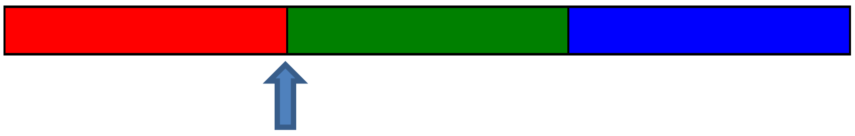
Tutorial - Week 9

Objectives: To practice with
- Text files
- Binary files

1. **What is the difference between a file and a file stream?**
2. **Open file `report.txt` for reading. Use appropriate error recovery in case of failure.**
3. **Write a fragment of a program that counts and displays the number of characters stored in a file. You can assume that the file has been opened successfully.**
4. **Open a binary file `results.dat` for output. If the file with such a name does not exist, it is created. If the file already exists, its content is preserved. Use appropriate error recovery in case of failure.**
5. **Write a fragment of a program, which reads all experiment data stored in a binary file block-by-block, calculates an average value for a block of data and prints it with 2 decimal point precision. All data are stored as type `float`. The block size is `128`. You can assume that the file has been opened successfully.**
6. **A color image file `image.raw` sequentially stores a block of red color samples, a block of green color samples followed by a block of blue color samples. Each block is of size 76800 elements of type unsigned char. Write a fragment of code that opens binary file `image.raw` for reading. Use appropriate error recovery in case of failure. Set the file position indicator to the beginning of the green color block. Use appropriate error recovery in case of failure.**



7. **Define a function `saveRecord()` that stores one structured variable into a binary file. You can assume that the file position indicator has already been set. Use the following structure definition:**

```
typedef struct
{
   char level;
   int  code;
}element;
```

8. **Assume the environment shown, and complete the statements that follow so that they are valid:**

```
#define NAME_LEN 50

#define SIZE 30

typedef struct {

   char name[NAME_LEN];
```

```
    int age;
    double income;
} person_t;
. . .
int num_err[SIZE];
person_t exec;
/* binary files */
FILE *nums_inp, *psn_inp, *psn_outp, *nums_outp;
/* text files */
FILE *nums_txt_inp, *psn_txt_inp, *psn_txt_outp;
nums_inp = fopen("nums.bin", "rb");
nums_txt_inp = fopen("nums.txt", "r");
psn_inp = fopen("persons.bin", "rb");
psn_txt_inp = fopen("persons.txt", "r");
psn_outp = fopen("persout.bin", "wb");
psn_txt_outp = fopen("persout.txt", "w");
nums_outp = fopen("numsout.bin", "wb");
```

a. **`fscanf(psn_txt__________________, "%s", __________________);`**

b. **`fwrite(num_err, __________________, __________________, nums_outp);`**

c. **`fprintf(psn_txt_outp, "%s %d %f\n", __________________, __________________, __________________);`**

9. **Write a void function `make_product_file` that would convert a text file containing product information to a binary file of `product_t` structures. The function's parameters are file pointers to the text input and binary output files.**

10. **Consider a file `empstat.txt` that contains employee records. The data for each employee consist of the employee's name (up to 20 characters), social security number (up to 11 characters), gross pay for the week (`double`), taxes deducted (double), and net pay (`double`) for the week. Each record is a separate text line in file `empstat.txt`. Write a program that will create a text file `report.txt` with the heading line:**

    NAME SOC.SEC.NUM GROSS TAXES NET

    followed by two blank lines and the pertinent information under each column heading. The program should also produce a binary file version of `empstat.txt` named `empstat.bin`.