FILE HANDLING IN C++

FILE CONCEPTS

- **Every program or sub-program consists of two major components:**
  - **algorithm and**
  - **data structures.**

- The algorithm takes care of the rules and procedures required for solving the problem and the data structures contain the data.
- The data is manipulated by the procedures for achieving the goals of the program as shown in Fig.

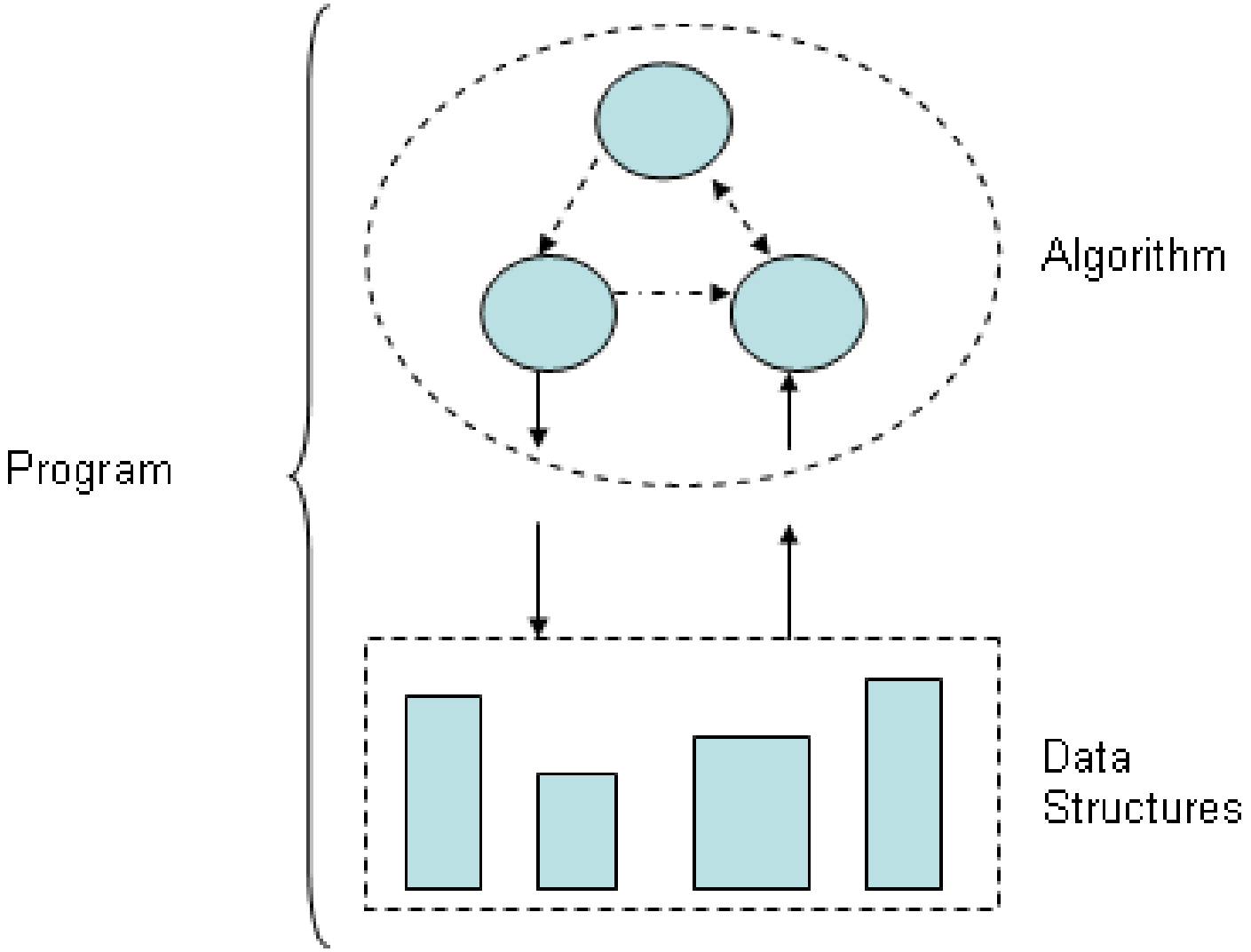

Fig. The structure of a program or sub-program

A data structure is volatile by nature in the sense that its contents are lost as soon as the execution of the program is over.

- Similarly, an object also loses its states after the program is over.

  If we want to permanently store our data or want to create persistent objects then it becomes necessary to store the same in a special data structure called file.

- The file can be stored on a second storage media such as hard disk. In fact, vary large data is always stored in a file.

File

"A file is a logical collection of records where each record consists of a number of items known as fields".

The records in a file can be arranged in the following three ways:

- **Ascending/Descending order**: The records in the file can be arranged according to ascending or descending order of a key field..
- **Alphabetical order:** If the key field is of alphabetic type then the records are arranged in alphabetical order.
- **Chronological order:** In this type of order, the records are stored in the order of their occurrence i.e. arranged according to dates or events. If the key-field is a date, i.e., date of birth, date of joining, etc. then this type of arrangement is used.

FILES AND STREAMS

In C++, a stream is a data flow from a source to a sink. The sources and sinks can be any of the input/output devices or files.

For input and output, there are two different streams called input stream and output stream.

Stream	Description
cin	standard input stream
cout	standard output stream
cerr	standard error stream

The standard source and sink are keyboard and monitor screen respectively

ifstream: It is the input file stream class. Its member function open( ) associates the stream with a specified file in an input mode.

In addition to **open(),** ifstream class inherits the following functions from istream class.

(i) get( ) (ii) getline( ) (iii) read( ) (iv) seekg( ) (iv) tellg( )

ofstream : It is the output file stream class. Its member function open( ) associates the stream with a specified file in an output mode.

In addition to **open(),** ofstream inherits the following functions from ostream class

(i) put( ) (ii) write( ) (iii) seekp( ), (iv) tellp( )

fstream : It supports files for simultaneous input and output. It is derived from ifstream, ofstream and iostream classes.

The functions associated with this stream are :

1. open : This associates the stream with a specified file.
2. close : It closes the stream.
3. close all : It closes all the opened streams
4. seekg : Sets current `get' position in a stream
5. seekp : Sets current `put' position in a stream
6. tellg : Returns the current `get' position in a stream
7. tellp : Returns the current `put' position in a stream.

OPENING AND CLOSING A FILE (Text Files)

A file can be opened in C++ by two methods:

1. **By using the constructor of the stream class to be used**
2. **By using the open( ) function of the stream class to be used**

For reading entire lines of text :

C++ provides **get( )** and **getline( )** functions as input member functions of the ifstream class.

It also provides a **put( )** function as output member function of the ofstream class.

OPENING THE FILES BY USING FUNCTION OPEN( )

```
ofstream newfile;              ...(i)
newfile.open ("test.dat");     ...(ii)
```

In the statement

(i) declares the stream newfile to be of type ofstream i.e. output stream. The statement .

(ii) assigns the file stream to the file called "test.dat". Thus, in the program the file "test.dat" would be known as newfile.

The major advantage of this method of opening a file is that more than one files can be opened at a time in a program.

READING AND WRITING BLOCKS AND OBJECTS(BINARY FILES)

The major advantage of binary files is that they require less memory space for storage of data. Moreover, these files can be used to read or write structured data such as structures, class objects etc.

STORING OBJECTS IN FILES

If the information contained in the object is very important then we must try to save it on auxiliary storage such as hard disk so that it can be reused as and when required.

Normally the contents of an object are lost as soon as the object goes out of scope or the program execution is over.

In fact, similar to records, objects can also be written into and instantiated from a file.

The objects which remember their data and information are called as persistent objects.

DETECTING END OF FILE

While reading a file, a situation can arise when we do not know the number of objects to be read from the file i.e. we do not know where the file is going to end?

A simple method of detecting end of file (eof) is by testing the stream in a while loop as shown below:

```
while (<stream>)
                    {
                     :
                    }
```

The condition <stream> will evaluate to 1 as long as the end of file is not reached and it will return 0 as soon as end of file is detected.

SUMMARY

- **Any thing stored on a permanent storage is called a file.**
- **A set of related data items is known as a record. The smallest unit of a record is called a field.**
- **A key field is used to uniquely identify a record.**
- **A file is a logical collection of records.**
- **In a serial file, the records are stored in the order of their arrival without regards to the key field.**
- **On the other hand, in a sequential file, the records are written in a particular order of the key field.**
- **The key field is also known as a primary key. 'ifstream' and 'ofstream' are input and output streams respectively.**
- **The objects that remember their data and information are called persistent objects.**
- **The function eof() returns 0 when it detects the end of file. A opened file must be closed after its usage.**