ......
```

Tutorial XII.1

Write a C program to open the file `data` in read-only mode and print all prime numbers present in it to stdout. Also print the count of prime numbers. Finally close the data file. Note that the count of integers present is not known *a priori*. So use `EOF` condition with `fscanf()`.

Tutorial XII.2

Write a C program that opens the file `data` in read-only mode, counts the number of primes present in it. Opens the file outData in the write-only mode, writes the prime count in it. Positions the current file pointer of data at the beginning by the rewind() call, reads one integer at a time from data. If it is prime, then writes it in outData. At the end it closes both the files.

rewind()

```
rewind(fpI);
```

Positions the file pointer at the beginning.

Data File: data3

```
06CS1029   AKASH  8.0 7.0 7.0 8.0
06MA1010   SANCHIT AGARWAL 7.0 6.0 7.0 8.0
.........................
06CH1016   VISWANATH 8.0 8.0 7.0 7.0
06CE1017   MANI KUMAR  7.0 6.0 6.0 7.0
06CS1028   GAURAV SAXENA 7.0 6.0 6.0 7.0
```

Data File: `data3`

There are six fields in each row (record) of the file.

- The first field is 8-character roll number.
- The second field is the name of a student. It may be at most 50 character long.
- Last four fields are the SGPAs of four semesters (of equal credits). These are of type `float`.

Structure student

We may store each student record as a data of the following type:

```
typedef struct studRec {
        char rollNo[9];
        char name[51];
        float sgpa[4];
        float cgpa ;
} student;
```

Tutorial XII.3

Write a C program that opens the file `data3` in read mode. Reads the student records in an *array of structure* of type `student`.
It computes the CGPA of each student and puts it in the `cgpa` fileld.
It prints the *roll number*, *name* and *cgpa* for each student in `stdout` and closes the file `data3`.

Command Line Arguments

The function `main()` can take arguments.
These are known as command line arguments.

Command Line Arguments

```
#include <stdio.h>
#include <stdlib.h>
int main(int c, char *v[]) // tutXII.ex2.c
{
    if(c < 2) printf("No argument\n");
    else {
        int n = atoi(v[1]), fact=1, i=1;
        for(i=1; i<=n; ++i) fact *= i;
        printf("%d! = %d\n", n, fact);
    }
    return 0;
}
```

Tutorial XII.4

Modify the program of tutorial XII - 3 so that the file name will be a command line argument.