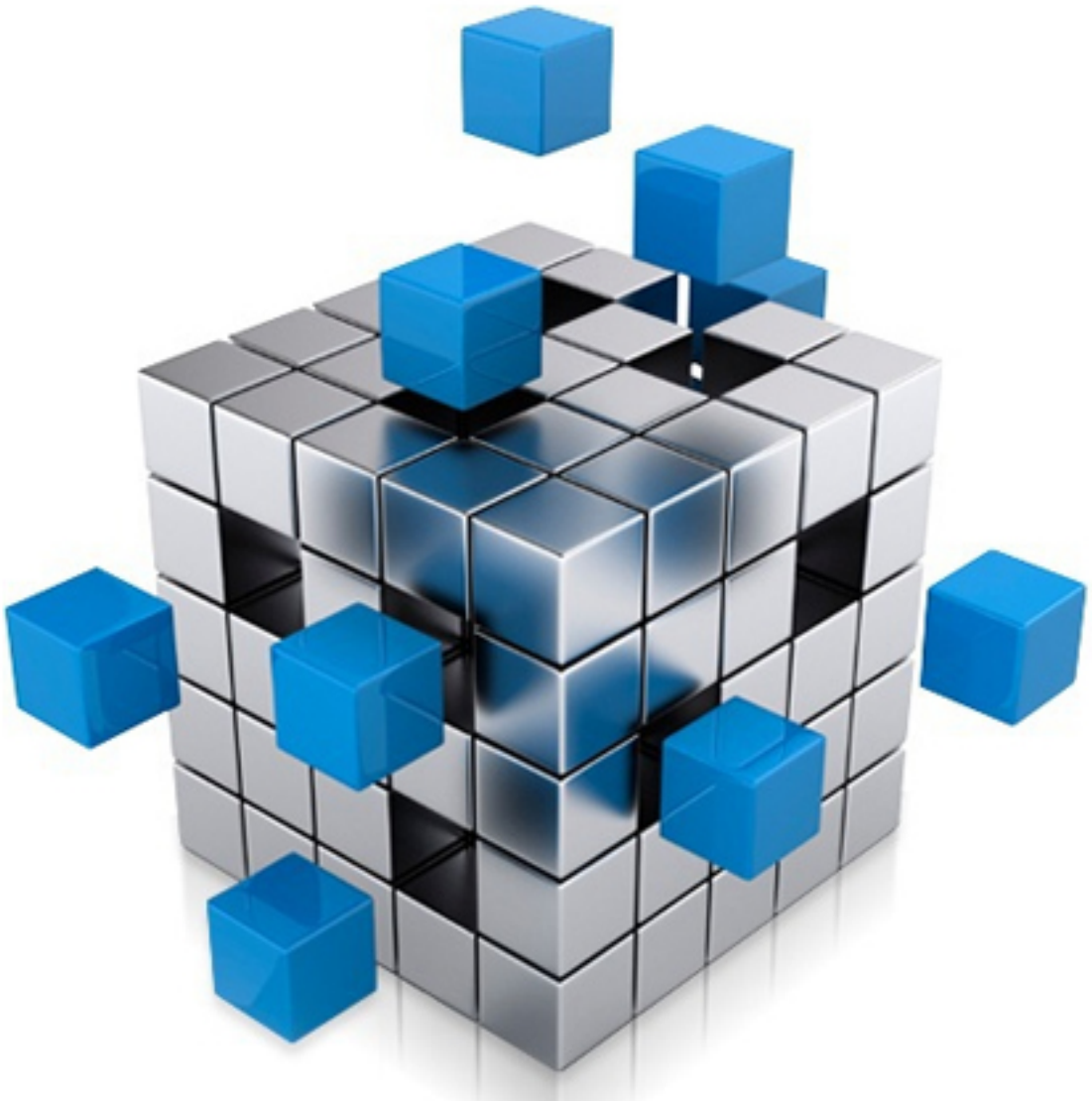
VTU eNotes On

Data Structures Using C



(Computer Science)

Subject: **Data Structure with C**

Topic: **Introduction to Data Structure**

Why Data Structures?

Problem solving is more relatd too understanding the problem, designing a solution and Implementing the solution, then What exactly is a solution?

In a very crisp way, it can be demonstarte as a solution which is equals to a program and it is also approxiamtley equals to algorithm.

Algorithm

An algorithm is a sequence of steps that take us from the input to the output. An algorithm must be Correct. It should provide a correct solution according to the specifications. Finite. It should terminate and general. It should work for every instance of a problem is efficient. It should use few resources (such as time or memory).

Data organization

Any algorithm we come up with will have to manipulate data in some way. The way we choose to organize our data directly affects the efficiency of our algorithm.

Solution = algorithm + data organization, Both components are strongly interconnected.

Information

The study of Computer Science includes study of information. Information in the substratum of entire field. In computer all information stored in form of a collection of bits, it is smallest unit of information, it has only one value

Data Type

It is representation of information using a set of value and set of operations required for deriving further results and consider an example A day's rain is expressed in discrete form as millimeter.

This value can be subjected to a set of operation such as add to derive the total rain in a year, Division to derive average rain in a year. Unstructured or **scalar**: Integer, float, char and Pointer, **homogenous**: Array, string, enum, structure and Union, **heterogeneous** :ADT like list, queue, stack, tree and Graph.

Data structure

It is way of organizing value with help of existing data types, ex: Accumulation of rain data for one year and apply some operation to derive statistical results. Data of 365 days need integer to store 365 value in the list- one dimension and 10 different regions require to store – 2D. It is a aggregation of different type of data by which the stored data can be made more explanatory. Hence, the Data structure is require through knowledge of data types available in a Programming Language.

Data structure can be also defined as, it is the mathematical model which helps to store and retrieve the data efficiently from primary memory. It helps to consistently maintain the data as well as the implem-n functions of interest for data.

Data structure(definition by Prof. S Sahani)

It is data object together with the relationship which exist the relationship among the instance and among the individual elements that compose instance. Relationship provided by specifying the function of interest. When study data structure, we are concern with representation of data object as well as the implementation functions of interest for data object. Representation of each data object should facilitate an efficient implementation of function.

Atomic and composite data

Atomic data types is a single and non decomposable entity. Set of atomic data type having identical properties and consider an example the book price: Rs:250.

Composite data type, which cannot be broken out into subfield that having meaning consider an example Mobile number.

Data structure

It is an aggregation of atomic and composite data types into a set with defined relationships. Structure is set of rules that holds data together. An arrangement of data in a computer's memory. Algorithms manipulate the data in these structures in order to accomplish some task. Consider an example like inserting an item, search for an item, sorting. In other worlds , it is conceptual and concrete ways to organize data for efficient storage and manipulation.

Why do we need data structure ?

Computer takes on more and more complex tasks and its software implementation and maintenance is difficult, also clean conceptual frame work allows more efficient more correct code. Argument against: Packages are already written, Why not just read documentation of their interfaces and use them? The more you know, the better you can choose the tools, You can modify tools, You can create entirely new tools, You are to become experts !

We will learning the following concepts

What are some of the common data structures, What are some ways we can implement them , How can we analyze their efficiency , How can we use them to solve some, practical problems and Known data structures are tools for solving your future problems.

The following ways are possible to used the data structure. As an actual way to store real-world data, let we consider an example queues as a tool to be used only within a program, and graphs as a model of real-world situations.

Importance of different data structure

Each data structure has different advantages and disadvantages, and will be useful for different types of applications and for example of fast access of memory , if we know the

Data in Different volume

The amount of data and instructions are stored in different hierarchical memory is varies. The CPU can hold very small amount of data than cache memory and cache accommodate lesser than main memory, similarly main memory have less space than secondary memory. The data structure can comfortable access the data from main memory, if it is beyond the main memory , but the scope of data structure limited to main memory only. If at all needed to access the data from secondary memory then we need a concept called data mining which is explained in the figure 1.

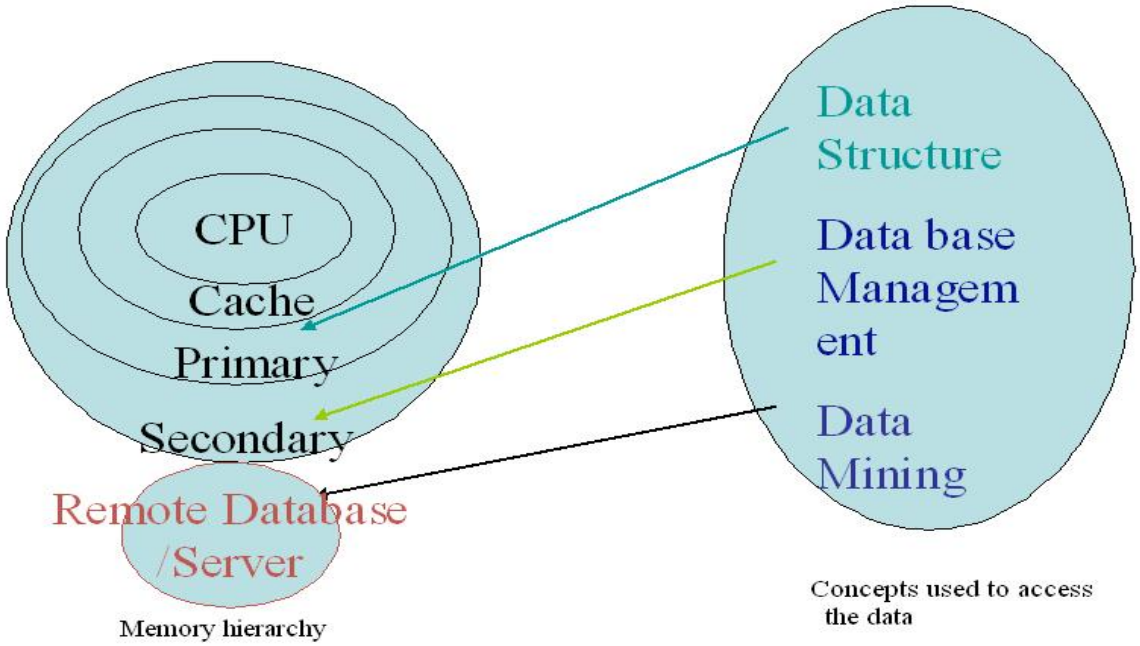


Figure 1: Concept used to access the data

UNIT- 2

Binary Files

Structure:

2.1 Classification Of Files

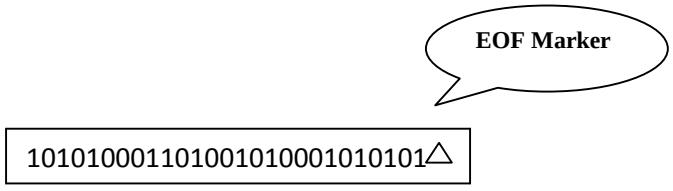
2.2 Files Modes

2.3 Standard Library Functions for Files

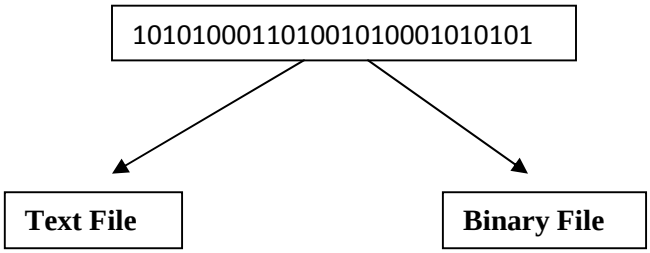
2.4 File Type Conversion

2.1 Classification Of Files

A file is collection of data. A precise definition to files could be “A file is a collection of related data stored in auxiliary storage device”. This definition includes two additionally and essential attribute of file that is data in file being related and storage device. The mere representation of file in machine is in form of 0’s and 1’s as shown



A proper organization is required so that program can interpret the data while reading. Based on this we have two class of files text and binary files.



A text file is file of characters. I cannot contain any other data types like int or float. If present stores them in to character equivalent formats. The character her may either be encoded in ASCII or EBCDIC.

On the other hand, a binary file is collection of data stored in the internal format of the computer. It can store data belonging to different data types like int, float or even structure but not another file. If the data is textual it represented by 1 bytes, numeric in two or more bytes and so on. Another major difference between the text and binary file is that in record storage. A record is collection of related data, it done in C using structure (*struct*). Binary file has logical sequence of records.

The term binary associated may be misleading, However, In computer memory all files have binary representation. Therefore data stored as text file can also be stroed as binary file. This just depends on how we interpret the data.

2.2 Files Modes

Any file has end-of-file (EOF) marker. A file can be opened in read state, write state or in error state. The error state occurs due to illegal operation done either in read or in write modes of file opening.

In read state, i.e., file is in read mode, if write operation is performed leads to error state. Similarly, In write state, if read operation is performed it leads to error state. Apart from this, we have append mode where data is written at the end of file, so data is appended. Speaking with respect to our binary file, the modes are *rb*, *wb* and *ab*.

Apart from these modes discussed we also have update mode, where we can open the file to perform any operation. There are three update modes *r+b*, *w+b*, *a+b* . In the r+b mode file is initially meant for reading and updating and later we can move to write state by positioning the file. Similarly the other modes.

Files in C can be opened using *fopen*() function and prototype is given

FILE *fopen(const char *filename, const char *mode);

The first parameter is the file location or path in which file is located and second parameter is the file modes. Let us summaries the file modes along with there meaning in tabular form.

Modes	**Meaning**
rb	Open the file for reading, Read start from beginning of the file. If the file doesn't exist then error is returned
wb	Open the file for writing, Write start from beginning of the file. If the file doesn't exist then it is created
ab	Open the file for append, New records are appended at the end If the file doesn't exist then it is created
r+b	Open file for both read and write. Initial read begins at start of the file. If the file doesn't exist then error is returned
w+b	Open file for both read and write, Initial write begins at start of the file, If the file doesn't exist then it is created
a+b	Open file for both read and write, Write starts at the end, If the file doesn't exist then it is created

2.3 Standard Library Functions for Files

Standard library function is the built-in function provided by C for file handling. We discuss this function with respect to our binary file and set of function can be categorized as

- File open/close
- Character Input/Output
- Formatted Input/Output
- Line Input/Output
- Block Input/Output
- File positioning
- File status
- System file operations

Opening and closing a file are basic operation performed on file fro which we have *fopen()* and *fclose()* function in library. The prototype and it usage in given below

FILE *fopen(const char *filename, const char *mode);

int fclose(FILE *fp);

Usage: fp = fopen(“abc.bin”,”rb”);
fclose(fp);

C make use of block Input/Outputfuction to read an write in to binary files. This is done by *fread()* and *fwrite()* function from the library. The prototype and usage is given below.

int fread(void *inarea, int item size, int count,FILE *fp);

int fwrite(void *inarea, int item size, int count,FILE *fp);

The first parameter is generic pointer to in area or input area from the memory which is usally a structure. The next two parameters are size of each item and number of items. Lastly, the file pointer.

Usage: amt_ read = fread(&e,sizeof(e),1,fp);

Where 'e' is some structure.

C provides three function to determine the file status-test end of file *feof(),* test-error *ferror()* and clear error *clearerr()*.

feof() is used to check if the end of file has been reached, if so it returns a true else return false. The prototype of the function is

int feof(FILE *fp)

Test error is used to check the error status of the file occurred due to various reasons. If error has occurred it return true else false. The prototype is

int ferror(FILE *fp)

The error status can be reset by using clearer() function and prototype is

void clearerr(FILE *fp)

The randomness of a file can be seen through the file positioning techniques, this can be don in C using *ftell()* –tell current pointer location, fseek()- moves the pointer to specified position, rewind() – moves the file pointer to the beginning of the file. The prototype for functions is given below.

long int ftell(FILE *stream);

void rewind(FILE *stream);

int fseek(FILE *stream, long offset, int wherefrom);

The *offset* in the fseek() function specified the number of bytes to be moved. The *wherefrom* indicates in which direction. There are three named constants provided by C they are:

SEEK_SET	0
SEEK_CUR	1
SEEK_END	2

The SEEK_CUR start the displacement from the current position to number of bytes specified. SEEK_END is from the end of the file

Apart from these library file C also provides functions to remove(),rename() and tmpfile(). The prototype is given below.

int remove(FILE *stream);

It returns 0 if successfully deleted else returns a non-zero value.

int rename(const char *oldfilename,
const char *newfilename);

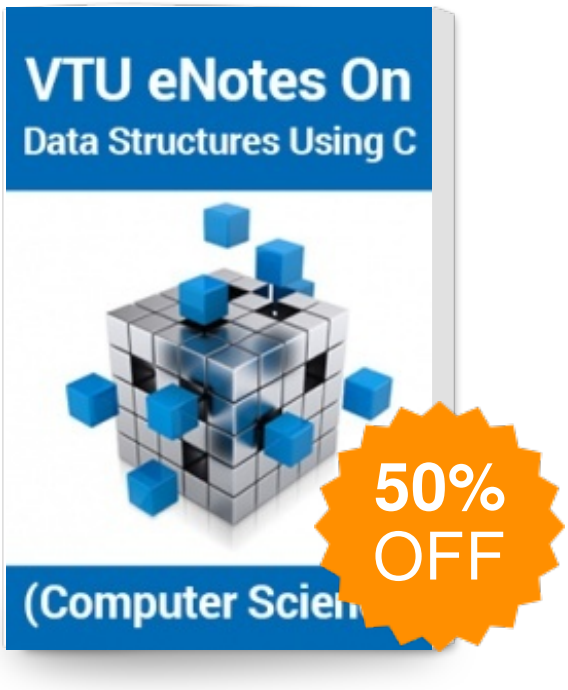
If the renaming is successful returns zero else non-zero.

FILE * tmpfile(void);

tmpfile() is used to create temporary output file. This can be done like this.

```
FILE *fp;
fp = tmpfile();
```

VTU eNotes On Data Structures Using C (Computer Science)



Publisher : VTU eLearning

Author : Panel Of Experts