EXAMPLES of BINARY file operations using the library <stdio.h>

BINARY files composed of sequence of numbers:

BIN-1: Store the sequence of numbers (from keyboard) into the file (**fwrite**)

BIN-2: Read and display the content of a binary file (**fread**)

BIN-3: Read the portion from inside the binary file (**fseek**, **fread**)

BIN-4: Append some numbers at the end of the binary file (**a -** ***append***)

BIN-5: Update / overwrite the portion of binary file
(**fseek**, **ftell**, **fwrite**)

BIN-6: Filtering – copying numbers matching the specified criteria using a loop:

```
while( fread(...)==1 )
   if( ... )
      fwrite(...);
```

Example BIN-1: Writing a sequence of numbers to a binary file

Input (text) data: **stdin**

```
1
2500
-18
1000000
```

Output file: **numbers.dat** **(4 × 8 bytes)**

```
0x00 0x00 0x00 0x00 0x00 0x00 0xF0 0x3F
0x00 0x00 0x00 0x00 0x00 0x88 0xA3 0x40
0x00 0x00 0x00 0x00 0x00 0x00 0x32 0xC0
0x00 0x00 0x00 0x00 0x80 0x84 0x2E 0x41
```

The contents of the "number.dat" file viewed with a text editor
(for example, with the system's "Notepad") looks as follows:

Output file: **numbers.dat** **-** displayed as a text **(32 = 4 × 8 characters)**

```
        đ?      Ł@      2Ŕ      €„.A
```

```
// The program iteratively reads consecutive numbers from the keyboard into the buffer
// and immediately copies the contents of the variable buffer (8 bytes) to the binary file

#include <stdio.h>
int main()
{  FILE* my_file = fopen( "numbers.dat", "wb" );
   if( my_file )
      {
         // retrieving the number of read-in data (in the example above it is equal 4)
         int  how_many_numbers;
         printf("Enter how many numbers you want to save in the file: ");
         scanf("%d", &how_many_numbers);

         double buffer;
         for(int i=0; i< how_many_numbers; i++)
            {
               printf("\nEnter the next number: ");
               scanf("%lf",&buffer);
               fwrite(&buffer, sizeof(double), 1, my_file );
            }
         fclose(my_file);
      }
   return 0;
}
```

Binary write: **fwrite(*data_pointer*, *block_size*, *amount_of_blocks*, *file*)**

Example BIN-2: Reads and displays the entire contents of a binary file

```
// The program iteratively gets (loads) from the file "numbers.dat",
// subsequent 8-byte portions of data, to the auxiliary variable buffer of type double
// and then displays them on the screen (send them to the text file stdout)

#include <stdio.h>
int main()
{
   FILE* my_file = fopen( "numbers.dat", "rb" );
   if( !my_file )
      {
         printf("File open error ");
         getchar();
      }
   else
      {
         double  buffer;
         int counter=0;

         // iterative loop to get successive portions of sizeof (double)
         while( fread(&buffer, sizeof(double), 1, my_file)==1 )
            {

               // displaying the loaded numbers on the console screen
               printf("%f\n", buffer );

               // Here, further processing of the loaded number can take place . . .
               //  for example, summation:       sum = sum+buffer;
               // or searching the minimum:
               //      if( buffer<minimum) minimum=buffer;

               // In this example, it is just the count of the number of loaded numbers:
               counter ++;
            }
         fclose(my_file);

         printf("\n End of reading. Number of loaded data =  %d",counter);
      }
   return 0;
}
```

Binary read: **fread(** ***data_pointer*****,** ***block_size*****,** ***amount_of_blocks*****,** ***file*** **)**
The **fread** and **fwrite** functions return the number of correctly processed blocks, this return value may be used to control read / write errors

Example BIN-3: Read the pointed number from inside of a binary file

```
// The program moves the file pointer ("read head")
// to the indicated place (distance in bytes from the beginning of the file)
// and then takes 8-bytes into a variable buffer of type double
// and displays its content on the console screen

#include <stdio.h>
int main()
{  FILE* my_file = fopen( "numbers.dat", "rb" );
   if( !my_file )
      printf("File open error ");
   else
      {  long position_of_item = 30;   // ← can by any position inside the file

         // file pointer shift command
         fseek(my_file, position_of_item * sizeof(double), SEEK_SET);

         double buffer_for_number;

         // reading 8-bytes, from the indicated file location, into the buffer_for_number
         if( fread(&buffer_for_number, sizeof(double), 1, my_file)==1 )
            printf("Loaded number = %f\n",buffer_for_number);
         else
            printf("Some error while reading the number from the file ");
         fclose(my_file);
      }
   printf("\nEnd of reading. Press ENTER ");
   fflush(stdin);    getchar();
   return 0;
}
```

Example BIN-4: Append three numbers (22,44,66) to the end of the file

```
#include <stdio.h>
int main()  {
   FILE* my_file = fopen( "numbers.dat", "ab" );        ← "a" like "append"
   if( !my_file )
      printf("File open error ");
   else
      {  double number;
         for(int i=1; i<=3; i++)
            {
               number = i*22;
               fwrite(&number, sizeof(double), 1, my_file);
            }
         fclose(my_file);
      }
   return 0;
}
```

Example BIN-5: Replace/overwrite the value of a number in a binary file

```
// The program moves the file pointer to the specified position,  in relation to the beginning
// and then (over)writes there 8-bytes from the variable new_value of type double

#include <stdio.h>
int main()
{  FILE* my_file = fopen( "numbers.dat", "r+b" );  ← "+" means „modification"
   if(my_file)
    {
      // reading the file size (and calculating the number of all double numbers in it )
      fseek(my_file,0,SEEK_END);
      long  file_length_in_bytes = ftell(my_file);
      long  all_numbers = file_length_in_bytes / sizeof(double);
      printf("This file contains %ld numbers <double>\n\n", all_numbers);

      // determining (loading) the position of the modified number
      long  overwritten_position;
      printf("Enter the position of the number to be overwritten: ");
      scanf("%ld", &overwritten_position);

      // checking if the given item is correct (is it inside the file?)
      if(overwritten_position >=0 && overwritten_position<all_numbers)
        {  double new_value;
           printf("\nEnter a new value for the number: ");
           scanf("%lf",&new_value);

           // moving the file pointer and writing the new value to the file
           fseek(my_file, overwritten_position*sizeof(double), SEEK_SET);
           fwrite(&newvalue, sizeof(double), 1, my_file);
        }
      fclose(plik);
    }
   return 0;
}
```

Example BIN-6: Filtering - copying numbers according to a given criterion

```
#include <stdio.h>  // copying only positive numbers to the second binary file
int main()
{  double number;
   FILE* input_file = fopen( "numbers.dat", "rb" );
   FILE* file_positive_only = fopen("only_positive_numbers.dat", "wb");
   if( input_file!=NULL && file_positive_only!=NULL )
      while( fread(&number, sizeof(double), 1, input_file)==1 )
         if( number>0 )
            fwrite(&number, sizeof(double), 1, file_positive_only);
   fclose(input_file);
   fclose(file_positive_only);
   return 0;
}
```