CHAPTER 9

FILE HANDLING

After completing this unit, you will be able to:

- know the binary and text file
- open the file in different modes
- know the concept of
- define stream
- use the following stream

1.1 INTRODUCTION

Q1. Give a brief introduction of file and file handling.

Ans.

A file is a collection of bytes stored on a secondary storage device, like hard disk. A file is simply a machine decipherable storage media where programs and data are stored for machine usage.

Explanation

In a file, the collection of byres may be interpreted. For example as characters, words, lines, paragraphs and pages from a textual document. Fields and records belonging to a database, or pixels from a graphical image. The meaning attached to a particular file is determined entirely by the data structures and operations used by a program to process the file.

File handling

File handling concept in C++ language is used for storing data permanently in computer. It provides a mechanism to write output of a program into a file and read from a file.

Operations involved in file handling

The basic operation involved in file handling are

1. **Opening file**
2. **Reading and writing file**
3. **Closing file**

1.1.1 TYPES OF FILES

Q2. Explain the types of files in C++

Ans

Types of files in C++

C++ divides files into two different types based on how they store data. These are

1. **Text files**
2. **Binary files**

1. **Text files**

   Text files can be a stream of characters that a computer can process sequent. It is not only processes sequentially but only in forward direction.

   Processing on characters

   Text files only process characters. They can only read or write data one character at a time. A text stream in C++ is a special kind of life. Depending on the requirements of the operating system, newline characters may be converted to or from carriage-return combinations depending on whether data is being written to or read from the file.

   Character conversion

Other character conversions may also occur to satisfy the storage requirements of the operating system. These translations occur transparently and they occur because the programmer has signaled the intention to process a text file

Functions for text

In C++ programming language, functions are provided that deal with lines of text but these still essentially process data one character at a time.

Operations on text file

A text file is usually opened for only one kind of operation (reading, writing or appending) at any given time

2. **Binary files**

Binary files is a collection of bytes

Difference between byte and character

In C programming language, a byte and a character are equivalent. Hence, a binary file is also referred to as a character stream but there are two essential differences.

1. No special processing of the data occurs and each byte of data is transferred to or from the disk unprocessed
2. C programming language places no constructs on the file and it may be read from, or written to in any manner chosen by the programmer

Processing of binary files

Binary files can be either processed sequentially or depending on the needs of the application they can processed using random access techniques.

Techniques of processing on files

In C++ programming language, processing a file using random access techniques involve moving the current file position to an appropriate place in the file before reading or writing data. This indicates a second characteristics of binary files.

They are generally processed using read and writes operations simultaneously.

Example

A database file will be created and processed as a binary file. A record update operation will involve locating the appropriate record. Reading the record into memory, modifying it in some way and finally writing the record back to disk at its appropriate location in the file.

These kinds of operations are common to many binary files but are rarely found in applications that process text files.

1.1.2 OPENING FILE

Q3. Explain the concept of opening of a file

Ans.

Syntax to open a file

To open a file the function open ( ) is used whose syntax is

MyFile.open(filename);

Here myFile is an internal variable, actually an object used to handle the file whose name is written in the parenthesis. To declare this variable the following statement is used

Ifstream myFile ;

The **dot (.)** operator is used between the variable **myFile** and open function **myFile** is an object of **ifstream** while **open ( )** is the function of **ifstream.** The argument for open function is the name of the file on the disk which should be enclosed in double quotes. The file name can be simple file name like **"sale.txt"**. it can be fully qualified path name like **"C:\myprogs\sale.txt"**

Q4. Describe the modes of opening file. Explain it with the help of program

Ans.

Modes of opening a file

While opening a file, we tell the computer what we want to do with it i.e we want to read the file or wrote into the file or want to modify it. In order to open a file in any desired mode the member function **open ( )** should take mode as an argument along with the file name

Syntax

Its general syntax is shown below

Open (filename, mode);

Here **filename** representing the name of the file to be opened and **mode** is an optional parameter with a combination of the following tags

Mode	Description
ios::in	*Open for input operations (Reading a file)*
ios::out	*Open for output operations (Writing a file)*
ios::binary	*Open in binary mode*
ios::ate	*Open a file for output and move the read/write control to the end of the file*
ios::app	*Append mode. All output to that file to be appended to the end*
ios::trunc	*If the file opened for output operations, its existing contents are deleted and replaced by the new one*

All these flags can be combined using the bitwise operator OR |

Example

If we want to open the file **test.bin** in binary mode to add data we could do it by the following call to member function **open ( )**

Ofstream myfile;

Myfile.open ("test.bin", ios: :out | ios : :app|ios: :binary);

Open ( ) member functions

Each one of the **open ( )** member functions of the classes **ofstream, ifstream** and **fstream** has a default mode that is used if the file is opened without a second argument

Class	Default Mode Parameter
ofstream	ios::out
ifstream	ios::in
fstream	ios::in | ios::out

For **istream** and **ofstream** classes, **ios::in** and **ios::out** are automatically and respectively assumed, even if a mode that does not include them is passed as second argument to the **open ( )** member function

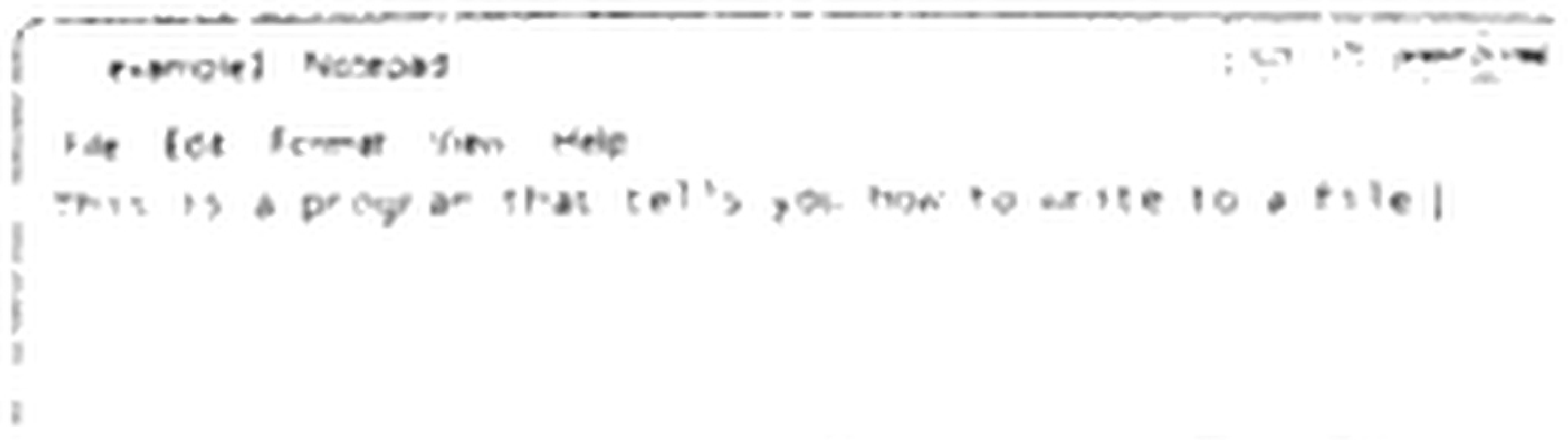
Program

Let us see an example program that creates a text file "**example1.txt"** and writes a line of text in it.

```
//opening a file and writing into it

#include <iostream.h>

#include <fstream.h>

#include <conio.h>

Int main ( )

{

Ofstream mylife;

Myfile.open ("example1.txt");

Myfile<< "This is a program that tells you how to write a file.\n";

Myfile.close( );

Getch();

Return 0;
}
```

Explanation

This code creates a file called **example1.txt** and inserts a sentence into it in the same way as we do with cout, but using the file stream myfile instead. Fig 9.1 shows the output of the above program in the form of a text file that is created in the same working directory.



```
This is a program that tell's you how to write to a file
```

Figure 9.1: Opening and Writing into a Text File

Q5. Explain the method of read input file in C++ programming language

Ans.

Reading input file

To read a word from the file we can write as

```
Myfile>>c;
```

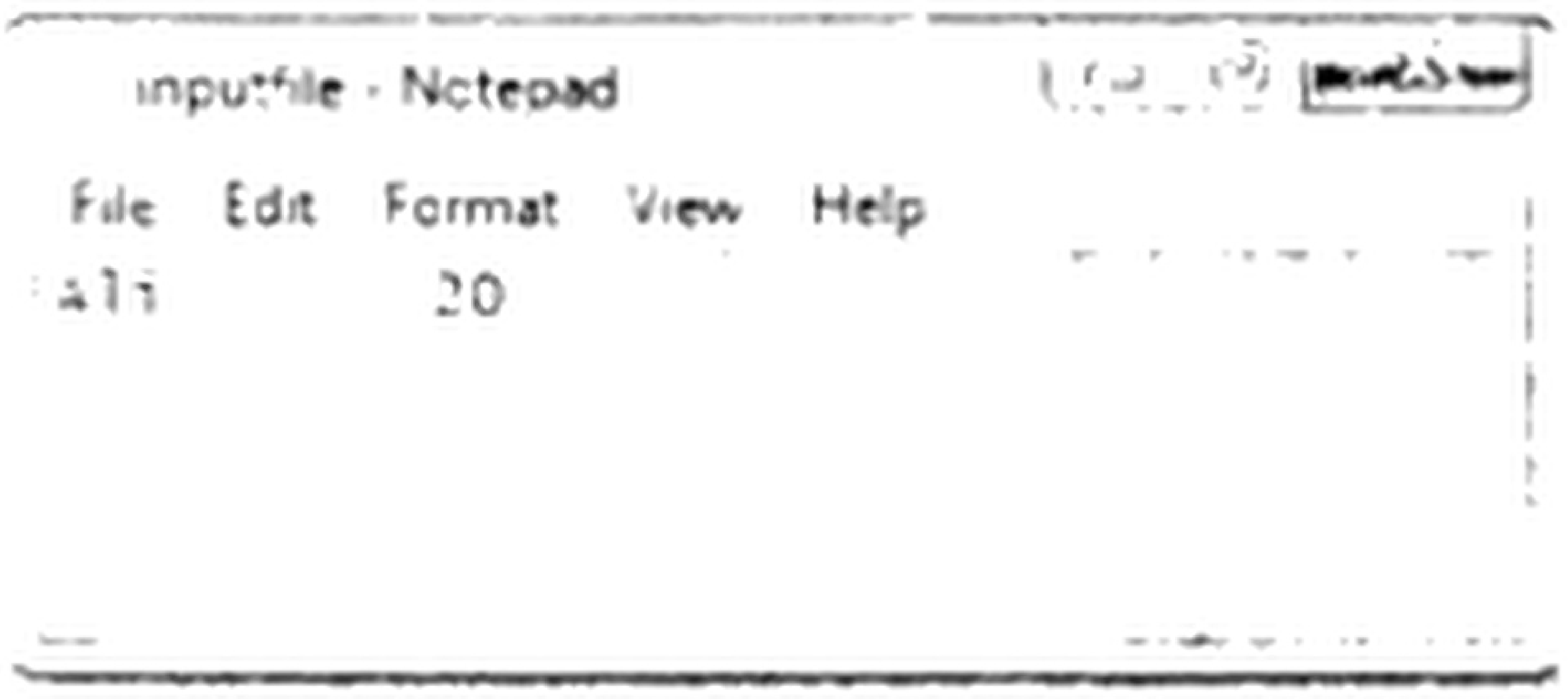
So the first word of the will be read in c, where **c** is a character array. It is similar as we used with cin. There are certain limitation to this. It can read just one word at one time. It means, on encountering a space it will stop reading further. Therefore, we have to use it repeadetly to read the complete file. We can also read multiple words at a time as

```
myFile>>c1>>c2>>c3;
```

the first word will be read in c1 second in c2 and third in c3. Before reading the file, we should know some information regarding the structure of the file. If we have a file of an employee, we should know that the first word is employee's name, second name is salary etc. so that we can read the first word in a string and second word in an int variable.

Program 1

Consider the following input file **inputfile.txt** shown in fig 9.2 which is read and displayed on the screen.



```
Ali         20
```

Figure 9.2: Input File to be Read

```
// Reading input file Program
#include <iostream.h>
#include <fstream.h>
#include <conio.h>
int main ()
{
ifstream myfile("c:\\inputfile.txt");
char ch [20];
int m;
myfile>>ch>>m;
cout<<ch<<"/t"<<m;
myFile.close(),
getch();
return 0,
}
```

Output of the program

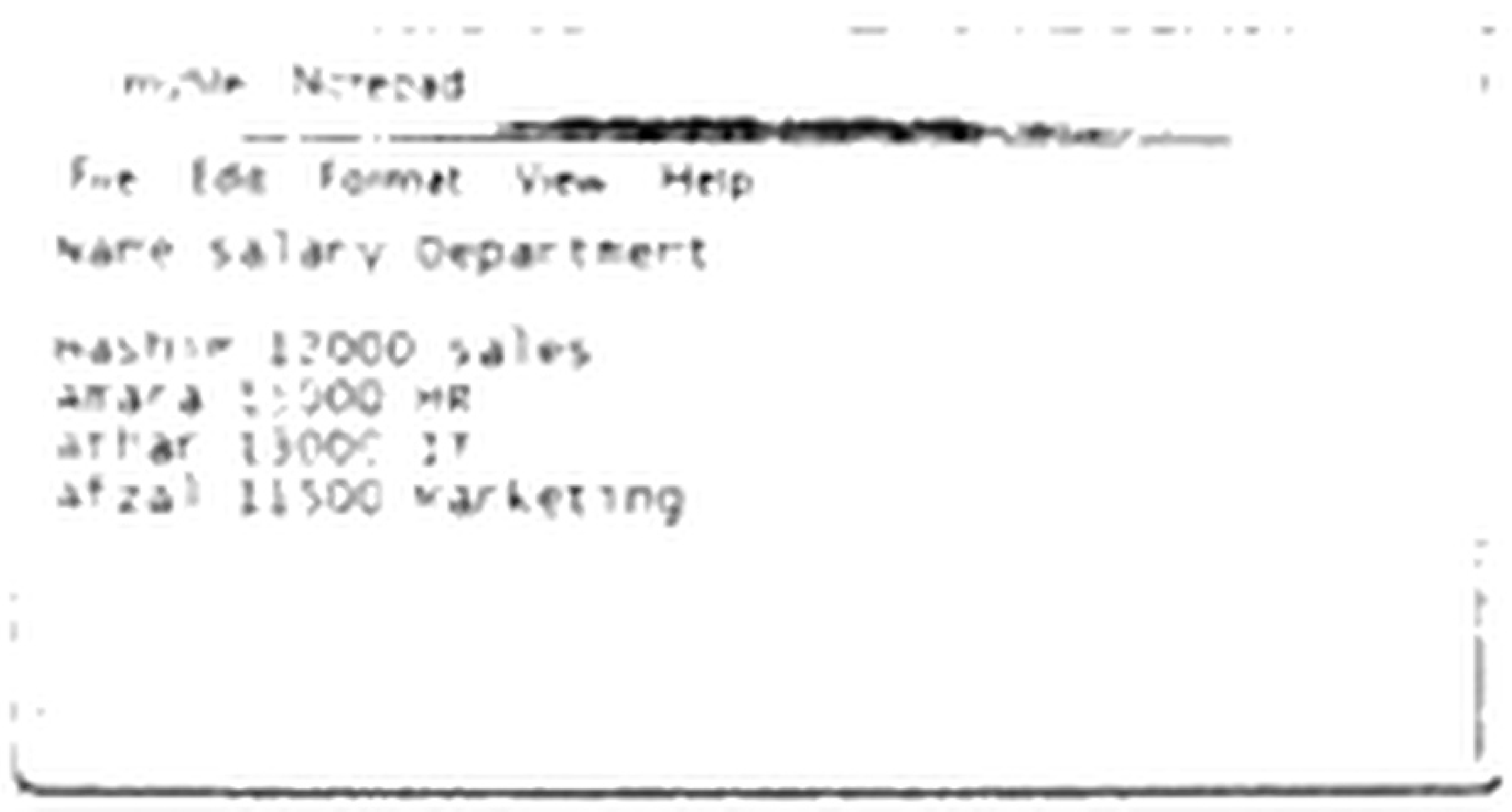
The output of the above program is shown in fig 9.3

```
Ali          20
```

Figure 9.3: Output of Reading Program

Program 2

Let us write another simple program to read from a file **myfile.txt** that is in the current directory and print it on the screen. **Myfile.txt** contains employee's name, salary and department in which they are employees. Fig 9.4 shows **myfile.txt** which is followed by the complete program to describe how it is opened and read and closed

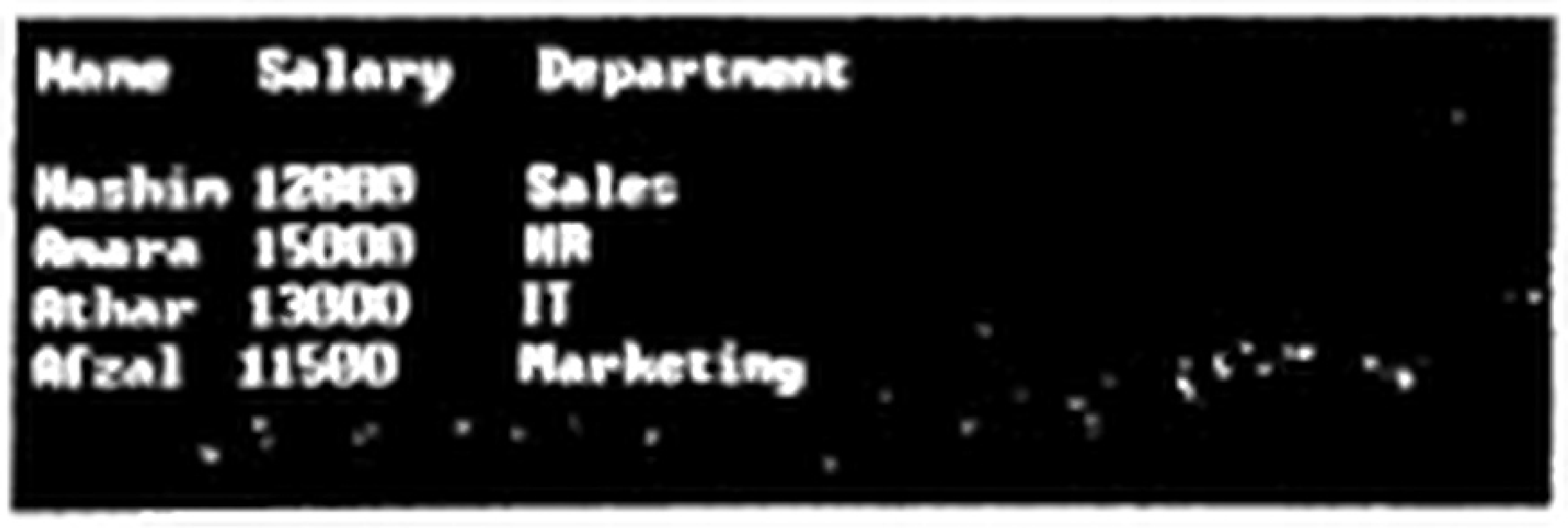


```
Name salary Department

hashim 12000 sales
amara 11000 HR
athar 13000 IT
afzal 11500 Marketing
```

Figure 9.4: Input File to be Read

```
// This program reads from the text file "myfile.txt" which contains the employee
       information
#include <iostream.h>
#include <fstream.h>
#include <conio.h>
main()
{
char name[50];       // used to read name of employee from file
char sal[10],        // used to read salary of employee from file
char dept[30];       // used to read dept of employee from file
ifstream inFile;     // Handle for the input file
char inputFileName[] = "myfile.txt";
// file name, this file is in the current directory
inFile.open(inputFileName);             // Opening the file
// checking that file is successfully opened or not
if ('inFile)
{
cout<<"Can't open input file named"<<inputfileName<<endl;
exit(1);
{
// Reading the complete file word by word and printing on screen
while (!inFile.eof())
{
inFile>>name>>sal>>dept;
cout<<name<<"\t"<<sal<<" \t"<<dept<<endl;
}
inFile.close();
getch();
return 0,
}
```

Output of the program



```
Name    Salary  Department

Hashim 12000    Sales
Amara  15000    HR
Athar  13000    IT
Afzal  11500    Marketing
```

Figure 9.5: Output of the Program that is Read from Input File "myFile.txt"

Q6. Explain the method to close file

Ans

Closing file

Once we have read the file it must be closed. It is the responsibility of the programmer to close the file. We can close the file by using the following statement

myFile.close ( ):

the function **close ( )** does not require any argument, as we are going to close the file associated with **myFile.** Once we close the file no file will be associated with **myFile**.

Q7. Describe the method to open files in binary mode

Ans

Opening files in binary mode

To open a file in a binary mode we need to set the file mode to **ios::binary**

Explanation

Suppose we have a binary file named as **test.dat** to open this file in binary mode we write the statement as

```
Ofstream myFile:
myFile.open("test.dat", ios::binary):
```

in order to write the data to a binary file, **write** method is used. This method is a member function of **ofstream** and **fstream** class

9.1.3 bof ( ) and eof ( )

Q8. Which functions are used in C++ to set the pointers in a file?

Ans.

C++ provides special functions **bof ( )** and **eof ( )** that are used to set the pointers to the beginning of a file and end of a file respectively

Bof ( ) – beginning-of-file

The **bof ( )** is a pointer which returns true if the current position of the pointer is at the beginning of the input file stream and false otherwise. It means that it tells the compiler whether the cursor is at the beginning of file or not.

Eof ( ) – end-of-file

The **eof ( )** is a pointer which returns true when there are no more data to be read from an input file stream and false otherwise. It means that this function checks whether control has reached to the end of file or not. This function is very useful in the case when we do not know the exact number of records in a file

Rules for using eof ( )

1. Always test for the end-of-file condition before processing data
2. Use a while loop for getting data from an input file stream

Program

```
//reading a text file using eof ( ) function

#include <iostream.h>

#include <conio.h>
```

```
#include <fstream.h>

Int main ( )

{

Char ch [50] ;

Ifstream flag ("c:\\TestRecord.txt");

Cout < <"Output is"< < end1;

While(!flag.eof( ) )

{

Flag>>ch;

Cout<<ch<<end1;

}

Flag.close ( );

Getch( );

Return0;

}
```

Output of the program

```
Output is

Ali 20

Anwar 15

Zainab 17

Zonash 21
```

Explanation

In the above program, the input file **TestRecord** is loaded and the **eof ( )** function detects the end of the file. The program reads the file record by record into the string variable ch. Which is then displayed on the screen

1.1.3 STREAM

Q9. Explain the concept of stream in C++

Ans

Stream

In C++ a stream is a sequence of bytes associated with a file. Most of the times streams are used to assure a good and secure flow of data between an application and file

Types of stream

1. Input stream
2. Output stream

1. **Input stream**

   Input streams take any sequence of bytes from an input devices such as keyboard, a file or a network

2. **Output stream**

   Output streams are used to hold output for a particular data consumer such as monitor, a file, or a printer

Reading and writing in files

If the file is only used for reading purpose then the header file **ifstream** is needed to be included. Similarly, if one wants to write in some file then the header file **ofstream** is needed. One can read, write and manipulate the same file by using the header file **fstream**

1.1.4 TYPES OF STREAMS

Q10. Explain the types of stream with the help of programs

Ans

The data can be read from and written to files with the help of following streams

1. Single character stream
2. String stream

1. **Single character stream**

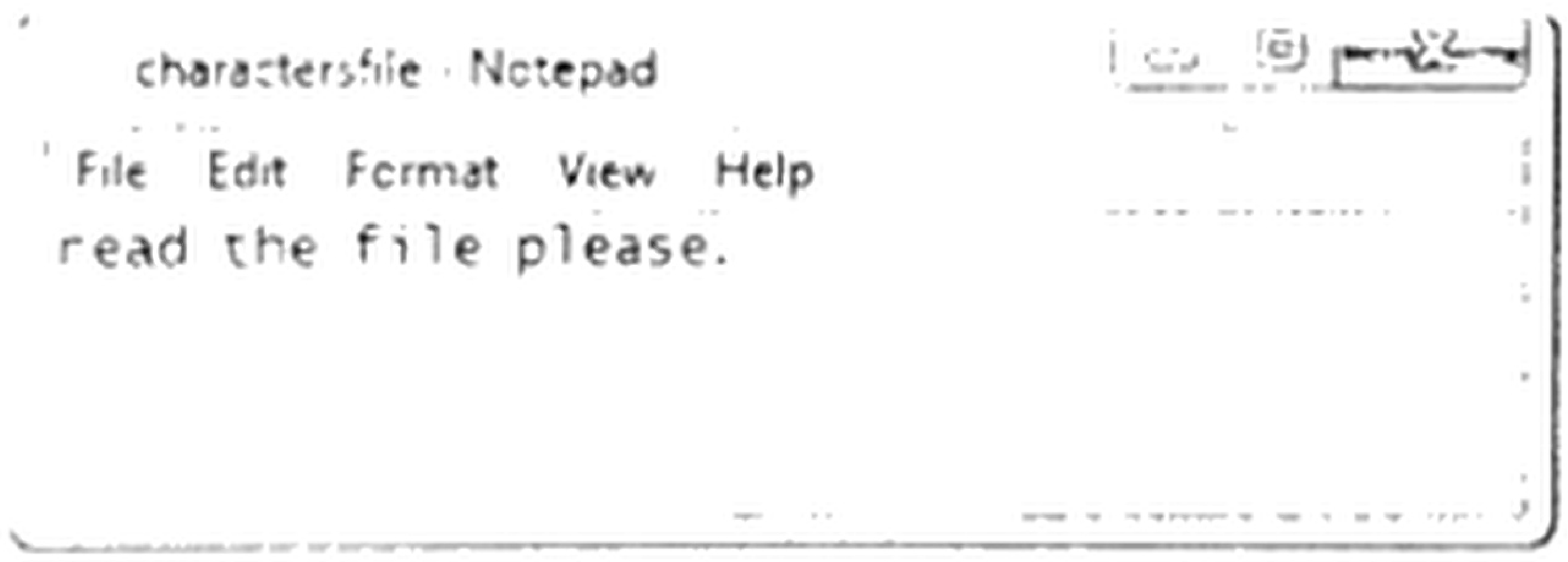
   Using single character stream the data can be read from and written to files character by character

   Reading files character by character

   The function get( ) is used to read data character by character from files

   Explanation

   Consider the following program that reads the data one character at a time from the file charactersfile.txt shown in fig 9.7 and display them on the screen



```
read the file please.
```

Figure 9.7: Input File to be Read Character by Character

Program

```
//character stream (read) program
#include <conio.h>
#include <iostream.h>
#include <fstream.h>
int main()
{
char ch,
ifstream reads("c.\\charactersfile.txt");
while(!reads.eof())
{
read.get(ch),
cout<<ch<<endl;
}
reads.close();
getch(),
return 0,
}
```

Output of the program

……

i. Writing files character by character

Similarly using single character stream data can be written to files character by character by using the function put( )

Program

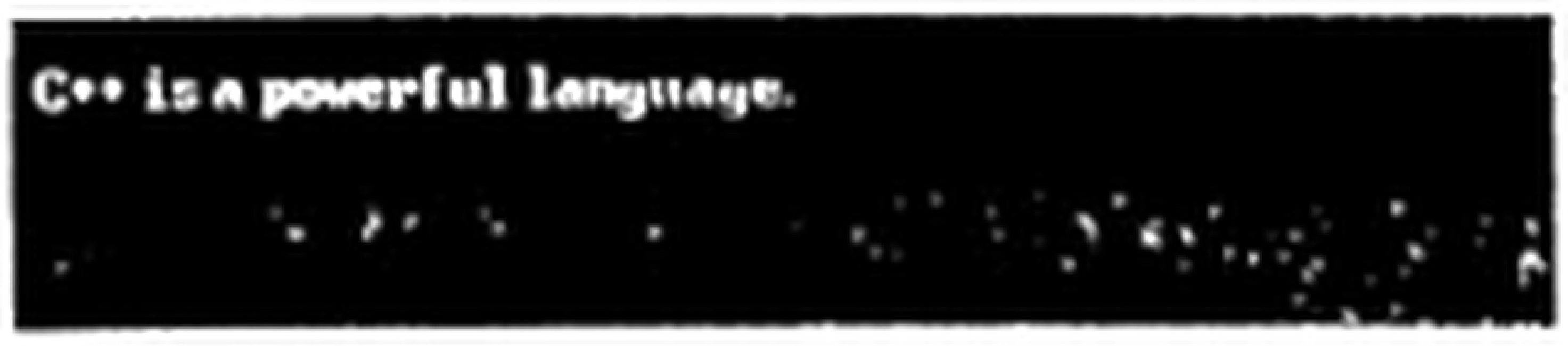
Consider the following program that writes data character by character to an empty files characterswrite.txt

```
//character stream (write) program
#include <conio.h>
#include <iostream.h>
#include <fstream.h>
Int main ( )
```

```
{
Char ch;
Ofstream writes ("d:\\characterswrite.txt");
For (int i=0; i<20; ++i)
{
Cin>>ch;
Cout < < writes.put (ch);
}
Writes.close ( )
Getch( );
Return 0;
}
```

Explanation

When the above program is executed characters are entered one by one from the keyboard. The characters entered above are written to the **characterswrite** file stored at secondary storage at location **D-drive** of the hard disk. The output file generated is shown in fig 9.8



```
C++ is a powerful language.
```

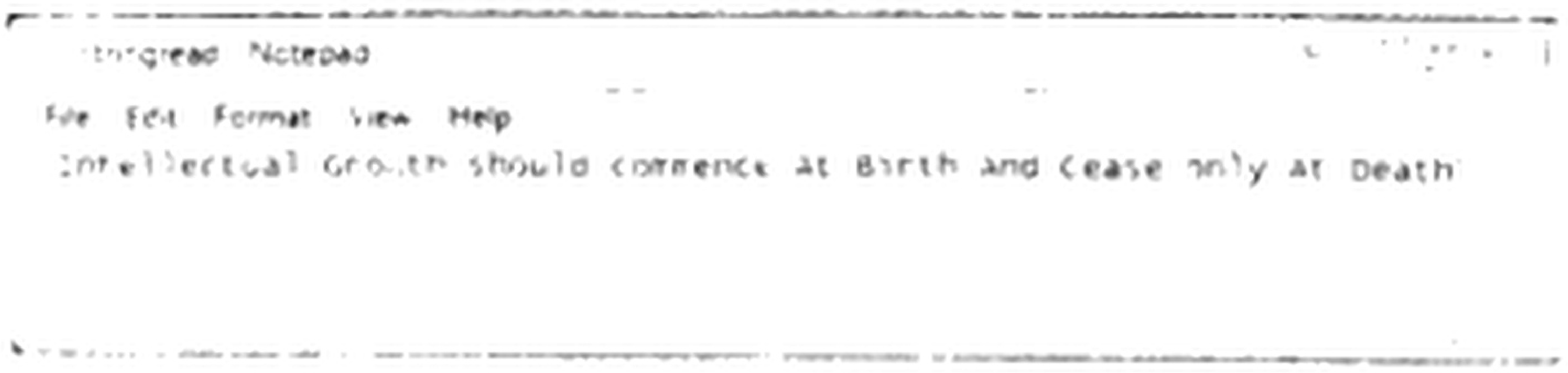
Figure 9.8: File Written by Character Stream

2. **String stream**

   A string stream is a stream which reads input from or writes output to an associated string

   Example

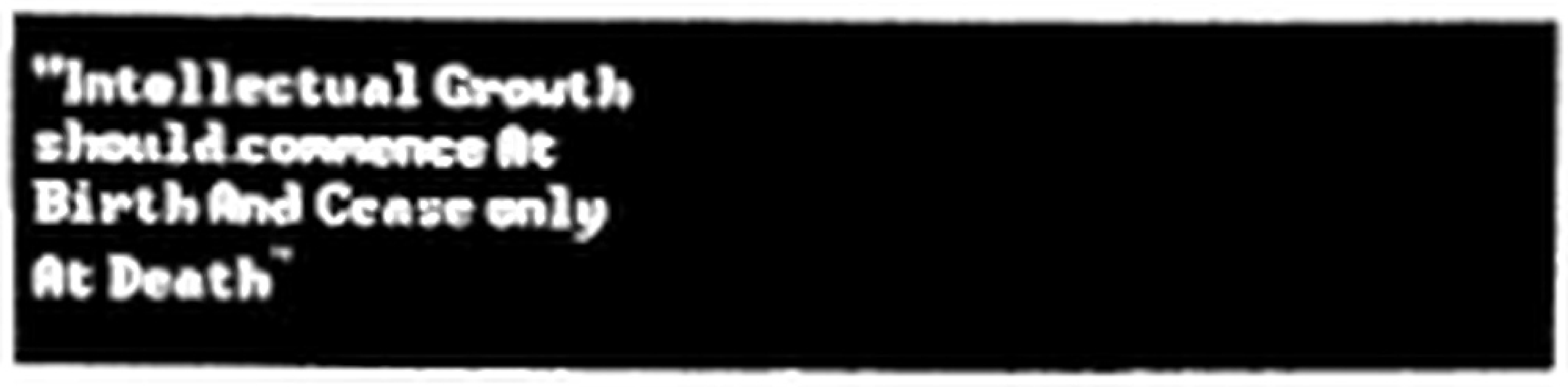
   Consider the text file stringread shown in fig 9.9



```
Intellectual Growth should commence At Birth And Cease only At Death
```

Figure 9.9: Input File that will be Read using String Stream

The following program reads the file stringread.txt into string str using getline ( ) function and display the result on the screen as shown in fig 9.10



```
"Intellectual Growth
should commence At
Birth And Cease only
At Death"
```

Figure 9.10: File Read by String Stream

```
//string stream (read) program
#include <conio.h>
#include <iostream.h>
#include <fstream.h>
#include <string>
int main()
{
char str[20];
ifstream reads("d:\\stringread.txt");
while(!reads.eof())
{
reads.getline(str,21);
cout<<str<<endl;
}
read.close();
getch();
return 0;
}
```

Key Points

- The combination of characters words sentences and paragraphs is called files
- The process of opening reading from and writing into files is termed as handling files
- There are two types of files text files and binary files
- Those files that store data in text format and are readable by human are called text file

- Binary files are those files that store data in binary format which is not readable by the human being but readable by the computers. Binary file are directly processed by the computers
- to open a file the function open ( ) is used
- there are different modes of opening files i.e binary mode, input mode and output mode etc
- the bof( ) is a pointer which true if the current position of the pointer is at the beginning the input file stream and false otherwise
- the eof ( ) is a pointer which returns true when there are no more data to be read, from an input file stream and false otherwise
- a stream can be thought of as a sequence of bytes of varying length is used as a buffer to hold data that is waiting to be processed