# CL118 Programming Fundamentals

# Lab 09
Array structures and File Processing in C

Instructors: Ms. Maham Mobin Sheikh
Ms. Atiya jokhio

**NATIONAL UNIVERSITY OF COMPUTER AND EMERGING SCIENCES**

# LAB 09

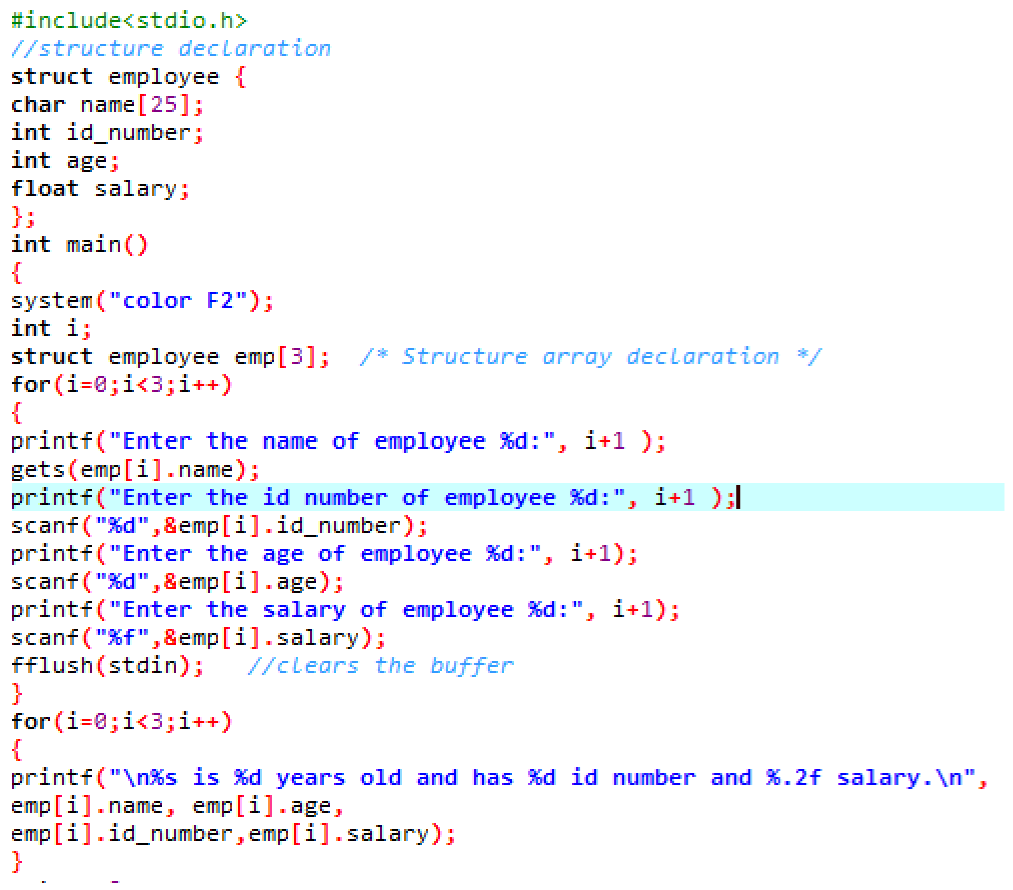
maham.mobin@nu.edu.pk | atiya.jokhio@nu.edu.pk

## Learning Objectives

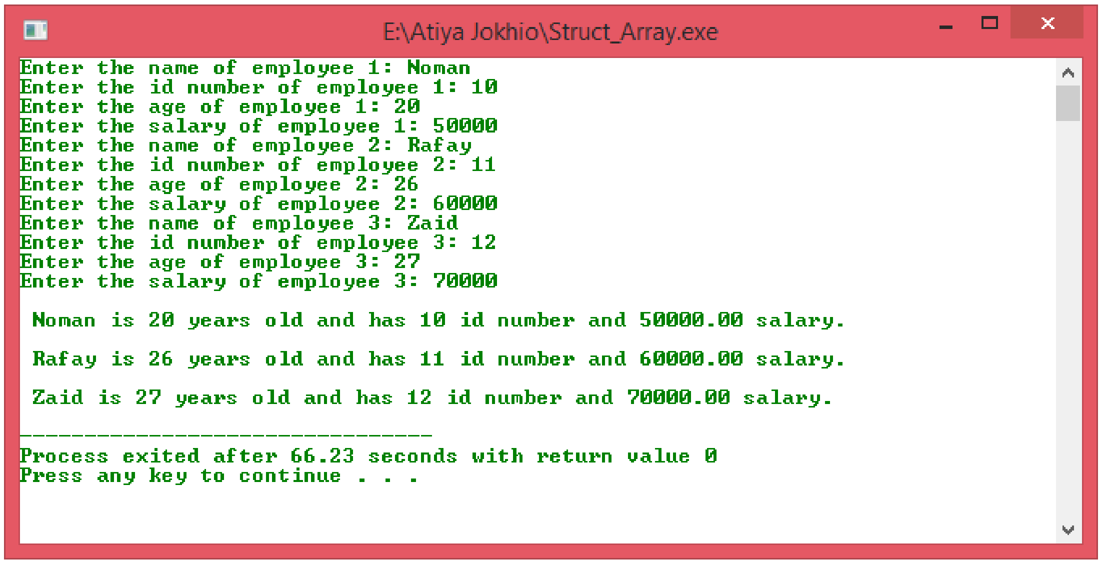
- Working with structures
- File Processing

### Array of Structures:

An array of structures in C can be defined as the collection of multiple structures variables where each variable contains information about different entities. The array of structures in C are used to store information about multiple entities of different data types.



```
#include<stdio.h>
//structure declaration
struct employee {
char name[25];
int id_number;
int age;
float salary;
};
int main()
{
system("color F2");
int i;
struct employee emp[3];  /* Structure array declaration */
for(i=0;i<3;i++)
{
printf("Enter the name of employee %d:", i+1 );
gets(emp[i].name);
printf("Enter the id number of employee %d:", i+1 );
scanf("%d",&emp[i].id_number);
printf("Enter the age of employee %d:", i+1);
scanf("%d",&emp[i].age);
printf("Enter the salary of employee %d:", i+1);
scanf("%f",&emp[i].salary);
fflush(stdin);   //clears the buffer
}
for(i=0;i<3;i++)
{
printf("\n%s is %d years old and has %d id number and %.2f salary.\n",
emp[i].name, emp[i].age,
emp[i].id_number,emp[i].salary);
}
```


E:\Atiya Jokhio\Struct_Array.exe

```
Enter the name of employee 1: Noman
Enter the id number of employee 1: 10
Enter the age of employee 1: 20
Enter the salary of employee 1: 50000
Enter the name of employee 2: Rafay
Enter the id number of employee 2: 11
Enter the age of employee 2: 26
Enter the salary of employee 2: 60000
Enter the name of employee 3: Zaid
Enter the id number of employee 3: 12
Enter the age of employee 3: 27
Enter the salary of employee 3: 70000

 Noman is 20 years old and has 10 id number and 50000.00 salary.

 Rafay is 26 years old and has 11 id number and 60000.00 salary.

 Zaid is 27 years old and has 12 id number and 70000.00 salary.

--------------------------------
Process exited after 66.23 seconds with return value 0
Press any key to continue . . .
```

## Structures with functions

Structure variables are passed by value by default. To pass an array by value, create a structure with the array as a member.

```
#include<stdio.h>
#include<string.h>
int i;
//structure declaration
struct employee {
char name[25];
int id_number;
int age;
float salary;
};
void Input(struct employee emp[], int count)
{
for(i=0;i<count;i++)
{
printf("Enter the name of employee: " );
scanf("%s", emp[i].name);
printf("Enter the id number of employee: " );
scanf("%d",&emp[i].id_number);
printf("Enter the age of employee: " );
scanf("%d",&emp[i].age);
printf("Enter the salary of employee: " );
scanf("%f",&emp[i].salary);
}
Display(emp,count);
}
void Display(struct employee emp[],int count)
{
for(i=0;i<count;i++)
{
printf("\n%s is %d years old and has %d id number and %.2f salary.\n",emp[i].name, emp[i].age,
emp[i].id_number,emp[i].salary);
}
}
int main()
{
struct employee emp[3];  /* Structure array declaration */
Input(emp,3);
return 0;
}
```

When a program is terminated, the entire data is lost. Storing in a file will preserve your data even if the program terminates. If you have to enter a large number of data, it will take a lot of time to enter them all. However, if you have a file containing all the data, you can easily access the contents of the file using few commands in C. You can easily move your data from one computer to another without any changes.

## File Operations

There are different operations that can be carried out on a file. These are:

1. Creation of a new file
2. Opening an existing file
3. Reading from a file
4. Writing to a file
5. Closing a file

Let us now write a program to read a file and display its contents on the screen. We will first list the program and show what it does, and then dissect it line by line.
For using FILE in our program, we need not go through the members of the FILE structure. We can just use FILE pointer to create and use FILE for different operations.

**Prototype**

FILE* filePointerName;

**Example:**

Code:

```
FILE *fp;
FILE *inFile;
FILE *outFIle
```

- **fopen()**

**Prototype**

fopen(“fileName”,”mode”);

**Example:**

Code:

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

- **fclose()**

**Prototype**

```
fclose(filePointer);
```

**Example:**

Code:

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

## File Access Mode

While opening a file, we have to specify the mode for opening the file. Here is a list of different modes for opening any file and usage of these mode.

r - **Read mode** - Read from a specified file which already exists. If the file doesn't exist or is not found, this operaiton fails.

r+ - **Read mode(+ some others)** - Read from and write into an existing file. The file must exist before the operation.

w - **Write mode** - Create new file with specified name. If the file exists, contents of the file is destroyed.

w+ - **Write mode(+ some others)** - Write into an file. It also permits to read the data.

a - **Append mode** - It appends content to the specified file. If the file doesn't exist, it is created first, and then contents are written into it.

a+ - **Append mode(+some others)** - Open a file for reading and appending.

**Example:**

Here is an example of opening a file in write mode. First, a file pointer fp is declared. Then fp is used to open a file named "test.txt" in 'w' mode, which stands for write mode. If fp is null, it means that file could not be opened due to any error.

Code:

```
int main()
{
   FILE *fp;
   fp = fopen("test.txt","w");
   if(fp==NULL)
   {
      printf("Error in openinng file");
   }
   else
```

```
    printf("File has been created successfully");
}
```

## fgetc() function

fgetc( )reads the character from the current pointer position, advances the pointer position so that it now points to the next character, and returns the character that is read, which we collected in the variable ch. Note that once the file has been opened, we no longer refer to the file by its name, but through the file pointer fp.

## Closing the File

1. When we have finished reading from the file, we need to close it.
2. This is done using the function fclose( )through the statement,

   **fclose ( fp ) ;**

## Simple program

```
//reading the content of a file.
#include<stdio.h>
int main( )
{
FILE *fp;
char ch;
fp = fopen ( "abc.txt", "r" );
while( 1 )
{
ch = fgetc ( fp );
if ( ch == EOF )
break;
printf ( "%c", ch );
}
fclose ( fp ); return 0;
}
```

## fputc() Function

It is a function called fputc( )which writes characters to a file.
As a practical use of these character I/O functions we can copy the contents of one file into another, as demonstrated in the program on the next slide

## Program: Copying contents of a File to another file

```
#include<stdio.h>
int main( )
{
FILE *fs,*ft;// fs=> source file and ft => target file
char ch;

fs = fopen("abc.txt", "r");
        if ( fs == NULL )
        {
        puts ("cannot open file");
        }
ft = fopen ("xyz.txt", "w");
        if (ft == NULL)
        {
        puts("cannot open file");
        }
while( 1 )
{
ch = fgetc ( fs );
if ( ch == EOF )
break;
else
fputc(ch,ft);
}
fclose(fs);
fclose(ft);
return 0;
}
```

## Strings in Files

Reading or writing strings of characters from and to files is as easy as reading and writing individual characters using fgets () and fputs()

## Example Program

```
/* Receives strings from keyboard and writes them to file */

#include <stdio.h>
main( )
{
FILE *fp ;
char s[80] ;
fp = fopen ( "POEM.TXT", "w" ) ;
if ( fp == NULL )
{
  puts ( "Cannot open file" ) ;
  exit(1) ;
  }
  printf ( "\nEnter a few lines of text:\n" ) ;
  while ( strlen ( gets ( s ) ) > 0 )
  {
  fputs ( s, fp ) ;
  fputs ( "\n", fp ) ;
  }
  fclose ( fp ) ;
}
```

**//Reading strings from the file and display on screen**

```
#include<stdio.h>
#include<string.h>
int main( )
{
FILE *fp;
char ch[10];
fp = fopen("xyy.txt", "r");
            if ( fp == NULL )
            {
            puts ("cannot open file");
            }
printf("The lines in files are : \n");
while(fgets(ch,10,fp)!=NULL)

printf("%s",ch);

fclose(fp);
}
```

## Record I/O in Files

So far we have dealt with reading and writing only characters and strings. What if we want to read or write numbers from/to file? Furthermore, what if we desire to read/write a combination of characters, strings and numbers? For this first we would organize this dissimilar data together in a structure and then use fprintf( ) and fscanf( ) library functions to read/write data from/to file.

Following program illustrates the use of structures for writing records of employees.

```
#include <stdio.h>
#include <stdlib.h>
int main( )
{
FILE *fp ;
char another = 'Y' ;
struct emp
{
char name[40] ;
int age ;
float bs ;
} ;
struct emp e ;
fp = fopen ( "EMPLOYEE.DAT", "w" ) ;
if ( fp == NULL )
{
puts ( "Cannot open file" ) ;
exit(1);
}
while ( another == 'Y' )
{
printf ( "\nEnter name, age and basic salary: " ) ;
scanf ( "%s %d %f", e.name, &e.age, &e.bs ) ;
fprintf ( fp, "%s %d %f\n", e.name, e.age, e.bs ) ;
printf ( "Add anotherrecord (Y/N) " ) ;
fflush ( stdin ) ;
another = getche( ) ;
}
fclose ( fp ) ; }
```

## LAB ACTIVITY

**Task # 1:**
Create a struct Rectangle with attributes length and width. Provide functions that calculate the perimeter and the area of the rectangle. The check() function should verify that length and width are each numbers larger than 0.0 and less than 20.0.

**Task # 2:**

Write a program to read a file and display contents with its line numbers.

**Task # 3:**

Write a program to copy one file to another. While doing so replace all lowercase characters to their equivalent uppercase characters.

**Task # 4:**

Create an employee management system using structures in C that allows user to add, display, and delete employees. It should store and display five data fields about each employee, which are:

- Name
- Date of Birth
- Employee ID
- Department
- Salary

The program should display a menu with the following options:

1.Add an Employee

2.Delete an Employee

3.Display All Employees

4.Exit