Fall 2019
Programming 1
#8
Files

Files

There are two types of files

Text Files: Used to store data as text.

Binary Files: Used to store data as bytes

Steps For Using Files		
1.Declare variables of type FILE to represent the files		
FILE *f1; → f1 is a pointer		
2. Open the files for reading / writing / appending.		
	Text Files	
Reading	f1 = fopen("data.txt" , "r");	
Writing	f1 = fopen("data.txt" , “w");	
Appending	f1 = fopen("data.txt" , “a");	
3. Read/write from/to the files.		
	Text Files	**Binary Files**
Read	fscanf	fread
Write	fprintf	fwrite
4. Close the files after processing the data		
fclose(f1); → f1 declared in 1st step		

Modes when opening file

r	opens a text file in reading mode
w	Opens or create a text file in writing mode.
a	opens a text file in append mode
r+	opens a text file in both reading and writing mode (error if file not found)
w+	opens a text file in both reading and writing mode (old data is lost)
a+	opens a text file in both reading and writing mode

Examples

Ex.1 Writing an integer and character array to file

```
int main( )
{
FILE *f1;  //Step 1
int x=5;
char str[] = “Hello”;
f1 = fopen(“abc.txt”, “w”); //Step 2
fprintf( f1, “%d %s”, x, str); //Step 3
fclose(f1); //Step 4
return 0;
}
```

Ex.2 Reading an integer and character array from file

```
void main( )
{
FILE *f1;  //Step 1
int x;
char str[100];
f1 = fopen(“abc.txt”, “r”); //Step 2
fscanf( f1, “%d %s”, &x, str); //Step
3
fclose(f1); //Step 4
}
```

<u>Ex.3</u> Copying file c:\\abc.txt to c:\\abc_copy.txt

```
int  main()
{
FILE *f1, *f2;
char ch;
f1 = fopen(“c:\\abc.txt”, “rt”);
f2 = fopen(“c:\\abc_copy.txt”, “wt”);
while(!feof(f1)) // while file doesn’t end
{
fscanf( f1, “%c”, &ch);
fprintf( f2, “%c”, ch);
}
fclose(f1);
fclose(f2);
return 0;
}
```

<u>Ex.4</u> Define a structure called student has Name, ID ,GPA then write a program to read data of 10 students and then write them in text file with path c:\\Database.txt

```
struct Student
{
   char name[25];
   int ID;
   float GPA;
};
int  main()
{
      struct Student S[10];
      FILE *f;
      int i;
      for( i = 0; i<10; i++)
      {
       scanf("%s", S[i].Name);
       scanf("%d", &S[i].ID);
       scanf("%f", &S[i].GPA);
      }
 f = fopen("c:\\DataBase.txt", "w");
for( i = 0; i <10; i++)
 {
          fprintf(f, "%s", S[i].Name);
          fprintf(f, "%d", S[i].ID);
          fprintf(f, "%f", S[i].GPA);
}
fclose(f);
return 0;}
```

Ex.5

Suppose you have a text file, called **“student_file.txt “**, in which each record consists of student identification number , student name, and final average score. In each record, data are separated by a blank character.

Develop a modular C program that reads this file into arrays; determines the letter grade for each student ; and prints on the monitor screen a table of student number , student name, final average score, and letter grade. It should also print the maximum, minimum, and average of final averages. The output should look like this :

```
STUDENT ID      STUDENT NAME        FINAL AVG     GRADE
    1000            Ali   Hany          87            B
    2000            Ghasan Maher        99            A
    ......          ..................  ...          ...
    ......          ..................   ...          ...
Minimum of final averages = 87
Maximum of final averages = 99
```

In Determining the final letter grades, use the following rules :

Final Average	Letter Grade
90-100	A
80 - 89	B
70 - 79	C
60 - 69	D

Solution

```
#include <stdio.h>

typedef struct {
int id;
char name[30];
float score;
} Student_t;

char gradeFromScore(float score)
{
if (score >= 90)
return 'A';
if (score >= 80)
return 'B';
if (score >= 70)
return 'C';
if (score >= 60)
return 'D';
return 'F';
}
```

```
int main()
{
int i;
float maxscore, minscore;
Student_t students[100];
int n = 0;
FILE* f;
f = fopen("students_file.txt", "r");
if (f != NULL)
{

// reading the file
while (!feof(f))
{
fscanf(f, "%d %s %f", &students[n].id,
students[n].name, &students[n].score);
fscanf(f, "\n");
n++;
}

fclose(f);
```

```
// calculation max and min score
maxscore = minscore =
students[0].score;
for (i=1; i<n; i++)
{
if (students[i].score > maxscore)
maxscore = students[i].score;
if (students[i].score < minscore)
minscore = students[i].score;
}

// printing all students
printf("All
Students:\n===============\n");
printf("STUDENT ID\tSTUDENT
NAME\tFINAL AVG\tGRADE\n");
for (i=0; i<n; i++)
{
printf("%d \t\t %s \t\t %.2f \t\t
%c\n", students[i].id,
students[i].name,
students[i].score,
gradeFromScore(students[i].score));
}
```

```
printf("Minimum of final averages =
%.2f\n", minscore);
printf("Maximum of final averages =
%.2f\n", maxscore);
}
else
printf("Error opening the fie!\n");
return 0;
}
```

Ex.6 Suppose you have a file called “clients.txt” that might be used in an accounts receivable system to help keep track of the amounts owned by a company’s credit clients. For each client, his record consists of the account number, the client’s name, and the client’s balance.

- Write a program that reads records from the file “clients.txt” and prints the contents of the records.

- Write a program that allows a credit manager to obtain lists of those customers with zero balance, credit balances, and debit balances.

```
#include <stdio.h>

typedef struct {
char name[30];
char number[15];
double balance;
} Client_t;
```

```
int main()
{
int i;
Client_t clients[100];
int n = 0;
FILE* f;

f = fopen("clients.txt", "r");
if (f != NULL)
{
// reading from file
       while (!feof(f))
        {
          fscanf(f, "%s %s %lf", clients[n].name,
clients[n].number, &clients[n].balance);
           fscanf(f, "\n");
             n++;
         }
     fclose(f);









// printing all the clients
```

```
 printf("All clients:\n==============\n");
for (i=0; i<n; i++)
 {
   printf("Client: %s \t %s \t %.2lf\n", clients[i].name,
clients[i].number,clients[i].balance);
}


// printing zero balance
printf("\nZero balance
clients:\n========================\n");

for (i=0; i<n; i++)
{
    if (clients[i].balance == 0)
          printf("Client: %s \t %s \t %.2lf\n",
clients[i].name, clients[i].number, clients[i].balance);
}




// printing credit balance
printf("\nCredit balance
clients:\n========================\n");
for (i=0; i<n; i++)
{
if (clients[i].balance > 0)
printf("Client: %s \t %s \t %.2lf\n", clients[i].name,
clients[i].number,
clients[i].balance);
}


// printing debit balance
printf("Debit balance
clients:\n=======================\n");
for (i=0; i<n; i++)
{
if (clients[i].balance < 0)
printf("Client: %s \t %s \t %.2f\n", clients[i].name,
clients[i].number,
clients[i].balance);
}
}
else
printf("Error opening the fie!\n");
return 0;
}
```

Binary Files

fwrite, fread

- Write and read data from an array “block of data to a file”
- fwrite (ptr , size, number , FILE *stream)

ptr: address of starting value that will be written
size: This is the size in bytes of each element to be written.
number : This is the number of elements,
FILE* stream : This is the pointer to a FILE object

Binary Files Examples

Ex.1 Writing an integer and character array to file

```
int main( )
{
FILE *f1;
int x=5;
char str[] = “Hello”;
f1 = fopen(“abc.txt”, “w”);
fwrite(&x, sizeof(int), 1, f1);
fwrite(str, sizeof(char), strlen(str), f1);
fclose(f1);
return 0;
}
```

Note→**fwrite** and **fread** parameters
1st param →address of element to be written
2nd param→ size of each element to be written in bytes
3rd param →number of elements to be written
4th param→ the pointer to a FILE.

<u>Ex.2</u> Reading an integer and character array from file

```
int main( )
{
FILE *f1;
float x;
char str[100];
f1 = fopen("c:\\abc.bin", "r");
fread(&x, sizeof(float), 1, f1);
fread(str, sizeof(char), 5, f1);
fclose(f1);
return 0;
}
```

Ex.3 **Copying file c:\\abc.bin to c:\\abc_copy.bin**

```
void main()
        {
            FILE *f1, *f2;
            char c;
            f1 = fopen(“c:\\abc.bin”, “r”);
            f2 = fopen(“c:\\abc_copy.bin”, “w”);
            while(!feof(f1))
                {
                    fread(&c, sizeof(char), 1, f1);
                    fwrite(&c, sizeof(char), 1, f2);
                }
            fclose(f1);
            fclose(f2);
        }
```

Ex.4 Define a structure called student has Name ,ID ,GPA then write a program to read data of 10 students and then write them in binary file

```
struct Student
{
      char name[25];
      int ID;
         float GPA;
};
int  main()
{
             struct Student S[10];
             FILE *f;
             int i;
             for( i = 0; i<10; i++)
             {
                    scanf("%s", S[i].Name);
                        scanf("%d", &S[i].ID);
                    scanf("%f", &S[i].GPA);
               }
              f = fopen("c:\\DataBase.bin", "w");
for(i = 0; i< 10; i++)
        fwrite(&S[i], sizeof(Student), 1, f);
OR
        fwrite(S, sizeof(Student), 10, f); → write 10
students once without for loop
fclose(f);
return 0;
}
```

C provides several different functions for reading and writing

getc(File *ptr): read character from file
putc(File *ptr): write character into file

Example

```
Void main()
{
FILE *f1;
char c;
f1= fopen (“INPUT.txt”, “w”);

while((c=getchar()) != EOF)
{
/*get char from keyboard until CTL-Z (end of file)*/
putc(c,f1);
}

 /*write a character to INPUT */
fclose(f1); /* close INPUT */
f1=fopen(“INPUT.txt”, “r”); /* reopen file but for reading
*/
while((c=getc(f1))!=EOF) /*read character from file*/
{
printf(“%c”, c);
}
fclose(f1);
}
```

fputs (string, FILE *ptr)

Writes a string to the specified file up to but not including the null character.

Example

```
int main () {
FILE *fp;
fp = fopen("file.txt", "w+");
fputs("This is c programming.", fp);
fputs("This is a system programming language.", fp);
fclose(fp);
return(0);
}
```

fgets (string, number of characters, file pointer)

```
int main () {
FILE *fp;
char str [60];
fp = fopen("file.txt" , "r");
fgets ( str, 60,fp);
printf( %s, str);
fclose(fp);
return(0);
}
```

How to jump to a given position
fseek (file-pointer, offset, position);

Position:

•0 (SEEK_SET) (beginning of the file)
•1 (SEEK_CUR) (current position of the pointer in the file)
•2 (SEEK_END) (end of the file)

•**offset**: number of locations to move from position

•Example:
•fseek (fp,m,0); /* move to (m+1)th byte in file */
•fseek (fp,-m, 1); /* move back by m bytes from current position */

•fseek (fp, -10, 2); /* move back 10 bytes from end of file*/

•ftell(fp) returns current byte position in file

•rewind(fp) resets position to start of file

Example

```
int main () {
 FILE *fp;
 fp = fopen("file.txt","w+");
 fputs(“ I love Cairo ", fp);
 fclose(fp);
 return(0);
}
```

Use fseek to change the file contents to “ I love Alexandria”

```
int main () {
 FILE *fp;
 fp = fopen("file.txt","w+");
 fputs(“ I love Cairo ", fp);
 fseek( fp, 7, SEEK_SET );
 fputs(" Alexandria", fp);
 fclose(fp);

 return(0);
}
```