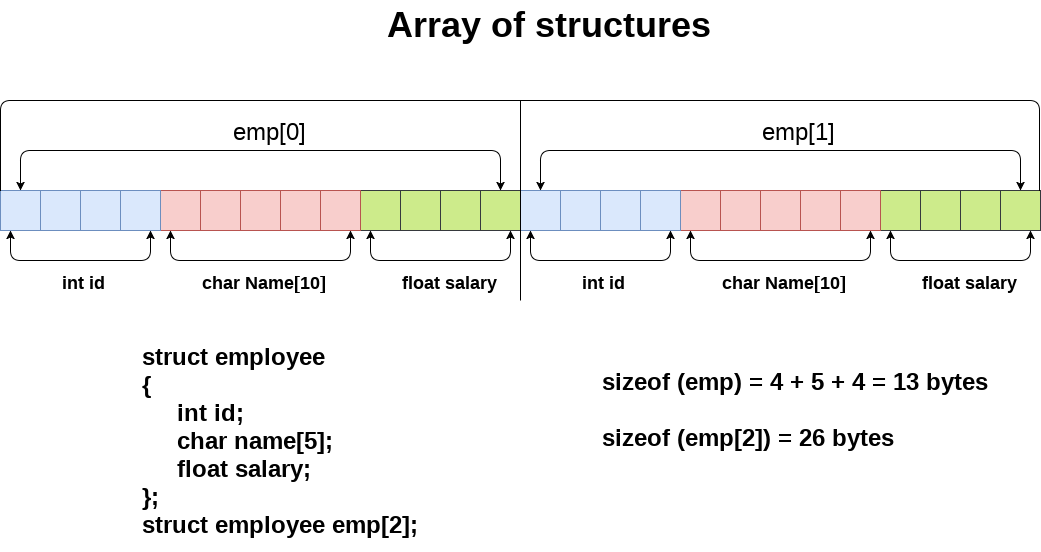
## 

|  |  |
| --- | --- |
| **CS118 Programming Fundamentals** | **LAB 09** Array of Structure, Filing |
| **NATIONAL UNIVERSITY OF COMPUTER AND EMERGING SCIENCES** | |

## **Array of Structures in C**

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. The array of structures is also known as the collection of structures.

****

Let's see an example of an array of structures that stores information of 10 students and prints it.

**Example**

|  |
| --- |
| 1. #include<stdio.h> 2. #include <string.h> 3. **struct** student{ 4. **int** rollno; 5. **char** name[10]; 6. }; 7. **int** main(){ 8. **int** i; 9. **struct** student st[10];   printf("Enter Records of 10 students");  **for**(i=0;i<10;i++){  printf("\nEnter Rollno:");  scanf("%d",&st[i].rollno);  printf("\nEnter Name:");  scanf("%s",&st[i].name);  }  printf("\nStudent Information List:");  **for**(i=0;i<5;i++){  printf("\nRollno:%d, Name:%s",st[i].rollno,st[i].name);  }  **return** 0;  } |

**Filing**

**Why files are needed?**

* 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 a few commands in C.

## **Types of Files**

1. Text files
2. Binary files

## **Text files**

Text files are the normal **.txt** files. You can easily create text files using any simple text editors such as Notepad. When you open those files, you'll see all the contents within the file as plain text. You can easily edit or delete the contents.

**2. Binary files**

Binary files are mostly the **.bin** files in your computer. Instead of storing data in plain text, they store it in the binary form (0's and 1's). They can hold a higher amount of data, are not readable easily, and provides better security than text files.

## **File Operations**

## In C, you can perform four major operations on files.

1. Creation of a new file
2. Opening an existing file
3. Reading from a file
4. Writing to a file
5. Moving to a specific location in a file.
6. Closing a file

## **Working with files**

When working with files, you need to declare a pointer of type file. This declaration is needed for communication between the file and the program.

1. FILE \*fptr;

## **Opening a file - for creation and edit**

Opening a file is performed using the fopen() function defined in the stdio.h header file.

**syntax for opening :**

ptr = fopen(“file\_name”,”mode”);

**For example**

fopen("D:\\K19\_XXXX\\C\_program.txt","w");

Let's suppose the file C\_program.txt doesn't exist in the location D:\\ K19\_XXXX. The first function creates a new file named C\_program.txt and opens it for writing as per the mode **'w'.** The writing mode allows you to create and edit (overwrite) the contents of the file.

**Opening Modes in Standard I/O**

| Mode | Meaning of Mode | During Inexistence of file |
| --- | --- | --- |
| R | Open for reading. | If the file does not exist, fopen() returns NULL. |
| Rb | Open for reading in binary mode. | If the file does not exist, fopen() returns NULL. |
| W | Open for writing. | If the file exists, its contents are overwritten. If the file does not exist, it will be created. |
| Wb | Open for writing in binary mode. | If the file exists, its contents are overwritten. If the file does not exist, it will be created. |
| A | Open for append. Data is added to the end of the file. | If the file does not exist, it will be created. |
| Ab | Open for append in binary mode. Data is added to the end of the file. | If the file does not exist, it will be created. |
| r+ | Open for both reading and writing. | If the file does not exist, fopen() returns NULL. |
| rb+ | Open for both reading and writing in binary mode. | If the file does not exist, fopen() returns NULL. |
| w+ | Open for both reading and writing. | If the file exists, its contents are overwritten. If the file does not exist, it will be created. |
| wb+ | Open for both reading and writing in binary mode. | If the file exists, its contents are overwritten. If the file does not exist, it will be created. |
| a+ | Open for both reading and appending. | If the file does not exist, it will be created. |
| ab+ | Open for both reading and appending in binary mode. | If the file does not exist, it will be created. |

## **Closing a File**

The file should be closed after reading/writing.

**syntax for closing:**

fclose(fptr);

Here, fptr is a file pointer associated with the file to be closed.

## **Fgetc function**

Let’s see a simple program of filing.

|  |
| --- |
| # include "stdio.h"  main( )  {  FILE \*fp ;  char ch ;  fp = fopen ( “PR1.C", "r" ) ;  while ( 1 )  {  ch = fgetc ( fp ) ;  if ( ch == EOF )  break ;  printf ( "%c", ch ) ;  }  fclose ( fp ) ;  } |

To read the file’s contents from memory there exists a function called fgetc( ). This has been used in our program as,

ch = fgetc ( fp ) ;

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 .

EOF is macro which signifies the end of file.

There is possibility the file you want to read may not exist or in a file you want to write this create a problem like, disk space may be insufficient to open a new file, or the disk may be write protected or the disk is damaged and so on.So for this we define this in our program,

if ( fp == NULL )

{ puts ( "cannot open file" ) ;

exit( ) ; }

}

## **fputs() function.**

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>    main( )  {  FILE \*fs, \*ft ;  char ch ;  fs = fopen ( "pr1.c", "r" ) ;  if ( fs == NULL )  {  puts ( "Cannot open source file" ) ;  exit(1) ;  }  ft = fopen ( "pr2.c", "w" ) ;  if ( ft == NULL )  {  puts ( "Cannot open target file" ) ;  fclose ( fs ) ;  exit(1) ;  }  while ( 1 )  {  ch = fgetc ( fs ) ;  if ( ch == EOF )  break ;  else  fputc ( ch, ft ) ;  }  fclose ( fs ) ;  fclose ( ft ) ;  } |

## **String (line) I/O in Files**

**Receives strings from keyboard and writes them to file.**

|  |
| --- |
| #include <stdio.h>  #include<string.h>  #include<stdlib.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 ) ;  } |

**Reads strings from the file and displays them on screen**

|  |
| --- |
| **#include "stdio.h"**  **#include<stdlib.h>**  **main( )**  **{**  **FILE \*fp ;**  **char s[80] ;**  **fp = fopen ( "POEM.TXT", "r" ) ;**  **if ( fp == NULL )**  **{**  **puts ( "Cannot open file" ) ;**  **exit(1) ;**  **}**  **while ( fgets ( s, 79, fp ) != NULL )**  **printf ( "%s" , s ) ;**  **fclose ( fp ) ;**  **}** |

## **Reading and writing to a file**

For reading and writing to a text file, we use the functions fprintf() and fscanf().

## **Writing to a text file**

Lets see a example of writing a text into a text file.

**Example 1: Write to a text file**

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  int main()  {  int num;  FILE \*fptr;  // use appropriate location if you are using MacOS or Linux  fptr = fopen("D:\\program.txt","w");  if(fptr == NULL)  {  printf("Error!");  exit(1);  }  printf("Enter num: ");  scanf("%d",&num);  fprintf(fptr,"%d",num);  fclose(fptr);  return 0;  } |

This program takes a number from the user and stores in the file program.txt.

After you compile and run this program, you can see a text file program.txt created in D drive. When you open the file, you can see the integer you entered.

## **Writing to a binary file**

Functions fread() and fwrite() are used for reading from and writing to a file on the disk respectively in case of binary files.To write into a binary file, you need to use the fwrite() function. The functions take four arguments:

* address of data to be written in the disk
* size of data to be written in the disk
* number of such type of data
* pointer to the file where you want to write.

**syntax**

fwrite(addressData, sizeData, numbersData, pointerToFile);

**Example 2: Write to a binary file**

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  struct threeNum  {  int n1, n2, n3;  };  int main()  {  int n;  struct threeNum num;  FILE \*fptr;  if ((fptr = fopen("D:\\program.bin","wb")) == NULL){  printf("Error! opening file");  // Program exits if the file pointer returns NULL.  exit(1);  }  for(n = 1; n < 5; ++n)  {  num.n1 = n;  num.n2 = 5\*n;  num.n3 = 5\*n + 1;  fwrite(&num, sizeof(struct threeNum), 1, fptr);  }  fclose(fptr);    return 0; } |

In this program, we create a new file program.bin in the D drive.

We declare a structure threeNum with three numbers - n1, n2 and n3, and define it in the main function as num.Now, inside the for loop, we store the value into the file using fwrite().

The first parameter takes the address of num and the second parameter takes the size of the structure threeNum. Since we're only inserting one instance of num, the third parameter is 1. And, the last parameter \*fptr points to the file we're storing the data.Finally, we close the file.

**Example 3: Read from a text file**

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  int main()  {  int num;  FILE \*fptr;  if ((fptr = fopen("D:\\program.txt","r")) == NULL){  printf("Error! opening file");  // Program exits if the file pointer returns NULL.  exit(1);  }  fscanf(fptr,"%d", &num);  printf("Value of n=%d", num);  fclose(fptr);    return 0;  } |

This program reads the integer present in the program.txt file and prints it onto the screen.

If you successfully created the file from previous example running this program will get you the integer you entered.

**Example 4: Read from a binary file.**

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  struct threeNum  {  int n1, n2, n3;  };  int main()  {  int n;  struct threeNum num;  FILE \*fptr;  if ((fptr = fopen("C:\\program.bin","rb")) == NULL){  printf("Error! opening file");  // Program exits if the file pointer returns NULL.  exit(1);  }  for(n = 1; n < 5; ++n)  {  fread(&num, sizeof(struct threeNum), 1, fptr);  printf("n1: %d\tn2: %d\tn3: %d", num.n1, num.n2, num.n3);  }  fclose(fptr);    return 0;  } |

In this program, you read the same file program.bin and loop through the records one by one.

In simple terms, you read one threeNum record of threeNum size from the file pointed by \*fptr into the structure num.

You'll get the same records you inserted in Example 3.

**Lab Task**

**Question # 01:**

Create a structure to specify data of customers in a bank. The data to be stored is: Account number, Name, Balance in account. Assume maximum of 20 customers in the bank.

(a) Write a function to print the Account number and name of each customer with balance below

Rs. 100.

(b) If a customer request for withdrawal or deposit, it is given in the form: Acct. no, amount, code (1 for deposit, 0 for withdrawal) Write a program to give a message, “The balance is insufficient for the specified withdrawal”.

**Question # 02:**

There is a structure called employee that holds information like employee code, name, date of joining. Write a program to create an array of the structure and enter some data (atleast 10 employee)into it. Then ask the user to enter current year. Display the names of those employees whose tenure is 3 or more than 3 years according to the given current date.

**Question # 03:**

Suppose a file contains student’s records with each record containing name of a student display information into display.

**Question # 04:**

Write a program that will read a file and count how many characters, spaces, tabs and newlines are present in it.

**Question # 05:**

Write a program that will read a file and shows only vowel character.

**Question # 06:**

Write a program to add the contents of one file at the end of another.

**Question # 07:**

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